SCUOLA DI DOTTORATO
UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

University of Milano-Bicocca

Department of Informatics, Systems and Communication Ph.D.

Program in Computer Science

XXXIII Cycle

iSwap: a bioinformatics pipeline for index switching in Illumina sequencing platforms

Ph.D. Candidate

Dr. Adriano De Marino, 827257

Tutor

Prof. Leonardo Mariani

Supervisor

Prof. Marco Antoniotti

Coordinator: *Prof. Leonardo Mariani*

Academic Year: 2019/2020

To my to madness

Acronyms

BAM, Binary-sequence Alignment Format

BP, Base Pair

CLI, Command Line Interface

CSV, Comma-Separated Values

cDNA, complementary DNA strand

DB, DataBase

DNA, Deoxyribonucleic Acid

FPR, False Positive Rate

GO, Gene Ontology

GTF, Gene Transfer Format

GT, Gene Therapy

GUI, Graphical User Interface

HIV, Human Immunodeficiency Virus

HSC, Hematopoietic Stem Cell

HSPC, Hematopoietic Stem-Progenitor Cell

IP, Internet Protocol

IS, Integration Site

LAM-PCR, Linear Amplification Mediated PCR

LC, Linker Cassette

LTR, Long Terminal Repeat

LV, Lentiviral Vector

miRNA (microRNA), Small non-Coding RNA Molecule

MLD, Metachromatic Leukodystrophy

NGS, Next Generation Sequencing

PCR, Polymerase Chain Reaction

RNA, Ribonucleic Acid

RV, Retroviral Vector

SAM, Sequence Alignment Format

SC, Sequence Count

scRNA-seq, single-cell RNA sequencing

SE, Single End

SLiM-PCR, Sonicated-Linker-Mediated-PCR

SR-Tiget, San Raffaele Telethon Institute for Gene Therapy

TAG, Sequence of Fixed Nucleotides

TSV, Tab-Separated Values

VCN, Vector Copy Number

Preface

This basis for this research originally stemmed from my passion for developing better methods of data storage and retention. As the world enters the digital age, generating huge amounts of data, there will be a greater need to produce excellent quality data. My passion is not only to discover, but to develop tools to break down research barriers. Helping my colleagues simplify procedures that have been difficult and time-consuming so far. In truth, I couldn't have reached my current level of success without a strong support group. My committee members, each of whom has provided patient advice and guidance throughout the research process. Thank you all for your unwavering support.

My work has been supported by many projects, both international and national funding:

- Telethon Foundation
- Ph.D. Program in Computer Science, University of Milano-Bicocca
- Igenomix Italy S.r.l. R&D department

Abstract

In Next generation sequencing technologies, hundreds or thousands of DNA samples can be sequenced simultaneously (multiplexing) and the obtained sequencing reads can be distinguished by the presence of sample-specific nucleotide sequences (indexes) embedded in the primers used for the DNA amplification. Custom bioinformatics pipelines, by reading the indexes present in the sequencing reads assign them to a specific sample (demultiplexing). Multiplexing however is plagued by **index switching**, a phenomenon occurring when free index primers are randomly fused to DNA sequences belonging to other unrelated samples of the library pool and resulting in the incorrect assignment of sequences to one or multiple wrong samples. In the field of gene therapy (GT) (see Appendix A), vector integration site (IS) studies heavily depend on sequencing of DNA fragments (containing proviral-cellular genome junctions) from several samples and are affected by index switching. This issue is particularly relevant in clonal tracking studies, where the level of shared IS between different cell lineages or different time points of the same GT patient are required to define the levels of multilineage reconstitution and estimate the number of stem cells and other calculations. Therefore, the spreading of IS between datasets caused by index switching could result in inflated sharing IS levels which could lead to misinterpretation of the results.

To evaluate the extent of index switching in IS analyses, we analysed 123,431,269 sequencing reads originating from a pool composed by 54 samples amplified in triplicate, each tagged by two indexes fused to the ends of the PCR products containing the LTR and Linker Cassette (LC) sequences resulting in 162 index combinations (combining a total of 48 LTR and 32 LC indexes). From this analysis we found that >95% of sequencing reads belonged to the correct 162 index combinations while a the remaining 5% of reads belonging to 1374 false index combinations resulting from frank events of index swapping. The levels of swapping were similar among the different LTR and LC indexes with an average of 1709 ± 3469 reads (range 9 to 52000) for false index combinations. We then evaluated the levels of sharing of univocally mapped IS between different samples and found that essentially all samples had different levels of

contamination. Overall, 91.5% of IS were assigned to a single sample, 7.25% were found shared in two samples and the remaining 1.25% were present in more than 2 samples. Focusing on a sample from a cell line with 6 knowns IS we calculated the spreading levels and their relative abundance on other samples. From this analysis we found that at least one of the 6 know IS were found in 13 unrelated samples out of 54 (24%). In 3 out of 13 samples the amount of contaminating reads from this cell line reached levels ranging from 13 to 40% of the entire dataset. These high levels of contaminations justified the development of new approaches for indexing switching correction in IS studies. To this aim we developed a set of algorithms that allows to remove contaminating sequences by comparing the level of IS sharing between technical replicates, eliminating sequences where multiple unique molecular identifiers are linked to the same shear site, comparing the relative abundances of each IS and other parameters that will be described.

We are now testing and validating in depth this approach in a controlled dataset of known IS. A final refinement of the method will complete the correction of the IS data, allowing to release reliable results without the noise introduced by sequencing artefacts. This study started with the integration site analysis, but after was extended to other different fields. In the thesis is showed a new method for cleaning dataset from this kind of contaminations.

Contents

ACRONYMS **5**

PREFACE **8**

ABSTRACT **10**

CONTENTS **13**

INTRODUCTION **16**

STATE-OF-THE-ART **19**

2.1 THE SOIL: AN UNEXPLORED SOURCE OF GENES 19
2.2 FROM THE FIRST MICROORGANISMS TO METAGENOMICS 19
2.3 A NEW WORLD: METAGENOMICS 21
2.4 NEXT GENERATION SEQUENCING PLATFORMS (NGS) 22
2.4.1 Comparing Sanger sequencing to NGS 25
2.4.2 Illumina/Solexa sequencing 25
2.5 SHOTGUN METAGENOMICS AND AMPLICONS ANALYSIS 28
2.6 RIBOSOMAL GENE 16S RRNA 30

ISWAP: A BIOINFORMATICS PIPELINE FOR INDEX SWITCHING IN ILLUMINA SEQUENCING PLATFORMS .. 33

3.0 BIOINFORMATICS PIPELINE 33
3.1 QUALITY CONTROLS AND FILTERS 34
3.2 ADAPTER REMOVAL AND TRIMMING 36
3.3 DEMULTIPLEXING 36
3.4 CHIMERA DETECTION 37
3.5 CREATING OTUS – CLUSTERING 38
3.6 USEARCH 39
3.7 CHOICE OF REPRESENTATIVE 42
3.8 RAM REQUIRED 43
3.8.1 Number optimization 46
3.8.2 Optimizing Numeric Columns with Subtypes 48
3.8.3 Optimizing object types using categoricals 52
3.9 SEQUENCE CALIBRATION 61
3.10 INDEX SWITCHING REMOVAL AND DATA RECONSTRUCTION 62
3.11 INDEX SWITCHING ASSIGNMENT PROCESS 66

RESULTS AND COMPARISON OF TOOLS FOR INDEX SWITCHING 70

4.1 WHY ARE WE DOING "DATA GENERATION"? 70
4.2 SEQUENCING'S LIBRARY....72
4.4 ISWAP APPLICATION 79
4.5 PERFORMANCES PRECISION AND RECALL....82
4.6 EXPERIMENTAL SOLUTIONS FOR INDEX SWITCHING 85

CONCLUSION 88

REFERENCES....91

APPENDIX 100

APPENDIX A GENE THERAPY 100
APPENDIX B LAB PROCEDURES 102
SONICATED LINKER-MEDIATED (SLIM)-PCR 102
B.1.2 Fusion Primers for SLiM-PCR....103
APPENDIX C INFORMATION SYSTEM....104
C.1 Computational Resources 104
C.2 Storage....105
C.2.1 NAS Server: QNAP TS-412 4-BAY....105

ACKNOWLEDGEMENTS....107

Introduction

With the rapid increase in throughput of next-generation sequencing technologies, an individual run of a typical sequencing machine (such as those produced by Illumina) generates many more reads than is necessary for interrogating single libraries generated by most functional genomic assays. To make efficient use of these machines, DNA libraries are typically pooled together prior to sequencing, in a process known as "multiplexing". Briefly, unique barcodes are ligated into the ends of the DNA molecules within each library before pooling. This incorporates a known sequence into each read, allowing the assignment of reads to their libraries of origin after sequencing. Multiplexing also ensures that technical effects are consistent across samples, avoiding batch effects between sequencing lanes or flow cells; and can provide robustness against the failure of sequencing lanes, which would otherwise result in the loss of entire samples. As such, multiplexing is widely considered to be standard practice for many sequencing experiments and is essential for cost-effective analysis of small libraries such as those in single-cell RNA sequencing (scRNA-seq), Gene Therapy, metagenomics studies.

The most recent DNA sequencing machines released by Illumina (HiSeq 3000/4000/X, X-Ten, and NovaSeq) use patterned flow cells to improve throughput and cost efficiency. On these new flow cells, the process of "seeding" DNA molecules into the patterned wells and amplification of the seeded DNA occur simultaneously. These machines have been in use for several years in a diverse range of genomic fields. However, it has been recently reported that the use of these machines can lead to the mislabelling of DNA molecules with the incorrect library barcode (Sinha et al. 2017). The mislabelling is likely driven by the extension of free barcode molecules using other DNA molecules as a template (Figure 1.1). The phenomenon has been acknowledged by Illumina [57], although estimates of swapping fractions vary between reports [58]. It is unclear whether a permanent solution to the problem will be forthcoming as rapid amplification after seeding is critical to the operation of the patterned flow cell machines [59].

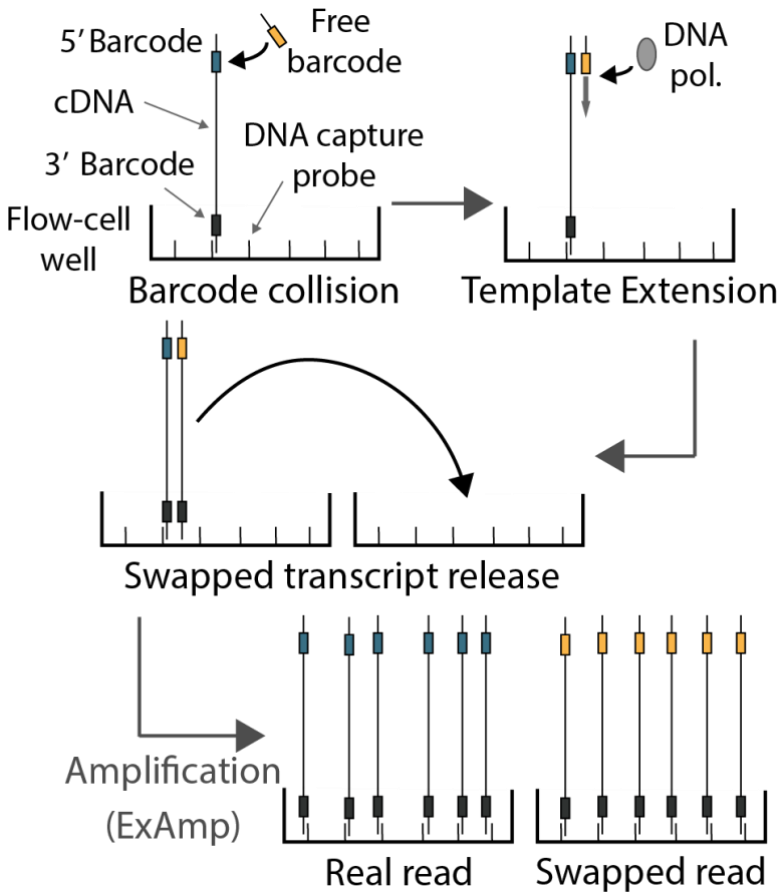


Figure 1.1 - Schematic of the mechanism of barcode swapping on the HiSeq 4000, as proposed by Sinha [59].

This "switching" of barcode labels (also called "hopping" or "swapping") is problematic for analyses of sequencing data. Reads labelled with a barcode specific to a given sample may have originated from any other multiplexed sample in the same pool, compromising the interpretation of the sample labels and their use in downstream analyses. This phenomenon is particularly relevant for metagenomics assays, where a large number of samples (i.e., differents soil) are necessarily multiplexed together for efficient use of sequencing resources.

James Hadfield's blog post [60] is the earliest identification of barcode swapping (December 2016). He also provides sensible experimental steps to reduce the severity of the switching. The [59] on bioRxiv uses single-cell RNA-seq protocols to identify the source of the swapping. Another [61], did not identify any increased switching on HiSeq X, using genome sequencing data from unbalanced heterozygotes. Illumina [57]

summarising the issue. [62] describe a method to deconvolve the effects of swapping from plate-based scRNA-seq assays, and suggest a method for estimating swapping rates using the distinctive "crosshair" patterns. [63] do not identify excessive barcode swapping on HiSeq X (they also have an excellent and in-depth introduction to the barcode swapping phenomenon). [58] have produced a very comprehensive overview of the swapping they have observed in their lab for a variety of bulk sequencing studies. Unfortunately, a bioinformatic tool to address this issue, have not yet been developed.

In the following thesis we will describe a tool, called iSwap that addresses these problems. We will quantify the effect of barcode switching in a variety of metagenomics datasets and we show that switching can create artefacts and false positive sequences in experiments.

State-of-the-art

2.1 The soil: an unexplored source of genes

Bacteria are a widespread and an essential presence in nature. They are the most ancient and common form of life on Earth since in the soil lives over 95% of biodiversity and such diversity can be attributed to microorganisms. Millions of microorganisms, lots of them still unknown, live in a single gramme of soil. They can be independent or live in symbiosis with other organisms [1].

Bacteria are involved in:

1. the transformation process of products such as bread, wine, beer;
2. ecological process like nitrogen fixation;
3. the equilibrium of biogeochemical cycles;
4. the degradation of complex polluting molecules;

The biodiversity of the microorganisms, including the chemical and metabolic processes involved, has a central role in maintaining the ecosystem in functional and efficient conditions. The equilibrium constructed in the microbic ecosystem, thanks to the stabilisation of the functional relations between different microorganisms, affects positively the plants and consequently the entire animal community [1]. Obviously, besides the positive effects here listed also exist negative effects, that are all the bacterial infective pathologies. One of the pathologies with the highest infective rate is tuberculosis, caused by the Mycobacterium tuberculosis, that kills approximately two millions person each year, especially in sub-Saharan Africa. The pathogenic bacteria contribute also at other world-spread relevant diseases, such as Streptococcus and Pseudomonas, that can cause pneumonia; or Shigella, Campylobacter and Salmonella enterica, wich can cause, food-related diseases. Other pathogenic bacteria can also cause infections like tetanus, typhoid fever, diphtheria, syphilis, leprosy, and other infections that provokes high mortality rates in newborns in third-world countries [2].

2.2 From the first microorganisms to metagenomics

The study of microorganisms began during the second half of 1600 with the improvement of the already existing optical microscopes by Robert Hooke, that also

discovered cells, but mostly at the research by Antoni van Leeuwenhoek, the first to demonstrate the existence of bacteria.

A great contribution was given by Pauster (200 years after Robert Hooke), and he is considered the founder of modern microbiology. His thesis demonstrates how in a sterile environment is not possible the creation of microorganisms ex Novo.

Robert Koch later (1876) was the first to grow a microorganism (*Bacillus anthracis*), out of animal organism and describe his life cycle. Koch also was the first to describe the role of a pathogenic agent on the onset of infective diseases. Koch's postulates were made from these observations.

New techniques and coloration protocols have been developed in the following years (eg. Gram's coloration protocol), that have allowed deeper studying of this new discipline, thus elevating greatly the solving power of microscope techniques. An important contribution was given by Carl Woese, who identified the Archea dominium in 1967, thus separating it from Bacteria dominium. Woese used molecular phylogeny techniques applicated to ribosomal RNA16s [3]; thus begins the era of studying bacteria through molecular analysis of the gene that codifies via lower ribosomal subunit (16S), now considered the fundamental region for the classification of bacteria and Archea. A decade after, the amplification techniques through polymerase chain reaction (PCR) and the creation of the first sequencing reaction by Frederick Sanger [4], boosted the studying of microorganisms using molecular techniques. Based on Sanger sequencing, different methods of PCR have been developed in order to study microbial communities. The study of microbial communities has an important role in the study of microorganisms: to comprehend the biology of a microorganism, the ecological context, and the way the microorganism interacts with other microorganisms must be understood [5]. The obstacle in knowing the growing conditions for each microorganism made it necessary the development of molecular methods, able to mark the cultivable or non-cultivable microorganisms, through the analysis of genetic sequences [6]. The Denaturing Gradient Gel Electrophoresis techniques (DGGE), and the terminal restriction fragment length polymorphism techniques (T-RFLP) have been developed to analyse the microbial communities not marked in different environments [7], [8]. The introduction of molecular techniques allows marking the microbic diversity in terms of

abundance based on the DNA extraction. The first experiments in this section have been conducted by Norman Pace, that used PCR to value the diversity of ribosomal RNA sequences [9]. The results of these studies brought Pace to develop the idea of cloning the DNA directly from the environment of samples already in 1985.

Without recognising it, Pace had realised the first metagenomic experiment through the random isolation and clonation of DNA from an environmental sample.

The term metagenomics was proposed for the first time in 1998 by Jo Handelsman, Jon Clardy, Robert M. Goodman, to specify a theoric collection of all the genomes belonging to the members of a microbial community of a specific environment [10].

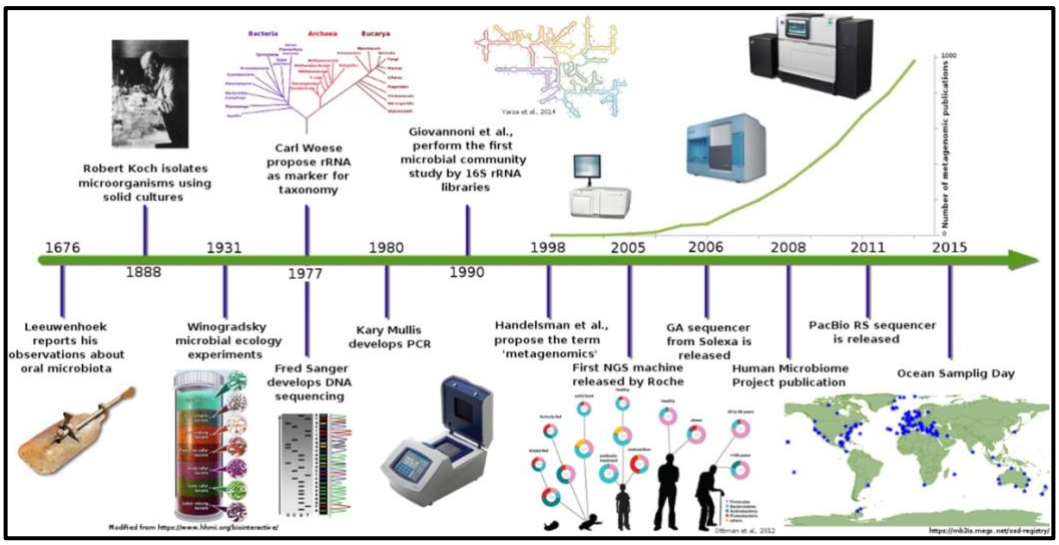


Figure 2.1 - History of metagenomics. The timeline shows the evolution of the studying method applied to microbic communities from Leeuwenhoek to modern era [11], [12].

2.3 A New World: metagenomics

Metagenomics is a genomics branch that studies complex microbial communities directly from their natural environment, avoiding the growth on culture medium.

Usually growth medium passage allows the identification of just 1-3% of the living microorganisms, causing the loss of around 97-99% of biodiversity information [13], [14]. Specific growth conditions exist for each microorganism (eg. the necessity of

specific metabolites or anaerobiosis conditions): it is impossible to identify all the microorganisms existing in a sample through the analysis of cultivation methods.

Metagenomics is a technique based on extraction and sequencing from the whole sample. In particular, targeted metagenomics is based on the analysis of 16S ribosomal DNA of bacteria.

With this method, it is possible to study bacteria communities in different habitats. The metagenomic analysis allows the identification of every single microorganism belonging to the community.

The applications of metagenomics are various: ecological, agricultural, agroindustrial, medical, zootechnical. The majority of these studies describes the existing bacteria communities, the interactions between these communities and with the environment or the hosting organism, in order to comprehend the complex biological processes they are involved in **Figure 1.2** shows some examples of environments in which metagenomic studies are applied.

Figure 2.2 – Representation of studying field for metagenomics.

2.4 Next Generation Sequencing Platforms (NGS)

The past sequencing techniques, known as "first generation sequencing", were based on extension strategies through primer for specific sequences. This method was determined

by Ray Wu at the University of Cornell in 1970. DNA polymerase and specific nucleotides marked with fluorophores have been used to sequence the cohesive extremities of lambda phage [15]–[17]. Between 1970 and 1973, Wu, Padmanabhan and other scientists demonstrated that this method could be applied to every sequence of DNA using specific synthetic primers. Sanger took this primer-based strategy to develop another method that could improve sequencing speed [4]. The first signals of the technological revolution in sequencing world began in 2005 with a publication on sequencing technique developed by 454 Life Science [18] and the polony-multiplex sequencing protocol by George Curchís lab [19] Both groups used a strategy that reduced enormously the quantity of reagents, amplifying the sequencing reactions. The strategy implied the simultaneously sequencing of hundreds of thousand fragments both on agarose and glass supports. Thus, it was possible to reduce sequencing reactions, with an increasing on the productivity of a capillary sequencer. Furthermore, avoiding to build libraries through cloning methods significative reduced costs and time necessary to sequence great-dimension genomes.

Said sequencing technologies, defined as NGS (Next-Generation Sequencing) are constantly evolving, and described in various reviews [20]–[22].

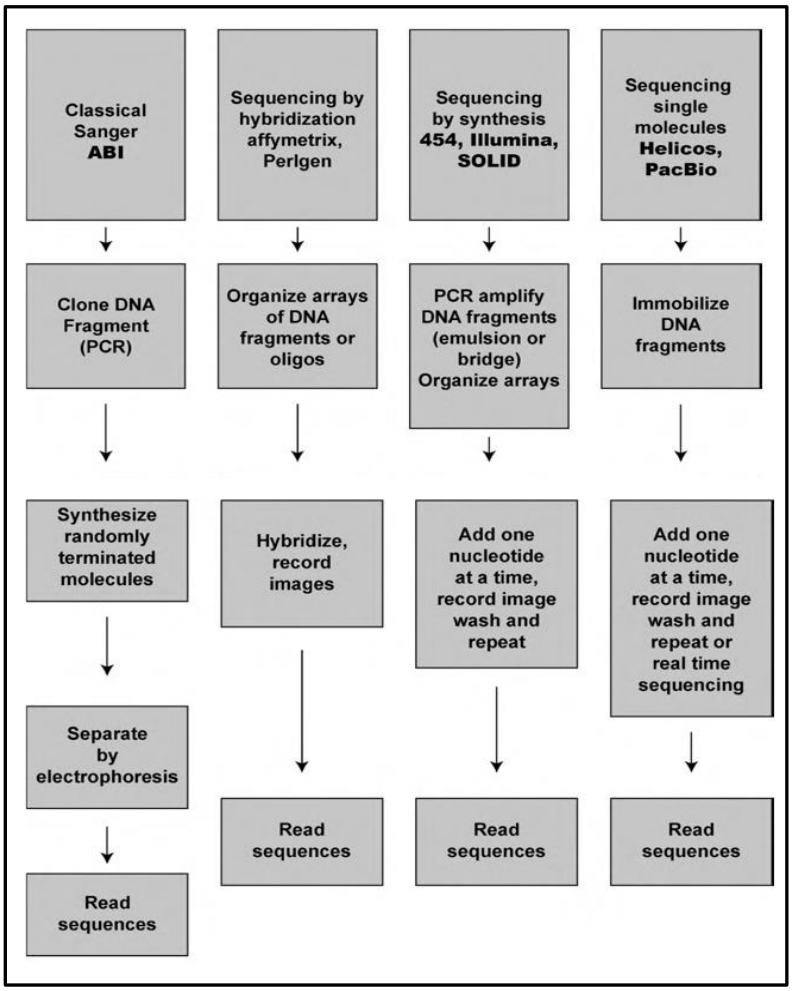


Figure 2.3 – *Different sequencing-methods* [23].

Sequencing technologies provide amplication phase and sequencing phase. In the amplication phase sequences are amplified, and a cluster of identic sequences is obtained for each sequence. A primer is periodically extended by one or more nucleotides every time and the resultant sequence is read at every step of DNA synthesis. This strategy differs from Sanger's, where an entire group of DNA molecules partial clones are synthetized and then analysed.

Figure 1.3 shows a flowchart of different sequencing methods.

Such methods differ in the strategy used to amplify sequences, in chemicals and length of resulting sequences, called read. They do share the faculty to sequence at least millions DNA fragments simultaneously [23].

2.4.1 Comparing Sanger sequencing to NGS

Advantages of NGS platforms are various, first of all, the increasing of productivity, modern machines reach a productivity level of 100 Gbp per race, compared to 70-100 kbp of the first automatic sequencing machines [24]. This feature was improved due to the simultaneous sequencing of hundreds of thousand, or millions of samples. The second advantage is due to the elimination of cloning phases preceding the sequencing procedure, replaced by annealing of specific adapters and followed by amplification (that has been eliminated in some sequencing techniques) thus reducing errors done by PCR. The reduction of volume reaction and the faculty of parallel sequencing reduced reagents necessary and consequently, the costs. NGS technologies simplified sequencing strategy, reduced artefacts, increased fastness in sequencing genomes and reduced costs, thus making possible the analysis of much more samples [24].

NGS technologies present some limitations for which they cannot be used for every type of sample. NGS produces short sequences, that implies errors since in genomes we have repeated sequences and there is necessity to increase covering to avoid mistakes. During libraries construction, it is necessary to follow fundamental procedures (fragmentation, adapters bonding, purification, PCR amplification etc.) in which it is possible to generate errors. Other limits are common mistakes and artefacts obtained during sequencing. Every method has its downside: for instance with 454 and Ion Torrent technology it is common to have errors when it comes to analysing short homopolymers. In conclusion, every method has its positive and negative features and must be chosen accurately considering the sequence to analyse and the result expected.

2.4.2 Illumina/Solexa sequencing

The most employed NGS platform is the Illumina/Solexa. The genetic material is fragmented and used to build a library, then the library is amplified. The molecules are denatured to form single filaments. After the association of specific linkers to the DNA sequences, they are transferred on a solid support, together with oligonucleotides complementary to adapters. The amplifying molecules process (the bridge PCR process) begins on the plate: both linkers bond themselves to the support, thus giving the DNA a bridge shape. On said structure is executed a PCR reaction, forming a double-shape

DNA filament. The molecule is denatured again to form two single filaments and the process is repeated cyclically until clusters of thousand copies of the same molecule are formed in an extremely reduced space. DNA molecules are denatured and one of the two filaments (reverse filament) is eliminated through a specific reaction, thus obtaining single-filament DNA cluster. The amplifying process is showed in figure 1.4

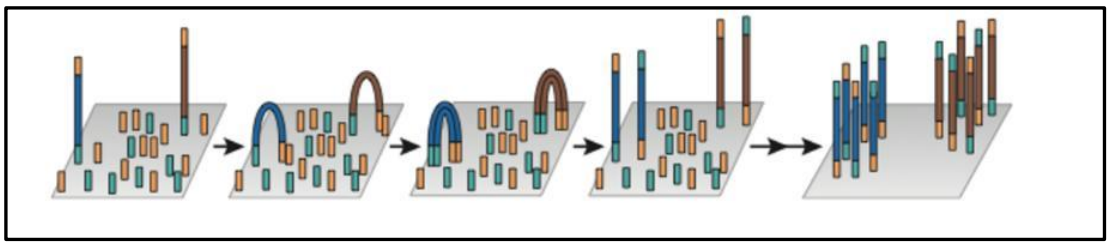


Figure 2.4 - Amplifying process of the Illumina technology (PCR bridge). The fragments with adapters (in yellow and green) bond with complementary oligonucleotides on the plate in a bridge shape. The amplification phases produce the complementary filament, it follows a denaturation, thus making the process be repeated cyclically. [21].

To begin the sequencing process the annealing of sequencing primer must be executed. The sequencing process is based on the cyclic reversible termination method or by sequencing. A sequencing cycle takes DNA polymerase and dNTP, to whom a fluorophore group (every nitrogen base has a fluorophore that can generate fluorescence at a specific wavelength in the visible spectrum) and an end group in 3' position. The end group prevents the polymerization of much nucleotides on synthesis filament, then at every cycle, only one nucleotide is added to each cluster, where is the complementary base on template stamp. A laser is projected on the plate, that stimulates the fluorescence of 3-position nucleotide, on each DNA fragment in synthesis. The image is registered and the end group on 3 position is eliminated, making the process continue cyclically. A scheme of the sequencing process is showed in figure 1.5.

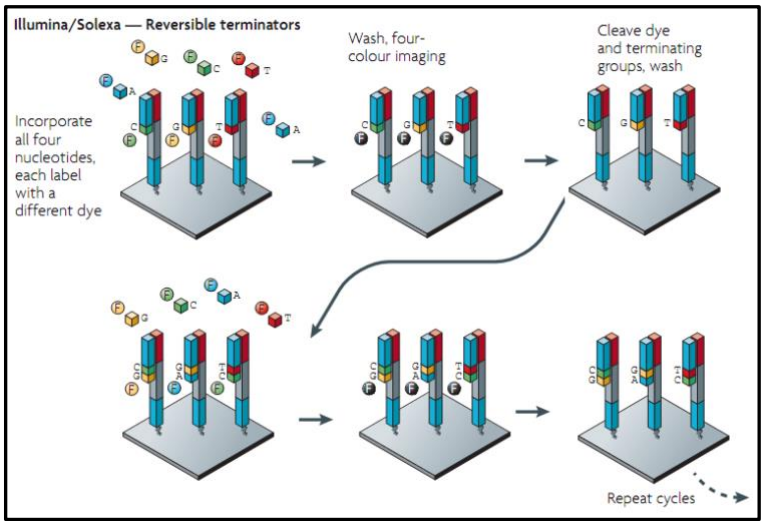


Figure 2.5 - Sequencing process using "by-sequencing" method of Illumina technology. Every molecule in figure represents a different cluster. The first line represents the first sequencing cycle. Nucleotides with specific fluorophore are added on synthesis filament. The light intensity is recorded and the elimination of 3-position group, then a new cycle starts. Modified by Metzker [24].

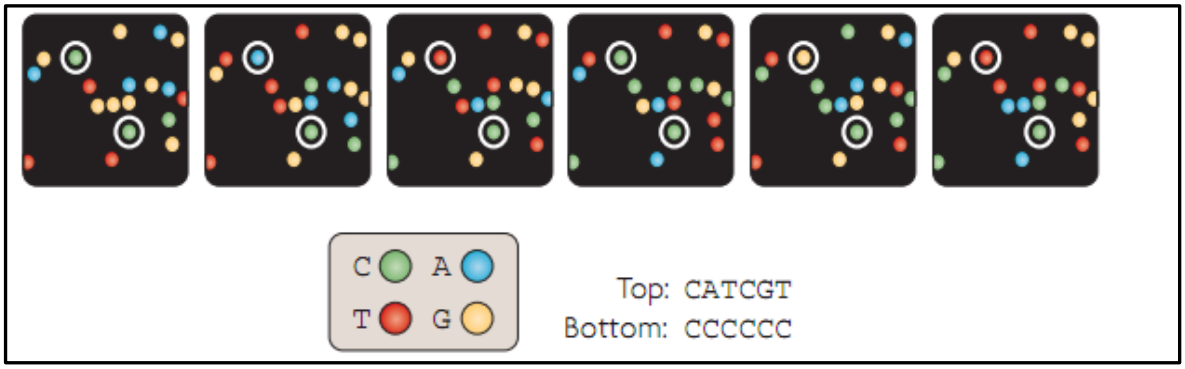


Figure 2.6 - Schematic representation of fluorescence interpretation and reconstruction of nucleotide sequence. Modified by Metzker [24].

Raw data obtained thanks to the sequencer are short sequences called read, whom sequences are registered in FASTQ files. A specific score is assigned to each base, the Phred Score, indicating the base quality. A filtering phase eliminates the low-quality bases, generally located at 3' position of every read.

2.5 Shotgun metagenomics and amplicons analysis

Microbic communities can be analysed considering their taxonomic complexity or their biological functions. There are two different strategies:

1. Amplicon 16S analysis, also known as "targeted metagenomics"
2. Metagenomic analysis also is known as "shotgun metagenomics"

Amplicons analysis is the most employed strategy to mark microbic communities considering only their taxonomic complexity. This method allows obtaining information on how many and which microorganisms exist in a sample, without informing on biological functions of the microorganisms. Said method allows defining the microbiota, the ensemble of all the microorganisms existing in the sample. A community is marked by its originating environment (eg. water, soil, biological tissue) and its DNA is extracted from all the cells existing in the sample. A taxonomic marker, common to every microorganism is isolated and amplified by PCR. The resulting amplicons are then sequenced and marked through bioinformatic analysis to define what and how many microorganisms are in the sample. Considering the analysis of bacteria and archaea, the amplicon analysis aims at the sequencing of codifying gene for RNA (16S), that together with other ribosomal RNAs and proteins, constitutes the minor unity of ribosome. Said gene, in both domains, appears to be a taxonomic and phylogenetic marker [25], [26]. This method revealed millions of microorganisms living on Earth [27]. The comparisons of 16S data in different samples highlight the relation between microbic diversities and different environments. Considering microbiota, a lot of studies allowed the comprehension of the interactions host-organism and the illness mechanisms associated with it [28], [29]. However, this analysis presents limits. First of all, it could not resolve a great part of diversity in a community due to problems with PCR [30]–[32]. Second, it has been demonstrated that the analysis of different regions of 16S gene, conducted on same samples, leads to different results of microbic diversity because different regions have different solving powers for taxas (Jumpstart Consortium Human Microbiome Project Data Generation Working, 2012; [33]. Furthermore, casual sequencing errors and non-correct amplicons assembling, eg. chimeras (a chimera in genetics is a single sequence of cDNA originated by two transcripted, considered as a contamination of the two transcripted [34], can lead to the production of artificial sequences difficult to

identify [35]. Third, the amplicon analysis only provides information on the taxonomic composition of microbic communities. It is impossible to comprehend the biological functions associated to taxas using this method. In some cases, phylogenetic reconstruction can be used to understand biological functions codified in a genome containing a specific 16S sequence [36]. The accuracy in determining the diversity of the microbiome depends on how accurate the genomic sequences are represented in databases. Lastly, the amplicon analysis is restricted to taxas analysis through known markers sequences that can be amplified. New sequences or totally divergent sequences are difficult to study with this method [37]. A shotgun method is an alternative approach to the study of microbic communities. The DNA is extracted from community cells, but instead of amplifying a specific sequence, all the DNA is fragmented, amplified and sequenced. The reads are on landmark genomes. Some reads will be analysed to obtain taxonomic information (eg. 16S) others to obtain functional information on existing organisms. This approach provides the simultaneous opportunities to explore two sides of microbic community: the taxonomic one (which microorganisms exist in the sample) and the functional one (what genes are codified and what processes are made). Of course, this method has limits as well. Firstly, to analyse metagenomic data is very difficult. Most of these difficulties are for instance, it is impossible to determine accurately from which a single read is derived. Furthermore, most of the microbic communities are extremely complex and it is difficult to obtain a complete map of all the reference genomes existing in the community [38], [39]. Secondly, DNA data originating from host microorganisms and not relevant for the analysis, could exist in the metagenomic data. to this end, molecular and computational data have been developed to sift though the relevant data [40]–[42]. Third, when generic contaminations are present in analysis data [43]. To identify and eliminate contaminate metagenomic sequences is problematic [44]. It is, in fact, difficult to determine which reads originate from a contaminated genome, despite identification and software for cleaning of contaminated sequences [45]. To conclude, the shotgun analysis tends to cost more than the targeted analysis, especially in complex communities, or when the parasite DNA exist in large quantities in the sample.

2.6 Ribosomal gene 16S rRNA

Ribosomal gene 16S rRNA (fig. 1.7) is composed of 10 preserved regions and 9 variable regions (fig. 1.8), has a lower evolution rate and it is present in all bacteria. The variation rate of sequences of the genes codifying for rRNA is extremely low compared to other genes, thus causing restriction to the rRNA structure, that must take a secondary defined structure and must interact with different proteins to form a functional ribosome. Consequently, this region is used to determine phylogenetic relations on wide evolutive distances. This type of RNA works as a molecular clock and permits accurate determining of phylogenetic distances.

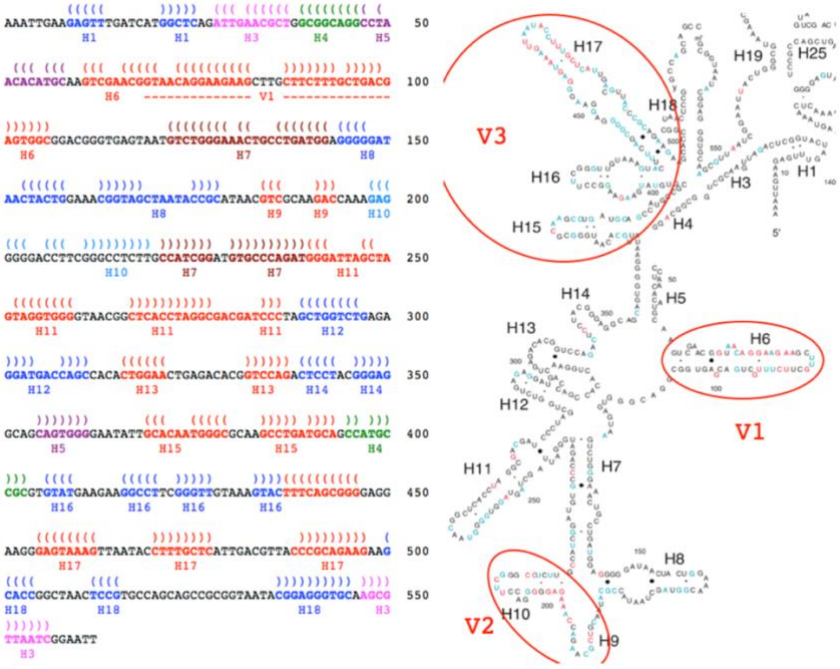


Figure 2.7 – Representation of variable regions V1, V2, V3 and genic sequence 16S rRNA [3].

The genic product of 16S rRNA is a region that constitutes the lower unit 30S of prokaryotes ribosomes [3]. Choosing the 16S rRNA gene as a phylogenetic marker to examine the microbic diversity and to identify and classify microorganisms is due to the difficulty in cultivating most of the microorganisms living in natural environments. Lane [9] firstly described the use of 16S rRNA gene to identify and classify non-cultivable microorganisms. In phylogenetics an Operational Taxonomic Unit (OTU) consists of a

taxonomic definition of a species or group of species, often used when only sequencing DNA data are available, basing on sequence similarity. The taxonomic assignment is based on the sequencing of the 9 variable regions of 16S rRNA gene (fig. 1.8). Species, genus, family and phylum are conventionally defined with a phylogenetic distance value of 0.03, 0.05, 0.10 and 0.20 respectively, basing on the entire length (1540 bp ca.) of the sequence of 16S rRNA gene [46]. Said method can lead to the uncorrected taxonomic assignation of **OTUs** (**O**perational **T**axonomy **U**nit), especially if the differences between genic sequences of different 16S rRNA are not equally distributed through the 16S rRNA gene, but are concentrated in some of the variable 9 regions [47]. This can happen if the analysis is selectively conducted on some variable regions. It has been demonstrated that some of the variable regions are more informative than others [48], thus allowing a taxonomic more correct than others [49], [50]. For global analysis of abundancy and diversity of complex microbiomes NGS platforms are employed [48], [51]–[53]. The variable regions, sequenced, are grouped in OTUs, with the same distance conventional value used for the integral sequence of 16S gene. In some recent studies, the single variable regions have been compared between themselves, comparing them with the entire sequence of the gene to estimate the resulting relative abundancies [51], [54], [55]. It has been demonstrated that the variable region affects the evaluation of relative abundancy of the OTUs. The analysis of regions V1-V2 (350 bp ca.) and the region V8 produce different OTUs homogeneities of the same sample taken from termites [53]. There is a crucial necessity to identify adapt variable regions, depending on the matrix analysed, that can provide relevant analysis of the microbiota [56].

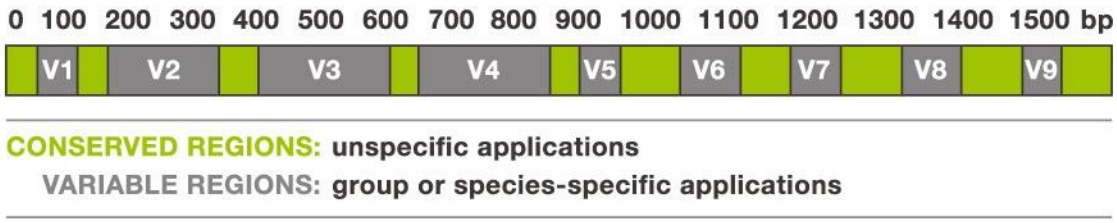


Figure 2.8 – Scheme of division of variable regions in genic sequence 16S rRNA.

It is proved that the variable regions do not always provide the same results if applied to different biological samples. In every matrix, depending on bacteria community the most informative variable regions must be identified. For example, faeces samples are more taxonomic informative, on variable regions V1-V3 [56]. In public databases are mostly represented the sequences corresponding to V1-V3 regions compared to the lower regions [56]. The partial sequences corresponding to this region will have more sequences to be compared to in the database, making the phylogenetic analysis easier.

iSwap: a bioinformatics pipeline for index switching in Illumina sequencing platforms

The NGS, in which millions of short DNA sequences are used as input to interpret biological phenomenon, has led to building stronger pipelines [64]. In informatics, the concept of a pipeline is used to indicate a group of software components, bond each one to others in a chain: the result of one of the elements is the input of the following one. The data flows through all the elements. Targeted metagenomics analysis, on 16S rRNA sequences, are commonly used to investigate on complex microbic communities [65]. Said analysis require specific bioinformatic tools to have a precise taxonomic view of the species, but with the emergence of the NGS platforms, it is necessary that each species be assigned to the correct sample in order to avoid misinterpretations of the results.

This thesis focuses on iSwap pipeline and how this tool is able to remove sequence artefacts due to index switching phenomenon.

3.0 Bioinformatics Pipeline

The bioinformatics pipeline iSwap, Figure 3.1, consists of several sequential steps that lead from raw sequencing data to the cleaned data without index switching.

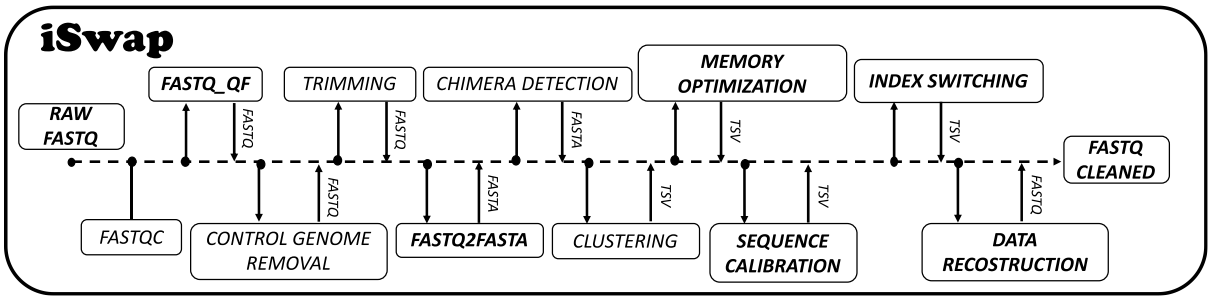


Figure 3.1 - iSwap, workflow. In bold custom programs.

The first step checks data in input (FASTQC) [66] and filters out the bad quality reads (FASTQ QF); adapters and control sequences are removed (CONTROL GENOME REMOVAL, TRIMMING)[67]; sequencing data are then converted in FASTA format

(**FASTQ2FASTA**) and prepared for chimera sequences detection (CHIMERA DETECTION)[68]. Thus, we proceed with the clustering phase where all the sequences are grouped by sequence identity (CLUSTERING)[69]; the next step is the memory optimization to downcast the memory usage and speed up the process (**MEMORY OPTIMIZATION**); after that, we have the sequence calibration to correct eventual sequencing errors (**SEQUENCE CALIBRATION**); finally, we are able to identify the index switching (**INDEX SWITCHING**); the last step is the data reconstruction to return to the initial file (**DATA RECOSTRUCTION and FASTQ CLEANED**).
In the following sections, main steps will be explored in detail.

3.1 Quality Controls and Filters

First of all, iSwap checks quality of raw sequence data coming from high throughput sequencing with FastQC [66], to provide graphical data reports. To filter out reads with low quality there are a huge list of third-party software but we decided to create a custom quality filter specific for our kind of microbial fragments, the bash script (**`fastq_qf.sh`**) available on GitHub. The filter consists in a window of 32bp (it contains the 12bp random barcodes, 8bp of barcodes for demultiplexing of samples and 12 for the UMI), Figure 3.2. This window is created with a parallel trimming tool (to improve performances), trimmomatic [67] and **`fastq_quality_filter.py`** a part of FASTX-toolkit [70], used with a set of parameters (-q 28 -p 95 -Q 33) then the program extracts only the high quality reads with **`fqextract_pureheader.py`**.

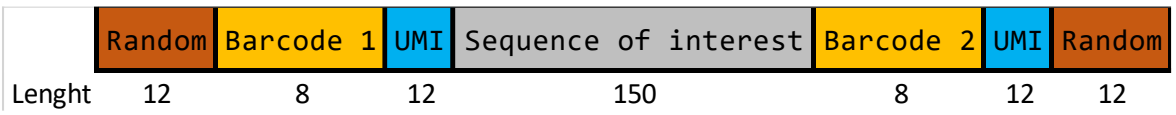


Figure 3.2 - Quality filter for reads

Thus, a list for high quality reads for 32bp of R1 and 12 bp of R2 is created but then the two lists are merged in one (with comm command in bash) and at the end only the good quality reads are maintained in the final FASTQs. To test the goodness of the quality

filter I run the new custom quality filter program on two different NGS sequencing runs (one with bad quality and the other with good quality, checked with FastQC), Figures 3.3, 3.4.

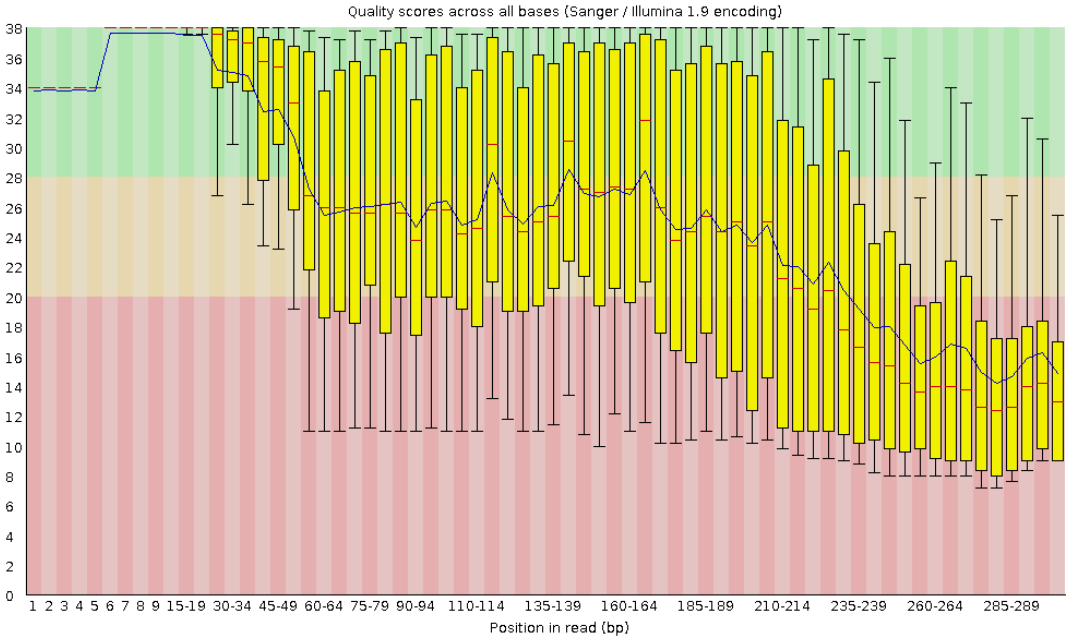


Figure 3.3 - FastQC indications in a bad run

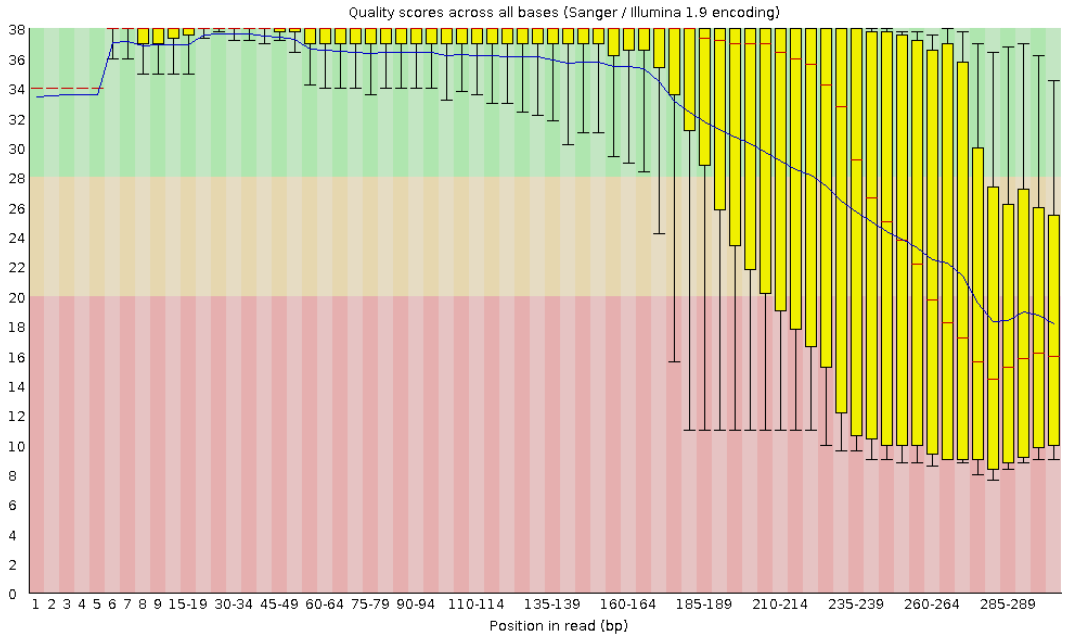


Figure 3.4 - FastQC indications in a good run

3.2 Adapter Removal and Trimming

Once the reads are filtered by quality the next step is to remove the reads mapping in PhiX and the first 12bp of random barcodes (not the TAGs).
PhiX is a reliable, adapter-ligated library used as a control for Illumina sequencing runs. It is also a quality control for cluster generation, sequencing, and alignment, and a calibration control for cross-talk matrix generation, phasing, and pre-phasing. Roughly for each run the amount of PhiX on the total is 25%. iSwap removes it aligning all the reads on PhiX genome with BWA-MEM (BWA with maximum exact match), producing a list of the reads that map on that genome and discard them using fqextract_pureheader. Generally, the 25% of the total reads are removed for PhiX.
The first 12bp of R1 must be removed from the read (instead the first 12bp of R2 are the TAG and must be removed from the read but conserved in a FASTA file for the quantification). To improve the performances, in term of time, I select trimmomatic (as described before) to use parallelism in server *Cerbero* (Appendix C).

3.3 Demultiplexing

Several samples are often sequenced at the same time, this technique is called multiplexing. To enable the redistribution of output reads into separate groups (demultiplexing), samples are tagged with individual barcode sequences.
We demultiplex out samples with **fastq-multx**, a part of the EA-Utils suite [71]. It identifies barcodes and uses them to demultiplex sequence data, producing a separate FASTQ file for each barcode. To demultiplex sequencing data, We developed a simple exact string pattern matching: the input is a list of barcode sequences that will be searched for at the beginning of each read (reads that do not contain any known tag are discarded). To avoid biases due to the possible misclassification of similar sequences, no mismatches are tolerated in this phase.

In a classic study of microbial community, at this point, we split the FASTQ file in multiple files, as described above, but to avoid index switching we will skip this step.

3.4 Chimera detection

Before chimera detection step, we convert the fastq file into a fasta format with a custom program called ***fastq2fasta.py*** available on GitHub. Thus, with a fasta file we are ready for the chimera detection. Chimeras are sequences formed from two or more biological sequences joined together. Amplicons with chimeric sequences can form during PCR. Chimeras are rare with shotgun sequencing but are common in amplicon sequencing when closely related sequences are amplified. Although chimeras can be formed by a number of mechanisms, the majority of chimeras are believed to arise from incomplete extension. During subsequent cycles of PCR, a partially extended strand can bind to a template derived from a different but similar sequence. This then acts as a primer that is extended to form a chimeric sequence [72] as showed in figure 3.5.

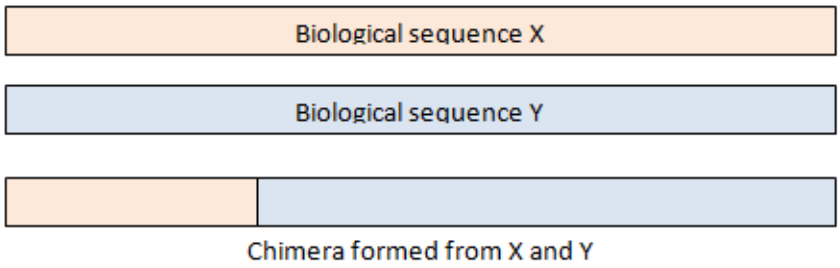


Figure 3.5 - An example of chimeric sequence formed

A chimeric template is created during one round, then amplified by subsequent rounds to produce chimeric amplicons. In 16S sequencing, we typically find that only a small fraction of reads is chimeric, perhaps of the order of 1% to 5%. However, when reads are clustered into groups of unique sequences, then we often find that a much larger fraction is chimeric [68]. For this reason, we choose to use UCHIME2 [68] for this step, a command line tool, in order to have as much as possible cleaned FASTQ from chimeras.

An example of the command line tool is given by:

```
usearch
```

- `uchime_ref reads.fasta`
- `db 16s_ref.udb`
- `uchimeout out.txt`
- `strand plus`
- `mode sensitive`

3.5 Creating OTUs – clustering

Clustering of OTUs is an essential step in microbial community analysis. Sequence errors, the presence of chimaeras, and the algorithm used for clustering significantly affect the quality of OTUs [73]. The term OTU is used to denote groups of microorganisms that share a high genetic sequence identity and are thus correlated considering phylogenetic. If sequences (reads) are grouped in the same OTU, it means that they have a sequence identity that characterises the common taxonomic rank between the two sequences.

OTUs can be generated through two different methods:

1. By aligning the reads on a reference database [74];
2. Through alignment between the reads themselves and thus without a reference database [75];

The alignment quality has a significant influence on the OTU clustering result [76]. Furthermore, it has been shown that alignment of sequences incorporating secondary structures generally increases the assignment of OTUs, at least as far as rRNA 16S sequences [77]. Alternatively, non-aligned cluster algorithms can be used, such as UCLUST [78] and CD-HIT [79]. These algorithms implement their own OTU creation process with an initial ordering of the sequences. In the case of UCLUST sequences are ordered in descending order of abundance, while in the case of CD-HIT in descending order of length.

Our work focused on non-aligned clustering algorithms; clustering algorithms were tested with usearch61, without references, and evaluated their effectiveness. In all cases, we use the **usearch** algorithm [78], but **usearch61_ref** align the reads on a reference database (reference-based) while **usearch61** does not use reference database for alignment (reference-free).

3.6 USEARCH

USEARCH is a clustering algorithm that is based on the principle that similar sequences tend to have "small words" in common.

These words have a fixed length k, and they are called k-mers. Compared to other programs using k-mers, USEARCH does not attempt to estimate the sequence identity based on k-mers correspondence, since the identity sequence correlates only approximately with the word count, especially for lower identities. Instead, USEARCH uses k-mer counts to target the database search (reference-based method) or between the sequences (reference free method). Consider the reference-based method on USEARCH.

For a query sequence as input, the sequences present in the reference database will be ordered by the number of k-mers found in common between the query and the target sequence, this value is indicated by the letter "U". Once the database is sorted, the first sequence, and then the one with the highest U-value, is selected and aligned with the sequence query. If the sequence identity value is above the threshold, the sequence query is inserted into a cluster that will have as centroid the sequence from the reference database. If the sequence identity between the query and the target sequences does not exceed the minimum threshold, then the comparison is done between the query and the next sequence in second position and thus has a slightly lower "U-value" than the first sequence analysed. The process continues iteratively but, to save time , if after “n” comparing with target sequences (usually a small number), none of them exceeds the minimum of alignment with the query sequence, the algorithm will move to the next query figure 3.6.

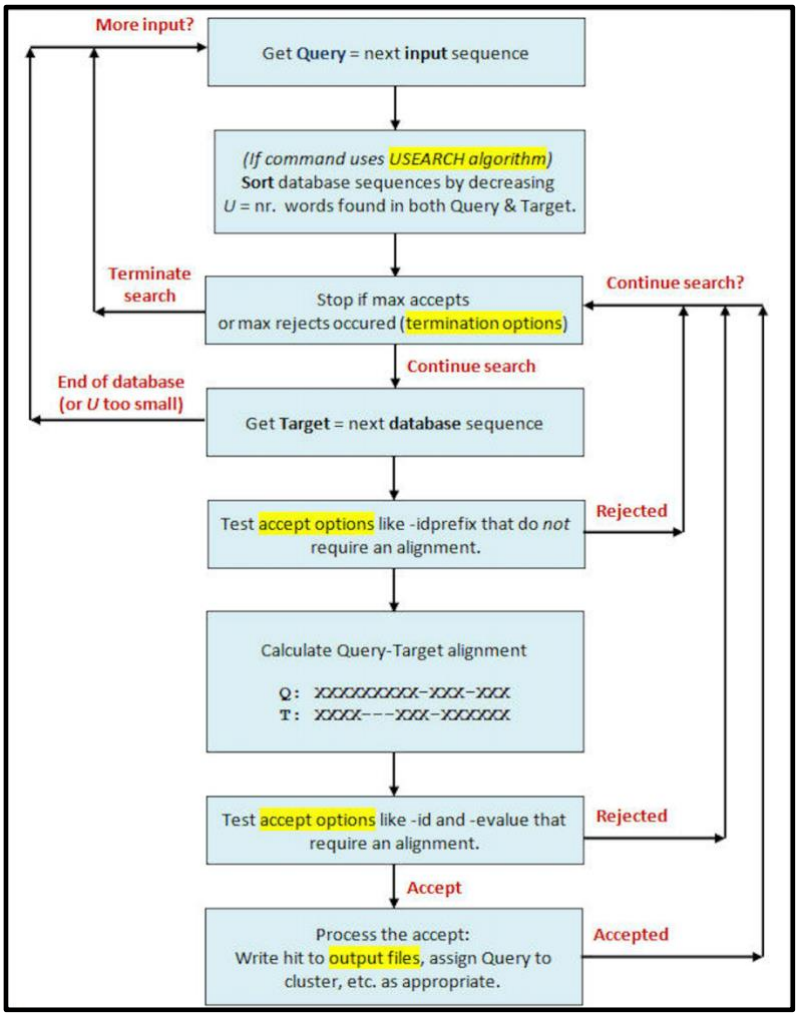


Figure 3.6 – USEARCH Flowchart Operation

If the query sequences are compared to the sequence targets in the U-ordinated way then we will have:

a) The first sequences of sorting are probably the sequences the most similar to the query:
b) In addition, as it analyzes sequences with a lower value of k-mers shared with the query, it will decrease the probability that the database has a suitable sequence for that cluster.

This means that the search can be performed extremely quickly if: (a) it meets the set threshold of identity, or (b) there is a certain amount of failures with only a slight loss of sensitivity. When the first query sequence is over, the second query sequence is then

analysed, and then the database is sorted considering the new k-mers identified in this second query sequence.

Termination options -maxaccepts and -maxrejects determine when the search has to stop. Maxaccepts is a value representing the number of maximum sequences that can be accepted within the cluster before passing to the next query, whereas maxrejects represents the maximum number of sequences that can be discarded before passing to the next query. This technique can dramatically improve speed, a decisive advantage for the large amount of data ever more widespread in biology. In the case of the reference free method, the sequences of the same dataset to be ordered respect to the U-value, and a sequence at a time, are compared to the dataset itself.

Clustering was executed with the "pick_otus.py" program of the QIIME pipeline [80]. An example of a command line is given by:

```
pick_otus.py    -m   [usearch61|usearch61_ref]
                -r   [Silva|Greengenes]
                -s   [0.95-0.99]
                –i   Dataset_Bioproject.fa
                -o   Picked_otus
```

The parameters indicate:

- m, otu_picking_method. Specify the clustering algorithm to use. Our study focuses on the use of usearch61 and usearch61_ref. [*default*: uclust]
- -r, refseqs_fp. Sequence files (reference databases) for comparison with the input file (-i). Both Silva and Greengenes have been used. Used only when the OTU clustering method is usearc61_ref.
- s, similarity. Threshold to classify two sequences in one OTU. By default, the program uses a threshold of 0.97.
- i, input_seqs_filepath. Sequence file.
- o, output_dir. Name of the output file.

After this operation we will get clusters as showed in figure 3.7.

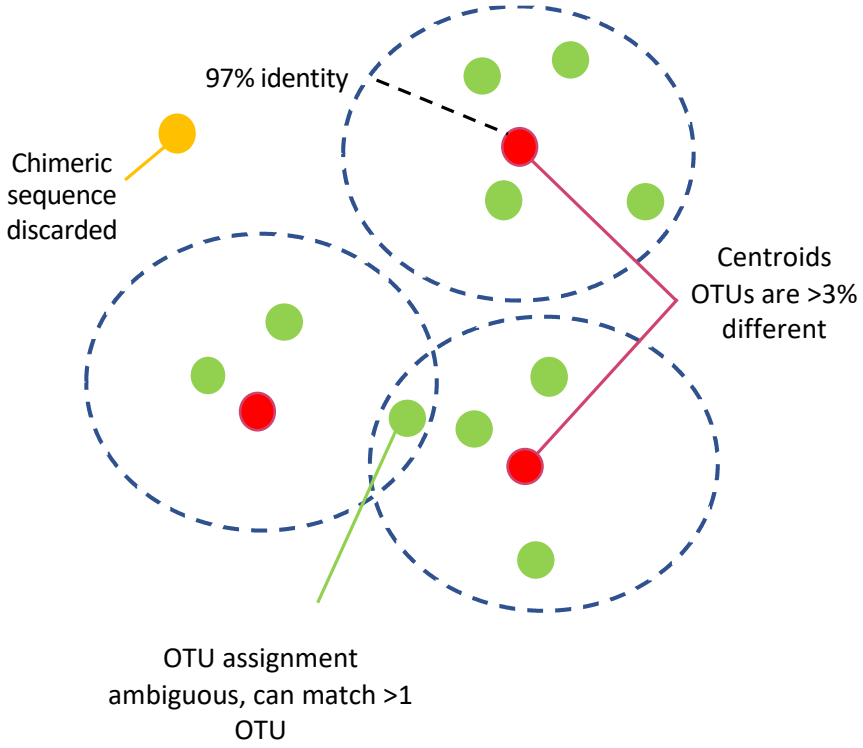


Figure 3.7 - OTU clustering. In red is the centroid sequence for each OTU in green the sequences inside the cluster with a sequence identity of 97%, in yellow the chimeric sequence discarded.

3.7 Choice of representative

The larger read in each OTU has been selected as a representative sequence, also called centroid. This step was performed by the "pick_rep_set.py" script of the QIIME pipeline [80] by setting the following parameters:

```
pick_rep_set.py-i   Picked_otus_file
                    -f   dataset_1kk_Seq.fa
                    -o   rep_set.fna
```

The parameters indicate:

- f, fasta_file. Sequences file given in input to the pick_otus.py

- i, input_file. Path for the OTUs file.
- o, output_dir. Path for the output *file*.

3.8 RAM required

After clustering phase, one key step is to load all the data into RAM memory to speed up the process of analysis, and this was a problem using python in the first year of my PhD, so I decided to optimize this step using the following tips. In this section, the example above derives from my first PhD year, so are not referred to microbial communities but to integration site analysis at the San Raffaele Hospital Milano.

When working in Python using pandas library, with small data (under 100 megabytes), performance is rarely a problem. When we move to larger data (100 megabytes to multiple gigabytes), performance issues can make run times much longer, and cause code to fail entirely due to insufficient memory. While tools like Spark can handle large data sets (100 gigabytes to multiple terabytes), taking full advantage of their capabilities usually requires more expensive hardware. And unlike pandas, they lack rich feature sets for high quality data cleaning, exploration, and analysis. For medium-sized data, we're better off trying to get more out of pandas, rather than switching to a different tool.

In this section, I'll explain about Python's memory usage with pandas, how to reduce a data frame's memory footprint by almost 90%, simply by selecting the appropriate data types for columns.

Import data in Python and taking a look at the first 7 rows.

```
In [0]:
import pandas as pd
input = pd.read_csv('dataset_analysis.csv', sep='\t')
input.head(7)
```

	Sample_ID	CHR	LOCUS	STRAND	Shearsite	RandomBC	Seq_Count
0	Sample_1	chr10	112643022	+	37	CGAAAGTCACAG	1.0
1	Sample_1	chr10	112643022	+	171	AGGGAGTCACAG	1.0
2	Sample_1	chr10	112643022	+	171	CCAAAGTTTTTAG	1.0
3	Sample_2	chr1	125445342	-	111	CCGGGGTCACCG	2.0
4	Sample_2	chr1	1233245	+	457	CAAACCTTTTCAG	1.0
5	Sample_3	chr16	23465234	+	158	TAGACGATCCACA	48.0
6	Sample_4	chr13	23115514	-	76	TCACAGCGACTTT	10.0

We can use the *DataFrame.info()* [81] method to give us some high level information about our data frame, including its size, information about data types and memory usage. By default, pandas approximate of the memory usage of the data frame to save time. Because we're interested in accuracy, we'll set the memory_usage parameter to 'deep' to get an accurate number.

```
In [1]:
input.info( memory_usage='deep' )

Out [1]:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8812147 entries, 0 to 8812146
Data columns (total 7 columns):
association_ID    object
shearsite         int64
randomBC          object
seq_count         float64
chr               object
locus             int64
strand            object
dtypes: int64(2), float64(1), object(4)
memory usage: 2.8 GB
```

Pandas has automatically detected types for us, with 2 numeric columns and 5 object columns. Object columns are used for strings or where a column contains mixed data types.

So, we can get a better understanding of where we can reduce this memory usage, let's take a look into how pandas stores data in memory.

Under the hood, pandas groups the columns into blocks of values of the same type. Here's a preview of how pandas stores the first twelve columns of our data frame.

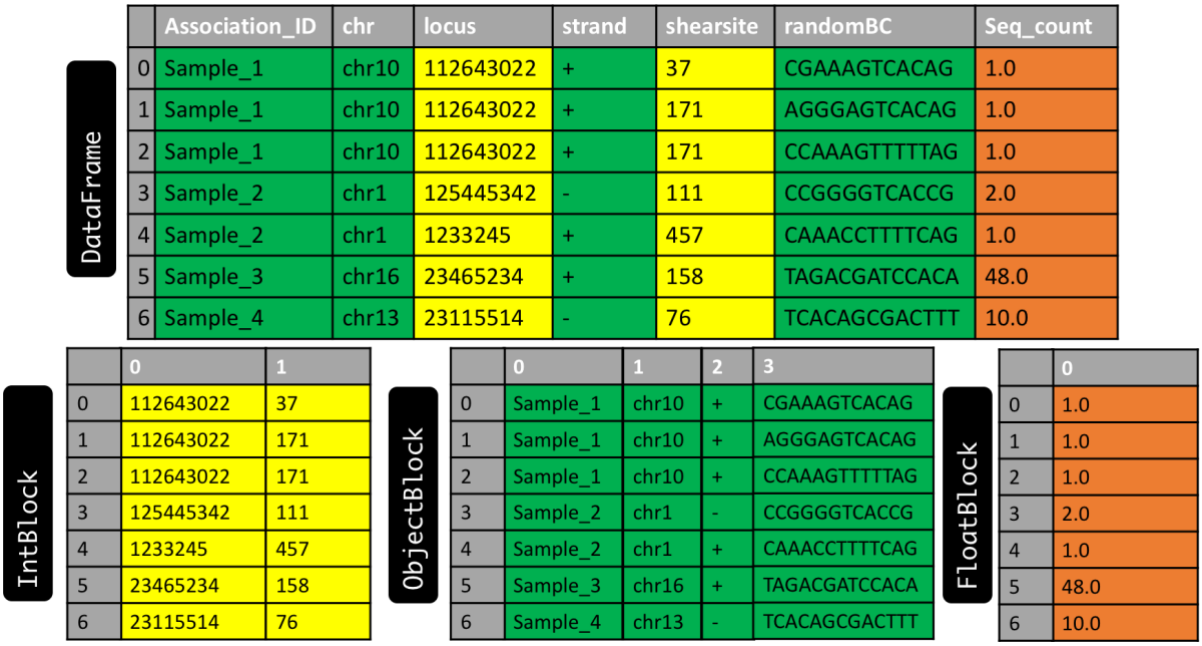


Figure 3.8 – Blocks rappresentation in pandas

The blocks don't maintain references to the column names. This is because blocks are optimized for storing the actual values in the data frame. The BlockManager class [81] is responsible for maintaining the mapping between the row and column indexes and the actual blocks.

Each type has a specialized class in the pandas.core.internals module. Pandas uses the ObjectBlock class to represent the block containing string columns, and the FloatBlock class to represent the block containing float columns. For blocks representing numeric values like integers and floats, pandas combines the columns and stores them as a NumPy ndarray. The NumPy ndarray is built around a C array, and the values are stored

in a contiguous block of memory. Due to this storage scheme, accessing a slice of values is incredibly fast. Because each data type is stored separately, I examined the memory usage by data type. So, at this point I checked the average memory usage for data type.

```
In [2]:
for dtype in ['float','int','object']:
        selected_dtype = input.select_dtypes(include=[dtype])
        mean_usage_b = selected_dtype.memory_usage(deep=True).mean()
        mean_usage_mb = mean_usage_b / 1024 ** 2
        print("Average memory usage for {} columns: {:03.2f}\ MB".format(dtype,mean_usage_mb))
Out [2]:
Average memory usage for float columns: 33.62     MB
Average memory usage for int columns:   44.82     MB
Average memory usage for object columns:529.44 MB
```

Immediately is showed that most of the memory is used by the 4 object columns.

3.8.1 Number optimization

As we mentioned briefly before, under the hood pandas represents numeric values as NumPy ndarrays and stores them in a continuous block of memory. This storage model consumes less space and allows us to access the values themselves quickly. Because pandas represent each value of the same type using the same number of bytes, and a NumPy ndarray stores the number of values, pandas can return the number of bytes a numeric column consumes quickly and accurately.

Many types in pandas have multiple subtypes that can use fewer bytes to represent each value. For example, the float type has the float16, float32, and float64subtypes. The number portion of a type's name indicates the number of bits that type uses to represent values. For example, the subtypes we just listed use 2, 4, 8 and 16 bytes, respectively. The following table shows the subtypes for the most common pandas types:

Memory usage	float	int	uint	datetime	bool	object
1 bytes		int8	uint8		bool	
2 bytes	float16	int16	uint16			
4 bytes	float32	int32	uint32			
8 bytes	float64	int64	uint64	datetime64		
variable						object

An int8 value uses 1 byte (or 8 bits) to store a value and can represent 256 values (2^8) in binary. This means that we can use this subtype to represent values ranging from -128 to 127 (including 0).

The `numpy.in` [82] class it's possible to verify the minimum and maximum values for each integer subtype. Let's look at an example:

```
In [3]:
import numpy as np
int_types = ["uint8", "int8", "int16"]
for it in int_types:
     print(np.iinfo(it))
In [3]:
Machine parameters for uint8
min = 0
max = 255
Machine parameters for int8
min = -128
```

```
max = 127
Machine parameters for int16
min = -32768
max = 32767
```

We can see here the difference between uint (unsigned integers) and int (signed integers). Both types have the same capacity for storage, but by only storing positive values, unsigned integers allow us to be more efficient with our storage of columns that only contain positive values.

3.8.2 Optimizing Numeric Columns with Subtypes

It's possible to use the function `pd.to_numeric()` [81] to **downcast** numeric types. Using `DataFrame.select_dtypes()` to select only the integer columns, then optimize the types and compare the memory usage.

```
# We're going to be calculating memory usage a lot,
# so I created a function to save some time.
In [3]:
# We're going to be calculating memory usage a lot,
# so we'll create a function to save us some time!
def mem_usage(pandas_obj):
        if isinstance(pandas_obj,pd.DataFrame):
                usage_b = pandas_obj.memory_usage(deep=True).sum()
        else: # we assume if not a df it's a series
                usage_b = pandas_obj.memory_usage(deep=True)
        usage_mb = usage_b / 1024 ** 2 # convert bytes to megabytes
        return "{:03.2f} MB".format(usage_mb)

In [4]:
input_int = input.select_dtypes(include=['int'])
converted_int = input_int.apply(pd.to_numeric, downcast='unsigned')
print(mem_usage(input_int))
print(mem_usage(converted_int))
```

```
Out [4]:
238.87 MB
70.48 MB


In [5]:
compare_ints = pd.concat([input_int.dtypes,converted_int.dtypes],axis=1)
compare_ints.columns = ['before','after']
compare_ints.apply(pd.Series.value_counts)
Out [5]:
       before  after
uint16     NaN    2.0
int64      2.0    NaN
```

There is a drop from 238 to 70 megabytes in memory usage, which is a more than 62% reduction. The overall impact on original data frame isn't massive though, because there are so few integer columns.

Thus, do the same thing with our float columns. At this point, all float columns were converted from `float64` to `float32`, giving us a 50% reduction in memory usage.

```
In [6]:
input_float = input.select_dtypes(include=['float'])
converted_float = input_float.apply(pd.to_numeric, downcast='float')
compare_float = pd.concat([input_float.dtypes,converted_float.dtypes],axis=1)
compare_float.columns = ['before','after']
compare_float.apply(pd.Series.value_counts)
print(mem_usage(input_float))      --->      67.00 MB
print(mem_usage(converted_float)) --->      33.00 MB
Out [6]:
         before  after
float32     NaN    1.0
float64     1.0    NaN
```

All float columns were converted from `float64` to `float32`, giving around 50% reduction in memory usage.

After I created a copy of the original data frame, assign these optimized numeric columns in place of the originals, and see the overall memory usage is now.

```
In [7]:
optimized_input = input.copy()
optimized_input[converted_int.columns] = converted_int
optimized_input[converted_float.columns] = converted_float
print(mem_usage(input))
print(mem_usage(optimized_input))
2848.00 MB -----> input
2731.00 MB -----> optimized_input
```

While I've reduced the memory usage of our numeric columns, overall I gained only reduced the memory usage of the data frame by 7%. Most of the gains are going to come from optimizing the object types.

Before, let's I focused my attention on how strings are stored in pandas compared to the numeric types.

The object type represents values using Python string objects, partly due to the lack of support for missing string values in NumPy. Because Python is a high-level, interpreted language, it doesn't have fine grained-control over how values in memory are stored [83]. This limitation causes strings to be stored in a fragmented way that consumes more memory and is slower to access. Each element in an object column is really a pointer that contains the "address" for the actual value's location in memory.

Below is a diagram showing how numeric data is stored in NumPy data types vs how strings are stored using Python's inbuilt types.

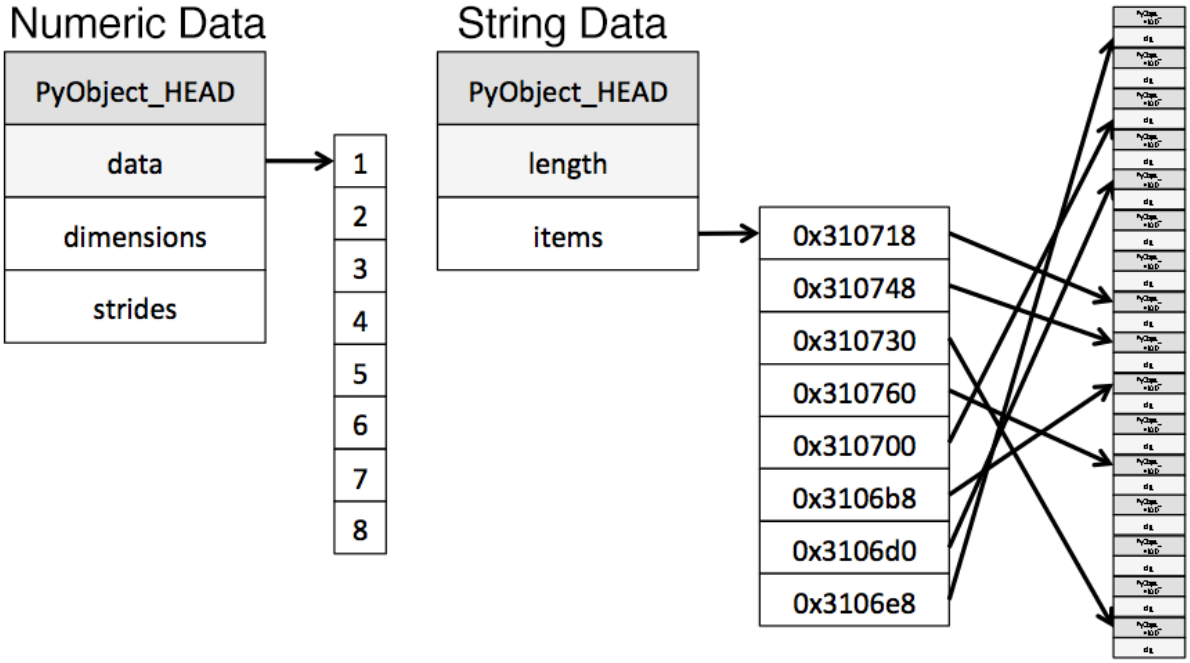


Figure 3.9 - Diagram from the post Why Python Is Slow.
(https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/) [83]

While each pointer takes up 1 byte of memory, each actual string value uses the same amount of memory that string would use if stored individually in Python. I used *sys.getsizeof()* to prove that out, first by looking at individual strings, and then items in a pandas series.

```
from sys import getsizeof
s1 = 'working out'
s2 = 'memory usage for'
s3 = 'strings in python is fun!'
s4 = 'strings in python is fun!'
for s in [s1, s2, s3, s4]:
    print(getsizeof(s))
60
65
74
```

```
74


obj_series = pd.Series(['working out',
    'memory usage for',
    'strings in python is fun!',
    'strings in python is fun!'])
obj_series.apply(getsizeof)


0 60
1 65
2 74
3 74
dtype: int64
```

The size of strings when stored in a pandas series are identical to their usage as separate strings in Python.

3.8.3 Optimizing object types using categoricals

Pandas introduced Categoricals [84] in version 0.15. The category type uses integer values under the hood to represent the values in a column, rather than the raw values. Pandas uses a separate mapping dictionary that maps the integer values to the raw ones. This arrangement is useful whenever a column contains a limited set of values. When it converted a column to the category dtype, pandas uses the most space efficient int subtype that can represent all of the unique values in a column.

Figure 3.10 - Conversion Representation using categories [84]

To get an overview of where might be able to use this type to reduce memory, I checked the number of unique values of each of our object types.

```
In [8]:
input_obj = input.select_dtypes(include=['object']).copy()
input_obj.describe()
Out [8]:
      association_ID      randomBC        chr           strand
count         8812147       8812147       8812147 8812147
unique        252           6118871       25            2
top           Sample_1      ATGAAGCACTCA  chr17         -
freq          205308        49      1220220          5104236
```

A quick glance reveals many columns where there are few unique values relative to the overall ~8.8 Milions in the data set.

Before dive too far in, I started by selecting just one of the object columns, and looking at what happens behind the scenes when I convert it to the categorical type.

Looking at the table above, only contains one unique value. To convert it to categorical just by using the astype() method.

```
In [9]:
dow = input_obj.chr
dow_cat = dow.astype('category')
print(dow.head())
print(dow_cat.head())


Out [9]:
0    chr10
1    chr10
2    chr10
3    chr10
4    chr10
Name: chr, dtype: object
0      chr10
1      chr10
2      chr10
3      chr10
4      chr10
Name: chr, dtype: category
```

```
Categories (25, object): [chr1, chr10, chr11, chr12, ..., chr9,
chrM, chrX, chrY]
```

Apart from the fact that the type of the column has changed, the data looks exactly the same. Giving a look under the hood at what's happening.
In the following code, I used the *Series.cat.codes* attribute to return the integer values the category type uses to represent each value.

```
In [10]:
dow_cat.head().cat.codes
Out [10]:

0 1
1 1
2 1
3 1
4 1
dtype: int8
```

Each unique value has been assigned an integer, and that the underlying datatype for the column is now int8. This column doesn't have any missing values, but if it did, the category subtype handles missing values by setting them to -1.
Lastly, looking at the memory usage for this column before and after converting to the category type.

```
In [11]:
print(mem_usage(dow))
print(mem_usage(dow_cat))
```

```
Out [11]:

416.00 MB

8.00 MB
```

I've gone from 416MB of memory usage to 8MB of memory usage, or a 98% reduction! Note that this particular column probably represents one of our best-case scenarios - a column with ~8Milions items of which there only 1 unique values.

While converting all of the columns to this type sounds appealing, it's important to be aware of the trade-offs. The biggest one is the inability to perform numerical computations. I can't do arithmetic with category columns or use methods like Series.min() and Series.max() [81] without converting to a true numeric dtype first.

I should stick to using the category type primarily for object columns where less than 50% of the values are unique. If all of the values in a column are unique, the category type will end up using *more* memory. That's because the column is storing all of the raw string values in addition to the integer category codes.

I wrote a loop to iterate over each object column, check if the number of unique values is less than 50%, and if so, convert it to the category type.

```
In [12]:
converted_obj = pd.DataFrame()
for col in input_obj.columns:
  num_unique_values = len(input_obj[col].unique())
  num_total_values = len(input_obj[col])
    if num_unique_values / num_total_values < 0.5:
      converted_obj.loc[:,col]=input_obj[col].astype('category')
    else:
      converted_obj.loc[:,col] = input_obj[col]
```

As before,

```
In [13]:
print(mem_usage(input_obj))
print(mem_usage(converted_obj))
```

```
compare_obj = pd.concat([input_obj.dtypes,converted_obj.dtypes],axis=1)
compare_obj.columns = ['before','after']
compare_obj.apply(pd.Series.value_counts)
Out [13]:
2647.00 MB
559.00  MB
            before  after
object         NaN    4.0
category       4.0    NaN
```

In this case, all object columns were converted to the category type, however this won't be the case with all data sets, so you should be sure to use the process above to check. What's more, memory usage for our object columns has gone from 2647MB to 559MB, or a reduction of 80%. Combining this with the rest of data frame and see where we sit in relation to the 2.6GB memory usage I started with.

```
In [14]:
optimized_input[converted_obj.columns] = converted_obj
print(mem_usage(optimized_input))
Out [14]:
643.00 MB


In [1]:
input.info( memory_usage='deep' )
Out [1]:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8812147 entries, 0 to 8812146
Data columns (total 7 columns):
association_ID    object
shearsite         int64
randomBC          object
```

```
seq_count          float64
chr                object
locus              int64
strand             object
dtypes: int64(2), float64(1), object(4)
memory usage: 2.8 GB

In [15]:
optimized_input.info(memory_usage='deep')
Out [15]:
<class 'pandas.core.frame.DataFrame'>
Int64Index: 8812147 entries, 0 to 8812146
Data columns (total 7 columns):
association_ID     category
shearsite          int16
randomBC           category
seq_count          float32
chr                category
locus              uint32
strand             category
dtypes: category(4), float32(1), uint16(1), uint32(1)
memory usage: 643.9 MB
```

So far, I've explored ways to reduce the memory footprint of an **existing** data frame. By reading the data frame in first and then iterating on ways to save memory, we were able to understand the amount of memory that we can expect to save from each optimization. As I mentioned earlier, however, often won't have enough memory to represent all the values in a data set. How can we apply memory-saving techniques when we can't even create the data frame in the first place?

Fortunately, it's possible specify the optimal column types when read the data set in. The `pandas.read_csv()` [81] function has a few different parameters that allow to do this. The dtype parameter accepts a dictionary that has (string) column names as the keys and NumPy type objects as the values.

```
In [16]:

dict =
{   'association_ID': 'category',
    'shearsite': 'int16',
    'randomBC': 'category',
    'seq_count': 'float32',
    'chr': 'category',
    'locus': 'uint32',
    'strand': 'category'
}
read_and_optimized=pd.read_csv('contamination_analysis.csv',dtype=dict,sep='\t')
print(mem_usage(read_and_optimized))
Out [16]:

643.00 MB
```

Now I can use the dictionary, to read in the data with the correct types.

By optimizing the columns, I've managed to reduce the memory usage in pandas from 2.8 GB to 643MB - an impressive 77% reduction!

Let's see in one of our case how the memory reduction works.. In figure 3.10 is showed the situation before the memory optimization, always in error due to the exceed of memory usage. The memory usage increases with increased amount of reads. After the memory optimization we get a massive 3.5 fold decrease of the memory usage as showed in figure 3.11.

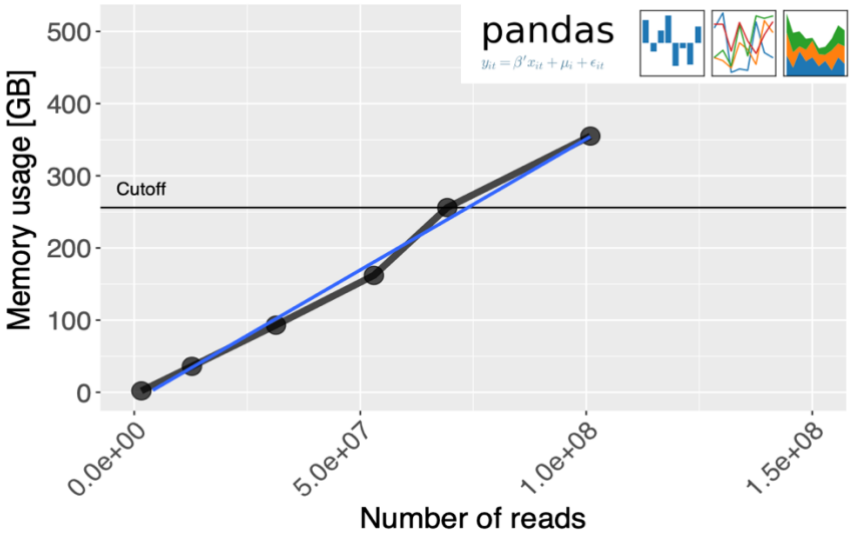


Figure 3.10 – Due to the excessive memory usage, something was unable, was unable for us performs normal actions. The memory usage increase with increased amount of reads

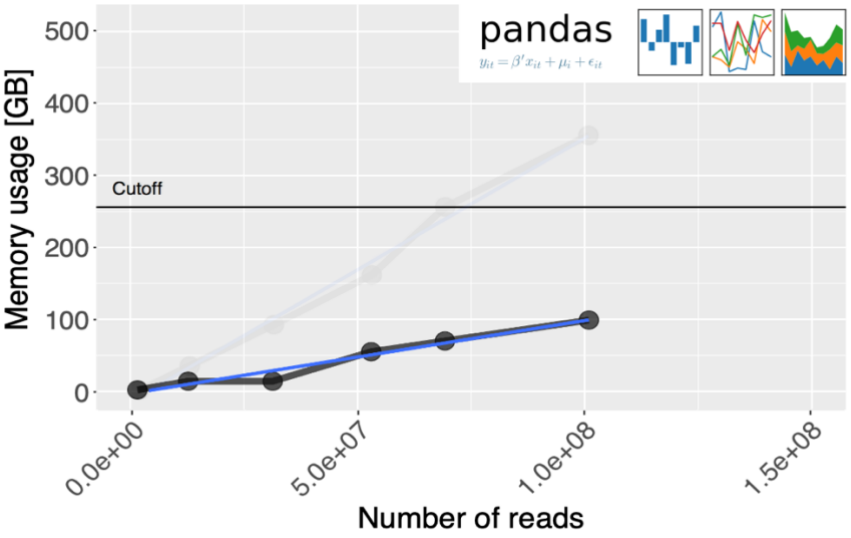
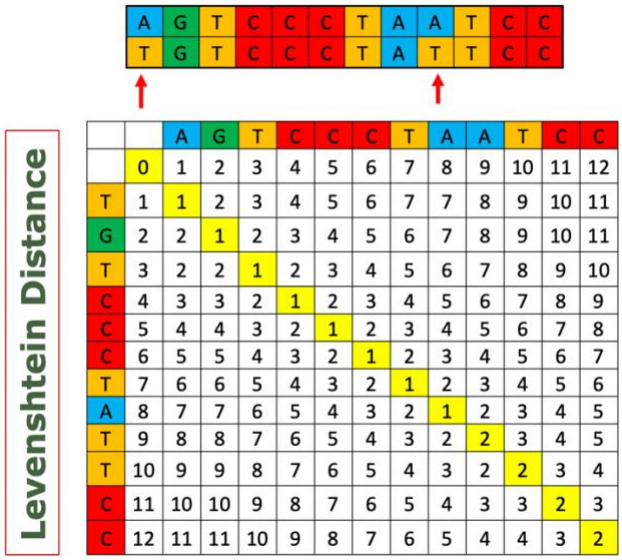


Figure 3.11 – Application of memory optimization measures. Thanks to this optimization we saved more than 300GB of memory usage.

3.9 Sequence calibration

After loading all data into memory, there is a very delicate phase that helps to avoid generating false positives. This phase is called sequence calibration: each OTU cluster is checked for possible sequencing errors due to the underlying sequencing machine setup. These errors are identified by means of the Levenshtein distance [85] . The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other. This type of errors increases when the length of the sequence read increases. For this reason, this correction is carried out only on a particular portion of the sequence inserted inside the read, called **UMI** (**U**nique **M**olecular **I**dentifier)[86]. UMI are a type of molecular barcoding that provides error correction and increased accuracy during sequencing. These molecular barcodes are short sequences used to uniquely tag each molecule in a sample library. UMIs are used for a wide range of sequencing applications, many around PCR duplicates in DNA and cDNA. UMI deduplication is also useful for RNA-seq gene expression analysis and other quantitative sequencing methods. [87]

In our case UMI is a random sequence of 12 nucleotides useful for uniquely tagging each sequence. By definition these sequences have a Levenshtein distance never less than 3 nucleotides. If, therefore, it should be that are find in a cluster of sequences (OTUs) 2 or more UMI sequences that have a Levenshtein distance less than 3, this is corrected because it is certainly a sequencing error.



		A	G	T	C	C	C	T	A	A	T	C	C
	0	1	2	3	4	5	6	7	8	9	10	11	12
T	1	1	2	3	4	5	6	7	7	8	9	10	11
G	2	2	1	2	3	4	5	6	7	8	9	10	11
T	3	2	2	1	2	3	4	5	6	7	8	9	10
C	4	3	3	2	1	2	3	4	5	6	7	8	9
C	5	4	4	3	2	1	2	3	4	5	6	7	8
C	6	5	5	4	3	2	1	2	3	4	5	6	7
T	7	6	6	5	4	3	2	1	2	3	4	5	6
A	8	7	7	6	5	4	3	2	1	2	3	4	5
T	9	8	8	7	6	5	4	3	2	2	3	4	5
T	10	9	9	8	7	6	5	4	3	2	2	3	4
C	11	10	10	9	8	7	6	5	4	3	3	2	3
C	12	11	11	10	9	8	7	6	5	4	4	3	2

Figure 3.11 - Edit distance matrix for two string of UMIs.

After this step each cluster it's ready for checking the present of index switching.

3.10 Index Switching removal and data reconstruction

The index switching detection and removal are described in the following. This is a very computationally intensive operation which was was parallelized using a MAP-REDUCE approach [88].

Each cluster is analyzed in parallel performing the necessary n^2 comparisons $O(n^2)$.

MapReduce is a framework for processing parallelizable problems across large datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware). Processing can occur on data stored either in a filesystem (unstructured) or in a database (structured). MapReduce can take advantage of the locality of data, processing it near the place it is stored in order to minimize communication overhead.

A MapReduce framework (or system) is usually composed of three operations (or steps):

1. **Map**: each worker node applies the map function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of the redundant input data is processed.
2. **Shuffle**: worker nodes redistribute data based on the output keys (produced by the map function), such that all data belonging to one key is located on the same worker node.
3. **Reduce**: worker nodes now process each group of output data, per key, in parallel.

MapReduce allows for the distributed processing of the map and reduction operations. Maps can be performed in parallel, provided that each mapping operation is independent of the others; in practice, this is limited by the number of independent data sources and/or the number of CPUs near each source. Similarly, a set of 'reducers' can perform the reduction phase, provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time, or that the reduction function is associative. While this process often appears inefficient compared to algorithms that are more sequential (because multiple instances of the reduction process must be run), MapReduce can be applied to significantly larger datasets than a single "commodity" server can handle – a large server farm can use MapReduce to sort a petabyte of data in only a few hours [89]. The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled – assuming the input data are still available.

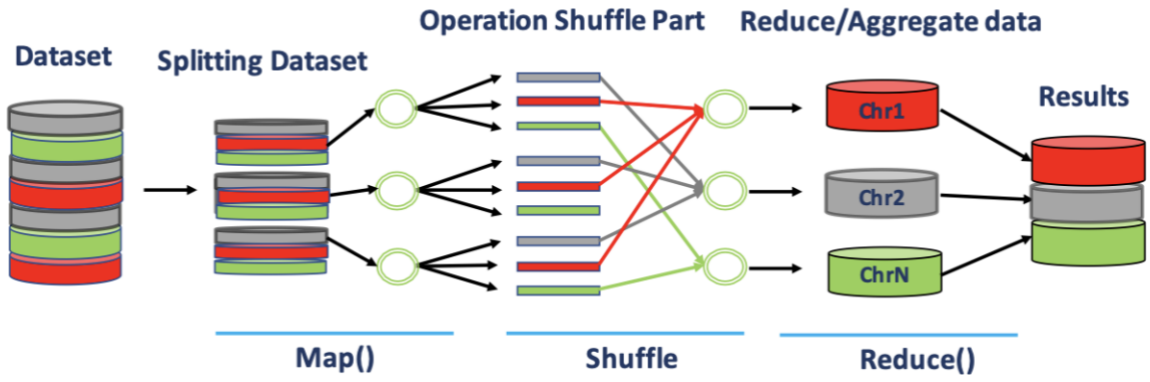


Figure 3.13 - MapReduce workflow.

Thanks to the MapReduce approach it's possible to analyze all the sequences a fast way. Figure 3.14 shows the result in terms of time required for this task obtained with MapReduce parallelization vs a regular sequential method.

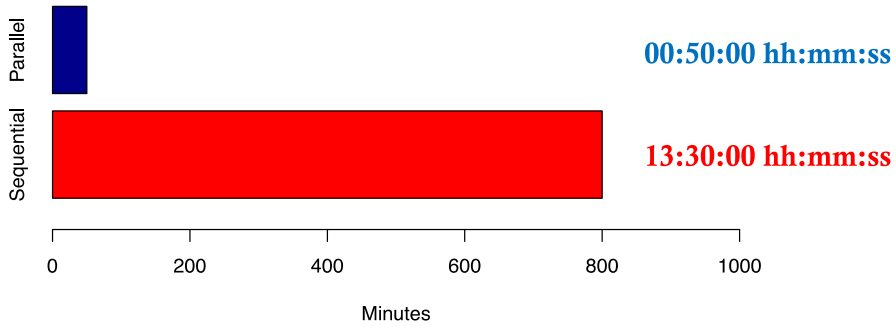


Figure 3.14 - Performances comparing Parallel method (blue) and sequential method (red) in time. Shorter bar is better.

Index switching can have two effects depending on whether the same sequence barcodes are present in both samples. If they are, the expression profile for each shared sequence barcode in the recipient sample will become a mixture with the corresponding profile in the donor sample. Otherwise, artefactual sequence may appear in the recipient sample, corresponding to the swapped-in sequence barcodes from the donor sample. This manifests itself as an increase in the number of shared sequence barcodes between samples.

UMI can help us to detect the present of index switching in addition to other useful information available like sequence length and which barcode are shared between sequences.

In the following example (figure 3.15) it's possible to understand how the mechanism to detect the index switching works. The table shows 3 different samples Yellow (E5), Blu (A5) and Violet (A3). Sample A5 contains 342 reads releated to a specific bacteria *Salmonella Enterica*, while sample E5 and A3 contains only 1 reads each of *Salmonella Enterica*. We know that samples A3 and E5 do not contain in origin this specific bacteria *Salmonella Enterica*. Infact, each sample E5 and A3 shared one barcode with the sample A5, but they shared also the same sequence length 73 and also the same UMI. Due to

the fact that the sequence length it's random for the sonication process (see Appendix B) and the UMI it's a random sequence of 12 nucleotides, this highlights the fact that this phenomenon it's not by chance but due to the index switching phenomenon. When the algorithm finds this case, it eliminates the reads assigned to samples E5 and A3 and assigns the sequences to sample A5.

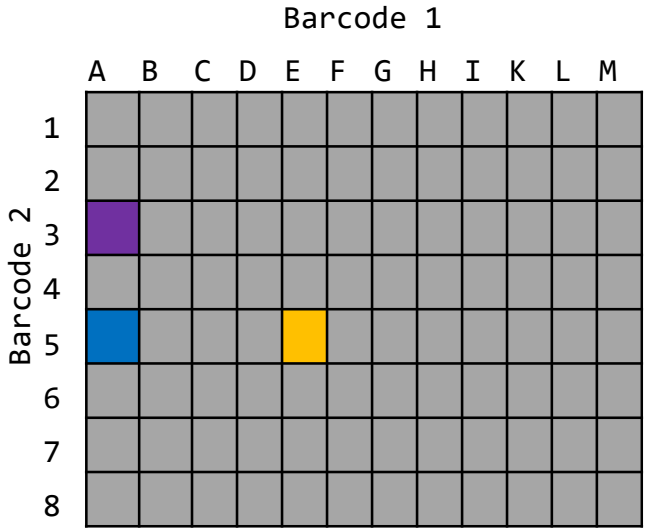


Figure 3.15 - Representation of how index switching spread around the dataset. A5 sample (blu - Donor) lose sequences in sample A3 and E5 (recipient samples).

Sample	Length	UMI	#Reads	Barcode 1	Barcode 2	Bacteria
Yellow	73	GCCTATGGTTGT	1	E	5	*S. Enterica*
Blu	73	GCCTATGGTTGT	342	A	5	*S. Enterica*
Violet	73	GCCTATGGTTGT	1	A	3	*S. Enterica*

Table 3.16 - Table of information for each cluster (OTU) obtained. In each cluster are collected the following information: sample name, sequence length, the UMI string, the number of reads for each sample, the barcode used for that sample, and the species of that bacteria belong to.

At the end, each cluster it's cleaned from the index switching and the FASTQ format it's reconstituted using ad hoc script developed for this task called **`tsv2fastq.py`** (not available on GitHub) that use all the information stored during the process of the pipeline to rebuild the file as in the beginning.

3.11 Index Switching assignment process

When index switching occurs and the difference in terms of sequence counts is not evident to easily assign the sequences to a particular sample as donor, we developed an advanced process to establish the source of contamination. Identification of the donor of contamination between samples is crucial step because we can't assign sequences to the wrong sample and risk to increase the false positive rate.

Given I the set of index switching, each i in I has a sequence count p . For each sample, we independently analyzed all index switching event: given S the set of samples, s the current sample and $s(n)$ all other samples $(S - s)$, for each patient s, for index switching i in Is we computed the ratio $\mathrm{p|is}/p|is(n)$, called index switching relative frequency (*iSRF*). We then analyzed the distribution of all *iSRF* in I to look for flexes and peaks.

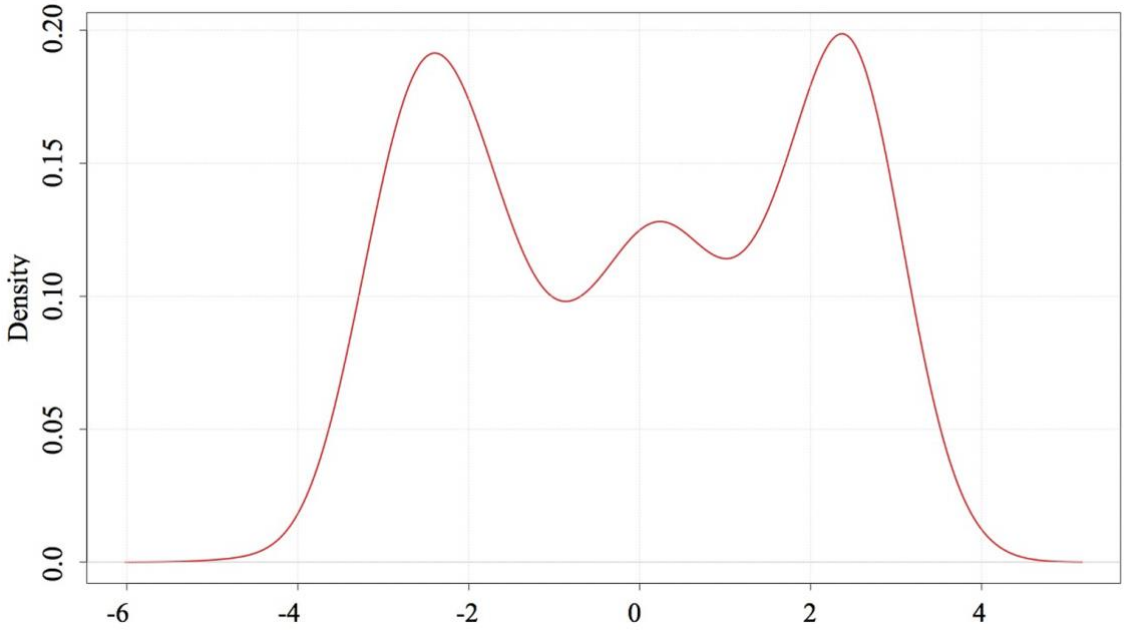


Figure 3.17 - Density plot of inter-sample index switching

Index switching scenarios between 2 samples

Index switching	Sample A	Sample B	Ratio A/B	Log10 A/B
1	1	100	0.01	-2.0
2	20	100	0.20	-0.7
3	100	100	1.00	0.00
4	150	100	1.50	0.18
5	250	100	2.10	0.40
6	1010	100	10.01	1.00

Table 3.18 - Theoretical use case scenarios for index switching between two samples.

Positive peak at +2 means that all index switching of this area carry sequence counts 20 times higher in the analyzed sample compared to the others. On the other hand, the index switching under peak at -2 have 20 times lower sequence counts in the same sample as compared to the others. The peak at 0 indicates that all these indexes switching have identical sequence counts among samples. The table below shows theoretical use case scenarios for two samples. Applying these theoretical scenarios to our empirical iSRF curve, we interpreted data as decision plot. Thus, the chosen threshold to set for contamination identification sample-based is 1, corresponding to 10 fold difference in linear scale.

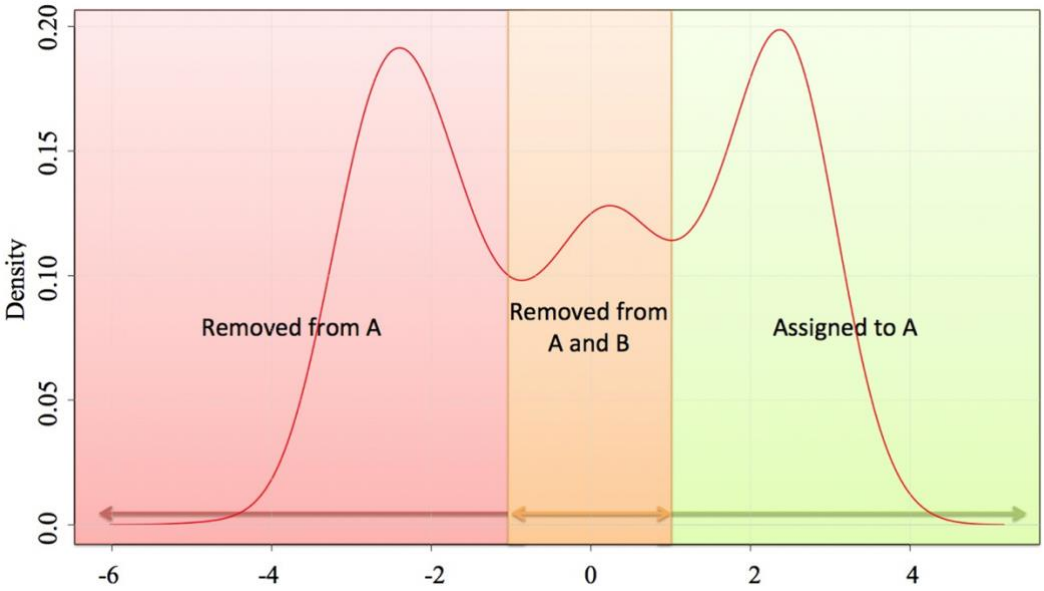


Figure 3.19 - Density plot of inter-samples index switching. Decision Plot.

Index switching scenarios between 2 patients

Index switching	Sample A	Sample B	Ratio A/B	Log10 A/B	Removed from A	Removed from B	Assigned to A	Assigned to B
1	1	100	0.01	-2.0	X			X
2	20	100	0.20	-0.7	X	X		
3	100	100	1.00	0.00	X	X		
4	150	100	1.50	0.18	X	X		
5	250	100	2.10	0.40	X	X		
6	1010	100	10.01	1.00		X	X	

Table 3.20 - Theoretical use case scenarios for index switching between two samples.

We obtained the results in Table 3.20, from Figure 3.19. We implemented this strategy to remove the index switching between datasets, comparing indeed the fold increase of a particular sequence in all conditions/cell population against the maximum frequency observed in other samples, describing the three possible outcomes previously described:

$$mFold = \left(\frac{\sum_{i=1}^{j} I}{max_{m=1}^{n}(I)} \right)$$

- $mFold > 10x$: Assigned to the current sample
- $1x < mFold < 10x$: Removed from the current sample and NOT assigned.
- $mFold > -10x$: Assigned to other sample

Results and comparison of tools for index switching

4.1 Why are we doing "data generation"?

Before continuing, let's define a few terms:

- Switching occurs between a **donor** library, from which the transcript originated, and **recipient** libraries, in which the swapped read is detected after sequencing (Figure 4.1).
- The **swapping fraction** is defined as the fraction of reads that have been mislabelled, from the set of all sequences reads in a pool of multiplexed libraries sequenced on a single flow cell lane.

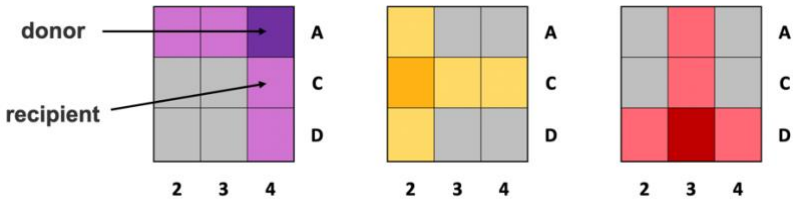


Figure 4.1 - Example of donor and recipient library, the dark colored cell is the donor, while the recipient library are the cells that share at least one barcode.

We consider two 96-well plates of 16s microbial communities. We used dual indices for sequence labelling, i.e., a different barcode was used at each end of the molecule. The barcodes used for each plate are from mutually exclusive sets - any barcode from one plate was never used on the other (Figure 4.2). For sequencing, all sequence libraries from the two plates were multiplexed.

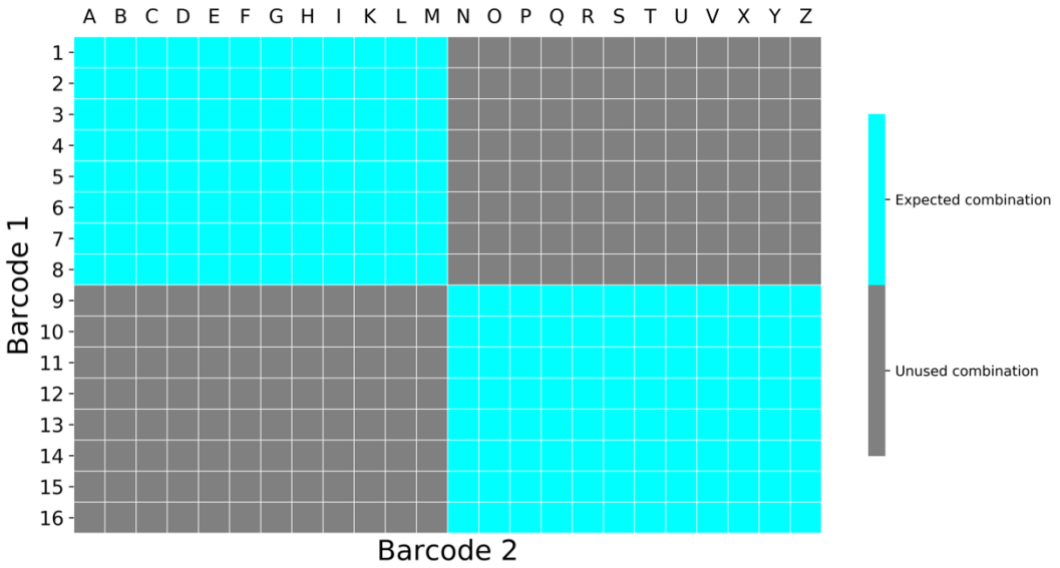


Figure 4.2 - Overview of the experimental design in our datasets. Each position of the plot represents a barcode combination. Each of the cyan blocks represents at 96-well plate. Barcode combinations in the grey positions were not used, and thus should not contain sequencing reads.

In these thesis, we take into account 192 soil samples. For these samples the taxonomic content was already known, thus the species of bacteria present in each individual soil was already known. This mean that the species of bacteria contained in sample H6 was known to be e.g. (*Pseudomonas aeruginosa*, *Escherichia coli*…ect.). Each soil sample was barcoded using a single available combination among those expected, as indicated in Figure 4.2. Briefly, soils were sorted into 96-well plates with 4 µ L of lysis buffer: 0.11% (v / v) Triton X-100 (Sigma), 12.5 mM DTT (Thermo Fisher Scientific), 2.5 mM dNTP mix (Thermo Fisher Scientific), and 2.3 U SUPERase In RNase inhibitor (Thermo Fisher Scientific). Annealing mix, composed of diluted ERCC RNA Spike-In Mix (Thermo Fisher Scientific) and 10 µ M oligo-dT30VN (Sigma), was added 1 µ L per well, and reverse transcription was performed using SuperScript II (Invitrogen). cDNA was amplified (23 PCR cycles) and purified with Ampure XP Beads (Agencourt) at 0.7 beads / 1 DNA (v / v). Library preparation was performed with the Nextera XT DNA Sample Preparation Kit using indexes from the Nextera XT Index Kit v2 Set A and Set D (Illumina). Libraries from each plate were pooled and purified with Ampure XP beads before quantification with the KAPA Library Quantification Kit (Roche). Library pools from each plate were combined in equimolar quantities. Library sequencing was

performed on an Illumina HiSeq4000, this operation was done for 100 times to study how the impact of index switching changed in each pool library.

4.2 Sequencing's library

In the absence of barcode swapping, it should be impossible to observe mapped reads with barcodes from each of the two different plates, i.e., there should be no mapped reads in the grey areas of Figure 4.2 as happen in case of Miseq sequencing showed in Figure 4.3. We will refer to these barcode combinations as "unused combinations", in comparison to the expected combinations in light blu as showed in figure 4.2. For the expected combinations where cells have been loaded, there are many mapped reads (Figure 4.2) as expected. In the unused barcode combinations, we observe a lower but non-zero number of mapped reads, consistent with the presence of barcode swapping.

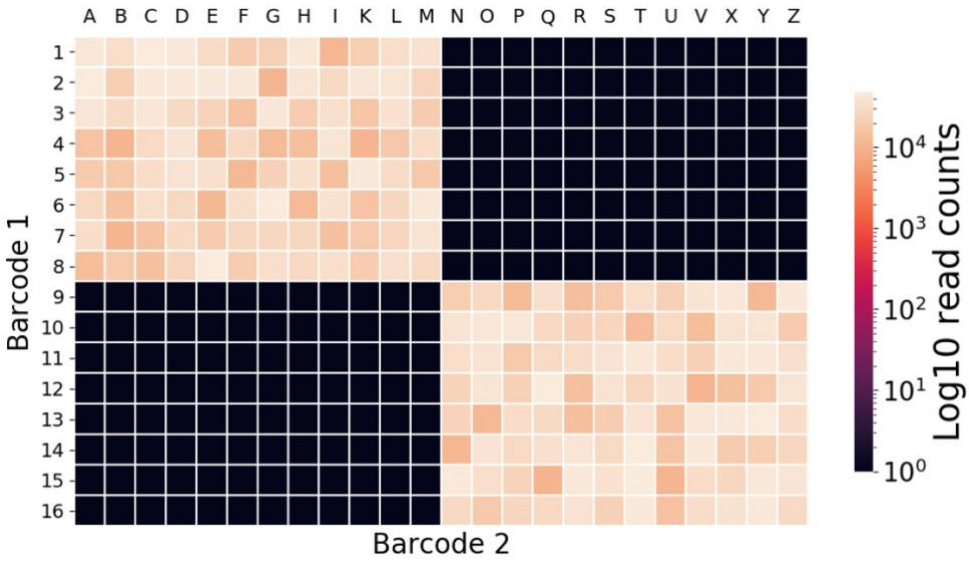


Figure 4.3 - Number of mapped reads per barcode combination in Miseq illumina platform, coloured on a $\log_{10}$ scale.

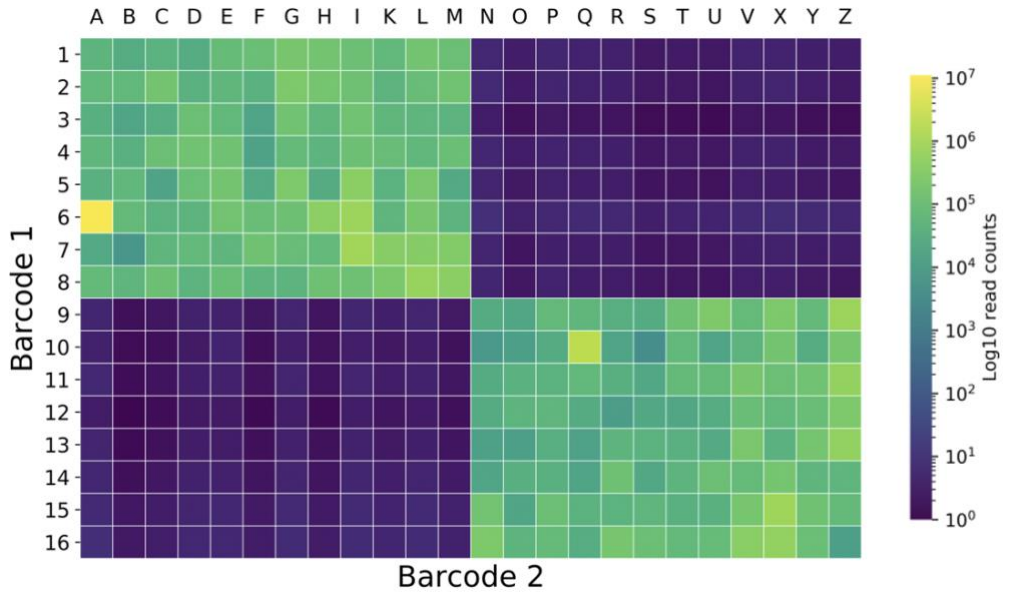


Figure 4.4 - Number of mapped reads per barcode combination in Hiseq illumina platform, coloured on a $\log_{10}$ scale. In this case, there are many reads in the unused combinations.

The distribution of total number of mapped reads (i.e., library sizes) for all combinations are shown in Figure 4.4 and Figure 4.5. The unused combinations have a median mapped-read library size that is 1.3% of the median size of the expected combinations. For Mapped-read we refer to the reads that mapped on bacteria genome. The total number of mapped reads assigned to unused combinations is 1.8% of that assigned to expected combinations.

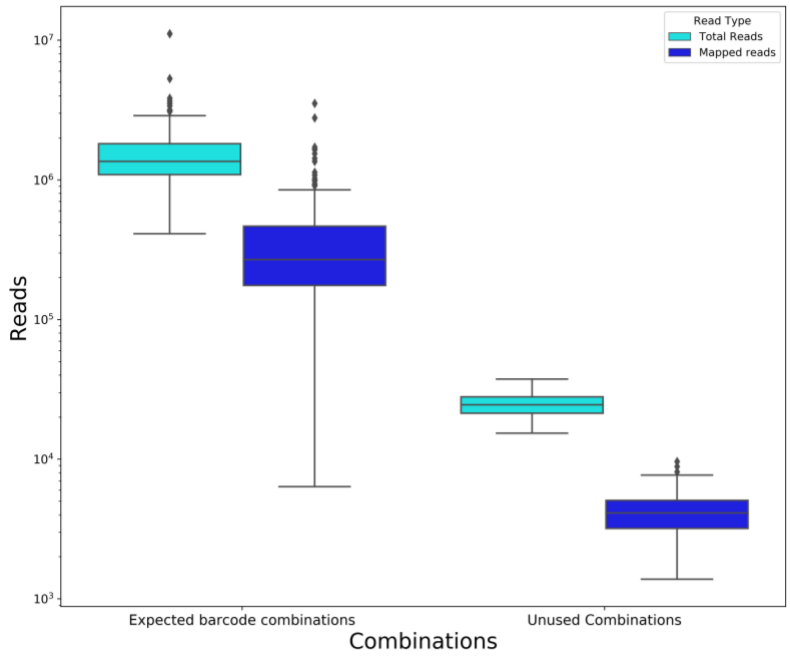


Figure 4.5 - Boxplots of the total number of reads for the Unused and expected barcode combinations. Dots represent barcode combinations that have totals more than 1.5 interquartile ranges from the edge of the box.

We focused on mapped reads as these are most relevant to downstream analyses. Nonetheless, we observe similar results for all reads, consistent with the ability of barcode swapping to affect all molecules on the flow cell. The median number of all reads assigned to an unused combination is 2.2% of that assigned to an expected combination, while the total number of reads assigned to unused combinations is 1.9% of the total number of reads assigned to expected combinations.

We emphasize that our results are highly robust to contamination from ambient RNA or human/bacterial sources. Regardless of the amount of contamination, the unused barcode combinations should not contain any reads, because these pairs of barcodes were never mixed in library preparation. Barcode swapping is the only possible mechanism for obtaining a substantial number of non-zero reads for these combinations. (We ignore the possibility of barcode sequencing errors causing misassignment of reads, which would be extremely unlikely for 8 bp barcodes that are well separated in base space.) This constitutes a difference between our experimental design and that of Sinha's [59] who estimated the swapping rate based on empty wells

that still contained barcodes. In their design, contamination would result in mapped reads in the empty wells and the appearance of an elevated rate of swapping.

4.3 Swapping fraction estimation on the HiSeq Sequencing platform

Denote each barcode combination as (i, j) where barcode $i \in 1, \dots, 16$ represents a row in Figure 4.6 with $(i = 1 \Rightarrow 1, i = 16 \Rightarrow 16)$ and barcode $j \in 1, \dots, 24$ represents a column $(j = 1 \Rightarrow A, j = 24 \Rightarrow Z)$. Let M_{lj} denote the number of seeded cDNA molecules that truly originate from this combination and let X_{lj} denote the number of mapped reads. M_{lj} therefore, represents the true source of reads, while X_{lj} represents the reported source after swapping. Impossible barcode combinations are those with $(1 \leq i \leq 8, N \leq j \leq Z)$ or $(9 \leq i \leq 16, A \leq j \leq M)$, and have $M_{lj} = 0$ by definition.

We assume that barcode swapping is rare, so it is unlikely that one molecule will undergo more than one round of swapping. This means that reads will only be transferred between combinations that already share a single barcode. This is illustrated in Figure 4.6. For that example, the combination Y6 (red position) would receive swapping contributions from the cells in the orange positions.

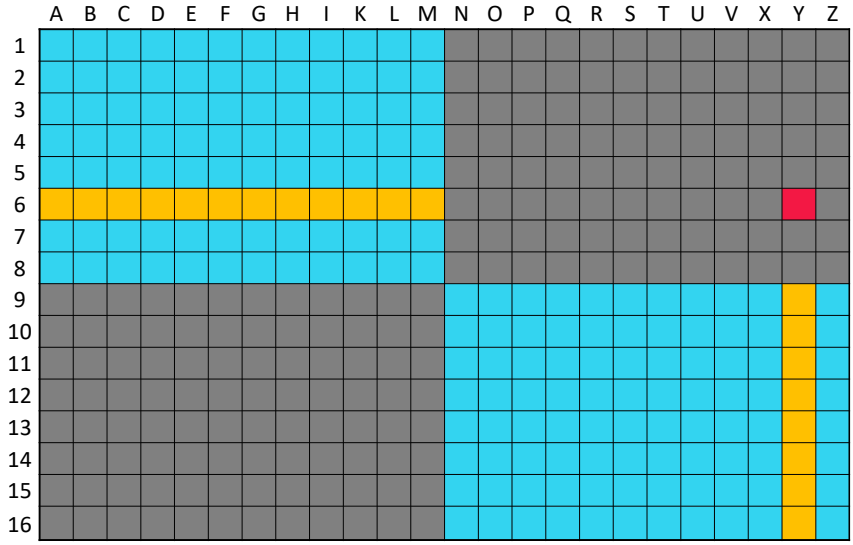


Figure 4.6 - A schematic of the expected barcode combinations that are potential donor libraries (orange) for swapping into a recipient library (red), an impossible combination (Y/6). Only a single barcode needs to be swapped for transcripts in the blue combinations to appear as reads in the red combination. Used barcode pairs are shown in light blue, and unused barcode pairs are shown in grey.

Let λ be the conversion rate of seeded cDNA molecules to mapped reads in the same cell library. This is probably less than 1 due to the presence of unmappable sequences (e.g., transcribed repeats). Moreover, a seeded PCR duplicate of a cDNA molecule (formed on the flow cell) would normally count as a new read for the gene in the original cell. However, if the duplicate molecule has swapped its barcode, the cell has effectively "lost" this additional read. This will further decrease the value of λ.

We further assume that the number of observed swapped reads is proportional to the number of molecules that are available for swapping. Define γ as the rate of swapping from any single donor library to any single recipient library, i.e., the proportion of molecules in the donor library that appear as mislabelled reads in the recipient. For each barcode combination, the number of reads can be modelled as

$$X_{ij} = \lambda M_{ij} + \gamma \left(\sum_{k \neq j} M_{ik} + \sum_{l \neq i} M_{lj} \right)$$

As barcode swapping is rare, γ should be very low such that $X_{lj} \approx \lambda M_{lj}$ for the expected barcode combinations where $M_{lj} > 0$. We further approximate X_{lj} for these expected combinations by replacing it with the observed X_{lj}. This means that, for each unused combination (i_*, j_*), we have

$$X_{i*j*} \approx \frac{\gamma}{\lambda}\left(\sum_{k \neq j*} M_{i*k} + \sum_{l \neq i*} M_{lj*}\right)$$

This represents a linear relationship between the library size for each impossible combination and the sum of the library sizes for all expected combinations with which it shares a single barcode. We estimate the parameters of this relationship by fitting a line to each X_{i_*,j_*} against the corresponding sum using ordinary least squares, as shown in Figure 4.7.

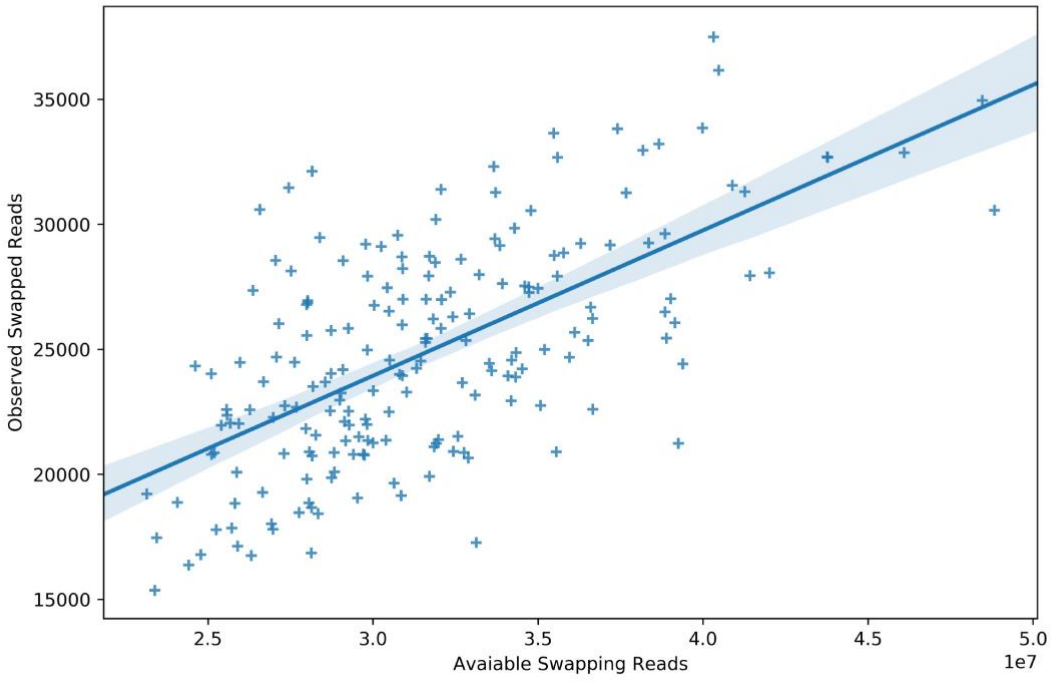


Figure 4.7 - Relationship between the available Swapping reads with the observed Swapped reads.

We estimate the total number of mislabelled reads across all combinations to be:

$$\gamma \sum_{i=1}^{16} \sum_{j=1}^{24} \left(\sum_{k \neq j}^{\infty} M_{ik} + \sum_{l \neq i}^{\infty} M_{lj} \right)$$

$$= 38\gamma \sum_{i=1}^{16} \sum_{j=1}^{24} M_{ij}$$

$$= 38\gamma \sum_{(i,j) \in \varepsilon}^{24} M_{ij}$$

$$\cong \frac{38\gamma}{\lambda} \sum_{(i,j) \in \varepsilon} X_{ij}$$

where ε is a set of all expected combinations. The multiplication by 38 is due to the fact that there are 38 available destinations for a single-barcode-swapped read from any single expected combination (Figure 4.8). In other words, we sum each M_{lj} 38 times in the first line of the above expression. This includes the unused barcode combinations as well as the real expected combinations.

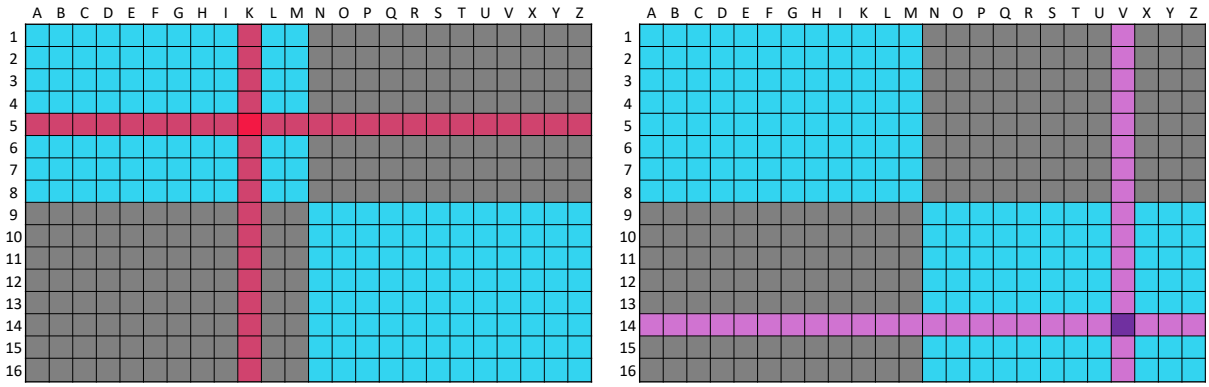


Figure 4.8 - Schematic illustrating the contribution of a donor library (dark red and dark violet) to each of 38 recipient libraries that share a single barcode (blue). Cell-loaded combinations are shown in blue, and unloaded combinations are shown in grey.

To obtain the swapping fraction in this experiment, we divide by the total number of mapped reads:

$$\frac{38\gamma}{\lambda}\left(\frac{\sum_{(i,j)\in\,\varepsilon}\,X_{ij}}{\sum_{i=1}^{16}\,\sum_{j=1}^{24}\,X_{ij}}\right)$$

This yields an estimated swapping fraction of 0.1272%. Notably, this is higher than the median-to-median fraction of 1.5%. This is because the median-to-median fraction only considered swapped reads in the unused barcode combinations, whereas this slope-estimated value considers reads that swap across the entire set of barcode combinations.

4.4 iSwap application

In this section we show the results obtained with the application of iSwap to our data. We applied iSwap on 100 different datasets always sequences with the same sample available. The the picture below shows the different color-scale in $\log_{10}$ before and after respectively the application of the pipeline.

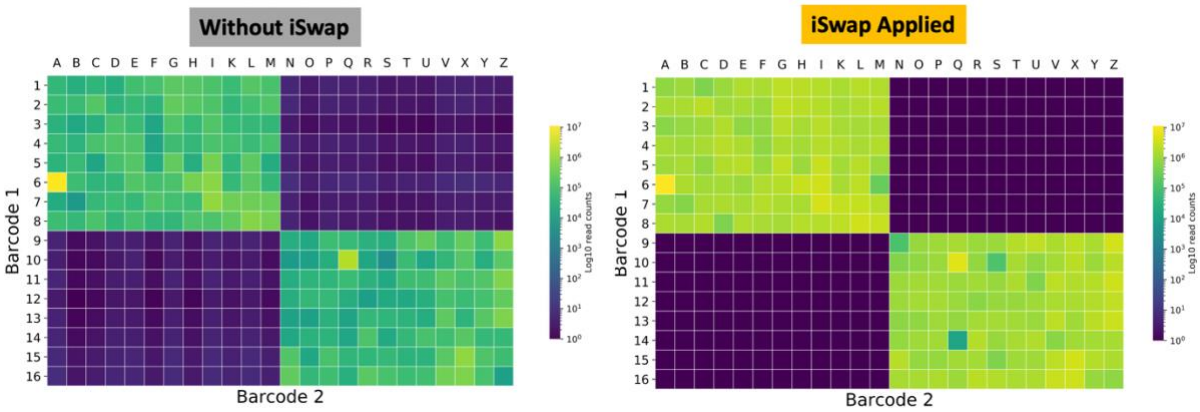


Figure 4.9 - Schematic illustrating of iSwap application before on the left and after on the right.

In Figure 4.9, before the application of the iSwap pipeline the colors are of the order of 10^5, while after the application of iSwap the colors change around 10^6, a significant increase in the reads for each sample, recovered from the unused combinations, but also from the expected combinations used.

Thus, colors are different due to the fact that many sequences are reassigned to the correct sample after algorithm's application. Not only does the unused combinations lose sequences, but the expected combination can also contain false positive sequences. The unused barcode combination still remains with something not being completely black.

In fact, if we go to see the number of reads behind combinations, we will see that a little amount of reads still remain, as showed in Figure 4.10. But the number of reads removed as false positive is huge.

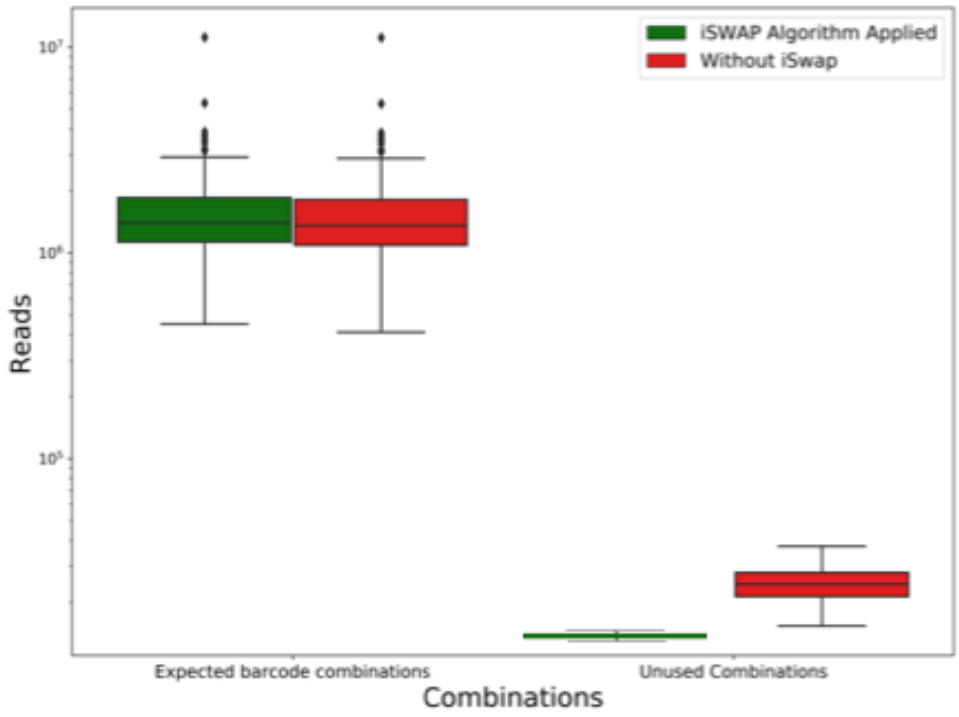


Figure 4.10 - Number of reads in expected and unused combination before and after the application of the iSwap algorithm.

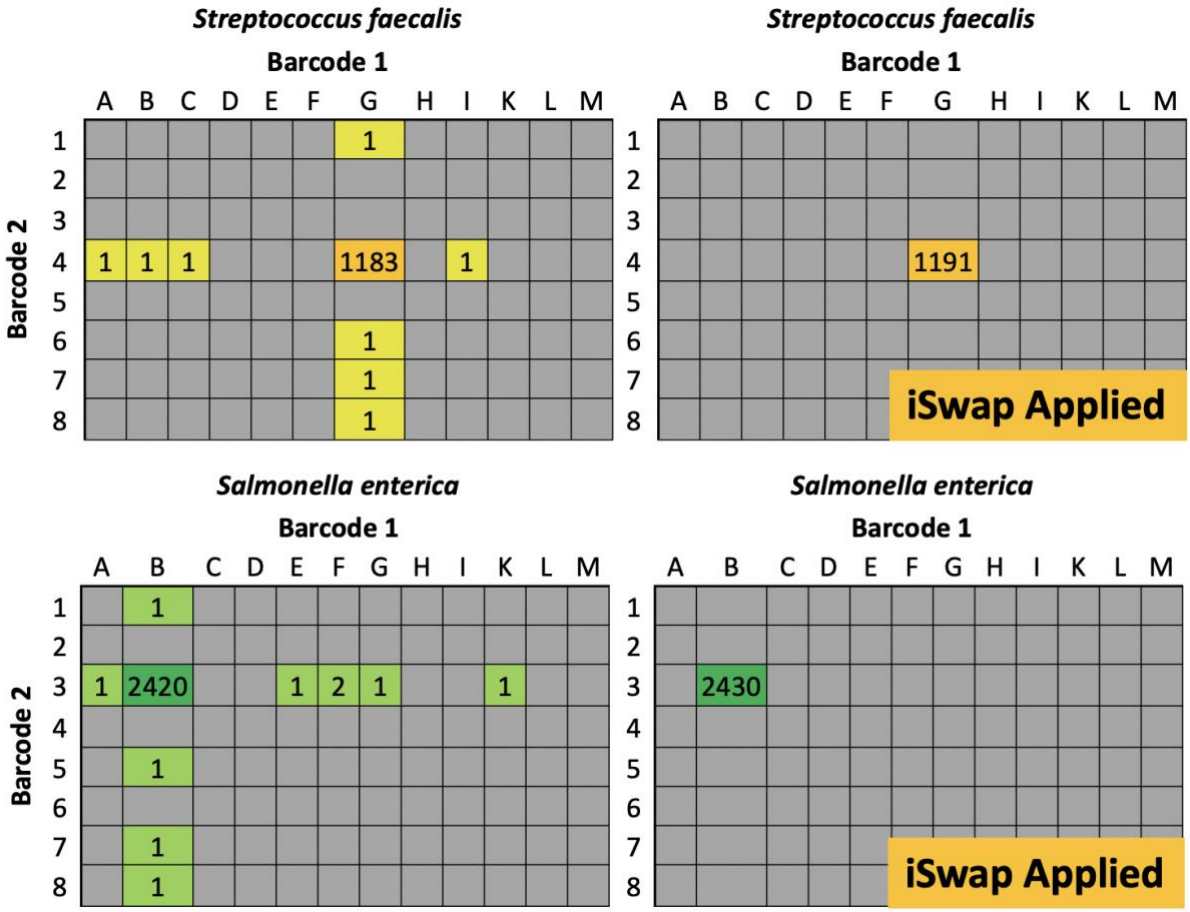


Figure 4.11 - Sequencing spread around the dataset, before and after application of the iSwap pipeline.

In Figure 4.11, we have a specific case of how the iSwap pipeline detects and removes and remove contaminations. In this diagram, the case before applying the algorithm is shown on the left and the case after applying iSwap on the right. In this example the bacterial species *Streptococcus faecalis* and *Salmonella enteric*a are taken into account, respectively for the samples *G4* and *B3*, these two bacterial species belonged exclusively to these two samples and it is possible to note how other sequences were found in other samples, after sequencing. All samples sharing one of the two donor sample barcodes. While, after the application of the iSwap, no streptococcal sequences were found in others samples from the donor *G4*, the same thing also applies to the case of *Salmonella enterica* linked to sample *B3*.

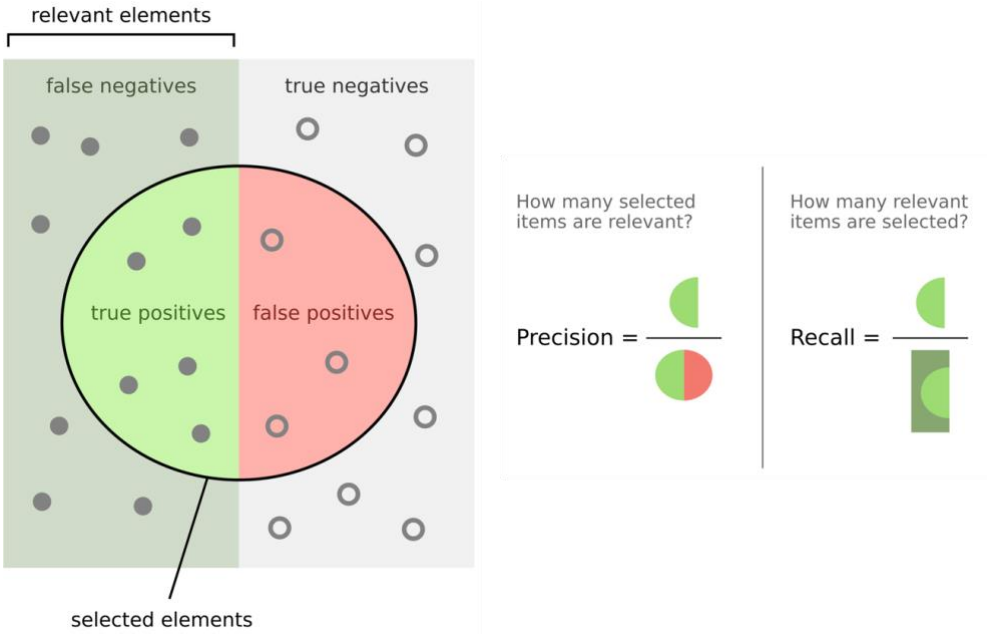
4.5 Performances Precision and Recall

Precision and **recall** are two common statistical classifications, used in different areas. Precision can be seen as a measure of accuracy or fidelity, while recall is a measure of completeness. In a statistical classification process, the precision for a class is the number of **true positive [TP]** (the number of objects labelled correctly as belonging to the class) divided by the total number of tags labelled as belonging to the class (the sum of true positive and **false positives [FP],** which are erroneously labelled objects as belonging to the class).

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

The recall is defined as the number of true positive divided by the total number of elements currently belonging to the class (the sum of true positive and **false negative [FN]**, which are objects that have not been labelled as belonging to the class but should be).

$$Recall = \frac{TP}{TP + FN} \tag{6}$$



In our study, TP are represented by sequences that have been properly assigned to a right sample. FP are the sequences that have been incorrectly assigned to a sample, while the FN are the sequences that have not been assigned to any sample and are "Unclassified". In a taxonomic assignment process, a precision of 1.0 for the UCLUST assignment method means that every sequence in the dataset that has been assigned to the right sample (but does not say anything about the number of sequences that have not been assigned to a taxonomy) while a recall value of 1.0 for the same algorithm means that each sequence of the dataset has been assigned to a sample (regardless of whether the assignment is correct or not). We can say that precision provides an index of how a sequence assignment method is accurate when assigning a sequence to a sample, while recall provides an index of the ability not to lose information.

From the study of precision and recall values, it was possible to calculate the F-measure representing the harmonic mean that gives the same "weight" to precision and recall.

$$F - measure = 2 * \frac{precision * recall}{precision + recall} \quad (7)$$

The F-measure was the metric that allowed us to evaluate and compare the different results obtained, including what combination of parameters was the one that showed a number between FP and FN lower than the others.

Our pipeline was compared with two other tools available online for the same purpose but in other field of application. [62], [90].

In the following table we show the results of precision and recall obtained on 100 different datasets.

Dataset	Algorithm	Precision	Recall	f-measure
1	iSwap	0.962108	0.919492	0.940318
1	Griffith	0.722793	0.896663	0.800394
1	Larsson	0.527619	0.339281	0.412991
2	iSwap	0.902574	0.89981	0.90119
2	Griffith	0.82944	0.691662	0.754312
2	Larsson	0.473043	0.407472	0.437816
…	…	…	…	…
99	iSwap	0.835217	0.845543	0.840348
99	Griffith	0.787892	0.625193	0.697176
99	Larsson	0.610769	0.432008	0.506066
100	iSwap	0.84959	0.97876	0.909612
100	Griffith	0.718509	0.730908	0.724656
100	Larsson	0.377993	0.558077	0.450713

Table 4.12 – Precision, Recall and F-measure tested on 100 different datasets using 3 different algorithms.

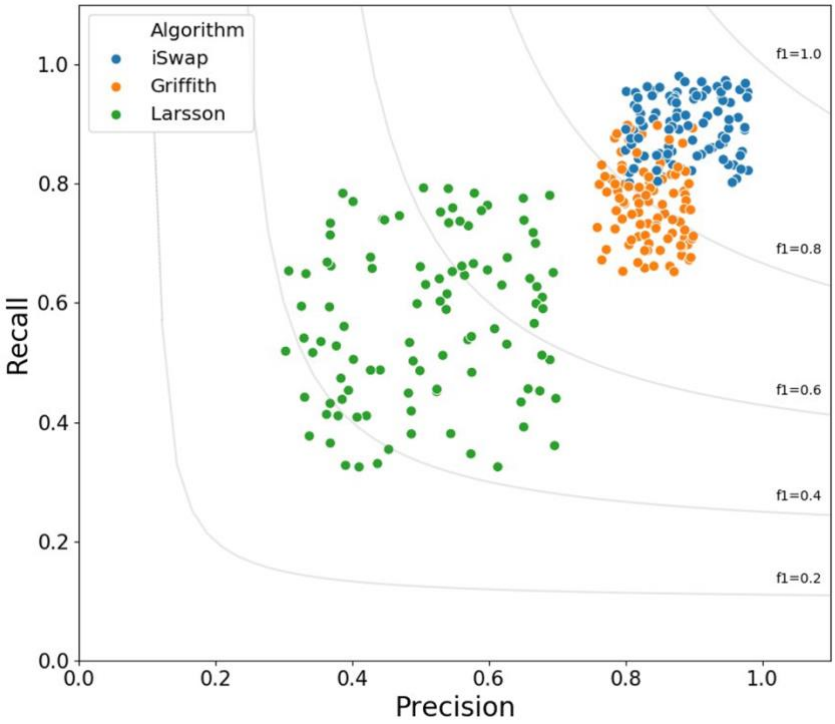


Figure 4.13 - Precision and Recall plot of 3 different tools.

These results are displayed in a plot in figure 4.13 where we noticed that iSwap performs better than the Griffith and Larsson methods for the detection and remotion of the index switching phenomenon. iSwap performs better in every single dataset than the other tools.

4.6 Experimental solutions for index switching

One proposed solution for the swapping problem are unique-at-both-ends indices. In these experiments, a cDNA molecule in a given cell's library is indexed with a unique barcode at each end. These barcodes are never reused for any other cell library in the same multiplexed set. A single barcode swap therefore moves a sequencing read into an unused barcode combination, not into another cell library. This is an effective solution for the multiplexing of a relatively low number of libraries for sequencing.

However, for single-cell RNA-seq, it may be desirable to sequence many hundreds of multiplexed samples (i.e., cells) together, particularly as sequencing facilities transition towards the use of higher-throughput machines. Consider the situation where we have μ unique barcode sequences. If barcodes are reused between cells in unique combinations, the maximum number of samples that can be multiplexed together is:

$$\left(\frac{\mu}{2}\right)^2$$

assuming that $\frac{\mu}{2}$ barcodes are exclusively used for 5' or 3' indexing. In contrast, the maximum number of combinations for unique-at-both-ends indexing is:

$$\frac{\mu}{2}$$

Clearly, unique-at-both-ends indexing severely restricts the throughput of multiplexing strategies.

In practice, barcodes are often used in a 96-well plate (of dimension 12 x 8). For every additional 20 barcodes that are available, assume that 8 are used to index rows (say, 5' indexing), and 12 are used to index columns (3' indexing). Here, the number of possible barcode combinations is:

$$\frac{\mathbf{8}\mu}{20} * \frac{\mathbf{12}\mu}{20} = \frac{\mathbf{6}\mu^{\mathbf{2}}}{25}$$

Again, this approach allows the multiplexing of very many more samples than unique-at-both-ends approaches. The difference in scaling of these values is illustrated in Figure 4.14

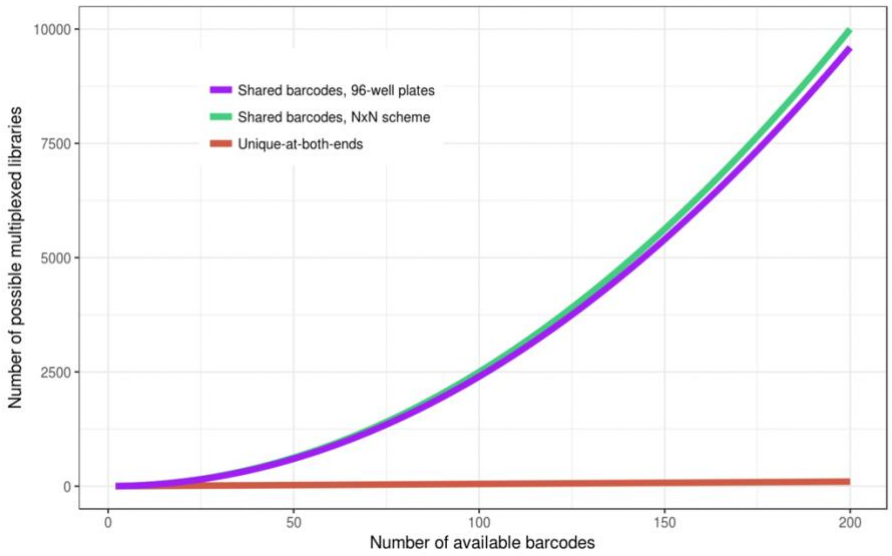


Figure 4.14 - Maximum number of libraries that can be multiplexed under different labelling schemes, as a function of the number of available barcodes.

Use of unique-at-both-ends barcoding is particularly problematic for methods such as sci-Seq [91], where the reuse of barcodes between cells generates the combinatorial complexity that allows massively high-throughput generation of cell libraries.

While we would encourage the use of experimental designs with unique-at-both-ends indexing where possible, it is clearly not a suitable approach for all experiments. This motivates our development of computational methods to address index switching.

Conclusion

Since the advent of Illumina's pattern flow cell technologies, the problem of index switching has begun to take hold. This phenomenon not only led to sequencing problems by generating new combinations of barcodes never used in the experimental design, but it also led to results problems by generating a good slice of false positives that in important studies such as single cell analysis, integration site, microbiome leads to misinterpreting the results. This is because a sample received a certain amount of sequences that does not actually belong to it. Index switching occurs when there is sequencing and different samples have been multiplexed at the same time. The multiplexing technique not only saves money for those who perform sequencing, but also a lot of time by being able to sequence many samples at the same time. It was pointed out that the use of single barcodes allows to solve the index switching problem, however this technique does not allow sequencing many samples at the same time due to the lack of the number of available barcodes. For this reason, we have decided to create a tool that is able to remove index switching. This study started finding a huge amount a contaminations in our datasets. This was demonstrated through the birthday paradox [91], calculating that the presence of integration sites found at the same point in the genome between different patients was highly unlikely. From that moment we started studying the problem using control sample and it turned out that it was index switching. The first version of iSwap was solely concerned with eliminating the problem once the data was analyzed, while in the version I presented in this thesis, however, it is an entire pipeline that allows you to obtain clean results without changing the input data. iSwap consists of a series of steps optimized to identify and eliminate index switching. The crucial processes of the analysis are the clustering phase which allows to obtain groups of sequences similar to each other and the optimization of memory, otherwise there would be a large consumption of the latter. Furthermore, thanks to the use of methods such as MapReduce, the analysis process is very fast, obtaining results in a few hours. iSwap identifies the donor of contamination and assigns lost sequences to it. If the assignment process is difficult, then iSwap proceeds with the elimination of the index

switching sequences from the donor and the recipients. iSwap process produces a low amount of data loss. Infact, the precision and recall results emerged, evaluated on 100 different datasets; on average, the precision is 0.81, while for recall 0.83 with a standard deviation of 0.18, excellent results when compared with the other methods of correction of index switching present today. iSwap appears to be a not only a CLI tool available for the correction of index switching, but which also performs better than the methods proposed by other authors.

In current innovative patterned flow cell technology offers an exceptional level of throughput for diverse sequencing applications. Patterned flow cells use distinct nanowells for cluster generation to make more efficient use of the flow cell surface area. This advanced flow cell design contributes to increased data output, reduced costs, and faster run times. Patterned flow cells are produced using semiconductor manufacturing technology. Starting with a glass substrate, patterned nanowells are etched into the surface for optimal cluster spacing. Each nanowell contains DNA probes used to capture prepared DNA strands for amplification during cluster generation. The regions between the nanowells are devoid of DNA probes. The process ensures that DNA clusters only form within the nanowells, providing even, consistent spacing between adjacent clusters and allowing accurate resolution of clusters during imaging. Maximal use of the flow cell surface leads to overall higher clustering.

In these current sequencing technologies iSwap finds an excellent response being able to solve a current problem that affects the data produced. Probably with the advent of new pattern flow cell technologies this problem could be overcome and the use of iSwap no longer necessary. However, the use of this new pattern flow cell technology significantly increases the amount of data produced and therefore the more data is produced, the greater the likelihood that index switching events will occur. We consider it difficult for this technology to be abandoned or outdated soon as it provides significant advantages from the point of view of costs and data production times. For this reason, iSwap is a very useful tool to solve the problem of index switching as long as these high-throughput sequencing technologies are not overcome.

References

[1] P. Nannipieri, J. Ascher, M. T. Ceccherini, L. Landi, G. Pietramellara, and G. Renella, "Microbial diversity and soil functions," *European Journal of Soil Science*, vol. 54, no. 4. pp. 655–670, 2003, doi: 10.1046/j.1351-0754.2003.0556.x.

[2] G. J. Chan, A. C. Lee, A. H. Baqui, J. Tan, and R. E. Black, "Risk of Early-Onset Neonatal Infection with Maternal Infection or Colonization: A Global Systematic Review and Meta-Analysis," *PLoS Med.*, vol. 10, no. 8, 2013, doi: 10.1371/journal.pmed.1001502.

[3] G. E. Fox, L. J. Magrum, W. E. Balch, R. S. Wolfe, and C. R. Woese, "Classification of methanogenic bacteria by 16S ribosomal RNA characterization," *Proc. Natl. Acad. Sci.*, vol. 74, no. 10, pp. 4537–4541, 1977, doi: 10.1073/pnas.74.10.4537.

[4] F. Sanger, S. Nicklen, and a R. Coulson, "DNA sequencing with chain-terminating inhibitors.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 74, no. 12, pp. 5463–7, 1977, doi: 10.1073/pnas.74.12.5463.

[5] R. M. Stubbendieck, C. Vargas-Bautista, and P. D. Straight, "Bacterial communities: Interactions to scale," *Frontiers in Microbiology*, vol. 7, no. AUG. 2016, doi: 10.3389/fmicb.2016.01234.

[6] M. Fakruddin and K. S. Bin Mannan, "Methods for Analyzing Diversity of Microbial Communities in Natural Environments," *Ceylon J. Sci. (Biological Sci.*, vol. 42, no. 1, pp. 19–33, 2013, doi: 10.4038/cjsbs.v42i1.5896.

[7] M. R. Cancilla, I. B. Powell, A. J. Hillier, and B. E. Davidson, "Rapid genomic fingerprinting of Lactococcus lactis strains by arbitrarily primed polymerase chain reaction with 32P and fluorescent labels," *Appl. Environ. Microbiol.*, vol. 58, no. 5, pp. 1772–1775, 1992.

[8] G. Muyzer and E. De Waal, "Profiling of complex microbial populations by denaturating gradient gel electrophoresis analysis of polymerase chain reaction-amplified genes coding for 16S rRNA," *Appl. Environ. Microbiol.*, vol. 59, 1993, [Online]. Available: http://aem.asm.org/cgi/content/abstract/59/3/695.

[9] D. J. Lane, B. Pace, G. J. Olsen, D. A. Stahl, M. L. Sogin, and N. R. Pace, "Rapid determination of 16S ribosomal RNA sequences for phylogenetic analyses.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 82, no. 20, pp. 6955–9, 1985, doi: 10.1073/pnas.82.20.6955.

[10] J. Handelsman, M. R. Rondon, S. F. Brady, J. Clardy, and R. M. Goodman, "Molecular

biological access to the chemistry of unknown soil microbes: a new frontier for natural products.," *Chem. Biol.*, vol. 5, no. 10, pp. R245–R249, 1998, doi: 10.1016/S1074-5521(98)90108-9.

[11] N. Ottman, H. Smidt, W. M. de Vos, and C. Belzer, "The function of our microbiota: who is out there and what do they do?," *Front Cell Infect Microbiol*, vol. 2, p. 104, 2012, doi: 10.3389/fcimb.2012.00104.

[12] P. Yarza *et al.*, "Uniting the classification of cultured and uncultured bacteria and archaea using 16S rRNA gene sequences," *Nat. Rev. Microbiol.*, vol. 12, no. 9, pp. 635–645, 2014, doi: 10.1038/nrmicro3330.

[13] J. I. Gordon, "Honor Thy Gut Symbionts Redux," *Science (80-.).*, vol. 336, no. 6086, pp. 1251–1253, 2012, doi: 10.1126/science.1224686.

[14] P. Hugenholtz, B. M. Goebel, and N. R. Pace, "Erratum: Impact of culture-independent studies on the emerging phylogenetic view of bacterial diversity (Journal of Bacteriology (1998) 180:18 (4765-4774))," *Journal of Bacteriology*, vol. 180, no. 24. p. 6793, 1998, doi: 0021-9193/98/$04.00+0.

[15] L. A. Onaga, "Ray Wu as Fifth Business: Deconstructing collective memory in the history of DNA sequencing," *Stud. Hist. Philos. Sci. Part C Stud. Hist. Philos. Biol. Biomed. Sci.*, vol. 46, no. 1, pp. 1–14, 2014, doi: 10.1016/j.shpsc.2013.12.006.

[16] R. Padmanabhan, E. Jay, and R. Wu, "Chemical synthesis of a primer and its use in the sequence analysis of the lysozyme gene of bacteriophage T4," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 71, no. 6, pp. 2510–2514, 1974, doi: 10.1073/pnas.71.6.2510.

[17] R. Wu and E. Taylor, "Nucleotide sequence analysis of DNA," *J. Mol. Biol.*, vol. 57, no. 3, pp. 491–511, 1971, doi: 10.1016/0022-2836(71)90105-7.

[18] M. Margulies *et al.*, "Genome sequencing in microfabricated high-density picolitre reactors.," *Nature*, vol. 437, no. 7057, pp. 376–80, 2005, doi: 10.1038/nature03959.

[19] J. Shendure *et al.*, "Accurate multiplex polony sequencing of an evolved bacterial genome.," *Science*, vol. 309, no. 5741, pp. 1728–32, 2005, doi: 10.1126/science.1117389.

[20] E. R. Mardis, "Next-Generation Sequencing Platforms," *Annu. Rev. Anal. Chem*, vol. 6, pp. 287–303, 2013, doi: 10.1146/annurev-anchem-062012-092628.

[21] J. Shendure and H. Ji, "Next-generation DNA sequencing," *Nat Biotechnol*, vol. 26, pp. 1135–1145, 2008, doi: 10.1038/nbt1486.

[22] S. Shokralla, J. L. Spall, J. F. Gibson, and M. Hajibabaei, "Next-generation sequencing technologies for environmental DNA research," *Molecular Ecology*, vol. 21, no. 8. pp.

1794–1805, 2012, doi: 10.1111/j.1365-294X.2012.05538.x.

[23] M. Delseny, B. Han, and Y. I. Hsing, "High throughput DNA sequencing: The new sequencing revolution," *Plant Science*, vol. 179, no. 5. pp. 407–422, 2010, doi: 10.1016/j.plantsci.2010.07.019.

[24] M. L. Metzker, "Sequencing technologies - the next generation.," *Nat. Rev. Genet.*, vol. 11, no. 1, pp. 31–46, 2010, doi: 10.1038/nrg2626.

[25] P. Hugenholtz and N. R. Pace, "Identifying microbial diversity in the natural environment: a molecular phylogenetic approach.," *Trends Biotechnol.*, vol. 14, no. 6, pp. 190–7, 1996, doi: 10.1016/0167-7799(96)10025-1.

[26] N. R. Pace, D. A. Stahl, D. J. Lane, and G. J. Olsen, "The Analysis of Natural Microbial Populations by Ribosomal RNA Sequences," in *Advances in Microbial Ecology*, 1986, pp. 1–55.

[27] N. R. Pace, "A molecular view of microbial diversity and the biosphere.," *Science*, vol. 276, no. 5313, pp. 734–740, 1997, doi: 10.1126/science.276.5313.734.

[28] B. D. Muegge *et al.*, "Diet drives convergence in gut microbiome functions across mammalian phylogeny and within humans.," *Science*, vol. 332, no. 6032, pp. 970–4, 2011, doi: 10.1126/science.1198719.

[29] P. J. Turnbaugh *et al.*, "A core gut microbiome in obese and lean twins," *Nature*, vol. 457, no. 7228, pp. 480–484, 2009, doi: 10.1038/nature07540.

[30] S. Hong, J. Bunge, C. Leslin, S. Jeon, and S. S. Epstein, "Polymerase chain reaction primers miss half of rRNA microbial diversity.," *ISME J.*, vol. 3, no. 12, pp. 1365–1373, 2009, doi: 10.1038/ismej.2009.89.

[31] R. Logares *et al.*, "Metagenomic 16S rDNA Illumina tags are a powerful alternative to amplicon sequencing to explore diversity and structure of microbial communities," *Environ. Microbiol.*, vol. 16, no. 9, pp. 2659–2671, 2014, doi: 10.1111/1462-2920.12250.

[32] T. J. Sharpton *et al.*, "PhyLOTU: A high-throughput procedure quantifies microbial community diversity and resolves novel taxa from metagenomic data," *PLoS Comput. Biol.*, vol. 7, no. 1, 2011, doi: 10.1371/journal.pcbi.1001061.

[33] Z. Liu, T. Z. Desantis, G. L. Andersen, and R. Knight, "Accurate taxonomy assignments from 16S rRNA sequences produced by highly parallel pyrosequencers," *Nucleic Acids Res.*, vol. 36, no. 18, 2008, doi: 10.1093/nar/gkn491.

[34] P. Unneberg and J. M. Claverie, "Tentative mapping of transcription-induced interchromosomal interaction using chimeric EST and mRNA data," *PLoS One*, vol. 2,

no. 2, 2007, doi: 10.1371/journal.pone.0000254.

[35] K. M. Wylie *et al.*, "Novel bacterial Taxa in the human microbiome," *PLoS One*, vol. 7, no. 6, 2012, doi: 10.1371/journal.pone.0035294.

[36] M. Langille *et al.*, "Predictive functional profiling of microbial communities using 16S rRNA marker gene sequences.," *Nat. Biotechnol.*, vol. 31, no. 9, pp. 814–21, 2013, doi: 10.1038/nbt.2676.

[37] T. J. Sharpton, "An introduction to the analysis of shotgun metagenomic data.," *Front. Plant Sci.*, vol. 5, no. June, p. 209, 2014, doi: 10.3389/fpls.2014.00209.

[38] K. Mavromatis *et al.*, "Use of simulated data sets to evaluate the fidelity of metagenomic processing methods," *Nat. Methods*, vol. 4, no. 6, pp. 495–500, 2007, doi: 10.1038/nmeth1043.

[39] D. R. Mende *et al.*, "Assessment of metagenomic assembly using simulated next generation sequencing data," *PLoS One*, vol. 7, no. 2, 2012, doi: 10.1371/journal.pone.0031386.

[40] Y. V. Chew and A. J. Holmes, "Suppression subtractive hybridisation allows selective sampling of metagenomic subsets of interest," *J. Microbiol. Methods*, vol. 78, no. 2, pp. 136–143, 2009, doi: 10.1016/j.mimet.2009.05.003.

[41] N. Delmotte *et al.*, "Community proteogenomics reveals insights into the physiology of phyllosphere bacteria.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 106, no. 38, pp. 16428–16433, 2009, doi: 10.1073/pnas.0905240106.

[42] T. Woyke *et al.*, "Symbiosis insights through metagenomic analysis of a microbial consortium," *Nature*, vol. 443, no. 7114, pp. 950–955, 2006, doi: 10.1038/nature05192.

[43] P. H. Degnan and H. Ochman, "Illumina-based analysis of microbial community diversity.," *ISME J.*, vol. 6, no. 1, pp. 183–94, 2012, doi: 10.1038/ismej.2011.74.

[44] V. Kunin, A. Copeland, A. Lapidus, K. Mavromatis, and P. Hugenholtz, "A bioinformatician's guide to metagenomics.," *Microbiol. Mol. Biol. Rev.*, vol. 72, no. 4, pp. 557–78, Table of Contents, 2008, doi: 10.1128/MMBR.00009-08.

[45] R. Schmieder and R. Edwards, "Fast identification and removal of sequence contamination from genomic and metagenomic datasets," *PLoS One*, vol. 6, no. 3, 2011, doi: 10.1371/journal.pone.0017288.

[46] P. D. Schloss and J. Handelsman, "Status of the microbial census," *Microbiol Mol Biol Rev*, vol. 68, no. 4, pp. 686–691, 2004, doi: 68/4/686 [pii]\r10.1128/MMBR.68.4.686-691.2004.

[47] E. STACKEBRANDT and B. M. GOEBEL, "Taxonomic Note: A Place for DNA-DNA Reassociation and 16S rRNA Sequence Analysis in the Present Species Definition in Bacteriology," *Int J Syst Bacteriol*, vol. 44, no. 4, pp. 846–849, 1994, doi: 10.1099/00207713-44-4-846.

[48] N. Youssef, C. S. Sheik, L. R. Krumholz, F. Z. Najar, B. A. Roe, and M. S. Elshahed, "Comparison of species richness estimates obtained using nearly complete fragments and simulated pyrosequencing-generated fragments in 16S rRNA gene-based environmental surveys," *Appl. Environ. Microbiol.*, vol. 75, no. 16, pp. 5227–5236, 2009, doi: 10.1128/AEM.00592-09.

[49] Z. Liu, C. Lozupone, M. Hamady, F. D. Bushman, and R. Knight, "Short pyrosequencing reads suffice for accurate microbial community analysis," *Nucleic Acids Res.*, vol. 35, no. 18, 2007, doi: 10.1093/nar/gkm541.

[50] Q. Wang, G. M. Garrity, J. M. Tiedje, and J. R. Cole, "Na??ve Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy," *Appl. Environ. Microbiol.*, vol. 73, no. 16, pp. 5261–5267, 2007, doi: 10.1128/AEM.00062-07.

[51] M. J. Claesson *et al.*, "Comparative analysis of pyrosequencing and a phylogenetic microarray for exploring microbial community structures in the human distal intestine," *PLoS One*, vol. 4, no. 8, 2009, doi: 10.1371/journal.pone.0006669.

[52] M. L. Sogin *et al.*, "Microbial diversity in the deep sea and the underexplored 'rare biosphere'.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 103, no. 32, pp. 12115–20, 2006, doi: 10.1073/pnas.0605127103.

[53] A. Engelbrektson *et al.*, "Experimental factors affecting PCR-based estimates of microbial species richness and evenness.," *ISME J.*, vol. 4, no. 5, pp. 642–647, 2010, doi: 10.1038/ismej.2009.153.

[54] L. Dethlefsen, S. Huse, M. L. Sogin, and D. A. Relman, "The pervasive effects of an antibiotic on the human gut microbiota, as revealed by deep 16s rRNA sequencing," *PLoS Biol.*, vol. 6, no. 11, pp. 2383–2400, 2008, doi: 10.1371/journal.pbio.0060280.

[55] S. M. Huse, L. Dethlefsen, J. A. Huber, D. M. Welch, D. A. Relman, and M. L. Sogin, "Exploring microbial diversity and taxonomy using SSU rRNA hypervariable tag sequencing," *PLoS Genet.*, vol. 4, no. 11, 2008, doi: 10.1371/journal.pgen.1000255.

[56] M. Kim, M. Morrison, and Z. Yu, "Evaluation of different partial 16S rRNA gene sequence regions for phylogenetic analysis of microbiomes," *J. Microbiol. Methods*, vol. 84, no. 1, pp. 81–87, 2011, doi: 10.1016/j.mimet.2010.10.020.

[57] D. Content and A. Data, "Index-Hopping-White-Paper-770-2017-004.Pdf," pp. 4–7, 2017.

[58] M. Costello *et al.*, "Characterization and remediation of sample index swaps by non-redundant dual indexing on massively parallel sequencing platforms," *BMC Genomics*, vol. 19, no. 1, pp. 1–10, 2018, doi: 10.1186/s12864-018-4703-0.

[59] R. Sinha *et al.*, "Index Switching Causes 'Spreading-Of-Signal' Among Multiplexed Samples In Illumina HiSeq 4000 DNA Sequencing," *bioRxiv*, p. 125724, 2017, doi: 10.1101/125724.

[60] J. Hadfield, "index-mis-assignment-between-samples." http://enseqlopedia.com/2016/12/index-mis-assignment-between-samples-on-hiseq-4000-and-x-ten/.

[61] A. Pérez *et al.*, "No 主観的健康感を中心とした在宅高齢者における 健康関連指標に関する共分散構造分析Title," *BMC Public Health*, vol. 5, no. 1, pp. 1–8, 2017.

[62] A. J. M. Larsson, G. Stanley, R. Sinha, I. L. Weissman, and R. Sandberg, "Computational correction of index switching in multiplexed sequencing libraries," *Nat. Methods*, vol. 15, no. 5, pp. 305–307, 2018, doi: 10.1038/nmeth.4666.

[63] T. van der Valk, F. Vezzi, M. Ormestad, L. Dalén, and K. Guschanski, "Estimating the rate of index hopping on the Illumina HiSeq X platform," *bioRxiv*, p. 179028, 2017, doi: 10.1101/179028.

[64] J. Leipzig, "A review of bioinformatic pipeline frameworks," *Brief. Bioinform.*, no. January, p. bbw020, 2016, doi: 10.1093/bib/bbw020.

[65] P. J. Turnbaugh *et al.*, "The human microbiome project.," *Nature*, vol. 449, no. 7164, pp. 804–10, 2007, doi: 10.1038/nature06244.

[66] Andrews S., "FastQC: a quality control tool for high throughput sequence data," 2010. http://www.bioinformatics.babraham.ac.uk/projects/fastqc.

[67] A. M. Bolger, M. Lohse, and B. Usadel, "Trimmomatic: A flexible trimmer for Illumina sequence data," *Bioinformatics*, vol. 30, no. 15, pp. 2114–2120, 2014, doi: 10.1093/bioinformatics/btu170.

[68] R. Edgar, "UCHIME2: improved chimera prediction for amplicon sequencing," *bioRxiv*, p. 074252, 2016, doi: 10.1101/074252.

[69] S. L. Westcott and P. D. Schloss, "De novo clustering methods outperform reference-based methods for assigning 16S rRNA gene sequences to operational taxonomic units.," *PeerJ*, vol. 3, p. e1487, 2015, doi: 10.7717/peerj.1487.

[70] H. Lab., "FASTX-Toolkit - FASTQ/A short-reads pre-processing tools," 2009. .

[71] E. Aronesty, "ea-utils - FASTQ processing utilities," 2010, [Online]. Available: https://expressionanalysis.github.io/ea-utils/.

[72] R. P. Smyth *et al.*, "Reducing chimera formation during PCR amplification to ensure accurate genotyping," *Gene*, vol. 469, no. 1–2, pp. 45–51, 2010, doi: 10.1016/j.gene.2010.08.009.

[73] P. D. Schloss, D. Gevers, and S. L. Westcott, "Reducing the effects of PCR amplification and sequencing Artifacts on 16s rRNA-based studies," *PLoS One*, vol. 6, no. 12, 2011, doi: 10.1371/journal.pone.0027310.

[74] T. Z. DeSantis *et al.*, "Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB," *Appl. Environ. Microbiol.*, vol. 72, no. 7, pp. 5069–5072, 2006, doi: 10.1128/AEM.03006-05.

[75] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res.*, vol. 22, no. 22, pp. 4673–4680, 1994, doi: 10.1093/nar/22.22.4673.

[76] P. D. Schloss, "The effects of alignment quality, distance calculation method, sequence filtering, and region on the analysis of 16S rRNA gene-based studies," *PLoS Comput. Biol.*, vol. 6, no. 7, p. 19, 2010, doi: 10.1371/journal.pcbi.1000844.

[77] P. D. Schloss, "Secondary structure improves OTU assignments of 16S rRNA gene sequences," *ISME J.*, vol. 7, no. 3, pp. 457–460, 2013, doi: 10.1038/ismej.2012.102.

[78] R. C. Edgar, "Search and clustering orders of magnitude faster than BLAST," *Bioinformatics*, vol. 26, no. 19, pp. 2460–2461, 2010, doi: 10.1093/bioinformatics/btq461.

[79] L. Fu, B. Niu, Z. Zhu, S. Wu, and W. Li, "CD-HIT: Accelerated for clustering the next-generation sequencing data," *Bioinformatics*, vol. 28, no. 23, pp. 3150–3152, 2012, doi: 10.1093/bioinformatics/bts565.

[80] J. G. Caporaso *et al.*, "QIIME allows analysis of high-throughput community sequencing data.," *Nat. Methods*, vol. 7, no. 5, pp. 335–6, 2010, doi: 10.1038/nmeth.f.303.

[81] Pandas Documentation, "Pandas Documentation https://pandas.pydata.org," [Online]. Available: https://pandas.pydata.org.

[82] Scipy.org, "numpy.info() https://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.iinfo.html." https://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.iinfo.html.

[83] "Why python is slow," 2014. https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/.

[84] Pandas, "Pandas Categories." https://pandas.pydata.org/pandas-docs/stable/user_guide/categorical.html.

[85] V. ~I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Sov. Phys. Dokl.*, vol. 10, p. 707, Feb. 1966.

[86] J. A. Sena *et al.*, "Unique Molecular Identifiers reveal a novel sequencing artefact with implications for RNA-Seq based gene expression analysis," *Sci. Rep.*, vol. 8, no. 1, pp. 1–13, 2018, doi: 10.1038/s41598-018-31064-7.

[87] T. Kivioja *et al.*, "Counting absolute numbers of molecules using unique molecular identifiers," *Nat. Methods*, vol. 9, no. 1, pp. 72–74, 2012, doi: 10.1038/nmeth.1778.

[88] H. Wickham, "The split-apply-combine strategy for data analysis," *J. Stat. Softw.*, vol. 40, no. 1, pp. 1–29, 2011, doi: 10.18637/jss.v040.i01.

[89] G. Czajkowski, "Sorting Petabytes with MapReduce – The Next Episode," 2014. https://ai.googleblog.com/2011/09/sorting-petabytes-with-mapreduce-next.html.

[90] J. A. Griffiths, A. C. Richard, K. Bach, A. T. L. Lun, and J. C. Marioni, "Detection and removal of barcode swapping in single-cell RNA-seq data," *Nat. Commun.*, vol. 9, no. 1, pp. 1–6, 2018, doi: 10.1038/s41467-018-05083-x.

[91] J. Cao *et al.*, "Comprehensive single-cell transcriptional profiling of a multicellular organism," *Science (80-.).*, 2017, doi: 10.1126/science.aam8940.

[114] L. Naldini, "Ex vivo gene transfer and correction for cell-based therapies," *Nature Reviews Genetics*. 2011, doi: 10.1038/nrg2985.

[115] L. Naldini, "Gene therapy returns to centre stage," *Nature*. 2015, doi: 10.1038/nature15818.

[116] C. Sheridan, "Gene therapy finds its niche," *Nat. Biotechnol.*, 2011, doi: 10.1038/nbt.1769.

[117] M. Sessa *et al.*, "Lentiviral haemopoietic stem-cell gene therapy in early-onset metachromatic leukodystrophy: an ad-hoc analysis of a non-randomised, open-label, phase 1/2 trial," *Lancet*, 2016, doi: 10.1016/S0140-6736(16)30374-9.

[118] A. Biffi *et al.*, "Lentiviral hematopoietic stem cell gene therapy benefits metachromatic leukodystrophy," *Science (80-.).*, 2013, doi: 10.1126/science.1233158.

[119] A. Aiuti *et al.*, "Lentiviral hematopoietic stem cell gene therapy in patients with wiskott-aldrich syndrome," *Science (80-.).*, 2013, doi: 10.1126/science.1233151.

[120] J. Kaiser, "American Society of Gene Therapy meeting. Retroviral vectors: a double-

edged sword.," *Science (80-.).*, 2005, doi: 10.1126/science.308.5729.1735b.

[121] S. Hacein-Bey-Abina *et al.*, "Efficacy of Gene Therapy for X-Linked Severe Combined Immunodeficiency," *N. Engl. J. Med.*, 2010, doi: 10.1056/nejmoa1000164.

[122] R. Craigie and F. D. Bushman, "HIV DNA integration," *Cold Spring Harb. Perspect. Med.*, 2012, doi: 10.1101/cshperspect.a006890.

[123] M. Schmidt *et al.*, "High-resolution insertion-site analysis by linear amplification-mediated PCR (LAM-PCR)," *Nat. Methods*, 2007, doi: 10.1038/nmeth1103.

[124] S. Firouzi *et al.*, "Development and validation of a new high-throughput method to investigate the clonality of HTLV-1-infected cells based on provirus integration sites," *Genome Med.*, 2014, doi: 10.1186/gm568.

Appendix

Appendix A Gene Therapy

Gene therapy [114], [115] is the therapeutic delivery of genetic material into patient's cells (Figure A.1) to treat disease. The first attempt, an unsuccessful one, at gene therapy (as well as the first case of medical transfer of foreign genes into humans not counting organ transplantation) was performed by Martin Cline on 10 July 1980 [46]. Cline claimed that one of the genes in his patients was active six months later, though he never published this data or had it verified and even if he is correct, it's unlikely it produced any significant beneficial effects treating ß-Thalassemia. After extensive research on animals throughout the 1980s and a 1989 bacterial gene tagging trial on humans, the first gene therapy widely accepted as a success was demonstrated in a trial that started on 14 September 1990, when Ashanthi DeSilva was treated for ADA-SCID [116]. At SR-Tiget, are treated several pathologies, such as Metachromatic Leukodystrophy (MLD) [117], [118], Wiskott Aldrich Syndrome (WAS) [119]. To delivery the therapeutic material we use generally *Lentiviral Vectors* (LV) that are based on HIV virus, but rendered harmless. Integration into host genome, the distinctive feature of retroviral vectors, should be considered as a *double-edged sword* when it comes to gene therapy [120]. Genomic integration ensures the stability of transgene (material that has been transferred naturally, or by any of a number of genetic engineering techniques from one organism to another) and persistent transgene expression in daughter cells following genome replication and cell division, but its randomness results in the risk of *insertional mutagenesis* by potentially disrupting tumor suppressor genes or activating oncogenes [121].

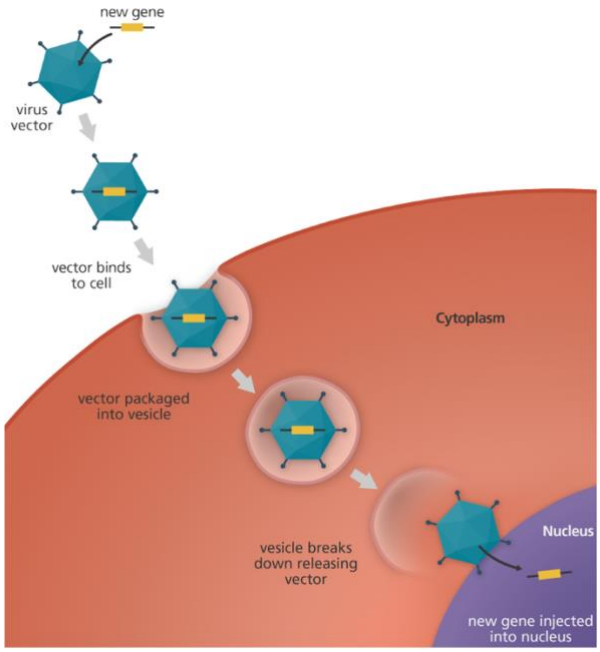


Figure A.1: How gene therapy works: the therapeutic lentiviral vector inserts the new gene into a cell. If the treatment is successful, the new gene will make a functional protein to treat a disease.
Figure 43

Insertional mutagenesis refers to mutation of an organism produced by the insertion of additional DNA material into the organism's preexisting DNA [122]. This process leads to the deregulation of genes in the neighborhood of the insertion sites and causes a perturbation of the cell phenotype, that can result into the rise of cancer. On the other hand, insertional mutagenesis is a forward genetic approach that has been used for the functional identification of novel genes involved in the pathogenesis of human cancers. The use of LVs (contrary to *Retroviral Vectors*, *1*-RV) and toxicity studies have allowed to obtain vectors now much safer and therefore able to be engineered perfectly for both behaviors (to treat a single-gene disease or to induce cancer).

Appendix B Lab procedures

Sonicated Linker-Mediated (SLiM)-PCR

The old LAM-PCR, although sensitive, allows only approximate clonal abundance estimations since before PCR amplification the genomic DNA is fragmented with restriction enzymes that, depending on the distance between the integrated vector and the site recognized by restriction enzyme will produce DNA fragments of different sizes upon amplification. Therefore, IS in long DNA fragments could be lost or amplified at low efficiency while short fragments would be favored and still produce sequences too short to be univocally mapped on the target cell genome. Moreover, the nucleotide composition at the vector/cell genome junction may impact on the PCR amplification efficiency impacting on the reliability of clonal quantifications based on sequence count statistics.

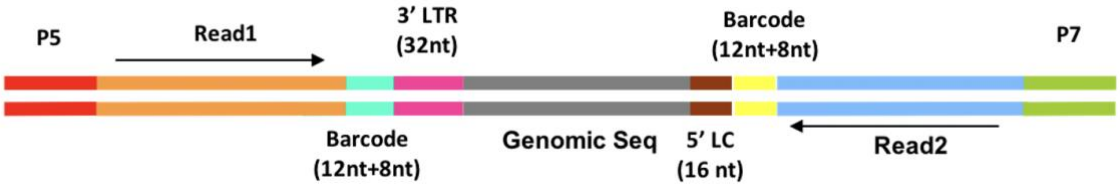


Figure B.2: SLiM-PCR, fragments (P5 and P7 are the Illumina adapters)

Figure 44

To avoid the biases produced by the exponential amplification and the use of restriction enzymes, my laboratory developed the new Sonicated Linker-Mediated (SLiM)-PCR. In this method, similar to [124], the genomic DNA of vector marked cells is sonicated to obtain randomly sheared fragments, ligated to a synthetic DNA linker cassette (LC, GTCACCGTGTCGTCAATCCT) needed as template for the successive PCR amplification. The tagged DNA is then used as template for PCR using oligonucleotides complementary to vector sequences and the linker cassette in order to specifically amplify the vector/cell genome junctions contained in between. Given that the random DNA fragmentation, achieved by sonication, occurs prior PCR amplification, a clonal population harboring the same IS will produce a number of DNA fragments containing

the vector/cell genome junctions of different sizes that will be proportional to the initial number of contributing cells. Therefore, counting the number of shear sites of the same IS allows to estimate the clonal abundance avoiding the PCR biases. Moreover, in the linker cassette ligated after DNA shearing, we included a 12 nucleotides sequence, a random barcode that is tagged to the sheared DNA fragments prior PCR amplification.

B.1.2 Fusion Primers for SLiM-PCR

```
>Fusion-LTR
AATGATACGGCGACCACCGAGATCTACACTCTTTCCCTACACGACGCTCTTCCGATCT
NNNNNNNNNNNNXXXXXXXXACCCTTTTAGTCAGTGTGGA

>Fusion-LC
GACGTGTGCTCTTCCGATCTNNNNNNNNNNNNA
```

Appendix C Information System

C.1 Computational Resources

This section illustrates the technologies used and the resources used during my PhD.

Computer 1 (Oracle)

Nodes	Cores	RAM	Operating System	Research Group
2	24	256GB*	Ubuntu Server 16.04 LTS	SR-Tiget

Table C.2: **Oracle**, HP Z840 Workstation. Intel(R) Xeon(R) CPU E52690 v3, 2.60GHz. The storage is composed of 5 HD with 4TB and 7K rpm and 1 primary disk of 500GB SSD. *Memory per node. **SR-Tiget.

Computer 2 (Cerbero)

Nodes	Cores	RAM	Operating System	Research Group
2	36	256GB*	Ubuntu Server 16.04 LTS	SR-Tiget

Table C.3: **Cerbero**, HP Z840 Workstation. Intel(R) Xeon(R) CPU E52690 v3, 2.60GHz. The storage is composed of 5 HD with 4TB and 7K rpm and 1 primary disk of 500GB SSD. *Memory per node.

Igenomix Server

Nodes	Cores	RAM	Operating System	Research Group
2	20	128GB	CentOS	Igenomix Italy Srl

Table B.4: Igenomix bioinformatics computational resources consist of 2 Intel Xeon Silver 4114 2.2G, 10c/20T, 9.6GT/s, 14M Cache, Turbo, Ht (85W) DDR4-2400

C.2 Storage

C.2.1 NAS Server: QNAP TS-412 4-BAY

CPU	DRAM	HDs	Operating System	File Sharing
Marvell 6281 1.2GHz	512MB DDRII	4x 3.5" SATAII 4x 2.5" SATAII	Embedded Linux	CIFS/SMB, AFP, NFS

Table B.5: TS-412 is a powerful yet easy to use networked storage center for backup, synchronization and remote access. It supports comprehensive RAID configuration and hot-swapping to allow hard drive replacement without system interruption. This NAS is available online with CIFS/SMB or AFP protocols for UNIX systems

Acknowledgements

Mi ero laureato alla magistrale in Bioinformatica a pieni voti da qualche mese; ed ero molto scettico se iniziare un dottorato o cercare lavoro in azienda. Tuttavia, avevo promesso a me stesso che avrei fatto il dottorato solo se ne avessi trovato uno in computer science. Ed eccoci qui, come se mi fosse piovuto dal cielo, questo viaggio è iniziato nel luglio del 2017 e carico di entusiasmo e aspettative altissime ho iniziato quello che è stato a tutti gli effetti il percorso più difficile della mia vita.

Un dottorato è qualcosa che ti cambia come persona, entri ragazzino e ne esci adulto.

O almeno pensi di esserlo.

Impari quella che è la dura legge del mondo lavorativo, la legge del più forte e quella che nessun collega ti è amico veramente, a parte rare eccezioni.

Impari a superare i tuoi limiti, ad andare oltre le paure.

Ricordo dell'ansia che mi paralizzava le gambe durante la mia prima presentazione ad una conferenza mondiale a Washington D.C. davanti a 1000 persone.

Ricordo la pressione nel cercare di rispettare le Deadline.

Ricordo tutte le notti in bianco a programmare per finire quello script

Ricordo tutte le notti a preparare la presentazione per il giorno dopo

Ricordo la prima domanda che mi fecero alla prima conferenza

Ricordo l'adrenalina prima di partire per una summer school

Ricordo il senso di relax fortissimo dopo aver finito un meeting lunghissimo

Ricordo le centinaia di cene fatte davanti al monitor

Ricordo tutte le volte che ho pensato: "basta ora lascio tutto e me ne vado a fare il pastore in Tibet"

Ci saranno momenti belli, momenti brutti, momenti in cui penserai di abbandonare. Il segreto è provare a spegnere il cervello e continuare senza pensarci troppo, perché quando sei arrivato in cima la vista da lassù ti darà indietro tutte le consapevolezze di cui avevi bisogno.