View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Padua@research

UNIVERSITÀ DEGLI STUDI DI PADOVA

Sede Amministrativa: Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

SCUOLA DI DOTTORATO DI RICERCA IN : Ingegneria dell'Informazione

INDIRIZZO: Scienza e tecnologia dell'informazione

CICLO: XXVI

QUALITY VALUE BASED MODELS AND METHODS FOR SEQUENCING DATA

Direttore della Scuola: Ch.mo Prof. Matteo Bertocco

Coordinatore d'indirizzo: Ch.mo Prof. Carlo Ferrari

Supervisore: Ch.mo Prof. Gianfranco Bilardi

Dottorando: Michele Schimd

Sommario

Isolata per la prima volta da Friedrich Miescher nel 1869 ed identificata nel 1953 da James Watson e Francis Crick, la molecola del DNA (*acido desossiribonucleico*) umano ha richiesto più di 50 anni perché fosse a disposizione della comunità internazionale per studi e analisi approfondite.

Le prime tecnologie di sequenziamento sono apparse attorno alla metà degli anni 70, tra queste quella di maggiore successo è stata la tecnologia denominata *Sanger* rimasta poi lo standard di fatto per il sequenziamento fino a che, agli inizi degli anni 2000, sequenziatori battezzati di *nuova generazione* (*Next Generation Sequencing (NGS)*) sono comparsi sul mercato. Questi ultimi hanno velocemente preso piede grazie ai bassi costi di sequenziamento soprattutto se confrontati con le precedenti macchine Sanger. Oggi tuttavia, nuove tecnologie (ad esempio *PacBio* di *Pacific Biosciences*) si stanno facendo strada grazie alla loro capacità di produrre frammenti di lunghezze mai ottenute prima d'ora. Nonostante la continua evoluzione nessuna di queste tecnologie è ancora in grado di produrre letture complete del DNA, ma solo parziali frammenti (chiamati *read*) come risultato del processo biochimico chiamato *sequenziamento*.

Un trend ricorrente durante l'evoluzione dei sequenziatori è rappresentato dalla crescente presenza di errori di sequenziamento, se nelle read Sanger in media una lettura su mille corrisponde ad un errore, le ultime macchine PacBio sono caratterizzate da un tasso di errore di circa il 15%, una situazione più o meno intermedia è rappresentata dalle read NGS all'interno delle quali questo tasso si attesta su valori attorno al 1%. È chiaro quindi che algoritmi in grado di processare dati con diversi caratteristiche in termini di errori di sequenziamento stanno acquisendo maggiore importanza mentre lo sviluppo di modelli *ad-hoc* che affrontino esplicitamente il problema degli errori di sequenziamento stanno assumendo notevole rilevanza. A supporto di queste tecniche le macchine sequenziatrici producono valori di qualità (*quality scores* o *quality values*) che possono esser messi in relazione con la probabilità di osservare un errore di sequenziamento.

In questa tesi viene presentato un modello stocastico per descrivere il processo di sequenziamento e ne vengono presentate due applicazioni: *clustering* di read e il *filtraggio* di read. L'idea alla base del modello è di utilizzare i valori di qualità come fondamento per la definizione di un modello probabilistico che descriva il processo di sequenziamento. La derivazione di tale modello richiede la definizione rigorosa degli spazi di probabilità coinvolti e degli eventi in essi definiti. Inoltre, allo scopo di sviluppare un modello semplice e trattabile è necessario introdurre ipotesi semplificative che agevolino tale processo, tuttavia tali ipotesi debbono essere esplicitate ed opportunamente discusse.

Per fornirne una validazione sperimentale, il modello è stato applicato ai problemi di clustering e filtraggio. Nel primo caso il clustering viene eseguito utilizzando le nuove misure D_2^q ottenute come estensione delle note misure *alignment-free* D_2 attraverso l'introduzione dei valori di qualità. Più precisamente anziché indurre un contributo unitario al conto della

frequenza dei k-mer (come avviene per le statistiche D_2), nelle misure D_2^q il contributo di un k-mer coincide con la probabilità dello stesso si essere corretto, calcolata sulla base dei valori di qualità associati. I risultati del clustering sono poi utilizzati per risolvere il problema del *de-novo assembly* (ricostruzione *ex-novo* di sequenze) e del *metagenomic binning* (classificazione di read da esperimenti di metagenomica).

Una seconda applicazione del modello teorico è rappresentata dal problema del *filtraggio di read* utilizzando un approccio senza perdita di informazione in cui le read vengono ordinate secondo la loro probabilità di correttezza. L'idea che giustifica l'impiego ti tale approccio è che l'ordinamento dovrebbe collocare nelle posizioni più alte le read con migliore qualità retrocedendo quelle con qualità più bassa. Per verificare la validità di questa nostra congettura, il filtraggio è stato utilizzato come fase preliminare di algoritmi per *mappaggio di read* e *de-novo assembly*. In entrambi i casi si osserva un miglioramento delle prestazione degli algoritmi quando le read sono presentate nell'ordine indotto dalla nostra misura.

La tesi è strutturata nel seguente modo. Nel Capitolo 1 viene fornita una introduzione al sequenziamento e una panoramica dei principali problemi definiti sui dati prodotti. Inoltre vengono dati alcuni cenni sulla rappresentazione di sequenze, read e valori di qualità. Alla fine dello stesso Capitolo 1 si delineano brevemente i principali contributi della tesi e la letteratura correlata. Il Capitolo 2 contiene la derivazione formale del modello probabilistico per il sequenziamento. Nella prima parte viene schematicamente presentato il processo di produzione di una coppia simbolo qualità per poi passare alla definizione di spazi di probabilità per sequenze e sequenziamento. Mentre gli aspetti relativo alla distribuzione di probabilità per la sequenza di riferimento non vengono considerati in questa tesi, la descrizione probabilistica del processo di sequenziamento è trattata in dettaglio nella parte centrale del Capitolo 2 nella cui ultima parte viene presentata la derivazione della probabilità di correttezza di una read che viene poi utilizzata nei capitoli successivi. Il Capitolo 3 presenta le misure D_2^q e gli esperimenti relativi al clustering i cui risultati sono frutto del lavoro svolto in collaborazione con Matto Comin e Andrea Leoni e pubblicato in [CLS14] e [CLS15]. Il Capitolo 4 presenta invece i risultati preliminari fin qui ottenuti per il filtraggio di read basato sui valori di qualità. Infine il Capitolo 5 presenta le conclusioni e delinea le direzioni future che si intendono perseguire a continuamento del lavoro qui presentato.

Abstract

First isolated by Friedrich Miescher in 1869 and then identified by James Watson and Francis Crick in 1953, the double stranded *DeoxyriboNucleic Acid (DNA)* molecule of *Homo sapiens* took fifty years to be completely reconstructed and to finally be at disposal to researchers for deep studies and analyses.

The first technologies for DNA sequencing appeared around the mid-1970s; among them the most successful has been *chain termination method*, usually referred to as *Sanger method*. They remained *de-facto* standard for sequencing until, at the beginning of the 2000s, *Next Generation Sequencing (NGS)* technologies started to be developed. These technologies are able to produce huge amount of data with competitive costs in terms of dollars per base, but now further advances are revealing themselves in form of *Single Molecule Real Time (SMRT)* based sequencer, like *Pacific Biosciences*, that promises to produce fragments of length never been available before. However, none of above technologies are able to *read* an entire DNA, they can only produce short fragments (called *reads*) of the sample in a process referred to as *sequencing*. Although all these technologies have different characteristics, one recurrent trend in their evolution has been represented by the constant grow of the fraction of errors injected into the final reads. While Sanger machines produce as low as 1 erroneous base in 1000, the recent PacBio sequencers have an average error rate of 15%; NGS machines place themselves roughly in the middle with the expected error rate around 1%.

With such a heterogeneity of error profiles and, as more and more data is produced every day, algorithms being able to cope with different sequencing technologies are becoming fundamental; at the same time also models for the description of sequencing with the inclusion of error profiling are gaining importance. A key feature that can make these approaches really effective is the ability of sequencers of producing *quality scores* which measure the probability of observing a sequencing error.

In this thesis we present a stochastic model for the sequencing process and show its application to the problems of *clustering* and *filtering* of reads. The novel idea is to use quality scores to build a *probabilistic framework* that models the entire process of sequencing. Although relatively straightforward, the developing of such a model goes through the proper definition of probability spaces and events on such spaces. To keep the model simple and tractable several simplification hypotheses need to be introduce, each of them, however, must be explicitly stated and extensively discussed.

The final result is a model for sequencing process that can be used: to give probabilistic interpretation of the problems defined on sequencing data and to characterize corresponding probabilistic answers (*i.e.,* solutions).

To experimentally validate the aforementioned model, we apply it to two different problems: *reads clustering* and *reads filtering*. The first set of experiments goes through the introduction of a set of novel *alignment-free* measures D_2^q resulting from the extension

of the well known D_2-type measures to incorporate quality values. More precisely, instead of adding a unit contribution to the k-mers count statistic (as for D_2 statistics), each k-mer contributes with an additive term corresponding to its probability of being correct as defined by our stochastic model. We show that this new measures are effective when applied to clustering of reads, by employing clusters produced with D_2^q as input to the problems of *metagenomic binning* and *de-novo assembly*.

In the second set of experiments conducted to validate our stochastic model, we applied the same definition of correct read to the problem of reads filtering. We first define *rank filtering* which is a *lossless* filtering technique that sorts reads based on a given criterion; then we used the sorted list of reads as input of algorithms for *reads mapping* and *de-novo* assembly. The idea is that, on the reordered set, reads ranking higher should have better quality than the ones at lower ranks. To test this conjecture, we use such filtering as pre-processing step of reads mapping and *de-novo* assembly; in both cases we observe improvements when our rank filtering approach is used.

This thesis is organized as follows. Chapter 1 gives a brief introduction of sequencing and of the main algorithmic challenges related to the sequencing data. In the same chapter are also briefly discussed issues about representation of: sequences, reads and quality values. At the end of Chapter 1 the thesis contribution and related works are outlined. Chapter 2 is devoted to the development of the probabilistic model for the sequencing process. It starts by schematically describing the physical process of producing a single *pair* symbol-quality and continues by defining probability spaces for the sequencing and for the reference. While aspects related to the sequencing processing are discussed and modeled in terms of these spaces, *prior* stochastic models for the sequence are not considered in this thesis. The last part of Chapter 2 derives a closed form for the probability of a read to be correct which is then extensively used in the subsequent chapters. Chapter 3 presents the result of the application of our model to the problem of reads clustering as implemented in our software `qCluster`; these results are based on a joint work with Matteo Comin and Andrea Leoni published in [CLS14] and [CLS15]. Chapter 4 presents the result obtained for the application of probabilistic model to reads filtering by using read correctness probability as sorting criterion on rank filtering defined in the same Chapter 4. Finally Chapter 5 gives conclusions and delineates the future direction we will investigate.

Ringraziamenti

Durante questi ultimi quattro anni, moltissime persone mi hanno (direttamente o indirettamente) aiutato, ognuno a modo proprio ha contribuito in modo decisivo al lavoro di tesi e ne condivide, quindi, una parte. Sebbene elencare tutte queste persone richiederebbe troppo spazio, ad alcune di loro mi sento di dover riservare un ringraziamento speciale

Un grazie ricolmo di amore e di infinita gratitudine lo merita mia moglie Claudia sul cui amore e supporto incondizionato posso sempre contare qualsiasi cosa accada. Mia madre, mio padre e tutti i miei fratelli meritano un ringraziamento speciale per essere stati sempre presenti. Un ringraziamento è doveroso anche nei confronti della mia famiglia acquisita che mi ha accolto come uno di loro in un modo che mai avevo visto prima, il loro prezioso supporto semplicemente non può essere quantificato.

Dal punto di vista accademico molte persone hanno contribuito in maniera unica e determinante al mio modo di pensare e di condurre la mia ricerca scientifica. Ovviamente un ringraziamento speciale lo voglio porre a Gianfranco che mi ha aiutato e supportato sin dal primo giorno. Sempre è stato (ed è ancora) di immensa ispirazione, molte delle discussioni che abbiamo avuto in questi anni hanno profondamente cambiato la mia visione di diversi settori disciplinari e, in generale, della vita. Una buona fetta di questa tesi è mutuata dall'eccellente lavoro condotto in collaborazione con Matteo al quale sono debitore per il supporto e per le preziose indicazioni sul mio lavoro. La mia carriera accademica non sarebbe nemmeno iniziata se non fosse stato per Fausto con il quale (ben prima dell'inizio dei nostri rispettivi dottorati) ho avuto piacevoli e proficui scambi di idee durante gli ultimi e dai quale è scaturito un embrionale lavoro che spero riusciremo presto a portare a termine. Un ringraziamento particolare va anche a tutti i miei colleghi (passati e presenti) del Laboratorio ACG che, oltre ad essere quotidiani compagni di ufficio, mi hanno anche aiutato con numerose idee, suggerimenti.

Da ultimo, ma non per importanza, un ringraziamento speciale lo voglio riservare a tutti i miei amici che in questi anni non si sono mai lamentati delle mie assenze per studio e/o lavoro, ma mi hanno sempre trattato come un fratello, semplicemente non potrei chiedere di meglio.

Acknowledgements

I should mention several people who (directly or indirectly) helped me during the last four years, every single one of them has given me something that is invaluable and each of them shares a fraction of this work. Naming all of them would require too much space but some deserves special mention.

First I have to immensely and lovely thank my wife Claudia, she always gives me unconditional support and love no matter what happens. My mother, my father and all my siblings have always been there since when I started and for this reason I really want to thank them. I'd also like to thank all the members of my acquired family, who welcomed me in a way that I've never seen before and their support simply is immeasurable.

For the academic part there are many people who uniquely and substantially contributed to my way of thinking, approaching and carrying on my research. Of course special thanks go to my advisor Gianfranco, he supported and helped me from the very first day I started, he has always been (and still is) a huge source of inspiration, many discussion we had profoundly changed my vision on many different topics. Part of this thesis has derived from the excellent work I carried on with Matteo who I'd like to thank in a special way for his support and precious suggestions and feedbacks on my work. My academic career would have never started not being for Fausto who deserves for this a special mention and many thanks for being my "sponsor" but also for the long and interesting discussions we had during the time he spent in Padova (started way before we started our Ph.D.) and, last but not least, for the work we made together which I hope will be soon resumed. Special mention is also due to all my colleagues (past and present) at the Advanced Computing Group that, during these years, have been not only daily mates, but also precious sources of suggestions and ideas.

Last but not least all of my friend share a part on this work, they never complained for me staying at home working, and they always treated me as a brother; they really are the best friends one could imagine.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Learning medicine consists in part of
learning the language of medicine
(Daniel Kahneman)

Since the publication of *On the origin of species* by *Charles Darwin* and even before, humankind has always tried to answer questions about its origins and evolution. Such inquiry drove scientific research through both huge spaces of universe and microscopic hidden corners of tissues and cells. One of the most important milestones of this journey has been the stunning discovery of the *DeoxyriboNucleic Acid (DNA)* molecule and the identification of its paramount role in biological processes. First isolated by Friedrich Miescher in 1869 and then identified by James Watson and Francis Crick in 1953 [WC53], the double stranded DNA molecule of *Homo sapiens* took fifty years to be completely reconstructed and to finally be at disposal to researchers for deep studies and analyses. Today, more than 150 years after its first observation, DNA molecule still remains the center of many projects aiming to fully understand the biology of different organisms. Many biologists, and specifically *molecular biologists*, all over the world are searching answers to many interesting questions about macroscopic phenomena (*e.g.,* chronic diseases, genetic disorders, ...) by observing this microscopic molecule which contains all of our genes. The amount of data they need to process at any time, combined with the complexity of the DNA, poses new challenges to biologists and, nowadays, simple expert data analysis is no more feasible leaving the usage of automatic tools for data processing the only available option.

1.1 Shotgun sequencing

As computational power became cheaper and extensively available, scientific communities of various fields began to take full advantage of it by developing innovative approaches and techniques to exploit such advances within their own research fields. Life sciences were no exception in this *gold rush*. What before was only theoretically possible, suddenly became feasible and many techniques started to gain interest and importance; as a result ambitious projects, like the reconstruction of the whole human genome, became more in handy than ever before.

A key role in this process has been played by the development of technologies able to produce a digitalized form of genetic material (DNA, RNA, ...) contained in biological samples (*e.g.,* cells, tissues, ...). From the computational standpoint, the DNA molecule can be described as an ordered sequence (*string*) of symbols (*characters*); each of these symbols represents one of the four possible *nucleotides* (or *bases*): **A**denine, **C**ytosine, **G**uanine and **T**hymine (A,C,G and T) composing the molecule.[1]

In an early work Rodger Staden [Sta79] introduced *shotgun sequencing* methodology which uses biochemical reactions to read the nucleotides of small fragments (often called *reads*) obtained from the original sequence (often called *reference*).

Shotgun sequencing, more precisely, refers to the process of preparing DNA samples for a *sequencer* that physically performs the read operations. Since these sequencer machines are based on biochemical reactions that limits the amount of consecutive nucleotides that can be read, the entire process outputs a whole collection of fragments. When considered isolated, each fragment covers a tiny fraction of the original sample, however, when considered as a whole set, the reference sequence is represented on average γ times, the parameter γ is called *coverage*.

What made shotgun sequencing really appealing was the development of *ad-hoc* algorithms able to process many of such fragments in a small amount of time. As shotgun sequencing began to be investigated and studied in detail, it was observed that, despite the increasing power of available supercomputers, the presence of repetitive and "complex" structures on genomic sequences, makes *brute-force* approaches too complex when not impossible to implement. A possible solution to this problem relies on biology experts that, by spending many hours on sequence analysis, identify zones of the genome that may be of particular interest in order to *classify* and *tag* interesting parts of the sequence under investigation and focus the computational efforts only on these parts. Although still useful and successful, such an approach

[1] In RNA **U**racil (U) is present in place of thymine.

can not scale with the ever increasing amount of available sequencing data, especially after a recent study by Djebali *et al.* [DDM+12] showed that the human DNA contains important and essential information in almost all of its parts (including those previously named *junk DNA*). It is now clear that the task of manually classifying different portions of the sequence can not be used anymore to reduce the amount of data to be processed. It is therefore necessary to develop *effective*, *efficient* and *scalable* algorithms to, first reconstruct sequence from sequencing data and then process it. An accurate and smart design of such algorithms has become even more important recently thanks to the explosion of shotgun sequencing experiments ignited by the advent of *Next Generation Sequencing (NGS)* technologies. As new data becomes available, the request of automated analysis tools increases; the bioinformatics community is continuously facing the thrilling challenge of developing and designing new methods to process data in a faster and more effective way than ever before.

1.1.1 Sanger sequencer, the first generation

The genomic era started around the mid-1970s with the development of the first sequencing technologies; among them the most successful has been *chain termination method* usually referred to as *Sanger method*, after Frederick Sanger, one of the authors of the original paper [SNC77].

The process starts by separating the two DNA strands by means of heating; a synthetic nucleotides fragment called *primer* is then attached to one of the strand. This primed template is successively inserted into a mixture containing the necessary reagents to start the *chain reaction*. The result of such reaction is then excited using *gel capillary electrophoresis* and, by means of either dyeing or radio labeling, the sample is read and translated into a sequence of bases.

Despite being one of the first available methods for DNA sequencing, Sanger technology possesses several desirable properties. Thanks to the relatively long reads (between 600 and 900 bases) and the high reliability (less than 1% error rate), this technology is still the first choice when high quality data is needed like, for example, in the completion of complex parts of genomic sequences. The main problem of Sanger methods is represented by its high costs (few thousands dollars per one million bases) especially when compared with *next generation sequencing* methods.

1.1.2 Next Generation Sequencing (NGS)

Sanger technology remained the *de-facto* choice for sequencing for more than thirty years, during this period huge projects were successfully completed. One of the most

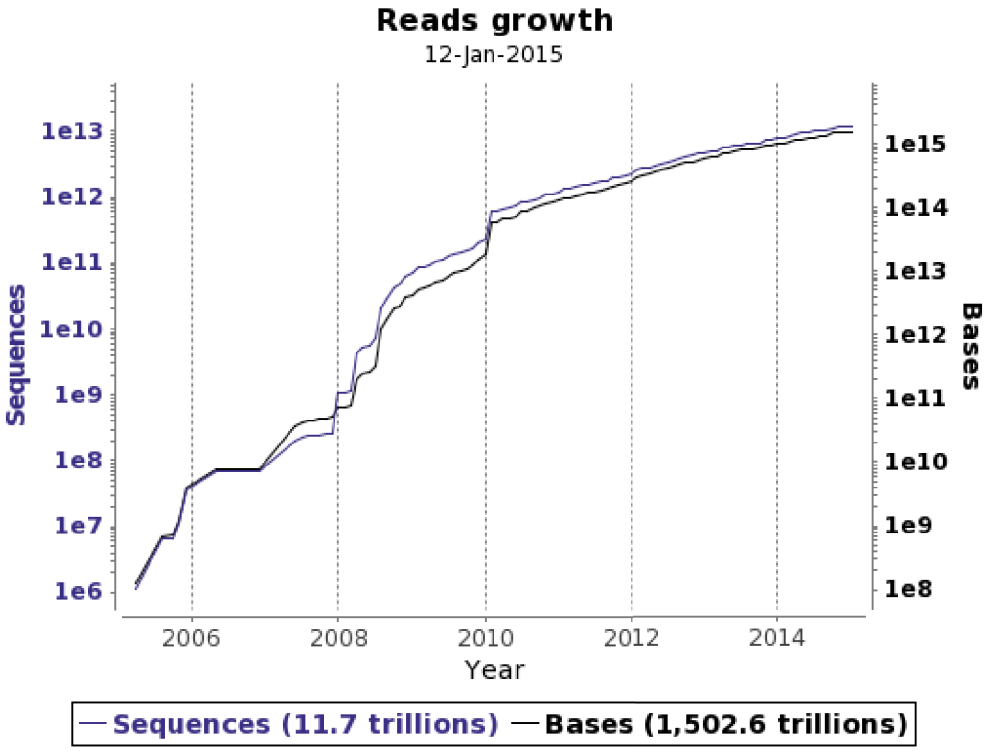


Figure 1.1: Growth rate for the *Sequence Read Archive* (SRA) [2]

important of them has been the *whole human genome* project that culminated with the first *non-draft* version of the human (*Homo sapiens*) genome resulting from the joint work of many research groups all over the world (Lander *et al.* [LLB+01]). At the beginning of the 2000s new sequencing methods started to be developed by several life science and bioengineering companies; as a result of this competitive market several novel methods for shotgun sequencing were available, all of them were (and still are) called *Next Generation Sequencing* (NGS) and sometimes referred to as *High Throughput Sequencing (HTS)*.

Such new technologies rapidly gained popularity within the scientific community, thanks to their low cost; by using NGS sequencers, it is currently possible to produce one million bases for as low as 0.1$, five order of magnitudes cheaper than Sanger sequencers. Such new methods have really changed the way the scientific community tackles at genomic projects (Metzker [Met09]); since their introduction the amount of massive sequencing projects initiated have increased day by day and, as groups all over the world started to share their experiments, the amount of data available rapidly increased as shown in Figure 1.1.

Among all of these projects it is worth mentioning the *1000 Genome Projects*[3]

[2] `http://www.ebi.ac.uk/ena/about/statistics` accessed January 12th, 2015
[3] `http://www.1000genomes.org/`

which aims to sequence a total of 1000 human individuals belonging to different races; as the project's site reports *[...] the goal of the 1000 Genomes Project is to find most genetic variants that have frequencies of at least 1% in the populations studied [...].* Finding such variants as a whole is much harder than the barely reconstruction of 1000 individuals genome which, in fact, is the first necessary step to be completed prior to data analyses. Tasks with such a complexity, unfeasible before NGS technologies appeared, pose new challenging problems to the computer science community which is today committed to the problems of efficiently storing, retrieving and analyzing huge amount of sequencing data.

One of the key feature of NGS methods is the ability of producing very large amount of data during a single experiment; this reduces the total cost of both reagents and experts needed to supervise the entire sequencing process and the combination of these two factors allows NGS sequencers to attain competitive costs in terms of dollars per base.

Although different NGS technologies apply different chemistry and different principles to obtain reads from sample(s), they all share the same high level procedure which is here summarized.

1. A collection of fragments of the sample DNA (*i.e., library*) is prepared and ligated to the ends of a synthetic DNA fragment (*i.e., adapter*). Different technologies use different adapters, this is one of the main source of differences of error models and *biases* between technologies.

2. The fragments are then immobilized into a solid surface in order to produce massive copies of the original fragments. The immobilization aspect is the key feature allowing NGS machines to produce millions of copies per single experiment.

3. Finally the copies are sequenced using platform specific methods and the final dataset is then created. During this step the machine must assess the actual bases of the sample; this operation is usually performed by dyeing or radio tagging and it's another discriminant between different sequencing technologies.

If on the one hand NGS technologies supply researchers with overwhelming amount of data, on the other hand this does not come for free, NGS sequencers have several drawbacks that complicate data analysis and processing.

When NGS sequencers were first introduced, they were able to produce fragments with length at most few hundreds of bases, back then this was a huge disadvantage when compared with Sanger method which was (and still is) able to produce fragments of length up to 700 bases; nowadays NGS methods can compete with Sanger

in terms of read length (although short reads still represent an important fraction of whole data publicly available). A second negative aspect of NGS data regards their reliability; next generation reads are known to be noisier than Sanger in terms of error rate defined as number of incorrectly sequenced bases over the total number of produced bases. If Sanger method has an average error rate below 1%, NGS methods, on average, produce data with a rate that is roughly one order of magnitude bigger (*i.e.,* reads with 1% errors or above are not uncommon). With such an error rate, the chances of miscalled bases within a single read become really high and, while designing and developing algorithms for NGS data, researchers must take this aspect into account.

Most of the NGS sequencers are able to produce *mate pairs* which are pairs of reads sequenced using the same sample but starting from the opposite ends of the same fragment. The advantage of mate pairs is the possibility of using different *libraries* with different fragment length. This allows sequencers to produce pair of reads with an approximately known spacing, called *insert size*, much larger than the actual read length. When repetitive parts of a reference sequence are longer than the read length, mate pairs represent the only possible way of solving them. Another problem where mate pairs have successfully been used is *scaffolding* where a collection of subsequences (usually *contigs* produced by an assembler software) need to be oriented and positioned with respect to each other.

A detailed survey on next generation sequencing technologies and on their application can be found in [Met09].

1.1.3 Future generation sequencing: PacBio

NGS methods are now the standard in sequencing technologies, however further advances are now revealing themselves. The most promising *future generation sequencer* has been introduced by *Pacific Biosciences* and is based on a new sequencing procedure known as *Single Molecule Real Time sequencing (SMRT)* introduced by Eid *et al.* in [EFG+09]. The commercial product, usually referred to as *PacBio* and recently presented and discussed by Carneiro *et al.* in [CRR+12], produces reads with characteristics that are considerably different from both NGS and Sanger ones. PacBio reads can be as long as ten thousands bases, a size never been available before, but the error rate on such reads is as high as 15% with 12% insertions, 2% deletions and only1% substitutions.

Reads of a never available before length and with a unique and peculiar profile of error distribution, make algorithms developed for all previous technologies not feasible or not effective for this type of data, therefore, while some approaches could

simply be adapted to work with newer datasets, in some other cases a design *from scratch* becomes necessary.

1.2 Algorithmic challenges

The available variety of sequencing technologies, each having its own *reads profile* (in terms of error, length, biases, ...), makes the task of designing and developing efficient and effective algorithms very challenging. In this scenario it is not true that *one size fits all* and approaches designed for one technology may be completely useless with another one. Each feature of the produced reads has an impact on the development of algorithms and it is important to be aware of all such aspects before starting design of new methods.

High coverage

As already mentioned, a big role in the success of NGS technologies can be attributed to their ability of performing massive sequencing at a relative low cost. Although desirable, high coverage generates huge amount of reads that need to be stored and processed, the choice of data structures and algorithms used is therefore crucial when it comes to NGS data.

Even when the most efficient approaches are adopted, datasets may still be too big and the only remaining solution relies on either *filtering* high and low quality data or reducing the size of the set by computing a new dataset that somehow maintains the important information of the original one.

Short reads

Genomic sequences are the result of the evolution process which the corresponding species experienced, while some organisms (like humans) have a relatively short history, many others (like plants), have gone through several millions years of history and evolution. Each time the DNA replicates, there is a small chance that the process creates mutations of the nucleotides sequence. There are several types of mutations that can happen during the replication process, most of them have the net effect of replicating portions of the sequence in other position of the genome itself, this creates complicated structures generically called *repeats*.

The characteristics of repeated structures on a real genomic sequence, are quite variable, the same sequence (like, for example, the human one) can contain few bases as well as thousands bases long repeats. The amount of copies of the same repeated sequence is also a variable factor, we may find structures that are repeated few times

as well as sequences that appear thousands of time in the same genome, finally the relative position of repeated regions can be adjacent (like *tandem repeats*) or far a part from each other.

Short reads make the process of discovering and resolving repeats really hard, when not impossible. For example, a read sequenced inside a repeat longer than the read itself, can not uniquely mapped into either of the positions where the repeat occurs on the reference. NGS technologies can currently be used to resolve short to mid long repeats, but some structures need to be resolved using different data (like, for example, mate pairs or PacBio reads).

Errors

Noisy data may look like a minor problems especially for rates 1% or below; however combining this with the issues already discussed (short reads, repetitive structures, ...) creates a challenging mix that complicates the design of algorithms. Methods for error correction are often part of the processing *pipeline* (*i.e.,* the chain of successive tools applied to the input set) and may rely on pre-processing (*e.g.,* filtering) or on "online" procedures that try to identify errors as "unexpected" behaviors of the algorithms (*e.g.,* divergence from a specific path). Moreover, with the new technologies (like PacBio) gaining interest, newly developed algorithms should be designed to work with new error profiles.

Although each single challenge may not seem that difficult to cope with (with the possible exception of repeats that in some situation make the solution not unique); the combined effect of all of them makes bioinformatics problem really hard when not impossible to solve without resorting to heuristics or approximate algorithms (Treangen and Salzberg, [TS11], Nagarajan and Pop [NP09] Pop and Salzberg [PS08]).

1.2.1 Assembly

As already mentioned, the biggest success obtained so far has been the reconstruction of the whole human genome sequence. The process of reconstructing a genetic sequence starting from fragments of thereof (*i.e.,* reads) is called *assembly*. Assembly can be carried without any knowledge of the sequence (*e.g.,* sequencing a newly discovered organism) in which case the problem is named *de-novo assembly*. When the sequence is reconstructed using another related sequence as guide, the problem is called *comparative assembly*. Since the former is a harder, more interesting and a prerequisite to the latter, in this thesis we will focus only on *de-novo* assembly.

Giving a formal definition of the problem of assembly is not straightforward; an

easy and correct definition states that the assembly is *the problem of reconstructing the (unknown) reference sequence from a collection of (known) reads.* This definition is, for all practical purposes, useless because of its recursive nature arising from the mention to the solution itself (*i.e.,* the reference sequence).

Several attempts to formalize the problem of assembly have been made; sometimes (especially in theoretical analyses) assembly is defined as the problem of *shortest common superstring (SCS)* where the unknown reference is the shortest string containing all the reads as subsequences (with possibly mismatches to account for sequencing errors). It easy to construct an example containing (simple) repeated structures that makes this definition incorrect for the problem of the assembly.

With the knowledge available today the only model sufficiently precise to describe a DNA, is the entire sequence of nucleotides or, in other words, the output of the assembly process. As a consequence, we can not have a precise definition of the solution of the assembly problem, at least from a mathematical standpoint, but we must deal with the fact that *the assembly problem can only be described heuristically.* Notwithstanding these technicalities, assembly is a fundamental problem in the field of biology, in fact it is probably the most important problem which need to be solved before starting the analysis of genetic material of an organism. Moreover there is abundant evidence that the problem is solvable in practice as attested by the increasing of the number of assemblies publicly available.

Reconstructing a sequence with millions or even billions of bases, using fragments no longer than 1000 bases, is a hard task regardless the sequencing technology used. The problem is even worse because of the complex structures that genomic sequences contains (*e.g.,* repeats, mutations, ...). sequencing data make the problem even harder especially when performed using very short reads nowadays accessible. Even with the advent of the future generation sequencing technologies, the assembly problem will still remain challenging because the reads length will remain much shorter than the genomic sequence to reconstruct and the errors within reads can only complicate the problem and its solution.

When NGS data became massively available, existing assembly algorithms turned out to be no more suitable; with hundreds of millions of reads it was impractical to perform pairwise overlaps between reads which was a necessary step of all of the assembly algorithms since then developed. Consequently the community started to develop new approaches with the main goal of designing algorithms and methods able to efficiently process NGS data. A survey on current most successful approaches for assembly using NGS data has been published by Miller *et al.* in [MKS10]; we give here a brief introduction and discussion of the most important aspects of these

methods.

According to [MKS10] assembly algorithms can be classified into three different categories: *greedy* algorithms, *Overlap Layout Consensus* and *De Bruijn graphs*.

Greedy

Greedy algorithms start by selecting a *seed* read (or set of reads), which represents the initial assembly and proceed by *finding the best alignment* to extend it. At each iteration the read giving the *best extension* is chosen and removed from the original set; the process continues until either the set is empty or no good enough alignment can be obtained with the remaining reads. This approach is clearly greedy since, once a read is used to extend the current assembly, it is removed from the input reads set and every successive extension will not *roll-back* the current decision. The advantage of greedy strategies is that they are fast and easy to implement but, on the other hand, it is very likely that the algorithm stops on a *suboptimal* solution (especially with big and noisy sets). Attempts to escape local maxima using randomized initial seeds, does not help because of the size of the solution space which contains $O(4^N)$ elements with N being size of the sequence to be reconstructed.

Overlap Layout Consensus (OLC)

Overlap Layout Consensus algorithms divide the task of producing an assembly into three subsequent phases. During the first *overlap* phase an *all-against-all* pairwise read comparison is performed to determine the best possible overlaps. Since this phase could require a lot of computational time, usually *seed* overlaps are detected using k-mer sharing (*i.e.,* identification of pairs of reads that share at least a certain amount of k-mers) which is easier to compute than an actual alignment, only for those pairs of reads that satisfy the k-mer requirement the actual alignment is then computed. In the second phase alignment relation between reads is *laid out* on a graph where each node represents a read and an arc between nodes exists if a valid alignment has been detected between the corresponding reads. After the graph has been simplified (*e.g.,* redundant structures are removed), the third step, consisting of a path detection algorithm, is run to identify a *consensus* sequence that is finally outputted as a candidate assembly.

De Bruijn graphs

There are two major problem with OLC algorithms, the first one is related to all-against-all pairwise alignment which is a time consuming operation and the second

problem is that the phase of path discovery requires to find a *hamiltonian path* which is known to be an NP-hard problem (Cormen *et al.* [CLR+01]). This last problem has been one of the main motivation to the development of assembly algorithms based on *De Bruijn* graphs where the idea is to create a so called *k-mers* graph, which is a simplified version of De Bruijn graph.[4] On k-mer graphs each node corresponds to a k long sequence called *k-mer* and edges represent all the $(k-1)$-mers that has been observed (*e.g.,* in a set of reads or in a long sequence).

The k-mer graph used by assemblers is constructed by scanning the collection of reads inputted, for each k-mer encountered in, at least one, read a node is created. Contrarily to the original definition of De Bruijn graphs, in k-mer graphs *not all possible k-mers have a corresponding node.* If the reads were sequenced without error and with *perfect* coverage (*i.e.,* sufficient to make the problem solvable), then the graph constructed from the read set and the (hypothetical) graph constructed from the reference sequence would be identical and would contain an *Eulerian path* representing the original reference sequence. One of the strength of this approach is that computing eulerian is an easy task in the sense that an eulerian path can be discovered in linear time. The problem is that we have no guarantee that the path we can identify in linear time corresponds to the desired assembly.

De Bruijn graph based approaches to the problem of assembly were first introduced by Pevzner *et al.* in [PTW01] and have since then considerably evolved [MKS10]. The experimental part of this thesis (chapters 3 and 4) uses `VELVET` software (Zerbino and Birney [ZB08]) which is a popular tool for *de-novo* assembly based on De Bruijn graphs.

1.2.2 Comparative genomics

Comparative genomics is the biology field that studies genetic sequences with the goal of identifying and classifying biological features shared by different organisms or by different individuals of the same species. Practically this can be achieved using genome analysis techniques to test the correlation between sequences. There are many known correlation structures for which specific algorithms exist: *variant detection*, *rare variation and burden testing*, *identification of de-novo mutations* are few of them and a detailed survey is given by Koboldt *et al.* in [KSL+13]. With the introduction of NGS data, the number of organisms that can be simultaneously compared has substantially increased, as a consequence of this, also the overall complexity of the problems increased. Moreover, using *High Performance Computing*

[4] De Bruijn graphs were originally proposed by De Bruijn and Erdos [dBE46] to represent all possible overlaps between k long sequences.

(HPC) techniques, we can nowadays perform comparative genomics analysis at the genome scale and *Genome-Wise Association Studies (GWAS)* are now becoming fundamental steps for all comparative genomics projects.

For many years genome assembly and sequence alignment have been essential primitives to perform comparative genomics studies however, with the advent of massive sequencing and NGS technologies, many of the approaches based on these primitives became no longer practical. When looking for variations between sequences, (short) reads need to be aligned to a reference allowing non perfect matching that are, indeed, the variation to be discovered. When performed on millions (or even billions) of reads, this task becomes compute intensive and standard alignment tools may not be the best choice.

Comparative genomics as a whole includes many different biological problems like: *genome comparison*, *metagenomic binning*, *variant discovery*, *phylogenetic tree reconstruction* and many others. Most of them, however, require mapping between sequences which is often implemented using alignment techniques described in the next paragraph.

Alignment of sequences

Since the beginning of the genomic era, researchers have mostly been interested in finding coding sections (*i.e.,* parts containing genetic information) of the human genome. More generally one of the main task has been (and still is) finding recurrent *patterns* (*i.e.,* substrings) and classifying them according to the role they have in the regulation of human biology. Pattern identification and reconstruction as well as many other problems at the heart of bioinformatics, resort to the fundamental algorithmic primitive of *alignment* (or *mapping*). Informally *alignment is the process of superimposing two different sequences in order to obtain the best possible match between the two of them.*

Alignment can be performed with or without mismatches which means that a certain degree of difference between superimposed sequences may be tolerated; due to mutations induced by evolutionary events, alignment is almost always performed allowing mismatches; how the mismatches are treated, is a matter of the specific algorithm.

One of the most popular methods for sequence alignment relies on a dynamic programming approach named *Smith-Waterman*, after the authors the original paper [SW81] published more than thirty years ago. The idea is to define a recurrence that assigns *scores* to: matches, mismatches, insertions and deletions. The algorithm starts with an *empty* alignment and, recursively, extends it using a scoring function

that penalizes mismatches, insertions and deletions. Given two sequences x and y with size n and m respectively, the Smith-Waterman algorithm computes an $n \times m$ matrix. This matrix is then used to derive the optimal sequence of string operations (*i.e.,* substitutions, insertions and deletions) that transforms x into y. The complexity of this approach is $O(nm)$ for time while the space (still $O(nm)$ with a naïve implementation) can be kept $O(m + n)$ using proper techniques. Another popular approach, first proposed and implemented by Altschul *et al.* in a tool called `BLAST` [AGM+90], relies on hash maps to perform fast alignment of sequences based on their k long subsequence (*i.e.,* k-mers). The alignment obtained using hash maps may be refined using dynamic programming algorithms or other algorithms if needed. With the advent of NGS data dynamic programming approach started to become impractical, the main reason does not lay on the complexity of a single alignment itself, rather on the number of alignments required when the input set contains millions of reads. Moreover the shorter the reads are the more likely they map in more than a single position of the reference sequence, this means that multiple positions can give the same (optimal) score for one given read.

Consequently, new alignment algorithms have been devised to specifically work on NGS data and, at the same time, *alignment-free* approaches to sequence comparison and pattern discovery have started to gain interest in the scientific community.

Alignment-free sequence comparison

The use of alignment tools like `BLAST` to assess the degree of correlation between two sequences is currently the dominant approach. Alignment-based methods produce good results only if the biological sequences under investigation share a reliable alignment, however there are cases where these methods cannot be applied. This can happen, for example, when the sequences being compared do not share any statistical significant alignment, a case that can occur when sequences come from distant related organisms, or they are functionally related but not orthologous (*i.e.,* coming from a common ancestor). Moreover, as discussed above, alignment methods are usually time consuming, thus they can not be applied to large-scale sequencing data produced by NGS technologies.

For these reasons *alignment-free* techniques are rapidly gaining interest, the basic idea is to avoid alignment (as the name suggests) and assess correlation using proper statistics measures that can be computed efficiently.

Alignment-free sequence comparison methodology was introduced during the mid 80s with the seminal paper of Edwin Blaisdell [Bla86] where the D_2 statistic was proposed to correlate different sequences based on the frequency of their constituent

k-mers, with k being an adjustable parameter. The idea, although simple, proved to be effective especially after several improvements have been developed by Reinart *et al.* [RCSW09] and by Wan *et al.* [WRSW10]. Recently, alignment-free techniques have been used to perform *assembly-free* sequence comparison using NGS data (Song *et al.* [SRZ+13], Comin and Schimd [CS14]), a good survey on the most recent advances in alignment-free techniques can be found in [SRR+13].

1.2.3 Clustering

Clustering (sometimes called *cluster analysis*) is the process of partitioning a set into κ disjoint subsets called *clusters*, in such a way that elements belonging to same cluster share some common *features* while being distinguishable from elements on a different partition. More precisely, given a *distance* measure (sometimes referred to as *dissimilarity* function) clustering constructs a partition such that distance between elements on the same cluster is minimized among all clusters. Note that the distance measure does not need to be a distance in the mathematical sense (in fact most of the alignment-free, like D_2, measures are not mathematical distances). A good survey on many different clustering techniques can be found in Xu and Wunsch [XW05].

Centroid based clustering In centroid based clustering the idea is to associate to each of the κ clusters a point of the input space called *centroid*. The partition is iteratively produced starting from κ *seed* centroids that are randomly generated. At each iteration the input set is scanned and each element is assigned to the cluster associated to the centroid that minimizes the distance; after all elements have been reassigned, a new set of centroid is computed using this new assignment (typically the new centroid is the average over all the elements assigned to the cluster). The procedure iterates until a stopping condition is reached (for example the maximum execution time is exceeded or there are no significant changes on the clusters between two consecutive iterations). This algorithm, and centroid based clustering in general, is usually referred to as *k-means*.

One weakness of k-means is represented by the initial random generation of the seed partition; to overcome as much as possible biases related to this generation, k-means is usually run several times and a function (*e.g.,* average) of the all different clusterings is computed as final output. This solution mitigates (but does not eliminate) the possibility of k-means producing local optima, however it has been shown that finding an optimal solution to k-means clustering is a NP-hard problem (Aloise *et al.* [ADHP09]). Another critics often moved to k-means is that it requires κ (the number of clusters) to be known in advance which is often an unrealistic hypothe-

sis. To cope with this problem many different extensions (most notably *hierarchical clustering*) have been developed and successfully applied in many different scenarios ([XW05]).

Clustering in bioinformatics Clustering has been successfully used on many different computational biology problems. The idea shared by most of these applications is that, by defining a proper distance measure between sequences, cluster analysis should be able to group together sequences that share some common biological features. A field where clustering can be profitably used is the one of *metagenomics*, for example a simple problem where clustering has been successfully used is the separation of reads produced by a metagenomics experiment: *metagenomic binning*. Such experiments use a single run of a sequencer machine to produce reads from many different organisms living in the same culture. Unfortunately reads coming from different organisms are indistinguishable by the sequencers which outputs a single set containing all (heterogeneous) fragments; as a consequence if the partition of all reads is needed, it must be inferred using computational tools like clustering (Solovyov and Lipkin [SL13]). In Chapter 3 our probabilistic quality value based on model is applied to clustering of reads, results of such experiments have been published by Comin *et al.* in [CLS14] and [CLS15]. Another application of clustering is as a preprocessing phase of the assembly algorithms, the idea is that assembly performed only on reads belonging to the same cluster, under certain circumstances, could give better result than assembling the entire dataset (*i.e.,* without clustering), as it turns out this is true for reasonably high sequencing coverage [SL13, CLS14].

1.3 Data representation

An aspect of bioinformatics which is often underestimated is the representation of reads and sequences into files that are stored on public databases available to be downloaded and processed. For several, equally valid, reasons there is not a unique standard for such representations. First of all in relatively recent fields, like bioinformatics, not enough time have passed for a convention to become a standard and the lack of a recognized authority further slows down the process of standardization. Also, in a rapidly changing field, the process of defining standards is complicated by the rapid technological evolution that may not be compatible with already defined formats and needs, therefore, to define new ones.

Remarkably the trend is now moving toward a more standardized approach, new technologies and tools (*e.g.,* sequencers, algorithms, ...) tend to adopt common

formats and the entire industry is embracing a more interoperable approach. Nowadays e relatively low number of different formats are becoming *de-facto* standards for the representation of genetic and sequencing data, we give in this section a brief introduction to some of these conventions.

IUPAC nucleotides representation For the representation of nucleotides and, more generally, of any possible subset of them, an almost universally adopted convention has been proposed by the *International Union of Pure and Applied Chemistry (IUPAC)*, Kozl and Listy [KL78]. More specifically the IUPAC standard defines, for every possible subset of the 4 nucleotides $\{A, C, G, T\}$, a letter that represents it. Trivially A, C, G and T are assigned to the corresponding nucleotides while other sets are encoded with different letters, for example the letter N stands for a**N**y of the 4 nucleotides.

1.3.1 Fasta and Fastq

Before being able to be automatically processed by algorithms, genomic sequences and sequencing data need to be stored in digital format. There are two main file formats for storing sequences data: `fasta` and `fastq`. These two formats have been developed to convey all the needed information about sequences. While `fasta` provides a structure to store information about the sequence and possibly *meta data* (*e.g.,* organism, public database reference number, length, ...), `fastq` format adds support to quality scores. Even if the two formats look similar, they have some minor, yet subtle, differences. Since this thesis presents methods and models that rely on quality scores, only `fastq` format is here briefly presented.

The general form of a `fastq` *entry* contains four lines:

```
1 @Header
2 Sequence
3 +Repeat Header (optional)
4 Qaulities
```

`@` and `+` symbols are used to identify header(s) (the header after `+` is optional and, usually, is a repetition of the header found after the character `@`). The sequence is represented as a string of characters from the *IUPAC* standard but most frequently only the four bases `A`, `C`, `G` and `T` are used with the special symbol `N` used to indicate a position that the sequencer has not been able to reliably identify.

Since `fastq` files are used to store reads, they have several of entries each of them

Format	**Offset** q_{off}	**Range** $[q_{min}, q_{max}]$
`fastq-sanger`	33	$[0, 93]$
`fastq-solexa`	64	$[-5, 62]$[6]
`fastq-illumina`	64	$[0, 62]$

Table 1.1: Summary of parameters for different quality scores encoding used in `fastq` files

representing a single read.[5]

1.3.2 Representation of quality scores

Quality scorse are integer values in the interval $\mathcal{Q} = [0, q_{max}]$ with $q_{max} < +\infty$ (usually $q_{max} = 50$), they are encoded in the fourth (and last) field of a `fastq` entry. The usual way of listing qualities is by encoding the actual integer value q with an ASCII character $enc(q) = q + q_{off}$ with q_{off} being a quality offset.

For example if $q_{off} = 64$ and $q = 30$ then the character $enc(q)$ would be the one with ASCII value 94 which is the `^` character.

Unfortunately there is no unique encode and different manufacturers use different convention, luckily there are three similar converging standards: `fastq-sanger`, `fastq-solexa` and `fastq-illumina`, the difference between all these representations is on the value of the offset q_{off} and on the range of values that are defined, Table 1.1 gives a summary of these parameters [CFG+10].

1.3.3 Color space

Most of the sequencing technologies produce as output files in either `fasta` or `fastq` format, where sequences are represented using nucleotide encoding as defined by the IUPAC standard. A notable exception is represented by the SOLiD technology of Applied Biosystems which uses a different sequence coding called *color space* or *2 base color code* discussed by Breu in [Bre10].

Color space defines an alphabet $\Sigma = \{0, 1, 2, 3\}$ of so called *colors*. Colors arise from the sequencing process implemented in SOLiD sequencers where each *dimer* (pair of bases) is sequenced in a single atomic operation. Given an initial base b, the

[5]This most of the times creates redundancy because part of the header (*e.g.,* the name of the sequenced organism) is usually shared by all reads.

[6]Solexa allows negative qualities because it uses a different equation to translates probability into scores, more precisely the equivalent of equation (2.3) for solexa model is given by

$$P_e(q) = (10^{-q/10} + 1)^{-1}$$

	A	**C**	**G**	**T**
A	0	1	2	3
C	1	0	3	2
G	2	3	0	1
T	3	2	1	0

Table 1.2: The di-base encoding matrix for SOLiD reads

base sequence can be reconstructed from a color sequence using the *di-base matrix* reported on Table 1.2. Let, for example, consider the sequence of colors 01120232 with the initial base A.

```
A 0 1 1 2 0 2 3 2
 A A C A G G A T C
```

The first A corresponds to the one given as *initializer* of the color space while the remaining bases are obtained via inversion of the di-base matrix.

The di-base coding has been derived to resemble the physical sequencing process (as mentioned earlier) but also to give a coding with specific symmetry properties that are extensively discussed in [Bre10].

Also SOLiD sequencer, as all the modern NGS machines, produce one quality score for each of the called color, however if one wants to use such scores in a probabilistic model, proper events (in terms of colors rather than bases) must be defined. In principle the model presented in Chapter 2 could be used with color space data provided that the proper interpretation of quality scores is used and that the alphabet Σ is always assumed to be the alphabet of colors rather than the one of bases.

`csfasta` and `qual` files Given the different nature of SOLiD data, the files outputted by these sequencers follow a slightly different convention. More specifically SOLiD sequencers output a pair of files. The first file is a so called *color space fasta* (`csfasta`) which is a `fasta` file where, instead of using IUPAC encoding for bases, the sequences are given in color space. The second file (`qual` file) contains the list of quality scores usually given as a list of integer.

A read in a `csfasta` file looks like follows

```
>487_14_960_F3
T11001333 [...]
```

and the corresponding quality scores entry in the `qual` file looks like

```
>487_14_960_F3
33 32 4 8 2 31 31 2 [...]
```

Note that, at the beginning of read an *initializing base* is always given, this, however, is not an actual sequenced base but just a necessary information supplied to allow conversion from color to base space.

From now on color space will not explicitly considered in our models, however, in principle it could be applied without major modifications; for this reason we will, as much as possible, refer to a generic *symbol* whenever either base or color could be used.

1.4 Thesis contributions

The main contribution of this thesis is the development of a stochastic model for the sequencing process and its application to the problems of *clustering* and *filtering* of reads. The idea is to use quality scores produced by sequencing machines to build a probabilistic framework that models the entire process of producing a set of reads each of which contains a sequence of symbols (*e.g.,* bases), as well as a sequence of quality scores. Although modeling sequencing process is not a new idea, in our opinion the inclusion of quality values has not received enough attention, to the best of our knowledge this is the first model that describes whole sequencing experiments incorporating quality values. We think that our model can be successfully used in many application, either to improve performance of already existing approaches, or to develop new methods and algorithms for bioinformatics problems.

We will present our model in Chapter 2 where we start from the interpretation of quality scores as phred scaled values of the probability of corresponding base being correct and arrive to the description of the process of producing an entire set of reads.

Generally speaking, each time a hypothesis is introduced, the model looses some of its expressiveness, unfortunately this cannot be avoided when describing complex phenomenon such as a sequencing experiment. Some of the hypotheses introduced throughout chapter 2 can be relaxed while other have been introduced with the only goal of assuming specific probability distribution and can be replaced by others (*e.g.,* with maximum likelihood estimators).

Notwithstanding the nature of these assumptions, we believe that proving a model to be effective and useful, even in a very restrictive environment, always gives a tool that can be used to solve specific problems.

The second part of thesis (corresponding to chapters 3 and 4) presents an experimental validation of the stochastic model with its application to two specific bioinformatics problems: *clustering* and *filtering* respectively. As mentioned above, proving a model to be effective is desirable and also necessary in a Ph.D. thesis, these two chapters present results which indicates that our model can be effectively used in real world problems. More precisely, Chapter 3 presents an application of the model to the problem of reads clustering using a quality value based alignment-free statistics, the approach is tested using standard measures (*e.g., recall*) and with its application to *de-novo* assembly and metagenomics binning; these results have been published in a joint work with Matteo Comin and Andrea Leoni ([CLS14, CLS15]).

In chapter 4 we will present a *rank filtering* approach where reads are sorted based on their quality, the main advantage of this technique is that all reads are still available to subsequent algorithms and, moreover, they are sorted so that algorithms can access the higher quality first and use the lower quality ones only if really needed. We then present preliminary results on such filtering approach applied to reads mapping and *de-novo* assembly.

1.5 Related works

The development of stochastic models for sequencing has been explored in several works, however none of them gives comprehensive description of a whole experiment nor they include quality values.

An approach very similar to the ours has been presented by Li *et al.* in [LRD08] where a software for read mapping called `MAQ` is presented. `MAQ` defines a probabilistic model for mapping based on quality values; it assigns a quality score to an entire read that generalizes the concept of quality value defined for single symbols. Although the model implemented in `MAQ` has some common ideas with the one we present in this thesis, there are major differences. First of all `MAQ` proposes a model for single reads rather than for an entire collection of reads, moreover `MAQ` considers only quality values associated with mismatches, while we will present a model that takes into account quality scores for all the called symbols. Finally `MAQ` has been developed with the specific goal of performing fast reads mapping using quality values while our model aims to be a more general framework that can be used to define different problems (including but not limited to mapping).

Another model similar to one presented in Chapter 2 is used in the *Genome Analysis ToolKit (GATK)* (McKenna *et al.* [MHB$^+$10]), where quality values are used to perform genotype inference using NGS and Bayesian inference.

Zhai *et al.* [ZRS+12] proposed a model for NGS data when the reference is given, the goal is to study the pattern distribution on sequencing data. They defined a simple model where the sampling of reads occurs accordingly to a probability distribution estimated from empirical data. The model proposed does not include quality values and is developed and tested only for estimation of pattern occurrence in NGS data.

Chapter 3 presents an application of our stochastic model to the problem of clustering of reads using alignment-free techniques. The D_2^q-type statistics presented are a generalization of the D_2-type statistics proposed by Reinert *et al.* [RCSW09] and by Wan *et al.* [WRSW10]. D_2-type were originally developed to measure dissimilarity between pair of sequences and only recently they have been extended by Song *et al.* [SRZ+13] to measure dissimilarity between pairs of sets containing NGS reads.

Centroid based clustering using alignment-free measures has been extensively tested by Solovyov and Lipkin [SL13] using `afcluster` which is the starting point of the `qCluster` software presented by Comin *et al.* [CLS14] and used to obtain the results of Chapter 3.

Predominant approaches to reads filtering partition the input set into two different subsets: one with *high* quality reads and the other with *low* quality ones. After the *pre-processing* step of filtering is performed, usually, the low quality set is discarded and the subsequent algorithms of the *pipeline* (*i.e., downstream* algorithms) use only high quality reads set.

Dohm *et al.* [DLBH07] proposed `SHARCGS` *de-novo* assembler which includes a preprocessing filter where reads are considered by the actual assembler only if a minimum length exact match is found. Li *et al.* [LZR+10] proposed `SOAP` *de-novo* assembler that filters reads based on observed k-mer frequencies. Some approaches to filtering make use of quality values, Sasson and Michael [SM10] apply complicated heuristics based on quality values to filter SOLiD reads, the idea is to use the qualities on the first part of a read (*e.g.,* the first 10 quality values) as *predictor* of the overall read quality. Petel and Jain [PJ12] developed `QC Tool` a toolkit that can be used to filter 454 and Illumina reads. Users can set the minimum percentage of a read that must have quality scores higher then a user defined value, afterwards the software performs several platform specific heuristics to trim and possibly discard reads. `MAQ` software [LRD08] can be used to filter reads based on quality values.

To the best of our knowledge, all approaches to filtering are *boolean* in the sense that reads are either high or low quality, therefore they substantially differ from the *rank filtering* approach which we will introduce in Chapter 4.

Chapter 2

Probabilistic model

Essentially, all models are wrong, but
some are useful
(George E. P. Box)

In this chapter we present the derivation of a probabilistic model for sequencing. Starting from the interpretation of quality values given by Ewing and Green [EG98], where qualities relates to the *error probability* of bases, we develop a model for the whole sequencing by means of successive generalization and with the support of proper hypotheses.

At the end of the chapter a simple application of this model is presented; more precisely we will present a form for the probability of observing a *correct read* or, equivalently, the probability for a read to be produced without sequencing errors. In chapters 3 and 4 this model is experimentally validated by applying it to the problems of *reads clustering* and *reads filtering* respectively.

Notation The derivation of a probabilistic model for a complex process, involves many equations and formula most of which represent intermediate results. To keep a clear presentation, we adopted a consistent notation throughout this entire chapter. We use the following conventions: Ω denotes a sample space, $\mathcal{F}$ the associated event space and P the probability function. To differentiate spaces, we will use subscripts, for example the sample space Ω_X will induce the event space:[1]

$$\mathcal{F}_X = \{e : e \subseteq \Omega_X\}$$

[1] All our spaces are finite

on which the probability function

$$P_X : \mathcal{F}_X \longrightarrow [0, 1]$$

is defined.

The formal notation for probability P_X can become really cumbersome, in order to maintain as clear presentation, we will often resort to the shortened notation

$$p_X(e_1, \ldots, e_n) := P_X(E_1 = e_1 \cap \ldots \cap E_n = e_n)$$

to represents the joint probability of the events $(E_i = e_i)$ with $e_i \in \mathcal{F}_X$. Consistent subscripts and usage of a lower p for such short version should guarantee that no confusion or ambiguity arise.

Since many spaces and parameters are used throughout the development of the model, it may help a quick reference to keep track of them; to this extent the reader may refer to Appendix A for the summary of conventional notation and meanings of variables commonly adopted in this chapter.

When referring to sequences and strings, we use the same notation adopted by Hopcroft *et al.* in [HMU06], more specifically

$$\Sigma^0 = \{\varepsilon\} \qquad \Sigma^N = \{s_1 \ldots s_N : s_j \in \Sigma\} \qquad \Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

where ε indicates the empty string. Substrings will be indicated using a subscript notation; that is, for a string $s_1 \ldots s_N$, substring from s_i and s_j $(i \leq j)$ included will be indicated with

$$s_{i,j} = s_i s_{i+1} \ldots s_{j-1} s_j.$$

When confusion may arise, sequences are indicated with bold fonts and single character with regular font, $\mathbf{s} = s_1 s_2 \ldots s_N$.

2.1 Sequencing process

Sequencing is the process of producing a collection of fragments called *reads* from a properly prepared genetic sample called *reference*. The physical processes used to obtain such fragments differ between sequencing technologies and has been briefly described in Chapter 1.

A sequencer machine takes as input a biologic sample of the reference S and

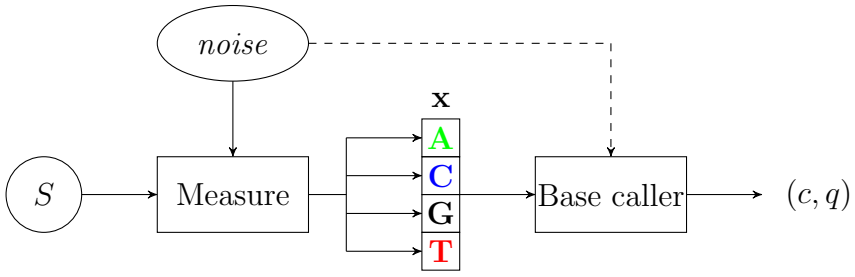


Figure 2.1: A simplified description of the physical process of producing the pair (c, q) of symbol quality from an input symbol S

produces as output a collection $\mathbf{R}$ of M reads.[2] Each read contains m pairs (c, q) with c a symbol over the alphabet Σ and q a quality score from the set $\mathcal{Q}$. As already discussed in section 1.3 Σ could be either the IUPAC alphabet (or some subset of it) or the set of four color, the set of quality $\mathcal{Q}$ is a set of non-negative integers $[0, q_{max}]$.

The reads produced are encoded into a `fastq` file for most of the sequencing technologies or in `csfasta` and `qual` files for SOLiD sequencers which uses *color space* encoding.

A simplified description of the sequencing for a single pair (c, q) of symbol $c \in \Sigma$ and quality score $q \in \mathcal{Q}$ is schematically depicted in Figure 2.1. Since sequencers produce reads instead of single symbols as output of a single fragment sequencing, the process described in Figure 2.1 introduces a slight simplification that helps understanding the overall process.

A sequence $S = s$ containing only the symbol $s \in \Sigma$ is measured, details of such measurement are beyond the scope of this thesis, but we can assume that physical measurements (*e.g.,* intensities at predefined wavelength) are numerical encoded into a vector $\mathbf{x}$. Once the vector $\mathbf{x}$ is completely filled the sequencer has, strictly speaking, concluded its operation on s and, moves to next *position* to possibly sequence the next symbol.

The production of the pair (c, q) from the vector $\mathbf{x}$ is performed by a software component named *base caller* and included in all sequencers. The base caller takes as input $\mathbf{x}$ and computes c and q as functions of $\mathbf{x}$: $\sigma(\mathbf{x})$ and $\pi(\mathbf{x})$ respectively.

For example in [EHWG98] Ewing *et al.* describe in detail the software *phred* and the algorithm to compute $\sigma(\mathbf{x})$ and in [EG98] Ewing and Green describe the algorithm to compute $\pi(\mathbf{x})$. In these works authors use raw traces outputted by sequencer to construct what they call the *parameters vector* $\mathbf{x}$ which is then used to compute $c = \sigma(\mathbf{x})$ and $q = \pi(\mathbf{x})$. Another example of base caller is included in PacBio sequencers and uses `GATK` software (McKenna *et al.* [MHB+10]) to compute

[2] We will often refer to $\mathbf{R}$ as either *reads set* or *input dataset*, note, however, data $\mathbf{R}$ is not a set in the mathematical sense of the term.

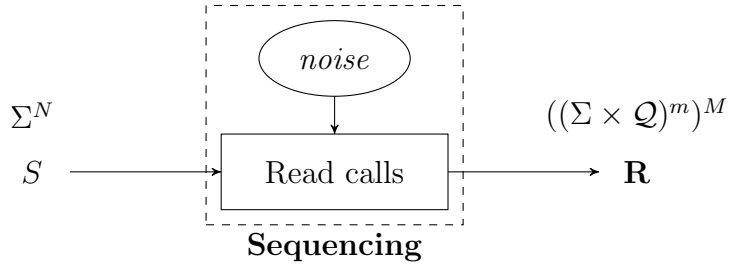


Figure 2.2: A schematic representation of the sequencing process, the input is a a sample reference S and the output is a collection $\mathbf{R}$ of reads.

$\sigma(\mathbf{x})$ and $\pi(\mathbf{x})$.

A real sequencing experiment produces M reads from the a sample $S = s_1 \dots s_N$ containing N symbols $s_j \in \Sigma$. Figure 2.2 gives a schematic representation of the input output relation of the sequencing process. That is, the sample sequence S defined over the set Σ^N is properly prepared and inputted to the sequencer which produces M reads; all reads, each containing m pairs (c, q), are inserted into an *unordered* collection $\mathbf{R}$ which is encoded and written in the output file(s) and comes from the space $((\Sigma \times \mathcal{Q})^m)^M$.

2.1.1 Quality values

A fundamental aspect of our model is the usage of quality values; we will use the definition of quality scores given in [EG98] to build a probabilistic model of the sequencing. This approach differs from the ones commonly adopted where qualities are used to define heuristics assessing the quality of sequenced data (*e.g.,* `MAQ` [LRD08]).

The phred software was introduced in [EHWG98] as an alternative to the ABI base caller, a remarkable innovation it introduces is represented by the unique calculation and encoding of quality scores. The idea is to use tracing data to compute a parameter vector $\mathbf{x}$ from which is then computed a measurement of the likelihood of a base to be incorrectly sequenced.

The space where measures $\mathbf{x}$ are defined is partitioned into 50 *cuts* each of which is associated to a particular error rate r. In [EG98] authors experimentally showed that the function $\pi(\mathbf{x})$ implemented by phred is a good estimator the real rate r defined as the number of incorrect bases divided by the total number of sequenced bases.

To better represent the domain of *low* error rate base calls, phred uses a non linear (*i.e.,* logarithmic) scale for the representation of r. More precisely for a given

error rate r, the corresponding score is given by

$$q_r = [-10 \log_{10}(r)]_{int} \tag{2.1}$$

where $[\cdot]_{int}$ represents the rounding to the nearest integer.

2.2 Probabilistic model

The first step toward the development of a stochastic model is the definition of the sample space associated to the sequencing process. In Section 2.1 we described the process in terms of an input reference S and the output collection of reads $\mathbf{R}$, Figure 2.2 also shows associated domains for input and output which we will now formally introduced as the probability spaces.

2.2.1 Reference sequence

The reference S is a sequence of symbols $s_1 s_2 \dots s_N$ each coming from the alphabet Σ. For given N the set of all sequences with length N is

$$\Omega_{\mathcal{G}} = \underbrace{\Sigma \times \Sigma \times \dots \times \Sigma}_{N \text{ times}} = \Sigma^N$$

which is also the sample space for the input reference S. Elements of the space $\Omega_{\mathcal{G}}$ are ordered N-tuples $(s_1, s_2, \dots, s_N)$ that can be put in one-to-one correspondence with strings in Σ^N.

There are many reasons for representing reference S as a random variable. First, in most of the cases, this sequence is unknown (*e.g., de novo* assembly) or known only partially (*e.g.,* comparative assembly). Second the model we present in this thesis stochastically describes relation between the input reference S and the output collection $\mathbf{R}$, to give this model the higher possible flexibility, a stochastic description of both input and output is the best choice. To keep the model simple, however, some aspects of both input S and output $\mathbf{R}$ must be deterministically defined. In particular one fundamental assumption is that the length of the reference $|S| = N$ is a finite and deterministic quantity.

Hypothesis 2.1 (Deterministic Reference Length)**.** *The length $|S| = N$ of the reference sequence S is known and does not stochastically vary, moreover $N < \infty$.*

With this hypothesis the set $\Omega_{\mathcal{G}}$ is deterministically defined and Since finite, therefore the probability $P_{\mathcal{G}}(S = \mathbf{s})$ for all $\mathbf{s} \in \Sigma^N$ are a sufficient do characterize

the probability function $P_{\mathcal{G}}(E = e)$ for every possible $e \in \mathcal{F}_{\mathcal{G}}$ (Papoulis and Pillai [PP02]).

Since we will focus on modeling sequencing process, we will assume that the probability distribution function $P_{\mathcal{G}}(S = \mathbf{s}) = p_{\mathcal{G}}(\mathbf{s})$ is given for each $\mathbf{s} \in \Sigma^N$. The probability $p_{\mathcal{G}}$ is sometimes referred to as *prior probability* for the sequence $\mathbf{s}$. For several (more or less simple) models the reader can consult few bibliographic references: Churchill [Chu89], Durbin [Dur98].

2.2.2 Sample space for sequencing

As described in Section 2.1, the output of the sequencing processing is a collection $\mathbf{R}$ of M reads

$$(\mathbf{c}, \mathbf{q})_1 (\mathbf{c}, \mathbf{q})_2 \dots (\mathbf{c}, \mathbf{q})_M.$$

Each read $(\mathbf{c}, \mathbf{q})_h$ is a pair of vectors $\mathbf{c}_h = c_{h,1} c_{h,2} \dots c_{h,m}$ and $\mathbf{q}_h = q_{h,1} q_{h,2} \dots q_{h,m}$ with $c_{h,\ell}$ being symbols in Σ and $q_{h,\ell}$ being qualities in $\mathcal{Q}$. The sample space for M reads of length m is defined as

$$\Omega_{\mathcal{R}} = \left[(\Sigma \times \mathcal{Q})^m \right]^M .$$

which assumes that the following hypothesis holds.

Hypothesis 2.2 (Deterministic Constant Read Length)**.** *The length m of a read $(\mathbf{c}, \mathbf{q})$ is known and does not stochastically vary, moreover, for a set $\mathbf{R}$ of M reads, m is the length of all of them.*

Also in this case the hypothesis of m not being a random variable is assumed to keep the model tractable.

Combining $\Omega_{\mathcal{G}}$ and $\Omega_{\mathcal{R}}$ Once the spaces for reference S and reads $\mathbf{R}$ are defined, we can combine them into a single one

$$\Omega_{\mathcal{G},\mathcal{R}} = \Omega_{\mathcal{G}} \times \Omega_{\mathcal{R}} = (\Sigma^N) \times \left((\Sigma \times \mathcal{Q})^m \right)^M \tag{2.2}$$

which is the space where events describing sequencing experiments are defined. That is, if a reference $S = \mathbf{s}$ is inputted to the sequencer and this produces the collection of reads $\mathcal{R} = \mathbf{R}$, the corresponding of $\mathcal{F}_{\mathcal{G},\mathcal{R}}$ is $(S = \mathbf{s} \cap \mathcal{R} = \mathbf{R})$.

Although this event is well defined, it is more interesting and useful to look at the probability of the event $(S = \mathbf{s} \mid \mathcal{R} = \mathbf{R})$. This is the event of S being the sequence $\mathbf{s}$, conditioned by the fact that the sequencer produced the $\mathbf{R}$ when $\mathbf{s}$ is

given as input. More generally, we will put more emphasis to the probabilities

$$P(\text{reference} \mid \text{reads})$$

characterizing *the probability of a reference given the reads*. The idea is that, being process of producing $\mathbf{R}$ dependent from the input $\mathbf{s}$, our model describes such dependency in terms of conditional probability. For example let say that we want to find the most likely (*i.e.,* with highest probability) sequence $\mathbf{s}^*$ given a set of reads $\mathbf{R}$. This problem has a straightforward description in terms of events in the space $\Omega_{\mathcal{G},\mathcal{R}}$

$$\mathbf{s}^* = \arg\max_{\mathbf{s}\in\Sigma^N} P\left(S = \mathbf{s} \mid \mathcal{R} = \mathbf{R}\right).$$

This is a simple way of using our model to define the problem of assembly. Note that, for a given length N of $\mathbf{s}^*$, the candidate assemblies are well defined and can be computed with a *naïve* with time exponential in N. This implementation simply enumerates all the 4^N sequences $\mathbf{s}$ and computes $P(S = \mathbf{s} \mid \mathcal{R} = \mathbf{R})$ for all of them. A preliminary work on *de novo* assembly with this model has been carried out by Baruzzo in [Bar13].

2.2.3 Single base call

We start developing our stochastic model from the probabilistic interpretation of quality scores given in [EG98] and briefly described in Section 2.1.

The process of producing a pair (c, q) from the sequence $S = s$ is defined on the space

$$\Omega_{\mathcal{G},\mathcal{C}} = \Sigma \times (\Sigma \times \mathcal{Q})$$

equivalent to (2.2) when $N, M, m = 1$. For convenience we defined $\Omega_{\mathcal{C}} = \Sigma \times \mathcal{Q}$, this is the space representing all possible pairs $(C = c \ \cap \ Q = q)$. Events on $\Omega_{\mathcal{G},\mathcal{C}}$ have the form

$$(C = c \ \cap \ Q = q \ \cap \ S = s)$$

corresponding to the sequencer producing the symbol $C = c$ with an associated quality $Q = q$ when the input is the sequence $S = s$.

In [EG98] quality values are related to the probability of correct symbol according to equation (2.1), in terms of the space just introduced this probability becomes:

$$P_e(q) := P(C \neq s \mid Q = q \cap S = s) = 10^{-q/10}. \tag{2.3}$$

sometimes referred to as *phred function* for brevity. Equation (2.3) formalizes, in

terms of the space $\Omega_{\mathcal{G},\mathcal{C}}$ the commonly accepted interpretation of the quality score $Q = q$ as the probability of the corresponding bases $C = c$ being wrong ($c \neq s$) when the symbol $S = s$ is sequenced.

Equation (2.3) is not sufficient to describe the probability function $P_{\mathcal{G},\mathcal{C}}$, because the probabilities of events $(C = c \mid Q = q \cap S = s)$ when $c \neq s$ are not specified. To fill this gap we assume that the *error probability* $P_e(q)$ is uniformly split among all the bases $c \neq s$ formally this assumption is given in the following hypothesis.

Hypothesis 2.3 (Uniform Error Probability Model)**.** *The probability of the reference sample S being the actual sequence s given the output of the sequencer (c, q) is*

$$P_{\mathcal{G},\mathcal{C}}(C = c \mid Q = q \cap S = s) = \begin{cases} 1 - P_e(q) & s = c \\ \dfrac{P_e(q)}{|\Sigma| - 1} & s \neq c \end{cases} . \tag{2.4}$$

Where $P_e(q)$ is the phred function defined in (2.3).

Note that, assuming Hypothesis 2.3, the function

$$p_e^U(c, q, s) := P_{\mathcal{G},\mathcal{C}}(C = c \mid Q = q \cap S = s)$$

defines a proper probability space

$$\sum_{c \in \Sigma} P_{\mathcal{G},\mathcal{C}}(C = c \mid Q = q \cap S = s) = \sum_{c \in \Sigma} p_e^U(c, q, s) = 1$$

which is the conditioned space with $(Q = q \cap S = s)$ being the conditioning event.

Let consider, for example, a machine producing the pair $(A, 20)$

$$P(C = A \mid Q = 20 \cap S = A) = 0.99$$

which says that the sequencing receiving the symbol $S = A$ and observing a *reading interference* components corresponding to quality $q = 20$ has a 99% chance of outputting the symbol $C = A$ while has $0.01/3$ chance of outputting each of the other 3 symbols C, G and T.

Hypothesis 2.3 reflects the lack of information about how sequencing errors are distributed; this information is sometimes heuristically incorporated but often it is just ignored. Future sequencers may be able to supply more detailed information about such events, we may also use empirical error distribution to slightly modify Hypothesis 2.3, in all these cases our model still remains valid and useful.

A notable exception to this is represented by insertions and deletions which are

not considered by our model (in fact it is not even possible to define them in the probability space $\Omega_{\mathcal{G},\mathcal{C}}$as it is now). One of the future extensions to our stochastic model should tackle this problem especially as future generation sequencers like PacBio start to gain popularity.

So far we gave a probabilistic description of events conditioned by the measured quality q and the input s. Most of the bioinformatics problems, however, aim to characterize S in terms of $\mathbf{R}$, therefore a more interesting probability is

$$p_{\mathcal{G}|\mathcal{C}}(c,q,s) := P_{\mathcal{G},\mathcal{C}}(S = s \mid C = c \cap Q = q) \tag{2.5}$$

which, with a simple application of the Bayes' theorem, can be turned into

$$\begin{aligned} p_{\mathcal{G}|\mathcal{C}}(c,q,s) &= \frac{P_{\mathcal{G},\mathcal{C}}(Q = q \cap S = s)}{P_{\mathcal{G},\mathcal{C}}(C = c \cap Q = q)} P_{\mathcal{G},\mathcal{C}}(C = c \mid Q = q \cap S = s) \\ &= \frac{p_{\mathcal{G},\mathcal{C}}(s,q)}{p_{\mathcal{G},\mathcal{C}}(c,q)} p_e^U(c,q,s). \end{aligned}$$

As we see, our model requires the knowledge of the marginal distributions $p_{\mathcal{G},\mathcal{C}}(s,q)$ and $p_{\mathcal{G},\mathcal{C}}(c,q)$. A possible simplification for $p_{\mathcal{G},\mathcal{C}}$ comes from the following hypothesis.

Hypothesis 2.4 (Quality sequence independence)**.** *The events $S = s$ and $Q = q$ are statistically independent on the space $\Omega_{\mathcal{G},\mathcal{C}}$*

$$P_{\mathcal{G},\mathcal{C}}(Q = q \cap S = s) = p_{\mathcal{Q}}(q)\ p_{\mathcal{G}}(s).$$

This hypothesis reflects the, reasonable, assumption that the sequencing machine, while calculating the quality score of a given position, does not take into account the actual value of the base. Another way of interpreting this assumption is that the machine does not change its prior distribution of quality scores $p_{\mathcal{Q}}(q)$ based on the current symbol s.[3] Note that for the phred software this hypothesis has been experimentally validated, in fact in [EG98] and [EHWG98] authors emphasize that all quality schemes should satisfy Hypothesis 2.4 as a property they call *predictivity*.

Using Hypothesis 2.4, we have

$$p_{\mathcal{G}|\mathcal{C}}(c,q,s) = \frac{p_{\mathcal{G}}(s)\ p_{\mathcal{Q}}(q)}{p_{\mathcal{C}}(c,q)} p_e^U(c,q,s) \tag{2.6}$$

[3]When events $S = s$ and $Q = q$ are independent we have

$$P_{\mathcal{G},\mathcal{C}}(Q = q \mid S = s) = P_{\mathcal{G},\mathcal{Q}}(Q = q).$$

As one would expect assuming a independent and identically distributed (i.i.d.) for prior distribution of all marginal probabilities $P_{\mathcal{G}}$, $P_{\mathcal{Q}}$ and $P_{\mathcal{C}}$, allows to simplify Equation (2.6) leaving the only $p_e^U(c, q, s)$. However, if we have some prior knowledge, such distributions can be adjusted to it; as an extreme example consider the situation where we know that $s \neq A$, then $p_{\mathcal{G}}(A) = 0$ and $p_{\mathcal{G}|\mathcal{C}}(c, q, A) = 0$.

This example shows the ability of our model to properly describe events based on the prior knowledge we have on the sequencing process.

As a final remark note that event (c, q) must have prior $p_{\mathcal{C}}(c, q) \neq 0$ otherwise Equation (2.6) is not defined. Practically this means that every pair (c, q) produced by the sequencer is a plausible output of the machine. Note that if the input dataset is used as an estimator for the distribution $p_{\mathcal{C}}(c, q)$, then this consistency is guaranteed to be satisfied by definition.

Let us shortly come back to the simple example of the pair $(A, 20)$, let us assume that the i.i.d. model holds so that $p_{\mathcal{G}|\mathcal{C}}(A, 20, s) = p_e^U(A, 20, s)$ this will give a probability of $S = s$ for each possible $s \in \{A, C, G, T\}$ and, once this is known, we could solve the problem of the maximum probability sequence described above

$$s^* = \underset{S \in \{A,C,G,T\}}{\arg\max}\ p_e^U(A, 20, s) = A$$

with probability

$$P(S = s^* \mid C = A \cap Q = 20) = 0.99\ .$$

Entropy of quality values

Chapters 3 and 4 present experimental validation of the model presented in this chapter, to provide a different justification for usage of quality values, we propose in this paragraph an information theoretic interpretation of quality values. Our aim is to quantify the information conveyed by a dataset when: quality scores are given for each sequenced symbol and when qualities are not given or, in other words, to quantify the information of quality scores. In both cases we will suppose that the prior distribution of quality scores is known.

The measures we are going to derive here are based on the concept of *entropy* as defined by Shannon in [Sha48] and here briefly recalled.

Definition 2.5 (Entropy)**.** *Given a discrete probability distribution function $p_X(x)$ over the space Ω_X the* entropy H_X *(or* Shannon entropy*) is*

$$H_X = -\sum_{x \in \Omega_X} p_X(x) \log_2 p_X(x) \tag{2.7}$$

where it is conventionally assumed that $0 \log_2 0 = 0$.[4]

Informally the entropy H_X measures the *uncertainty* of a random process that generates symbol of Ω_X with probability distribution p_X. Entropy can also be interpreted as the minimum number of bits needed to code outcomes of the process. A more detailed discussion on entropy and its properties is given in Shannon [Sha48] and Cover and Thomas [CT06].

We use the entropy to measure the uncertainty contained on a symbol call performed by a sequencer; to this extent we compute two different entropies. The first one is the average entropy H_Σ when quality value q is given and the second is the average entropy $\widetilde{H}_\Sigma$ when qualities are not known. In both cases we average using the prior probability $p_\mathcal{Q}$ of quality scores; we will use Equation (2.4) for the probability $p_\Sigma(c)$ of symbols where $P_e(q)$ is the usual phred function (2.3)

To compute H_Σ we first need to calculate the entropy $H_\Sigma(q)$ representing the entropy of a pair (c, q); as observed in [Sha48] the entropy is not dependent from the actual symbol c nor it depends from the the sequenced symbol $S = s$.

$$\begin{aligned} H_\Sigma(q) &= -\sum_{c \in \Sigma} p_\Sigma(c) \log_2 \left(p_\Sigma(c)\right) \\ &= -(1 - P_e(q)) \log_2 \left(1 - P_e(q)\right) - P_e(q) \log_2 \frac{P_e(q)}{(|\Sigma| - 1)}. \end{aligned}$$

For example for $q = 20$ corresponding to a probability $P_e(q) = 0.01$ the entropy $H(q) \approx 0.097$; for $q = 5$ (which is considered very low quality) $H(q) \approx 1.401$. The maximum entropy is attained for p_Σ uniform [CT06]; in our model this corresponds to $P_e(q) = 3/4$ and (according to Equation (2.1)) $q \approx 1.25$. This shows that high values of entropy (*i.e.,* close to the maximum) arise only for low qualities, this trend is further amplified by the non linear nature of the phred scale. Usually real datasets have distribution with average quality score between 25 and 35, the number of bits we would ignore if we discarded quality scores should be relatively small, however we expect the combined effect of all of them to be significant.

The entropy H_Σ is the obtained as the average $H_\Sigma(q)$ over weighted over all qualities $q \in \mathcal{Q}$:

$$H = \sum_{q \in \mathcal{Q}} p_\mathcal{Q}(q) H_\Sigma(q).$$

In the case where qualities q are not given , but the prior distribution is $p_\mathcal{Q}(q)$

[4] This can be formalized with a continuity argument.

Dataset	H_Σ	$\widetilde{H}_\Sigma$	ΔH_Σ
`SRR023794` *(H. pylori)*	0.1184	0.1865	0.0681
`SRR017901` *(Z. mobilis)*	0.0243	0.0359	0.0115
`ERR164429` *(L. pneumophila)*	0.1276	0.2917	0.1642
`SRR959247` *(E. coli)*	0.0352	0.0955	0.0604
`Mason` *($P = 0.01$ $M = 10^5$)*	0.0031	0.0319	0.0281

Table 2.1: Entropy of single symbol calculated when qualities are known H_Σ and when qualities are not given $\widetilde{H}_\Sigma$ for all the different datasets used throughout the thesis.

known; we calculate the *average error probability*

$$\widetilde{P}_e = \sum_{q \in \mathcal{Q}} p_{\mathcal{Q}}(q) P_e(q)$$

and use this to compute the entropy

$$\widetilde{H}_\Sigma = -(1 - \widetilde{P}_e) \log_2 (1 - \widetilde{P}_e) - \widetilde{P}_2 \log_2 \frac{\widetilde{P}_e}{|\Sigma| - 1}$$

To estimate the amount of information that quality scores bring when they are given we simply use the difference between H_Σ and $\widetilde{H}_\Sigma$:

$$\Delta H_\Sigma = |\widetilde{H}_\Sigma - H_\Sigma|.$$

Entropy for real datasets We computed H_Σ and $\widetilde{H}_\Sigma$ for many of the datasets used throughout the chapters 3 and 4, results are presented in Table 2.1. We used $|\Sigma| = 4$, Equation (2.4) for p_Σ and $p_{\mathcal{Q}}(q)$ estimated from the dataset itself. That is, for a given dataset $\mathbf{R}$ with M reads $(\mathbf{c}, \mathbf{q})_h$ each with length m, let $occ(q, \mathbf{q})$ be the number of values q in the scores vector $\mathbf{q}$, the prior distribution is defined as

$$p_{\mathcal{Q}}(q) = \frac{\sum\limits_{(\mathbf{c},\mathbf{q}) \in \mathbf{R}} occ(q, \mathbf{q})}{Mm}$$

or, in other words, $p_{\mathcal{Q}}(q)$ is the frequency of q observed in the dataset $\mathbf{R}$.

2.2.4 Single read

In this section we model the experiment of producing one m pairs read (*i.e.,* $M = 1$) from sequencing the reference S containing $m \geq 1$ total positions (*i.e.,* $N = m$), extension to general N will be given at the end of this section. Each position of the

read contains a pair (c_ℓ, q_ℓ), $\ell \in [1, m]$, the read is represented as a pair of vectors $(\mathbf{c}, \mathbf{q})$ with $\mathbf{c} = c_1 \dots c_m$ and $\mathbf{q} = q_1 \dots q_m$. The reference is a sequence $\mathbf{s} = s_1 \dots s_m$ of m symbols from the alphabet Σ.

The probability space (2.2) becomes $(N = m, M = 1)$

$$\Omega_{\mathcal{G},r} = \Sigma^m \times (\Sigma \times \mathcal{Q})^m$$

where we look at the probability

$$\begin{aligned} p_{\mathcal{G}r}(\mathbf{c}, \mathbf{q}, \mathbf{s}) &= P_{\mathcal{G},r}(S = \mathbf{s} \mid C = \mathbf{c} \cap Q = \mathbf{q}) \\ &= \frac{P_{\mathcal{G},r}(C = \mathbf{c} \cap Q = \mathbf{q} \mid S = \mathbf{s}) p_{\mathcal{G},r}(S = \mathbf{s})}{P_{\mathcal{G},r}(C = \mathbf{c} \cap Q = \mathbf{q})} \\ &= \frac{p_{\mathcal{G}}(\mathbf{s})}{p_r(\mathbf{c}, \mathbf{q})} P_{\mathcal{G},r}(C = \mathbf{c} \cap Q = \mathbf{q} \mid S = \mathbf{s}) \end{aligned}$$

We now introduce two hypotheses which allow us to greatly simplify the derivation of a closed form for $p_{\mathcal{G}|r}$. The first assumes that two different pairs are independently sequenced, given the reference sequence; a similar assumption has also been done in `MAQ` [LRD08]. The second hypothesis assumes that a given pair is sequenced independently from all the positions other than the current one.

Hypothesis 2.6 (Symbol conditional independence)**.** *Given a read* $(\mathbf{c}, \mathbf{q})$ *of length* m*, the events of sequencing the two pairs* (c_i, q_i) *and* (c_j, q_j) *are statistically independent for* $i \neq j$*.*

$$\begin{aligned} &P_{\mathcal{G},r}((C_i = c_i \cap Q_i = q_i) \cap (C_j = c_j \cap Q_j = q_j)) \\ &\quad = P_{\mathcal{G},r}(C_i = c_i \cap Q_i = q_i) P_{\mathcal{G},r}(C_j = c_j \cap Q_j = q_j) \end{aligned}$$

Hypothesis 2.7 (Local Sequencing)**.** *The production of the pair* (c_ℓ, q_ℓ) *from position* j *of the reference sequence* S *is statistically independent from all symbols* s_i *with* $i \neq j$*:*

$$P_{\mathcal{G},r}(C_\ell = c_\ell \cap Q_\ell = q_\ell \mid S = \mathbf{s}) = P_{\mathcal{G},r}(C_\ell = c_\ell \cap Q_\ell = q_\ell \mid S_j = s_j)$$

As a consequence of Hypothesis 2.6 we can write

$$P_{\mathcal{G},r}(C = \mathbf{c} \cap Q = \mathbf{q} \mid S = \mathbf{s}) = \prod_{\ell=1}^{m} P_{\mathcal{G},r}(C_\ell = c_\ell \cap Q_\ell = q_\ell \mid S = \mathbf{s}).$$

and, combining it with Hypothesis 2.7 we get

$$p_{\mathcal{G}|r}(\mathbf{c},\mathbf{q},\mathbf{s}) = \frac{p_{\mathcal{G}}(\mathbf{s})}{p_r(\mathbf{c},\mathbf{q})}\prod_{\ell=1}^{m}\frac{p_{\mathcal{C}}(c_\ell,q_\ell)}{p_{\mathcal{G}}(s_\ell)}p_{\mathcal{G}|\mathcal{C}}(c_\ell,q_\ell,s_\ell). \tag{2.8}$$

Note that when i.i.d. model is assumed for all marginal distributions $p_{\mathcal{G}}$, $p_{\mathcal{C}}$ and p_r and the uniform model of p_e^U (Hypothesis 2.3) is used for $p_{\mathcal{G}|\mathcal{C}}$ Equation (2.8) becomes

$$p^{iid}_{\mathcal{G}|r}(\mathbf{c},\mathbf{q},\mathbf{s}) = \prod_{\ell=1}^{m} p_e^U(c_\ell,q_\ell,s_\ell) \tag{2.9}$$

which is extensively used in the next chapters and briefly discussed in Section 2.3 at the end of this chapter.

General N

Sequencers are not able to reproduce a copy of the entire input sequence S into one single read; they can only generate fragments with length $m \ll N$.

A model can not ignore this crucial aspect of the sequencing and this paragraph, first describe and then include into what developed so fa, the process of *positioning*.

There are many factors that influence the position where sequencers perform the actual reading operations. Some of these are: *Polymerase Chain Reaction (PCR)* amplification, *DNA cloning*, *library preparation*. In general different technologies have different limitations that induce biases in the sequencing process. Keeping track of the all possible aspects related to positioning is not easy, we try here to give a model flexible enough to be used in as many cases as possible.

Let start by defining the space for *positioning* as the set of any possible position j of the reference S. For reads with length m, we will consider valid positions only the ones in the interval $\Omega_{pos} = [1, \bar{N}]$, where $\bar{N} = N - m + 1$. With this definition we avoid limit cases where either reads are shorter than m bases (which would violate Hypothesis 2.2) because sequenced at the edges of S or reads come from *fictitious* reference because of reading process exceeding the actual sequence. A possible exception (which will not be considered here) is represented by prokaryotes DNA where the reference sequence *wraps around* itself and S is a *circular sequence*, in this case the space of allowed positions is represented by any of the position of S, $[1, N]$.

A probability space where both reads and positioning are modeled as a random

variables is defined by[5]

$$\Omega_{\mathcal{G},\mathcal{P}} := \Sigma^N \times [1, \bar{N}] \times (\Sigma \times \mathcal{Q})^m.$$

Let now consider the event

$$(C = \mathbf{c} \cap Q = \mathbf{q} \mid S = \mathbf{s} \cap J = j)$$

which represents the production of read $(\mathbf{c}, \mathbf{q})$ once the probe is attached to position $J = j$ of the reference sequence S. In principle the sequencing operation is not limited to the only positions $S_j, \ldots, S_{j+m-1}$, however with the goal of keeping the model simple we now introduce a hypothesis that restrict the sequencing process to be dependent from only the actual sequenced positions. The following hypothesis is similar to Hypothesis 2.7 and, in some sense, extends to the whole read the concept of *local sequencing*. 2.7).

Hypothesis 2.8 (Read Local Sequencing)**.** *For a given position $j \in [1, \bar{N}]$ the sequencing of a m symbols read $(\mathbf{c}, \mathbf{q})$ from j, only depends only from symbols of S at positions $j, j+1, \ldots, j+m-1$.*

$$\begin{aligned} &P_{\mathcal{G},\mathcal{P}}(C = \mathbf{c} \cap Q = \mathbf{q} \mid S = \mathbf{s} \cap J = j) \\ =&P_{\mathcal{G},\mathcal{P}}(C = \mathbf{c} \cap Q = \mathbf{q} \mid S_{j,\ldots,j+m-1} = \mathbf{s}_{j,\ldots,j+m-1}) \end{aligned} \tag{2.10}$$

Note that the right hand side of (2.10) is equivalent to (2.8) except for the probability space on which it is defined. What Hypothesis 2.8 is stating is that the sample space $\Omega_{\mathcal{G},\mathcal{P}}$, conditioned to the event $J = j$ (sequencer positioning the probe on S_j), is equivalent to the event of a sequencer having to produce one read of length m from the m long sequence $S_j \ldots S_{j+m-1}$.

Unfortunately when a sequencer produces the read $(\mathbf{c}, \mathbf{q})$ does not give any clue about the position j, when considering a single read, therefore, we must assume that every possible position $j \in [1, \bar{N}]$ could be the origin of the sequencing, in other words the event $(C = \mathbf{c} \cap Q = \mathbf{q} \cap S = \mathbf{s})$ is the marginal distribution of $P_{\mathcal{G},\mathcal{P}}$ over

[5]The equation implicitly defines the space

$$\Omega_{\mathcal{P}} = \Omega_{pos} \times \Omega_r = [1, \bar{N}] \times (\Sigma \times \mathcal{Q})^m$$

.

all the possible positions j. This gives

$$\begin{aligned} p_{\mathcal{G},\mathcal{P}}(\mathbf{c},\mathbf{q},\mathbf{s}) &= \sum_{j=1}^{\bar{N}} p_{\mathcal{G},\mathcal{P}}(\mathbf{c},\mathbf{q},\mathbf{s}|j)p_{\mathcal{G},\mathcal{P}}(j) \\ &= \sum_{j=1}^{\bar{N}} p_{\mathcal{G},r}(\mathbf{c},\mathbf{q},\mathbf{s}_{j,\dots,j+m-1})p_{pos}(j). \end{aligned} \tag{2.11}$$

2.2.5 Set of reads

We now extend our model to describe the entire sequencing; that is, the production of M reads $(\mathbf{c},\mathbf{q})_h$, $h = 1,\dots,M$ from a reference sequence $\mathbf{s}$. We will suppose constant size m for all the reads (see Hypothesis 2.2), the sample space is the one defined in Equation (2.2)

$$\Omega_{\mathcal{G},\mathcal{R}} = \Sigma^N \times ((\Sigma \times \mathcal{Q})^m)^M .$$

For notational reasons we define the event of producing M reads as

$$\mathcal{R} := \{(C,Q)_h : h = 1,\dots,M\},$$

the collection of reads as

$$\mathbf{R} = \{(\mathbf{c},\mathbf{q})_h : h = 1,\dots,M\}$$

so that we can write the probability of producing M reads as

$$P_{\mathcal{G},\mathcal{R}}\left(\bigcap_{h=1}^{M}(C_h = \mathbf{c}_h \cap Q_h = \mathbf{q}_h)\right) = P_{\mathcal{G},\mathcal{R}}(\mathcal{R} = \mathbf{R})$$

These events can represent the input-output relation of sequencing process as described in Section 2.1 and depicted in Figure 2.2. In the space $\Omega_{\mathcal{G},\mathcal{R}}$ we concentrate our attention to the probability

$$p_{\mathcal{G}|\mathcal{R}}(\mathbf{R},\mathbf{s}) := P_{\mathcal{G},\mathcal{R}}(S = \mathbf{s} \mid \mathcal{R} = \mathbf{R}) = \frac{P_{\mathcal{G},\mathcal{R}}(\mathcal{R} = \mathbf{R} \mid S = \mathbf{s})P_{\mathcal{G},\mathcal{R}}(S = \mathbf{s})}{P_{\mathcal{G},\mathcal{R}}(\mathcal{R} = \mathbf{R})}$$

which is the probability of the *input* sequence S being equal to $\mathbf{s} = s_1 \dots s_N$, given that the output reads collection is $\mathbf{R} = (\mathbf{c},\mathbf{q})_1 \dots (\mathbf{c},\mathbf{q})_M$.

Similarly to what done for the sequencing of m symbols (see Hypothesis 2.6) we assume the following hypothesis.

Hypothesis 2.9 (Read Conditional Independence)**.** *The events of producing two reads* $(\mathbf{c},\mathbf{q})_i$ *and* $(\mathbf{c},\mathbf{q})_h$ *are statistically independent given the reference sequence* $S=\mathbf{s}$.

This assumption allows us to simplify $p_{\mathcal{G}|\mathcal{R}}$,

$$p_{\mathcal{G}|\mathcal{R}}(\mathbf{R},\mathbf{s})=\prod_{h=1}^{M}\frac{P_{\mathcal{G},\mathcal{R}}((C,Q)_h=(\mathbf{c},\mathbf{q})_h \mid S=\mathbf{s})P_{\mathcal{G},\mathcal{R}}(S=\mathbf{s})}{P_{\mathcal{G},\mathcal{R}}(\mathcal{R}=\mathbf{R})}$$

which can be expanded using all results of previous sections since

$$P_{\mathcal{G},\mathcal{R}}((C,Q)_h=(\mathbf{c},\mathbf{q})_h \mid S=\mathbf{s})=p_{\mathcal{G}|r}(\mathbf{c}_h,\mathbf{q}_h,\mathbf{s}).$$

In other words, the probability of the event $(C,Q)_h=(\mathbf{c},\mathbf{q})_h$ (*i.e.,* producing read $(\mathbf{c},\mathbf{q})$) of the space $\Omega_{\mathcal{G},\mathcal{R}}$ given the reference sequence $S=\mathbf{s}$ can be viewed as the restriction to the case with $M=1$ and can be explicitly obtained using equations (2.8) and (2.11).

A final adjustment needs to be done before giving the final form of $p_{\mathcal{G}|\mathcal{R}}$. The above probability describes the event of observing the following *ordered sequence* of reads

$$((\mathbf{c},\mathbf{q})_1,(\mathbf{c},\mathbf{q})_2,\ldots,(\mathbf{c},\mathbf{q})_M)$$

however we are interesting in the *unordered list* [6]

$$\langle(\mathbf{c},\mathbf{q})_1,(\mathbf{c},\mathbf{q})_2,\ldots,(\mathbf{c},\mathbf{q})_M\rangle\,.$$

It easy to prove that we need to adjust the probability above by using the *multinomial coefficient*

$$\binom{M}{\mu_1,\mu_1,\ldots,\mu_M}=\frac{M!}{\mu_1!\mu_2!\cdots\mu_M!}$$

where μ_h represent the number of occurrences of the read $(\mathbf{c},\mathbf{q})_h$ in $\mathbf{r}$. In most cases this correction factor can be approximated with $M!$ because the chance of observing two identical reads (*i.e.,* containing the same symbols sequence $\mathbf{c}$ and the same qualities sequence $\mathbf{q}$) is negligible (and decreases as m increases), therefore, the terms μ_h, usually, are all equal to 1.

[6] Although sequencers usually give reads as numbered sequences, such numbering is only a convenient index.

We can finally give a closed form for the probability $p_{\mathcal{G}|\mathcal{R}}$

$$p_{\mathcal{G}|\mathcal{R}}(\mathbf{R},\mathbf{s}) = \binom{M}{\mu_1 \dots \mu_M} \frac{p_{\mathcal{G}}(\mathbf{s})}{p_{\mathcal{R}}(\mathbf{R})} \prod_{h=1}^{M} \sum_{j=1}^{\bar{N}} \frac{p_{pos}(j) p_{\mathcal{G}}(\mathbf{s})}{p_r((\mathbf{c},\mathbf{q})_h)} \prod_{\ell=1}^{m} \frac{p_{\mathcal{C}}(c_{h,\ell}, q_{h,\ell})}{p_{\mathcal{G}}(s_{j+\ell-1})} p_e^U(c_{h,\ell}, q_{h,\ell}, s_{j+\ell-1})$$

2.3 Case study: error probability of a sequence

In the previous sections we derived a probabilistic model for the sequencing process which can be used to characterize problems in terms of a probabilistic framework. In this section we discuss the problem of finding the probability of a read (or more generally a sequence) to be correct. Although simple, this problem arises very often and can be applied to many different scenarios (see chapters 3 and 4).

Informally a read $(\mathbf{c},\mathbf{q})$ is correct if the sequence of symbols $\mathbf{c} = c_1 \dots c_m$ does not contain any sequencing error. In other words, if j is the position of the reference $S = \mathbf{s}$ where the read is sequenced, then

$$c_1 c_2 \cdots c_m = s_j s_{j+1} \cdots s_{j+m-1}$$

To give a more formal definition of the event *read correct*, we first need to define the sample space properly. The space involves a read $(\mathbf{c},\mathbf{q})$ and a reference sequence S; with the same convention used in previous sections, such space is

$$\Sigma^N \times [1,\bar{N}] \times (\Sigma \times \mathcal{Q})^m.$$

In this space the probability of a read $(\mathbf{c},\mathbf{q})$ correctly sequenced from position $J = j$ of the reference $S = \mathbf{s}$ is given by

$$P(S_{j,\dots,j+m-1} = \mathbf{c} \cap J = j \mid \mathbf{C} = \mathbf{c} \cap \mathbf{Q} = \mathbf{q}).$$

Since we want this probability regardless the actual position of sequencing $J = j$, the correctness probability is defined as the marginal distribution

$$p_C(\mathbf{c},\mathbf{q}) := \sum_{j \in [1,\bar{N}]} P(S_{j,\dots,j+m-1} = \mathbf{c} \cap J = j \mid \mathbf{C} = \mathbf{c} \cap \mathbf{Q} = \mathbf{q}). \tag{2.12}$$

We now assume that positioning is statistically independent from all remaining events

and all position are equiprobable

$$\begin{aligned} p_C(\mathbf{c},\mathbf{q}) &= \sum_{j\in[1,\bar{N}]} P(S_{j,\dots,j+m-1} = \mathbf{c} \cap J = j \mid \mathbf{C} = \mathbf{c} \cap \mathbf{Q} = \mathbf{q}) \\ &= \sum_{j\in[1,\bar{N}]} P(S_{j,\dots,j+m-1} = \mathbf{c} \mid \mathbf{C} = \mathbf{c} \cap \mathbf{Q} = \mathbf{q}) P(J = j) \\ &= \frac{1}{\bar{N}} \sum_{j\in[1,\bar{N}]} P(S_{j,\dots,j+m-1} = \mathbf{c} \mid \mathbf{C} = \mathbf{c} \cap \mathbf{Q} = \mathbf{q}). \end{aligned}$$

if the prior distribution for S is also assumed to be i.i.d. we can further simplify p_C:

$$p_C(\mathbf{c},\mathbf{q}) = P(S_{j,\dots,j+m-1} = \mathbf{c} \mid \mathbf{C} = \mathbf{c} \cap \mathbf{Q} = \mathbf{q})$$

and now we can use the results of previous sections and Equation (2.9) to give a very simple form

$$p_C(\mathbf{c},\mathbf{q}) = p^{iid}_{\mathcal{G}|r}(\mathbf{c},\mathbf{q},\mathbf{c}) = \prod_{\ell=1}^{m} (1 - P_e(q_\ell)). \tag{2.13}$$

which assumes i.i.d. distribution for all prior marginal distributions. Note how, with this hypotheses, the read correctness probability is equivalent to the probability of the sequence S being equal to $\mathbf{c}$ given that the read produced is $(\mathbf{c},\mathbf{q})$.

Finally recall that

$$P_e(q_\ell) = 10^{-q_\ell/10}$$

is the phred function as defined in Equation (2.3).

Next chapters will use this formulation to derive improved version of clustering algorithm in Chapter 3 and filtering of reads in Chapter 4.

Chapter 3

Quality value based clustering

God does not play dice
(Albert Einstein)

In this chapter we present a reads clustering approach based on the application of our stochastic model to alignment-free measures. To the best of our knowledge this is the first study that performs comparison of reads data by combining quality value information and k-mers count. A family of alignment-free measures called D_2^q-type is presented and proved superior to other statistics through a set of experiments on simulated and real sequencing data.

Experimental results show improvements in terms of *precision* and also show that our novel measures D_2^q can be used to boost performance of *de novo assembly* and *metagenomic binning*.

This chapter is based on a joint work with Matteo Comin and Andrea Leoni presented during the 14th Workshop on Algorithms in Bioinformatics (WABI) (Wrocław, Poland, September 8 – 10, 2014) [CLS14, CLS15]; `qCluster` software is freely available to be used for research purposes (`http://www.dei.unipd.it/~ciompin/main/qcluster.html`).

3.1 Alignment free techniques

Alignment-based methods (*e.g.,* `BLAST` [AGM+90]) have been used for quite some time to establish similarity between sequences, in some cases, however, they are not suitable for this task. For example if two genes with a substantial different evolutionary history are found in the same genome, they can not be mapped back to the same common ancestor using alignment based techniques due to the divergence between them.

Furthermore, because of mutation events (*e.g.,* rearrangements) alignment based techniques can not be used for the comparison of whole genomes even between sequences belonging to different specimen of the same species (Sims *et al.* [SJWK09], Comin and Verzotto [CV12b, CV12a]). Despite the considerable research conducted to develop heuristics to speed-up the process, alignment methods are still excessively time consuming, which makes them not appropriate for large-scale sequencing data like the one produced by *Next Generation Sequencing (NGS)* technologies (Song *et al.* [SRZ+13], Comin and Schimd [CS14]). For these reasons a number of alignment-free techniques have been proposed over the last decades (Vinga and Almeida [VA03], Song *et al.* [SRR+13]).

The idea of alignment-free techniques is to use simple summary statistics calculated over the entire sequence without performing any alignment operation. For example many popular statistics are based on the count of k-mers contained in a given genetic sequence. Since similar sequences share similar k-mer count statistics, they can be used to define distance measures to compute similarity between the two sequences.

To prove that alignment-free techniques can effectively be used, the scientific community has derived many different measures that have been successfully applied to several bioinformatics problems. For example, researchers have obtained interesting results (especially for distant related species) on the construction of phylogenetic trees, a task traditionally conducted using multiple-sequence alignment tools (Dai and Wang [DW08]). Alignment-free measures have also been used to: study evolutionary relationships among different organisms (Sims *et al.* [SJWK09], Gao and Qi [GQ07], Qi *et al.* [QLH04]), reconstruct phylogenies of whole genomes (Sims *et al.* [SJWK09], Comin and Verzotto [CV12b, CV12a]), detection of enhancers in ChIP-Seq data (Göke *et al.* [GSLV12], Kantorovitz *et al.* [KRS07]) and entropic profiles (Comin and Antonello [CA13]); for a comprehensive review of alignment-free measures and applications we refer the reader to [VA03] and [SRR+13].

All above approaches apply alignment-free methods to genomic sequences, therefore they require the actual sequences to be known prior to their execution. If such reference is not available, *reads* coming from sequencing experiments must be assembled into *contigs* and then *scaffolded* into a *candidate reference*. As discussed in the introduction (Section 1.2.1), *de-novo* assembly is one of the most challenging problem in bioinformatics, this makes *assembly-free* approaches more appealing and, sometimes, necessary. As a consequence, comparison of genomes based on NGS data has recently become an important research topic (Song *et al.* [SRZ+13], Comin and Schimd [CS14]).

3.1.1 D_2 alignment-free measures

One of the first paper introducing alignment-free method was published in 1986 by Blaisdell [Bla86]. Back then approaches to calculate similarity between sequences, without requiring any alignment, were promising alternatives used to speed-up database searches. In his seminal paper, Blaisdell proposed a statistic, called D_2, to compute the correlation between sequences based on their k-mers count. More specifically D_2 measures the correlation between the number of occurrences of all k-mers appearing in two sequences.

Formally, let X and Y be two sequences from an alphabet Σ^*, for a given word $\mathbf{w}$ of length k we define X_w as the number of times word $\mathbf{w}$ appears in the sequence X when overlaps are allowed (Y_w is defined analogously). For example, given the sequence $X = ATCGAGAG$ and the word $\mathbf{w} = GAG$, $X_w = 2$ since $\mathbf{w}$ occurs on positions 4 and 6 of X.

For a fixed $k \geq 1$ all the X_w define a vector $\mathbf{X}$ with 4^k components, the D_2 statistic is the inner product of the word vectors $\mathbf{X}$ and $\mathbf{Y}$:

$$D_2 = \mathbf{X} \cdot \mathbf{Y} = \sum_{w \in \Sigma^k} X_w Y_w.$$

The D_2 measures is based on the idea that the more similar two sequences are, the higher its numeric values is due to the high number of shared occurrences of k-mers. However, it was shown by Lippert *et al.* [LHW02] that such a statistic can be biased by the stochastic noise of each sequence and, in extreme cases, the statistical power decreases so much that D_2 becomes meaningless.

To address this issue another statistic, called D_2^z, was introduced by Kantorovitz *et al.* in [KRS07], the idea is to compute a normalization of D_2 as

$$D_2^z = \frac{D_2 - \mu_{D_2}}{\sigma_{D_2}}$$

where μ_{D_2} and σ_{D_2} are the expectation and the standard deviation of D_2, respectively. Although the D_2^z similarity improves D_2, it is still dominated by the specific variation of each pattern from the sequences. To account for different distributions of the k-mers, Reinert *et al.* [RCSW09] and Wan *et al.* [WRSW10] defined two new statistics named D_2^* and D_2^s.

Let $\tilde{X}_w = X_w - (N_X - k + 1)p_w$ and $\tilde{Y}_w = Y_w - (N_Y - k + 1)p_w$ where p_w is the prior probability of $\mathbf{w}$ and N_X and N_Y are the lengths of X and Y respectively.

Under the assumption that $N_X = N_Y = N$ the statistics D_2^* and D_2^s are defined as

$$D_2^* = \sum_{w \in \Sigma^k} \frac{\tilde{X}_w \tilde{Y}_w}{(N-k+1)p_w}$$

$$D_2^s = \sum_{w \in \Sigma^k} \frac{\tilde{X}_w \tilde{Y}_w}{\sqrt{\tilde{X}_w^2 + \tilde{Y}_w^2}}.$$

extensions also considering different length sequences (*i.e.,* $N_X \neq N_Y$) has been given by Ren *et al.* in [RSS+13]. but usually this case is not considered.[1]

3.1.2 Quality value extension to D_2 statistics

This section introduces our extension of D_2 statistics that incorporates quality values, the idea is to assign a *weight* to each observed k-mer using the probability of a sequence $(\mathbf{c}, \mathbf{q})$ to be correct according to Equation (2.13):

$$p_C(\mathbf{c}, \mathbf{q}) = \prod_{\ell=1}^{m} (1 - P_e(q_\ell)).$$

from now on, we will assume that the function $P_e(q_\ell)$ corresponds to the phred function (2.3)

$$P_e(q) = 10^{-q/10}$$

because it reflects the interpretation of quality scores as produced by modern sequencers and extensively discussed in Chapter 2 (Section 2.1) and in [EG98, EHWG98].

More formally, let $(\mathbf{c}, \mathbf{q})$ be a pair of k long vectors of symbols $\mathbf{c} = c_1 \dots c_k$ and qualities $\mathbf{q} = q_1 \dots q_k$. We indicate with $(\mathbf{c}, \mathbf{q})_{\ell,\ell+k-1}$ the restriction of $(\mathbf{c}, \mathbf{q})$ to positions $\ell, \ell+1, \dots, \ell+k-1$; that is,

$$(\mathbf{c}, \mathbf{q})_{\ell,\ell+k-1} = (c_\ell \dots c_{\ell+k-1},\ q_\ell \dots q_{\ell+k-1})\,.$$

Let also define the function $\mathbb{1}(a, b)$ as the *indicator* function

$$\mathbb{1}(\mathbf{a}, \mathbf{b}) = \begin{cases} 1 & \text{if } \mathbf{a} = \mathbf{b} \\ 0 & \text{otherwise} \end{cases}$$

where two words $\mathbf{a}$ and $\mathbf{b}$ are equal (*i.e.,* $\mathbf{a} = \mathbf{b}$) if and only if $|\mathbf{a}| = |\mathbf{b}| = k$ and

[1] Since we are going to derive measures for a collection of reads rather than sequences, this case is even less interesting since the normalization are usually given as a function of the size for reads set.

$a_i = b_i$ for all $i = 1, 2, \ldots, k$.

For a word $\mathbf{w} \in \Sigma^k$ and a sequence $X \in \Sigma^N$, we define the *weighted k-mers count* X_w^q as

$$X_w^q = \sum_{\ell=1}^{N-k+1} \mathbb{1}(\mathbf{c}_{\ell,\ell+k-1}, \mathbf{w}) p_C((\mathbf{c}, \mathbf{q})_{\ell,\ell+k-1})$$

or, equivalently:

$$X_w^q = \sum_{i \in \{i|\ \mathbf{w} \text{ occurs in } X \text{ at position } i\}} p_C((\mathbf{c}, \mathbf{q})_{\ell,\ell+k-1})$$

In other words each occurrence of word $\mathbf{w}$ in the sequence X contributes with a value $p_C(\mathbf{w}, \mathbf{c})$ to the final computation of X_w^q, where $\mathbf{q}$ is the associated quality scores vector.

Next we define

$$\tilde{X_w^q} = X_w^q - (N - k + 1) p_w E[P_w] \tag{3.1}$$

where $N = |X|$ is the length of X, p_w is the prior probability of the word $\mathbf{w}$ and the expected number of occurrences $(N-k+1)p_w$ is multiplied by $E[P_w]$ which represents the expected probability of k-mer $\mathbf{w}$ based on the quality scores (discussed later). For two sequences X and Y with same length N, we define our quality value based alignment-free statistics as follows

$$\begin{aligned} D_2^q &= \sum_{\mathbf{w} \in \Sigma^k} X_w^q Y_w^q \\ D_2^{*q} &= \sum_{\mathbf{w} \in \Sigma^k} \frac{\tilde{X_w^q} \tilde{Y_w^q}}{(N - k + 1) p_w E[P_w]} \\ D_2^{sq} &= \sum_{\mathbf{w} \in \Sigma^k} \frac{\tilde{X_w^q} \tilde{Y_w^q}}{\sqrt{\tilde{X_w^q}^2 + \tilde{Y_w^q}^2}}. \end{aligned} \tag{3.2}$$

we call these three alignment-free measures D_2^q-type.

3.1.3 Calculation of $E[P_w]$

In the definition of D_2^q-type statistics (3.2) and in the definition of the auxiliary *weighted k-mer frequency* (3.1), we introduced the normalization factor $E[P_w]$, this quantity can be interpreted as the prior probability of observing the word $\mathbf{w}$; using the notation of Chapter 2

$$P_w := P_r(\mathbf{C} = \mathbf{w}) = \sum_{\mathbf{q} \in \mathcal{Q}^k} P_r(\mathbf{C} = \mathbf{w} \cap \mathbf{Q} = \mathbf{q})$$

where $k = |\mathbf{w}|$ is the length of the word $\mathbf{w}$. In practice this quantity is not easy to estimate for several reasons. First the distribution of events ($\mathbf{C} = \mathbf{w} \cap \mathbf{Q} = \mathbf{q}$) and the same also holds for events ($\mathbf{Q} = \mathbf{q}$) (*i.e.,* the marginal distribution of $P_{\mathcal{Q}}$). Furthermore, as k increases, the number of terms in the summation increases exponentially with it; therefore this definition of P_w can only be used for small values of k.

If the set $\mathbf{R}$ of all the reads is large enough, we can estimate the prior probability using the posterior relative frequency (*i.e.,* frequency observed on $\mathbf{R}$); a similar approach is also implemented in `MAQ` [LRD08].

We defined two different approximations for $E[P_w]$, the first one is the average error probability of the k-mer $\mathbf{w}$ among all reads $\mathbf{x} \in \mathbf{R}$:

$$E\left[P_w\right] \approx \frac{\sum\limits_{\mathbf{x}\in\mathbf{R}} X_w^q}{\sum\limits_{\mathbf{x}\in\mathbf{R}} X_w} \tag{3.3}$$

we call this *Average Word Probability* (AWP). The second approximation defines the average quality for positions ℓ in $\mathbf{w}$ over all the occurrences of $\mathbf{w}$ in $\mathbf{R}$:

$$\overline{\mathbf{q}_w}[\ell] = \frac{\sum\limits_{\mathbf{x}\in\mathbf{R}} \sum\limits_{\{i:\mathbf{x}_i=\mathbf{w}\}} \mathbf{q}_{i+\ell}}{\sum\limits_{\mathbf{x}\in\mathbf{R}} X_w}$$

and uses it compute $E[P_w]$

$$E[P_w] \approx \prod_{\ell=1}^{k} \left(1 - P_e(\overline{\mathbf{q}_w}[\ell])\right) \tag{3.4}$$

we call this second approximation *Average Quality Probability (AQP).*

3.1.4 Accounting for erroneous call

We have seen that, for a k bases long word $\mathbf{w}$ with quality vector $\mathbf{q}$, the corresponding weight added to X_w^q is given by the $p_C(\mathbf{w}, \mathbf{q})$. Given the probabilistic nature of the pair $(\mathbf{w}, \mathbf{q})$, the same word should add to all the wighted frequencies $X_{\mathbf{c}}^q$ a (possibly small) contribution given by $p_C(\mathbf{c}, \mathbf{q})$ for all $\mathbf{c} \in \Sigma^k$.

For example let consider the case where $k = 1$, suppose that $\Sigma = \{A, C, G, T\}$, $\mathbf{w} = w_1 = A$ and $\mathbf{q} = q_1$ satisfies $P_e(q_1) = 0.3$. With the model given so far the pair $(\mathbf{w}, \mathbf{q})$ would only contributes to the term X_A^q (and precisely with the additive term $1 - P_e(q_1) = 0.7$). A slightly more complex model is the *uniform error probability*

which has been defined in Hypothesis 2.3, Equation (2.4) and assumes that all symbols other than A receive the same 0.1 contribution, in other words the current read (w_1, q_1) would induce the following contributions to X_w^q

X_A^q	X_C^q	X_G^q	X_T^q
0.7	0.1	0.1	0.1

where (using the terminology of Comin *et al.* [CLS14]) the *missing quality is redistributed* among all the *neighbor* words of $\mathbf{w}$.

The extension to the case where $k > 1$ is straightforward, but, in principle, it would require to compute $p_C(\mathbf{c}, \mathbf{q})$ for each possible $\mathbf{c} \in \Sigma^k$ a task feasible only when k is small. We decided to limit the adding contributions only to words that differ on one single base for each position. That is, for each position ℓ of a pair $(\mathbf{w}, \mathbf{q})$, we consider all the words $\tilde{\mathbf{w}}$ such that $\tilde{w}_\ell \neq w_\ell$, compute the corresponding $p_C(\tilde{\mathbf{w}}, \mathbf{q})$ and we it to $X_{\tilde{w}}^q$.

For example, given $\mathbf{w} = TGACCA$ and assuming that $1 - P_e(q_3) = 0.3$ we would have that $\tilde{\mathbf{w}} = TGxCCA$ would contribute to X_w^q with 0.1 for all $x \neq A$.

We also tested a different model for redistribution (*i.e.,* a slightly modified version of (Equation 2.4)) where value $P_e(q_\ell)$ is completed to a probability space based on the relative frequency of bases:

$$p_e^{prop}(c, q, s) = \begin{cases} 1 - P_e(q) & \text{if } c = s \\ P_e(q) \frac{f_w(c)}{\sum_{c_i \neq s} f_w(c_i)} & \text{otherwise} \end{cases}$$

where $f_w(c)$ is the relative frequency of symbol $c \in \Sigma$ within the word $\mathbf{w}$.[2] This gives the same contribution of p_e^{iid} when $c = s$ while, when considering neighbor words $\tilde{\mathbf{w}}$, the *redistribution* for base $x \neq c_\ell$ is proportional to the frequency of x in $\mathbf{w}$.

Considering again example, $\mathbf{w} = TGACCA$, we have: $f_w(A) = f_w(C) = 1/3$ and $f_w(G) = f_w(T) = 1/6$, if p_e^{prop} is used then the induced weights are:

T	G	**A**	C	C	A
X	X	**0.7**	X	X	X
T	G	**C**	C	C	A
X	X	**0.15**	X	X	X
T	G	**G**	C	C	A
X	X	**0.075**	X	X	X
T	G	**T**	C	C	A
X	X	**0.075**	X	X	X

[2]The subscripts ℓ here has been removed to avoid cumbersome notation, but what is actually used to compute the contribution is $p_e^{prop}(c_\ell, q_\ell, c_\ell)$.

which is slightly different then the previous case.

We tested quality redistribution with p_e^{prop} and result (presented in [CLS14]) are discussed later in this chapter.

3.2 Alignment-free based reads clustering

In [SL13] Solovyov and Lipkin presented one of the first comparison of alignment-free measures when applied to reads clustering. They focused their attention to k-mer counts-based clustering of reads coming from different genes and different species. They showed that D_2-type measures, in particular D_2^*, can effectively and efficiently detect and cluster reads from the same gene or species. In [CLS14] we presented an extension to this approach that incorporates quality values through the usage of the D_2^q-type measures presented above.

Clustering is the process of partitioning a given input set into κ distinct disjoint subsets, called *clusters*, in such a that elements of the same cluster have minimum distance between them and maximum distance with elements of different clusters. Centroid clustering associates to each cluster one point on the space of input elements called *centroid*. Each element is then assigned to the cluster for which the distance to the centroid is minimized. One of the most commonly used centroid clustering algorithm is *k-means* (in fact *centroid clustering* and *k-means* are often used as synonyms, although the former refers to the mathematical problem and the latter to one possible algorithm to solve it).

In [SL13] authors presented `afcluster` software which uses k-means to compute the clustering of reads based on several distance measures: euclidean norm L_2, Kullback-Liebler divergence (KL) and its symmetrized version (Symm KL) and D_2 statistics. Starting from this software we developed `qCluster` [CLS14] by incorporating the computation of the D_2^q-type statistics using both AWP and AQP prior probability estimators and the redistribution of quality values (*q-red*).

The software takes as input a `fastq` file and performs centroid-based clustering (k-means) of the reads based on the counts and the quality of k-mers. [3]

To avoid as much as possible biases due to the initial random generation of centroids, the final results is constructed as the *consensus* cluster over several runs; that is, k-means is run L times and a consensus is compute from the L results (*e.g.,* average, maximum, ...). Since some of the implemented distances (symmetrized KL, D_2^*) do not guarantee to converge [SL13], we implemented a stopping criterion

[3] Reader should pay attention that the parameter k determining the length of k-mers is completely unrelated to the number of cluster κ.

that stops execution if the number of iterations without improvements exceeds a certain threshold; in this case, the best solution found is returned.

All implemented measures can be computed in linear time and space, this complexity is desirable with large datasets (like the one produced by NGS sequencers).

3.3 Experimental results

3.3.1 Evaluation model

To evaluate the performance of D_2^q-type and all other measures, we preformed several experiments on both simulated and real data.

Assessing performance of clustering algorithm requires us to measure how good clusters are constructed and to which extent they agree with a hypothetical *perfect clustering*. Unfortunately in most cases such ideal result is not available, we decided therefore to use simulated and real data which allow us to compute such information (*i.e.,* the ideal clustering). The most used measures for clustering evaluation are: *recall*, *precision* and a combination of the two called *f-measure* which are briefly discussed next.

When clustering is performed each input element ends up in either the wrong or the right cluster (according to the ideal clustering). Moreover a given cluster may contain elements that belong to it as well as elements that were intended to belong to another cluster. For any cluster κ_i let define the following sets.

True Positive (TP) contains all the elements that have been correctly clustered on κ_i,

False Positive (FP) contains all elements that have been inserted in cluster κ_i, but they were intended for some other cluster,

True Negative (TN) contains all the elements that are not in κ_i and were supposed to be on some other cluster and

False Negative (FN) contains all elements that are not in κ_i, but they should have been in it.

The cardinalities of these sets are used to define the aforementioned measures.

Definition 3.1 (Precision, Recall and F-measure)**.** *For a given cluster κ_i define*

$$P = \frac{|TP|}{|TP| + |FP|} \qquad \textit{Precision}$$

$$R = \frac{|TP|}{|TP| + |FN|} \qquad \textit{Recall}$$

$$F = 2\frac{P \cdot R}{P + R} \qquad F\textit{-Measure}$$

In other words the precision indicates the fraction of elements that were correctly clustered among all the clustered elements while the recall indicates the fraction of correctly clustered elements among all the elements that were supposed to be inserted in that cluster, finally the F-measure is a summary measure of both precision and recall.

Note that is necessary to use both recall and precision measure to obtain meaningful results because account must be taken of the fact that the recall itself doesn't reveal unbalancing between different clusters. For example in the extreme case where one cluster gathers all the reads while the other clusters are empty, the recall would be 1 for the first clusters, even though the overall clustering would very poor (in fact all other clusters would have no elements for which precision, recall and F-measure would not even be defined).

For brevity and to avoid the presentation of redundant data, the results presented here are only given in terms of recall. Since the recall (as well as the precision and F-measure) is defined for each cluster, we need to find a way of producing a single recall for the whole clustering (*i.e.,* a recall that is representative of all clusters rather then of a single one). For each clustering $\kappa_1 \dots \kappa_K$ we searched the clusters with highest true positives TP count and returned the recall of this cluster as the final recall R, moreover, in order to have a more robust measure, we performed the same experiments several time and average the obtained recalls. In all our experiments elements of the input set are reads sequenced from a reference S, these reads could be either simulated or taken from public databases. On datasets for which we don't have the ideal cluster, we need a method to construct the various sets necessary to the calculation of recall (*i.e.,* true positive TP and false negative FP), in other words we need a way to labeling reads with the cluster they belong to. To this extent, we first identified the cluster in which the reads from the sequence S are more numerous and labeled it as the *official* cluster for S. Reads of a sequence inserted in the correct official cluster are true positives, an reads of the same sequence grouped into other clusters are false negatives; we then use Definition 3.1 for the calculation the recall.

3.3.2 Experimental setup

For simulations we used the dataset of human mRNA genes downloaded from NCBI[4], which has also been used in [SL13]. We randomly select 50 sets of 100 sequences with length ranging between 500 and 10000 bases. From each sequence, $M = 10000$ reads of length $m = 200$ were generated using `mason`[5] reads simulator (Holtgrewe [Hol10]) with different parameters, (*e.g.,* percentage of mismatches, read length, ...). We applied `qCluster` using different distances, to the whole set of reads and then we measured the quality of the clusters produced in terms of recall.

Experiments have been conducted with varying values for: length of k-mers, number of clusters, length of reads and average error rate (substitutions only). Clustering were produced using the following distance types: D_2^*, D_2, L_2, KL, Symm KL and compared with D_2^{*q} in all its variants, using both AWP (3.3) and AQP (3.4) forms for $E[P_w]$, with and without quality redistribution (q-red). In order to avoid as much as possible biases due to the initial random generation of centroids, each algorithm was executed 5 times with different random seeds and the clustering with the lower *distortion*[6] (as defined in [SL13]) was chosen.

3.3.3 Results

We discuss here some of the results obtained on simulated and real data, a more comprehensive presentation can be found in [CLS14].

Table 3.1 reports the recall while varying error rates, number of clusters and k-mer length. As expected, for all distances, the recall decreases with the number of clusters (in fact recall would be 1 with only one cluster *i.e.,* no clustering). Interestingly, quality value based measures performs better than any other distance with both low and high error rates (with the possible exception of error free reads where D_2^* shows slightly better perform, not shown here); this confirms that the use of quality values can improve clustering accuracy. In these set of experiments the use of AQP for $E[P_w]$ estimation is more stable and better performing compared with formula AWP. We also noted that contribution of quality redistribution (q-red) is limited, although it seems to have some positive effect. This empirically shows that (unless high accuracy is needed) the computational effort necessary to compute the quality redistribution could be saved without compromising the quality of produced clusters. An extension of Table 3.1 including the same measures with no sequencing errors

[4] `http://ftp.ncbi.nih.gov/refseq/H_sapiens/mRNA_Prot/`

[5] `http://seqan.de/projects/mason.html`

[6] In centroid base clustering the *distortion* is the sum of squared distances of clustered elements from the associated centroid, in [SL13] this definition is extended to reads clustering.

and 5% error rate (published in [CLS14]) confirms the trend summarized here.

A second set of experiments were performed to test the sensitivity of different measures when different error profiles are used. That is, we tested how the distribution of substitutions, insertions and deletions affects the clustering and robust it is against sensibly different error profiles exposed by future generation technologies (like PacBio [CRR+12]).

Table 3.2 shows the recall of all the tested measures with different error profiles; we observed performance similar to the one of Table 3.1. It is interesting to note that, among the different types of sequencing errors, deletions seem to cause a drop of recall more evident than mismatches and insertions (regardless the distance measure used). Surprisingly, our D_2^q statistics performed well even when insertions and deletions are massively inserted (third and fourth columns of Table 3.2), despite the fact that the model does not explicitly consider such events.

3.3.4 Clustering and assembly

Assembly is one of the most challenging computational problems in bioinformatics; it time-consuming with highly variable outcomes for different datasets (Birney [Bir11], Miller *et al.* [MKS10]). Currently large datasets can only be assembled on high performance computing systems equipped with large number of powerful CPU and huge chunks of memory.

Clustering has been used as preprocessing, prior to assembly, to improve memory requirements as well as the quality of the assembled contigs [SL13, BJKG11]. Here we test whether the quality of assembly with real read data can be improved by using alignment-free based clustering with the goal of validating our D_2^q measures.

We used `VELVET` assembler (Zerbino and Birney [ZB08]) which is one of the most popular assembly tool for NGS data. We considered two different genomes: *Helicobacter pylori* and *Zymomonas mobilis* and used the reads datasets *SRR023794* (for the former) and *SRR017901* (for the latter), with about 117 and 23.5 millions bases respectively (corresponding to about 10× coverage for both species). We applied clustering algorithm, with $k = 3$ (k-mers length), and grouped reads into two and three clusters. For each output of clustering algorithm, we run `VELVET` to produce a set of contigs that is then merged into a single output sequence.

In order to evaluate the quality of clustering, we compare this merged sequence to the assembly obtained without clustering (*i.e.,* using of the whole set of reads). Commonly used metrics such as number of contigs, N_{50} and percentage of mapped contigs are presented in Tables 3.3 and 3.4. When merging contigs from different clusters, some contigs might be very similar or they can cover the same region of

Distance	3%	10%
2 clusters		
D_2^*	0,813	0,801
D_2^{*q} AQP	**0,815**	**0,810**
D_2^{*q} AQP q-red	**0,815**	**0,810**
D_2^{*q} AWP	0,806	0,802
D_2^{*q} AWP q-red	0,806	0,802
L_2	0,806	0,801
KL	0,809	0,802
Symm, KL	0,809	0,802
D_2	0,807	0,801
3 clusters		
D_2^*	0,689	0,662
D_2^{*q} AQP	**0,696**	0,689
D_2^{*q} AQP q-red	**0,696**	**0,691**
D_2^{*q} AWP	0,646	0,638
D_2^{*q} AWPq-red	0,646	0,637
L_2	0,673	0,657
KL	0,687	0,672
Symm, KL	0,686	0,669
D_2	0,668	0,654
4 clusters		
D_2^*	0,613	0,574
D_2^{*q} AQP	0,621	0,602
D_2^{*q} AQPq-red	**0,622**	**0,605**
D_2^{*q} AWP	0,563	0,535
D_2^{*q} AWP q-red	0,560	0,533
L_2	0,551	0,540
KL	0,548	0,536
Symm, KL	0,549	0,538
D_2	0,547	0,538
5 clusters		
D_2^*	0,539	0,500
D_2^{*q} AQP	**0,545**	0,532
D_2^{*q} AQP q-red	0,54	**0,533**
D_2^{*q} AWP	0,475	0,463
D_2^{*q} AWP q-red	0,475	0,461
L_2	0,472	0,453
KL	0,488	0,468
Symm, KL	0,488	0,468
D_2	0,464	0,449
$k = 2$		
(a)		

3%	10%
2 clusters	
0,819	0,794
0,822	**0,809**
0,822	0,807
0,807	0,802
0,807	0,802
0,806	0,801
0,809	0,802
0,808	0,802
0,806	0,800
3 clusters	
0,707	0,668
0,711	0,679
0,712	**0,681**
0,662	0,646
0,662	0,644
0,677	0,663
0,689	0,675
0,688	0,673
0,671	0,655
4 clusters	
0,616	0,551
0,617	0,572
0,617	**0,573**
0,571	0,555
0,570	0,555
0,565	0,543
0,558	0,537
0,554	0,539
0,549	0,540
5 clusters	
0,534	0,462
0,544	**0,489**
0,545	0,487
0,494	0,470
0,494	0,470
0,495	0,465
0,501	0,476
0,500	0,474
0,482	0,455
$k = 3$	
(b)	

Table 3.1: Recall of clustering of mRNA simulated reads (10000 reads of length 200) for different measures, error rates, number of clusters and parameter k.

Distance	NO ERRS	SUB = 10%	INS = 10% SUB = 10%	DEL = 10% SUB = 10%
D_2^*	0.862	0.832	0.793	0.809
D_2^{*q} AQP	0.862	**0.864**	**0.864**	**0.861**
D_2^{*q} AQP q-red	0.862	0.856	0.851	0.853
D_2^{*q} AWP	0.863	0.852	0.842	0.848
D_2^{*q} AWP q-red	0.863	0.855	0.848	0.851
L_2	0.863	0.852	0.844	0.849
D_2	0.861	0.852	0.843	0.848
KL	**0.868**	0.855	0.844	0.85
Simm, KL	0.865	0.853	0.843	0.848

Table 3.2: Recall of clustering of mRNA with 5000 simulated reads (reads of length 200, $k = 2$ and 2 clusters) using different errors distribution for substitution (SUB) insertion (INS) and deleetion (DEL).

Distance	Mapped Contigs	N50	Number of Contigs	Genome Coverage
No Clustering	93.55%	112	22823	0,828
D_2^*	93.97%	138	28701	0,914
D_2^{*q} AQP	94.09%	141	29065	0,921
D_2^{*q} AQP q-red	94.13%	141	29421	0,920
D_2^{*q} AWP	94.36%	137	28425	0,907
D_2^{*q} AWP q-red	94.36%	137	28549	0,908
L_2	94.24%	135	28297	0,904
KL	94.19%	135	28171	0,903
Symm, KL	94.27%	134	27999	0,902
D_2	**94.33%**	134	28019	0,903

Table 3.3: Comparison of assembly with and without clustering preprocess ($k = 3$, 2 clusters). The assembly with Velvet is evaluated in terms of mapped contigs, N_{50}, number of contigs and genome coverage. The dataset used is SRR017901 (23.5M bases, 10x coverage) that contains reads of *Z. mobilis*.

Distance	Mapped Contigs	N50	Number of Contigs	Genome Coverage
No Clustering	96.97%	122	16724	0.729
D_2^{*q} AQP q-red	**98.49%**	175	**41086**	**0.994**
D_2^*	98.38%	174	40156	**0.994**
L_2	98.16%	175	36798	0.986
KL	98.28%	178	37717	0.990
Simm, KL	98.30%	182	37217	0.990
D_2	98.22%	**186**	34866	0.987

Table 3.4: Comparison of assembly with and without clustering preprocess ($k = 3$, 3 clusters). The assembly with Velvet is evaluated in terms of mapped contigs, N50, number of contigs and genome coverage. The dataset used is SRR023794 (117MBases) that contains reads of *H. pylori*.

Distance	4 cluster	3 cluster
D_2^*	0.798	0.791
D_2^{*q} AQP q-red	0.798	0.769
D_2^{*q} AWP q-red	**0.801**	**0.826**
L_2	0.643	0.734
KL	0.787	0.805
Simm, KL	0.772	0.792
D_2	0.739	0.771

Table 3.5: Metagenomic reads classification of *H. pylori* (*SRR023794*), *Z.s mobilis* (*SRR017901*), *E.coli* (*FXAWNEV04*) and *L. pneumophila* (*ERR164429*). The recall for different measures with $k = 4$ and 3 and 4 clusters.

the genome, this can artificially increase these measures. We compute therefore a less biased measure as the percentage of the genome covered by the contigs (last column).

The introduction of clustering as a preprocessing step increases the number of contigs and the N_{50}; we think that a more relevant result are represented by the increments of the genome coverage with an improvement up to 10% with respect to the assembly without clustering. The relative performance between the distance measures is very similar to the case observed with simulated data (previous section) and D_2^{*q} with expectation AQP and quality redistribution is again the best performing.

More experiments should be conducted in order to prove that assembly can benefit from the clustering preprocessing; however this first preliminary tests show that, at least for some configuration, a 10% improvement on the genome coverage can be obtained.

3.3.5 Metagenomics classification with clustering

Clustering algorithm for sequencing reads data can be effectively used to group together reads coming from the same organism [CLS14, SL13]. A natural application of clustering is, therefore, the classification of reads coming from metagenomics experiments, this task is usually referred to as *metagenomics binning* or just *binning*.

The problem can be formally stated as follows, given a set of reads $\mathbf{R}$ containing sequencing data coming from a (possibly unknown) number K of different organisms, produce a partition of $\mathbf{R}$ into K subsets in such a way that reads sequenced from the same organism end up in the same set while reads coming form different organisms, are assigned to different sets. We performed few preliminary tests on metagenomic binning, we constructed a set $\mathbf{R}$ with $M = 100000$ reads coming from different organism: *Helicobacter pylori*, *Escherichia coli*, *Zynomonas Mobilis*

and *Legionella Pneunophila*, **R** is constructed by sampling the reads set for different organisms such that the proportion of reads is uniform between all the species. We run `qCluster` with 3 and 4 clusters, few preliminary results are presented in Table 3.5. Quality value based measures perform better with respect to other distances, the other two D_2^q measure not indicated on the table (*i.e.,* without quality redistribution) give results similar suggesting that redistribution of quality values can be skipped if computational time becomes an issue.

This chapter showed that the stochastic model developed in Chapter 2, when applied to alignment-free measures, can be used to define similarity measures (*i.e.,* what we called D_2^q measures) which can then be used to improve performance of clustering.

We have also seen that the model is robust against some complications that naturally arose during the development of a tractable model. For example we have seen that, although not explicitly considered, insertions and deletions have, at least for the purpose of clustering, minor impact on the overall performance.

Chapter 4

Quality value based filtering

God always takes the simplest way
(Albert Einstein)

In this chapter we apply the stochastic model presented in Chapter 2 to the problem of *reads filtering* which is the process of classifying reads based on their quality. More specifically we will use the correctness probability p_C derived in Chapter 2 (Section 2.3) and defined in Equation (2.13) as sorting key for reads; the sorted collection is then passed to subsequent (*downstream*) algorithms.

To appraise the effectiveness of this approach we used sorted and sorted sets as input of: *de novo assembly* and *reads mapping*, experimental results are presented and discussed throughout this chapter. We observed general improvements of downstream algorithms when quality values based filtering is applied as preprocessing step. Both *de-novo* assembly and reads mapping seem to benefit from our filtering approach and we think that further experiments will confirm this claim.

This chapter starts by giving a little introduction to reads filtering and to our *rank filtering* approach, we then move to its application to reads mapping first and *de-novo* assembly afterwards.

In this chapter the contribution of quality values is conveyed by the probability of a read being correct defined in Equation (2.13). For this reason we prefer, whenever possible, to use a lighter notation in this chapter, more specifically to refer to a single read we will often $\mathbf{r}$ instead of $(\mathbf{c}, \mathbf{q})$.

When several subsequences are involved the subscript notation $\mathbf{c}_{i,\ldots,j}$ could become unclear, in such cases we will a *square bracket* notation; that is,

$$\mathbf{c}_{i,\ldots,j} = c_i c_{i+1} \ldots c_j = \mathbf{c}[i, \ldots, j].$$

4.1 Reads filtering

As discussed in the introductory chapter, modern sequencers provide researchers with huge amount of data that need to efficiently processed. The task of analyzing data to infer properties of corresponding genetic sequence, involves many steps that all together constitute what is often called a *processing pipeline* or simply *pipeline*; one of the first of steps is this chain is represented by *reads filtering*.

Informally filtering is the process of classifying reads based on their quality, how such quality is defined and computed depends on specific filters. For example reads may be labeled as *high* quality if a minimum amount of match is found (Dohm *et al.* [DLBH07]) or when certain constraints on quality scores are met (Sasson and Michael [SM10]).

Boolean filtering Once a classification criterion for read quality is defined, the input collection of reads $\mathbf{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_M\}$ can be partitioned into two subsets $\mathbf{R}_H$ and $\mathbf{R}_L$, the first containing *high* quality reads and the second containing *low* quality ones, we call this approach *boolean filtering*.[1] Usually only the set $\mathbf{R}_H$ is passed to subsequent algorithms while $\mathbf{R}_L$ is simply discarded; this behavior is usually accepted because removal of lower quality reads do not appreciably change the final results; this is mainly due to the overwhelming amount of sequencing data available.

4.2 Rank filtering

Boolean filters usually rely on certain constraints that are either met or not by a given read, based on this test reads are classified as high or low quality reads and accordingly inserted into one of the set $\mathbf{R}_H$ and $\mathbf{R}_L$. Even in the rare cases when set $\mathbf{R}_L$ is not discarded, reads within the same set do not have any reciprocal order. In other words given two reads $\mathbf{r}_1, \mathbf{r}_2$ we have no way of deciding which of the two is *better than the other* or (possibly) if they are "equally good" or, more formally, there isn't a *total ordering* between reads.

To solve this problem we propose a different approach to reads filtering; that is, to each read $\mathbf{r} \in \mathbf{R}$, we assign a numeric value $d(\mathbf{r})$ such that, for two reads $\mathbf{r}_1, \mathbf{r}_2 \in \mathbf{R}$, if $\mathbf{r}_1$ is better than $\mathbf{r}_2$ (according to the defined criterion), then $d(\mathbf{r}_1) > d(\mathbf{r}_2)$. This introduces a *total ordering* between reads which allows us to accordingly sort the input collection $\mathbf{R}$ and use the sorted version $\mathbf{R}_{sort} = sort(\mathbf{R})$ as input to the downstream algorithms.

[1] The term *boolean* is not used in literature to refer this specific approach. We decided to adopt it to easily distinguish from *rank filtering*.

Rank filtering has many advantages over boolean filtering. First reads of the original set $\mathbf{R}$ are all inserted in the set $\mathbf{R}_{sort}$, in other words rank filtering is a *lossless* procedure. In some cases, however, not reducing the size of the input may not be the most efficient choice. For example, in graph based assembly algorithms, the size of the graph generally increases with the number of reads inputted. Keeping the graph compact is generally a goal of assembling algorithms, not discarding reads may, therefore conflict with this objective and could also worse performance. This situations, however, can be easily avoided by truncating the set $\mathbf{R}_{sort}$ when certain constraints are violated. For example we can stop processing reads as soon as the $d(\mathbf{r})$ drops under a certain threshold (similarly to what happens with boolean filtering) or we can decide to stop as soon as the algorithm reaches a "critical" point (*e.g.,* graph occupancy exceeds main memory size). This example reveals a second advantage of rank filtering; since reads have been ordered in such a way that *better reads rank higher*, we can add to processing algorithms stopping criteria that are based on this order. The idea is that improvements to the final solution should become less significant as we move downwards the sorted collection $\mathbf{R}_{sort}$. This approach, however, is not always viable, algorithms that do not iteratively use reads to construct the solution, can not be modified to stop at certain point of the input $\mathbf{R}_{sort}$; in these cases boolean filtering can still be used.

Rank filtering also comes with some disadvantages. First we need to properly define the function $d(\mathbf{r})$, a good quality measure for reads must not only be available, but it must also be computable in reasonable time. Secondly, sorting procedure may increase the overall complexity of the pipeline (especially when subsequent algorithms run linearly in the number of reads) and may become a bottleneck of the whole experiment. For large datasets it may not even be possible to sort reads internally (*i.e.,* in main memory); in these cases algorithms for *external sorting* must be used with a further increase in the total execution time.

4.2.1 Rank filtering based on quality value

We introduce now a quality measure for reads $d(\mathbf{r})$ based on the stochastic model presented in Chapter 2. Similarly to what done in Chapter 3 for the definition of D_2^q measures, we will make use of the correctness probability p_C defined in equation (2.13).

Given a read $\mathbf{r} = (\mathbf{c}, \mathbf{q})$ with length m, the probability of $\mathbf{r}$ being correct is

$$d(\mathbf{r}) := p_C(\mathbf{r}) = \prod_{\ell=1}^{m} (1 - P_e(q_\ell))$$

where $P_e(q_\ell)$ is, as usual, the phred function $P_e(q) = 10^{-q/10}$ discussed at the beginning of Chapter 2.

In the next sections we will use rank filtering as preprocessing step of *reads mapping* and *de novo assembly*; preliminary results will be presented and briefly discussed. Throughout the remaining of this chapter $d(\mathbf{r}) = p_C(\mathbf{r})$ will be used as measure of the quality of reads, some alternatives will be discussed in Chapter 5.

4.3 Results on mapping

The first application used to test our rank filtering approach is *reads mapping*. Informally mapping is the process of identifying positions of a reference sequence S where a given sequence $\mathbf{r}$ (*e.g.,* a read) most likely comes from. Mapping can also be referred to a whole collection $\mathbf{R}$ of sequences in which case mapping $\mathbf{R}$ refers to mapping all elements of $\mathbf{R}$ separately.

A very common definition of mapping problem is given in terms of a *scoring function*. That is, given two sequences x and y with length $|x|$ and $|y|$ such that $|x| < |y|$ and given a scoring function $f(x, y, j)$ the *mapping* of x into y is a set of positions $J^* = \{j_1^*, j_2^*, \ldots\}$, $j_i^* \in [1, |y|]$ such that,

$$j_i^* = \arg\max_{j \in [1,|y|]} f(x, y, j) \qquad \forall j_i^* \in J^*. \tag{4.1}$$

That is, the mapping consists of all positions j_i^*, $i = 1, 2, \ldots$ where the scoring function f is maximized. Some approaches, however, define the scoring function as distance between sequences, in this cases the mapping is the set of all positions that *minimize* the function f, in general mapping resorts *optimization* of the function f.

Different mapping tools differ from each other mainly on the definition of the scoring function f. Most of them define f recursively; that is, the mapping of subsequence $x_{1,\ldots,i}$ into subsequence $y_{\ell,\ldots,j}$ is defined as a recursive relation:

$$f(x_{1,\ldots,i}, y_{\ell,\ldots,j}, \ell) = F(x_{1,\ldots,i-1}, y_{\ell,\ldots,j}, \ell)$$

where F is a proper function. Notable example of this approach are represented by: Levenshtein or edit distance [Lev66], Needleman-Wunsch distance [NW70], Smith-Waterman distance [SW81] and `BLAST` tool (Altschul *et al.* [AGM+90]). In all cases of practical interest , the defined scoring recurrence can be optimized using *dynamic programming* techniques, consequently, all these mapping approaches, can be computed in time and space $O(|x||y|)$. Unfortunately, in most cases, this time complexity is also a *lower bound* in the sense that, every algorithm for global maxima

calculation, requires time $\Theta(|x||y|)$.

In real cases y is the reference sequence (which can contain billions of symbols), when all reads from the set $\mathbf{R}$ must be mapped into y, using dynamic programming algorithm may require $O(M|x||y|)$ complexity for the whole task; in most cases this time is quadratic in $|y|$.[2] For large reference sequences quadratic algorithms are unfeasible and, in some cases, usage of alignment-free techniques (see Chapter 3), local alignment (*e.g.,* `BLAST`, [AGM+90]) or approximate solution (*e.g.,* `MAQ` [LRD08]) may be the only available choice.

Hamming distance To test the effectiveness of our rank filtering approach, we used a simple scoring function based on the concept of *hamming distance* (Hamming [Ham50]). Let x and y be two sequences defined over the alphabet Σ with the same length m. The *Hamming distance* $H(x, y)$ between x and y is defined as

$$H(x, y) = \sum_{i=1}^{m} \mathbb{1}(x_i, y_i) \tag{4.2}$$

where $\mathbb{1}$ is the indicator function

$$\mathbb{1}(a, b) = \begin{cases} 1 & a = b \\ 0 & \text{otherwise} \end{cases} .$$

In other words Equation (4.2) represents the number of mismatches between sequences x and y.

$H(x, y)$ can be used as a simple scoring function for mapping of reads, in this case Equation (4.1) becomes

$$j^* = \arg\max_{j \in [1,|y|]} \left[-H(x, y_{j,\ldots,j+|x|-1}) \right]. \tag{4.3}$$

Note that, being $H(x, y)$ a distance measure, optima can be found by either minimizing $H(x, y)$ or by maximizing $-H(x, y)$.

[2] For a sequencing experiment with coverage γ

$$\sum_{h=1}^{M} |x_h| = \gamma |y|$$

in particular for constant read length: $M|x| = \gamma|y|$ and $O(M|x||y|) = O(|y|^2)$.

4.3.1 Evaluation model

Given a read $\mathbf{r}$ and a N bases reference S we can use a linear time alignment algorithm (*i.e.,* pattern matching Cormen *et al.* [CLR+01]) to compute $H(\mathbf{r}, S, j)$ for a fixed position j, to compute Equation (4.3) we must run this algorithm $O(N)$ times with an overall complexity $O(mN)$. The task of mapping reads coming from a collection $\mathbf{R}$ with M requires total time $O(mMN) = O(N^2)$.

This asymptotic complexity for mapping algorithm can be unacceptable when either M, N or both are large; we decided to further simplify our evaluation model and consider only reads that *perfectly* map (*e.g.,* without mismatches) back to the reference sequence. More formally we say that a read $\mathbf{r} = (\mathbf{c}, \mathbf{q}) \in \mathbf{R}$ *perfectly match* a position j of $S = s_1 \dots s_N$ if

$$H(\mathbf{c}, s_{j,\dots,j+m-1}) = 0.$$

As a metric for evaluating the effectiveness of filtering, we used the *percentage of reads without mismatches* at different ranks. The idea is that, if reads are sorted from higher to lower quality, this percentage would ideally be a non-decreasing function of the number of considered reads. An ideal sorting criterion, would rank at the highest positions the $\tilde{M}$ reads not containing any mismatch. In this case the percentage of reads without mismatches would be 100% for all the first $\tilde{M}$ points, and then decreasing according to $1/h$ as h increases toward M.[3] When no sorting at all is implemented, such a percentage should roughly remain constant to the value $\tilde{M}/M$ which is also the value to which both curves (for sorted and unsorted sets) converge.

Identifying a good measure to test rank filtering was not straightforward; we wanted such a measure to be independent from any aspect not related to the ranking and, at the same time, giving a numerical indication of reads quality relatively to the rank of reads. We think that the selected measure is a good trade-off between expressiveness and simplicity especially given the preliminary nature of these experiments; of course before drawing final conclusions other measures and tests must be performed to confirm results presented here.

We also appraised rank filtering when used as pre-processor to *de-novo* assembly, more specifically we evaluated the produced assembly in terms of: N_{50}, contigs length and other established measures and observed how they change as the input varies

[3] More precisely the ideal curve is

$$g(h) = \begin{cases} 1 & 1 \leq h \leq \tilde{M} \\ \frac{\tilde{M}}{h} & \tilde{M} \leq h \leq M \end{cases}$$

where $\tilde{M}$ is the total number of reads without errors.

based on our sorting criterion.

To identify the reads without errors we used a technique based on k-mers mapping which is described in the next section. This approach has been chosen because can be easily implemented in time $O(N \log N)$ on average (see Proposition 4.1).

4.3.2 Algorithmic approach

Given two reads $\mathbf{r}_1 = (\mathbf{c}, \mathbf{q})_1$ and $\mathbf{r}_2 = (\mathbf{c}, \mathbf{q})_2$ with the same length m, we say that $\mathbf{r}_1$ and $\mathbf{r}_2$ *perfectly match* (or simply *match*) if $c_{1,\ell} = c_{2,\ell}$ for all $\ell = 1, \ldots, m$; that is,

$$H(\mathbf{c}_1, \mathbf{c}_2) = 0.$$

For a constant parameter k, a k-mer is a sequence of length k; when $k \leq m$ each of the read $\mathbf{r}_1$ and $\mathbf{r}_2$ contains exactly $\bar{m} = m - k + 1$ k-mers. Let $\mathcal{K}_k(\mathbf{r}_1)$ be the *ordered sequence* of k-mers in $\mathbf{r}_1$ and let $\mathcal{K}_k(\mathbf{r}_2)$ be defined analogously for $\mathbf{r}_2$; that is, for a read $\mathbf{r} = (\mathbf{c}, \mathbf{q})$:

$$\mathcal{K}_k(\mathbf{r}) = (\mathbf{c}[1, \ldots, k], \mathbf{c}[2, \ldots k+1], \ldots, \mathbf{c}[m-k+1, \ldots, m])\,.$$

It is easy to prove that, if two reads $\mathbf{r}_1, \mathbf{r}_2$ perfectly match then $\mathcal{K}_k(\mathbf{r}_1) = \mathcal{K}_k(\mathbf{r}_2)$; in other words, if two reads match, then the corresponding k-mers list must be identical (the converse is also true). As trivial corollary is the, if reads $\mathbf{r}_1$ and $\mathbf{r}_2$ match, then

$$\mathbf{c}_1[1, \ldots, k] = \mathbf{c}_2[1, \ldots, k]$$

in other words a necessary condition for $\mathbf{r}_1$ and $\mathbf{r}_2$ to perfectly match is that they share the same first k-mer.

If we now consider the sequence S with length $N \geq m \geq k$ and a read $\mathbf{r} = (\mathbf{c}, \mathbf{q})$ with length m, we say that $\mathbf{r}$ *perfectly map* at position j of S if

$$H(\mathbf{c}, S[j, \ldots, j+m-1]) = 0$$

and, similarly to the two reads case, a necessary condition for $\mathbf{r}$ to perfectly map at position j is that $\mathbf{c}[1, \ldots k]$ aligns without mismatches with $S[j, \ldots, j+k-1]$. Our approach is based on the idea that, for a read $\mathbf{r}$ and a sequence S, candidate mapping positions can be find using *seeds* positions j of S for which

$$H(\mathbf{c}[1, \ldots, k], S[j, \ldots, j+k-1]) = 0$$

and, only if this requirements is met, perform the actual computation of the hamming distance. For constant k the task of finding all reads that perfectly map to some positions of S can be computed in time $O(N \log N)$ on average using proper data structures. [4]

The overall idea is to compute an indexed version $\mathcal{I}_S$ of $\mathcal{K}_k(S)$ which allows retrieval of $S[j, \ldots, j+k-1]$ in average time $O(1)$. Then, for each of the M reads, we scan the list of seeds positions j in average time $O(\log N)$ and test if the current read matches the position j.

Computation of the index $\mathcal{I}_S$ To attain an average $O(N \log N)$ complexity for the mapping, we need to compute the index $\mathcal{I}_S$ in the same (or better) asymptotic time. Algorithm 1 shows the pseudocode for the procedure `KmerIndex` which creates

Algorithm 1 Procedure to map all k-mers of a given sequence

```
1:  procedure KmerIndex(S, k)
2:      N ← Length(S)
3:      I ← ∅
4:      kmer ← S[1..k]
5:      repeat                                  ▷ Scans through all the k-mers of S
6:          I[kmer] ← I[kmer] ∪ i               ▷ Store every observed k-mer
7:          i ← i + 1
8:          kmer ← kmer[2..k] ∪ S[i]
9:      until i > N
10:     return I
11: end procedure
```

$\mathcal{I}_S$. This procedure simply scans all the k-mers on a reference sequence S (loop on lines 5 – 9) and keeps track of the position where they appear (line 6). If the index $\mathcal{I}_S$ is implemented using a hash table, the retrieval of an element takes on average $O(1)$ (Cormen *et al.* [CLR+01]) and there are an average of $O(\log N)$[5] elements to scan for each retrieved element. Of course these complexities hold as long as the k-mers are (roughly) uniformly distributed.

Finding seed k-mer Once the index $\mathcal{I}_S$ is computed, the procedure to map each read starts. Algorithm 2 gives a possible implementation that, under the hypotheses of Proposition 4.1, runs in average time $O(N \log N)$. Given the read $\mathbf{r}$, its first k-mer $\kappa = r_1 \ldots r_k$ is computed (line 4), afterwards the set $\mathcal{P}$ of all the occurrences of κ in

[4] The worst case, however, remains $O(N^2)$.

[5] This complexity is true *with high probability* as defined in Mitzenmacher and Upfal [MU05] and holds for implementations of the hash table and for hashing the distributes k-mers in a random fashion.

Algorithm 2 Procedure to map k-mers against reference for reads set

```
1: procedure SeedKmerMapping(S, R, k)
2:     I_S ← KmerIndex(S, k)                          ▷ Maps k-mers of reference S
3:     for r ∈ R do
4:         κ ← r[1...k]                                          ▷ First k-mer of r
5:         P ← I_S[κ]                                ▷ All position where κ occurs in S
6:         for j ∈ P do
7:             if Match(S[j...j+m−1], r) then
8:                 RecordMatch((r, j))                        ▷ We found one match for r
9:                 break
10:            end if
11:        end for
12:    end for
13: end procedure
```

S is retrieve from $\mathcal{I}_S$ (line 5). For each positions $j \in \mathcal{P}$, if $\mathbf{r}$ perfectly matches with $S[j, \ldots, j+m-1]$ (line 7), it is recorded and the algorithms move to the next read, otherwise the next position in $\mathcal{P}$ is considered.

The worst case complexity of Algorithm 2 is quadratic, however if we assume that k-mers of S are (roughly) uniformly distributed on Σ^k, than the complexity becomes $O(N \log N)$ on average.

Proposition 4.1. *Procedure* `SeedKmerMapping` *has average time complexity*

$$O(N \log N + Mm \log N) = O(N \log N) \tag{4.4}$$

if k-mers are uniformly distributed on S and procedure `KmerIndex` *runs in $O(N \log N)$ average time.*

Proof. The first term $O(N \log N)$ comes from the complexity of `KmerIndex` for k-mer are distributed uniformly. The **if** statement in line 7 can be implemented in time proportional to m using linear time pattern matching algorithms (Cormen *et al.* [CLR+01]), and the loop of line 6 is executed, on average, $\log N$ times (because of the uniformity of k-mers). Therefore the outer loop (line 3) runs i total time $O(Mn \log N)$, using the fact that $Mm = \gamma N$ with constant coverage γ, the claim follows. □

These algorithms have implemented using C++ language[6] and has been used to conduct experiments presented and discussed in the remaining of this chapter.

[6] `https://github.com/skimmy/lib-bio/`

4.3.3 Experiments

In this section we present experimental results for the evaluation of rank filtering described in previous sections.

Reference sequence To keep experiments relatively fast, we decided to use a short DNA sequence, the *Zaire Ebolavirus* GenBank number KJ660348.2[7] with 18959 bases has been used.

Reads data When aligning real sequencing data against the associated reference sequence, the number of errors (*i.e.,* mismatches) contained in the read is not necessarily a good measure of the performance of the alignment technique. In particular evolutionary events like mutations (*e.g.,* Single Nucleotide Polymorphisms – SNPs) between the reference sequence and the *real* sequence can induce spurious mismatches.

The model we are using for filtering reads does not take into account mutations and the experimental setup should avoid as much as possible the presence of such events; we therefore decided to use only simulated reads for this first set of experiments. That is, after reference S is chosen, reads are generated using reads simulator softwares to guarantee that all reads come exactly from S. By doing this we completely eliminated the problem of spurious errors due to mutation and only detect mismatches that are caused by sequencing artifacts.

Two reads simulators have been used, one is `mason` also used during experiments on clustering (see Chapter 3) and the second one is a custom written generator that produces reads with a simple *Independent and Identical Distributed (i.i.d.)* model. The former has been chosen to reflect as much as possible a real sequencing experiment (while still avoiding mutation issues) and the second has been developed to test rank filtering when sequencing fits the model assumed by the sorting criterion.

Being central part of the stochastic model under test, quality values have been generated using specific error profiles; `mason` simulator has been used with the *illumina* preset, probability of substitution (`-pmm` option) set to 0.01 and probabilities for insertions and deletion (`-pi` and `-pd` options) set to 0; all the remaining parameters have been left to their default values.

Our custom reads simulator takes as input a probability distribution for quality scores and uses it to generate quality scores for each of the sequenced symbol. After having generated a quality score q, the simulator performs a substitution with probability $p = P_e(q)$ (Equation (2.3)), the "new" base is chosen at random (*i.e.,*

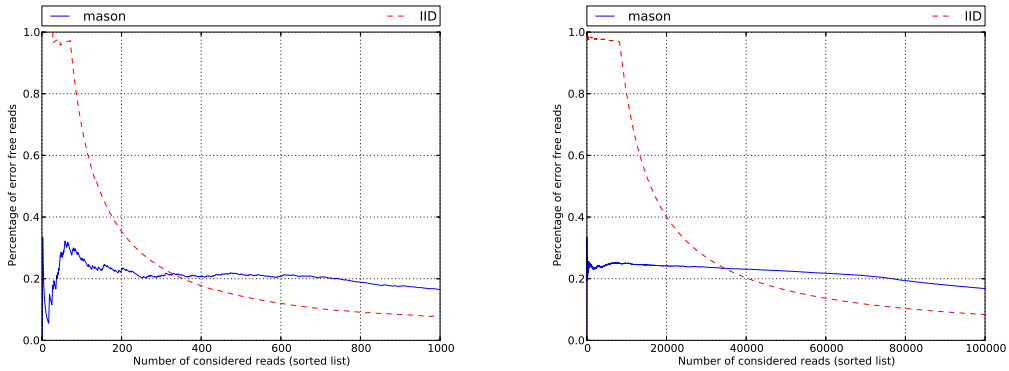
[7] `http://www.ncbi.nlm.nih.gov/nuccore/KJ660348`

with probability 1/3) from any of the bases different from the real one. For example after generating the pair $(G, 20)$, a random number $\rho \in [0, 1)$ is generated, if $\rho \leq P_e(20) = 0.01$ then a substitution occurs and the sequenced base is chosen uniformly (*i.e.,* with probability 1/3) from the set $\{A, C, T\}$.

To keep the two datasets as much as possible consistent with each other, we measured the quality value distribution produced by `mason` (when run with the aforementioned parameters) and used it as the input of our custom generator.

To evaluate filtering we used the percentage of reads without mismatches as described in previous sections.

Results Figure 4.1 shows results of rank filtering while varying the number of generated reads M. These two graphs show the percentage of reads without errors (y-axis) as we move the sorted collection $\mathbf{R}_{sorted}$ from the top to bottom (toward positive x direction). As expected, reads at the top of the sorted list are more



(a) Ebola with $M = 1000$ (b) Ebola with $M = 100000$

Figure 4.1: Fraction of reads without errors as a function of the amount of reads considered within the sorted set $\mathbf{R}_{sort}$ for `mason` (solid blue curve) and i.i.d. (dashed red curve) simulators with $M = 1000$ reads (a) and $M = 100000$ reads (b).

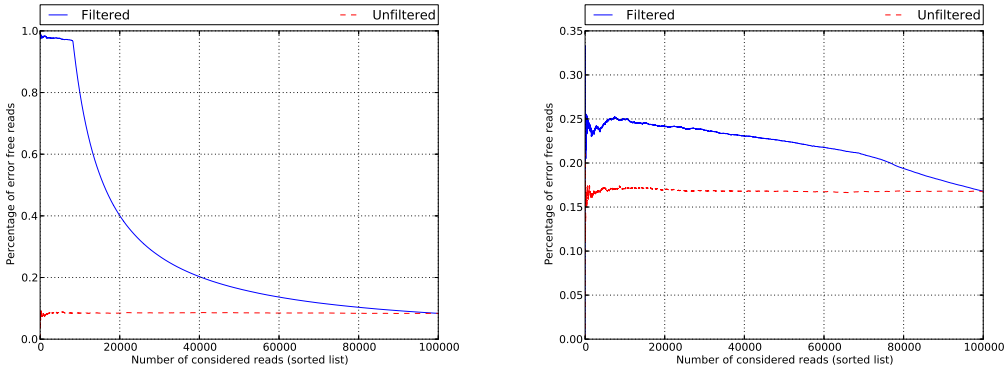
likely to perfectly match with the reference, as we move down the ranked list (move toward positive direction of x-axes), the fraction of total reads that have no errors diminishes. We also observed as i.i.d. generator attains performance significantly better than using `mason` (as expected).

Note that `mason` generates a fraction of perfect matching reads higher than i.i.d. simulator run with the same quality scores distribution for. This is a little surprising and indicates that the distribution of quality values generated by `mason` does not reflects an i.i.d. model, this, although expected, arise the problem of tweaking our stochastic model to reflect this aspect.[8]

[8]We have not yet done this tests because we want first evaluating the behavior with real se-

Finally we see (as one would expect) that the size of the set $\mathbf{R}$, does not play a major role on filtering, the only appreciable difference is represented by the initial fluctuation which is, however, mostly caused by the measure we are testing.

Figure 4.2 presents result while running the mapping algorithm on the sorted list $\mathbf{R}_{sort}$ and on the unsorted one $\mathbf{R}$ using both simulators. We see how rank filtering improves mapping of reads with respect to no filtering with both i.i.d. (Figure 4.2 (a)) and `mason` (Figure 4.2 (b)) simulators. As for the previous set of experiments we observe a remarkable difference between the two simulators suggesting that the model used to generate reads (*i.e.,* the empirical model for sequencing) plays an important role in the performance of rank filtering.



(a) Results for i.i.d. simulator

(b) Results for `mason` simulator

Figure 4.2: Fraction of reads perfectly matching with the reference for *filtered* list $\mathbf{R}_{sort}$ (blue solid curve) and *unsorted* list $\mathbf{R}$ (red dashed curve) with i.i.d. (a) and `mason` (b) reads generators.

4.4 Results on assembly

The second application where we tested rank filtering approach is as preprocessing step of *de novo* assembly. *De novo* assembly is the process of reconstructing a reference sequence S starting from a set of reads $\mathbf{R}$ sequenced from S. More realistically unique reconstruction of S is not possible when $\mathbf{R}$ comes from a real sequencing experiments; sequencing errors and the presence of complex structures (*e.g.,* repeats) in the reference, make the solution to assembly ambiguous in the sense that more *candidate* assembly are usually identified.

Assemblers try their best to reconstruct as much as possible of the sequence S by producing the longest subsequences of S that they can infer; these long fragments are called *contigs*. Assemblers, can not deduce relative ordering of contigs, this

quencing data in order to avoid *over-fitting* of our model.

task is usually performed using long reads or mate pairs in a subsequent step called *scaffolding*.

As done in Chapter 3, we used `VELVET` assembler (Zerbino and Birney [ZB08]) and calculated standard performance measures (in particular contigs N_{50}, see Section 4.4.1) to test how output contigs set changes as the input varies.

4.4.1 Evaluation model

Evaluating assembly algorithms and their result is not an easy task, the main problem is that we don't have the *real* sequence (*i.e.,* the one that we would like to reconstruct) and, consequently, relating the *reconstructed* sequence with the real one is hard when not impossible.[9] Another difficulty is represented by the fact that different assemblers produce outputs that may significantly differ. Moreover parameters tuning plays an important role and can significantly influence results of assemblers even for the same tool and on the same input data. We also need to precisely define which characteristics of assemblers are desirable and which instead could be overseen. For example we may prefer a fragmented assembly that covers most of the original sequence rather then a less fragmented one that spans a lower fraction of the reference sequence.

For these reasons many different measures are have been defined to asses performance of assembly algorithms. In the previous chapter we tested assembly on clusters using: *mapped contigs* (percentage of outputted contigs that map back to the reference), contigs N_{50} (explained below), *number of contigs* produced and *genome coverage* (fraction of original sequence covered by some contig). Results presented in this chapter are given as contig N_{50} which is briefly introduced next.

Contig N_{50} One of the most used metric to assess assembler's quality is called *contig* N_{50} (usually simply referred to as N_{50}). According to Miller *et al.* [MKS10]

> The contig N_{50} is the length of the smallest contig in the set that contains the fewest (largest) contigs whose combined length represents at least 50% of the assembly.

Another similar (but more convoluted) definition has been given by Earl *et al.* in [EBJ+11].

> The N_{50} of an assembly is a weighted median of the lengths of the sequences it contains, equal to the length of the longest sequence S, such

[9] This does not apply to *comparative assembly* which, however, is not considered in this thesis.

that the sum of the lengths of sequences greater than or equal in length to S is greater than or equal to half the length of the genome being assembled.

The idea is to give higher weight to longer contigs rather than to shorter ones. This should ensure that assemblers producing less fragmented output, performs better (in terms of N_{50}) than those producing more fragmented. This statistics, however, does not take into account any information about the actual coverage of the contigs once mapped to the reference (*e.g.,* few long contigs covering the same region may induce a better N_{50} than shorter but best covering ones) which is the reason why we introduced the genome coverage in tables 3.3 and 3.4.

We decided, notwithstanding above critics, to show here results on contigs N_{50} for several reasons.

- N_{50} is a *de-facto* standard measure appearing in most of the studies on assembly algorithms, as such it is well understood and accepted as a reliable way of measuring performance of assembling software.

- Our experiments compare outputs produced by the same assembler (*i.e.,* `VELVET`) always run with the same parameters, in other words we are not evaluating the efficacy of the assembler, but the efficacy of our rank filtering.

- The `VELVET` software used to perform simulations, computes the N_{50} statistics *off-the-shelf* giving a good way for comparing different runs using the same algorithms and the same definition of N_{50}.

4.4.2 Experiments

This section presents experimental results obtained using `VELVET` assembler applied to the sets $\mathbf{R}_{top}$ and $\mathbf{R}_{bottom}$ both containing all the M reads for the input dataset $\mathbf{R}$ the first sorted according to our rank filtering sorting criterion, the second reversing such an order. That is, for the quality measure $d(\mathbf{r})$ defined as read correctness probability p_C (Equation 2.13)), the collection $\mathbf{R}_{top}$ satisfies

$$d(\mathbf{R}_{top}[i]) \geq d(\mathbf{R}_{top}[j]) \qquad \forall i \leq j$$

and $\mathbf{R}_{bottom}$ is the reversed version of $\mathbf{R}_{top}$ which satisfies

$$d(\mathbf{R}_{bottom}[i]) \leq d(\mathbf{R}_{bottom}[j]) \qquad \forall i \leq j.$$

α	**Nodes**	**N50**	**Max**	**Total**	**Reads**	**Used Reads**
1	3927	5493	25265	4422893	1753096	99.21%
0.95	3296	6007	25780	4424905	1665307	99.2%
0.9	3260	5992	25783	4425387	1577701	99.2%
0.75	3292	5889	25368	4425363	1314954	99.22%
0.5	4849	2301	14662	4298977	876484	99.2%
0.25	9103	463	6427	2754771	423257	95.81%
0.1	5162	277	3959	1086962	153884	87.09%
0.05	2845	245	3693	540707	72996	82.62%

Table 4.1: Output of `VELVET` for the dataset SRR959247 with $\mathbf{R}_{top}$ input as α varies.

Note that $\mathbf{R}_{top}$ and $\mathbf{R}_{bottom}$ are *ordered collections*; when dealing with sequencing data, this ordering is enforced by the way reads are stored (*e.g.,* the order of entries in a `fastq` file).

Dataset As opposed to experiments conducted on reads mapping, we decided to use a *real* (*i.e.,* not simulated) set of reads. This decision is partly motivated by the fact that we wanted to test rank filtering in a real environment and partly because, by using only N_{50}, we don't need to map contigs to the reference and, therefore, we don't have to worry about spurious mismatches due to mutations. We used the library *E. coli, AT, S, N* accession number `SRR959247`[10] containing about 1.7 millions reads of average length 176 bases for the *Escherichia Coli str. K-12 substr. DH10B* organism (reference available with accession number NC_010473.1[11]) summing to an approximate coverage of 74×; reads have been produced using *Illumina HiSeq 2000* sequencer.

The sets $\mathbf{R}_{top}$ and $\mathbf{R}_{bottom}$ have been "truncated" so that only the $\lfloor \alpha M \rfloor$ highest rank reads are considered (note that in $\mathbf{R}_{bottom}$ higher rank means lower p_C). We used different values of α from 1 (whole dataset) to 0.05 (only the top 5% of the entire set).

To evaluate the quality of assemblies we relied on the statistics outputted by `VELVET` software specifically: the number of nodes contained in the final graph, the contig N_{50}, the length of the longest contig(s) produced, the total length of all contigs and the number of reads used to construct the contigs. As an additional we also calculated the percentage of total input reads used to construct the output contigs:

$$\frac{\text{aligned reads}}{\text{total reads}}\%.$$

[10] `http://www.ncbi.nlm.nih.gov/sra/?term=SRR959247`
[11] `http://www.ncbi.nlm.nih.gov/nuccore/NC_010473.1`

α	**Nodes**	**N50**	**Max**	**Total**	**Reads**	**Used Reads**
1	3927	5493	25265	4422893	1753096	99.21%
0.95	4791	5011	25153	4390174	1663150	99.07%
0.9	4890	4644	25151	4376319	1567818	98.58%
0.75	4860	4481	29441	4357874	1308355	98.72%
0.5	5519	3508	21054	4318706	857819	97.09%
0.25	13580	405	10012	3255457	404420	91.55%
0.1	3717	152	2888	346460	65084	36.83%
0.05	1578	137	1397	83681	31723	35.91%

Table 4.2: Output of `VELVET` for the dataset SRR959247 with $\mathbf{R}_{bottom}$ input as α varies.

`VELVET` setup Since the focus of the experiments was on the effectiveness of rank filtering using quality values, we run all the simulations with the same assembly parameters with the only exception of the *expected coverage* (`-exp_cov`) which has been adjusted based on the number of reads considered (*i.e.,* αM).

More precisely we experimentally estimated the optimal values for the size of k-mers which has been set to 21 and for the *coverage cutoff* (`-cov_cutoff`) set to 9. These values guarantee a relatively high quality due to the high k while the coverage cutoff (used to discard *undercovered* k-mers supposedly erroneous) has been chosen by experimentally determining the optimal value within the interval $[3, 15]$.

Results and discussion Tables 4.1 and 4.2 shows the results on sets $\mathbf{R}_{top}$ and $\mathbf{R}_{bottom}$ respectively, they contain the output statistics produced by `VELVET` as α varies. A graphic representation of the N_{50} values is given in Figure 4.3.

As we expected, the value of contig N_{50} decreases as the number of reads considered decreases, this reflects the fact that the chosen dataset contains high quality reads, in fact the measured average quality of the set is 36.4 corresponding to an average error rate 0.02%.

When $\mathbf{R}_{top}$ and $\mathbf{R}_{bottom}$ are compared, we observe similar trends for the N_{50} with the former underperforming the latter in every experiment with the exception of the case $\alpha = 0.5$. This observations enforce our claim that using quality value based filtering improves the performance of the assembly.

What is more interesting is the percentage of reads used to construct contigs (last columns of tables 4.1 and 4.2). While these values are comparable in both sets for $\alpha \geq 0.5$, we see that, as predicted, when considering small fraction of the set $\mathbf{R}_{bottom}$ (*i.e.,* $\alpha \leq 0.1$) the percentage of "good" reads dramatically drops. This behavior reflects the fact that in $\mathbf{R}_{bottom}$ the top α fraction of reads corresponds exactly to the bottom α fraction of $\mathbf{R}_{bottom}$.

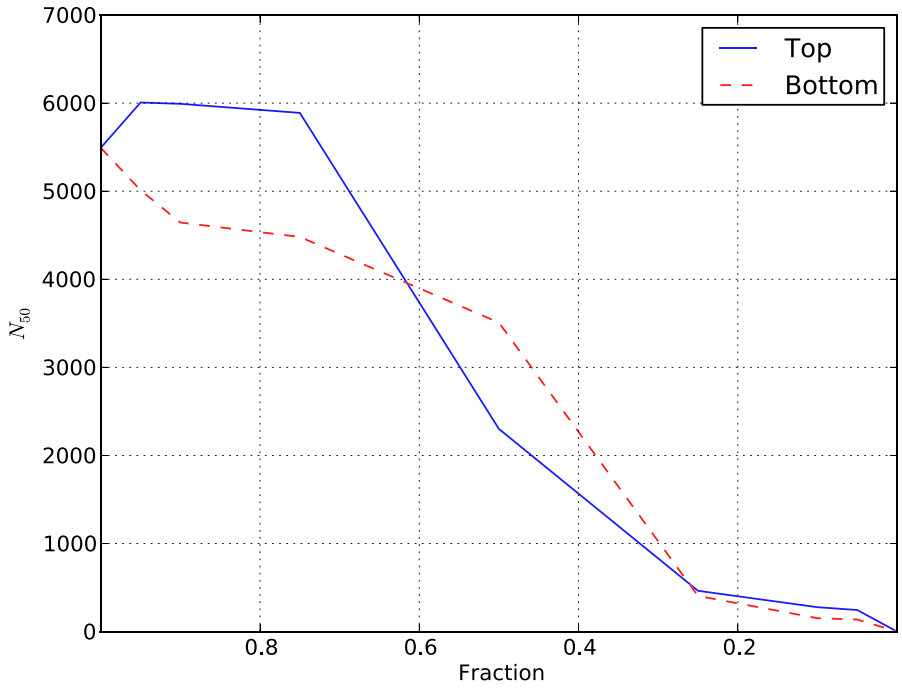


Figure 4.3: N_{50} obtained by `VELVET` on dataset SRR959247 while using $\mathbf{R}_{top}$ (blue solid line) and $\mathbf{R}_{bottom}$ (red dashed line) input sets.

In this chapter we gave preliminary experimental results on *rank filtering* using the quality value based model developed in Chapter 2. We shown that this novel approach to reads filtering helps using sequencing data in a more effective and efficient way. We also proved that mapping and assembly benefit from a preprocessing step of rank filtering. This results, although preliminary, represent a further validation of the model presented in Chapter 2 and, at the same time, give a tool that can already be used to perform reads filtering by simply sorting reads during the preprocessing step.

Chapter 5

Conclusions and future directions

Einstein, don't tell God what to do
(Niels Bohr)

Drawing conclusions of a scientific work is not easy; every time new results are presented, they arise new questions that require further investigations with this cycle, apparently, rolling indefinitely. It is always useful, however, to mark milestones and check points during the development of a theory; the conclusive chapter of a Ph.D. thesis is the perfect spot for such considerations.

In this thesis we presented a novel stochastic model for the description of the sequencing process; the main disruption with previously developed models is in the incorporation of quality values as part of the model itself. To the best of our knowledge this is the first work that uses these scores to give a stochastic representation the entire sequencing process.

Starting from the interpretation of quality values as correctness probability of the associated symbol, we built, step by step, a stochastic model that allows us to describe, in a formal probabilistic framework, many different aspects of the sequencing process like: production of a single pair (c, q) of symbol c and quality q, process of positioning within the reference sequence and many others.

We showed that the model can be effectively used to develop a new family of alignment-free measure which we called D_2^q-type. We experimentally proved that these measures can improve performance of the clustering of reads which, in turn, helps boosting algorithms for *de-novo* assembly and metagenomics binning. The experiments performed with `qCluster` and `VELVET` softwares, showed that D_2^q-type statistics perform better than their cousins D_2, indicating that the inclusion of quality values in k-mers frequency count, is effective and improves the overall statistical power of these alignment-free measures.

We also applied the same model to the problem of *reads filtering*; we defined

a new filtering paradigm, which we called *rank filtering* that, instead of discarding low quality reads, defines a *total ordering* between them such that higher quality reads rank higher than lower quality ones. In a set of preliminary tests, we showed that this approach is effective when our quality value based probabilist model is used as ranking criterion. We also observed some limitations of our model while aligning filtered reads generated by `mason` software; we think that minor changes to our model could lead to better in this cases as well, however a set of experiments on real datasets should be performed to evaluate filtering of real reads.

Overall results show that the usage of quality values has positive impact on many bioinformatics problem, we think that the approach presented in this thesis can be further improved in terms of both theoretical model and its application; some of these future extensions and applications are discussed next.

Future directions

While in Chapter 2 we developed a stochastic model describing the whole process of sequencing, in chapters 3 and 4 we showed experimental validation of only one part of this model. What presented in this thesis should be interpreted more as a starting point for new research rather an endpoint. There are many ways in which our model could be extended and many situations in which it could be profitably employed. Moreover some aspects presented in the previous chapters could to be refined and extended to give the model more descriptive power.

The remaining of this chapter is devoted to briefly summarize the possible future directions we plan to investigate.

Extension to the model A first, somehow obvious, aspect that can be enhanced and refined is the definition of the theoretical model presented in Chapter 2. Many details of the sequencing process have not been considered in the current form of the model and may be incorporated in future refinements.

A central idea of our approach is the usage of quality values to model the error in sequencing data. Most of this thesis assumed that distribution of errors is uniform among *uncalled* symbols, this has been formalized in Hypothesis 2.3 and Equation (2.4). This assumption does not include sequencing errors like *insertions* and *deletions* because modeling them would require a more complicated description, in particular many of the independence hypotheses we assumed should be revisited to properly consider such errors. Even more importantly, there is no unique interpretation of quality values when insertions or deletions occur; one of the reason for this

is that major causes of these types of errors can not be detected with data processed by base callers. We want to include these errors in near future extensions especially because, as future generation sequencing will become more and more important the necessity of models fitting new error profiles will increase.

An issue related to sequencing processing, which has not be considered in Chapter 2, is represented by the, so called, *reads orientation*. A sequenced read could come from either the *forward* or the *reverse* strand of the DNA molecule;[1] when it is outputted no indication is available about the strand it comes from. We think that this aspect could be easily included to the model, but we also think that the contribution of this extensions will be limited in those cases (like, for example, k-mers count) where reads orientation does not play a fundamental role.

Another aspect of the sequencing process related to NGS reads, is the possibility of using *mate pairs*. In our first version of the model, we avoid the inclusion of this feature because their effectiveness becomes really relevant mostly when dealing with complicated structures of reference sequences (*e.g.,* solving repeats), which is an aspect we did not directly considered in the applications presented throughout this thesis (except for assembly which, however, has not yet been directly defined in terms of our model). Moreover, the two pairs could be *separated* and used as distinct reads if needed. This *destroys* the information of pairing, but maintains constant the coverage of the whole experiment. Of course inclusion of mate pairing as part of the model should have benefit in terms of performance although this effect would probably be only partial for the problems we presented in this thesis (*e.g.,* clustering, filtering, ...).

Further experimental validation In chapters 3 and 4, we presented applications of our model to problems of *clustering* and *filtering*. Both these applications used the same definition, given in Equation (2.13), of the *read correctness probability*. In both cases we observed improvements on the tested scenarios, however, while results in Chapter 3 have already been published [CLS14, CLS15], the ones presented in Chapter 4 are still preliminaries and need further investigations. More specifically we want to test how *rank filtering* performs on mapping of real data (*i.e.,* not from simulated reads) and also test the application of our sorting criterion to future generation sequencing data (especially PacBio). Another aspect we plan to study, is the comparison of our measure with other quality value based filtering, for example comparing our model with the one defined in `MAQ` software for classification of reads based on quality scores. We think that results in these scenarios combined with the

[1] Of course this does not apply to single stranded molecules.

ones presented in Chapter 4 will give a good validation of our quality valued based approach for reads filtering.

Given the interesting results obtained by our measure of read correctness, we plan to apply it to new problems. In particular we want to carry out experiments to test our D_2^q type alignment-free measures as distance measure between phylogeny trees, we think that, also in this case, our quality values based measures could outperform D_2-type ones.

All these applications focus on the probability of a single read to be correct, however the model presented in Chapter 2 gives a more powerful and comprehensive tool that can be used to describe an entire sequencing experiment.

With the goal of experimentally test our model in all of its parts, we identified *de-novo* assembly as interesting application where our model could represent a breakthrough with respect to the *state of the art*. Our goal is to define, in terms of our stochastic model, the problem of assembly and then use such definition to develop a novel assembler that takes advantage of it. *De-novo* assembly is a really challenging problem, we can already give a rough definition as *maximum likelihood assembly*, but currently we don't have a feasible algorithm to solve it (Baruzzo [Bar13]). We need to find characterization of the optimal solutions for the assembly such that their computation becomes feasible.

Other applications In Chapter 2 we concentrated our efforts on finding probability of a reference given a set of reads. However, if we already know the reference and, of course the collection of reads, we can apply our model to estimate sequencing parameters. For example, in many cases we assumed that positioning is uniformly distributed on the reference; it is known that this assumption is not coherent with most of the sequencers. If we had a known reference, we could have used our model to precisely define the distribution of positioning for a particular sequencer and, possibly, use it to refine our model. This, parameter estimation application will become very important in the near future as new sequencing technology will be widely adopted and their characteristics need to be determined.

Computational aspects One aspect that has been, mostly, ignored in this thesis is related to the computational aspects of the algorithms we used throughout the experimental validation of the model. However, the problem of efficiently processing sequencing data is still an open field especially as both models for sequencing and computation paradigms evolve according to technological advances.

Since the advent of next generation sequencers, the cost of producing sequencing data has dramatically decreased and nowadays this costs are negligible when com-

pared with the costs of supercomputers needed to process all the generated data. This is also true in terms of time; it is not uncommon that, to process sequencing data produced in one day, algorithms need to run for several days on supercomputers.

In this scenario developing fast and scalable algorithms is challenging and requires careful design of processing algorithms. For example, given the large amount of data to be processed, it is necessary to develop algorithms able to fully exploit the *hierarchical* nature of memory in modern architectures. Moreover parallelism has become a key aspect more so today with the advent of *Graphics Processor Unit (GPU)* that promises very high theoretical performance, but require algorithms to be re-engineered in order to fully exploit their architectures.

We think that the future of bioinformatics relies on the ability of exploiting the entire information produced by sequencers (for example by including quality scores) using the ever increasing computation power available. Bioinformaticians need to develop expertise on both theoretical modeling of problems and on optimization strategies, in a situation where both this aspects evolve at astonishing rates, they must be able to keep the connection between the two aspects and present elegant, powerful and innovative solutions to old and new problems. What is even more thrilling and exciting is that, as technologies evolve, this challenging becomes more and more complicated and, therefore, more and more stimulating.

Appendix A

Notation

Mathematics is written for
mathematicians
(Nicolaus Copernicus)

This appendix contains a short description of symbols commonly used in the development of the stochastic model in chapter 2. Although most of this notation may apply to other chapters, it is possible to encounter some discrepancies when applied in this other cases. Finally keep in mind that there may be cases (hopefully rarely) where variables could have different meanings from the one described in this appendix (even within chapter 2), however they will be clearly identifiable from the context.

Sequence, read and reads set and reads collection

Σ	Alphabet
$\mathcal{Q}$	Set of quality scores
S	Reference sequence
N	Length of the reference sequence S, $N = \|S\|$
$(\mathbf{c}, \mathbf{q})$	Read as a pair of vectors of symbols $\mathbf{c}$ and qualities $\mathbf{q}$
$\mathbf{r}$	Read produced by sequencing experiment
m	Length of a read, $m = \|\mathbf{r}\| = \|\mathbf{c}\| = \|\mathbf{q}\|$
$\mathbf{R}$	Set or collection of reads, $\mathbf{R} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots\}$
M	Number of read in a given set, $M = \|\mathbf{R}\|$
k	Length of a k-mer

Indexes and other variables

ℓ	Usually used to refer a position within a read, $1 \leq \ell \leq m$
h	Usually used to index reads on a read set, $1 \leq h \leq M$
j	Usually used to refer a position within the sequence, $1 \leq j \leq N$
$c_{h,\ell}$	Symbol in position $\ell = 1, \ldots, m$ of h-th read
$q_{h,\ell}$	Quality score in position $\ell = 1, \ldots, m$ of h-th read

Probability spaces and probability functions

Ω	Sample space (in all cases Ω is a finite set)
$\mathcal{F}$	Event space, $\mathcal{F} = \{E : E \subseteq \Omega\}$
P	Probability function, $P : \mathcal{F} \longrightarrow [0, 1]$
p	Shorten version of probability function, $p(e) = P(E = e)$

Conventions

Reads set and collection Often throughout the thesis a unordered collection of reads, identified with $\mathbf{R}$ is referred to as *reads set* or *dataset*, however they are not sets in a mathematical sense.

Composed spaces The development of stochastic model presented in chapter 2 involves defining sample spaces Ω from previously defined. The space $\Omega_{X,Y}$ usually refers to the cartesian product $\Omega_X \times \Omega_Y$.

Conditioned spaces Many events assume the form $(X = x \mid Y = y)$, since conditional probabilities induce a probability space ([PP02]), in many cases the space $\Omega_{X|Y}$ will be implicitly used by the definition of the probability function $p_{X|Y}(x, y) = P_{X,Y}(X = x \mid Y = y)$.

Bibliography

[ADHP09] D. Aloise, A. Deshpande, *et al.* Np-hardness of euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[AGM+90] S. F. Altschul, W. Gish, *et al.* Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403 – 410, 1990.

[Bar13] G. Baruzzo. *A maximum likelihood approach to genome assembly.* Master's thesis, Department of Information Engineering, Padova, Italy, 2013.

[Bir11] E. Birney. Assemblies: the good, the bad, the ugly. *Nature methods*, 8(1):59–60, 2011.

[BJKG11] E. Bao, T. Jiang, *et al.* Seed: efficient clustering of next-generation sequences. *Bioinformatics*, 27(18):2502–2509, 2011.

[Bla86] B. E. Blaisdell. A measure of the similarity of sets of sequences not requiring sequence alignment. *Proceedings of the National Academy of Sciences*, 83(14):5155–5159, 1986.

[Bre10] H. Breu. A theoretical understanding of 2 base color codes and its application to annotation, error detection, and error correction. Technical report, Applied biosystems by life technologies, 2010.

[CA13] M. Comin and M. Antonello. Fast computation of entropic profiles for the detection of conservation in genomes. In *Pattern recognition in bioinformatics*, pages 277–288 (Springer), 2013.

[CFG+10] P. J. A. Cock, C. J. Fields, *et al.* The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic Acids Research*, 38(6):1767–1771, 2010.

[Chu89] G. Churchill. Stochastic models for heterogeneous dna sequences. *Bulletin of Mathematical Biology*, 51:79–94, 1989. 10.1007/BF02458837.

[CLR+01] T. H. Cormen, C. E. Leiserson, *et al. Introduction to algorithms*, volume 2 (MIT press Cambridge), 2001.

[CLS14] M. Comin, A. Leoni, and M. Schimd. Qcluster: Extending alignment-free measures with quality values for reads clustering. In *Algorithms in Bioinformatics*, pages 1–13 (Springer), 2014.

[CLS15] M. Comin, A. Leoni, and M. Schimd. Clustering of reads with alignment-free measures and quality values. *Algorithms for Molecular Biology*, To appear, 2015.

[CRR+12] M. Carneiro, C. Russ, *et al.* Pacific biosciences sequencing technology for genotyping and variation discovery in human data. *BMC Genomics*, 13(1):375, 2012.

[CS14] M. Comin and M. Schimd. Assembly-free genome comparison based on next-generation sequencing reads and variable length patterns. *BMC Bioinformatics*, 15(Suppl 9):S1, 2014.

[CT06] T. M. Cover and J. A. Thomas. Elements of information theory 2nd edition. 2006.

[CV12a] M. Comin and D. Verzotto. Alignment-free phylogeny of whole genomes using underlying subwords. *Algorithms for Molecular Biology*, 7(1), 2012.

[CV12b] M. Comin and D. Verzotto. Whole-genome phylogeny by virtue of unic subwords. In *Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on*, pages 190–194 (IEEE), 2012.

[dBE46] N. G. de Bruijn and P. Erdos. A combinatorial problem. *Koninklijke Nederlandse Akademie v. Wetenschappen*, 49(49):758–764, 1946.

[DDM+12] S. Djebali, C. A. Davis, *et al.* Landscape of transcription in human cells. *Nature*, 489(7414):101–108, 2012.

[DLBH07] J. C. Dohm, C. Lottaz, *et al.* Sharcgs, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome research*, 17(11):1697–1706, 2007.

[Dur98] R. Durbin. *Biological sequence analysis: probabilistic models of proteins and nucleic acids* (Cambridge university press), 1998.

[DW08] Q. Dai and T. Wang. Comparison study on k-word statistical measures for protein: From sequence to 'sequence space'. *BMC bioinformatics*, 9(1):394, 2008.

[EBJ+11] D. Earl, K. Bradnam, *et al.* Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome research*, 21(12):2224–2241, 2011.

[EFG+09] J. Eid, A. Fehr, *et al.* Real-time dna sequencing from single polymerase molecules. *Science*, 323(5910):133–138, 2009.

[EG98] B. Ewing and P. Green. Base-calling of automated sequencer traces using phred. ii. error probabilities. *Genome research*, 8(3):186–194, 1998.

[EHWG98] B. Ewing, L. Hillier, *et al.* Base-calling of automated sequencer traces usingphred. i. accuracy assessment. *Genome research*, 8(3):175–185, 1998.

[GQ07] L. Gao and J. Qi. Whole genome molecular phylogeny of large dsdna viruses using composition vector method. *BMC evolutionary biology*, 7(1):41, 2007.

[GSLV12] J. Göke, M. H. Schulz, *et al.* Estimation of pairwise sequence similarity of mammalian enhancers with word neighbourhood counts. *Bioinformatics*, 28(5):656–663, 2012.

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

[HMU06] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)* (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA), 2006.

[Hol10] M. Holtgrewe. Mason–a read simulator for second generation sequencing data. *Technical Report FU Berlin*, 2010.

[KL78] K. Kozl and C. Listy. Biochemical nomenclature and related documents. *Chem. Listy*, 72:288–305, 1978.

[KRS07] M. R. Kantorovitz, G. E. Robinson, and S. Sinha. A statistical method for alignment-free comparison of regulatory sequences. *Bioinformatics*, 23(13):i249–i255, 2007.

[KSL+13] D. C. Koboldt, K. M. Steinberg, *et al.* The next-generation sequencing revolution and its impact on genomics. *Cell*, 155(1):27–38, 2013.

[Lev66] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707. 1966.

[LHW02] R. A. Lippert, H. Huang, and M. S. Waterman. Distributional regimes for the number of k-word matches between two random sequences. *Proceedings of the National Academy of Sciences*, 99(22):13980–13989, 2002.

[LLB+01] E. S. Lander, L. M. Linton, *et al.* Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001.

[LRD08] H. Li, J. Ruan, and R. Durbin. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858, 2008.

[LZR+10] R. Li, H. Zhu, *et al.* De novo assembly of human genomes with massively parallel short read sequencing. *Genome research*, 20(2):265–272, 2010.

[Met09] M. L. Metzker. Sequencing technologiesthe next generation. *Nature Reviews Genetics*, 11(1):31–46, 2009.

[MHB+10] A. McKenna, M. Hanna, *et al.* The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome research*, 20(9):1297–1303, 2010.

[MKS10] J. R. Miller, S. Koren, and G. Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315 – 327, 2010.

[MU05] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis* (Cambridge University Press), 2005.

[NP09] N. Nagarajan and M. Pop. Parametric complexity of sequence assembly: theory and applications to next generation sequencing. *Journal of computational biology*, 16(7):897–908, 2009.

[NW70] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

[PJ12] R. K. Patel and M. Jain. Ngs qc toolkit: a toolkit for quality control of next generation sequencing data. *PloS one*, 7(2):e30619, 2012.

[PP02] A. Papoulis and S. U. Pillai. *Probability, random variables, and stochastic processes* (Tata McGraw-Hill Education), 2002.

[PS08] M. Pop and S. L. Salzberg. Bioinformatics challenges of new sequencing technology. *Trends in Genetics*, 24(3):142–149, 2008.

[PTW01] P. A. Pevzner, H. Tang, and M. S. Waterman. An eulerian path approach to dna fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17):9748–9753, 2001.

[QLH04] J. Qi, H. Luo, and B. Hao. Cvtree: a phylogenetic tree reconstruction tool based on whole genomes. *Nucleic acids research*, 32(suppl 2):W45–W47, 2004.

[RCSW09] G. Reinert, D. Chew, *et al.* Alignment-free sequence comparison (i): statistics and power. *Journal of Computational Biology*, 16(12):1615–1634, 2009.

[RSS+13] J. Ren, K. Song, *et al.* Multiple alignment-free sequence comparison. *Bioinformatics*, 29(21):2690–2698, 2013.

[Sha48] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[SJWK09] G. E. Sims, S.-R. Jun, *et al.* Alignment-free genome comparison with feature frequency profiles (ffp) and optimal resolutions. *Proceedings of the National Academy of Sciences*, 106(8):2677–2682, 2009.

[SL13] A. Solovyov and W. I. Lipkin. Centroid based clustering of high throughput sequencing reads based on n-mer counts. *BMC Bioinformatics*, 14(1):268, 2013.

[SM10] A. Sasson and T. P. Michael. Filtering error from solid output. *Bioinformatics*, 26(6):849–850, 2010.

[SNC77] F. Sanger, S. Nicklen, and A. R. Coulson. Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977.

[SRR+13] K. Song, J. Ren, *et al.* New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Briefings in bioinformatics*, page bbt067, 2013.

[SRZ+13] K. Song, J. Ren, *et al.* Alignment-free sequence comparison based on next-generation sequencing reads. *Journal of Computational Biology*, 20(2):64–79, 2013.

[Sta79] R. Staden. A strategy of dna sequencing employing computer programs. *Nucleic acids research*, 6(7):2601–2610, 1979.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[TS11] T. J. Treangen and S. L. Salzberg. Repetitive dna and next-generation sequencing: computational challenges and solutions. *Nature Reviews Genetics*, 13(1):36–46, 2011.

[VA03] S. Vinga and J. Almeida. Alignment-free sequence comparison – a review. *Bioinformatics*, 19(4):513–523, 2003.

[WC53] J. D. Watson and F. H. Crick. Molecular structure of nucleic acids. *Nature*, 171(4356):737–738, 1953.

[WRSW10] L. Wan, G. Reinert, *et al.* Alignment-free sequence comparison (ii): theoretical power of comparison statistics. *Journal of Computational Biology*, 17(11):1467–1490, 2010.

[XW05] R. Xu and I. Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[ZB08] D. R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome research*, 18(5):821–829, 2008.

[ZRS+12] Z. Zhai, G. Reinert, *et al.* Normal and compound poisson approximations for pattern occurrences in ngs reads. *Journal of Computational Biology*, 19(6):839–854, 2012.