POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Matematica
Modelli Matematici e Simulazioni Numeriche

Tesi di Laurea Magistrale

Identification of tumour molecular markers that predict PARP inhibitors response

Relatore
Prof. Luigi Preziosi

Candidato
Gaia Stievano

Co-relatori
Prof. Alberto Bardelli
Dott. Giorgio Corti

Anno Accademico 2021-2022

Ai miei nonni
Nitta e Batti

Abstract

The process of cellular duplication occurs in all living organisms and forms the basis for biological inheritance. This mechanism involves DNA replication that, being crucial for the preservation of genetic material, counts a great number of control mechanisms to verify its correctness. Discontinuities in a strand of the DNA double helix, known as single-strand breaks (SSBs), can sometimes occur and, if not repaired appropriately, could pose a serious threat to genetic stability and cell survival. Consequently, cells have evolved efficient mechanisms for their fixation, involving Poly ADP-ribose polymerase (PARP), a family of proteins whose main role is to detect and signal SSBs to the enzymatic machinery involved in their repair (SSBR). The importance of this process is highlighted by the fact that unrepaired SSBs could cause in proliferating cells the blockage or collapse of DNA replication forks, leading to the formation of double-strand breaks (DSBs). Although cells possess a remarkable capacity to accurately repair such DSBs using homologous recombination (HR), acute increases in cellular levels of SSBs could saturate this pathway, leading to genetic instability and/or cell death. Defects in HR repair mechanism, caused by mutations in the pathway, confer the so-called "BRCAness" phenotype. Inactivation of this pathway causes HR deficiency (HRD), resulting in high levels of genomic abnormalities. Recent studies have demonstrated the importance of a good predictor of the biological status of an HR-deficient tumour: the set of tumours that show BRCAness and that can be selectively sensitive to PARP inhibitors includes a wide range of sporadic breast, ovarian and also colorectal, cancers. All these features led Nik-Zainal's research group to use a weighted model, called "HRDetect", to identify mutational signatures predictive of BRCA deficiency. This type of analysis requires the tumour and the matched normal samples to correctly compare and extract the somatic variations caused by the tumour, discarding the germinal ones.
The work carried out within this thesis is aimed at overcoming this limitation, being able to correctly predict the score provided by the HRDetect algorithm in situations in which the normal sample is not available (e.g. cell lines). Through the introduction of a new "metanormal" sample it has been possible to unlink the tumour sample from the matched normal, replacing it in the comparison. However, since the tumour is not "matched" with the respective normal, series of germline mutations are not properly filtered as such appearing in the set of extracted somatic variations.
To correctly emulate the expected HR scores, obtained by direct comparison between the tumour sample and the normal matched sample of the considered patient, different strategies were developed. These latter made it possible to extract somatic mutations increasingly accurately, reducing the prediction errors related to the presence of germline mutations. Each strategy was applied to all the breast cancer samples by running the algorithms via the Linux shell, supported by the Python programming language and the interpreted language "AWK". The application of the final strategy correctly predicted HR deficiency/proficiency in 98.6% of the 77 samples considered. The consistency of these results, through future implementation of the strategy to a larger number of samples, could contribute significantly to biomedical research.

Contents

List of Tables 8

List of Figures 9

1 Introduction 11
- 1.1 What is cancer? 11
  - 1.1.1 Cancer classification 12
  - 1.1.2 Cancer biology 12
- 1.2 Key concepts of genetics 14
  - 1.2.1 All genetic information, or "genome" 14
  - 1.2.2 Chromosomes: the genome building blocks 16
  - 1.2.3 The concept of gene 16
  - 1.2.4 Somatic and germline mutations 19
  - 1.2.5 Principal mutation classes 19
  - 1.2.6 The concept of signature 22

2 Homologous Recombination Deficiency 25
- 2.1 BRCAness phenotype 25
- 2.2 BRCAness in breast cancer 27
- 2.3 Quantitatively define HR Deficiency 29

3 Predicting BRCA1/BRCA2 deficiency in breast cancer 31
- 3.1 A HR status classifier 31
- 3.2 Other genetic factors related to BRCAness 32
- 3.3 Application of HRDetect in other cancer types 32
- 3.4 Robustness, stability and generalisability 33

4 How is the HRDetect score predicted? 35
- 4.1 From WGS analysis to the score 35
- 4.2 Validation on original mapped data 37
- 4.3 HRDetect score validation on raw reads data 38

5 Prediction on patients without matched normal tissue 41
5.1 Overcome the HRDetect limitations . . . 41
5.2 Dataset description . . . 44
5.3 Strategy A . . . 46
5.3.1 Algorithms . . . 47
5.3.2 Results obtained with Strategy A for breast cancer . . . 57
5.4 Strategy B . . . 59
5.4.1 Methods . . . 60
5.4.2 Results . . . 64
5.5 Strategy C . . . 65
5.5.1 Methods . . . 66
5.5.2 Results . . . 73
5.6 Strategy D . . . 74
5.6.1 Methods . . . 78
5.6.2 Results . . . 85
5.7 Ensemble strategy . . . 86
5.7.1 Methods . . . 89
5.7.2 Results . . . 90

6 Applying this method to colorectal cancer patients 97
6.1 Introduction . . . 97
6.2 Dataset description . . . 100
6.3 Results on CRC dataset . . . 101

Conclusions 105

Future Perspectives 107
Exploiting HRDetect score on CRC cell lines bank . . . 107
Predicting BRCAness on Formalin Fixation and Paraffin Embedding samples . . . 108

A Description of file formats 111
A.1 The FASTQ format . . . 111
A.2 The BAM format . . . 112
A.3 The VCF format . . . 115

B Model description 119
B.1 Lasso logistic regression modeling . . . 119

C Code Implementation 123
C.1 Introduction . . . 123
C.2 Get_indel_from_freq-distr.py . . . 124
C.3 Get_SNV_from_freq-distr.py . . . 128
C.4 Recalibrate_HRD_A.sh . . . 131
C.5 Get_VAF-range_indel.py . . . 137
C.6 Get_VAF-range_SNV.py . . . 139
C.7 Recalibrate_HRD_B.sh . . . 140

C.8 Recalibrate_HRD_C.sh . . . 145
C.9 Get_Indel_freq-distr_anyPerc.NORM.sh . . . 150
C.10 Get_SNV_freq-distr_anyPerc.NORM.sh . . . 152
C.11 Get_Indel_freq-distr_anyPerc.sh . . . 154
C.12 Get_SNV_freq-distr_anyPerc.sh . . . 154
C.13 _get_mirror_distribution.py . . . 154
C.14 _subtract_distribution.py . . . 155
C.15 Get_freq-distr_anyPerc_from_distribution.NORM.sh . . . 157
C.16 Subtract_germline_distribution.SNV.sh . . . 159
C.17 Subtract_germline_distribution.Indel.sh . . . 159
C.18 Recalibrate_HRD_D.sh . . . 160
C.19 Ensemble_strategy.sh . . . 162

Bibliography 167

List of Tables

5.1 Dataset Description . . . 45
5.2 Indel/SNV median freq from N-T . . . 51
5.3 Indel/SNV median freq from N-T_A . . . 52
5.4 Results obtained with Strategy A for breast cancer. . . . 92
5.5 Results obtained with Strategy B for breast cancer . . . 93
5.6 Results obtained with Strategy C for breast cancer . . . 94
5.7 Results obtained with Strategy D for breast cancer. . . . 95
5.8 Results obtained with Ensemble Strategy for breast cancer . . . 96
6.1 CRC dataset description . . . 101
6.2 Results obtained with Ensemble Strategy for CRC . . . 102
A.1 Alignment section of the SAM format . . . 113
A.2 Body of the VCF format . . . 117
B.1 Lasso logistic regression . . . 122

List of Figures

1.1 Hallmarks of cancer . . . 13
1.2 Human Genome . . . 15
1.3 Chromosomes structure . . . 17
1.4 Principal mutation classes . . . 20
1.5 Examples of Single Base Substitution (SBS) Signatures . . . 23
2.1 PARP inhibitors: Treatment for BRCA mutant Breast Cancer . . . 27
4.1 Workflow of the Nik-Zainal's research group . . . 36
4.2 HRDetect score validation . . . 37
4.3 Pipeline comparison . . . 38
5.1 Normalized VAF for N-T . . . 42
5.2 Normalized VAF for MTN-T . . . 43
5.3 Normalized VAF for MTN-T and N-T . . . 43
5.4 Somatic and germline VAF . . . 44
5.5 Dataset balancing . . . 46
5.6 Selection of a specific range of frequency from the normalized VAF . . . 47
5.7 Recalibrate_HRD_A.sh core workflow . . . 54
5.8 Content of recalibrate A (PD24215a) . . . 55
5.9 Content of SNV-sampling, Indel-sampling (PD24215a) . . . 56
5.10 Content of RANDOM1.-files (PD24215a) . . . 57
5.11 Content of file.out and file.data-matrix (PD24215a) . . . 58
5.12 Selection of a specific range of frequency from the normalized VAF . . . 59
5.13 Recalibrate_HRD_B.sh core workflow . . . 62
5.14 Content of SNV-sampling, Indel-sampling (PD24215a) . . . 63
5.15 Recalibrate_HRD_C.sh core workflow . . . 67
5.16 Content of recalibrate C (PD24215a) . . . 69
5.17 Content of SNV-sampling, Indel-sampling (PD24215a) . . . 70
5.18 Content of RANDOM-*.files (PD24215a) . . . 71
5.19 Content of file.out and file.data-matrix (PD24215a) . . . 72
5.20 Comparison between average and sample-specific distributions . . . 74
5.21 Comparison between sample-specific distributions . . . 75
5.22 Representation of all the curves . . . 76
5.23 Representation of all the sample-specific curves . . . 77
5.24 Recalibrate_HRD_D.sh core workflow . . . 84
5.25 Stacking workflow . . . 87

5.26 Representation of the errors 88
5.27 Representation of the errors (total) 91
6.1 The age-standardized incidence and mortality rates 98
6.2 Colon cancer disease progression 99
6.3 Representation of the project roadmap 108
6.4 Representation of the Mutational signatures 109

Chapter 1

Introduction

This chapter is aimed at reviewing some basic aspects of biology and focuses attention on two main points.
On one hand, some of the fundamental concepts that characterize modern genetics are introduced. More precisely, the definitions of *genome*, *gene*, *chromosome*, as well as the concepts of *genetic signature*, *somatic/germline mutation* and *mutational class*, are provided.
On the other hand, some of the peculiar characteristics that give cancer its leading role among the main causes of death in the world are represented.

For complete information on the subject, one can refer to: [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12].

1.1 What is cancer?

Cancer is a vast category of diseases that can begin in practically any organ or tissue of the body and spread to other organs when abnormal cells proliferate uncontrollably, invade adjacent regions of the body, and/or move to other organs.
When cancer cells proliferate and reproduce themselves, they often form into a clump known as a tumour.
Tumours pressurise, crush, and destroy non-cancerous cells and tissues, causing many of the symptoms of cancer.
Tumours are classified as benign or malignant. Because benign tumours are not cancerous, they do not develop or spread to the same amount as cancerous tumours, and they are rarely life-threatening.
On the other hand, malignant tumours grow and spread to other parts of the body. Metastasis is the process by which cancer cells spread from the initial tumour location to other sections of the body, and it is a leading cause of cancer death.

1.1.1 Cancer classification

Cancer remains the number two cause of death in the world, second only to heart disease [1]. It refers to a wide range of disorders that all stem from unregulated cellular development.
Commonly divided into benign or malignant tumours, cancers are further defined and classified by their cell type, tissue or organ of origin.
Following this last partition, four main types of cancer to mention are [2]:

- *Carcinomas.* A carcinoma originates in the skin or the tissue that covers the surface of internal organs and glands. Carcinomas are the most prevalent kind of cancer, forming solid tumours in most cases.
- *Sarcomas.* Sarcomas begin in the body's supporting and connecting tissues. Fat, muscles, nerves, tendons, joints, blood arteries, lymph vessels, cartilage, and bone can all form sarcomas.
- *Leukemias.* Leukemia is a kind of blood cancer. When healthy blood cells begin to alter and expand uncontrolled, leukemia develops.
- *Lymphomas.* Lymphoma is a cancer that starts in the lymphatic system, which is a network of tubes and glands that aids in the fight against infection.

1.1.2 Cancer biology

Cancer has afflicted multicellular living beings for about 200 million years, and evidence of cancer among progenitors of contemporary humans dates back over a million years. Cancer is now the world's second biggest cause of mortality, accounting for an estimated 9.6 million fatalities in 2018, or one in every six deaths [3]. Men's cancers include lung, prostate, colorectal, stomach, and liver cancer, whereas women's cancers include breast, ovarian, colorectal, lung, cervical, and thyroid cancer.
Cancer cells are distinct from normal cells in a variety of ways (as shown in Figure 1.1). Cancer cells, for example, grow in the absence of signals instructing them to do so, but normal cells only grow in response to such signals. Cancer cells ignore signals that normally tell cells to stop dividing or die (a process known as programmed cell death, or apoptosis), invading nearby areas and spreading to other parts of the body, whereas normal cells stop growing when they come into contact with other cells and do not move around the body. Cancer cells also instruct blood arteries to expand in the direction of tumours. This allows blood arteries to deliver oxygen and nutrition to tumours while also removing waste materials.
The immune system generally destroys damaged or aberrant cells, but cancer cells can hide from the immune system and deceive it into assisting them in staying alive and growing. Some cancer cells, for example, persuade immune cells to protect the tumour rather than attack it.
Some cancer cells have twice as many chromosomes as normal cells and accumulate various chromosome alterations, such as duplications and deletions of chromosome sections. Furthermore, some cancer cells rely on different types of nutrients than regular cells, and

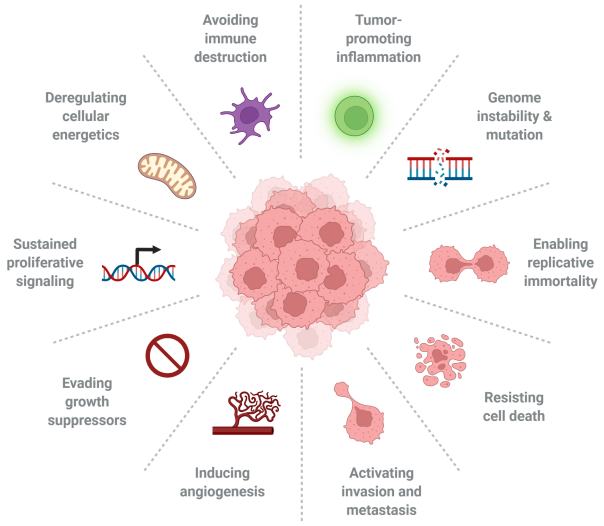

Figure 1.1. Hallmarks of cancer [13]

they derive energy from nutrients in a different method than the majority of normal cells. This allows cancer cells to multiply at a faster rate.
Cancer cells often rely on aberrant activities so heavily that they can't survive without them. This has led to the development of medicines that target the aberrant characteristics of cancer cells.
Cancer is a genetic disease, which means it's a subtype of uncommon disease caused by one or more genotype abnormalities, such as gene mutations or chromosome modifications, that might induce one or more pathologies.
Cancer-causing genetic alterations can arise as a result of parental inheritance, DNA damage produced by dangerous substances in the environment, including chemicals in cigarette smoke and UV rays from the sun, or errors that occur during cell division.
The body generally destroys cells with damaged DNA before they develop malignant, but this ability is compromised by ageing, which explains why cancer risk increases as one gets older.
It's crucial to remember that each person's cancer has a unique set of genetic changes, and that as the cancer progresses, more changes will occur, and that different cells within the same tumour may have different genetic mutations.
Proto-oncogenes, tumour suppressor genes, and DNA repair genes are the three primary types of genes that are affected by genetic alterations that contribute to cancer.
Proto-oncogenes play a role in normal cell division and proliferation. These genes can become cancer-causing genes (or oncogenes) if they are mutated in specific ways or are more active than usual, allowing cells to grow and survive when they shouldn't.
Tumour suppressor genes are also involved in cell growth and division; in fact, cells with particular tumour suppressor gene mutations may divide uncontrollably.
DNA repair genes are responsible for repairing damaged DNA. Cells with mutations in

these genes are more likely to generate mutations in other genes and chromosome alterations, such as chromosome duplications and deletions. These alterations may lead the cells to become malignant if they occur together.
Thirty years ago, scientists couldn't come up with a logical explanation for what causes a cell to become malignant. Cancer was known to be caused by cells proliferating uncontrollably within the body, and that chemicals, radiation, and viruses could cause this alteration, but the particular mechanism was unknown.
However, research during the last three decades has transformed our understanding of cancer. This breakthrough was made possible in large part by the invention and deployment of molecular biology tools, which allowed researchers to investigate and describe aspects of individual cells in ways that were unthinkable a century earlier.
We now understand cancer to be a disease of molecules and genes, and we even know more about these.
Indeed, as we learn more about these genes, we will be able to design intriguing new ways for preventing, delaying, and even reversing the alterations that lead to cancer.

1.2 Key concepts of genetics

Genetics is a field of biology that studies genes, genetic diversity, and heredity in living things.
The discovery of genes, the basic components responsible for inheritance, gave rise to genetics. It may be described as the study of genes at all levels, including how they function in cells and how they are handed on from parents to children.
Modern genetics focuses on deoxyribonucleic acid, or DNA, the chemical material that makes up genes, and how it impacts the chemical reactions that make up biological activities within the cell.
The completion of the sequencing of the human genome, and the understanding of the genes it encodes, is leading to a new era in medicine, which includes the ability to use individual genetic profiles to predict, prevent, and prognosticate disease.

1.2.1 All genetic information, or "genome"

The human genome is the complete sequence of nucleotides that makes up the genetic makeup of Homo sapiens, including nuclear DNA and mitochondrial DNA.
It has an inventory of approximately 3.2 billion DNA base pairs containing approximately 20,000 protein-coding genes.
A reference genome (also known as a reference assembly) is a digital nucleic acid sequence database created by scientists to represent the set of genes in a single idealised individual organism of a species [5].
Reference genomes do not correctly reflect the set of genes of any single particular organism since they are generated through the sequencing of DNA from a number of different contributors.
A reference, on the other hand, gives a haploid mosaic of various DNA sequences from each

donor. Multiple species of viruses, bacteria, fungi, plants, and mammals have reference genomes.
The Human Genome Project (2003) has identified a reference sequence, which is used globally in the biomedical sciences. The study also found that non-coding DNA totals 98.5%, more than expected, and therefore only about 1.5% of the total DNA length is based on coding sequences. Reference genomes are frequently used as a blueprint for creating new genomes, allowing them to be assembled considerably more swiftly and inexpensively than the Human Genome Project.
The reference genome gives a decent approximation of each single individual's DNA for a large portion of the genome.
However, in areas where allelic diversity is substantial, the reference genome may differ greatly from that of other people.
Reference genomes may be found in a variety of places online, including Ensembl ([14]) and the UCSC Genome Browser ([15]).
On December 17, 2013, the Genome Reference Consortium released the human reference genome GRCh38. It's the most recent version, and it's the one I'm going to use in the following study.

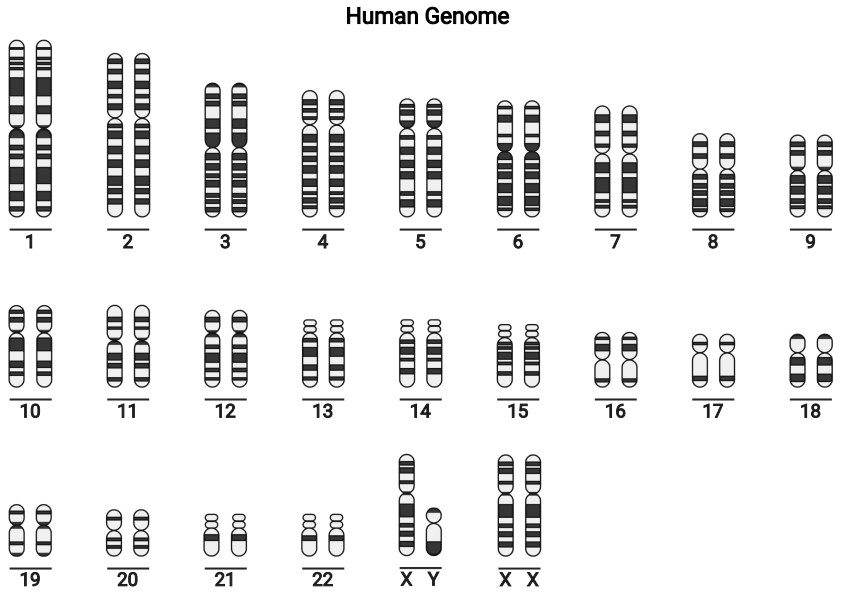

Figure 1.2. Chromosomal equipment of the Human Genome

1.2.2 Chromosomes: the genome building blocks

The nuclear DNA of the human genome is grouped into 24 types of chromosomes (Figure 1.2): 22 autosomes, plus two sex-determining chromosomes (X chromosome and Y chromosome) [16].
A chromosome is made up of a single, very long DNA helix that contains thousands of genes (Figure 1.3). A set of big, linear chromosomes stores the bulk of eukaryotic genes.
In addition to genes, eukaryotic chromosomes contain sequences that guarantee that DNA is replicated without end sections being destroyed and sorted into daughter cells during cell division.
The centromere is required for proper chromatid segregation during mitotic and meiotic anaphase. Telomeres are long stretches of repetitive sequences that cap the ends of linear chromosomes and prevent degradation of coding and regulatory regions during DNA replication. Replication origins are sequence regions where DNA replication is initiated to make two copies of the chromosome.
In particular, man is part of living organisms which have two copies of each chromosome in somatic cells, called homologous chromosomes. This condition is called diploidy.
Diploidy is most commonly caused via sexual reproduction. The copies of a chromosome are genetically identical (but with different alleles from the two parents). An exception is in the case of sex-linked chromosomes in the male, where, instead of two copies of the X chromosome as in the female, an X is associated with a different, much shorter chromosome called a Y.
There are also haploid animals and cells, in which each chromosome is duplicated just once. Gametes, which combine during the fertilisation process to generate the zygote (the first diploid cell in the new creature that includes a copy of both parents' genetic makeup), are an example of haploid cells.
Polyploidy, in addition to diploidy and haploidy, is common in cultivated plants: these organisms have more than two copies of their chromosomal composition.

1.2.3 The concept of gene

On each chromosome, at specific locations called *locus*, are placed units of genetic information: the genes [6].
Members of a population can have various alleles at the locus, each with a slightly different sequence.
This information is found in the nucleus of eukaryotes (such as animals, plants, and fungi). Small subsets of genes separate from those present in the nucleus can also be discovered in the mitochondria (in mammals) and chloroplasts (in plants).
The number of genes in an organism's genome varies greatly between species. The genome of the bacteria Mycoplasma genitalium, for example, includes just 517 genes, compared to the estimated 20,000 to 25,000 in the human genome.
As introduced earlier, DNA represents the building block of chromosomes. It is made up of four different nucleotide subunits, each of which contains a five-carbon sugar, a phosphate group, and one of the four bases adenine, cytosine, guanine, or thymine (Figure 1.3). Because adenine and thymine align to form two hydrogen bonds, but cytosine and

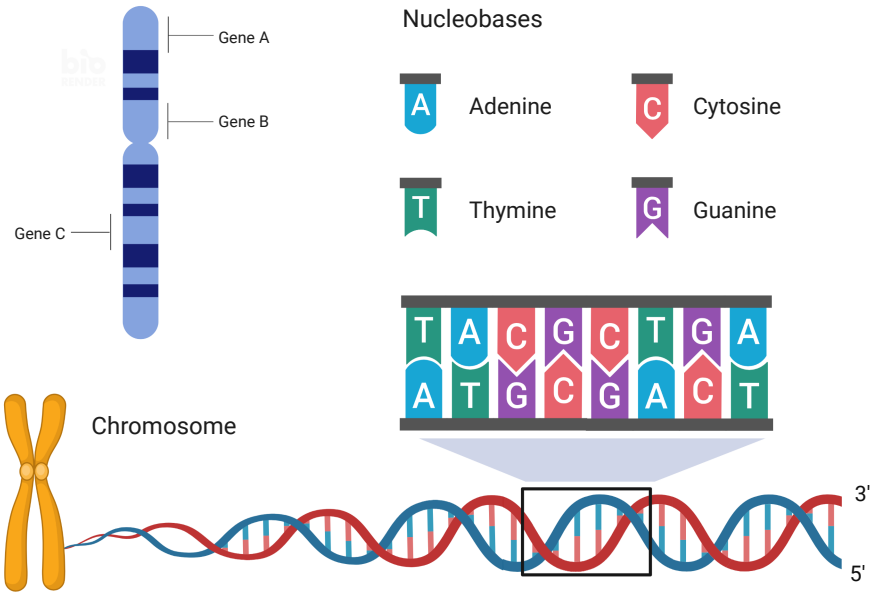

Figure 1.3. Chromosomes structure

guanine form three hydrogen bonds, base pairing is particular.
In a double helix, the two strands must be complementary, with their base sequences matching in such a way that it is possible to reconstruct the nucleotide sequence of the complementary strand of a DNA strand whose sequence is known. In particular, for each nucleotide there is only one complementary nucleotide, i.e. that can form a hydrogen bond with the first: adenine and thymine are complementary to each other, as well as guanine and cytosine.
Transcribing genes contained in DNA into RNA, a second kind of nucleic acid that is extremely similar to DNA but whose monomers contain the sugar ribose rather than deoxyribose, is the first step in the expression of genes encoded in DNA.
In addition to thymine, RNA contains the nucleotide uracil.
RNA molecules are single-stranded and less stable than DNA ones. Codons, which are three-nucleotide sequences that act as the "words" in the genetic "language", are found in genes that code for proteins.
The genetic code determines the relationship between codons and amino acids during protein translation.
All known creatures have almost identical genetic codes. The genetic code is determined by the nucleotide sequence along a strand of DNA. When the product of a gene is required, the segment of the DNA molecule containing that gene splits. A strand of RNA with bases complementary to those of the gene is produced from free nucleotides in the cell during transcription. Introns (non-coding nucleotide sequences) are removed from the

main transcript after transcription by editing and splicing mechanisms. A functioning strand of mRNA is the consequence of these activities. This is a common step in the creation of mRNA for most genes, but other genes have several methods to splice the original transcript, resulting in distinct mRNAs and, in turn, different proteins. The mRNA then travels to ribosomes, where the translation, or protein synthesis, process takes place. A second form of RNA, transfer RNA (tRNA), lines up the nucleotides on mRNA with particular amino acids during translation.
One amino acid is coded for by each set of three nucleotides. A polypeptide chain is made up of one or more linked amino acids formed according to the sequence of nucleotides; all proteins are made up of one or more linked polypeptide chains.
Experiments have revealed that many genes in organisms' cells remain dormant for most of the time, if not all of the time. Thus, it appears that a gene may be turned on or off at any moment in both eukaryotes and prokaryotes; this process is known as gene regulation. In higher organisms, the sequence of events connected with gene expression is regulated at numerous levels and is frequently impacted by the presence or absence of molecules known as transcription factors. These factors can operate as activators or enhancers at the most fundamental level of gene regulation, which is the rate of transcription. At different periods and in certain types of cells, certain transcription factors control the creation of RNA from genes. Transcription factors frequently bind to the promoter, or regulatory area, of higher organisms' genes. Some also are controlled at the translational and posttranslational levels.
Although DNA replication is often quite exact, mistakes (mutations) do occur. In eukaryotic cells, the error rate per nucleotide per replication can be as low as 10^{-8}, but for certain RNA viruses, it can be as high as 10^{-3}. This indicates that each human cell adds 1–2 additional mutations every generation.
Point mutations, in which a single base is changed, and frameshift mutations, in which a single base is inserted or deleted, are examples of small alterations generated by DNA replication and the aftermath of DNA damage.
Missense (changing a codon to encode a different amino acid) or nonsense (a premature stop codon) mutations can both affect the gene.
Larger mutations can be induced by chromosomal abnormalities such as duplication, deletion, rearrangement, or inversion of substantial parts of a chromosome due to recombination mistakes.
Furthermore, when fixing physical damage to the molecule, DNA repair systems might produce mutational mistakes. When healing double-strand breaks, for example, the repair is more vital to life than restoring a perfect replica.
One of the possible causes of gene mutations may not be genetic but instead epigenetic in origin, i.e. it can be related to heritable phenotypic modifications that do not entail changes in the DNA sequence [17].
Modifications in gene activity and expression are the most common epigenetic changes, although the phrase can also refer to any heritable phenotypic change.
External or environmental influences may have an effect on cellular and physiological phenotypic features, or they may be a natural aspect of development.
The term epigenetic also refers to the alterations themselves, which are functionally meaningful changes to the genome that do not entail a nucleotide sequence change. Even though

these modifications do not entail changes in the underlying DNA sequence of the organism, they can endure for numerous generations.
Instead, non-genetic factors affect the organism's genes to function (or "express themselves") differently. The process of cellular differentiation is an example of an epigenetic modification in eukaryotic biology.

1.2.4 Somatic and germline mutations

The mutational theory of cancer suggests that *driver* mutations in DNA sequence confers proliferative advantage on a cell, resulting in the development of a neoplastic clone. This latter, in general, is a collection of identical cells with a shared ancestor, implying that they are all descended from the same cell.
Many cancers are theoretically a single clone of cells since they are derived from a single sufficiently altered cell.
However, during cell division, one of the cells can become more altered and acquire new traits, resulting in the formation of a new clone.
In recent years, this notion of cancer initiation has been questioned and several tumours, including malignant mesothelioma, have been claimed to have polyclonal origin, i.e. produced from two or more cells or clones.
Some driver mutations are inherited in the germline, but the vast majority emerge in somatic cells during the cancer patient's lifetime, along with a slew of "passenger" mutations that aren't linked to cancer formation.
A mature or developing person's germline is the line (sequence) of germ cells that contain genetic material that can be handed down to a descendent.
A germ cell is a member of the differentiation line that is responsible for the transmission of genetic material to children. The expression is used in contrast to somatic cell, which refers to the cells that make up an organism's body, or soma.
Specifically, depending on the context considered, the term "*somatic*" is used also to refer to cancer cells.
The numerous tissues which in complex organisms go to constitute the organs and, in turn, the systems, are made up of aggregates of somatic cells. Only germline mutations may be passed down from generation to generation [7].
Endogenous and exterior mutagen exposures, aberrant DNA editing, replication errors, and defective DNA maintenance are among the mutational processes that generate these mutations.

1.2.5 Principal mutation classes

A defect in a single gene can occur in different mutational signatures of all classes, including base substitutions, insertions/deletions (commonly named "Indel" mutations), copy number rearrangements and rearrangements.
These multiple mutational signatures are distinguished from most biomarkers as they are the direct consequence of non-repair of the DSB. In particular, the term Single Nucleotide Variant (SNV)[9] refers to a specific case of base substitution for which a single nucleotide (adenine, thymine, cytosine, or guanine) in the genome sequence is altered.

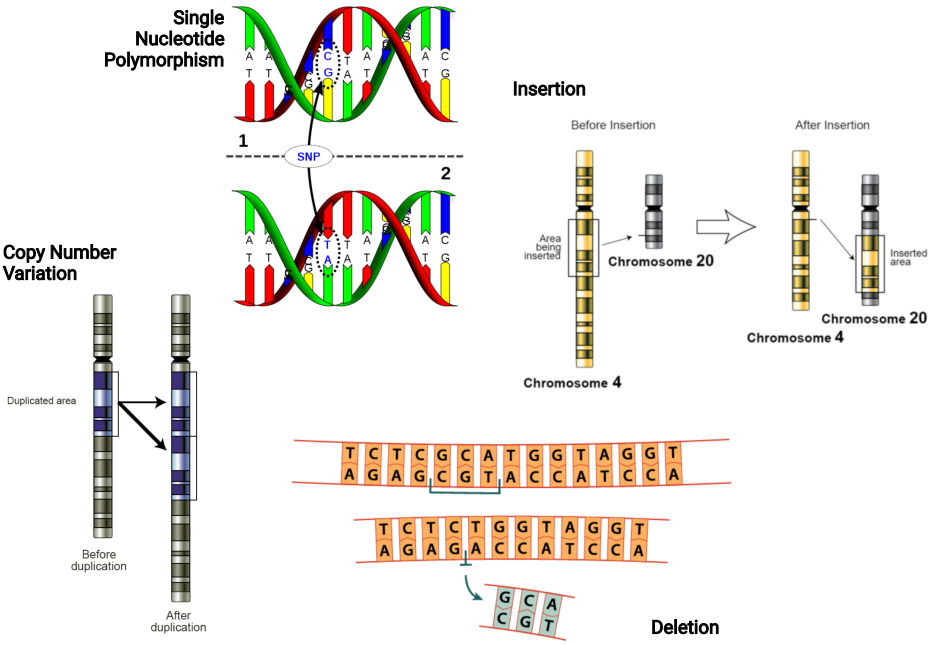

Figure 1.4. Principal mutation classes

These DNA sequence variations may be rare or common in a population and sometimes they are referred to as common single nucleotide polymorphisms (SNPs)[10].
A SNP is a variation of the genetic material of a single nucleotide, such that the polymorphic allele is present in the population in a proportion greater than 1% (Figure 1.4).
SNPs can occur within a coding sequence of a gene, within an intronic region, or within an intergenic region.
When SNPs occur within a gene or in a regulatory region near a gene, they may play a more direct role in disease by affecting the gene's function [12].
SNPs within a gene, however, do not necessarily modify the coded amino acid sequence, since the genetic code is degenerated.
Since RNA is a linear polymer of four distinct nucleotides and since each nucleotide triplet (codon) designates an amino acid, there are 4 to the third power (64) potential triplets.
However, because proteins contain just 20 distinct amino acids, most amino acids are encoded by more than one codon, indicating that the genetic code is degenerated.
A SNP that generates the same peptide in all its forms is called a synonym (synonymous).
SNPs that are not in a coding sequence can, however, have negative consequences on splicing or binding of transcription factors.
Common SNPs make up 90% of all human genetic variations. SNPs with an allele frequency of 1% or more occur every 100-300 base pairs or so throughout the genome. On average, two out of three SNPs show a variation between cytosine and thymine.

Most SNPs have no effect on health or development, however, some of these genetic differences have proven to be very important in the study of human health.
Researchers have found SNPs that may help predict an individual's response to certain drugs, susceptibility to environmental factors such as toxins, and risk of developing particular diseases.
SNPs can also be used to track the inheritance of disease genes within families.
The main classes of mutation include Indel mutations. The term Indel refers instead to the abbreviation of a mutation/recombination event that can be part of two classes: an insertion or a cancellation, that may have occurred over the years and which have produced differences in DNA sequences under examination (Figure 1.4).
Single mutations, even in non-coding and hence seemingly unimportant areas like the promoter, can have profound phenotypic consequences. In reality, the promoter is crucial because it binds a series of proteins when a gene has to be switched on, due to precise patterns that proteins detect. It is possible to lose the connection between the transcription factor, which is needed to activate transcription, and the promoter if the latter is mutated. As a result, a point mutation on the promoter might cause the definitive gene to shut down completely, implying that this protein will no longer be generated since it will never be able to bind a regulatory protein or initiating factor essential for transcription to the mutation location.
It's also possible that, contrary to popular belief, the promoter mutation can bind to the initial transcription factor very well but will never unbind it, implying that the gene will be transcribed even when it's not needed, resulting in the production of a large amount of gene product, which can lead to disease development.
In addition to single nucleotide substitution, insertion and deletion can occur as a result of the DNA polymerase slipping on the replicating DNA.
The presence of a series of repeated nitrogenous bases in some areas of DNA is common, which enhances the likelihood of a polymerase mistake during the copying step (for example instead of copying 4 bases it can copy 3 or 5). This procedure can result in the insertion (insertion) or deletion (deletion) of a base, but it can also entail a larger number of bases. Indel mutations are common because the polymerase, although having proofreading function, is still an enzyme that tends to slide (slippage) and therefore to generate mistakes when repeated bases are present, resulting in mutations that can lead to disease development.
Depending on the reason that generated the mutation/recombination, it is possible to distinguish 3 classes of indel mutations:

- **Indel short**, 1-5 consecutive bases long that may have been caused by a genome transcription error;
- **Indel medium**, long from 100 to 30k bases that may have been caused by the insertion of transposable elements;
- **Indel long**, more than 30k bases long and which may have been caused by genetic recombination.

On the other hand, more generally, Structural Variants (SV) ([8], [11]) are referred to as changes in the genomic landscape that can alter gene expression levels and thus lead to

disease development.
The most common and best studied SVs in blood malignancies are chromosomal translocations; here, parts of two genes that are normally on different chromosomes come into close proximity due to a failure in DNA repair. As a consequence, fusion proteins, which show a different function and/or cellular localization compared to the two original proteins, are expressed, sometimes even at different levels.
The identification of chromosomal translocations is often used to identify the specific disease a patient is suffering from.
Copy number variation (CNV) is a phenomenon in which sections of the genome are repeated (Figure 1.4). It is a type of structural variation: specifically, it is a type of duplication or deletion event that affects a considerable number of base pairs.
Approximately two-thirds of the entire human genome may be composed of repeats and 4.8–9.5% of the human genome can be classified as copy number variations.

1.2.6 The concept of signature

As outlined above, somatic mutations found in cancer genomes may be the consequence of different processes, such as incorrect DNA replication, exposure to endogenous or exogenous mutagens, but also enzymatic modification of DNA, or its incorrect repair [18].
Different mutational processes generate observable patterns of mutation within the genome. These patterns are called *mutational signatures* [19].
Until recently, human cancer mutational signatures were studied using a small number of commonly altered cancer genes, most notably TP53.
Recent advancements in sequencing technology have overcome previous scale limitations: hundreds of somatic mutations may now be discovered in a single tumour sample, allowing decipherment of mutational signatures even when many mutational processes are active.
Furthermore, because most mutations in cancer genomes are "passengers", they lack significant selection fingerprints.
All types of mutations (such as substitutions, indels, and rearrangements) as well as any additional mutation characteristics, such as the mutation's sequence context or the transcriptional strand on which it occurs, can be included in the collection of attributes that characterise a mutational signature.
Mutational signatures may be extracted using base substitutions and information on the sequence context of each mutation is also supplied.
There are 96 mutations in this categorization because there are six kinds of base substitution: C>A, C>G, C>T, T>A, T>C, T>G (It is chosen C>* and T>* because of the reverse complement that makes the other substitutions identical). This 96-synonym categorization is especially effective for differentiating mutational signatures that generate the same changes in various sequence contexts (Figure 1.5). Only one or two of the 96 potential replacement mutations are prominent in some signals, suggesting extraordinary specificity of mutation type and sequence context (for example signature 10). Others, on the other hand, have a roughly equal distribution of all 96 variations (for example signature 3).
There are signatures that are primarily defined by C>T mutations (signatures 1A/B, 6, 7, 11, 15, 19), C>A (4, 8, 18), T>C (5, 12, 16, 21), and T>G mutations (9, 17), as well

Figure 1.5. Examples of Single Base Substitution (SBS) Signatures [20]

as signatures that reveal distinct combinations of mutation classes (2, 13, 14). At least two mutational markers were found in most cancer classifications, with a maximum of six in malignancies of the liver, uterine, and stomach. Although some of these discrepancies might be explained by differences in signature extraction capacity, it appears that certain tumours have a more complex repertoire of mutational mechanisms than others.

Chapter 2

Homologous Recombination Deficiency

This chapter introduces the concept of *BRCAness*, which outlines the foundation of the study covered in the following chapters.
A general introduction of the *BRCAness phenotype* is presented first, then the analysis is limited to breast cancer and proceeds with a quantitative definition of its peculiarities.

For more detailed information regarding defects in the HR repair mechanism, you can refer to: [21], [22].

2.1 BRCAness phenotype

Cellular duplication is a process that happens in all living organisms and provides the foundation for biological heredity. This mechanism involves DNA replication, which, since it is so important for the preservation of genetic material, has a plethora of checkpoints to ensure its accuracy. Single-strand breaks (SSBs) are discontinuities in a strand of the double helix of DNA that can develop and, if not repaired properly, can represent a major danger to genetic stability and cell survival [23]. As a result, cells have evolved effective systems for their repair, which include Poly ADP-ribose polymerase (PARP), a family of proteins involved in DNA repair and apoptosis, whose primary function is to identify and signal SSBs to the enzymatic machinery involved in their repair (SSBR). The significance of this process is underscored by the fact that unrepaired SSBs can induce the blocking or collapse of DNA replication forks in growing cells, resulting in the creation of double-strand breaks (DSBs). Although cells have a remarkable ability to repair such DSBs utilizing homologous recombination (HR), acute increases in SSB levels in the cell might overwhelm this mechanism, resulting in genetic instability and/or cell death. Defects in the *homologous recombination* (HR) repair mechanism [24], caused by mutations in BRCA1/2 or other genes like RAD51 and PALB2, confer the so-called '*BRCAness*' phenotype in other tumour types including breast and ovarian malignancies [21].
Many chromosomal rearrangements are common in cancers caused by inactivating BRCA1

or BRCA2 mutations. BRCA1-mutated cancers, but not BRCA2-mutated cancers, have a high number of rearrangement signature 3 small tandem duplications [22].
Signature 5 deletions are seen in large numbers in cancers with BRCA1 or BRCA2 mutations. There were no additional rearrangement signatures linked to BRCA1- or BRCA2-null instances. In hierarchical clustering analysis, certain breast tumours without detectable BRCA1/2 mutations or BRCA1 promoter methylation displayed similar characteristics and clustered with BRCA1/2-null tumours.
Other mutant or promoter methylation genes may be exerting comparable effects in these circumstances, or the BRCA1/2 mutations may have been undetected.
The HR pathway, which includes BRCA1 and BRCA2, is required for high-fidelity DNA double strand break (DSB) [23] repair. Inactivation of such genes causes HR deficiency (HRD), which results in elevated levels of genomic abnormalities.
Base substitution, insertions/deletions, and rearrangement mutational signals may be more accurate indicators of poor homologous recombination-based DNA double-strand break repair and drug sensitivity.
HRD is a common feature of many malignancies, and it's more common in breast and ovarian cancers.
In the clinic, the presence of a germline BRCA1/2 mutation is now the most common genetic biomarker for HRD. However, there are some disadvantages to germline testing:

- It is reliant on the completeness and accuracy of clinical variant annotation databases;
- Epigenetic silencing is ignored;
- Current clinical genetic testing misses partial/complete deletions of the BRCA1/2 loci, resulting in BRCA1/2 status reporting based on the wild type allele from contaminating normal tissue;
- HRD can be caused solely by somatic events. Furthermore, the focus on BRCA1/2 ignores the inactivation of other genes in the HR pathway.

As a result, patients may get ineffective treatment or lose out on treatment options, necessitating the creation of improved HRD biomarkers.
Somatic passenger mutations, which can be discovered easily by whole-genome sequencing (WGS), have recently been proven to give insights into the mutational processes that happened before and during carcinogenesis, opening the way for new clinical tumour diagnostic prospects. HRD cancers rely on alternative, more error-prone mechanisms for DSB repair, such as *microhomology mediated end-joining* (MMEJ)[25]. MMEJ is an error-prone repair mechanism that involves alignment of micro-homologous sequences internal to the broken ends before joining, and is associated with deletions and insertions that mark the original break site, as well as chromosome translocations. It is frequently associated with chromosome abnormalities such as deletions, translocations, inversions and other complex rearrangements, resulting in a distinctive mutational footprint across the genome that may be utilised to diagnose HRD independent of the underlying aetiology (whether genetic or epigenetic).
The use of this method in primary tumours indicated that HRD is more common than BRCA1/2-deficient breast cancer tumours, and that it occurs at varied rates in various

cancer types. HRD rates in late metastatic cancer, on the other hand, are unknown, despite the fact that they are the individuals who are increasingly being targeted with customised therapy, such as PARP inhibitors for BRCA insufficiency.

2.2 BRCAness in breast cancer

Hereditary mutations in the BRCA1 and BRCA2 genes cause around 1-5 percent of breast cancers [21], which are also particularly susceptible to poly (ADP-ribose) polymerase (PARP) inhibitors. PARP is a nuclear enzyme (protein) that plays a role in a variety of cellular activities, the most important of which are DNA repair and programmed cell death (apoptosis). Poly(ADP-ribose) polymerases-1 and -2 (PARP-1 and PARP-2) are nuclear enzymes that use NAD+ as a substrate to create ADP-ribose polymers.

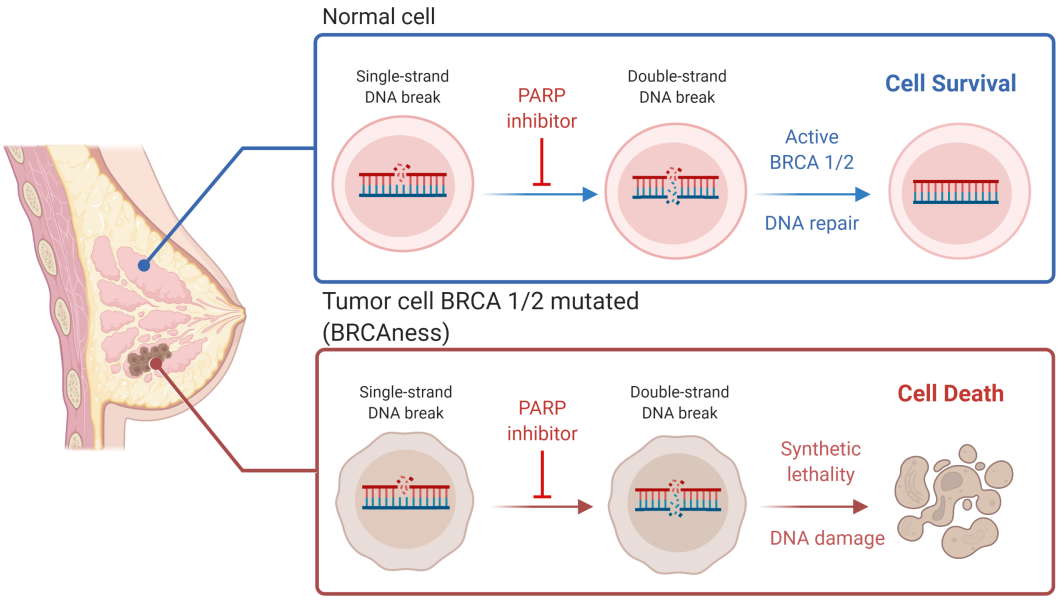

Figure 2.1. PARP inhibitors: Treatment for BRCA mutant Breast Cancer

PARP-1 and -2 are key components of Base Excision Repair (BER), which is responsible for repairing DNA damage caused by radiation and monofunctional alkylating chemicals. PARP-1 and -2 are involved in DNA damage repair, transcriptional activities, cell cycle and cell death control. PARP-1 is also involved in the generation of oxidant species that cause DNA damage during inflammation. It has recently been discovered to be directly implicated in the angiogenic process, implying that blocking both PARP-1 and PARP-2 might improve chemotherapy effectiveness and limit tumour growth via an anti-angiogenic mechanism.

PARP is involved in the repair of DNA damage caused by chemotherapeutic drugs (alkylating agents and topoisomerase I inhibitors): it is involved in the repair of alkylated bases.
The phenomenon of tumour resistance to treatment is caused by PARP activation. Inhibiting PARP, on the other hand, reduces cancers' capacity to tolerate alkylating drugs while increasing their susceptibility to chemotherapy.
Damaged single-stranded and double-stranded (DSB) DNA fragments accumulate in the nucleus when PARP is inactivated. The damaged DNA's DSBs go unrepaired, causing cell growth, cell division, and finally tumour cell death to halt.
Other kinds of cancer with germline and/or somatic mutations in BRCA1 and/or BRCA2 are likewise sensitive to PARP inhibitors. As a result, detecting cancers caused by BRCA1 or BRCA2 loss is critical.
Somatic substitutions, rearrangement patterns, insertions/deletions, and mutational signatures have all recently been linked to BRCA malfunction. BRCA1 and BRCA2 heterozygous germline mutations increase the lifetime risk of breast, ovarian, and other cancers. The BRCA1 and BRCA2 proteins have a variety of tasks, including maintaining genomic integrity by homologous recombination, which entails HR-mediated double-strand break (DSB) repair. The deletion of the wild-type allele causes these genes, which are known as tumour suppressors, to become fully dormant. BRCA1 and BRCA2 inactivation causes tumours to be deficient in HR, making them particularly vulnerable to chemicals that enhance the need for HR (Figure 2.1).
PARP inhibitors are an example of therapeutic medicines that induce the replication fork to halt and collapse, resulting in an increase in DSBs. Failure to execute HR-dependent DSB repair results in cancer cells being selectively killed. Preclinical and clinical investigations on breast and ovarian cancer have demonstrated the effectiveness of the PARP inhibitor in family individuals with BRCA1 and BRCA2 mutations.
Inhibition of PARP, on the other hand, has uses beyond the therapy of germline altered cancers. In high-grade serous ovarian cancer with germinal or somatic BRCA1 or BRCA2 mutations, effective PARP inhibition maintenance treatment has been reported. This has resulted in a focus on identifying the molecular characteristics of tumours deficient in BRCA1 or BRCA2 in cases where the genes are inactivated through germline, somatic, or secondary mechanisms, such as promoter DNA hypermethylation or inactivation of a related gene in the HR pathway.
As a result, gene-specific sequencing procedures (including sequencing of all known HR genes), copy number-based approaches (to calculate homologous recombination deficit index (HRD) and genomic "scars"), and functional HR competency tests have all been developed to identify BRCA1/BRCA2 deficits.
The findings acquired using the approaches previously outlined, on the other hand, have had minimal success in terms of prediction.
Recent studies have demonstrated the importance of a good predictor of the biological status of an HR-deficient tumour, because the set of tumours that show BRCAness and may be selectively sensitive to PARP inhibitors is not limited to the lower percentage of familial breast and ovarian cancers with BRCA1 or BRCA2 mutations, but also includes a wider range of sporadic breast and ovarian cancers, as well as other cancers of interest in this study, such as CRC.

Following recent technological advancements, sequencing costs have decreased significantly, allowing for the detection of all somatic mutations in human cancers, including base substitutions, insertions/deletions, copy number rearrangements, and aberrations, using whole genome sequencing (WGS).
An in-depth examination revealed mutation patterns, also known as somatic mutational signatures, which are the physiological readout of DNA damage and repair mechanisms during carcinogenesis. These markers indicate current and previous exposure to endogenous metabolic degradation, environmental stressors including UV light, and deficits in DNA repair programs like HR.
As a result, mutational signatures showing BRCA1/BRCA2 deficiency in germline mutated cancers can be utilised to predict BRCA1/BRCA2 deficiency in other malignancies that exhibit it. Signature 3 was first discovered in a small subset of breast cancers, defined by a base substitution that allowed researchers to separate germline tumours with BRCA1/BRCA2-null alleles from sporadic tumours; the study was then expanded to include malignancies of the pancreas, ovary, and stomach.
Despite this, it is unable to determine an effective threshold for distinguishing BRCA deficient from BRCA proficient malignancies using a single mutational signature.
Whole genome sequencing has just been more widely available, which has brought fresh insight into the differentiation of these malignancies.
A single mutation in a single gene, such as BRCA1 or BRCA2, results in at least five mutational signatures of various kinds, including base substitutions, indels, and rearrangements.
These numerous mutational fingerprints differ from other biomarkers in that they are the direct result of the DSB not being repaired. As a result, this finding will be used in the subsequent analyses to quantitatively describe the genomic properties of BRCA1/BRCA2 inactivation, offering a whole genome sequencing-based predictor for the diagnosis of HR-deficient cancers.

2.3 Quantitatively define HR Deficiency

It was feasible to detect twenty-four samples containing genetic predisposition mutations in BRCA1 (n = 5) and BRCA2 (n = 19) using Nik-Zainal's work [21], which entailed the sequencing of the breast cancer genomes of 560 patients.
The wild-type allele was lost in 22 of the latter, which was predicted to result in total inactivation of the relevant protein. The overrepresentation of base substitution signatures 3 or 8, the rearrangement of signature 5, and a substantial number of deletions bigger than three nucleotide bases, characterise the genomic profile of the 22 people studied.
Furthermore, BRCA1-null tumours displayed an excess of rearrangement signature 3 mutations and a slight contribution of rearrangement signature 1 mutations.
The 22 BRCA1- tumours were compared to 235 additional cases of sporadic breast cancer with quiescent genomic profiles, other than those of BRCA1/BRCA2-null tumours, in order to quantify the features of the BRCA1/BRCA2 deficiency.
The study included the extraction of twelve basic substitution mutational signatures, two

indel, and six rearrangement, which allowed the HRD copy number indices to be calculated, as well as the use of a logistic regression model to count the mutational signatures and HRD indices (log transformed and normalised to allow comparability between genomic parameters).
An iterative validation technique was utilised to guarantee that the parameters identified as predictors of BRCA1/BRCA2 deficiency were robust and generalizable.
Ninety percent of the samples were used for the selection of model parameters. On the remaining 10% of data, the weights for each parameter were examined. Finally, five unique characteristics were determined to transmit the biggest difference between BRCA1/BRCA2-deficient cancers and sporadic breast tumours: microhomology-mediated indel, HRD index, base substitution signature 3, rearrangement signature 3, and rearrangement signature 5.
The model's efficacy in predicting BRCA1/BRCA2 deficiency, as well as detecting any additional cancers with comparable features to germline BRCA1/BRCA2-null tumours, was evaluated using the specified parameters on a dataset of 560 breast tumours.
An additional 90 tumour samples with a likelihood of BRCA1/BRCA2 deficit greater than 0.7 were detected as a result of extending the analysis to the initial dataset, contributing to a 20% rise in the overall proportion of patients with a high level of BRCA1/BRCA2.
This finding prompted the researchers to look for further germline and/or somatic mutations in BRCA1 and BRCA2 in the dataset.
Thirty-three individuals were found to have pathogenic germline mutations in BRCA1 or BRCA2, as well as somatic inactivation of the second allele.
This has significant clinical and potential genetic counselling implications for those affected and their families in terms of therapeutic choices, as it involves a significant increase in the number of individuals harbouring alleles of familial predisposition to cancer compared to the known number at the start of the study.

Chapter 3

Predicting BRCA1/BRCA2 deficiency in breast cancer

In this chapter, the aim is to illustrate a fundamental tool that this study uses in order to predict the BRCAness phenotype: HRDetect.
It is intended to focus primarily on the application and performance of the tool, both relative to breast and other cancer samples.

For information regarding the processing of HRDetect, reference can be made to the works of the Nik-Zainal group [22], [21].

3.1 A HR status classifier

With a larger starting data set, enriched with 77 samples (22 with known germline mutations, 33 with new germline mutations and 22 with somatic mutations), the same genomic characteristics previously identified as predictive parameters for the 22 null germline samples (with the addition of signature 8) were identified.
Base substitution signatures 3 and 8, rearrangement signatures 3 and 5, microhomology-mediated deletions, and the HRD index are among these traits [22].
The availability of a larger data set has allowed to confirm the stability of the critical parameters chosen for distinguishing between BRCA1 and BRCA2-deficient tumours, as well as identify interactions between genomic covariates, which could be potential causes of the amplification of cooperating signature effects.
Although correlations were discovered, the performance of the model with interactions did not improve on the predictions given by the model without interactions.
As a result, Nik-Zainal chose to keep a basic model with independent genomic parameters.
In this context, the HRDetect predictor of BRCA1/BRCA2 deficiency was targeted to that group [21].

HRDetect's performance was demonstrated to be superior to current techniques of diagnosing BRCA1 and BRCA2 insufficiency, based on the use of distinct mutational signatures.
HRDetect correctly identified BRCA1/BRCA2 deficit in 560 people using a probabilistic cutoff of 0.7, with a sensitivity of 98.7%, detecting 124 samples with a score greater than 0.7, including additional 47 samples with a high chance of BRCA1/BRCA2 deficiency. It has been shown effective in alternate sequencing methodologies and has been verified on separate cohorts of breast, ovarian, and pancreatic cancers.
When all of the classes of mutational signatures are combined, a higher proportion of people with breast cancer who have BRCA1/BRCA2 deficiency (up to 22%) than previously thought (1–5%) may exhibit preferential therapeutic sensitivity to PARP inhibition.

3.2 Other genetic factors related to BRCAness

Three samples exhibited mutations in the HR genes, despite having high HRDetect scores and no biallelic BRCA1 or BRCA2 mutations [21]. One in particular had a high HRDetect score and a profile that was normally linked with BRCA2 nullity, whereas the other paternal allele was preserved despite having a BRCA2 germline mutation.
As a result, this patient was an outlier in which the genetic nullity of BRCA2 in the tumour could not be proven, making it impossible to rule out the inactivation of the wild-type allele by other mechanisms.
Inactivating monoallelic somatic mutations in other HR repair genes, such as *ATR* and *ATM*, were not linked to increased BRCA1/BRCA2 deficient scores, which was surprising. Furthermore, among cancers with high HRDetect scores, no gene from the HR gene list could be identified as a contributor.
Notably, susceptibility alleles for high and moderate penetrance germinal breast cancer, such as *TP53*, *PTEN*, *ATM*, *CHEK2*, *ATR*, *RAD50*, *CDH1*, *STK11*, and *PALB2*, were not linked to a genomic profile or a high likelihood of BRCA1/BRCA2 deficiency.
This demonstrates not only the importance of knowing the status of the alternative parental allele when interpreting mutation data, but also that nearly a third of tumours with high HRDetect scores that predict deficiency are not states associated with either a genome profile or a high probability of BRCA1/BRCA2 deficiency and thus cannot be authenticated as null BRCA1 or BRCA2 through genetic and/or epigenetic means.
On the other hand, given the tumours' resemblance to null BRCA1 and BRCA2 cancers, it's worth considering the biological similarities between them, as well as the fact that they're likely to respond to PARP inhibition in a similar way.

3.3 Application of HRDetect in other cancer types

HRDetect was applied to genome-wide sequencing of different malignancies, including pancreatic and ovarian cancer, to see if it may be used for other tumour types[21]. This was accomplished by analysing the available BAM data, extracting mutation signatures, and obtaining copy number profiles utilising the somatic mutation invocation process.
Despite the distributions of HRDetect scores being significantly different from those

obtained for breast cancer samples, HRDetect exhibited good sensitivity in detecting BRCA1/BRCA2-null tumours for this kind of tumours (around 100 percent).

3.4 Robustness, stability and generalisability

A 10-fold nested cross-validation technique was used to assess the robustness and generalizability of the learnt weights, which included the usage of 10 external folds and the availability of 10% of the data [21]. The remaining 90% of the data is instead utilised to pick model parameters related to BRCA1 and BRCA2 deficiencies, which are examined on the inner folds for a range of values that determine the sparsity of the results.
The model coefficients are calculated for each of the 10 folds, revealing that the findings are consistently non-zero for each of the genomic parameters identified as different.
Finally, the stability of each coefficient was determined by randomly selecting half of the samples in the training set and counting the number of times each genetic trait was identified as different (i.e. it had a coefficient other than zero). Each coefficient was non-zero after iterating over the subsampling for 100 iterations.

Chapter 4

How is the HRDetect score predicted?

In this chapter I proceed to retrace the workflow, previously carried out by the research group of Nik-Zainal, which allowed me to obtain as a result the HR score of interest for the study.
On the one hand, I focus the attention on the use of two different pipelines. On the other hand, I proceed with the validation of the results obtained by the group from the original data and from the data remapped with a more recent version of the reference genome.

To obtain complete information on this topic, you can refer to:[22], [21]

4.1 From WGS analysis to the score

Once the data set, consisting of samples of individuals with breast cancer, used by Nik-Zainal's group in their study, was obtained, one of the first objectives I wanted to pursue was to validate the results of this study.
This was possible by retracing the workflow previously carried out by the research group and submitting to it a fraction of the samples they identified. As can be seen in the Figure 4.1, the workflow used, starting from the original alignment file, involves the use of a first pipeline called *cpgwgs*, the results of which, after the application of appropriate filters, are subjected to a second pipeline, the *HRDetect* one [21], through which it is finally possible to have access to the HRDetect score (final result of the workflow).
The pipeline used by Nik-Zainal's group, indicated as *cgpwgs* pipeline (Cancer Genome Project Whole Genome Sequencing) or as *Sanger* pipeline (as the CG project is by Sanger), requires two alignment files (in BAM format) as input: one related to the tumour tissue of the patient under examination and one related to the corresponding normal tissue.
The comparison between the two tissues is essential for the identification of somatic mutations, which are of interest in this case.
Specifically in fact, as already mentioned above, the germline mutation is a mutation that occurs in germ cells or sex cells or in eggs and spermatozoa. Since this mutation is present

in gametes, it is passed on to the next offspring and also, every cell in the entire organism affects this germline mutation.
The rest of the cells other than the germline or sex cells are the somatic cells of an organism.
Somatic mutation is the mutation that occurs in a single cell of the body. Thus, this type of mutation is localised only in the tissue derived from the mutated cell and therefore does not affect every cell in the organism, unlike the germline mutation.

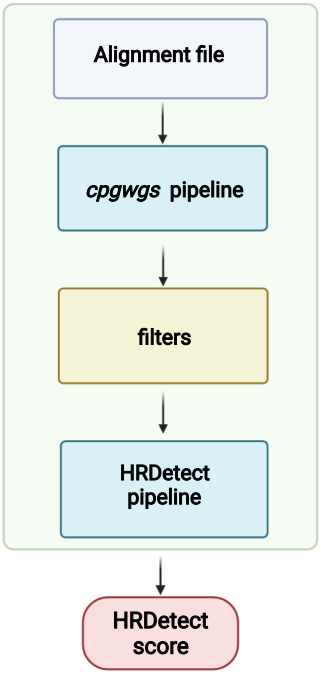

Figure 4.1. Workflow of the Nik-Zainal's research group

This aspect is of significant importance, since to run the workflow it is necessary to be in possession of both samples so that the pipeline can make the comparison between them.
In view of this "limit" of the pipeline supplied by the Sanger group, an important objective will be outlined relating to the identification of a "surrogate" of the normal tissue sample when this is not available due to lack of raw data.
This also opens a window towards a possible future study which provides for a refinement of the workflow so that it is no longer essential to arrange the healthy tissue related to the associated tumour sample under examination.
I also want to underline that, following the same steps taken by the research group, the BAM files used as input are the result of the alignment to the reference human genome (in the hg19 version) using Burrows – Wheeler aligner, BWA (v0.5.9).
Running the *cgpwgs* pipeline results in a series of files divided by variant types, collected in the same folder. Specifically, the output consists of four files relating to the types of variants corresponding to the results provided by four algorithms [22]: *CaVEMan* (Cancer

Variants Through Expectation Maximisation [26]) was used to call single nucleotide variations (SNV); Indels in the tumour and normal genomes were called using modified *Pindel* version 2.0 [27]; Structural variants (SV) were discovered using the *BRASS* (BReakpoint AnalySiS) algorithm [28] through discordantly mapping paired-end reads followed by de novo local assembly using *Velvet* to determine exact coordinates and features of breakpoint junction sequence; allele-specific copy number analysis of tumours was performed using *ASCAT* (v2.1.1) to generate integral allele-specific copy number profiles for the tumour cells.
In particular, the files relating to SNV, SV and Indel are presented in the VCF format described above (generally in the compressed form "file.vcf.gz"), while the file containing the information relating to the change in the copy number is in tabular form.
At this point of the workflow, the files obtained as output are subjected to the application of filters, taking on the characteristics required at the input of the HRDetect pipeline.
I also want to underline that the two pipelines, just described briefly in their operation, are independent from each other, that is, they do not "communicate" with each other directly; for this purpose "intermediary" filters have been introduced between the pipelines, in order to calculate the HRDetect score starting from the original raw data.

4.2 Validation on original mapped data

Following the procedure described above, it was possible to make a comparison between the scores obtained by Nik-Zainal's group[21], to which from this moment on I will refer to the "expected" values, and those obtained by retracing the workflow they described by means of the same data provided in input.
Through a comparison that provides for a focus on 30 of the patient samples examined by the research group, it is clear that almost all of the results confirm those predicted by the study.

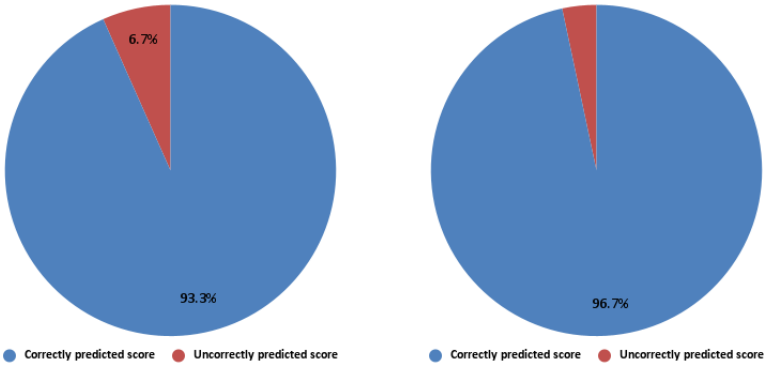

Figure 4.2. HRDetect score validation on original mapped data (on the left) and on remapped data

In particular, 28 out of 30 of the compared samples show a score coinciding with the

expected one, making an incorrect estimate in only 6,7% of cases (Figure 4.2).
I want to underline the fact that the values of the scores obtained are not co-identifying from a mathematical/numerical point of view, but in terms of classification as HR proficient/deficient.
It is also assumed that for the two cases in which the score obtained differs from the expected one, the cause of the failure is linked to possible variations in the input files compared to those used in the actual by the Nik-Zainal group.

4.3 HRDetect score validation on raw reads data

Once the raw data of the HRDetect paper have been used to try to reproduce the same scores published by the Nik-Zainal group [21], obtained by mapping on an old version of the reference genome (hg19) using the algorithm BWA aln (*Burrows - Wheeler Aligner*, v0.5.9), it was tried to proceed forward by remapping the same data differently (Figure 4.3). In particular, it was decided to use a more recent version of the reference genome (currently the most advanced is indicated as hg38) used routinely, as well as to carry out the remapping by means of a different mapper (BWA mem), which is the most recent algorithm and currently in common use, recreating the situation in which the research group of the Candiolo IRCCS generally works.

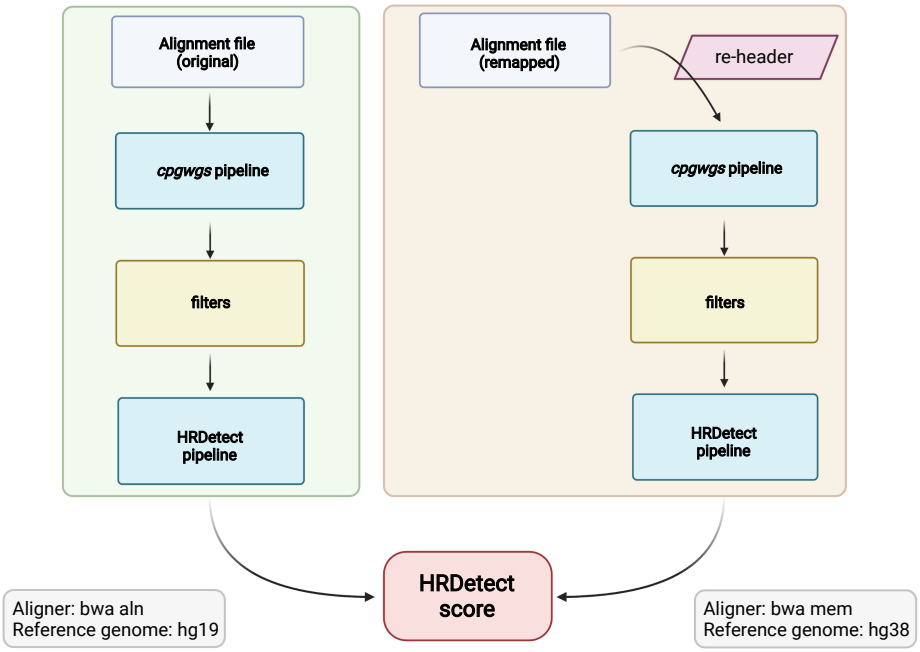

Figure 4.3. Comparison of the two pipelines used for the validation on original mapped data and on remapped data

Specifically, to carry out the remapping, the alignment files in the BAM format (data

originally made available by the Nik-Zainal group) were initially converted into the FASTQ format: from the BAM files it was possible to extract the sequence data; starting from the FASTQ format, the reads were extracted and it was possible to complete the remapping by reporting the data in the BAM format, using the most recent mapper and the reference genome in the hg38 version.
The next step instead involves the reinsertion of the header section, in order to add information necessary for the operation of the Sanger pipeline that was present in the original BAM files but which are not present in the files obtained with the remapping.
Once these first steps have been completed, the BAMs obtained, containing information similar to the initial ones, can be used as input for the previously described workflow, which involves the execution of the cpgwgs pipeline, followed by filtering the data and obtaining the HR score by means of the HRDetect pipeline.
Following the procedure described above, it was possible to make again a comparison between the scores obtained by Nik-Zainal's group, indicated as expected values, and those obtained by retracing the workflow they described using the remapping results as input.
Through a comparison that provides for a focus on the same 30 samples as the patients examined previously, it is clear also in this case that almost all of the results confirm those predicted by the study.
In particular, 29 out of 30 of the compared samples show a score coinciding with the expected one, incorrectly estimating only with a probability of 3,3% as shown in Table 4.2 (lower than the previous case).
As before, I want to underline the fact that the values of the scores obtained are not co-identifying from a mathematical/numerical point of view, but in terms of classification as HR proficient/deficient.

Chapter 5

Prediction on patients without matched normal tissue

This chapter is intended to introduce the main objective of the thesis work through the development of different strategies and of a new sample called *meta*normal.
Specifically, all the strategies adopted, in order to emulate the HR scores predicted by the Nik-Zainal's research group with the use of the new sample introduced, are retraced, allowing the overcoming of the limitations imposed by the pipelines by construction.
The dataset is described and each strategy is examined in detail, outlining the insights behind the development as well as the code implemented for the realisation.
Finally, the results obtained are proposed and compared with the expected scores.

To obtain complete information it is possible to make reference to [21], [29], [22].

5.1 Overcome the HRDetect limitations

As already mentioned, the pipeline used by Nik-Zainal's group [21], indicated as *cgpwgs* pipeline, requires two alignment files (in BAM format) as input: one relating to the tumour tissue of the patient under examination and one relating to the corresponding normal tissue.
Specifically, I want to emphasise that the two alignment files, related to the two samples, are necessary for the operation of the *cgpwgs* pipeline just mentioned and it is not possible to use it otherwise.
In particular, for this type of analysis the match between normal and tumour sample is necessary in order to be able to make the comparison (Figure 5.1) and extract only the somatic variations due to the tumour, discarding the healthy variations of the patient, i.e. the germinal ones.
Consequently, the next step was to identify a way in which it was possible to make the prediction of HR status without the patient's normal match, for example to be able to make a prediction for cell lines for which you do not have the normal sample.
This need identifies the main purpose of this thesis work.

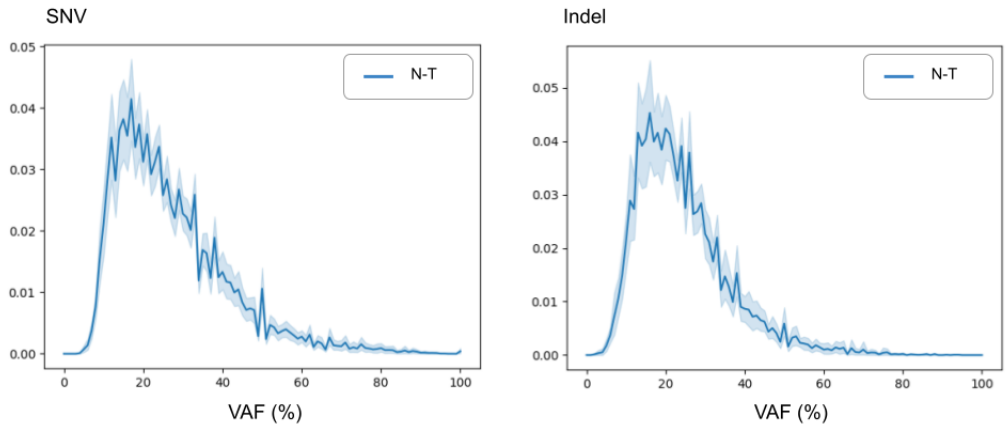

Figure 5.1. Normalized Variant Allele Frequency (VAF) distribution for N-T comparisons for SNV (on the left) and Indel (on the right) variations

To implement this, a new sample was constructed, taking the name of *meta*normal: a normal sample resulting from the merge of a good number of sequences of normal tissues, which presents characteristics such as to be able to be used as input of the workflow previously described.
The construction of the metanormal requires the use of data from normal tissue samples, as these intrinsically contain a greater amount of information (describing frequency curves of bell-shaped variations due to the diploidy of the sample itself and presenting variable depth of sequencing). On the other hand, one could also think of sampling the reference genome by extracting "dummy" reads, but in doing so one would lose all the information due to the SNPs, leading to a "flat" frequency distribution (due to the haploidy of the reference) and furthermore the depth of sequencing would be constant.
Specifically, in order to maintain the average read depth [30] (that describes the number of reads that cover/map each single base) comparable to that of the normal samples originally used, the reads were randomly sampled for each of the normal patient samples taken into consideration; the merge of the reads taken from the random sampling finally allowed to obtain the metanormal sample.
At this point, in order to highlight the characteristics that could distinguish the N-T distribution from the MTN-T distribution, I chose to focus on the frequency distributions of single nucleotide variations (SNV) and on insertions/deletions (Indel), as they are among the most important types of variation in order to determine the HR score and thus constitute a summary of the occurrence of variations.
By graphically representing these distributions for each metanormal-tumour comparison (Figure 5.2), relative to each breast cancer sample involved in the study (83 in this case),

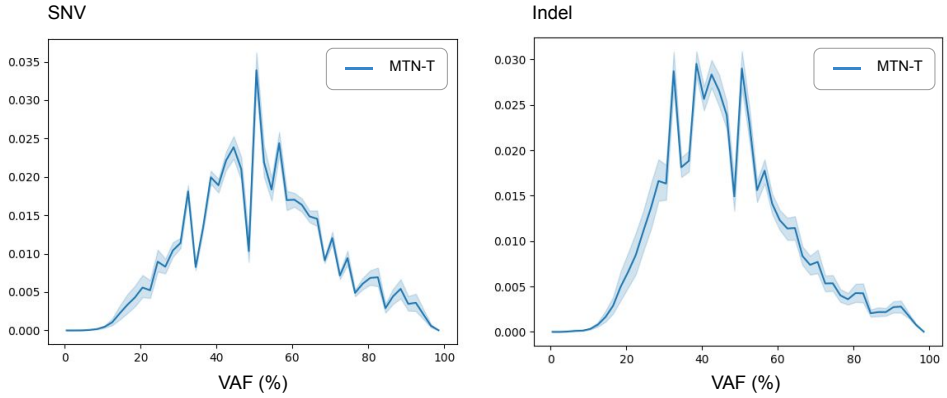

Figure 5.2. Normalized Variant Allele Frequency (VAF) distribution for MTN-T comparisons for SNV (on the left) and Indel (on the right) variations

it is possible to observe a typical bell-shaped trend identified by the median of the distributions, around which the different cases considered are distributed.

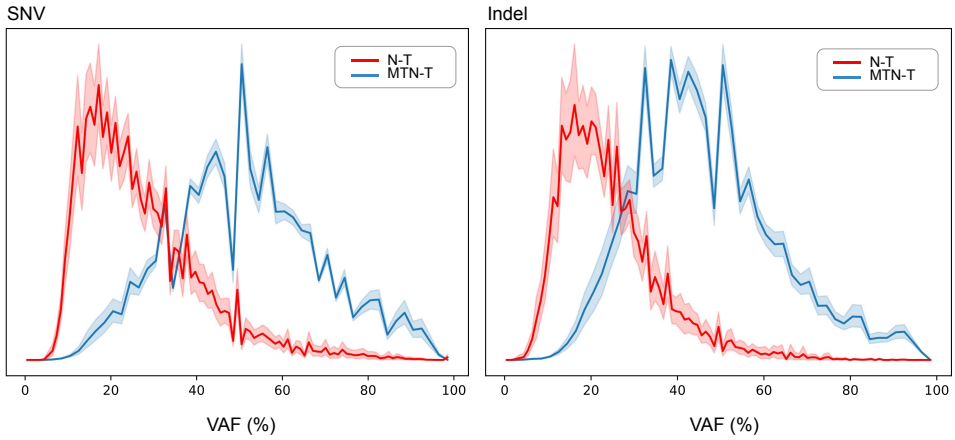

Figure 5.3. Normalized Variant Allele Frequency (VAF) distribution for MTN-T (blue) and N-T (red) comparisons for SNV (on the left) and Indel (on the right) variations

Thus delineating the bell distribution, as well as its confidence interval, it is possible to

ascertain that the curve is spiked around 50% as expected, since it is dominated by germinal variations for diploid samples.
It is necessary to underline in particular that the comparison in this case is made between the tumour samples and the metanormal constructed as described above. It follows that, since the tumour is not "matched" with the respective normal (relating to the patient under examination), series of germline mutations are not properly filtered as such, so they appear in the set of extracted somatic variations.
On the other hand, representing the distributions of the SNV and Indel frequencies for each of the comparisons between the tumour samples and the corresponding normal samples, the median trend outlined is again bell-shaped (Figure 5.3), around which the confidence intervals delineated by the individual curves for each patient examined can be observed.
Unlike the previous case, however, the peak of the distribution is decentralised and shifted towards lower frequencies (around 20%), recounting the tumour content present in the cells.
In particular, it is emphasised that in this case the focus is on the comparison between matched samples of normal and tumour tissue, which entails the removal of the germinal component from the tumour, thus obtaining a distribution relative to somatic variations (Figure 5.4).

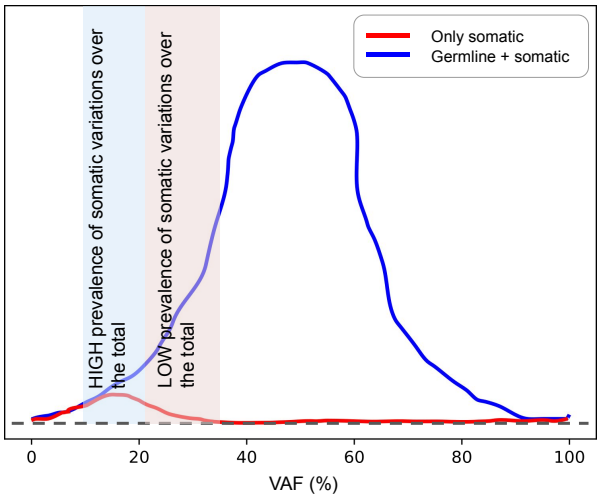

Figure 5.4. Variant Allele Frequency (VAF) distribution (not normalized) for somatic and germline mutations

5.2 Dataset description

DNA was extracted from 560 breast cancers and normal tissue (peripheral blood lymphocytes, adjacent normal breast tissue or skin) by Nik-Zainal's group[21]. The latter specified that no statistical methods were used to predetermine the sample size. For this

study specifically, 77 samples were downloaded from the total 560 delineated by the group during their research [Table 5.1]. These correspond to the only samples made available by the group that are BAM-formatted and therefore the only ones that can be submitted to the pipeline described above.
As can be seen from the pie chart (Figure 5.5), the data set is reasonably balanced: referring to the expected data, that is to the values of the scores obtained by the Nik-Zainal group, 54.5% of the samples taken into consideration are HR proficient, 39% of the samples is HR deficient, while the remaining 6.5% relates to samples identified as doubtful, not classifiable as HRP or HRD.

Patient	Expected Score	Patient	Expected Score	Patient	Expected Score
PD22036a	0.000	PD23579a	0.0000	PD24219a	NA
PD22251a	0.003	PD24182a	0.9996	PD24220a	0.0012
PD22355a	0.9999	PD24186a	0.7788	PD24221a	0.0078
PD22358a	0.9170	PD24189a	0.0000	PD24223a	0.0013
PD22359a	0.0018	PD24190a	0.0141	PD24224a	0.0006
PD22360a	0.9748	PD24191a	0.8996	PD24225a	0.0013
PD22361a	0.0004	PD24193a	0.0000	PD24302a	0.0002
PD22362a	0.0022	PD24195a	0.0009	PD24303a	0.8697
PD22363a	0.9723	PD24196a	NA	PD24304a	0.8740
PD22364a	0.0035	PD24197a	0.9978	PD24306a	0.9660
PD22365a	0.0062	PD24199a	0.0096	PD24307a	0.0006
PD22366a	0.9995	PD24200a	0.0272	PD24308a	NA
PD23550a	0.0047	PD24201a	0.9969	PD24314a	0.0040
PD23554a	0.9964	PD24202a	0.9997	PD24318a	0.0216
PD23558a	0.9946	PD24204a	0.0001	PD24320a	0.0002
PD23559a	0.1954	PD24205a	0.9970	PD24322a	0.0009
PD23561a	0.0000	PD24206a	0.0007	PD24325a	0.1706
PD23562a	0.9998	PD24207a	0.0297	PD24326a	0.0008
PD23563a	0.9993	PD24208a	0.0186	PD24327a	0.0010
PD23564a	0.0000	PD24209a	0.0692	PD24329a	0.0002
PD23565a	0.0351	PD24212a	0.9987	PD24332a	0.0021
PD23566a	0.9999	PD24214a	0.0030	PD24333a	0.0023
PD23567a	0.9983	PD24215a	NA	PD24335a	0.0083
PD23570a	0.0071	PD24216a	0.0055	PD24336a	0.0006
PD23577a	0.9999	PD24217a	0.0004	PD24337a	1.0000
PD23578a	0.9941	PD24218a	0.0047		

Table 5.1. Dataset Description

Breast cancer whole-genome sequence BAM files are available from the European Genome-phenome Archive (EGA)[22]. EGAS00001001178 is the EGA accession number for the

560 breast tumors used in the early development of HRDetect. This includes both whole-genome sequence BAM files and SNP6 array CEL files. The accession number for the 80 additional breast cancers used for validation is EGAD00001002740 (sequence BAM files).

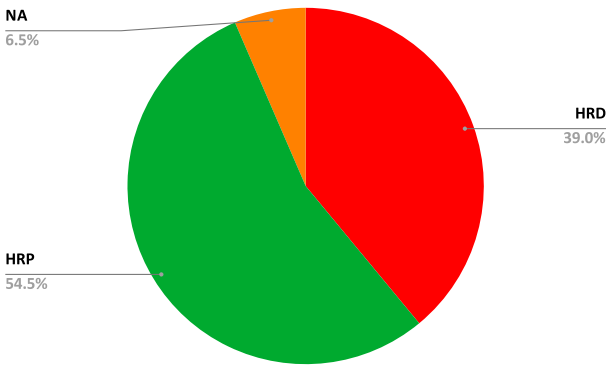

Figure 5.5. Dataset balancing

5.3 Strategy A

The previous observations, relating to the differences that emerge between the distributions obtained by considering the comparison between the tumour sample and the respective normal sample (N-T) and the comparison between the tumour sample and the metanormal sample (MTN-T), led to the development of a strategy through which it was possible, starting from data obtained with the metanormal, emulate the distribution obtained with the matched normal sample.
This strategy, which I will refer to as Strategy A, has the ultimate goal of correctly predicting the HR score starting from the comparison between the tumour samples and the constructed metanormal.
This strategy directly considers only what concerns the slice of allelic frequencies containing the greatest content of somatic mutations. In particular, in fact, from the point of view of statistical information, it is possible to realise that, when the normal matched sample (which would subtract all the germinal part leaving the somatic part) is not available, in the mix between germinal and somatic (dirty data obtained with the MTN-T comparison), most of the somatic information of interest is in the region where the N-T distribution peaks.
In this case, it is sufficient to consider the portion of variations that reside between the frequencies of 10 and 20% (range in which the distribution of the N-T comparison is spiked), both as regards single nucleotide variations and insertions/deletions (Figure 5.6).
Thus, the concept of Strategy A consists in removing everything that resides before 10% and after 20%.

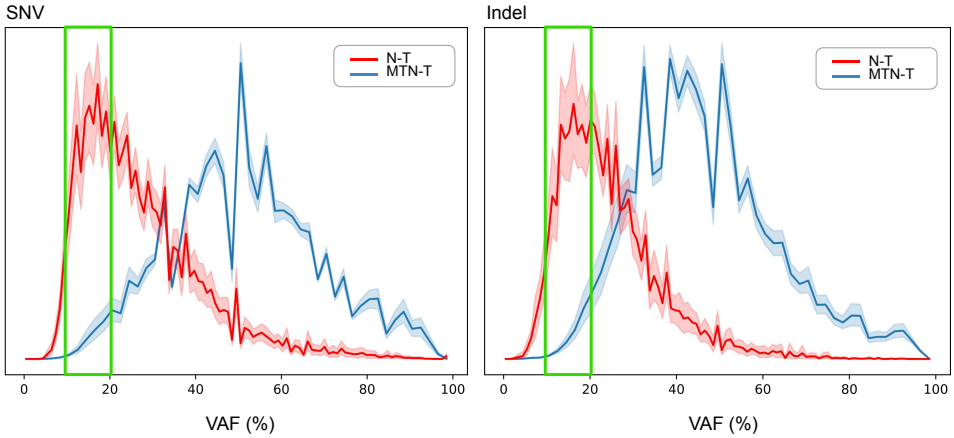

Figure 5.6. Selection of a specific range of frequency (10-20%) from the normalized Variant Allele Frequency (VAF) distributions for MTN-T (blue) and N-T (red) comparisons for SNV (on the left) and Indel (on the right) variations

5.3.1 Algorithms

The writing of the different scripts presented has been carried out mainly through the use of the Linux shell.
Specifically, through the Bash environment (acronym for Bourne Again SHell), which represents a textual shell of the GNU project used in Unix and Unix-like operating systems, such as GNU/Linux.
Bash is a command interpreter that allows the user to interface with its operating system using a set of preset functions, as well as run applications and scripts.
It can execute instructions that are supplied to it, and it can use input and output redirection to cascade many programs in a software pipeline, passing the previous command's result as input to the next command.
It also comes with a simple native scripting language that allows you to accomplish more complicated operations by using variables, functions, and flow control structures in addition to collecting a sequence of instructions in a script.
These features determined the choice of the Bash shell for the realisation of the algorithms of this thesis work.
On the other hand, the dynamism, simplicity and flexibility of the Python language meant that this programming language was used to carry out certain operations within the algorithms that outline the different strategies, in order to improve their readability and comprehension.
In addition, the use of the interpreted programming language "AWK" allowed the simple manipulation of textual data, both in the form of files and data streams from standard input.

Get_indel_from_freq-distr.py

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that the reference distribution file has been provided followed by the VCF file relating to the Indel variations, saved respectively as REF_fd and INDEL_fd. The read_ref_distribution function is defined.
It requires that a reference distribution file (corresponding to the distribution values of the red curve in the Figure 5.6, indicated as a reference) is passed as a parameter in the format as the GSL-histogram output.
GSL-histogram is a program for the GNU Scientific Library. It takes three arguments, gsl-histogram $xmin$ $xmax$ $[n]$, specifying the upper and lower bounds of the histogram and the number of bins. It then reads numbers from "stdin" (standard input), one line at a time, and adds them to the histogram.
When there is no more data to read it prints out the accumulated histogram, providing as output a file containing line by line at the far left of the interval, far right of the interval, and the number of variations in that specific interval.
The function read_ref_distribution, starting from the reference distribution file in the format as the GSL-histogram output, returns an ordered nested list containing, for each index, the three decimal numbers corresponding respectively to the start, end and number of variations of the interval.
By calling the previously defined function read_ref_distribution, the Indel frequency distribution, initially saved in the "distr" list, is copied into the "distr_dict" dictionary, using the "start end" pair (corresponding to the start and end of interval).
Then the function get_FORMAT_ID is defined.
As previously introduced, the VCF format includes a header, whose lines are identified by the "##" symbols and a body which lists all the variants.
In particular, one of the 9 mandatory columns identified by the "#" symbol is represented by the FORMAT, i.e. the list of annotations relating to the relationship of each variant with each sample, therefore concerning the genotype.
Furthermore, there are a number of columns equal to the number of samples, in which the value of the annotations present in the FORMAT column is reported.
The function get_FORMAT_ID, given a row of the VCF file and the identification name of the specific field of the FORMAT (corresponding to ID), has the aim of extracting the value contained in the column relating to the tumour sample in the position corresponding to ID in the FORMAT. In particular, if the VCF line considered were the following:

```
CHROM  POS     ID          REF    ALT     QUAL  FILTER    INFO
20     14370   rs6054257   G      A       29    PASS      NS=3;DP=14;AF=0.5;
FORMAT       NA00001          NA00002          NA00003
GT:GQ:DP:HQ  0|0:48:1:51,51   1|0:48:8:51,51   1/1:43:5
```

calling the get_FORMAT_ID (VCF_row, "GQ") function would return "48" for the

$NA00002$ sample.
At this point, once the dictionary keys have been saved in "bins", sorted by coordinates identified by the extremes of the intervals considered, I want to assign the Indel variations to each interval.
In particular, a dictionary called "Indels" is defined, in which each bin is associated with a certain number of lines of the VCF file for which the variant allele frequency (VAF), obtained by the parse_VCF function, falls within the defined range from bin.
In particular, to extract the frequencies of Indel variations starting from a VCF format file relating to the Indel variations, it is necessary to consider two specific fields of the FORMAT column, described in the header of the file as follows:

```
## FORMAT = <ID = FD, Number = 1, Type = Integer,
Description = "Fragment depth">
## FORMAT = <ID = FC, Number = 1, Type = Integer,
Description = "Fragment calls">
```

The first step therefore involves calling the get_FORMAT_ID function, in order to extract from the tumour sample the values corresponding to the positions identified by "FC" and "FD" in the FORMAT. Subsequently, the percentage frequencies of Indel variations are obtained through the ratio between the two identified values taken in the order described, multiplied by one hundred. The parse_VCF function cited above, starting from a VCF file relating to the Indel variations, has the aim of returning the value of the variant allele frequency, followed by the corresponding entire line of the VCF file. The crucial aspect of the Get_indel_from_freq-distr.py algorithm consists in the realisation of the sampling of the Indel variations through the definition of the so-called "*support-unit*". This process in principle provides for the calculation of ratios as the ratio between the number of lines of the VCF associated with a given bin (information obtained by calculating the length of the Indels[bin] list) and the frequency calculated for that specific bin (corresponding to the third value contained in the distr list for each index). All reports will then be saved in the "ratios" list.
On the other hand, in the "non_zero_ratios" list, all the elements of the "ratios" list other than zero will be saved. At this point the "support-unit" variable takes on the minimum value among those identified in the "non_zero_ratios" list, unless this is empty, in which case the "support-unit" is set equal to zero.
For each bin, the product between the "support-unit" and the number of variations identified for that interval (saved in distr_dict [bin]) is subsequently calculated, stored in the variable N. If N is lower than the number of rows of the VCF associated with the bin considered (N <len(Indels[bin])), then N is randomly selected from the lines of the VCF associated with the bin considered and are saved in "sampled_indels"; otherwise (if N>=len(Indels [bin])) all lines of the VCF associated with the considered bin are saved in "sampled_indels".
Finally, all the contents of "sampled_indels" are printed.

Get_SNV_from_freq-distr.py

As seen for Get_indel_from_freq-distr.py, initially it is verified that the script has been

launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that the reference distribution file is provided, followed by the VCF file relating to the SNV variations, saved respectively as REF_fd and SNV_fd.
The two functions read_ref_distribution and get_FORMAT_ID are defined.
In particular, these are analogous to the functions of the same name described for the Get_indel_from_freq-distr.py script, therefore: the function get_FORMAT_ID , given a row of the VCF file and the identification name of the specific field of the FORMAT (corresponding to ID), has the aim of extracting the value contained in the column relating to the tumour sample in the position corresponding to ID in the FORMAT; the function read_ref_distribution , starting from the reference distribution file in the format as the GSL-histogram output, returns an ordered nested list containing, for each index, the three decimal numbers corresponding respectively to the start, end and number of variations of the interval.
By calling the previously defined function read_ref_distribution , the SNV frequency distribution, initially saved in the "distr" list, is copied into the "distr_dict" dictionary, using the "start end" pair (corresponding to the start and end of interval).
Differently from what was observed for the Get_indel_from_freq-distr.py script, in this case to extract the frequencies of SNV variations, starting from a VCF format file relating to the SNV variations, it is necessary to consider only one specific field of the FORMAT column, described in the header of the file as follows:

```
##FORMAT= <ID= PM, Number= 1, Type= Float,
Description= "Proportion of mutated allele">
```

The first step therefore involves calling the get_FORMAT_ID function, in order to extract from the tumour sample the values corresponding to the position identified by "PM" in the FORMAT.
Subsequently, the percentage values of frequencies of SNV variations are obtained by multiplying the identified value by one hundred.
The parse_VCF function, starting from a VCF file relating to the SNV variations, has the aim of returning the value of the variant allele frequency, followed by the corresponding entire line of the VCF file.
At this point, once the dictionary keys have been saved in "bins", sorted by coordinates identified by the extremes of the intervals considered, I want to assign the Indel variations to each interval. In particular, a dictionary called "SNV" is defined, in which each bin is associated with a certain number of lines of the VCF file for which the variant allele frequency (VAF), obtained by the parse_VCF function, falls within the defined range from bin.
From this point on, the algorithm proceeds similarly to what was observed in the case of the Indel variations, retracing step by step all the actions described for the Get_indel_from_freq-distr.py script.
Therefore, taking up the conclusive part of the algorithm, the crucial aspect consists in the realisation of the sampling of the SNV variations through the definition of the so-called "*support-unit*".
This process in principle provides for the calculation of ratios as the ratio between the

number of lines of the VCF associated with a given bin (information obtained by calculating the length of the SNV[bin] list) and the frequency calculated for that specific bin (corresponding to the third value contained in the distr list for each index).
All reports will then be saved in the "ratios" list.
On the other hand, in the "non_zero_ratios" list, all the elements of the "ratios" list other than zero will be saved. At this point the "support-unit" variable takes on the minimum value among those identified in the "non_zero_ratios" list, unless this is empty, in which case the "support-unit" is set equal to zero.
For each bin, the product between the "support-unit" and the number of variations identified for that interval (saved in distr_dict[bin]) is subsequently calculated, stored in the variable N.
If N is lower than the number of rows of the VCF associated with the bin considered (N <len (SNV [bin])), then N is randomly selected from the lines of the VCF associated with the bin considered and are saved in "sampled_SNV "; otherwise (if N>=len(SNV [bin])) all lines of the VCF associated with the considered bin are saved in "sampled_SNV". Finally, all the contents of "sampled_SNV" are printed.

Recalibrate_HRD_A.sh

The first step concerns the acquisition of the files required for the script's execution through input.
Initially, it is acquired the name of the directory where the script is executed, saved in BIN_DIR. From this directory it is possible to go back to the Indel_median-freq_from_N-T_A and SNV_median-freq_from_N-T_A files, contained in the directory itself, corresponding to the reference distribution files in the format as the GSL-histogram output, with a little modification.

0	10	0
10	20	0.0381007
20	30	0.03151765
30	40	0.01347675
40	50	0.00401542
50	60	0.000756227
60	70	0
70	80	0
80	90	0
90	100	0

0	10	0.000308725
10	20	0.03462785
20	30	0.0273783
30	40	0.0159768
40	50	0.006558705
50	60	0.00233802
60	70	0.000655551
70	80	0.0002412795
80	90	0
90	100	0

Table 5.2. Median distribution of the Indel (on the left) and SNV (on the right) variations resulting from the comparison between the tumour sample and the associated normal (N-T distribution)

The Indel_median-freq_from_N-T and SNV_median-freq_from_N-T files (as shown in

the Table 5.2) respectively represent the median distribution of the SNV and Indel variations resulting from the comparison between the tumour sample and the associated normal, corresponding to the red curve in the graphs previously illustrated.
The Indel_median-freq_from_N-T_A and SNV_median-freq_from_N-T_A files, in particular, are obtained by considering only the value in correspondence with the bin defined between 10 and 20 and setting all others equal to zero (as shown in the Table 5.3).

0	10	0
10	20	0.0381007
20	30	0
30	40	0
40	50	0
50	60	0
60	70	0
70	80	0
80	90	0
90	100	0

0	10	0
10	20	0.03462785
20	30	0
30	40	0
40	50	0
50	60	0
60	70	0
70	80	0
80	90	0
90	100	0

Table 5.3. Median distribution of the Indel (on the left) and SNV (on the right) variations resulting from the comparison between the tumour sample and the associated normal (N-T distribution), limited to the portion of variations that reside between the frequencies of 10 and 20%

This trick specifically allows to select only the information contained in the 10-20 range, "cutting" the portion of variations that reside between the frequencies of 10 and 20%.
This occurs since in that range the distribution of the N-T comparison is strongly spiked, leading to exclude the frequencies with a negligible incidence compared to that of the peak.
Exporting the described files is followed by verification that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that the directory.files has been provided followed by the name of the tumour sample considered, saved respectively as FILES_DIR and TUMOUR.
In particular, the first input refers to a folder obtained as a result of the execution of the *cpgwgs* pipeline and the subsequent application of filters, corresponding to the input of the HRD pipeline.
The second input instead refers to the name of the tumour sample, which can be identified in the terminal part of the header of the BAM file of the tumour sample, preceded by the abbreviation "SM:" (as can be seen in the figure for example for patient *RC100851*).

```
@RG    ID:1    LB:dummy_LB    PL:dummy_PL    PU:dummy_PU    SM:RC100851_T
```

The section dealing with input acquisition is followed by a section dedicated to the description of the functions used within the script.
As previously introduced the VCF format includes a header, whose lines are identified by the "##" symbols and a body which lists all the variants. Inside the body, the information

relating to each variant is organised according to 9 mandatory columns that are identified by the "#" symbol.
The sampling_indel function, starting from a VCF file relating to the Indel variations (first input supplied to the function), has the aim of returning a new VCF file, saved with the name indicated in correspondence with the second input of the function, characterised by the same header of the starting one followed by the corresponding lines of the VCF file selected by sampling.
Specifically, the function begins with the extraction of the header and the first line of the body of the VCF file taken as input and proceeds with the execution of the script created in Python described above (Get_indel_from_freq-distr.py) providing it as input the median distribution of the Indel variations resulting from the comparison between the tumour sample and the associated normal and the VCF file.
Subsequently, the function foresees the ascending ordering of the rows of the VCF obtained following the execution of the sampling, sorted alphabetically by its column 1 (corresponding to the chromosome on which the variant is present) and sorted numerically by its column 2 (corresponding to the position on which the variant call was made respectively).
Finally, the resulting file, including the header followed by the VCF lines selected by sampling, is compressed in the form "file.gz" and saved in the file corresponding to the second input of the function.
The last step involves the generation of the VCF file in the form “vcf.gz” and of its index file “vcf.gz.tbi”.
Subsequently, the sampling_SNV function is defined. It is analogous to the sampling_indel function just described, with the only difference that it involves the execution of the Python script Get_SNV_from_freq-distr.py providing it as input the median distribution of the SNV variations resulting from the comparison between the tumour sample and the associated normal and the VCF.
The main body of the algorithm (viewable in the Appendix C in the form of pseudocode) can be simply described through the identification of four fundamental steps, as observable in the Figure 5.7.
Initially, the ***recalibrate_HRD_A.sh*** script foresees the execution of the sampling on the data that describe the distribution obtained from the comparison between the tumour sample and the normal equivalent.
In particular, the sub-folders of the FILES_DIR folder provided as the first input to the script are initially defined: specifically, the two paths FILES_DIR/recalibrateA/Indel-sampling and FILES_DIR/recalibrateA/SNV-sampling are generated (Figure 5.8).
The first step is then performed inside the recalibrateA folder, separately for SNV and Indel.
In particular, the sampling_indel function (defined previously) is launched receiving as parameters the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.RANDOM-1.gz (contained in the Indel-sampling folder).
Similarly, the sampling_SNV function (previously defined) is launched receiving as parameters the file in the form SNV.*.vcf.gz (contained in FILES_DIR) followed by the SNV.*.vcf.RANDOM-1.gz file (contained in the folder SNV-sampling).
Consequently, in the Indel-sampling and SNV-sampling folders (Figure 5.9), it will be

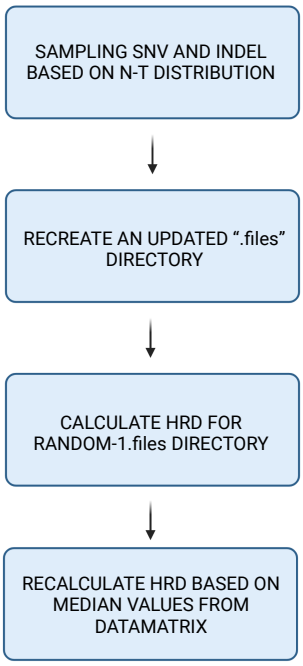

Figure 5.7. Recalibrate_HRD_A.sh core workflow

possible to access the files containing the sampling results on the data describing the distribution obtained from the comparison between the tumour sample and the normal equivalent, respectively for insertions/deletions and for single nucleotides variants.
In particular, in this case, the Strategy A directly considers only what concerns the slice of allelic frequencies containing the greatest content of somatic mutations, without sampling. In this regard, it is in fact possible to note that the sampling_indel and the sampling_SNV functions are launched only once.

During the second step, the ***recalibrate_HRD_A.sh*** script takes care of recreating an updated '.files' directory.
Specifically, for the file in the form Indel.*_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz (for example Indel.PD24215a_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz) contained in the folder Indel-sampling, generated during Step 1, is performed a series of actions outlined below.
Directory is initially defined and created in the form RANDOM-1.files (Figure 5.10; the files Indel.*, CNV.*, SNV.*, SV.* (Indel.PD24215a_vs_hg38_metanormal.annot.vcf.gz, ...), contained in the FILES_DIR folder, are copied into this directory.
At this point, the Indel replacement occurs, which involves overwriting the file in the form Indel.*gz contained in the respective RANDOM-1.files directory (Indel.PD24215a_vs_hg38 _metanormal.annot.vcf.gz) with the file in the form Indel.*_vs_hg38_metanormal .annot.vcf.RANDOM-1.gz (Indel.PD24215a_vs_hg38_metanormal.annot.

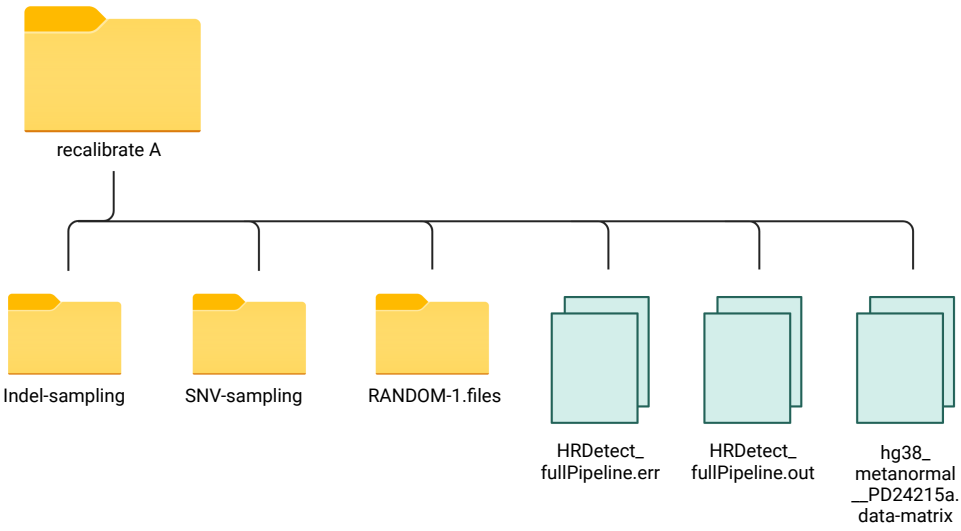

Figure 5.8. Content of the *recalibrate A* folder for patient PD24215a

vcf.RANDOM-1.gz).
The same is done for the file in the form Indel.*gz.tbi, replaced with the file in the form Indel.*_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz.tbi.
The same process is repeated analogously for the replacement of the SNVs inside the RANDOM-1.files directory: in particular the SNV.*_vs_hg38_metanormal.annot .muts.vcf.RANDOM-1.gz file contained in SNV-sampling (for example SNV.PD24215a_vs _hg38_metanormal.annot.muts.vcf.RANDOM-1.gz) overwrites the file SNV.*gz contained in the respective RANDOM-1.files (SNV.PD24215a_vs_hg38_metanormal.annot.muts.vcf.gz, considering the previous example), as well as for the file in the form SNV.*gz.tbi.

During the third step, the ***recalibrate_HRD_A.sh*** script takes care of calculating HRD for the RANDOM directory. Specifically, for the directory in the form RANDOM-1.files, contained in the recalibrateA folder, the pipeline HRDetect_fullPipeline-hg38.AUTO.sh is run, passing it as parameters the sample name of the tumour (saved in $TUMOUR) followed by the directory.
The script output will be saved in the HRDetect_fullPipeline.out file while any errors in the HRDetect_fullPipeline.err file, both contained in RANDOM-1.files directory.

During the fourth step, the ***recalibrate_HRD_A.sh*** script takes care of recalculating HRD based on median values from datamatrix.
In particular, the HRDetect pipeline requires an input data frame "data_matrix", which

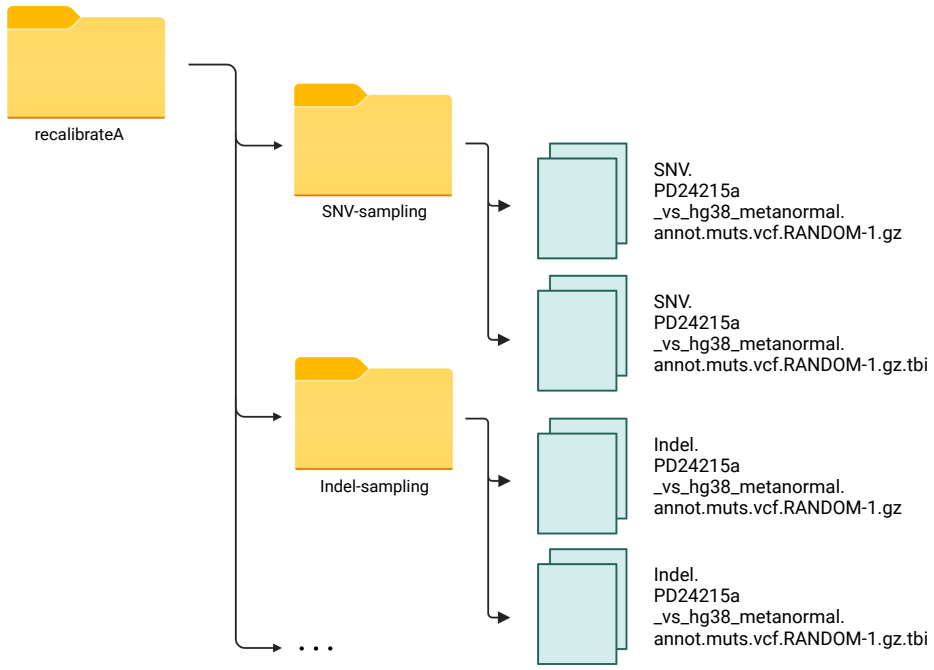

Figure 5.9. Content of *SNV-sampling* and *Indel-sampling* folders for patient PD24215a

contains a sample in each row and one of six necessary features in each column. The six features are:

- proportion of deletions at microhomology (del.mh.prop),
- number of mutations of substitution signature 3 (SNV3),
- number of mutations of rearrangement signature 3 (SV3),
- number of mutations of rearrangement signature 5 (SV5),
- HRD LOH (Loss of Heterozygosity) index (hrd),
- number of mutations of substitution signature 8 (SNV8).

Initially, for the HRDetect_fullPipeline.out file (contained in the folder in the form RANDOM-1.files), generated during the previous step, I look for the values that I need to define the data matrix (Figure 5.11). Through the "awk" command, I look for the values corresponding to the fields of del.mh.prop, SNV3, SNV8, SV3, SV5 and hrd, as well as the name of the sample, in the RANDOM-1.files/HRDetect_fullPipeline.out. At this point the column names corresponding to del.mh.prop, SNV3, SV3, SV5, hrd, SNV8, the name of the sample and the values associated with the previous columns, are saved in hg38_metanormal___*.data-matrix. Finally, the pipeline _HRDetect-score_from_data-matrix.R is launched, passing it as input the hg38_metanormal___*.data-matrix. The

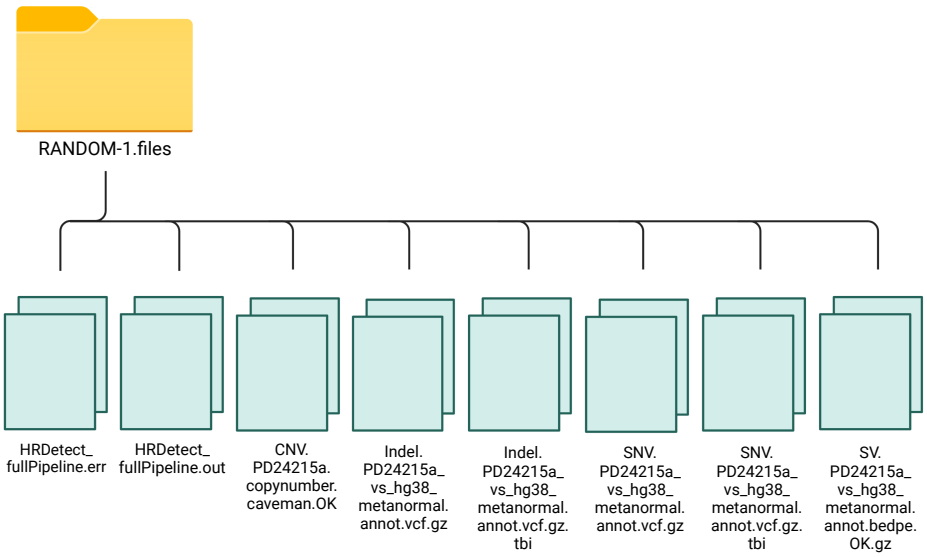

Figure 5.10. Content of *RANDOM1.-files* folder for patient PD24215a

script output will be saved in the HRDetect_fullPipeline.out file while any errors in the HRDetect_fullPipeline.err, both contained in the recalibrateA folder (Figure 5.8).

5.3.2 Results obtained with Strategy A for breast cancer

Using the data of Nik-Zainal you therefore have a score that is the expected one derived from the match between the normal and his tumour and then the result that is predicted using the metanormal.
In particular, according to the work carried out by Nik-Zainal's group, score values higher than 0.70 are defined as HR deficient while score values lower than 0.30 are considered HR proficient. All scores between 0.30 and 0.70 are indicated as doubtful, i.e. it is not possible to attribute proficiency or deficiency on the basis of the observed value.
Taking this classification into account, as can be seen in the Table 5.4, a color code was used that identified HR deficiency with red, HR proficiency with green and doubtful situations with orange, for which it was not possible to express.
Analyzing the results, it is possible to note how overall Strategy A does not provide faithful results to those expected, obtained by comparing the tumour sample and the normal matched one.
In general, in fact, out of 73 samples of breast cancer patients, exactly 18 scores are different from those predicted.
In particular, it should be noted that the samples for which the expected score was doubtful

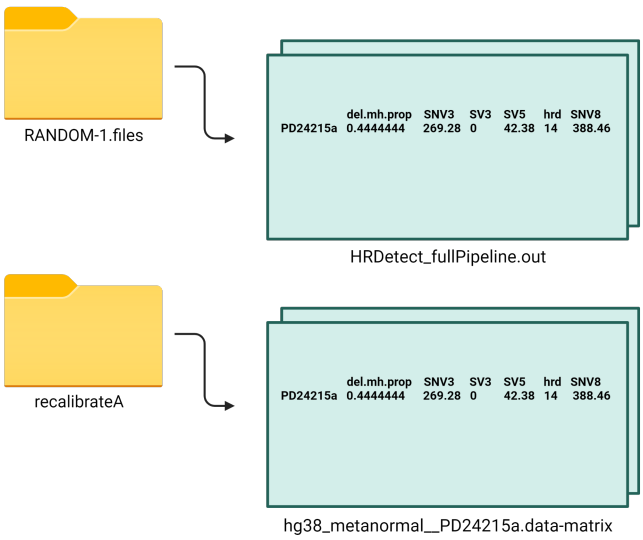

Figure 5.11. Content of *HRDetect_fullPipeline.out* and *hg38_metanormal_ _PD24215a.data-matrix* files for patient PD24215a

were excluded from the study (4 out of 77); this because, since it is not possible to express with certainty regarding the data obtained from the comparison between the tumour sample and the normal matched one, in the same way the result obtained by comparing the tumour sample with the metanormal could not have been considered as certain.
A particularly important aspect to underline is that in this case not all the samples that are expected to be HR deficient are actually correctly predicted by Strategy A; in fact, specifically, for 2 out of 18 samples HR deficiency is predicted when HR proficiency is expected. Similarly, not all samples that are expected to be HR proficient are predicted correctly by Strategy A: the PD22360a sample is an exception, unique among the 18 erroneously predicted samples, for which HR proficiency is predicted although it is HR deficient.
It follows that the most common errors attributable to Strategy A are related to the declaration of doubtful situations for samples that are expected as HR proficient or HR deficient.
Specifically, again with reference to these cases which were incorrectly assessed, 15 out of 17 samples are indicated with the color orange, of which 13 are HR proficient, while the remaining 2 are HR deficient.
In summary, Strategy A correctly predicts the expected score 55 times out of 73 samples considered as a whole, committing a percentage error of 24.66%.

5.4 Strategy B

With the same objective as the previous one, a further strategy has been developed, to which I will refer with the name of Strategy B, in order to improve the prediction of the HR score.
Strategy B represents in particular an evolution of Strategy A which aims to determine in the "optimal" frequency range, that is the window in which the greatest number of somatic variations reside.
In particular, in fact, from the point of view of statistical information, it is possible to realise that, when the normal matched sample (which would subtract all the germinal part leaving the somatic part) is not available, in the mix between germinal and somatic (dirty data obtained with the MTN-T comparison), most of the somatic information of interest is in the region where the N-T distribution peaks.
In this case, it is sufficient to consider the portion of variations that reside between the frequencies of 5 and 35% (range in which the distribution of the N-T comparison is spiked), both as regards single nucleotide variations and insertions/deletions (Figure 5.12).
Thus, the concept of Strategy B consists in removing everything that resides before 5% and after 35%.
In this case, specifically, the width of the window considered is equal to 30%, so there is an overlap with Strategy A, but with a larger window width and thus more successfully.

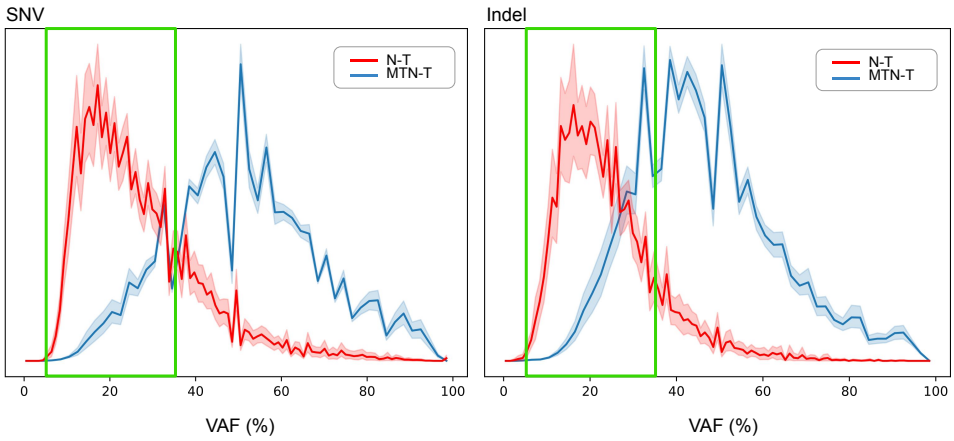

Figure 5.12. Selection of a specific range of frequency (5-35%) from the normalized Variant Allele Frequency (VAF) distributions for MTN-T (blue) and N-T (red) comparisons for SNV (on the left) and Indel (on the right) variations

In particular, it should be emphasised that, similarly to Strategy A, Strategy B does not provide for any type of sampling; on the other hand, Strategy B foresees the choice of a particular window (shifted by 5% to the left with respect to the previous strategy and with an increased delta of 10%) and a cut at the ends of the window currently considered

for the calculation of the score.

5.4.1 Methods

Get_VAF-range_indel.py

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that the lower and upper extremes of the considered interval were provided, followed by the VCF file relating to the Indel variations, saved respectively as start, end and INDEL_fd.
The definition of the functions in the Get_VAF-range_indel.py script perfectly follows the one made for the Get_indel_from_freq-distr.py script, consequently only some crucial aspects are reported.
Specifically, the function get_FORMAT_ID , given a row of the VCF file and the identification name of the specific field of the FORMAT (corresponding to ID), has the aim of extracting the value contained in the column relating to the tumour sample in the position corresponding to ID in the FORMAT.
parse_VCF function, starting from a VCF file relating to the Indel variations, has the aim of returning the value of the variant allele frequency, followed by the corresponding entire line of the VCF file.
The function read_ref_distribution , starting from the reference distribution file in the format as the GSL-histogram output, returns an ordered nested list containing, for each index, the three decimal numbers corresponding respectively to the start, end and number of variations of the interval.
In particular, a dictionary called "Indels" is defined, in which the bin is associated with a certain number of lines of the VCF file for which the variant allele frequency (VAF), obtained by the parse_VCF function, falls within the defined range from bin.
It is of particular importance to underline that, unlike what was observed for the Get_indel_from_freq-distr.py script, in this case a single specific interval is considered, of which the extremes are supplied as input.
At the same time, the crucial aspect of the ***Get_indel_from_freq-distr.py*** algorithm consists in the realisation of the sampling of the Indel variations through the definition of the so-called "support-unit"; in particular, for each bin, the product between the "support-unit" and the number of variations identified for that interval was calculated and stored in the variable N. If N was lower than the number of rows of the VCF associated with the bin considered, then N was randomly selected from the lines of the VCF associated with the bin considered and were saved in "sampled_indels"; otherwise all lines of the VCF associated with the considered bin were saved in "sampled_indels".
Finally, all the contents of "sampled_indels" were printed. In this case, however, in the ***Get_VAF-range_indel.py*** script, neither the "support-unit" nor sampling is used, in fact, once the VCF lines associated with the specific bin considered have been identified, these are printed in their wholeness.

Get_VAF-range_SNV.py

The two functions, get_FORMAT_ID and read_ref_distribution, are defined.
These are equal to the functions of the same name described for the Get_indel_from_freq-distr.py script, therefore in particular: the function get_FORMAT_ID, given a row of the VCF file and the identification name of the specific field of the FORMAT (corresponding to ID), has the aim of extracting the value contained in the column relating to the tumour sample in the position corresponding to ID in the FORMAT; the function read_ref_distribution, starting from the reference distribution file in the format as the GSL-histogram output, returns an ordered nested list containing, for each index, the three decimal numbers corresponding respectively to the start, end and number of variations of the interval.
Differently from what was observed for the ***Get_indel_from_freq-distr.py*** script, in this case to extract the frequencies of SNV variations, starting from a VCF format file relating to the SNV variations, it is necessary to consider only one specific field of the FORMAT column, described in the header of the file as follows:

```
##FORMAT= <ID= PM, Number= 1, Type= Float,
Description= "Proportion of mutated allele">
```

Specifically, the definition of the parse_VCF function differs from that made for the Get_indel_from_freq-distr.py and Get_VAF-range_indel.py scripts, but it perfectly follows that outlined for the Get_SNV_from_freq-distr.py script. It follows that the parse_VCF function, starting from a VCF file relating to the SNV variations, has the aim of returning the value of the variant allele frequency, followed by the corresponding entire line of the VCF file.
From this point on, the algorithm proceeds similarly to what was observed in the case of the Indel variations, retracing step by step all the actions described for the Get_VAF-range_indel.py script.
The only substantial difference consists in the files provided in input, in fact, it is verified that the lower and upper extremes of the considered interval are provided followed by the VCF file relating to the SNV variations, saved respectively as start, end and SNV_fd.
Again it is possible to emphasise, unlike what was observed for the Get_SNV_from_freq-distr.py script, in this case a single specific interval is considered, of which the extremes are supplied as input.
At the same time, the crucial aspect of the ***Get_SNV_from_freq-distr.py*** algorithm consists in the realisation of the sampling of the SNV variations through the definition of the product between the "support-unit" and the number of variations identified for that interval, stored in the variable N.
Then, if N was lower than the number of rows of the VCF associated with the bin considered, N was randomly selected from the lines of the VCF associated with the bin considered and were saved in "sampled_SNV"; otherwise all lines of the VCF associated with the considered bin were saved in "sampled_SNV".
Finally, all the contents of "sampled_SNV" were printed. In this case, however, in the ***Get_VAF-range_SNV.py*** script, neither the "support-unit" nor sampling is used, in fact, once the VCF lines associated with the specific bin considered have been identified,

these are printed in their wholeness.

Recalibrate_HRD_B.sh

As observed for all the scripts described above, the initial section is dedicated to acquiring the inputs needed to run the script itself.
I verify that the script has been launched correctly, introducing a check that verifies that what is needed for the script to run has been provided as input. In particular, I check that the name of the directory where the script will be executed has been provided, followed by the name of the tumour sample and the name of the output directory where the results will be printed, saved respectively as FILES_DIR, TUMOUR and OUT_DIR.
At this point I proceed with the definition of variables such as L (which is assigned the value 30), corresponding to the VAF range or the "width" of the interval considered, and $SHIFT$ (which is assigned the value 5), corresponding to the sliding window shift.
It proceeds with the creation of the directory OUT_DIR that will contain the results produced by the execution of the script and with the definition of some necessary functions for the same execution.

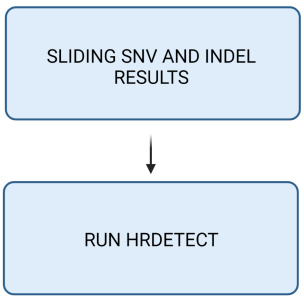

Figure 5.13. Recalibrate_HRD_B.sh core workflow

As previously introduced the VCF format includes a header, whose lines are identified by the "##" symbols and a body which lists all the variants. Inside the body, the information relating to each variant is organised according to 9 mandatory columns that are identified by the "#" symbol. The sliding_indel function, starting from a VCF file relating to the Indel variations (third input supplied to the function), has the aim of returning a new VCF file, saved with the name indicated in correspondence with the fourth input of the function, characterised by the same header of the starting one followed by the corresponding lines of the VCF file selected by sliding.
Specifically, the function begins with the extraction of the header and the first line of the body of the VCF file taken as input and proceeds with the execution of the script created in python described above (Get_VAF-range_indel.py) providing it as input the lower and upper extremes of the considered interval and the VCF file.
Subsequently, the function foresees the ascending ordering of the rows of the VCF obtained following the execution of the sliding, sorted alphabetically by its column 1 (corresponding

to the chromosome on which the variant is present) and sorted numerically by its column 2 (corresponding to the position on which the variant call was made respectively).
Finally, the resulting file, including the header followed by the VCF lines selected by sampling, is compressed in the form "file.gz" and saved in the file corresponding to the second input of the function.
The last step involves the generation of the VCF file in the form “vcf.gz” and of its index file “vcf.gz.tbi”.
The sliding_SNV function is analogous to the sliding_indel function just described, with the only difference that it involves the execution of the python script Get_VAF-range_SNV.py providing it as input the lower and upper extremes of the considered interval and the VCF.
The definition of the run_HRDetect function follows.
Starting from the sample name of the tumour and the SNV, SV, INDEL, CNV files given as input, the function has the aim of returning the results obtained by the running of the HRDetect pipeline, using the listed input as parameters.
The main body of the algorithm (viewable in the Appendix C in the form of pseudocode) can be simply described through the identification of two fundamental steps, as observable in the Figure 5.13.

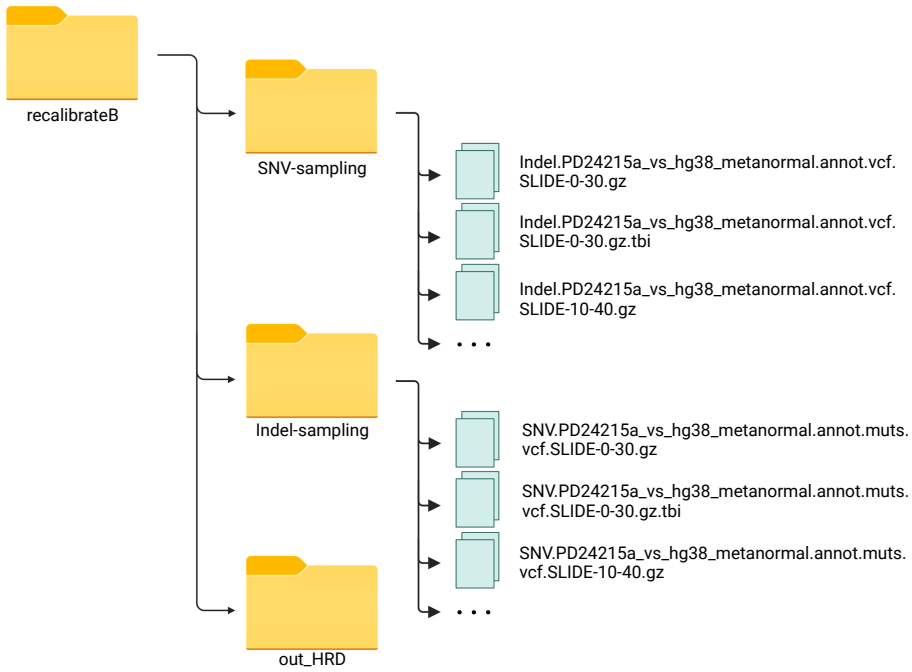

Figure 5.14. Content of *SNV-sampling* and *Indel-sampling* folders for patient PD24215a

Initially, the ***Recalibrate_HRD_B.sh*** script foresees the execution of the sliding on the SNV and Indel results.
In particular, the sub-folders of the OUT_DIR folder provided as the third input to

the script are initially defined: specifically, the three paths OUT_DIR/Indel-sliding, OUT_DIR/SNV-sliding and OUT_DIR/out_HRD are generated (figure 5.14.
The first step is then performed inside the OUT_DIR folder, separately for SNV and Indel.
In particular, the sliding_indel function (defined previously) is launched for each "start" in a sequence ranging from 0 to 100-$L (ie 70) with a step of 5 ($SHIFT = 5) receiving as parameters the values contained in $start and in $end, the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.SLIDE-$start-$end.gz (contained in the Indel-sliding folder), where "start" represents the index that is updated at each iteration. Similarly, the sliding_SNV function (previously defined) is launched for each "start" in the sequence [0,5,10, ... 65,70] receiving as parameters the values contained in $start and in $end, the file in the form SNV.*.vcf.gz (contained in FILES_DIR) followed by the file SNV.*.vcf.SLIDE-$start-$end.gz (contained in the SNV-sliding folder).
Consequently, in the Indel-sliding and SNV-sliding folders, it will be possible to access the files containing the sliding results on the data for insertions / deletions and for single nucleotides variants.
During the second and last step, the ***Recalibrate_HRD_B.sh*** script takes care of calculating the HRD score, running the function run_HRDetect. Specifically, for each file in the form Indel.*.gz, files generated during Step 1 contained in the Indel-sliding folder, the function run_HRDetect is run, passing it as parameters the sample name of the tumour (saved in $TUMOUR) followed by the files: SNV.*$range.gz contained in SNV-sliding (e.g. SNV.PD24215a_vs_hg38_metanormal.annot.muts.vcf.SLIDE-0-30.gz for patient PD24215a and range 0-30) , $FILES_DIR/SV.*.gz (SV.PD24215a_vs_hg38_metanormal .annot.bedpe.OK.gz), Indel-sliding/Indel.*.gz (Indel.PD24215a_vs_hg38_metanormal. annot.vcf.SLIDE-0-30.gz) and $FILES_DIR/CNV.*.OK (CNV.PD24215a.copynumber.caveman.OK).
The script output will be saved in the INDEL-$range.SNV- $range.HRDetect_fullPipeline.out file while any errors in the INDEL-$range.SNV-$range.HRDetect_fullPipeline.err file, both contained in $OUT_HRD directory.

5.4.2 Results

Again, using the data of Nik-Zainal one therefore has a score that is the expected one derived from the match between the normal and his tumour and then the result that is predicted using the metanormal.
In particular, in accordance with what has been explained for Strategy A, score values higher than 0.70 are defined as HR deficient while score values lower than 0.30 are considered HR proficient. All scores between 0.30 and 0.70 are indicated as doubtful, i.e. it is not possible to attribute proficiency or deficiency on the basis of the observed value.
Using this categorization, a color code was created, as shown in Table 5.5, that recognized HR deficiency with red, HR proficiency with green, and dubious situations with orange.
When looking at the data, it's easy to see how Strategy B mostly matches (but not exactly) the predicted findings produced by comparing the tumor sample to the normal matched sample.
In general, in fact, out of 73 samples of breast cancer patients, only 7 scores are different

from those predicted.
In particular, it should be noted, as previously done, that the samples for which the expected score was doubtful were excluded from the study (4 out of 77); this because, since it is not possible to express with certainty regarding the data obtained from the comparison between the tumour sample and the normal matched one, in the same way the result obtained by comparing the tumour sample with the metanormal could not have been considered as certain.
A particularly important aspect to underline is that in this case all the samples that are predicted to be HR deficient by Strategy B are actually such, as expected, unlike what happened for Strategy A.
It follows that the only errors attributable to Strategy B are related to the declaration of doubtful situations for samples that are expected as HR proficient or HR deficient. Specifically, again with reference to these incorrectly assessed cases, 2 out of 7 samples are indicated with the orange colour when they are HR proficient, while the remaining 5 samples are indicated with the orange colour when they are HR deficient.
In summary, Strategy B correctly predicts the expected score every time it shows the colour red or green, therefore HR deficiency or HR proficiency, committing a percentage error of 9.59% when identifying doubtful samples.

5.5 Strategy C

The failure of Strategy A found in a relevant number of cases (particularly with regard to the prediction of HR deficient scores), but also the hope of further improving the results achieved with Strategy B, has led to the development of a new strategy, which is called Strategy C, with the ultimate goal of predicting correctly the HR score starting from the comparison between the tumour samples and the constructed metanormal.
This strategy differs from the previous ones since, instead of considering only what concerns the slice of allelic frequencies containing the greatest content of somatic mutations, it directly extracts the frequencies starting from the average frequency distribution in the N-T comparison.
In this case, it is therefore not sufficient to "cut" the portion of variations that reside between two specific frequencies, as done for Strategy A and B, it is necessary to sample the complete data obtained with the metanormal, both as regards single nucleotide variations and insertions/deletions (Figure 5.3).
This strategy, which I will refer to as Strategy C, involves sampling the data with the (complete) metanormal in order to have a frequency range that instead reflects the distribution of the N-T comparison.
In particular, in fact, from the point of view of statistical information, it is possible to realise that, when the normal matched sample (which would subtract all the germinal part leaving the somatic part) is not available, in the mix between germinal and somatic (dirty data obtained with the MTN-T comparison), most of the somatic information of interest is in the region where the N-T distribution peaks.
This process therefore provides for a sampling of the "dirty" initial data, which, although it does not produce a very precise result, should not damage the correctness of the result

itself; this is because the signatures are extracted from the initial data, which in turn are statistical products, therefore, from a statistical point of view, not necessarily all the data must be good (somatic), but most of the data is sufficient. In this way there is a certain confidence that most (not all and not exactly) of the germline mutations have been removed.
To extrapolate the data, especially somatic, I try to reproduce the N-T curve. According to the proportion of the variations for each bin of the N-T distribution, the same proportion is extracted but starting from the distribution obtained with the metanormal, corresponding to the whole datum.
Specifically, I try to understand how much it is necessary to scale the MTN-T curve, extracting a certain number of variations per bin, so that it is below the N-T curve.
In reality, the extraction is independent of the number of variations, which is part of the calculation of the ratios / proportions: for example, the maximum mode peak of the N-T distribution does not coincide with the peak of the mode of the MTN-T distribution and, in particular, when variations are extracted at the peak of the mode of the MTN-T distribution (ie at a frequency of 50%) in reality the N-T distribution is almost zero, resulting in almost zero extraction.

5.5.1 Methods

The algorithm associated with Strategy C is created by introducing some specific changes to the one seen for Strategy A.
In particular, it is possible to notice how the entire flowchart coincides almost perfectly with the one described above, both as regards the Get_SNV_from_freq-distr.py script and Get_indel_from_freq-distr.py.
In light of the fact that some changes have been made, in order to adapt the workflow to the new strategy introduced, the new script created in bash will take the name of Recalibrate_HRD_C.sh (differentiating from that relating to Strategy A, called Recalibrate_HRD_A.sh).

Recalibrate_HRD_C.sh

The first step concerns the acquisition of the files required for the script's execution through input.
Initially, it is acquired the name of the directory where the script is executed, saved in BIN_DIR.
From this directory it is possible to go back to the Indel_median-freq_from_N-T and SNV_median-freq_from_N-T files, contained in the directory itself, corresponding to the reference distribution files in the format as the GSL-histogram output.
The Indel_median-freq_from_N-T and SNV_median-freq_from_N-T files (shown in the Table 5.2) respectively represent the median distribution of the SNV and Indel variations resulting from the comparison between the tumour sample and the associated normal, corresponding to the red curve in the graphs previously illustrated.
The first disparity that emerges between the two scripts (Recalibrate_HRD_A.sh and Recalibrate_HRD_C.sh) consists precisely in the files acquired as input.

In fact, it is possible to observe for the Strategy A script the definition of INDEL_NT_DISTRIBUTION as $BIN_DIR/ Indel_median-freq_from_N-T_A and of SNV_NT_DISTRIBUTION as $BIN_DIR/ SNV_median-freq_from_N-T_A, where $BIN_DIR represents the name of the directory in which the script is executed, within which you can find the files Indel_median-freq_from_N-T_A and SNV_median-freq_from _N-T_A, corresponding to the reference distribution file in the format of the GSL-histogram output with a small modification.

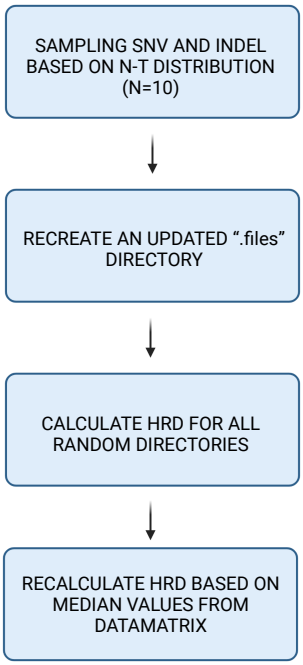

Figure 5.15. Recalibrate_HRD_C.sh core workflow

In particular, these files are obtained by copying all the values contained in the Indel_median-freq_from_N-T and SNV_median-freq_from_N-T files, respectively, by considering only the value in correspondence with the bin defined between 10 and 20 and setting all others equal to zero (as shown in the Table 5.3).
This trick allows in particular to select only the information contained in the range 10-20, "cutting" the part of variations that reside between the frequencies of 10 and 20%, as required by Strategy A.
Exporting the described files is followed by verification that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the directory.files has been provided followed by the name of the tumour sample considered, saved respectively as FILES_DIR and TUMOUR.
In particular, the first input refers to a folder obtained as a result of the execution of the *cpgwgs* pipeline and the subsequent application of filters, corresponding to the input of

the HRD pipeline.
The second input instead refers to the name of the tumour sample, which can be identified in the terminal part of the header of the BAM file of the tumour sample, preceded by the abbreviation "SM:" (as can be seen for example for patient RC100851).

```
@RG    ID:1    LB:dummy_LB    PL:dummy_PL    PU:dummy_PU    SM:RC100851_T
```

The section dealing with input acquisition is followed by a section dedicated to the description of the functions used within the script.
The sampling_indel function, starting from a VCF file relating to the Indel variations (first input supplied to the function), has the aim of returning a new VCF file, saved with the name indicated in correspondence with the second input of the function, characterised by the same header of the starting one followed by the corresponding lines of the VCF file selected by sampling.
Specifically, the function begins with the extraction of the header and the first line of the body of the VCF file taken as input and proceeds with the execution of the script created in python described above (Get_indel_from_freq-distr.py) providing it as input the median distribution of the Indel variations resulting from the comparison between the tumour sample and the associated normal and the VCF file.
Subsequently, the function foresees the ascending ordering of the rows of the VCF obtained following the execution of the sampling, sorted alphabetically by its column 1 (corresponding to the chromosome on which the variant is present) and sorted numerically by its column 2 (corresponding to the position on which the variant call was made respectively).
Finally, the resulting file, including the header followed by the VCF lines selected by sampling, is compressed in the form "file.gz" and saved in the file corresponding to the second input of the function. The last step involves the generation of the VCF file in the form “vcf.gz” and of its index file “vcf.gz.tbi”.
Subsequently, the sampling_SNV function is defined. It is analogous to the sampling_indel function just described, with the only difference that it involves the execution of the python script Get_SNV_from_freq-distr.py providing it as input the median distribution of the SNV variations resulting from the comparison between the tumour sample and the associated normal and the VCF.
The main body of the algorithm (viewable in the Appendix C in the form of pseudocode) can be simply described through the identification of four fundamental steps, as observable in the Figure 5.15.
Initially, the Recalibrate_HRD_C.sh script foresees the execution of the sampling on the data that describe the distribution obtained from the comparison between the tumour sample and the normal equivalent.
In particular, the sub-folders of the FILES_DIR folder provided as the first input to the script are initially defined: specifically, the two paths FILES_DIR/recalibrateC/Indel-sampling and FILES_DIR/recalibrateC/SNV-sampling are generated (Figure 5.16).
The first step is then performed inside the recalibrateC folder, separately for SNV and Indel. In particular, the sampling_indel function (defined previously) is launched N times receiving as parameters the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.RANDOM-$i.gz (contained in the Indel-sampling folder),

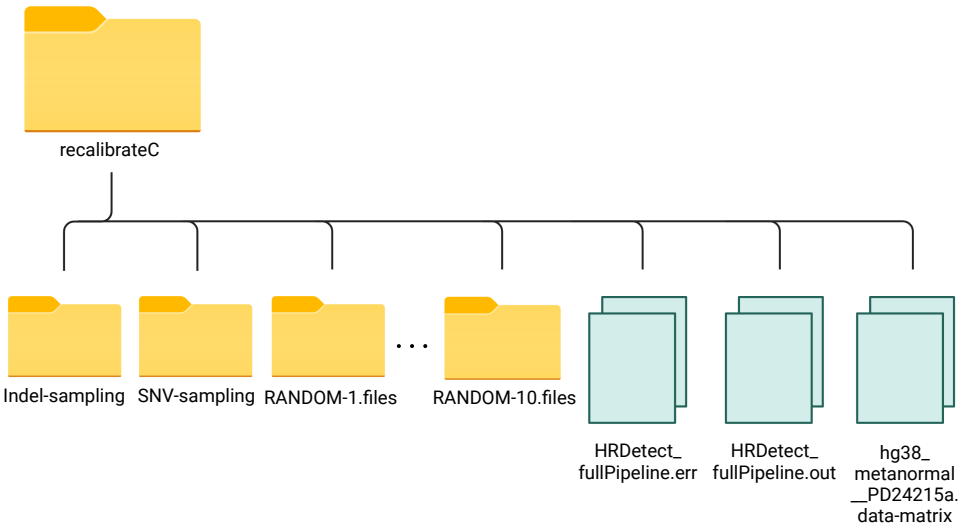

Figure 5.16. Content of *recalibrate C* folder for patient PD24215a

where "i" represents the index that is updated at each iteration proceeding from 1 to N. Similarly, the sampling_SNV function (previously defined) is launched N times receiving as parameters the file in the form SNV.*.vcf.gz (contained in FILES_DIR) followed by the SNV.*.vcf.RANDOM-$i.gz file (contained in the folder SNV-sampling).
Consequently, in the Indel-sampling and SNV-sampling folders, it will be possible to access the files containing the sampling results on the data describing the distribution obtained from the comparison between the tumour sample and the normal equivalent, respectively for insertions/deletions and for single nucleotides variants (Figure 5.17).
During the second step, the Recalibrate_HRD_C.sh script takes care of recreating an updated '.files' directory.
Specifically, for each file in the form Indel.*_vs_hg38_metanormal.annot.vcf.RANDOM-*.gz (for example Indel.PD24215a_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz) contained in the folder Indel-sampling, generated during Step 1, is performed a series of actions outlined below.
Directories are initially defined and created in the form RANDOM-*.files (RANDOM-1.files referring to the previous example); the files Indel.*, CNV.*, SNV.*, SV.* (Indel.PD24215a_vs_hg38_metanormal.annot.vcf.gz, ...), contained in the FILES_DIR folder, are copied into these directories.
At this point, the Indel replacement occurs, which involves overwriting the file in the form Indel.*gz contained in the respective RANDOM-*.files directory (Indel.PD24215a_vs _hg38_metanormal.annot.vcf.gz) with the file in the form Indel.*_vs_hg38_metanormal

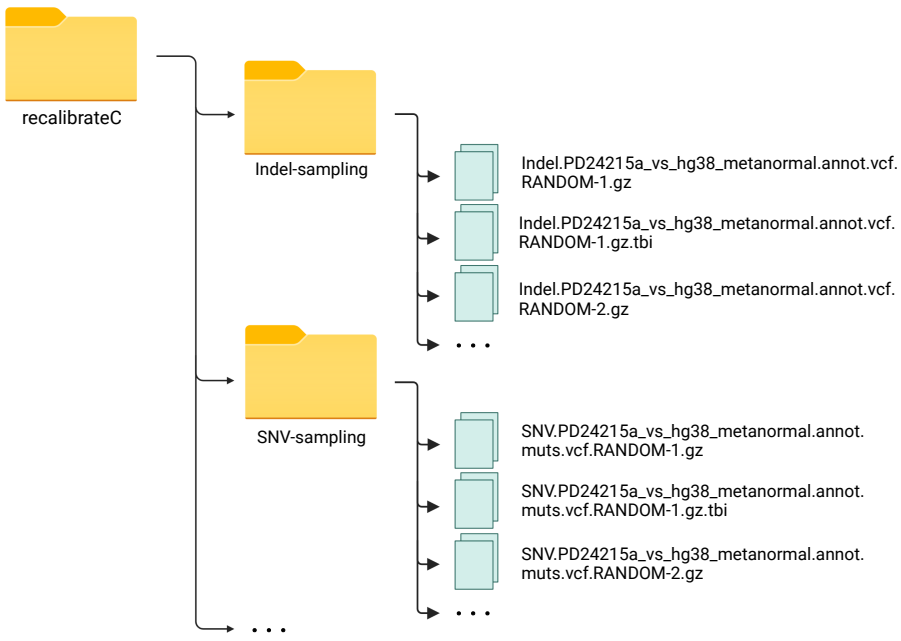

Figure 5.17. Content of *SNV-sampling* and *Indel-sampling* folders for patient PD24215a

.annot.vcf.RANDOM-*.gz (Indel.PD24215a_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz).
The same is done for the file in the form Indel.*gz.tbi, replaced with the file in the form Indel.*_vs_hg38_metanormal.annot.vcf.RANDOM-*.gz.tbi. The same process is repeated analogously for the replacement of the SNVs inside the RANDOM-*.files directory: in particular the SNV.*_vs_hg38_metanormal.annot.muts.vcf.RANDOM-*.gz file contained in SNV-sampling (for example SNV.PD24215a_vs_hg38_metanormal.annot .muts.vcf.RANDOM-1.gz) overwrites the file SNV.*gz contained in the respective RANDOM-*.files (SNV.PD24215a_vs_hg38_metanormal.annot.muts.vcf.gz considering the previous example), as well as for the file in the form SNV.*gz.tbi.
During the third step, the Recalibrate_HRD_C.sh script takes care of calculating HRD for all RANDOM directories.
Specifically, for each directory in the form RANDOM-*.files, contained in the recalibrateA folder (Figure 5.18), the pipeline HRDetect_fullPipeline-hg38.AUTO.sh is run, passing it as parameters the sample name of the tumour (saved in $TUMOUR) followed by the directory. The script output will be saved in the HRDetect_fullPipeline.out file while any errors in the HRDetect_fullPipeline.err file, both contained in RANDOM-*.files directory.

During the fourth step, the Recalibrate_HRD_C.sh script takes care of recalculating HRD based on median values from datamatrix. In particular, the HRDetect pipeline requires an input data frame "data_matrix", which contains a sample in each row and one

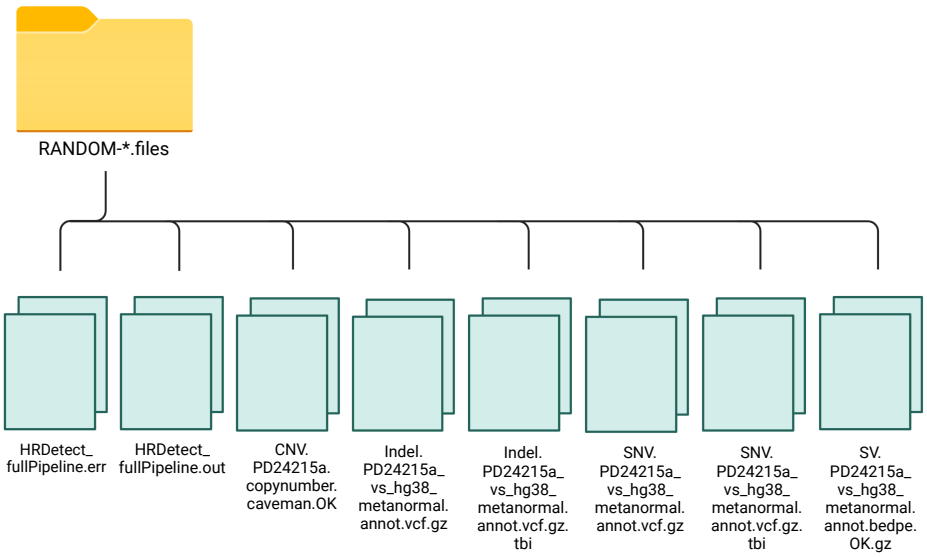

Figure 5.18. Content of *RANDOM-*.files* folder for patient PD24215a

of six necessary features in each column. The six features are:

- proportion of deletions at microhomology (del.mh.prop),
- number of mutations of substitution signature 3 (SNV3),
- number of mutations of rearrangement signature 3 (SV3),
- number of mutations of rearrangement signature 5 (SV5),
- HRD LOH (Loss of Heterozygosity) index (hrd),
- number of mutations of substitution signature 8 (SNV8).

Initially, for each of the HRDetect_fullPipeline.out files (contained in the folders in the form RANDOM-*.files), generated during the previous step, I look for the values corresponding to the fields of del.mh.prop, SNV3 and SNV8.
Of the N values (in this case N is defined equal to 10) extracted for each field from the HRDetect_fullPipeline.out files (since the RANDOM-*.files folders are N and each contains only one HRDetect_fullPipeline.out file), the medians are calculated (per column), obtaining three numbers saved in variables MH, SNV_3, SNV_8.
Through the "awk" command, look for the values corresponding to the fields of del.mh.prop, SNV3, SNV8, SV3, SV5 and hrd, as well as the name of the sample, in the RANDOM-1.files/HRDetect_fullPipeline.out.

Figure 5.19. Content of *HRDetect_fullPipeline.out* and *hg38_metanormal_ _PD24215a.data-matrix* files for patient PD24215a

At this point the "original" values corresponding to del.mh.prop, SNV3 and SNV8 are replaced with the median values calculated previously (ie MH, SNV_3, SNV_8) and are saved in hg38_metanormal___*.data-matrix the column names corresponding to del.mh.prop, SNV3, SV3, SV5, hrd, SNV8, the name of the sample and the values associated with the previous columns, taking into account the replacements (sample, MH, SNV_3, SV3, SV5, hrd, SNV_8).
Finally, the pipeline _HRDetect-score_from_data-matrix.R is launched, passing it as input the hg38_metanormal___*.data-matrix (Figure 5.19).
The script output will be saved in the HRDetect_fullPipeline.out file while any errors in the HRDetect_fullPipeline.err, both contained in the recalibrateC folder.
As it is possible to observe comparing the two scripts, the algorithm realised for Strategy C represents a generalisation of the one initially introduced for Strategy A.
In fact it is possible to obtain the same result by executing the script Recalibrate_HRD_A.sh or by executing the script Recalibrate_HRD_C.sh taking care to set $N = 1$ (which corresponds to the definition of the number of samplings performed).
In this way, during the Step 1, the function sampling_indel (previously defined) is launched N times (N is defined equal to 10 for the Strategy C, while N is defined equal to 1 for the Strategy A) receiving as parameters the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.RANDOM-$i. gz (contained in the folder Indel-sampling), where "i" represents the index that is updated at each iteration proceeding from 1 to N.
Similarly, the function sampling_SNV (previously defined) is launched N times receiving as parameters the file in the form SNV.*.vcf.gz (contained in FILES_DIR) followed by

the file SNV.*.vcf.RANDOM-$i.gz (contained in the folder SNV-sampling).
In this way, Strategy A directly considers only what concerns the slice of allelic frequencies containing the highest content of somatic mutations, without sampling (in fact you can see that the functions sampling_indel and sampling_SNV are launched only once).
At the same time instead, Strategy C involves sampling the data with the (full) metanormal to have a frequency range that reflects the distribution of the N-T comparison.

5.5.2 Results

Using the data of Nik-Zainal one therefore has a score that is the expected one derived from the match between the normal and his tumour and then the result that is predicted using the metanormal.
Score values more than 0.70 are classified as HR deficient, while score values lower than 0.30 are defined as HR proficient, according to what has been discussed for Strategy A and Strategy B. All scores between 0.30 and 0.70 are classified as dubious, meaning that the observed value cannot be used to identify them. A color code was constructed using this categorisation, as shown in Table 5.6, that highlighted HR deficiency with red, HR proficiency with green, and ambiguous scenarios with orange.
Analysing the results, it is possible to note how overall Strategy C traces well (although not perfectly) the expected results, obtained by comparing the tumour sample and the normal matched one.
In general, in fact, out of 73 samples of breast cancer patients, only 6 scores are different from those predicted. In particular, it should be noted that the samples for which the expected score was doubtful were excluded from the study (4 out of 77); this because, since it is not possible to express with certainty regarding the data obtained from the comparison between the tumour sample and the normal matched one, in the same way the result obtained by comparing the tumour sample with the metanormal could not have been considered as certain.
A particularly important aspect to underline is that in this case all the samples that are predicted to be HR deficient by Strategy C are actually such, as expected, unlike what happened for example for Strategy A.
Likewise, all the samples that are predicted to be HR deficient by Strategy C are actually such, as expected.
It follows that the only errors attributable to Strategy C, as noted for Strategy B, are related to the declaration of doubtful situations for samples that are expected as HR proficient or HR deficient (most common errors attributable to Strategy A).
Specifically, again with reference to these incorrectly assessed cases, 4 out of 6 samples are indicated with the orange color when they are HR proficient, while the remaining 2 out of 6 samples are indicated with the orange color when they are HR deficient.
For two of the latter, in particular PD24304a and PD22360, it is also possible to note that the predicted score (respectively 0.6575 and 0.5324) is remarkably close to the 0.70 value indicated as the threshold, showing a tendency to red.
In summary, Strategy C correctly predicts the expected score every time it shows the color red or green, therefore HR deficiency or HR proficiency, committing a percentage error of 8,22% when identifying doubtful samples, significantly lower than that identified

by Strategy A.

5.6 Strategy D

Up to this moment, the strategies developed were aimed at emulating a frequency distribution of the expected variations, corresponding to the comparison between the tumour sample and the matched normal sample.
These studies have underlined how the type of reasoning used allows to obtain sufficiently satisfactory results. On the other hand, however, some of the cases examined tend to flee from these techniques, reporting different scores from those expected.

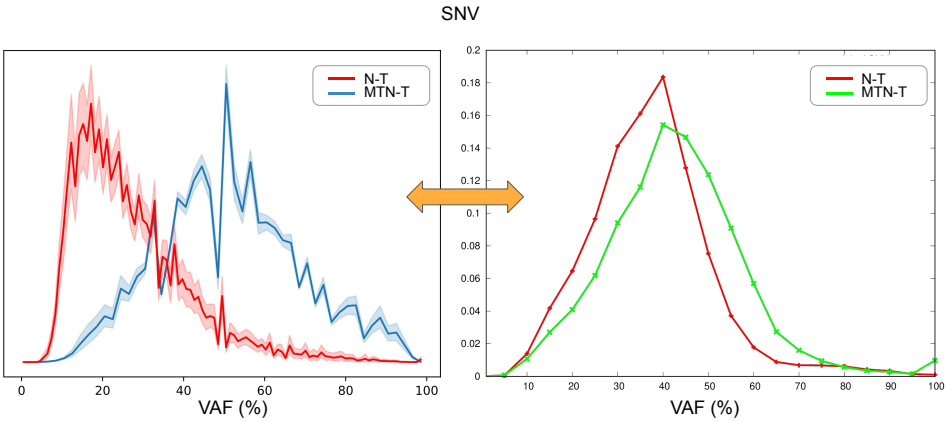

Figure 5.20. The average distribution of the frequencies of SNV for the N-T and MT-N-T comparisons (on the left) is compared with the sample-specific distribution of the frequencies of SNV for the N-T and MTN-T comparisons (on the right)

One of the possible responses to the misclassification of a certain number of samples from breast cancer patients lies in the fact that the contribution of somatic variations is not generalizable but is sample-specific, that is, dependent on the patient considered. The Figure 5.20 presented here is an example of what has just been described.
In particular, on the left it is again possible to observe the average distribution of the frequencies of single nucleotide variations obtained for the comparison between the tumour sample and the respective normal (indicated in red) and that obtained through the comparison between the tumour sample and the metanormal (indicated in blue), calculated using samples of the breast cancer data set, as previously described.
On the right, the analysis presented is related to a single breast cancer patient.
The distributions of the frequencies of single nucleotide variations obtained for the comparison between the tumour sample and the respective normal (indicated in red) and through the comparison between the tumour sample and the metanormal (indicated in

green) are presented. The patient considered in this case challenges the idea of a multi-sample generalisation of the distribution due to the contribution of somatic variations.
As can be observed from the Figure 5.20, although the green curve is comparable to the blue curve, the two red curves are considerably different, despite the fact that both are the result of a comparison between the tumour sample and the normal matched sample. So the curve for the specific patient deviates from that described as median.

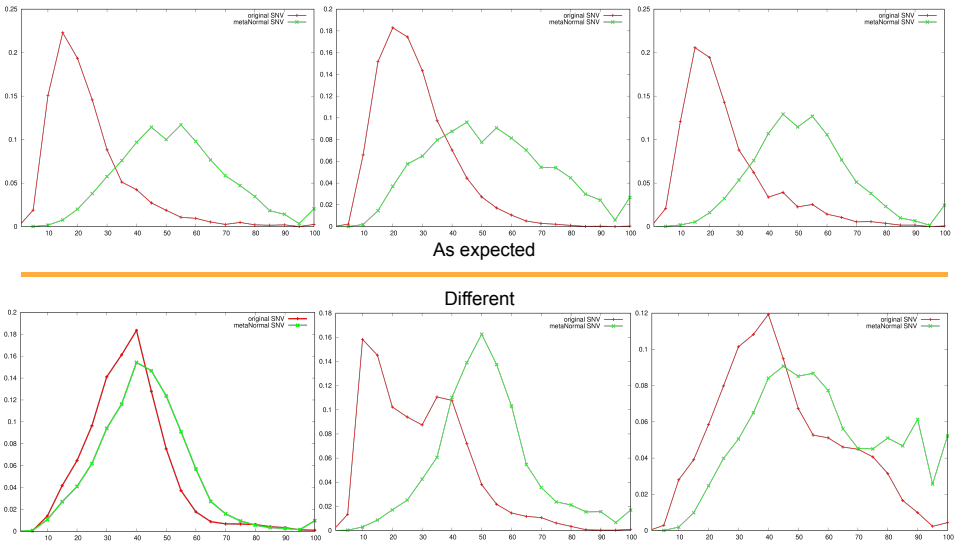

Figure 5.21. The sample-specific distributions of the frequencies of SNV, resulting from the N-T and MTN-T comparisons, are represented. The distinction is made between the samples for which the trend is the expected one (red curve Figure 5.20), indicated in the upper part of the Figure, and the samples for which the trend is different (lower part of the Figure).

At this point the question related to the correctness of the generalisation of the mean distribution of the variations in the N-T comparison arises spontaneously; trying to reproduce a distribution that is as close as possible to an "expected" distribution that does not reflect reality for the specific patient, could generate a confusion which would be reflected in the calculation of the HR score.
To analyse this aspect, different examples are then taken into consideration.
First of all it is necessary to specify that the analysis will be focused on the distributions of single nucleotide variations, but could be performed in parallel on the insertions/deletions as done previously.
The illustrated examples (Figure 5.21), for which green is used to describe the distribution of the MTN-T comparison while red to describe the one of the N-T comparison, show how in some cases the trend is the expected one (corresponding to the red curve of the previous figure), but in other cases, such as those represented in the lower part of the figure, this is not the case.
It is possible to note that some cases show trends that are not unimodal but bimodal, for which the expected/real distribution presents a double peak, which strongly deviates

from the trend shown by the generalised expectation. An hypothesis could be that the second peak, not represented by the theoretical red curve, may contain information that I would lose using the generalisation.

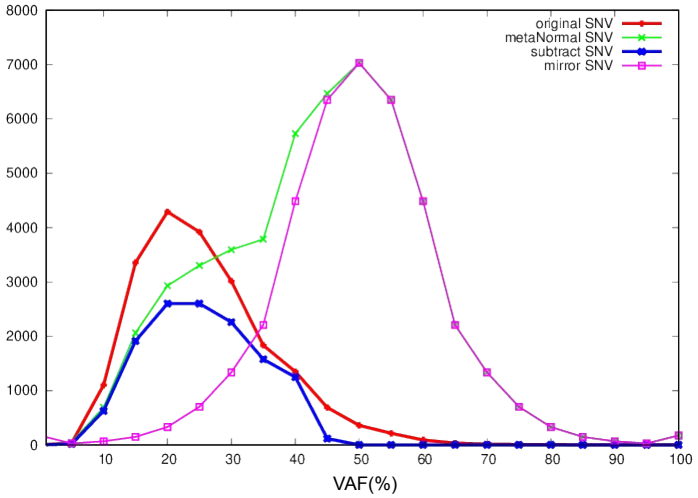

Figure 5.22. Representation of the distribution of the frequencies of SNV resulting from the N-T (red) and MTN-T (green) comparisons, as well as the mirror (purple) and subtraction (blue) curves, for a specific examined sample.

Furthermore, some samples in Figure 5.21, such as for example the one represented in the figure at the bottom right, have for both curves a good representation of frequencies above 50%.
These almost rare samples are probably purer (more tumorous and more clonal) and have very different trends compared to the generalised curve, underlined by the shift of the peak from 20% in the generalised to 40% in the real one.
The objective at this point becomes that of creating an expected curve that is sample/-patient specific and this provides the basis for the development of a new strategy which I will refer to as Strategy D.
The idea is based on the hypothesis that the contribution provided by the germline mutations can be seen by referring to the right tail of the distribution obtained as a result of the comparison between the metanormal and the selected tumour sample (indicated in green in the Figure 5.22), selecting in particular the section of the curve that corresponds to frequencies above 50%.
In this regard, assuming that the somatic mutations reside mainly in the frequencies below 50%, the idea is to use the trend of the curve for frequencies above 50% as a template, making it a mirror to obtain the trait corresponding to frequencies below 50%.
The result is a sample-specific bell function, symmetrical about the vertical axis defined by the equation $x = 50\%$.
I can now assume that the bell distribution I created (depicted in the Figure 5.22 by a

purple line) is completely germinal, and that the somatic component corresponds to the point-by-point subtraction of the MTN-T distribution (green) from the one I just constructed.
The resulting curve (indicated by the blue color in the Figure 5.22) appears more rounded in the part corresponding to frequencies around 20%.
The latter, which from a certain point of view represents more of a probability distribution of frequencies rather than a distribution of frequencies, will be generated by subtraction for each sample and will be used as an expected curve for subsequent studies, replacing the N-T comparison distribution.

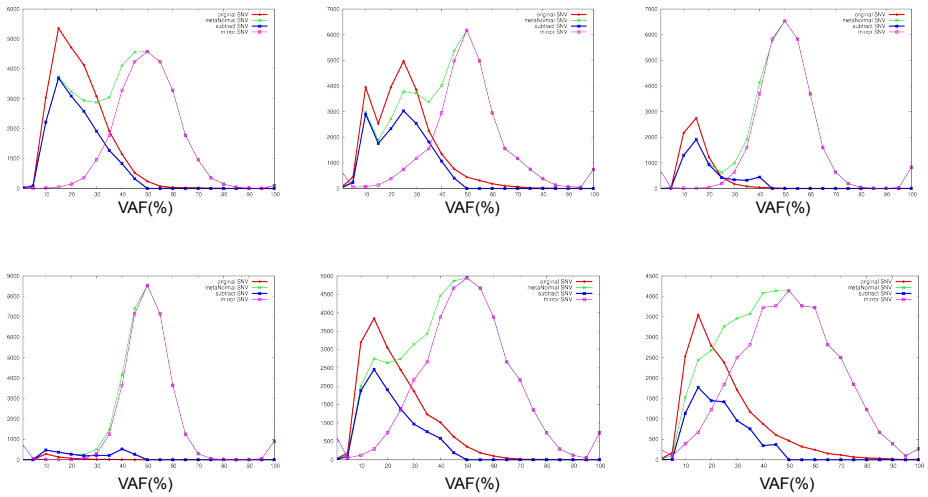

Figure 5.23. Representation of the distribution of the frequencies of SNV, resulting from the N-T (red) and MTN-T (green) comparisons, and of the mirror (purple) and subtraction (blue) curves, for different examined samples.

By observing the representative graph in Figure 5.22, it is of fundamental importance to observe how much the generated subtraction curve (blue) follows the expected red curve (the N-T comparison distribution).
Obviously the mirror curve (purple) does not perfectly represent the totality of the germinal variations and consequently some useful data could be lost.
Despite this, thanks to the use of the subtraction curve, it is possible to trace/predict a bimodal trend of the distribution of the N-T comparison, managing to reproduce even poorly represented trends.
In particular, it is necessary to underline that the subsection curve for construction will always go to zero over 50%, since starting from 50% the cast is coincident and therefore the subtraction is zero; it will therefore be possible to evaluate the behaviour of the curve only for frequencies below 50%.
Once the subtraction distribution has been outlined, the goal becomes to use this curve as an expected curve, different sample by sample (Figure 5.23), in order to repeat a sampling

similar to that carried out during Strategy C.

5.6.1 Methods

Get_Indel_freq-distr_anyPerc.NORM.sh

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that the VCF file relating to the Indel variations has been provided followed by the length of the bin, saved respectively as $INDEL$ and L.
Through the use of the "awk" command, the length of the bin acquired in input is used to calculate the number of total bins contained in an interval from 0 to 100, which is saved in the variable N.
The length of the bin is also used for the definition of the start and end, corresponding respectively to the left end of the first bin and the right end of the last bin outlined.
Specifically, these do not correspond directly to the values 0 and 100 as it is advisable for graphic reasons to translate the two positions so that, once the data has been used to create a histogram, the latter is centred and not superimposed on the axis lines.
At this point, the values corresponding to the number of bins, the start and the end are printed. Remember in particular that, as previously introduced, the VCF format includes a header, whose lines are identified by the "##" symbols, the 9 mandatory columns identified by the "#" symbol and a body which lists all the variants.
In this regard, I proceed by extraction of all the lines of the VCF file taken as input with the exception of those preceded by the symbol "#" (therefore only the lines belonging to the body of the file); for each of these lines splitting of the string of the last column of the VCF file into smaller strings using the ":" as a separator, saving these strings in the "t" array.
In fact, remember that to extract the frequencies of Indel variations, starting from a VCF format file relating to the Indel variations, it is necessary to consider two specific fields of the FORMAT column, described in the header of the file as follows:

```
## FORMAT = <ID = FD, Number = 1, Type = Integer,
Description = "Fragment depth">
## FORMAT = <ID = FC, Number = 1, Type = Integer,
Description = "Fragment calls">
```

I therefore proceed with the extraction of the value in the tumour sample corresponding to the position of "FC" in the FORMAT, saved in $t[n2]$ and with the extraction of the value in the tumour sample corresponding to the position of "FD" in the FORMAT, saved in $t[n2-1]$.
Subsequently, the frequencies of Indel variations are obtained through the ratio between the two identified values taken in the order described. Therefore, the result of multiplying the quotient obtained by 100 is printed, in order to obtain a percentage value of the variant allele frequency.
Finally, using the gsl-histogram command it was possible to organise the obtained data in

the required output format. In fact, the command takes three arguments, gsl-histogram $[-u]$ $xmin$ $xmax$ $[n]$, specifying the upper and lower bounds of the histogram and the number of bins.
It then reads numbers from "stdin", one line at a time, and adds them to the histogram. If $-u$ is given, histogram is normalised so that the sum of all bins is unity.

Get_SNV_freq-distr_anyPerc.NORM.sh

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that the VCF file relating to the SNV variations is provided followed by the length of the bin, saved respectively as SNV and L.
The algorithm proceeds similarly to what was observed in the case of the Indel variations, retracing step by step all the actions described for the Get_Indel_freq-distr_anyPerc .NORM.sh script.
Through the use of the "awk" command, the length of the bin acquired in input is used to calculate the number of total bins contained in an interval from 0 to 100, which is saved in the variable N. The length of the bin is also used for the definition of the start and end, corresponding respectively to the left end of the first bin and the right end of the last bin outlined.
Specifically, these do not correspond directly to the values 0 and 100 as it is advisable for graphic reasons to translate the two positions so that, once the data has been used to create a histogram, the latter is centred and not superimposed on the axis lines.
At this point, the values corresponding to the number of bins, the start and the end are printed.
Remember in particular that, as previously introduced, the VCF format includes a header, whose lines are identified by the "##" symbols, the 9 mandatory columns identified by the "#" symbol and a body which lists all the variants.
In this regard, I proceed by extraction of all the lines of the VCF file taken as input with the exception of those preceded by the symbol "#" (therefore only the lines belonging to the body of the file); for each of these lines splitting of the string of the last column of the VCF file into smaller strings using the ":" as a separator, saving these strings in the "t" array.
Differently from what was observed for the Get_Indel_freq-distr_anyPerc.NORM.sh, in this case to extract the frequencies of SNV variations, starting from a VCF format file relating to the SNV variations, it is necessary to consider only one specific field of the FORMAT column, described in the header of the file as follows:

```
##FORMAT= <ID= PM, Number= 1, Type= Float,
Description= "Proportion of mutated allele">
```

I therefore proceed with the extraction of the value in the tumour sample corresponding to the position of "PM" in the FORMAT, saved in $t[n2]$.
Subsequently, the percentage values of frequencies of SNV variations are obtained by multiplying the identified value by one hundred.

Finally, using the gsl-histogram command it was possible to organise the data obtained in the required output format, as seen for Get_Indel_freq-distr_anyPerc.NORM.sh script.

Get_Indel_freq-distr_anyPerc.sh

The algorithm is identical to that described for Get_Indel_freq-distr_anyPerc.NORM.sh script.
The only difference lies in the use of the gsl-histogram command, which was used in the previous cases with the $-u$ feature, which implies the creation of a normalised histogram so that the sum of all bins is unity.
In the case of Get_Indel_freq-distr_anyPerc.sh the command is executed without this feature, consequently the obtained histogram is not normalised.

Get_SNV_freq-distr_anyPerc.sh

The algorithm is identical to that described for Get_SNV_freq-distr_anyPerc.NORM.sh script.
The only difference lies in the use of the gsl-histogram command, which was used in the previous cases with the $-u$ feature, which implies the creation of a normalised histogram so that the sum of all bins is unity. In the case of Get_SNV_freq-distr_anyPerc.sh the command is executed without this feature, consequently the obtained histogram is not normalised.

_get_mirror_distribution.py

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that the distribution file relating to the Indel variations or to the SNV variations has been provided.
In particular, the input requested by this python script corresponds to the output in the format obtained as a result of the Get_SNV_freq-distr_anyPerc.sh, Get_Indel_freq-distr_anyPerc.sh, Get_SNV_freq-distr_anyPerc.NORM.sh, Get_Indel_freq-distr _anyPerc.NORM.sh scripts described above.
For each line of the distribution file acquired as input, the line is initially split using " " as separator, through the split() function, after it has been deprived of spaces at the beginning and at the end using the strip() function.
Through the Python function map(), the float() function is applied to the split line (which, starting from a number or a string containing decimal points, returns a floating point number).
The three decimal numbers obtained for each line of the distribution file are subsequently saved in the list called "distribution".
At this point, for each index in the distribution list the mean value of the bin $(i[0]+i[1])/2)$, the frequency associated with the bin $(i[2])$, the lower end of the bin $(i[0])$ and the upper

end of the bin ($i[1]$) are saved in order in the list x_y. Through the zip function all the elements of x_y and of the ordered x_y list (sorted in descending order) are associated, obtaining a tuple with the elements of the objects according to their position order.
For each i, index of the list x_y, and for each j, index of the ordered list x_y, I ask if the first element of the i-th row of the list x_y is less than the first element of the j-th row of the list x_y in decreasing order: if the answer is yes, the second and third element of the i-th row of the list x_y followed by the first element of the j-th row of the ordered list x_y (corresponding respectively to the right end of the bin and to the frequency relative to the bin, extracted from the first list, and to the left end of the bin, extracted from the second list); otherwise, the second, the third and the first element of the i-th row of the list x_y are printed (corresponding respectively to the right end of the bin, the frequency relative to the bin and the left end of the bin, extracted from the first list).

_subtract_distribution.py

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that the distribution file relating to the Indel variations (or to the SNV variations), obtained as output of the script Get_Indel_freq-distr_anyPerc.sh (or Get_SNV_freq-distr_anyPerc.NORM.sh), has been provided, followed by the output obtained by _get_mirror_distribution.py by using the distribution file relating to the Indel variations (or to the SNV variations) as input.
Once the required distribution files have been acquired as input, for each line of the distribution1, the line is initially split using " " as separator, through the split() function, after it has been deprived of spaces at the beginning and at the end using the strip() function. Through the python function map(), the float() function is applied to the split line (which, starting from a number or a string containing decimal points, returns a floating point number).
The three decimal numbers obtained for each line of the distribution file are subsequently saved in the list called $d1$. The same procedure is carried out for each line of the distribution2 file and the results are saved in the $d2$ list.
At this point, for each index in the list $d1$ the mean value of the bin $(i[0] + i[1])/2)$, the frequency associated with the bin $(i[2])$, the lower end of the bin $(i[0])$ and the upper end of the bin $(i[1])$ are saved in order in the list x_y1. The same procedure is performed for each index in the $d2$ list and the results are saved in the x_y2 list.
The lengths of the two lists just described are then calculated and a check is made on them to verify that the two distributions compared have the same size; if I do not have the same dimension, an error is returned as it is not possible to calculate the difference between the distributions provided in input.
The enumerate() function is used to iterate through all the elements of the list x_y1, having both the indices (n) and the values of the list (i) as variables to manage.
In particular, it is saved in $x1$ the first value contained in the i-th value of the x_y1 list (corresponding to the average value of the bin of the i-th row) while in $x\_y2[n][0]$ there is

the first value contained in the n-th value of the x_y2 list (corresponding to the average value of the bin of the n-th row).
At this point a check is made to verify that the bin identified in distribution1 exists in distribution2, if this is not verified an error is returned as it is not possible to calculate the difference between the distributions provided in input for that bin.
Finally, the difference between the second value contained in the i-th value of the x_y1 list (corresponding to the value of the frequency associated with the bin of the i-th row) and the second value contained in the n-th value is calculated of the x_y2 list (corresponding to the value of the frequency associated with the bin of the n-th row); the third and fourth value contained in the i-th value of the x_y1 list (corresponding to the extremes of the bin of the i-th row) are then printed, followed by the calculated difference if the difference is positive, otherwise by zero.

Get_freq-distr_anyPerc_from_distribution.NORM.sh

The algorithm proceeds similarly to what was observed in the case of Get_Indel_freq-distr_anyPerc.NORM.sh, retracing step by step all the actions described for the script.
Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that the distribution file relating to the Indel variations or to the SNV variations has been provided followed by the length of the bin, saved in $DISTR and $L respectively.
In particular, the first input requested by this script corresponds to the output in the format obtained as a result of the _get_mirror_distribution.py and _subtract_distribution.py scripts described above.
As noted above, through the use of the "awk" command, the length of the bin acquired in input is used to calculate the number of total bins contained in an interval from 0 to 100, which is saved in the variable N.
The length of the bin is also used for the definition of the start and end, corresponding respectively to the left end of the first bin and the right end of the last bin outlined.
At this point, the values corresponding to the number of bins, the start and the end are printed.
Differently from what has been seen for the Get_Indel_freq-distr_anyPerc.NORM.sh script, using the "awk" command on the $DISTR file again, for each line of the file the average value of the bin $((right\_extreme + left\_extreme)/2)$ is saved in x and the value of the variation frequency relative to bin is saved in y. At this point, x is printed y times and finally, using the gsl-histogram command it is possible to organise the obtained data in the required output format (normalised histogram).

Subtract_germline_distribution.SNV.sh

The script Subtract_germline_distribution.SNV.sh, supplied in input the VCF file relating to the metanormal (in the form metaNormal.SNV.vcf [.gz]), a name chosen as the

basename of the subsequent outputs of the script and the length of the bin, executes in succession the scripts described above in order to graphically represent the distribution relative to the metanormal, the "subtraction" distribution and the "mirror" distribution, both normalised and non-normalized.
The main body of the algorithm is viewable in the Appendix C in the form of pseudocode.

Subtract_germline_distribution.Indel.sh

The script Subtract_germline_distribution.Indel.sh, supplied in input the VCF file relating to the metanormal (in the form metaNormal.Indel.vcf [.gz]), a name chosen as the basename of the subsequent outputs of the script and the length of the bin, executes in succession the scripts described above in order to graphically represent the distribution relative to the metanormal, the "subtraction" distribution and the "mirror" distribution, both normalised and non-normalized.
The algorithm is identical to that described for Subtract_germline_distribution.SNV.sh script therefore it is not reported.

Recalibrate_HRD_D.sh

The algorithm associated with Strategy D is created by introducing some specific changes to the one seen for Strategy C.
In particular, it is possible to notice how the entire flowchart coincides almost perfectly with the one described above for Strategy C (Figure 5.24, both as regards the Get_SNV_from_freq-distr.py and Get_indel_from_freq-distr.py.
In light of the fact that some changes have been made, in order to adapt the workflow to the new strategy introduced, the new script created in bash will take the name of Recalibrate_HRD_D.sh (differentiating from that relating to Strategy C, called Recalibrate_HRD_C.sh and from that relating to Strategy A, called Recalibrate_HRD_A.sh).
Specifically, the only disparities that emerge between the two scripts made in bash are identified by:

1. In the acquisition of input files, associated with the initial section of the algorithm, it is possible to observe for both scripts that the first step is to get the name of the directory where the script is executed, saved in BIN_DIR. From this directory it is possible to go back to the Indel_median-freq_from_N-T and SNV_median-freq_from_N-T files (Table 5.2), contained in the directory itself, corresponding to the reference distribution files in the format as the GSL-histogram output.
   These respectively represent the median distribution of the SNV and Indel variations resulting from the comparison between the tumour sample and the associated normal, corresponding to the red curve in the graphs previously illustrated.
   Exporting the described files is followed by verification that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
   Specifically, for the Strategy C script, it is verified that the directory.files has been

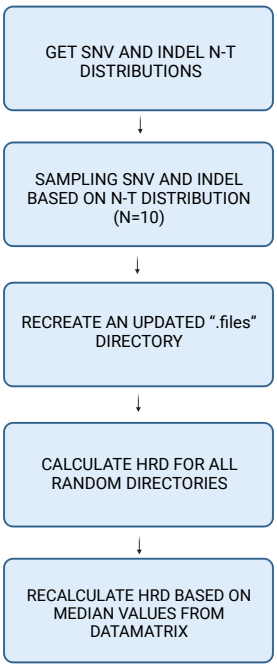

Figure 5.24. Recalibrate_HRD_D.sh core workflow

provided followed by the name of the tumour sample considered, saved respectively as FILES_DIR and TUMOUR.
On the other hand, for the script related to Strategy D, it is verified that the directory.files has been provided followed not only by the name of the tumour sample considered but also by $L_BIN, corresponding to the chosen length of the bin.

2. Always corresponding to the initial section of the script, it is possible to observe for the script of Strategy C the definition of $N = 10$ which corresponds to the definition of the number of sampling performed. In fact, the Recalibrate_HRD_C.sh script foresees the execution of the sampling on the data that describe the distribution obtained from the comparison between the tumour sample and the normal equivalent. In particular, the sub-folders of the FILES_DIR, folder provided as the first input to the script are initially defined: specifically, the two paths FILES_DIR/recalibrateC/Indel-sampling and FILES_DIR/recalibrateC/SNV-sampling are generated.
The first step is then performed inside the recalibrateC folder, separately for SNV and Indel.
In particular, the sampling_indel function (defined previously) is launched N times receiving as parameters the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.RANDOM-$i.gz (contained in the Indel-sampling folder), where "i" represents the index that is updated at each iteration proceeding from 1 to N.

Similarly, the sampling_SNV function (previously defined) is launched N times receiving as parameters the file in the form SNV.*.vcf.gz (contained in FILES_DIR) followed by the SNV.*.vcf.RANDOM-$i.gz file (contained in the folder SNV-sampling).
On the other hand, Strategy D differs from the previous one since it sets the value of N equal to 50 (five times the value chosen for Strategy C).
The sub-folders of the FILES_DIR folder, provided as the first input to the script, are initially defined: specifically, the two paths FILES_DIR/recalibrateD.$L_BIN/Indel-sampling and FILES_DIR/recalibrateD. $L_BIN/SNV-sampling are generated to differentiate it from recalibrateC, referring to the Strategy C).
I go to the recalibrateD.$L_BIN folder and the sampling_indel function is launched fifty times receiving as parameters the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.RANDOM-1.gz (contained in the Indel-sampling folder). Similarly, for the sampling_SNV function.

3. As regards the central body of the script created for Strategy D, this differs from the one created for Strategy C as it includes a further step.
   Specifically, in addition to the steps Sampling on N-T distribution, Recreate an updated '.files' directory, Calculate HRD for all RANDOM directories, Recalculate HRD based on median values from datamatrix, previously described, the "step zero" is introduced.
   In particular, the introduction of the new step involves the execution of the two scripts created in bash $BIN_DIR/Subtract_germline_distribution.SNV.sh and $BIN_DIR/Subtract_germline_distribution.Indel.sh. The latter, receiving in input respectively $FILES_DIR/SNV.*.vcf.gz, "SNV_distribution" (name chosen as base-name relative to the script outputs), $L_BIN and $FILES_DIR/Indel.*.vcf.gz, "Indel_distribution", $L_BIN, allows the exportation of the distribution (respectively named SNV_NT_DISTRIBUTION and INDEL_NT_DISTRIBUTION) resulting from the subtraction of the "mirror" distribution from the metanormal one.

5.6.2 Results

Using the data of Nik-Zainal one therefore has a score that is the expected one derived from the match between the normal and his tumour and then the result that is predicted using the metanormal.
In particular, in accordance with what has been explained for the other strategies illustrated above, score values higher than 0.70 are defined as HR deficient while score values lower than 0.30 are considered HR proficient.
All scores between these two values are indicated as doubtful, i.e. it is not possible to attribute proficiency or deficiency on the basis of the observed value.
Taking this classification into account, as can be seen in the Table 5.7, a color code was used that identified HR deficiency with red, HR proficiency with green and doubtful situations with orange, for which it was not possible to express.
Analysing the results, it is possible to note how overall Strategy D does not provide faithful results to those expected, obtained by comparing the tumour sample and the normal matched one.

In general, in fact, out of 73 samples of breast cancer patients, exactly 12 scores are different from those predicted.
In particular, it should be noted that the samples for which the expected score was doubtful were excluded from the study (4 out of 77); this because, since it is not possible to express with certainty regarding the data obtained from the comparison between the tumour sample and the normal matched one, in the same way the result obtained by comparing the tumour sample with the metanormal could not have been considered as certain.
A particularly important aspect to underline is that in this case all the samples predicted to be HR deficient by Strategy D are actually such, as expected, like what happened for Strategy C.
On the other hand, not all the samples that are predicted to be HR proficient by Strategy D are actually such, as expected: specifically 6 among the 12 erroneously predicted samples, for which HR proficiency is predicted although they are HR deficient.
It follows that common errors attributable to Strategy D (similarly to what happened for Strategy C and for Strategy A) are related to the declaration of doubtful situations for samples that are expected as HR proficient or HR deficient.
Specifically, again with reference to these cases which were incorrectly assessed, 6 out of 12 samples are indicated with the colour orange, of which 1 is HR proficient, while the remaining 5 are HR deficient.
In summary, Strategy D correctly predicts the expected score 61 times out of 73 samples considered as a whole, committing a percentage error of 16.44%, significantly higher than that identified by Strategy C and by Strategy B, but lower than that outlined by Strategy A.

5.7 Ensemble strategy

In order to improve the results obtained through the predictions made by the strategies described above, the Ensemble Strategy is introduced, which draws its origins from ensemble learning.
Ensemble learning is a general meta approach to machine learning that seeks better predictive performance by combining the predictions from multiple models[29].
Although there appears to be no limit to the number of ensembles that may be created for a predictive modelling task, the area of ensemble learning is dominated by three approaches.
Ensemble learning approaches are divided into three categories [29]:

- **Bagging**: This method seeks to generate a group of classifiers with similar relevance. Each model will vote on the prediction's result at the moment of classification, and the aggregate output will be the class with the most votes.

- **Boosting**: Unlike bagging, each classifier has a set weight on the final grade. The accuracy error that each model will incur throughout the learning phase will be used to determine this weight.

- **Stacking**: Unlike bagging, where the output is the outcome of a vote, stacking introduces a new classifier (called a meta-classifier) that leverages the predictions of other sub-models to do extra learning.

In this case, the methodology that will be used, will take its cue from the class of methods called Stacking.
Stacked Generalisation, or stacking, is an ensemble approach for finding a varied group of members by modifying the model types fit on the training data and combining predictions with a model.

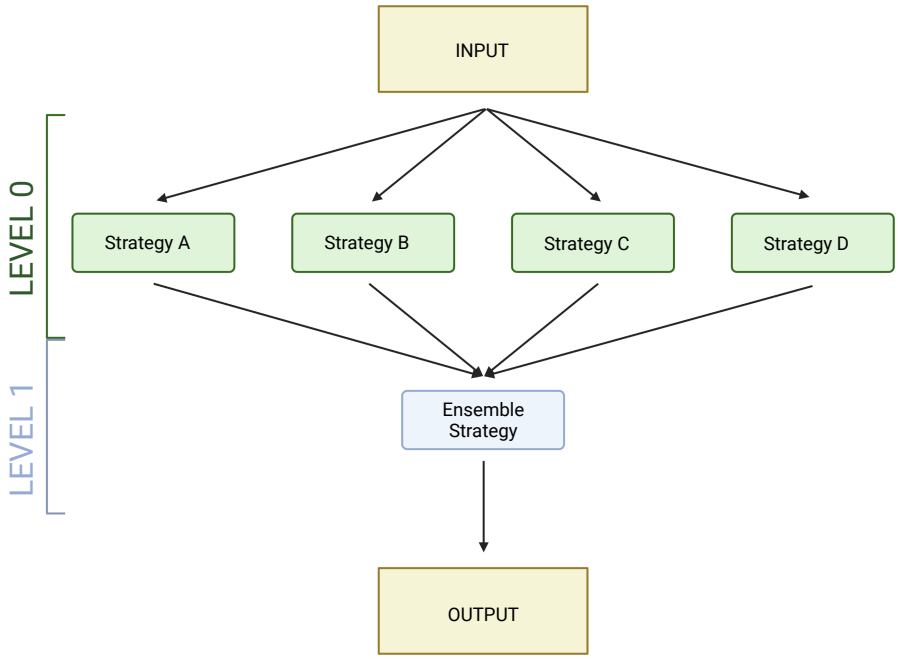

Figure 5.25. Stacking workflow

Stacking has its own terminology, with level-0 models referring to ensemble members and level-1 models referring to the model that is used to integrate the forecasts.
Although additional levels of models can be utilised, the most frequent strategy is a two-level hierarchy of models.
Instead of a single level-1 model, one may has three or five level-1 models and a single level-2 model that integrates level-1 model predictions to generate a forecast.
The essential aspects of stacking may be summarised as follows:

- The training dataset has not been altered.
- Each ensemble member has its own algorithm.
- A model for determining the optimal way to integrate forecasts.

The multiple models utilised as ensemble members provide diversity.
As a result, it's preferable to utilise a collection of models that are taught or built in a variety of methods, guaranteeing that they make various assumptions and, as a result, have less associated prediction mistakes.
Specifically, the developed strategy sees as level 0 models, i.e. as members of the ensemble, strategies C, A, D and B (as shown in Figure 5.25).
By applying these models to the same dataset, consisting of 77 samples of patients with breast cancer, it was in fact possible to obtain predictions regarding the status of the HR score. Using Nik-Zainal's data one gets a score that is the expected one derived from the match between the normal and his tumour and then the result that is predicted using the metanormal with the previously listed strategies.
In particular, in accordance with what has been explained before, score values higher than 0.70 are defined as HR deficient while score values lower than 0.30 are considered HR proficient.
All scores between 0.30 and 0.70 are indicated as doubtful, i.e. it is not possible to attribute proficiency or deficiency on the basis of the observed value.
Taking this classification into account, a color code was used that identified HR deficiency with red, HR proficiency with green and doubtful situations with orange, for which it was not possible to express. In this way it was possible to associate a color to each prediction made through each strategy, depending on the score obtained.

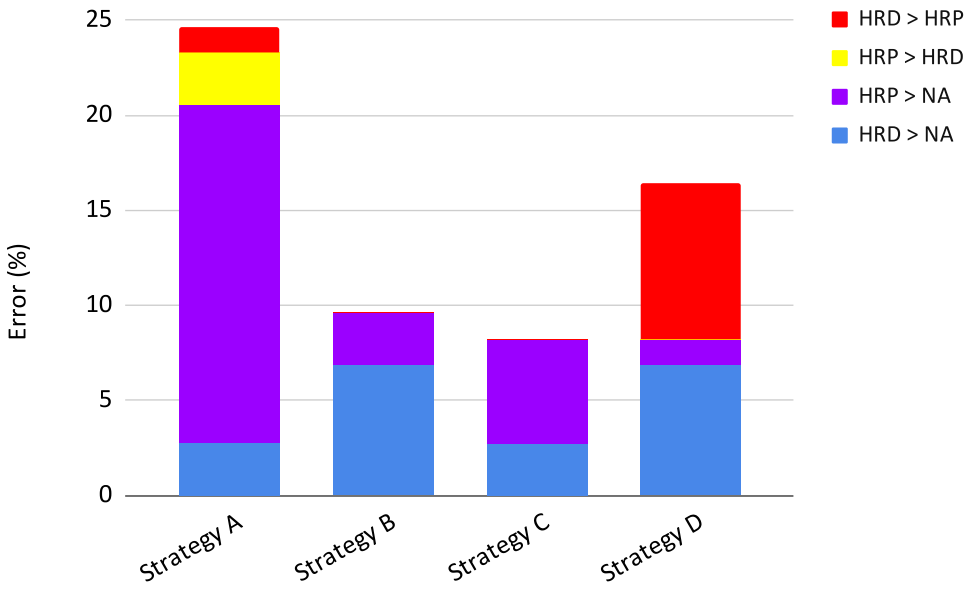

Figure 5.26. Representation of the errors outlined by the previously described strategies through application on the breast cancer dataset examined, differentiated by type of error.

Analysing the results, it is possible to note how overall Strategy C traces well the expected results, obtained by comparing the tumour sample and the normal matched one.

In general, in fact, out of 73 samples of breast cancer patients, only 6 scores are different from those predicted.
In particular, it should be noted that the samples for which the expected score was doubtful (4 out of 77) were excluded from the study; this because, since it is not possible to express with certainty regarding the data obtained from the comparison between the tumour sample and the normal matched one, in the same way the result obtained by comparing the tumour sample with the metanormal could not have been considered as certain.
A particularly important aspect to underline is that in this case all the samples that are predicted to be HR deficient by Strategy C are actually such, as expected, as could also be seen for Strategy B.
Precisely for this reason and since Strategy C is associated with the lowest percentage error (Figure 5.26), for the execution of the new strategy introduced, it was decided to use the results obtained starting from Strategy C as a reference, in order to improve the predictions made by it.
The level 1 model, used to combine the forecasts obtained through the level 0 models, therefore envisages considering Strategy C as a reference and aims to answer the question: *"Is the sample HRD ?"*, correcting the erroneous predictions that in this case are related to the declaration of doubtful situations for samples that are expected as HR proficient or HR deficient, as explained above.
The level 1 model is therefore used exclusively on the samples that have been predicted as *"not HRD"*, so on samples declared HRP or doubtful.
Specifically, the model assigns the value "1" to the HR proficiency (therefore to the green color), the value "-1" to the HR deficiency (therefore to the red color) while assigning a null value to all cases marked in orange as doubtful situations.
In this way, for each of the 50 patients considered, each strategy will be associated with a numerical value, relative to the prediction made by the strategy itself. By summing up the values for each patient separately, 50 values are obtained, which track the contributions provided by the strategies overall.
At this point the decision-making phase intervenes: if the result obtained for the single patient sample is positive, HR proficiency (green color) is assigned as prediction, if it is negative, HR deficiency (red color) is assigned otherwise (the result is null) recognizes the situation again as doubtful (indicating the corresponding sample with orange color).

5.7.1 Methods

Ensemble_strategy.sh

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input.
Specifically, it is verified that at least one directory, containing the information related to the application of the different strategies to the patient considered, has been provided.
By iterating on each directory in the form directory.files (relative to a specific sample examined) provided as input, it was possible to obtain for each of them the HR score obtained by using the Ensemble Strategy.

Specifically, through the use of the "awk" command, it was possible to extract from the files in the form HRDetect_fullPipeline.out, contained in the recalibrateB, recalibrateC, recalibrateA, recalibrateD folders, respectively, the HR scores obtained as a result of the execution of Strategies B, C, A, D on the samples of the patients considered.
As anticipated, since Strategy C is associated with the lowest percentage error, for the execution of the new strategy introduced, it was decided to use the results obtained starting from Strategy C as a reference, in order to improve the predictions made by it.
In this regard, the "*ensemble*" variable is introduced, which is associated with the HR score obtained by performing the Strategy C; in particular, the value 2, 1 or 0 to ensemble is assigned depending on whether the score is respectively >0.70, <0.30 or between 0.30 and 0.70 (extremes included).
It follows that the Ensemble Strategy will be applied only if the value contained in the ensemble variable is different from 2, which implies that the score predicted by the Strategy C is doubtful, between 0.30 and 0.70, or that the score predicted is less than 0.30 (HRP). In the other case, $ensemble = 2$, the script will simply provide the result obtained through Strategy C, used as a reference, printing that the sample is HRD.
Assuming that ensemble takes the value 0 or 1 and therefore the Ensemble Strategy comes into play, the model assigns the value "1" to the HR proficiency (therefore to score values lower than 0.30), the value "-1 "to the HR deficiency (therefore to score values higher than 0.70) while assigning a null value to all scores between 0.30 and 0.70 (extremes included), as outlined above.
In this way each strategy will be associated with a numerical value, relative to the prediction made by the strategy itself, which in this case have been named $R1$, $R3$, $R4$ (referring respectively to strategies B, A and D).
At this point, the values obtained are added and the decision phase takes place: if the result obtained for the single patient sample (ie for the current file directory) is positive, HR proficiency is assigned as prediction, if it is negative, HR deficiency is assigned, otherwise (the result is null) the situation is recognized as doubtful and assigns the value "NA" (Not Available).

5.7.2 Results

As can be seen in the Table 5.8, a color code was used that identifies HR deficiency with red, HR proficiency with green and doubtful situations with orange, for which it was not possible to express.
Analysing the results, it is possible to note how overall Ensemble Strategy traces well (although not perfectly) the expected results, obtained by adding the single values assigned for each of the previously examined strategies.
In general, in fact, out of 50 samples corresponding to breast cancer patients for whom Strategy C predicted a “non-HRD” score, only 1 score is different from those predicted.
In particular, it should be noted, as previously done, that the samples for which the expected HRD score was perfectly replicated by Strategy C were excluded from the study; this is because it has been chosen to use Strategy C as a reference.
A particularly important aspect to underline is that in this case all the samples that are predicted to be HR deficient by Ensemble Strategy are actually such, as expected.

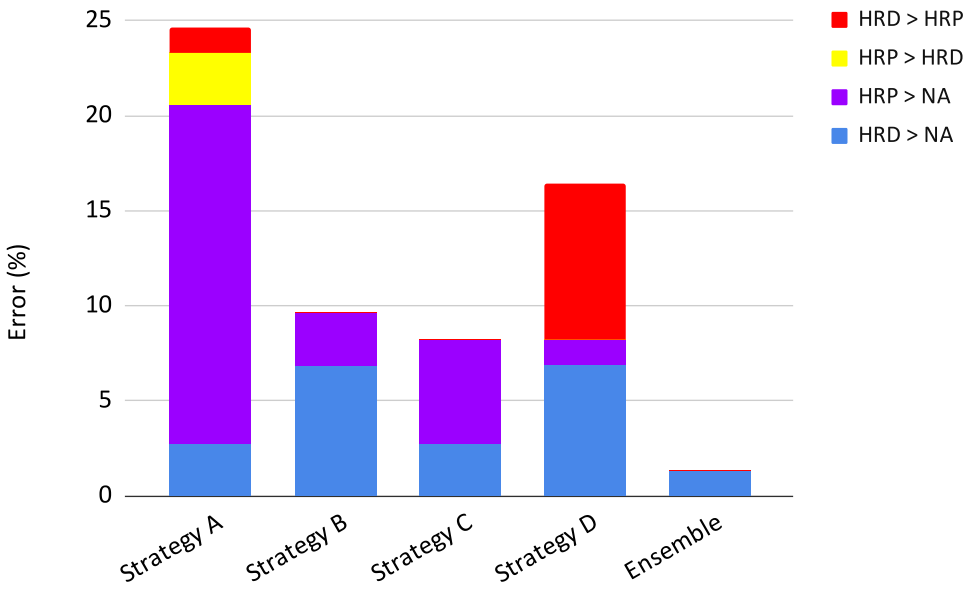

Figure 5.27. Representation of the errors outlined by the strategies through application on the breast cancer dataset examined, differentiated by type of error.

At the same time, all of the samples that are predicted to be HR proficient by Ensemble Strategy are actually such, as expected. It follows that the only error attributable to the Ensemble Strategy is related to the declaration of doubtful situation for a sample that is expected as HR deficient (specifically the sample PD22360a).
In summary, Ensemble Strategy correctly predicts the expected score in 72 of the 73 cases considered.
It follows that, by applying the new strategy introduced, it is possible to bring the percentage error committed by Strategy C from 8.22% to 1.37%, passing from 6 errors to 1 error on 73 patients examined (Figure 5.27).

Patient	Expected Score	Strategy A	Patient	Expected Score	Strategy A
PD22036a	0.000	0.396	PD22251a	0.003	0.2097
PD22355a	0.9999	1.0000	PD22358a	0.9170	0.9933
PD22359a	0.0018	0.0001	PD22360a	0.9748	0.0152
PD22361a	0.0004	0.0430	PD22362a	0.0022	0.3783
PD22363a	0.9723	0.3414	PD22364a	0.0035	0.0232
PD22365a	0.0062	0.5297	PD22366a	0.9995	0.9977
PD23550a	0.0047	0.5409	PD23554a	0.9964	0.8665
PD23558a	0.9946	0.9993	PD23559a	0.1954	0.9889
PD23561a	0.0000	0.0016	PD23562a	0.9998	0.9977
PD23563a	0.9993	0.9965	PD23564a	0.0000	0.0002
PD23565a	0.0351	0.6148	PD23566a	0.9999	0.9024
PD23567a	0.9983	0.9997	PD23570a	0.0071	0.1508
PD23577a	0.9999	0.9991	PD23578a	0.9941	0.9984
PD23579a	0.0000	0.0001	PD24182a	0.9996	0.9972
PD24186a	0.7788	0.5665	PD24189a	0.0000	0.0002
PD24190a	0.0141	0.5206	PD24191a	0.8996	0.9748
PD24193a	0.0000	0.0003	PD24195a	0.0009	0.0877
PD24196a	NA	NA	PD24197a	0.9978	0.9997
PD24199a	0.0096	0.2393	PD24200a	0.0272	0.0575
PD24201a	0.9969	0.9685	PD24202a	0.9997	0.9997
PD24204a	0.0001	0.0012	PD24205a	0.9970	0.9986
PD24206a	0.0007	0.0796	PD24207a	0.0297	0.1936
PD24208a	0.0186	0.4868	PD24209a	0.0692	0.1333
PD24212a	0.9987	0.9991	PD24214a	0.0030	0.0023
PD24215a	NA	NA	PD24216a	0.0055	0.5763
PD24217a	0.0004	0.0953	PD24218a	0.0047	0.0210
PD24219a	NA	NA	PD24220a	0.0012	0.0007
PD24221a	0.0078	0.2384	PD24223a	0.0013	0.0827
PD24224a	0.0006	0.0163	PD24225a	0.0013	0.1822
PD24302a	0.0002	0.0695	PD24303a	0.8697	1.0000
PD24304a	0.8740	0.9751	PD24306a	0.9660	1.0000
PD24307a	0.0006	0.1001	PD24308a	NA	NA
PD24314a	0.0040	0.4013	PD24318a	0.0216	0.4321
PD24320a	0.0002	0.0054	PD24322a	0.0009	0.1117
PD24325a	0.1706	0.3457	PD24326a	0.0008	0.0069
PD24327a	0.0010	0.1367	PD24329a	0.0002	0.1331
PD24332a	0.0021	0.6388	PD24333a	0.0023	0.8560
PD24335a	0.0083	0.0507	PD24336a	0.0006	0.3386
PD24337a	1.0000	0.9956			

Table 5.4. Results obtained with Strategy A for breast cancer.

Patient	Expected Score	Strategy B	Patient	Expected Score	Strategy B
PD22036a	0.000	0.211	PD22251a	0.003	0.1552
PD22355a	0.9999	0.9840	PD22358a	0.9170	0.8585
PD22359a	0.0018	0.0751	PD22360a	0.9748	0.9348
PD22361a	0.0004	0.1514	PD22362a	0.0022	0.1820
PD22363a	0.9723	0.3721	PD22364a	0.0035	0.0921
PD22365a	0.0062	0.0901	PD22366a	0.9995	0.9974
PD23550a	0.0047	0.0864	PD23554a	0.9964	0.9975
PD23558a	0.9946	0.9692	PD23559a	0.1954	0.1255
PD23561a	0.0000	0.0105	PD23562a	0.9998	0.9981
PD23563a	0.9993	0.8455	PD23564a	0.0000	0.0002
PD23565a	0.0351	0.1579	PD23566a	0.9999	0.8685
PD23567a	0.9983	0.9975	PD23570a	0.0071	0.1328
PD23577a	0.9999	0.9057	PD23578a	0.9941	0.9942
PD23579a	0.0000	0.0001	PD24182a	0.9996	0.9887
PD24186a	0.7788	0.3871	PD24189a	0.0000	0.0002
PD24190a	0.0141	0.2261	PD24191a	0.8996	0.8587
PD24193a	0.0000	0.0029	PD24195a	0.0009	0.6269
PD24196a	NA	NA	PD24197a	0.9978	0.9988
PD24199a	0.0096	0.0509	PD24200a	0.0272	0.4198
PD24201a	0.9969	0.8127	PD24202a	0.9997	0.9523
PD24204a	0.0001	0.1456	PD24205a	0.9970	0.9830
PD24206a	0.0007	0.2054	PD24207a	0.0297	0.0293
PD24208a	0.0186	0.1449	PD24209a	0.0692	0.1443
PD24212a	0.9987	0.5900	PD24214a	0.0030	0.0919
PD24215a	NA	NA	PD24216a	0.0055	0.1826
PD24217a	0.0004	0.1339	PD24218a	0.0047	0.0922
PD24219a	NA	NA	PD24220a	0.0012	0.1067
PD24221a	0.0078	0.1484	PD24223a	0.0013	0.1211
PD24224a	0.0006	0.1527	PD24225a	0.0013	0.1988
PD24302a	0.0002	0.1081	PD24303a	0.8697	0.8989
PD24304a	0.8740	0.5106	PD24306a	0.9660	0.5147
PD24307a	0.0006	0.0288	PD24308a	NA	NA
PD24314a	0.0040	0.1468	PD24318a	0.0216	0.1303
PD24320a	0.0002	0.0115	PD24322a	0.0009	0.1622
PD24325a	0.1706	0.0493	PD24326a	0.0008	0.0179
PD24327a	0.0010	0.1479	PD24329a	0.0002	0.2216
PD24332a	0.0021	0.0711	PD24333a	0.0023	0.0524
PD24335a	0.0083	0.1621	PD24336a	0.0006	0.1540
PD24337a	1.0000	0.9934			

Table 5.5. Results obtained with Strategy B for breast cancer

Patient	Expected Score	Strategy C	Patient	Expected Score	Strategy C
PD22036a	0.000	0.215	PD22251a	0.003	0.1570
PD22355a	0.9999	0.9955	PD22358a	0.9170	0.9926
PD22359a	0.0018	0.0133	PD22360a	0.9748	0.5324
PD22361a	0.0004	0.0428	PD22362a	0.0022	0.2851
PD22363a	0.9723	0.7002	PD22364a	0.0035	0.0649
PD22365a	0.0062	0.2702	PD22366a	0.9995	0.9942
PD23550a	0.0047	0.1677	PD23554a	0.9964	0.9909
PD23558a	0.9946	0.9897	PD23559a	0.1954	0.2104
PD23561a	0.0000	0.0052	PD23562a	0.9998	0.8948
PD23563a	0.9993	0.9870	PD23564a	0.0000	0.0002
PD23565a	0.0351	0.0956	PD23566a	0.9999	0.8490
PD23567a	0.9983	0.9987	PD23570a	0.0071	0.1316
PD23577a	0.9999	0.9986	PD23578a	0.9941	0.9954
PD23579a	0.0000	0.0001	PD24182a	0.9996	0.9946
PD24186a	0.7788	0.8016	PD24189a	0.0000	0.0002
PD24190a	0.0141	0.3162	PD24191a	0.8996	0.8060
PD24193a	0.0000	0.0009	PD24195a	0.0009	0.1563
PD24196a	NA	NA	PD24197a	0.9978	0.9988
PD24199a	0.0096	0.1602	PD24200a	0.0272	0.0609
PD24201a	0.9969	0.9033	PD24202a	0.9997	0.9699
PD24204a	0.0001	0.0300	PD24205a	0.9970	0.9919
PD24206a	0.0007	0.2000	PD24207a	0.0297	0.0536
PD24208a	0.0186	0.0759	PD24209a	0.0692	0.1559
PD24212a	0.9987	0.9327	PD24214a	0.0030	0.0029
PD24215a	NA	NA	PD24216a	0.0055	0.0682
PD24217a	0.0004	0.1332	PD24218a	0.0047	0.0522
PD24219a	NA	NA	PD24220a	0.0012	0.0326
PD24221a	0.0078	0.1262	PD24223a	0.0013	0.0707
PD24224a	0.0006	0.0932	PD24225a	0.0013	0.1895
PD24302a	0.0002	0.0804	PD24303a	0.8697	0.9648
PD24304a	0.8740	0.6575	PD24306a	0.9660	0.9763
PD24307a	0.0006	0.0388	PD24308a	NA	NA
PD24314a	0.0040	0.3782	PD24318a	0.0216	0.3340
PD24320a	0.0002	0.0122	PD24322a	0.0009	0.0654
PD24325a	0.1706	0.0305	PD24326a	0.0008	0.0211
PD24327a	0.0010	0.1759	PD24329a	0.0002	0.1045
PD24332a	0.0021	0.1828	PD24333a	0.0023	0.2251
PD24335a	0.0083	0.0714	PD24336a	0.0006	0.4682
PD24337a	1.0000	0.9954			

Table 5.6. Results obtained with Strategy C for breast cancer

Patient	Expected Score	Strategy D	Patient	Expected Score	Strategy D
PD22036a	0.000	0.066	PD22251a	0.003	0.0309
PD22355a	0.9999	0.9650	PD22358a	0.9170	0.1631
PD22359a	0.0018	0.0682	PD22360a	0.9748	0.4249
PD22361a	0.0004	0.0521	PD22362a	0.0022	0.0675
PD22363a	0.9723	0.1964	PD22364a	0.0035	0.0524
PD22365a	0.0062	0.0511	PD22366a	0.9995	0.8291
PD23550a	0.0047	0.0135	PD23554a	0.9964	0.7752
PD23558a	0.9946	0.2500	PD23559a	0.1954	0.0920
PD23561a	0.0000	0.0192	PD23562a	0.9998	0.9133
PD23563a	0.9993	0.6807	PD23564a	0.0000	0.0002
PD23565a	0.0351	0.0754	PD23566a	0.9999	0.8226
PD23567a	0.9983	0.8918	PD23570a	0.0071	0.0640
PD23577a	0.9999	0.8997	PD23578a	0.9941	0.7013
PD23579a	0.0000	0.0001	PD24182a	0.9996	0.6659
PD24186a	0.7788	0.1518	PD24189a	0.0000	0.0002
PD24190a	0.0141	0.0953	PD24191a	0.8996	0.0000
PD24193a	0.0000	0.0049	PD24195a	0.0009	0.2889
PD24196a	NA	NA	PD24197a	0.9978	0.9893
PD24199a	0.0096	0.0563	PD24200a	0.0272	0.2342
PD24201a	0.9969	0.7634	PD24202a	0.9997	0.8669
PD24204a	0.0001	0.0752	PD24205a	0.9970	0.4079
PD24206a	0.0007	0.1138	PD24207a	0.0297	0.0327
PD24208a	0.0186	0.0002	PD24209a	0.0692	0.1269
PD24212a	0.9987	0.2970	PD24214a	0.0030	0.0786
PD24215a	NA	NA	PD24216a	0.0055	0.1511
PD24217a	0.0004	0.0375	PD24218a	0.0047	0.0553
PD24219a	NA	NA	PD24220a	0.0012	0.0141
PD24221a	0.0078	0.3287	PD24223a	0.0013	0.1737
PD24224a	0.0006	0.0702	PD24225a	0.0013	0.2410
PD24302a	0.0002	0.0575	PD24303a	0.8697	0.9399
PD24304a	0.8740	0.8556	PD24306a	0.9660	0.3596
PD24307a	0.0006	0.0370	PD24308a	NA	NA
PD24314a	0.0040	0.0525	PD24318a	0.0216	0.0945
PD24320a	0.0002	0.0149	PD24322a	0.0009	0.0660
PD24325a	0.1706	0.0492	PD24326a	0.0008	0.0179
PD24327a	0.0010	0.0560	PD24329a	0.0002	0.1207
PD24332a	0.0021	0.0585	PD24333a	0.0023	0.0565
PD24335a	0.0083	0.0489	PD24336a	0.0006	0.0837
PD24337a	1.0000	0.8429			

Table 5.7. Results obtained with Strategy D for breast cancer.

Patient	Expected Score	Ensemble	Patient	Expected Score	Ensemble
PD22036a	0.000	HRP	PD22251a	0.003	HRP
PD22355a	0.9999	HRD	PD22358a	0.9170	HRD
PD22359a	0.0018	HRP	PD22360a	0.9748	NA
PD22361a	0.0004	HRP	PD22362a	0.0022	HRP
PD22363a	0.9723	HRD	PD22364a	0.0035	HRP
PD22365a	0.0062	HRP	PD22366a	0.9995	HRD
PD23550a	0.0047	HRP	PD23554a	0.9964	HRD
PD23558a	0.9946	HRD	PD23559a	0.1954	HRP
PD23561a	0.0000	HRP	PD23562a	0.9998	HRD
PD23563a	0.9993	HRD	PD23564a	0.0000	HRP
PD23565a	0.0351	HRP	PD23566a	0.9999	HRD
PD23567a	0.9983	HRD	PD23570a	0.0071	HRP
PD23577a	0.9999	HRD	PD23578a	0.9941	HRD
PD23579a	0.0000	HRP	PD24182a	0.9996	HRD
PD24186a	0.7788	HRD	PD24189a	0.0000	HRP
PD24190a	0.0141	HRP	PD24191a	0.8996	HRD
PD24193a	0.0000	HRP	PD24195a	0.0009	HRP
PD24196a	NA	NA	PD24197a	0.9978	HRD
PD24199a	0.0096	HRP	PD24200a	0.0272	HRP
PD24201a	0.9969	HRD	PD24202a	0.9997	HRD
PD24204a	0.0001	HRP	PD24205a	0.9970	HRD
PD24206a	0.0007	HRP	PD24207a	0.0297	HRP
PD24208a	0.0186	HRP	PD24209a	0.0692	HRP
PD24212a	0.9987	HRD	PD24214a	0.0030	HRP
PD24215a	NA	NA	PD24216a	0.0055	HRP
PD24217a	0.0004	HRP	PD24218a	0.0047	HRP
PD24219a	NA	NA	PD24220a	0.0012	HRP
PD24221a	0.0078	HRP	PD24223a	0.0013	HRP
PD24224a	0.0006	HRP	PD24225a	0.0013	HRP
PD24302a	0.0002	HRP	PD24303a	0.8697	HRD
PD24304a	0.8740	HRD	PD24306a	0.9660	HRD
PD24307a	0.0006	HRP	PD24308a	NA	NA
PD24314a	0.0040	HRP	PD24318a	0.0216	HRP
PD24320a	0.0002	HRP	PD24322a	0.0009	HRP
PD24325a	0.1706	HRP	PD24326a	0.0008	HRP
PD24327a	0.0010	HRP	PD24329a	0.0002	HRP
PD24332a	0.0021	HRP	PD24333a	0.0023	HRP
PD24335a	0.0083	HRP	PD24336a	0.0006	HRP
PD24337a	1.0000	HRD			

Table 5.8. Results obtained with Ensemble Strategy for breast cancer

Chapter 6

Applying this method to colorectal cancer patients

This chapter is intended to extend the above studies to a new type of cancer: colorectal cancer (CRC).
Initially, the chapter provides an explanation of the reasons why it is reasonable to proceed through the extension to new types of tissues, as well as a general presentation of the CRC and its role in the global disease scenario.
Consequently, the focus is on the introduction of the new data set and the presentation of the results obtained in the prediction of the HR score with CRC patients.

For complete information on the subject, one can refer to: [31],[32], [33], [34].

6.1 Introduction

Colorectal cancer is the second most prevalent cancer in women and the third most frequent cancer in men: incidence and fatality rates are around 25% lower in women than in males[33].
Every year, colorectal cancer accounts for around 10% of all diagnosed malignancies and cancer-related deaths globally.
These rates also vary by geography (Figure 6.1), with the greatest rates being seen in the most developed nations.
The global incidence of colorectal cancer is expected to rise to 25 million new cases in 2035, owing to continued growth in emerging nations[34]. These have been ascribed mostly to countrywide screening programs and greater colonoscopy uptake in general, however lifestyle and nutritional modifications may also play a role [32].
In contrast, a concerning increase in patients younger than 50 years old who present colorectal cancer, particularly rectal cancer and left-sided colon cancer, has been documented.

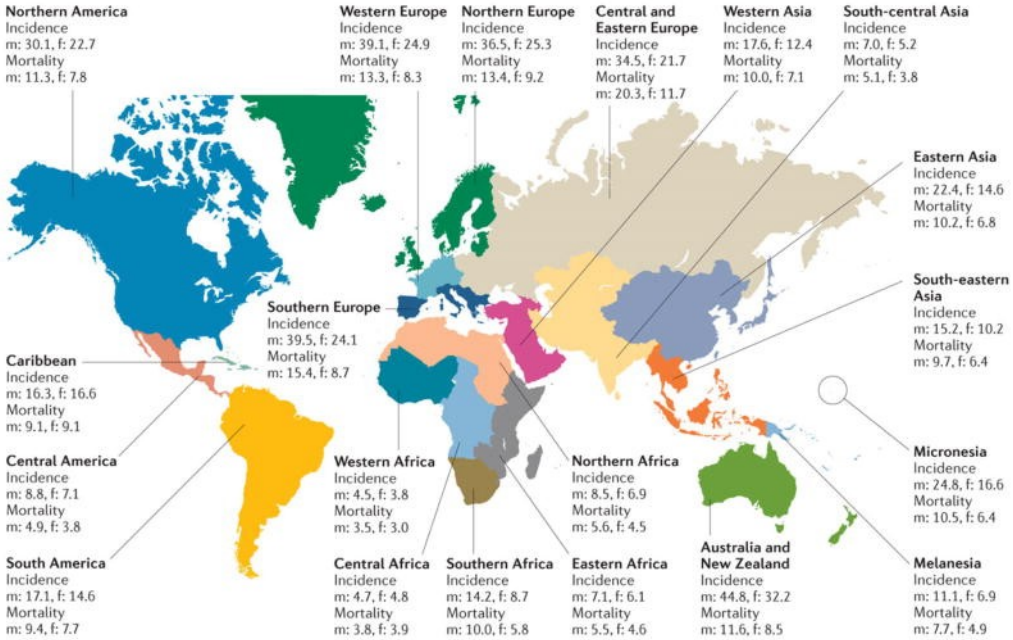

Figure 6.1. The age-standardized incidence and mortality rates in men (m) and women (f) (per 100.000 people) across geographic zones [34]

Although there may be a link between genetics, lifestyle, obesity, and environmental factors, the actual causes of this rise remain unknown.
Male sex and rising age have continuously demonstrated high relationships with illness incidence in epidemiological research.
Colorectal cancer is caused by a combination of inherited and environmental risk factors. Positive family history appears to play a role in 10–20 percent of all colorectal cancer patients, with risk variables based on the number and degree of afflicted relatives as well as the age of colorectal cancer diagnosis.
Despite the fact that multiple genome-wide association studies of colorectal cancer have effectively discovered cancer susceptibility genes linked to colorectal cancer risk, the majority of variables that cause inheritance remain unknown and need to be investigated further.
Making the diagnosis is critical because it allows the patient to develop an optimal colorectal cancer prevention strategy, as well as an optimal surveillance strategy for extracolonic cancers if applicable, optimal treatment in the event of incident colorectal cancer, and appropriate surveillance advice for relatives at risk.
To enhance the diagnosis of this hereditary disease, a comprehensive molecular examination of tumour tissue in patients of any age or a subset of those younger than 70 years is currently employed.
While CRC is curable in its early, localised stages, around 25% of people are diagnosed with the disease already metastatic. As a result, metastatic CRC (mCRC) continues to

be a significant public health issue.
Colorectal cancers are caused by pre-malignant polypoid lesions (Figure 6.2).
The multistep evolution seen at the histopathological level is driven by the ongoing accumulation of genetic and epigenetic alterations.
The high intertumour heterogeneity at the genomic level is connected to the large variability in prognosis and response to different therapies among CRC patients at all stages.
As a result, biomarkers must be developed in order to categorise patients into prognostic groups depending on the biology of each tumour.

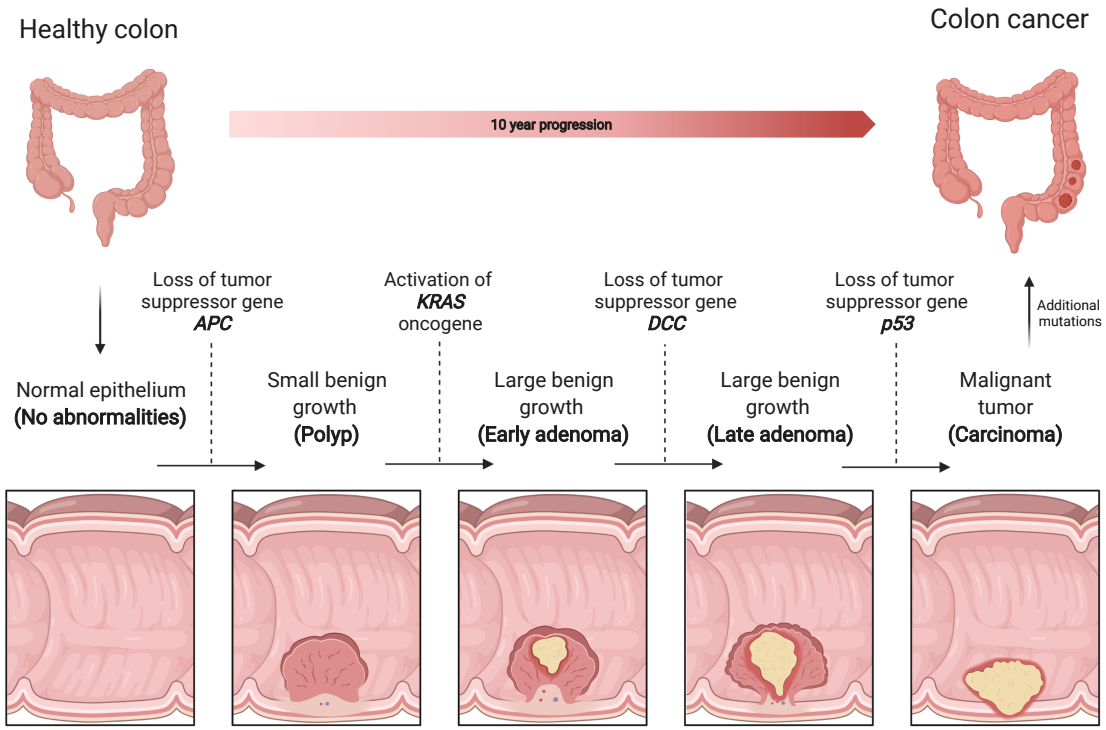

Figure 6.2. Colon cancer disease progression

In recent years, significant efforts have been undertaken to address this problem, and different genetic and epigenomic features of CRC have been disclosed, suggesting that CRC is a far more intricate disease than radiological and histological studies would suggest.
Finally, therapeutic treatment for the great majority of metastatic CRC patients is based on a "one-size-fits-all" strategy, which is impeded by a lack of thorough knowledge of each patient's biological and molecular landscape.
Preclinical knowledge of the molecular origins of CRC nature and heterogeneity is improving, paving the way for more individualized and personalised therapies.
Preclinical CRC characterization has just recently begun to inform clinical therapy of this

variable and complicated condition.
Over the last fifteen years, the treatment of metastatic colorectal cancer (mCRC) has improved thanks to the introduction of EGFR (Epidermal growth factor receptor) targeted therapy, antiangiogenic agents, and intensive triplet chemotherapy regimens based on fluoropyrimidines, oxaliplatin, and irinotecan [31].
Immune checkpoint drugs have been shown to provide long-lasting responses in a subset of roughly 5% of mCRC patients with mismatch repair deficiency (MMRd) or microsatellite instability (MSI).
Targeted therapy trials are proceeding with positive results in subgroups of molecularly selected CRCs, such as BRAF mutant or HER2 amplified cases, and these targeted combinations are anticipated to reach clinical practice in the near future.
However, median overall survival in mCRC patients has reached a plateau, ranging from 18 months in RAS mutant and right colonic tumours to 42 months in BRAF/RAS wild-type and left colonic cancers.
In conclusion, while combining and fine-tuning the use of cytotoxics, targeted treatments, and immunotherapy has improved overall survival in mCRC patients, their impact has been incremental rather than transformative.
As a result, better disease management and overall survival in mCRC patients are needed, especially for individuals who aren't candidates for targeted medicines or immunotherapy.
Furthermore, intrinsic and acquired drug resistance, as well as neuropathy associated with oxaliplatin-based regimens, limit the duration of long-term illness care and must be addressed as a clinical concern.
Molecular profiling of large CRC datasets has found subsets of CRC patients with anomalies in DNA repair pathways, and some of these cases may respond to therapeutic treatment.
BRCA1 germline pathogenic mutations are being linked to anomalies in the homologous recombination (HR) repair pathway as a risk factor for CRC, particularly early-onset CRC. Importantly, up to 15% of people have germline or somatic genetic anomalies in HR repair genes, according to current studies.
However, toxicity and a lack of patient selection have prevented the development of PARP inhibitors in CRC, either alone or in combination with other cytotoxic treatments.
These premises therefore provide interesting conditions for the development of research in this direction, motivating the extension of the analysis carried out previously on breast cancer to colorectal cancer.

6.2 Dataset description

Despite recent targeted and immunological treatment advancement, the prognosis in patients affected with stage IV or metastatic disease is still dismal.
Particularly, the subset of patients affected by BRAF V600E mutant metastatic CRC, accounting for 8-10% of all CRC, achieves overall survival most of time inferior to 20 months. Differently, around 20% of BRAF V600E mutant CRC patients survives beyond 24 months from initial diagnosis.
However, no molecular biomarkers have been so far identified capable of picking up this

peculiar subset of BRAF mutant CRC.

Patient	Expected Score
CL120464	0.007
DL261241	0.004
DLM180561	0.000
GA280852	0.000
IRCC36-A	0.007
IRCC77-B	0.014
MMA050852	0.005
PV181152	0.019
RC100851	0.746
SN020557	0.014

Table 6.1. CRC dataset description

Genetic alteration occurring in the DNA damage response (DDR) pathway represents a well-known and effective target of treatment in many cancer types but in CRC the role of these alterations is still to be addressed.
In particular, 10 BRAF V600E mutant CRC patients, of whom frozen samples were available, were identified in a cohort of BRAF mutant CRC patients treated at Niguarda Cancer Centre.
On these samples it is planned to perform Whole Genome Sequence (WGS) to assess whether a specific Homologous Recombination Deficiency (HRD) signature can be identified in CRC.
As can be seen from the Table 6.1, the results set is not absolutely balanced: referring to the expected data, that is to the values of the scores obtained by the comparison between the tumour sample and the associated normal sample, 90% of the samples taken into consideration are HR proficient, 10% of the samples are HR deficient, while no samples are identified as doubtful, not classifiable as HRP or HRD.
Specifically, only patient RC100851 falls into this 10% (indicated in red).

6.3 Results on CRC dataset

Once the new data set is introduced, featuring 10 samples from colorectal cancer patients, the same analyses performed previously for the breast cancer data set are performed on it.
In particular, for the breast cancer data set the data provided by the group of Nik-Zainal derived from the comparison between a tumour sample and the matched normal sample. On the other hand, for the new data set, the scores obtained through the pipeline described above by comparing the tumour sample and the associated normal sample are used as expected data; in fact, for each of the CRC samples taken into consideration, although it is a very limited data set, both the tumour sample and the corresponding

normal (healthy) sample are available, thus allowing to proceed with the classical analysis that requires N-T comparison.
Therefore, unlike what occurred for breast cancer samples, in this case there is no validation of the results by the Nik-Zainal group.
The other scores presented instead represent the results of the application of the various strategies to the comparison between the tumour sample and the metanormal (previously defined).
In particular, in accordance with what has been explained for the other strategies illustrated above, score values higher than 0.70 are defined as HR deficient while score values lower than 0.30 are considered HR proficient.
All scores between 0.30 and 0.70 are indicated as doubtful, i.e. it is not possible to attribute proficiency or deficiency on the basis of the observed value.
Taking this classification into account, as can be seen in the Table 6.1, a color code was used that identified HR deficiency with red, HR proficiency with green and doubtful situations with orange, for which it was not possible to express.
As can be seen from the expected scores, the available data set is not only limited but also considerably unbalanced: only one sample (specifically RC100851, highlighted in the Table 6.1) is HR deficient, while all the others are characterised by HR proficiency.
Based on the results obtained previously from the breast cancer data set, I chose to use the Ensemble Strategy to perform the HR score prediction, as it was identified as the most accurate strategy.

Patient	Expected Score	Ensemble
CL120464	0.007	HRP
DL261241	0.004	HRP
DLM180561	0.000	HRP
GA280852	0.000	HRP
IRCC36-A	0.007	HRP
IRCC77-B	0.014	NA
MMA050852	0.005	HRP
PV181152	0.019	NA
RC100851	0.746	HRD
SN020557	0.014	HRP

Table 6.2. Results obtained with Ensemble Strategy for CRC

The level 1 model, used to combine the forecasts obtained through the level 0 models, therefore envisages considering Strategy C as a reference and aims to correct its erroneous predictions, related to the declaration of doubtful situations for samples that are expected as HR proficient and to the failure in the case of particular interest identified by sample RC100851, as explained above.
The level 1 model is therefore used on the samples predicted as "non-HRD", so in this case it is used on all the samples of the data set.

Specifically, the model assigns the value "1" to the HR proficiency (therefore to the green color), the value "-1" to the HR deficiency (therefore to the red color) while assigning a null value to all cases marked in orange as doubtful situations.
In this way, for each of the 10 patients considered, each strategy will be associated with a numerical value, relative to the prediction made by the strategy itself.
By summing up the values for each patient separately, 10 values are obtained, which track the contributions provided by the strategies overall.
At this point the decision-making phase intervenes: if the result obtained for the single patient sample is positive, HR proficiency (green color) is assigned as prediction, if it is negative, HR deficiency (red color) is assigned, otherwise (the result is null) recognizes the situation again as doubtful (indicating the corresponding sample with "NA").
As can be seen from the Table 6.2, the introduction of the Ensemble Strategy allows us to correctly predict the expected score in 8 of the 10 cases considered.
In particular, it is of crucial importance to point out that the two errors can be delineated as due to "background noise", while the HR score related to the patient of interest RC100851 is correctly predicted.
Therefore, the only sample that is predicted as HR deficient by Ensemble Strategy is actually HR deficient, and the same holds true for samples that are predicted as HR deficient.
It follows that the only errors attributable to the Ensemble Strategy relate to the statement of questionable status for two samples that are predicted as HR proficient (specifically sample IRCC77_B and sample PV181152).
As a result, using the most recent strategy, it is feasible to properly estimate the HR score in 80 percent of instances, with a 20 percent error rate, equal to two patients out of ten examined.

Conclusions

A serious threat to genetic stability and cell survival could be discontinuities in one strand of the DNA double helix, known as single-strand breaks (SSBs). The importance of repairing SSBs is highlighted by the fact that these breaks could cause proliferating cells to block or collapse DNA replication forks, leading to the formation of double-strand breaks (DSBs). Cells have a remarkable ability to accurately repair such DSBs using homologous recombination (HR). Defects in the HR repair mechanism confer the so-called 'BRCAness' phenotype and can lead to genetic instability and/or cell death. Inactivation of these genes causes HR deficiency (HRD), resulting in high levels of impairment of double strand repair, a frequent driver of tumorigenesis. Recent studies have shown that the set of tumours exhibiting BRCAness and which may be selectively sensitive to PARP inhibitors is probably not limited to the small percentage of familial breast and ovarian cancers with BRCA1 or BRCA2 but extends to a larger fraction of sporadic breast and ovarian cancers and ovarian cancers, as well as other cancers.
In light of the importance of identifying a good predictor of the HR status of a tumour, Nik-Zainal's research group introduced a weighted model called HRDetect in order to accurately detect BRCA1/BRCA2-deficient samples. This type of analysis requires tumour and matched normal samples to compare and correctly extract somatic variations caused by the tumour, while discarding germline variations. The importance of overcoming this limitation, being able to correctly predict the score provided by the HRDetect algorithm in situations where the normal sample is not available (e.g. cell lines), defined the main objective of this thesis work. Through the introduction of a new "metanormal" sample it was possible to disconnect the tumour sample from the matched normal, replacing it in the comparison. As a result, since the tumour is not "matched" to its normal, the germline mutation series are not properly filtered as such appearing in the set of somatic variations extracted. In order to correctly emulate the expected HR scores, obtained from a direct comparison between the tumour sample and the matched normal sample of the patient considered, several strategies were developed. These strategies allowed increasingly accurate extraction of somatic mutations, reducing prediction errors related to the presence of germline mutations.
The application of an Ensemble strategy, which has its origins in a general meta-approach to machine learning that seeks better predictive performance by combining results from multiple models, correctly predicted HR deficiency and HR proficiency in 98.6% of the 77 breast cancer samples considered. A particularly important aspect to underline is that in this case all the samples that are predicted to be HR deficient or HR proficient by

Ensemble are actually such, as expected. It follows that the only error attributable to the strategy is related to the declaration of doubtful situation for a sample that is expected as HR deficient. In summary, my latest strategy correctly predicts the expected score in 72 of the 73 cases considered, lowering the percentage of error to 1.37%.
The consistency of these results and the importance of the development of PARP inhibitors in CRC, motivated the extension of the analysis previously performed on breast to colorectal cancer samples. For each of the samples of the new dataset, both the tumour sample and the corresponding normal (healthy) sample are available, thus allowing to proceed with the classical analysis that requires N-T comparison, whose scores became the expected data. The other scores presented instead represent the results of the application of the various strategies to the comparison between the tumour sample and the metanormal.
The introduction of the Ensemble Strategy in this case correctly predicts the expected score in 8 of the 10 cases considered: the only sample that is predicted as HR deficient by the strategy is actually HR deficient, and the same holds true for samples that are predicted as HR deficient. These results therefore show that with this approach it is indeed possible to correctly predict the HRDetect score even in those particular situations where the normal sample is missing, making it interesting to extend the analysis to other areas. This demonstrates the potential diagnostic value for the stratification of patients towards treatment with, for example, ADP-ribose polymerase inhibitors (PARPi), contributing significantly to biomedical research.
The possible improvement of the proposed analysis opens doors in terms of exploration of historical and existing clinical studies.
First of all, given the availability of one of the largest academic bank on CRC at Candiolo IRCCS, it would be interesting to extend the application on colorectal cancer cell lines, for which a tumour sample can be extracted but it is not possible to have the normal sample matched, perfectly suiting to the purpose identified in this thesis work.
Furthermore, another aspect of great interest would be that of being able to predict the HR score starting from the comparison between the normal sample of a patient and the associated tumour sample preserved in formalin. Formalin fixation and paraffin embedding (FFPE) of patient material remains standard practice in clinical pathology labs around the world. With regard to this, the pathology archive of any large hospital is likely to contain tens of thousands of FFPE blocks, so enabling accurate genomic analysis of FFPE material would unlock the enormous translational research potential of these vast collections of archival material.

Future Perspectives

In this chapter the attention is turned to some of the future perspectives that can be identified regarding the studies exposed in the previous chapters.
On the one hand the focus is on colorectal cancer cell lines, with regard to their biological importance due to the large availability of data.
On the other hand the study turns to the samples fixed in formalin, as objects of new predictions.

For more detailed information regarding these aspects, you can refer to: [35],[36].

Exploiting HRDetect score on CRC cell lines bank

Although advances in genomics during the last decade have opened new avenues for translational research and allowed the direct evaluation of clinical samples, there is still a need for reliable preclinical models to test therapeutic strategies.
Human cancer-derived cell lines [35] are the most widely used models to study the biology of cancer and to test hypotheses to improve the efficacy of cancer treatment.
On the other hand, given the availability of one of the largest academic banks on CRC cell lines at Candiolo IRCCS, it is of great interest to focus the attention on these latter.
Specifically, a cell line is defined as a clonal, genetically stable population of neoplastic or normal cells capable of proliferating in vitro.
While neoplastic derived cells are able to proliferate indefinitely, normal cells undergo a phenomenon of clonal senescence and after a defined number of cell divisions lose the ability to divide.
In this case the focus is on cancer cell lines, for which therefore, by definition, it will be possible to extract a tumour sample but it will not be possible to have the normal sample matched.
This aspect is perfectly suited to the purpose identified for this thesis work, aimed at determining the HR score of a specific sample from a tumour sample available. In this regard, the study related to the prediction of the HR score for CRC cell lines can be identified as a future perspective.
Similarly, the prediction can be associated with the study of sensitivity to treatment with PARP inhibitors, since it is possible to test treatments more easily and feasibly on cell lines than on mice or patients, introducing a translational face in this research.

Predicting BRCAness on Formalin Fixation and Paraffin Embedding samples

In order to contextualise the content of this paragraph, it is important to go over some of the steps previously discussed.
Once the raw data of the HRDetect paper have been used to try to reproduce the same scores published by the Nik-Zainal group, obtained by mapping on an old version of the reference genome (hg19) using bwa aln (Burrows-Wheeler aligner or BWA, v0.5.9), it was tried to proceed forward by remapping the same data differently (Figure 6.3).
In particular, as described before, it was decided to use a more recent version of the reference genome (currently the most advanced is indicated as hg38) used routinely, as well as to carry out the remapping by means of a different mapper (bwa mem), recreating the situation in which the research group of the Candiolo IRCCS generally works.
As already mentioned, the pipeline used by Nik-Zainal's group, indicated as cgpwgs pipeline, requires two alignment files (in BAM format) as input: one relating to the tumour tissue of the patient under examination and one relating to the corresponding normal tissue.
In particular, for this type of analysis the match between normal and tumour sample is necessary in order to be able to make the comparison and extract only the somatic variations due to the tumour, discarding the typical variations of the patient, i.e. the germinal ones.
Consequently, the next step was to identify a way in which it was possible to make the prediction of HR status without the patient's normal match (Figure 6.3).
To implement this, what has taken the name of "metanormal" was constructed: a normal sample resulting from the "sum" of a good number of sequences of normal tissues, which presents characteristics such as to be able to be used as input of the workflow previously described.

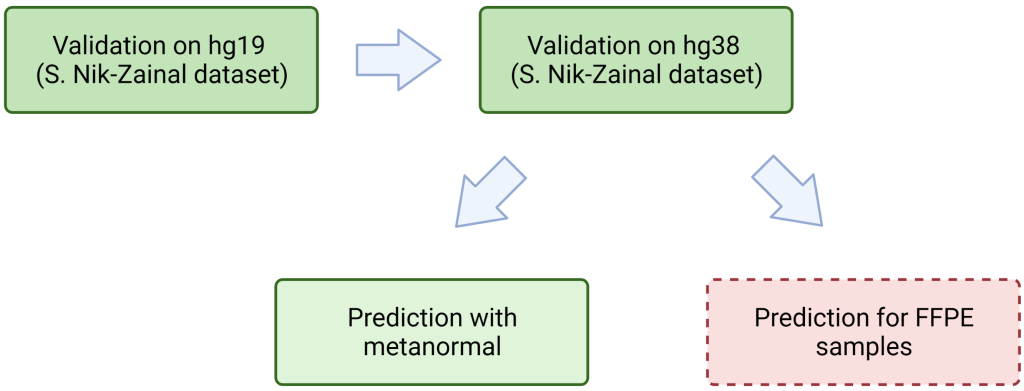

Figure 6.3. Representation of the project roadmap

Another aspect of great interest in this regard would be that of being able to predict the HR score starting from the comparison between the normal sample of a patient and the associated tumour sample preserved in formalin (Figure 6.3).
Formalin fixation and paraffin embedding (FFPE)[36] of patient material remains standard practice in clinical pathology labs around the world.
This technique preserves tissue morphology and enables immunohistochemical analysis for clinical diagnosis.
However, genomic analysis of DNA extracted from FFPE blocks is problematic, because formalin fixation negatively impacts DNA quantity and quality compared to fresh frozen (FF) material.
In fact, formalin is a recognized mutagen and sequencing of DNA derived from FFPE material is known to be riddled with artifactual mutations. The FFPE signature is dominated by C>T transitions caused by cytosine deamination, and has very high similarity to COSMIC signature SBS30 (base excision repair deficiency due to inactivation mutations in NTHL1, Figure 6.4).
Further, it can be shown that chemical repair of formalin-induced DNA lesions, a process that is routinely performed as part of sequencing library preparation, leads to a signature highly similar to COSMIC signature SBS1 (spontaneous deamination of methylated cytosine, Figure 6.4).
On the other hand, the pathology archive of any large hospital is likely to contain tens of thousands of FFPE blocks, so enabling accurate genomic analysis of FFPE material would unlock the tremendous translational research potential of these vast collections of archival material.

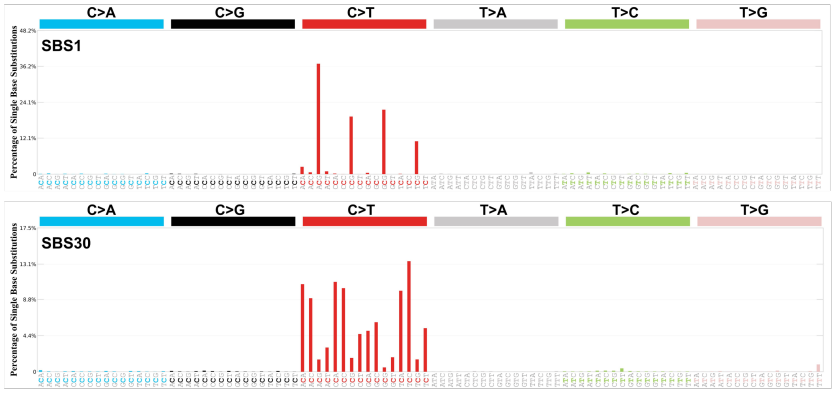

Figure 6.4. Representation of the mutational signatures SBS1 (above) and SBS30 (below) [20]

A further step forward could be that of being able to predict the HR score starting from the comparison between the tumour sample of a patient preserved in formalin and the metanormal, bypassing the need to have both the normal sample and the fresh tumour sample. The possibility to use these very numerous samples could have wider impact, accelerating a broad spectrum of oncological research.

Appendix A

Description of file formats

This chapter is aimed at presenting some of the main file formats used for the manipulation of biological data from a bioinformatics point of view.
Specifically, a detailed description of the structure of the FASTQ, BAM, SAM and VCF file formats is provided, for which some useful examples are given.

For complete information on the subject, one can refer to: [37], [38], [39]

A.1 The FASTQ format

FASTQ format is a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores [37]. Both the sequence letter and quality score are each encoded with a single ASCII character for brevity.
It has recently become the de facto standard for storing the output of high-throughput sequencing instruments. A FASTQ file normally uses four lines per sequence:

- *Line 1* begins with a '@' character and is followed by a sequence identifier and an optional description.
- *Line 2* is the raw sequence letters.
- *Line 3* begins with a '+' character and is optionally followed by the same sequence identifier (and any description) again.
- *Line 4* encodes the quality values for the sequence in Line 2, and must contain the same number of symbols as letters in the sequence.

A FASTQ file containing a single sequence might look like this:

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
```

```
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

In this context it can be helpful to introduce some important definitions [30]:

- the "***read***": a small synthesis DNA sequence obtained from a sequencing reaction.
- the "***read depth***": the number of reads that cover (map) each single base.
  From this data it is possible to calculate:
  - the "***total read depth***": the sum of all the depths across a particular coordinate range such as a gene or a peak. This is NOT exactly the same as [read length] $\times$ [num reads, i.e. fastq lines $\div$ 4] $\div$ [length of reference in bp] because not all reads in the fastq will get aligned or completely aligned.
  - the "***average read depth***": [Total Read Depth] $\div$ [number of base pairs (coordinates) in aligned region or reference]
- the "***coverage***": the sum of all the depths across a particular coordinate range such as a gene or a peak.

A.2 The BAM format

All available information relating to the samples studied are contained in files saved in the BAM format. *Binary Alignment Map* (BAM) [38] is the comprehensive raw data of genome sequencing; it consists of the lossless, compressed binary representation of the *Sequence Alignment Map* (SAM).
SAM is a text-based format originally for storing biological sequences aligned to a reference sequence developed by Heng Li and Bob Handsaker et al [40]. The overall TAB-delimited version of the format is widely used for storing data, such as nucleotide sequences, generated by next generation sequencing technologies.
SAM files can be analysed and edited with the software SAMtools.
The SAM format consists of a Header and an Alignment section: the Header section must be prior to the alignment section if it is present in particular headings begin with the '@' symbol, which distinguishes them from the alignment section; Alignment sections have 11 mandatory fields, as well as a variable number of optional fields (shown in Table A.1).
The meaning of each record in the header section is specified by a two-letter code that immediately follows the symbol '@'; each contains a list of attributes (separated by tabs), where each attribute follows the format

$$TAG : VALUE$$

- TAG = a string of two characters;
- $VALUE$ = value of the attribute).

Specifically:

Col	Field	Type	Brief description
1	QNAME	String	Query template NAME
2	FLAG	Int	bitwise FLAG
3	RNAME	String	References sequence NAME
4	POS	Int	1- based leftmost mapping POSition
5	MAPQ	Int	MAPping Quality
6	CIGAR	String	CIGAR string
7	RNEXT	String	Ref. name of the mate/next read
8	PNEXT	Int	Position of the mate/next read
9	TLEN	Int	observed Template LENgth
10	SEQ	String	segment SEQuence
11	QUAL	String	ASCII of Phred-scaled base QUALity+33

Table A.1. Alignment section of the SAM format

- The @*HD* record represents the header of the Header Section

```
@HD VN:[version] SO:[sorting]
```

- Each @*SQ* record represents a reference sequence

```
@SQ SN:[ref_name] LN:[length]
```

  The @SQ record provides for other (optional) attributes including SP, the whose value is the name of the species of the reference.

- Each @*RG* record represents a group of reads

```
@RG ID:[group_id] SM:[sample]
```

  Other optional attributes include:

  – LB:[library]
  – DS:[description]
  – PU:[platform_unit]
  – DT:[date]

- Each @*PG* record represents (alignment) software

```
@PG ID:[program_id] PN:[program_name]
```

Other optional attributes include:

- VN:[program_version]
- CL:[command_line]

Example Header Lines:

```
@HD VN:1.0 SO:coordinate
@SQ SN:1 LN:249250621 AS:NCBI37 UR:file:/data/ref/human_g1k_v37.fasta
    M5:1b22b98cdeb4a9304cb5d48026a85128
@SQ SN:2 LN:243199373 AS:NCBI37 UR:file:/data/ref/human_g1k_v37.fasta
    M5:a0d9851da00400dec1098a9255ac712e
@SQ SN:3 LN:198022430 AS:NCBI37 UR:file:/data/ref/human_g1k_v37.fasta
    M5:fdfd811849cc2fadebc929bb925902e5
@RG ID:UM0098:1 PL:ILLUMINA PU:HWUSI-EAS1707-615LHAAXX-L001 LB:80
    DT:2010-05-05T20:00:00-0400 SM:SD37743 CN:UMCORE
@RG ID:UM0098:2 PL:ILLUMINA PU:HWUSI-EAS1707-615LHAAXX-L002 LB:80
    DT:2010-05-05T20:00:00-0400 SM:SD37743 CN:UMCORE
@PG ID:bwa VN:0.5.4
@PG ID:GATK TableRecalibration VN:1.0.3471
    CL:Covariates=[ReadGroupCovariate, QualityScoreCovariate,
    CycleCovariate, DinucCovariate, TileCovariate],
    default_read_group=null, default_platform=null, force_read_group=null,
    force_platform=null, solid_recal_mode=SET_Q_ZERO,
    window_size_nqs=5, homopolymer_nback=7, exception_if_no_tile=false,
    ignore_nocall_colorspace=false, pQ=5, maxQ=40, smoothing=1
```

Among the 11 mandatory fields described in the alignment sections of the SAM format files, the ones of most interest are:

- **FLAG** : 16-bit integer to be interpreted as a string of flags 0|1; every flag has a precise meaning (for example if the read has been reverse&complemented or if the alignment is primary)
- **RNAME** (String): identifier of the reference sequence (or reference) to which the read is aligned
- **POS** (Int): position (on the reference) where the alignment starts
- **MAPQ** (Int): quality of the alignment equal to $-10 \cdot \log_{10}(p)$, where p is the probability that the alignment position is wrong

- **CIGAR** (String): defines the operations that provide the alignment of the query with respect to the reference. The operations operations are encoded in uppercase letters:

  M: match/mismatch

  I: insertion in the reference

  D: deletion in the reference

  N: insertion in the reference due to spliced alignment

  S: soft clipping (of the query sequence)

  H: hard clipping (of the query sequence)

  P: silent "deletion" (padding)

- **SEQ** (String): query sequence (or query), sequence produced in a sequencing experiment (also called read).

Example Alignments:
This is what the Alignment section of a SAM file looks like:

```
1:497:R:-272+13M17D24M 113 1 497 37 37M 15 100338662 0
CGGGTCTGACCTGAGGAGAACTGTGCTCCGCCTTCAG 0 ;==-==9; >>>>>=>>>>>>>>>>>=>>>>>>>>>>
XT:A:U NM:i:0  SM:i:37  AM:i:0  X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0
MD:Z:37

19:20389:F:275+18M2D19M 99 1 17644 0 37M = 17919 314
TATGACTGCTAATAATACCTACACATGTTAGAACCAT >>>>>>>>>>>>>>>>>>>><<>>><<>>4::>>:<9
RG:Z:UM0098:1 XT:A:R NM:i:0 SM:i:0 AM:i:0 X0:i:4 X1:i:0 XM:i:0 XO:i:0 XG:i:0
MD:Z:37

19:20389:F:275+18M2D19M 147 1 17919 0 18M2D19M = 17644 -314
GTAGTACCAACTGTAAGTCCTTATCTTCATACTTTGT ;44999;499<8<8<<<8<<><<<<<><7<;<<<>><<
XT:A:R NM:i:2 SM:i:0 AM:i:0 X0:i:4 X1:i:0 XM:i:0 XO:i:1 XG:i:2 MD:Z:18^CA19

9:21597+10M2I25M:R:-209 83 1 21678 0 8M2I27M = 21469 -244
CACCACATCACATATACCAAGCCTGGCTGTGTCTTCT <;9<<5><<<<><<<><><><><9><><>>9>>><>
XT:A:R NM:i:2 SM:i:0 AM:i:0 X0:i:5 X1:i:0 XM:i:0 XO:i:1 XG:i:2 MD:Z:35
```

A.3 The VCF format

The *Variant Call Format* (VCF) [39] is a tab-separated text file format used in bioinformatics to collect genomic variants and is currently at version 4.3. The format allows to describe in tabular format the most common genomic variants of a genome, together with

the possibility of inserting annotations and metadata.
Several tools and software libraries have also been developed to be able to manipulate the format. The format includes a header, whose lines are identified by the "##" symbols and a body which lists all the variants. Inside the body, the information relating to each variant is organised according to 9 mandatory columns (shown in Table A.3).
The 9 mandatory columns are identified by the "#" symbol.

1. **CHROM** - chromosome on which the variant is present (e.g. chr1 or 1)
2. **POS** - position on which the variant call was made.
3. **ID** - identification of the variant. If absent, the symbol is read.
4. **REF** - reference allele present on the specified position.
5. **ALT** - allele or list of alternative alleles.
6. **QUAL** - quality score of the reading of the alternative allele.
7. **FILTER** - result or filters with which the variant was selected.
8. **INFO** - list of annotations relating to the variant defined by a pair <key> = [, value].
9. **FORMAT** - list of annotations relating to the relationship of each variant with each sample, therefore concerning the genotype.

Furthermore, there are a number of columns equal to the number of samples, in which the value of the annotations present in the FORMAT column is reported. The VCF format supports both single-sample and multiple-sample variant calling.

Example:
This is what the header section of a VCF file looks like:

```
##fileformat=VCFv4.3
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
```

```
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS ID  REF ALT QUAL FILTER  INFO FORMAT   NA00001
```

$CHROM$	POS	ID	REF	ALT	$QUAL$	$FILTER$
20	14370	$rs6054257$	G	A	29	$PASS$
20	17330	.	T	A	3	$q10$
20	1110696	$rs6040355$	A	G,T	67	$PASS$
20	1230237	.	T	.	47	$PASS$
20	1234567	$microsat1$	GTC	$G,GTCT$	50	$PASS$

$INFO$	$FORMAT$	$NA00001$
$NS = 3; DP = 14; AF = 0.5; DB; H2$	$GT : GQ : DP : HQ$	0|0 : 48 : 1 : 51,51
$NS = 3; DP = 11; AF = 0.017$	$GT : GQ : DP : HQ$	0|0 : 49 : 3 : 58,50
$NS = 2; DP = 10; AF = 0.333,0.667$	$GT : GQ : DP : HQ$	1|2 : 21 : 6 : 23,27
$NS = 3; DP = 13; AA = T$	$GT : GQ : DP : HQ$	0|0 : 54 : 7 : 56,60
$NS = 3; DP = 9; AA = G$	$GT : GQ : DP$	0|1 : 35 : 4

Table A.2. Body of the VCF format

Appendix B

Model description

The aim of this chapter is to present the model that led to the development of the HRDetect algorithm.
Specifically, the Lasso logistic regression model is introduced in order to distinguish samples subject to BRCA1/BRCA2 deficiency, translating this peculiarity into the coefficients of the regression itself.

For further information about the model pursued, please refer to : [21]

B.1 Lasso logistic regression modeling

The purpose of Nik-Zainal's group was to devise a method for detecting genetic features connected to BRCAness that could be used to determine the chance that a tumour sample was HR deficient at some time in its history. The method, which was based on whole genome sequencing, classified the number of copies based on the signatures of single base substitutions, indels, and rearrangements, as well as HRD indexes [21]. It was possible to detect BRCA-proficient samples by manually analysing the genomic graphs of overall mutation patterns, assuming that BRCA1/BRCA2-proficient malignancies are frequently described as almost genomically stable and quiescent in the mutational profile.
The algorithm requires the following inputs:

1. the counts of the mutations associated with each signature of the single-base substitutions: signatures 1,2, 3, 5, 6, 8,13, 17, 18, 20 and 26;
2. indel with microhomology at the junction of the indel breaking point, indel in the polynucleotide repeat tracts and other complex indel as proportions;
3. counts of rearrangements associated with each signature of the RS1-RS6 rearrangements;

4. HRD index.

Because some samples had a significantly larger number of substitutions than others, the genomic features had to be log transformed, specifically according to the formula:

$$x' = \ln(x+1)$$

Similarly, the converted data was standardised since the ranges of values for each mutation class were so dissimilar. As a result, each feature has a zero mean and unit standard deviation, allowing them to be compared to one another.

$$x'' = \frac{x' - mean(x')}{s.d.(x')}$$

To identify samples from individuals who did not have BRCA1 or BRCA2 deficient from those who did, a lasso logistic regression model was applied. The logistic model, often known as the *logit* model, is a nonlinear regression model used in statistics when the dependent variable is dichotomous.
The model's purpose is to determine the likelihood that an observation will yield one of the dependent variable's values.
A regression model with a dichotomous dependent variable, i.e. a variable with only two possible values of 0 and 1, determines the chance that this variable will acquire value 1. Because the probabilities are defined as being between $[0, 1]$, a linear regression model would not be acceptable; in fact, it would yield values from the whole set $\mathbb{R}$.
Take, for example, the following linear model:

$$Pr(Y = 1|X = x) = \beta_0 + \beta_1 X$$

where:

- Pr indicates the probability;
- Y is the dichotomous dependent variable
- X is the vector of independent variables or regressors $X_1 ... X_k$
- β is the vector of parameters $\beta_0 ... \beta_k$

Through variable selection, the lasso technique allows for the learning and weighting of the genetic information most significant to predicting BRCA1/BRCA2 status. The coefficients learnt from the genetic features of the BRCA1/BRCA2-proficient and BRCA1/BRCA2-deficient samples supplied to the algorithm are represented by the letter β.
It is possible to obtain the optimal values assumed by the coefficients minimising the function:

$$\min_{((\beta_0,\beta))\in\mathbb{R}^{p+1}} \left(-\left[\frac{1}{N} \sum_{i=1}^{N} y_i \cdot (\beta_0 + x_i^T \beta) - \log\left(1 + e^{\beta_0 + x_i^T \beta}\right) \right] + \lambda \|\beta\|_1 \right.$$

where:

- y_i is the BRCA status of a sample: $y_i = 1$ for BRCA1/BRCA2-null samples, $y_i = 0$ otherwise;
- β_0 is the intercept, interpreted as the log of odds of $y_i = 1$ when $x_i^T = 0$;
- β is a vector of weights, each corresponding to a genomic feature (as shown in Table B.1);
- p is the number of features characterizing each sample;
- N is the number of samples; x_i^T is the vector of features characterizing the ith sample;
- λ is the penalty promoting the sparseness of the weights, as learned through nested cross-validation;
- $\lambda\|\beta\|_1$ is the L_1 norm of the vector of weights (i.e., the sum of the absolute values of all entries of the coefficient vector)

Because positive weights reflect the biological presence of mutational processes due to BRCA1/BRCA2 deficiency, all samples are evaluated for the presence of relevant mutational signatures associated with BRCA1/BRCA2 deficiency, regardless of whether these signatures were the dominant mutational process in cancer. This ensures that some mutational processes that may be overshadowed by hypermutating mutational phenotypes in specific types of malignancies are recognized. Finally, the lasso logistic regression model is used to assign a probabilistic score to any new sample that is being analyzed, using the normalized exposures of mutational processes in the sample (x_i^T) and applying the parameters of the model (β) as follows:

$$P(C_i = BRCA) = \frac{1}{1 + e^{-(\beta_0 + x_i^T \beta)}}$$

where:

- C_i is the variable encoding the status of the ith sample;
- β_0 is the intercept weight;
- x_i^T is the vector encoding features of the ith sample;
- β is the vector of weights (as shown in Table B.1).

Genomic feature name	Feature label	Weight
Intercept	intercept	-13.52352584
Proportion of deletions at microhomology	del.mh.prop	5.889097823
HRD index	hrd	1.752273557
Substitution Signatures 3	e.3	1.721594565
Rearrangement Signature 3 (RS3)	SV3	1.285002819
Rearrangement Signature 5 (RS5)	SV5	0.38123313
Rearrangement Signature 1 (RS1)	SV1	0
Rearrangement Signature 2 (RS2)	SV2	0
Rearrangement Signature 4 (RS4)	SV4	0
Rearrangement Signature 6 (RS6)	SV6	0
Substitution Signatures 1	e.1	0
Substitution Signatures 2	e.2	0
Substitution Signatures 5	e.5	0
Substitution Signatures 6	e.6	0
Substitution Signatures 8	e.8	0
Substitution Signatures 13	e.13	0
Substitution Signatures 17	e.17	0
Substitution Signatures 18	e.18	0
Substitution Signatures 20	e.20	0
Substitution Signatures 26	e.26	0
Insertions	ins	0
Proportion of deletions at repeats	del.rep.prop	0
Proportion of deletion not at microhomology or repeats	del.none.prop	0

Table B.1. Weights from Lasso logistic regression learning phase, obtained by using 22 previously known BRCA1 and BRCA2 germline null samples

Appendix C

Code Implementation

This chapter is aimed at illustrating all the algorithms used in the course of the study to elaborate the different strategies.
In particular, the presentation of each code is accompanied by comments as well as descriptive sections that deal in detail with each step of the workflow.

For more information on the programming languages used, please refer to: [41],[42],[43]

C.1 Introduction

The writing of the different scripts presented has been carried out mainly through the use of the Linux shell.
Specifically, through the Bash environment (acronym for bourne again shell), which represents a textual shell of the GNU project used in Unix and Unix-like operating systems, such as GNU/Linux [41].
Bash is a command interpreter that allows the user to interface with its operating system using a set of preset functions, as well as run applications and scripts.
It can execute instructions that are supplied to it, and it can use input and output redirection to cascade many programs in a software pipeline, passing the previous command's result as input to the next command.
It also comes with a simple native scripting language that allows you to accomplish more complicated operations by using variables, functions, and flow control structures in addition to collecting a sequence of instructions in a script.
These features determined the choice of the Bash shell for the realisation of the algorithms of this thesis work.
On the other hand, the dynamism, simplicity and flexibility of the Python language [43] meant that this programming language was used to carry out certain operations within the algorithms that outline the different strategies, in order to improve their readability and comprehension.
In addition, the use of the interpreted programming language "AWK" [42] allowed the simple manipulation of textual data, both in the form of files and data streams from

standard input.

C.2 Get_indel_from_freq-distr.py

- Definition of functions

```
def get_FORMAT_ID(VCF_row, ID):
        FORMAT_string = VCF_row[8]  #extraction of the field relating to the FORMAT
        TUMOR = VCF_row[10] #extraction of the field relating to the tumor sample
        idx = FORMAT_string.split(':').index(ID) #identification of the position of ID in the FORMAT
        return TUMOR.split(':')[idx] #extraction of the value in the tumor sample in idx position
```

As previously introduced the VCF format includes a header, whose lines are identified by the "##" symbols and a body which lists all the variants. In particular, one of the 9 mandatory columns identified by the "#" symbol is represented by the FORMAT, i.e. the list of annotations relating to the relationship of each variant with each sample, therefore concerning the genotype. Furthermore, there are a number of columns equal to the number of samples, in which the value of the annotations present in the FORMAT column is reported.
The function get_FORMAT_ID, given a row of the VCF file and the identification name of the specific field of the FORMAT (corresponding to ID), has the aim of extracting the value contained in the column relating to the tumour sample in the position corresponding to ID in the FORMAT.
In particular, if the VCF line considered were the following:

```
CHROM  POS    ID         REF   ALT    QUAL  FILTER   INFO
20     14370  rs6054257  G     A      29    PASS     NS=3;DP=14;AF=0.5;
FORMAT       NA00001          NA00002          NA00003
GT:GQ:DP:HQ  0|0:48:1:51,51   1|0:48:8:51,51   1/1:43:5
```

calling the get_FORMAT_ID (VCF_row, "GQ") function would return "48" for the "NA00002" sample.

```
def parse_VCF(fd):
        for row in fd:  #iteration over each line of the VCF file
                if row[0] == '#':  #exclusion of all lines preceded by the symbol '#' corresponding to the header and description of the columns
                        continue
                row = row.strip().split('\t') #splitting of the entire line into strings using the tab as a separator
                support = int(get_FORMAT_ID(row, "FC")) #extraction of the value in the tumor sample corresponding to the position of "FC" in the FORMAT
```

```
7                 depth = int(get_FORMAT_ID(row, "FD")) #extraction of
    the value in the tumor sample corresponding to the position of
    "FD" in the FORMAT
8                 VAF = 100*support/depth  #calculation of the variant
    allele frequency
9     yield (VAF, row)  #extraction of the VAF value followed by the
    respective VCF row
```

To extract the frequencies of Indel variations, starting from a VCF format file relating to the Indel variations, it is necessary to consider two specific fields of the FORMAT column, described in the header of the file as follows:

```
## FORMAT = <ID = FD, Number = 1, Type = Integer,
Description = "Fragment depth">
## FORMAT = <ID = FC, Number = 1, Type = Integer,
Description = "Fragment calls">
```

The first step therefore involves calling the get_FORMAT_ID function, in order to extract from the tumour sample the values corresponding to the positions identified by "FC" and "FD" in the FORMAT. Subsequently, the percentage frequencies of Indel variations are obtained through the ratio between the two identified values taken in the order described, multiplied by one hundred. The parse_VCF function described above, starting from a VCF file relating to the Indel variations, has the aim of returning the value of the variant allele frequency, followed by the corresponding entire line of the VCF file.

```
1  def read_ref_distribution(fd):
2          distr = []  #creation of the list in which the values will
       be saved
3          for row in fd: #iteration over each line of the reference
       distribution file
4                 row = row.strip().split() #splitting of the entire
       line into strings using the default separator
5                 start = float(row[0]) #saving of the first string
       contained in the line of the file (corresponding to the
       beginning of the range) in a variable
6                 end = float(row[1]) #saving of the second string
       contained in the line of the file (corresponding to the end of
       the range) in a variable
7                 value = float(row[2]) #saving of the third string
       contained in the line of the file (corresponding to the number
       of variations in the interval) in a variable
8                 distr.append((start,end, value)) #addition of
       numeric values to the list
9                 distr.sort() # sort by coordinate of the list
10         return distr
```

The read_ref_distribution function described above requires that a reference distribution file (corresponding to the distribution values of the red curve in the figure, indicated as a reference) be passed as a parameter in the format as the GSL-histogram

output. GSL-histogram is a demonstration program for the GNU Scientific Library. It takes three arguments, gsl-histogram *xmin xmax* [*n*], specifying the upper and lower bounds of the histogram and the number of bins. It then reads numbers from 'stdin', one line at a time, and adds them to the histogram. When there is no more data to read it prints out the accumulated histogram, providing as output a file containing line by line at the far left of the interval, far right of the interval, and the number of variations in that specific interval. The function read_ref_distribution, starting from the reference distribution file in the format as the GSL-histogram output, returns an ordered nested list containing, for each index, the three decimal numbers corresponding respectively to the start, end and number of variations of the interval.

- Central body of the script

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the reference distribution file has been provided followed by the VCF file relating to the Indel variations, saved respectively as REF_fd and INDEL_fd.

```
if len(sys.argv) != 2+1:  #checking the correctness of the number of
    files supplied in input
        sys.exit("""
        Usage: %s reference_freq-distr Indel.file.vcf|-
        """ % sys.argv[0])   #printing of information relating to
   the type of input requested if those provided are not correct
REF_fd = open(sys.argv[1])  #acquisition of the input reference
   distribution file
INDEL_fd = (sys.argv[2] != '-') and open(sys.argv[2]) or sys.stdin #
   acquisition of the VCF file relating to the Indel variations
```

By calling the previously defined function read_ref_distribution, the Indel frequency distribution, initially saved in the "distr" list, is copied into the "distr_dict" dictionary, using the "start end" pair (corresponding to the start and end of interval).

```
 #reading the distribution
distr = read_ref_distribution(REF_fd) #saving in distr the list
   obtained by calling the function
            distr_dict = {} #dictionary initialization
for i in distr: #iteration on each index of the list
        distr_dict[(i[0], i[1])] = i[2]  #dictionary definition
bins = sorted(distr_dict.keys()) #saving of sorted dictionary keys
   in bins
```

At this point, once the dictionary keys have been saved in "bins", sorted by coordinates identified by the extremes of the intervals considered, I want to assign the Indel variations to each interval. In particular, a dictionary called "Indels" is defined, in which each bin is associated with a certain number of lines of the VCF file for which the variant allele frequency (VAF), obtained by the parse_VCF function, falls within the defined range from bin.

```
# reading the VCF and assigning the Indel variations to the bins
Indels = {} #dictionary initialization
for i in bins: #iteration on each index of the list
        Indels[i] = [] #list initialization
for f, row in parse_VCF(INDEL_fd):  #iteration on each couple frequency, row result returned by the function
  for b in bins: #iteration on each index of the list
              if f <= b[1]: #checking if the frequency is lower than the far right of the range, it answers the question: does the frequency fall into the current bin?
                        Indels[b].append(row) #If the answer to the previous question is yes, then the VCF line is associated with the current bin through the "Indels" dictionary
                        break #in the moment the VCF line has been assigned to a specific bin, I move to the next line
```

The crucial aspect of the algorithm consists in the realisation of the sampling of the Indel variations through the definition of the so-called "support-unit". This process in principle provides for the calculation of ratios as the ratio between the number of lines of the VCF associated with a given bin (information obtained by calculating the length of the Indels [bin] list) and the frequency calculated for that specific bin (corresponding to the third value contained in the distr list for each index). All reports will then be saved in the "ratios" list. On the other hand, in the "non_zero_ratios" list, all the elements of the "ratios" list other than zero will be saved. At this point the "support-unit" variable takes on the minimum value among those identified in the "non_zero_ratios" list, unless this is empty, in which case the "support-unit" is set equal to zero. For each bin, the product between the "support-unit" and the number of variations identified for that interval (saved in distr_dict [bin]) is subsequently calculated, stored in the variable N. If N is lower than the number of rows of the VCF associated with the bin considered (N <len (Indels [bin])), then N is randomly selected from the lines of the VCF associated with the bin considered and are saved in "sampled_indels"; otherwise (if N> = len (Indels [bin])) all lines of the VCF associated with the considered bin are saved in "sampled_indels". Finally, all the contents of "sampled_indels" are printed.

```
# Realisation of the "support_unit"
ratios = [len(Indels[(start, end)])/value for (start, end, value) in distr if value != 0] #calculation of ratios as the ratio between the number of lines of the VCF associated with a given bin and the frequency calculated for that specific bin
non_zero_ratios = [i for i in ratios if i!=0] #saving of all the elements of the "ratios" list other than zero
if non_zero_ratios == []:
        support_unit = 0 #the "support-unit" is set equal to zero because all the ratios are equal to zero
        print("WARNING: empty ratios list.... setting support_unit to 0.", file=sys.stderr)
else:
        support_unit = min(non_zero_ratios) #"support-unit" variable takes on the minimum value among those identified in the "non_zero_ratios" list
```

```
for b in bins: #iteration on each index of the list
        N = int(support_unit*distr_dict[b]) #definition of N as the product between the "support-unit" and the number of variations identified for that interval
        if N<len(Indels[b]): #If N is lower than the number of rows of the VCF associated with the bin considered
                sampled_indels = random.sample(Indels[b], N) #N rows are randomly selected from the lines of the VCF associated with the bin considered and they are saved in "sampled_indels"
        else:
                sampled_indels = Indels[b] #all lines of the VCF associated with the considered bin are saved in "sampled_indels"
        for i in sampled_indels: #iteration on each element of the list
                print('\t'.join(i)) #all the content of "sampled_indels" is printed in the form of a single string consisting of tuples separated by tabs
```

C.3 Get_SNV_from_freq-distr.py

- Definition of functions
  - get_FORMAT_ID (VCF_row, ID)
  - read_ref_distribution (fd)

```
def parse_VCF(fd):
        for row in fd:  #iteration over each line of the VCF file
                if row[0] == '#':  #exclusion of all lines preceded by the symbol '#' corresponding to the header and description of the columns
                        continue
                row = row.strip().split('\t') #splitting of the entire line into strings using the tab as a separator
                VAF = 100*float(get_FORMAT_ID(row, "PM"))  #calculation of the variant allele frequency
                yield (VAF, row)  #extraction of the VAF value followed by the respective VCF row
```

Differently from what was observed for the ***Get_Indel_from_freq-distr.py*** script, in this case to extract the frequencies of SNV variations, starting from a VCF format file relating to the SNV variations, it is necessary to consider only one specific field of the FORMAT column, described in the header of the file as follows:

```
##FORMAT= <ID= PM, Number= 1, Type= Float,
Description= "Proportion of mutated allele">
```

The first step therefore involves calling the get_FORMAT_ID function, in order to extract from the tumour sample the values corresponding to the position identified

by "PM" in the FORMAT. Subsequently, the percentage values of frequencies of SNV variations are obtained by multiplying the identified value by one hundred. The parse_VCF function described above, starting from a VCF file relating to the SNV variations, has the aim of returning the value of the variant allele frequency, followed by the corresponding entire line of the VCF file.

- Central body of the script

Initially,it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the reference distribution file is provided followed by the VCF file relating to the SNV variations, saved respectively as REF_fd and SNV_fd.

```
if len(sys.argv) != 2+1:  #checking the correctness of the number of
    files supplied in input
        sys.exit("""
        Usage: %s reference_freq-distr SNV.file.vcf|-
        """ % sys.argv[0])   #printing of information relating to
    the type of input requested if those provided are not correct
REF_fd = open(sys.argv[1])  #acquisition of the input reference
    distribution file
SNV_fd = (sys.argv[2] != '-') and open(sys.argv[2]) or sys.stdin #
    acquisition of the VCF file relating to the SNV variations
```

By calling the previously defined function read_ref_distribution, the SNV frequency distribution, initially saved in the "distr" list, is copied into the "distr_dict" dictionary, using the "start end" pair (corresponding to the start and end of interval).

```
   #reading the distribution
distr = read_ref_distribution(REF_fd) #saving in distr the list
    obtained by calling the function
            distr_dict = {} #dictionary initialization
for i in distr: #iteration on each index of the list
        distr_dict[(i[0], i[1])] = i[2]  #dictionary definition
bins = sorted(distr_dict.keys()) #saving of sorted dictionary keys
    in bins
```

At this point, once the dictionary keys have been saved in "bins", sorted by coordinates identified by the extremes of the intervals considered, I want to assign the Indel variations to each interval. In particular, a dictionary called "SNV" is defined, in which each bin is associated with a certain number of lines of the VCF file for which the variant allele frequency (VAF), obtained by the parse_VCF function, falls within the defined range from bin.

```
 # reading the VCF and assigning the Indel variations to the bins
SNV = {} #dictionary initialization
for i in bins: #iteration on each index of the list
        SNV[i] = [] #list initialization
for f, row in parse_VCF(SNV_fd):  #iteration on each couple
    frequency, row result returned by the function
```

```
  for b in bins: #iteration on each index of the list
              if f <= b[1]: #checking if the frequency is lower than the far right of the range, it answers the question: does the frequency fall into the current bin?
                          SNV[b].append(row) #If the answer to the previous question is yes, then the VCF line is associated with the current bin through the "SNV" dictionary
                          break #in the moment the VCF line has been assigned to a specific bin, I move to the next line
```

The crucial aspect consists in the realisation of the sampling of the SNV variations through the definition of the so-called "support-unit". This process in principle provides for the calculation of ratios as the ratio between the number of lines of the VCF associated with a given bin (information obtained by calculating the length of the SNV[bin] list) and the frequency calculated for that specific bin (corresponding to the third value contained in the distr list for each index). All reports will then be saved in the "ratios" list. On the other hand, in the "non_zero_ratios" list, all the elements of the "ratios" list other than zero will be saved. At this point the "support-unit" variable takes on the minimum value among those identified in the "non_zero_ratios" list, unless this is empty, in which case the "support-unit" is set equal to zero. For each bin, the product between the "support-unit" and the number of variations identified for that interval (saved in distr_dict [bin]) is subsequently calculated, stored in the variable N. If N is lower than the number of rows of the VCF associated with the bin considered (N <len (SNV [bin])), then N is randomly selected from the lines of the VCF associated with the bin considered and are saved in "sampled_SNV "; otherwise (if N>=len(SNV [bin])) all lines of the VCF associated with the considered bin are saved in "sampled_SNV". Finally, all the contents of "sampled_SNV" are printed.

```
 # Realisation of the "support_unit"
ratios = [len(SNV[(start, end)])/value for (start, end, value) in distr if value != 0] #calculation of ratios as the ratio between the number of lines of the VCF associated with a given bin and the frequency calculated for that specific bin
non_zero_ratios = [i for i in ratios if i!=0] #saving of all the elements of the "ratios" list other than zero
if non_zero_ratios == []:
        support_unit = 0 #the "support-unit" is set equal to zero because all the ratios are equal to zero
        print("WARNING: empty ratios list.... setting support_unit to 0.", file=sys.stderr)
else:
        support_unit = min(non_zero_ratios) #"support-unit" variable takes on the minimum value among those identified in the "non_zero_ratios" list
for b in bins: #iteration on each index of the list
        N = int(support_unit*distr_dict[b]) #definition of N as the product between the "support-unit" and the number of variations identified for that interval
        if N<len(SNV[b]): #If N is lower than the number of rows of the VCF associated with the bin considered
```

```
12                  sampled_indels = random.sample(SNV[b], N) #N rows
    are randomly selected from the lines of the VCF associated with
    the bin considered and they are saved in "sampled_SNV"
13          else:
14                  sampled_SNV = SNV[b] #all lines of the VCF
    associated with the considered bin are saved in "sampled_SNV"
15          for i in sampled_SNV: #iteration on each element of the list
16                  print('\t'.join(i)) #all the content of "
    sampled_indels" is printed in the form of a single string
    consisting of tuples separated by tabs
```

C.4 Recalibrate_HRD_A.sh

- Acquisition of input files

```
1  BIN_DIR=$(readlink -e $(dirname $0)) #get the name of the directory
       where the script is executed
2  INDEL_NT_DISTRIBUTION=$BIN_DIR/Indel_median-freq_from_N-T_A #
       extraction of the reference distribution file in the format as
       the GSL-histogram output (median distribution of the Indel
       variations resulting from the comparison between the tumour
       sample and the associated normal)
3  SNV_NT_DISTRIBUTION=$BIN_DIR/SNV_median-freq_from_N-T_A #extraction
       of the reference distribution file in the format as the GSL-
       histogram output (median distribution of the SNV variations
       resulting from the comparison between the tumour sample and the
       associated normal)
4  export INDEL_NT_DISTRIBUTION SNV_NT_DISTRIBUTION BIN_DIR
5  if [ $# -ne 2 ]; then  #checking the correctness of the number of
       files supplied in input
6          echo "Usage: $0 directory.files tumour_sample" >&2 #printing
        of information relating to the type of input requested if those
        provided are not correct
7          echo "Recalibrate SNV and Indel files and recalculate HRD."
       >&2
8          exit 78
9  fi
10 FILES_DIR=$1 #acquisition of the input directory.files (result of
       the Sanger pipeline and the application of filters)
11 TUMOUR=$2  #acquisition of the name of the tumour sample provided in
        input
12 export TUMOUR
13
14
```

The first step is to get the name of the directory where the script is executed, saved in BIN_DIR. From this directory it is possible to go back to the Indel_median-freq_from_N-T_A and SNV_median-freq_from_N-T_A files, contained in the directory itself, corresponding to the reference distribution files in the format as the

GSL-histogram output, with a little modification. The Indel_median-freq_from_N-T and SNV_median-freq_from_N-T files (Table 5.2) respectively represent the median distribution of the SNV and Indel variations resulting from the comparison between the tumour sample and the associated normal, corresponding to the red curve in the graphs previously illustrated. The Indel_median-freq_from_N-T_A and SNV_median-freq_from_N-T_A files (Table 5.3), in particular, are obtained by replacing the value obtained by adding 1 to the original value in correspondence with the bin defined between 10 and 20 . This trick specifically allows you to select only the information contained in the 10-20 range, "cutting" the portion of variations that reside between the frequencies of 10 and 20%. This occurs since in that range the distribution of the N-T comparison is strongly spiked, leading to exclude the frequencies with a negligible incidence compared to that of the peak. Exporting the described files is followed by verification that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the directory.files has been provided followed by the name of the tumour sample considered, saved respectively as FILES_DIR and TUMOUR. In particular, the first input refers to a folder obtained as a result of the execution of the *cpgwgs* pipeline and the subsequent application of filters, corresponding to the input of the HRD pipeline. The second input instead refers to the name of the tumour sample, which can be identified in the terminal part of the header of the BAM file of the tumour sample, preceded by the abbreviation "SM:"(as can be seen in the figure for example for patient RC100851).

```
@RG    ID:1    LB:dummy_LB    PL:dummy_PL    PU:dummy_PU    SM:RC100851_T
```

- Definition of functions

```
function sampling_indel {
# $1: vcf.gz
# $2: out ($OUT_I/$(basename $1))
    (
            openAll $1 | \
                        grep '^#' #extraction of the header and the first line of the body of the VCF file taken as input
            openAll $1 | \
                       python3 $BIN_DIR/get_indel_from_freq-distr.py $INDEL_NT_DISTRIBUTION - | \
#execution of the script created in python described above providing as input the median distribution of the Indel variations resulting from the comparison between the tumour sample and the associated normal and the VCF file
                       sort -k1,1 -k2,2n
#ascending ordering of the VCF rows obtained following the execution of the sampling, sorted alphabetically by its column 1 (corresponding to the chromosome on which the variant is present) and sorted numerically by its column 2 (corresponding to the position on which the variant call was made respectively)
       ) | \
```

```
13          bgzip > $2 #compression of the file in the form file.gz and
    saving of the overall result obtained in the file corresponding
    to the second input of the function
14          tabix $2 #generation of the VCF file in the form vcf.gz and
    of its index file vcf.gz.tbi
15  }
16
```

As previously introduced the VCF format includes a header, whose lines are identified by the "##" symbols and a body which lists all the variants. Inside the body, the information relating to each variant is organised according to 9 mandatory columns that are identified by the "#" symbol. The sampling_indel function described above, starting from a VCF file relating to the Indel variations (first input supplied to the function), has the aim of returning a new VCF file, saved with the name indicated in correspondence with the second input of the function, characterised by the same header of the starting one followed by the corresponding lines of the VCF file selected by sampling. Specifically, the function begins with the extraction of the header and the first line of the body of the VCF file taken as input and proceeds with the execution of the script created in python described above (***Get_indel_from_freq-distr.py***) providing it as input the median distribution of the Indel variations resulting from the comparison between the tumour sample and the associated normal and the VCF file. Subsequently, the function foresees the ascending ordering of the rows of the VCF obtained following the execution of the sampling, sorted alphabetically by its column 1 (corresponding to the chromosome on which the variant is present) and sorted numerically by its column 2 (corresponding to the position on which the variant call was made respectively). Finally, the resulting file, including the header followed by the VCF lines selected by sampling, is compressed in the form "file.gz" and saved in the file corresponding to the second input of the function. The last step involves the generation of the VCF file in the form “vcf.gz” and of its index file “vcf.gz.tbi”. The sampling_SNV function is analogous to the sampling_indel function just described, with the only difference that it involves the execution of the python script ***Get_SNV_from_freq-distr.py*** providing it as input the median distribution of the SNV variations resulting from the comparison between the tumour sample and the associated normal and the VCF.

```
1   function sampling_SNV {
2 # $1: vcf.gz
3 # $2: out ($OUT_S/$(basename $1))
4     (
5             openAll $1 | \
6                     grep '^#' #extraction of the header and the
    first line of the body of the VCF file taken as input
7             openAll $1 | \
8                     python3 $BIN_DIR/get_SNV_from_freq-distr.py
    $SNV_NT_DISTRIBUTION - | \
9 #execution of the script created in python described above providing
     as input the median distribution of the SNV variations
    resulting from the comparison between the tumour sample and the
    associated normal and the VCF file
```

```
                    sort -k1,1 -k2,2n
#ascending ordering of the VCF rows obtained following the execution
     of the sampling, sorted alphabetically by its column 1 (
    corresponding to the chromosome on which the variant is present)
     and sorted numerically by its column 2 (corresponding to the
    position on which the variant call was made respectively)
    ) | \
        bgzip > $2 #compression of the file in the form file.gz and
    saving of the overall result obtained in the file corresponding
    to the second input of the function
        tabix $2 #generation of the VCF file in the form vcf.gz and
    of its index file vcf.gz.tbi
}
export -f sampling_indel sampling_SNV #export of the two functions
    described above

```

- Central body of the script

```
I create a "recalibrateA" subfolder in the FILES_DIR folder acquired
     as input
I position myself inside the "recalibrateA" subfolder within "
    recalibrateA"
I create the "Indel-sampling" and "SNV-sampling" subfolders
```

1. Sampling on N-T distribution

```
I print "Sampling SNV and Indel results based on N-T frequency
    distribution"

#Indel
I launch the sampling_indel function (previously defined)
    passing it as parameters to the file in the form Indel.*.vcf
    .gz (contained in FILES_DIR) followed by the file Indel.*.
    vcf.RANDOM-1.gz contained in the Indel-sampling folder

#SNV
I launch the sampling_SNV function (defined previously) passing
    it as parameters to the file in the form SNV.*.vcf.gz (
    contained in FILES_DIR) followed by the SNV.*.vcf.RANDOM-1.
    gz file contained in the Indel-sampling folder

```

Initially, the ***recalibrate_HRD_A.sh*** script foresees the execution of the sampling on the data that describe the distribution obtained from the comparison between the tumour sample and the normal equivalent. In particular, the sub-folders of the FILES_DIR folder provided as the first input to the script are initially defined: specifically, the two paths FILES_DIR/recalibrateA/Indel-sampling and FILES_DIR/recalibrateA/SNV-sampling are generated. The first step is then performed inside the recalibrateA folder, separately for SNV and Indel. In particular, the sampling_indel function (defined previously) is launched receiving as parameters the file in the form Indel.*.vcf.gz (contained in FILES_DIR)

followed by the file Indel.*.vcf.RANDOM-1.gz (contained in the Indel-sampling folder). Similarly, the sampling_SNV function (previously defined) is launched receiving as parameters the file in the form SNV.*.vcf.gz (contained in FILES_DIR) followed by the SNV.*.vcf.RANDOM-1.gz file (contained in the folder SNV-sampling). Consequently, in the Indel-sampling and SNV-sampling folders, it will be possible to access the files containing the sampling results on the data describing the distribution obtained from the comparison between the tumour sample and the normal equivalent, respectively for insertions/deletions and for single nucleotides variants. In particular, in this case, the Strategy A directly considers only what concerns the slice of allelic frequencies containing the greatest content of somatic mutations, without sampling. In this regard, it is in fact possible to note that the sampling_indel and the sampling_SNV functions are launched only once.

2. Recreate an updated '.files' directory

```
I print 'Recreate an un updated '.files' directory'
I define Indel_file as the basename of the file in the form
    Indel.*gz contained in the FILES_DIR folder
I define Indel_file_noExt as an indel_file of the .gz extension
I define rnd as the  Indel-sampling/$indel_file_noExt.*.gz file
I define SMP as the second "field" of rnd basename after using
    the '.' as a separator.
I define X as rnd without the .gz extension
I define RND as the last "field" of X after using the '.' as a
    separator.
I define and create the directory RND_DIR as $RND.files
I repeat for each i in CNV Indel SNV SV
     I copy files in form $i.* contained in the FILES_DIR folder
     in the RND_DIR directory created
#I replace the Indel
Overwrite rnd on Indel_file in the RND_DIR directory
Overwrite rnd.tbi on Indel_file.tbi in the RND_DIR directory
#I replace the SNV
I define SNV_file as the file basename in the form  SNV.*gz
    contained in the FILES_DIR folder
I define SNV_file_noExt as SNV_file without the .gz extension
Overwrite the file SNV.$SMP.annot.muts.vcf.$RND.gz contained in
    the SNV-sampling folder on the file $SNV_file in the RND_DIR
     directory
Overwrite the file SNV.$SMP.annot.muts.vcf.$RND.gz.tbi contained
     in the SNV-sampling folder on the file $SNV_file.tbi in the
     RND_DIR directory

```

During the second step, the ***recalibrate_HRD_A.sh*** script takes care of recreating an updated '.files' directory. Specifically, for the file in the form Indel.*_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz (for example Indel.PD24215a_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz) contained in the folder Indel-sampling, generated during Step 1, is performed a series of actions outlined below. Directory is initially defined and created in the form

RANDOM-1.files; the files Indel.*, CNV.*, SNV.*, SV.* (Indel.PD24215a_vs_hg38_metanormal.annot.vcf.gz, ...), contained in the FILES_DIR folder, are copied into this directory. At this point, the Indel replacement occurs, which involves overwriting the file in the form Indel.*gz contained in the respective RANDOM-1.files directory (Indel.PD24215a_vs_hg38_metanormal.annot.vcf.gz) with the file in the form Indel.*_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz (Indel.PD24215a_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz). The same is done for the file in the form Indel.*gz.tbi, replaced with the file in the form Indel.*_vs_hg38_metanormal.annot.vcf.RANDOM-1.gz.tbi. The same process is repeated analogously for the replacement of the SNVs inside the RANDOM-1.files directory: in particular the SNV.*_vs_hg38_metanormal.annot.muts.vcf.RANDOM-1.gz file contained in SNV-sampling (for example SNV.PD24215a_vs_hg38_metanormal.annot.muts.vcf.RANDOM-1.gz) overwrites the file SNV.*gz contained in the respective RANDOM-1.files (SNV.PD24215a_vs_hg38_metanormal.annot.muts.vcf.gz considering the previous example), as well as for the file in the form SNV.*gz.tbi.

3. Calculate HRD for the RANDOM directory

```
I print 'Calculate HRD for the RANDOM directory'
I place myself inside the RANDOM-1.files directory
I run the pipeline HRDetect_fullPipeline-hg38.AUTO.sh, contained
    in the $BIN_DIR folder (previously defined and containing
   the name of the directory where the script is executed)
   passing it as parameters the sample name of the tumour (
   saved in $TUMOUR) followed by the RANDOM-1.files directory.
   The script output will be saved in the HRDetect_fullPipeline
   .out file while any errors in the HRDetect_fullPipeline.err
   file

```

During the third step, the ***recalibrate__HRD__A.sh*** script takes care of calculating HRD for the RANDOM directory. Specifically, for the directory in the form RANDOM-1.files, contained in the recalibrateA folder, the pipeline HRDetect_fullPipeline-hg38.AUTO.sh is run, passing it as parameters the sample name of the tumour (saved in $TUMOUR) followed by the directory. The script output will be saved in the HRDetect_fullPipeline.out file while any errors in the HRDetect_fullPipeline.err file, both contained in RANDOM-1.files directory.

4. Recalculate HRD based on median values from data matrix

```
I print 'Finally, recalculate HRD based on median values from
    data matrix coming from the RANDOM directory'
I define FIRST_RANDOM as the RANDOM-1.files directory
I define DATA_MATRIX as $FILES_DIR without the extension '.files
    ' to which I add '.data-matrix' in the queue
```

```
4 Through the 'awk' command, I look for the values corresponding
    to the fields of del.mh.prop, SNV3,SNV8,SV3,SV5 and hrd, as
    well as the name of the sample, in $FIRST_RANDOM/
    HRDetect_fullPipeline.out .
5 I save in $DATA_MATRIX the column names corresponding to del.mh.
    prop, SNV3,SV3, SV5, hrd, SNV8, the name of the sample and
    the values associated with the previous columns.
6 Finally, I launch the pipeline _HRDetect-score_from_data-matrix.
    R, contained in the $BIN_DIR folder (previously defined and
    containing the name of the directory where the script is
    executed) passing it as parameters the $DATA_MATRIX. The
    script output will be saved in the HRDetect_fullPipeline.out
     file while any errors in the HRDetect_fullPipeline.err.
7
```

During the fourth step, the ***recalibrate__HRD__A.sh*** script takes care of recalculating HRD based on median values from datamatrix. In particular, the HRDetect pipeline requires an input data frame "data_matrix", which contains a sample in each row and one of six necessary features in each column. The six features are:

- proportion of deletions at microhomology (del.mh.prop),
- number of mutations of substitution signature 3 (SNV3),
- number of mutations of rearrangement signature 3 (SV3),
- number of mutations of rearrangement signature 5 (SV5),
- HRD LOH (Loss of Heterozygosity) index (hrd),
- number of mutations of substitution signature 8 (SNV8).

Initially, for the HRDetect_fullPipeline.out file (contained in the folder in the form RANDOM-1.files), generated during the previous step, I look for the values that I need to define the data matrix. Through the "awk" command, I look for the values corresponding to the fields of del.mh.prop, SNV3, SNV8, SV3, SV5 and hrd, as well as the name of the sample, in the RANDOM-1.files/HRDetect_fullPipeline.out. At this point the column names corresponding to del.mh.prop, SNV3, SV3, SV5, hrd, SNV8, the name of the sample and the values associated with the previous columns, are saved in hg38_metanormal__*.data-matrix. Finally, the pipeline _HRDetect-score_from_data-matrix.R is launched, passing it as input the hg38_metanormal___*.data-matrix. The script output will be saved in the HRDetect_fullPipeline.out file while any errors in the HRDetect_fullPipeline.err, both contained in the recalibrateA folder.

C.5 Get_VAF-range_indel.py

- Definition of functions
  - get_FORMAT_ID (VCF_row, ID)
  - read_ref_distribution (fd)

  - parse_VCF(fd)

- Central body of the script

  Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the lower and upper extremes of the considered interval were provided, followed by the VCF file relating to the Indel variations, saved respectively as start, end and INDEL_fd.

```
if len(sys.argv) != 3+1:  #checking the correctness of the number of files supplied in input
    sys.exit("""
    Usage: %s start end Indel.file.vcf|-
    """ % sys.argv[0])   #printing of information relating to the type of input requested if those provided are not correct
start = float(sys.argv[1]) #acquisition of the lower extreme of the considered interval
end = float(sys.argv[2]) #acquisition of the upper extreme of the considered interval
INDEL_fd = (sys.argv[2] != '-') and open(sys.argv[2]) or sys.stdin #acquisition of the VCF file relating to the Indel variations
```

  In particular, a dictionary called "Indels" is defined, in which the bin is associated with a certain number of lines of the VCF file for which the variant allele frequency (VAF), obtained by the parse_VCF function, falls within the defined range from bin.

```
Indels = {(start, end): []}  #dictionary initialization
for f, row in parse_VCF(INDEL_fd): #iteration on each couple frequency, row result returned by the function
    if start <= f <= end: #checking if the frequency is lower than the far right of the range and higher than the far left. It answers the question: does the frequency fall into the current bin?
        Indels[(start, end)].append(row)#If the answer to the previous question is yes, then the VCF line is associated with the current bin through the "Indels" dictionary
for k, v in Indels.items(): #iteration over all the key-value pairs present in the dictionary
    for i in v:  #iteration on all the lines of the VCF associated with the specific bin considered
        print('\t'.join(i)) #all the lines of the VCF associated with the specific bin considered, contained in the dictionary, are printed in the form of a single string consisting of tuples separated by tabs
```

  It is of particular importance to underline that, unlike what was observed for the ***Get_indel_from_freq-distr.py*** script, in this case a single specific interval is considered, of which the extremes are supplied as input. At the same time, the crucial aspect of the ***Get_indel_from_freq-distr.py*** algorithm consists in the

realisation of the sampling of the Indel variations through the definition of the so-called "support-unit"; in particular, for each bin, the product between the "support-unit" and the number of variations identified for that interval was calculated and stored in the variable N. If N was lower than the number of rows of the VCF associated with the bin considered, then N was randomly selected from the lines of the VCF associated with the bin considered and were saved in "sampled_indels"; otherwise all lines of the VCF associated with the considered bin were saved in "sampled_indels". Finally, all the contents of "sampled_indels" were printed. In this case, however, in the ***Get_VAF-range_indel.py*** script, neither the "support-unit" nor sampling is used, in fact, once the VCF lines associated with the specific bin considered have been identified, these are printed in their wholeness.

C.6 Get_VAF-range_SNV.py

- Definition of functions
  - get_FORMAT_ID (VCF_row, ID)
  - read_ref_distribution (fd)
  - parse_VCF(fd)
- Central body of the script

  Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the lower and upper extremes of the considered interval are provided followed by the VCF file relating to the SNV variations, saved respectively as start, end and SNV_fd.

```
1  if len(sys.argv) != 3+1:  #checking the correctness of the number of
       files supplied in input
2          sys.exit("""
3          Usage: %s start end SNV.file.vcf|-
4          """ % sys.argv[0])   #printing of information relating to
       the type of input requested if those provided are not correct
5  start = float(sys.argv[1]) #acquisition of the lower extreme of the
       considered interval
6  end = float(sys.argv[2]) #acquisition of the upper extreme of the
       considered interval
7  SNV_fd = (sys.argv[2] != '-') and open(sys.argv[2]) or sys.stdin #
       acquisition of the VCF file relating to the SNV variations
8  In particular, a dictionary called "SNV" is defined, in which the
       bin is associated with a certain number of lines of the VCF file
        for which the variant allele frequency (VAF), obtained by the
       parse_VCF function, falls within the defined range from bin.
9  SNV = {(start, end): []}  #dictionary initialization
10 for f, row in parse_VCF(SNV_fd): #iteration on each couple frequency
       , row result returned by the function
```

```
11          if start <= f <= end: #checking if the frequency is lower
    than the far right of the range and higher than the far left. It
     answers the question: does the frequency fall into the current
    bin?
12                  SNV[(start, end)].append(row)#If the answer to the
    previous question is yes, then the VCF line is associated with
    the current bin through the "SNV" dictionary
13 for k, v in SNV.items(): #iteration over all the key-value pairs
    present in the dictionary
14          for i in v:  #iteration on all the lines of the VCF
    associated with the specific bin considered
15                  print('\t'.join(i)) #all the lines of the VCF
    associated with the specific bin considered, contained in the
    dictionary, are printed in the form of a single string
    consisting of tuples separated by tabs
```

Again it is possible to emphasise, unlike what was observed for the ***Get_SNV_from_freq-distr.py*** script, in this case a single specific interval is considered, of which the extremes are supplied as input. At the same time, the crucial aspect of the ***Get_SNV_from_freq-distr.py*** algorithm consists in the realisation of the sampling of the SNV variations through the definition of the product between the "support-unit" and the number of variations identified for that interval, stored in the variable N. Then, if N was lower than the number of rows of the VCF associated with the bin considered, N was randomly selected from the lines of the VCF associated with the bin considered and were saved in "sampled_SNV"; otherwise all lines of the VCF associated with the considered bin were saved in "sampled_SNV". Finally, all the contents of "sampled_SNV" were printed. In this case, however, in the ***Get_VAF-range_SNV.py*** script, neither the "support-unit" nor sampling is used, in fact, once the VCF lines associated with the specific bin considered have been identified, these are printed in their wholeness.

C.7 Recalibrate_HRD_B.sh

- Acquisition of input files

```
1 BIN_DIR=$(readlink -e $(dirname $0)) #get the name of the directory
    where the script is executed
2 if [ $# -ne 3 ]; then
3          echo "Usage: $0 directory.files tumour_sample out_dir" >&2 #
    checking the correctness of the number of files supplied in
    input
4          echo "Recalibrate SNV and Indel files and recalculate HRD."
    >&2 #printing of information relating to the type of input
    requested if those provided are not correct
5          exit 78
6 fi
7 FILES_DIR=$(readlink -e $1)  #acquisition of the input directory.
    files (result of the Sanger pipeline and the application of
    filters)
```

```
8  TUMOUR=$2 #acquisition of the name of the tumour sample provided in
       input
9  OUT_DIR=$3 #acquisition of the output directory in which the results
       will be saved
10 L=30 #VAF range ('width' of the interval)
11 SHIFT=5 # sliding window shift
12 export TUMOUR
13 mkdir $OUT_DIR #creation of the output directory that will contain
       the results obtained by the script
14
```

I verify that the script has been launched correctly, introducing a check that verifies that what is needed for the script to run has been provided as input. In particular, I verify that the name of the directory where the script will be executed has been provided, followed by the name of the tumour sample and the name of the output directory where the results will be printed, saved respectively as FILES_DIR, TUMOUR and OUT_DIR. At this point I proceed with the definition of variables such as L (which is assigned the value 30), corresponding to the VAF range or the "width" of the interval considered, and SHIFT (which is assigned the value 5), corresponding to the sliding window shift. It proceeds with the creation of the directory OUT_DIR that will contain the results produced by the execution of the script and with the definition of some necessary functions for the same execution.

- Definition of functions

```
1  function sliding_indel {
2  # $1: start
3  # $2: end
4  # $3: vcf.gz
5  # $4: out ($OUT_I/$(basename $1)
6          (
7                  openAll $3 | \
8                          grep '^#' #extraction of the header and the
       first line of the body of the VCF file taken as input
9                  openAll $3 | \
10                         python3 $BIN_DIR/get_VAF-range_indel.py $1
       $2 - | \
11 #execution of the script created in python described above providing
       as input the extremes of the interval considered and the VCF
       file
12                         sort -k1,1 -k2,2n
13 #ascending ordering of the VCF rows obtained following the execution
       of the sliding, sorted alphabetically by its column 1 (
       corresponding to the chromosome on which the variant is present)
       and sorted numerically by its column 2 (corresponding to the
       position on which the variant call was made respectively)
14         ) | \
15         bgzip > $4 #compression of the file in the form file.gz and
       saving of the overall result obtained in the file corresponding
       to the second input of the function
16         tabix $4 #generation of the VCF file in the form vcf.gz and
       of its index file vcf.gz.tbi
```

As previously introduced the VCF format includes a header, whose lines are identified by the "##" symbols and a body which lists all the variants. Inside the body, the information relating to each variant is organised according to 9 mandatory columns that are identified by the "#" symbol. The sliding_indel function described above, starting from a VCF file relating to the Indel variations (third input supplied to the function), has the aim of returning a new VCF file, saved with the name indicated in correspondence with the fourth input of the function, characterised by the same header of the starting one followed by the corresponding lines of the VCF file selected by sliding. Specifically, the function begins with the extraction of the header and the first line of the body of the VCF file taken as input and proceeds with the execution of the script created in Python described above (***get_VAF-range_indel.py***) providing it as input the lower and upper extremes of the considered interval and the VCF file. Subsequently, the function foresees the ascending ordering of the rows of the VCF obtained following the execution of the sliding, sorted alphabetically by its column 1 (corresponding to the chromosome on which the variant is present) and sorted numerically by its column 2 (corresponding to the position on which the variant call was made respectively). Finally, the resulting file, including the header followed by the VCF lines selected by sampling, is compressed in the form "file.gz" and saved in the file corresponding to the second input of the function. The last step involves the generation of the VCF file in the form “vcf.gz” and of its index file “vcf.gz.tbi”.

```
function sliding_SNV {
# $1: start
# $2: end
# $3: vcf.gz
# $4: out ($OUT_I/$(basename $1)
        (
                openAll $3 | \
                        grep '^#' #extraction of the header and the first line of the body of the VCF file taken as input
                openAll $3 | \
                        python3 $BIN_DIR/get_VAF-range_SNV.py $1 $2 - | \
#execution of the script created in python described above providing as input the extremes of the interval considered and the VCF file
                        sort -k1,1 -k2,2n
#ascending ordering of the VCF rows obtained following the execution of the sliding, sorted alphabetically by its column 1 (corresponding to the chromosome on which the variant is present) and sorted numerically by its column 2 (corresponding to the position on which the variant call was made respectively)
        ) | \
        bgzip > $4 #compression of the file in the form file.gz and saving of the overall result obtained in the file corresponding to the second input of the function
        tabix $4 #generation of the VCF file in the form vcf.gz and of its index file vcf.gz.tbi
```

The sliding_SNV function is analogous to the sliding_indel function just described, with the only difference that it involves the execution of the python script ***get_VAF-range_indel.py*** providing it as input the lower and upper extremes of the considered interval and the VCF.

```
function run_HRDetect {
        local SMP=$1 #acquisition of the tumour sample provided in input
        local SNV=$(readlink -e $2)  #acquisition of SNV file provided in input
        local SV=$(readlink -e $3) #acquisition of SV file provided in input
        local INDEL=$(readlink -e $4) #acquisition of INDEL file provided in input
        local CNV=$(readlink -e $5) #acquisition of CNV file provided in input
        Rscript $BIN_DIR/HRDetect_fullPipeline-hg38.R $SMP $SNV $SV $INDEL $CNV
#running of HRDetect pipeline passing it as parameters SMP, SNV, SV, INDEL, CNV in the described order
}
export -f sliding_indel sliding_SNV run_HRDetect
```

The run_HRDetect function described above, starting from the sample name of the tumour and the SNV, SV, INDEL, CNV files given as input, has the aim of returning the results obtained by the running of HRDetect pipeline, using the listed input as parameters.

- Central body of the script

```
I position myself inside the $OUT_DIR folder provided in input
Within $OUT_DIR I create the "Indel-sliding", "SNV-sliding" and "out_HRD" subfolders
```

1. **Sliding**

```
I print "Sliding SNV and Indel results"
#Indel
I repeat for each 'start' in a sequence ranging from 0 to 100-$L (i.e. 70) with a step of $SHIFT (5):
I define $end, corresponding to the upper extreme of the bin, as the sum of the value of $start and $L (equal to 30)
I launch the sliding_indel function (previously defined) passing it as parameters the values contained in $start and in $end followed by the file in the form Indel.*.vcf.gz (contained in FILES_DIR) and the file in the form Indel.*.vcf.SLIDE-$start-$end.gz (contained in the Indel-sliding folder)
#SNV
I repeat for each 'start' in a sequence ranging from 0 to 100-$L (i.e. 70) with a step of $SHIFT (5):
I define $end, corresponding to the upper extreme of the bin, as the sum of the value of $start and $L (equal to 30)
```

```
I launch the sliding_indel function (previously defined) passing
    it as parameters the values contained in $start and in $end
    followed by the file in the form SNV.*.vcf.gz (contained in
    FILES_DIR) and the file in the form SNV.*.vcf.SLIDE-$start-
   $end.gz (contained in the SNV-sliding folder)

```

Initially, the ***Recalibrate_HRD_B.sh*** script foresees the execution of the sliding on the SNV and Indel results. In particular, the sub-folders of the OUT_DIR folder provided as the third input to the script are initially defined: specifically, the three paths OUT_DIR/Indel-sliding, OUT_DIR/SNV-sliding and OUT_DIR/out_HRD are generated. The first step is then performed inside the OUT_DIR folder, separately for SNV and Indel. In particular, the sliding_indel function (defined previously) is launched for each "start" in a sequence ranging from 0 to 100-$L (ie 70) with a step of 5 ($SHIFT = 5) receiving as parameters the values contained in $start and in $end, the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.SLIDE-$start-$end.gz (contained in the Indel-sliding folder), where "start" represents the index that is updated at each iteration. Similarly, the sliding_SNV function (previously defined) is launched for each "start" in the sequence [0,5,10, ... 65,70] receiving as parameters the values contained in $start and in $end, the file in the form SNV.*.vcf.gz (contained in FILES_DIR) followed by the file SNV.*.vcf.SLIDE-$start-$end.gz (contained in the SNV-sliding folder). Consequently, in the Indel-sliding and SNV-sliding folders, it will be possible to access the files containing the sliding results on the data for insertions/deletions and for single nucleotides variants.

2. **Run HRDetect**

```
I print "Running HRDetect"
I repeat for each "indel" in Indel-sliding/Indel.*.gz (ie I
    cycle on all files generated during Step 1 contained in the
    Indel-sliding folder)
      Using the "awk" command, I extract the range from the
    basename of $indel and I save it in the variable "range"
      I define "snv" as the file in the form SNV.*$range.gz
    contained in SNV_sliding folder
      I define "OUT" as $OUT_HRD/INDEL-$range.SNV-$range
      I create the tmp-sliding.INDEL-$range.SNV-$range directory
      I position myself into the newly created tmp-sliding.INDEL
    -$range.SNV-$range directory
      I run the function run_HRDetect (previously defined)
    passing it as parameters the sample name of the tumour (
    saved in $TUMOUR) followed by the $snv, $SV, $indel, $CNV
    files. The script output will be saved in the $OUT.
    HRDetect_fullPipeline.out file while any errors in the $OUT.
    HRDetect_fullPipeline.err file
Finally, I delete the folders in the form tmp-sliding.*
```

During the second and last step, the ***Recalibrate_HRD_B.sh*** script takes

care of calculating the HRD score, running the function run_HRDetect. Specifically, for each file in the form Indel.*.gz, files generated during Step 1 contained in the Indel-sliding folder, the function run_HRDetect is run, passing it as parameters the sample name of the tumour (saved in $TUMOUR) followed by the files: SNV.*$range.gz contained in SNV_sliding (e.g. SNV.PD24215a_vs_hg38_metanormal.annot.muts.vcf.SLIDE-0-30.gz for patient PD24215a and range 0-30) , $FILES_DIR/SV.*.gz (SV.PD24215a_vs_hg38_metanormal.annot.bedpe.OK.gz), Indel-sliding/Indel.*.gz (Indel.PD24215a_vs_hg38_metanormal.annot .vcf.SLIDE-0-30.gz) and $FILES_DIR/CNV.*.OK (CNV.PD24215a.copynumber .caveman.OK). The script output will be saved in the INDEL-$range.SNV-$range.HRDetect_fullPipeline.out file while any errors in the INDEL-$range.SNV-$range.HRDetect_fullPipeline.err file, both contained in $OUT_HRD directory.

C.8 Recalibrate_HRD_C.sh

- Acquisition of input files

```
1  BIN_DIR=$(readlink -e $(dirname $0)) #get the name of the directory
       where the script is executed
2  INDEL_NT_DISTRIBUTION=$BIN_DIR/Indel_median-freq_from_N-T#extraction
        of the reference distribution file in the format as the GSL-
       histogram output (median distribution of the Indel variations
       resulting from the comparison between the tumour sample and the
       associated normal)
3  SNV_NT_DISTRIBUTION=$BIN_DIR/SNV_median-freq_from_N-T #extraction of
        the reference distribution file in the format as the GSL-
       histogram output (median distribution of the SNV variations
       resulting from the comparison between the tumour sample and the
       associated normal)
4  export INDEL_NT_DISTRIBUTION SNV_NT_DISTRIBUTION BIN_DIR
5  if [ $# -ne 2 ]; then  #checking the correctness of the number of
       files supplied in input
6          echo "Usage: $0 directory.files tumour_sample" >&2 #printing
        of information relating to the type of input requested if those
        provided are not correct
7          echo "Recalibrate SNV and Indel files and recalculate HRD."
       >&2
8          exit 78
9  fi
10 FILES_DIR=$1 #acquisition of the input directory.files (result of
       the Sanger pipeline and the application of filters)
11 TUMOUR=$2  #acquisition of the name of the tumour sample provided in
        input
12 N=10 #definition of the sampling number
13 export TUMOUR
14
```

The first step is to get the name of the directory where the script is executed, saved

in BIN_DIR. From this directory it is possible to go back to the Indel_median-freq_from_N-T and SNV_median-freq_from_N-T files (Table 5.2), contained in the directory itself, corresponding to the reference distribution files in the format as the GSL-histogram output. These files respectively represent the median distribution of the SNV and Indel variations resulting from the comparison between the tumour sample and the associated normal, corresponding to the red curve in the graphs previously illustrated. Exporting the described files is followed by verification that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the directory.files has been provided followed by the name of the tumour sample considered, saved respectively as FILES_DIR and TUMOUR. In particular, the first input refers to a folder obtained as a result of the execution of the cpgwgs pipeline and the subsequent application of filters, corresponding to the input of the HRD pipeline. The second input instead refers to the name of the tumour sample, which can be identified in the terminal part of the header of the BAM file of the tumour sample, preceded by the abbreviation "SM:" (as can be seen in the figure for example for patient RC100851).

```
@RG    ID:1    LB:dummy_LB    PL:dummy_PL    PU:dummy_PU    SM:RC100851_T
```

- Definition of functions
  - sampling_Indel
  - sampling_SNV
- Central body of the script

The main body of the algorithm (viewable in the form of pseudocode) can be simply described through the identification of four fundamental steps, as observable in the figure.

```
I create a "recalibrateC" subfolder in the FILES_DIR folder acquired
    as input
I position myself inside the "recalibrateC" subfolder within "
   recalibrateC"
I create the "Indel-sampling" and "SNV-sampling" subfolders
```

1. **Sampling on N-T distribution**

```
I print "Sampling SNV and Indel results based on N-T frequency
    distribution" #Indel
I repeat for each i in N (where N is the variable previously
    defined as sampling number, set equal to 10)
       I launch the sampling_indel function (previously defined)
    passing it as parameters to the file in the form Indel. *.
    vcf.gz (contained in FILES_DIR) followed by the file Indel.
    *. vcf.RANDOM-$i.gz contained in the Indel-sampling folder

#SNV
```

```
I repeat for each i in N (where N is the variable previously
    defined as sampling number, set equal to 1)
      I launch the sampling_SNV function (defined previously)
    passing it as parameters to the file in the form SNV.*.vcf.
    gz (contained in FILES_DIR) followed by the SNV.*.vcf.RANDOM
    -$i.gz file contained in the Indel-sampling folder
```

Initially, the ***recalibrate_HRD_C.sh*** script foresees the execution of the sampling on the data that describe the distribution obtained from the comparison between the tumour sample and the normal equivalent. In particular, the sub-folders of the FILES_DIR folder provided as the first input to the script are initially defined: specifically, the two paths FILES_DIR/ recalibrateC/Indel-sampling and FILES_DIR/recalibrateC/SNV-sampling are generated. The first step is then performed inside the recalibrateC folder, separately for SNV and Indel. In particular, the sampling_indel function (defined previously) is launched N times receiving as parameters the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.RANDOM-$i.gz (contained in the Indel-sampling folder), where "i" represents the index that is updated at each iteration proceeding from 1 to N. Similarly, the sampling_SNV function (previously defined) is launched N times receiving as parameters the file in the form SNV.*.vcf.gz (contained in FILES_DIR) followed by the SNV.*.vcf.RANDOM-$i.gz file (contained in the folder SNV-sampling). Consequently, in the Indel-sampling and SNV-sampling folders, it will be possible to access the files containing the sampling results on the data describing the distribution obtained from the comparison between the tumour sample and the normal equivalent, respectively for insertions/deletions and for single nucleotides variants.

2. **Recreate an updated '.files' directory**

```
I print "Recreate an un updated '.files' directory"
I define Indel_file as the basename of the file in the form
    Indel.*gz contained in the FILES_DIR folder
I define Indel_file_noExt as an indel_file of the ".gz"
    extension
I repeat for each rnd in Indel-sampling/$indel_file_noExt.*.gz (
    ie I cycle on all files generated during Step 1 contained in
     the Indel-sampling folder)
    I define SMP as the second "field" of rnd basename after
    using the "." as a separator.
    I define X as rnd without the ".gz" extension
    I define RND as the last "field" of X after using the "." as
     a separator.
    I define and create the directory RND_DIR as $RND.files
    I repeat for each i in CNV Indel SNV SV
        I copy files in form $i.* contained in the FILES_DIR
    folder in the RND_DIR directory created
    #I replace the Indel
    Overwrite rnd on Indel_file in the RND_DIR directory
    Overwrite rnd.tbi on Indel_file.tbi in the RND_DIR directory
    #I replace the SNV
```

```
15      I define SNV_file as the file basename in the form  SNV.*gz
        contained in the FILES_DIR folder
16      I define SNV_file_noExt as SNV_file without the ".gz"
        extension
17      Overwrite the file SNV.$SMP.annot.muts.vcf.$RND.gz contained
         in the SNV-sampling folder on the file $SNV_file in the
        RND_DIR directory
18      Overwrite the file SNV.$SMP.annot.muts.vcf.$RND.gz.tbi
        contained in the SNV-sampling folder on the file $SNV_file.
        tbi in the RND_DIR directory
```

During the second step, the ***recalibrate__HRD__C.sh*** script takes care of recreating an updated '.files' directory. Specifically, for each file in the form Indel.*__vs__hg38__metanormal.annot.vcf.RANDOM-*.gz (for example Indel.PD24215a__vs__hg38__metanormal.annot.vcf.RANDOM-1.gz) contained in the folder Indel-sampling, generated during Step 1, is performed a series of actions outlined below. Directories are initially defined and created in the form RANDOM-*.files (RANDOM-1.files referring to the previous example); the files Indel.*, CNV.*, SNV.*, SV.* (Indel.PD24215a__vs__hg38__metanormal.annot .vcf.gz, ...), contained in the FILES_DIR folder, are copied into these directories. At this point, the Indel replacement occurs, which involves overwriting the file in the form Indel.*gz contained in the respective RANDOM-*.files directory (Indel.PD24215a__vs__hg38__metanormal.annot.vcf.gz) with the file in the form Indel.*__vs__hg38__metanormal.annot.vcf.RANDOM-*.gz (Indel.PD24215a__vs __hg38__metanormal.annot.vcf.RANDOM-1.gz).
The same is done for the file in the form Indel.*gz.tbi, replaced with the file in the form Indel.*__vs__hg38__metanormal.annot.vcf.RANDOM-*.gz.tbi. The same process is repeated analogously for the replacement of the SNVs inside the RANDOM-*.files directory: in particular the SNV.*__vs__hg38__metanormal .annot.muts.vcf.RANDOM-*.gz file contained in SNV-sampling (for example SNV.PD24215a__vs__hg38__metanormal.annot.muts.vcf.RANDOM-1.gz) overwrites the file SNV.*gz contained in the respective RANDOM-*.files (SNV.PD24215a __vs__hg38__metanormal.annot.muts.vcf.gz considering the previous example), as well as for the file in the form SNV.*gz.tbi.

3. **Calculate HRD for all RANDOM directories**

```
1 I print "Calculate HRD for all RANDOM directories"
2 I repeat for each i in the $RND.files folders (in the form
      RANDOM-*.files ) created in step 2
3       I place myself inside the $i directory
4       I run the pipeline HRDetect_fullPipeline-hg38.AUTO.sh,
        contained in the $BIN_DIR folder (previously defined and
        containing the name of the directory where the script is
        executed) passing it as parameters the sample name of the
        tumour (saved in $TUMOUR) followed by the $i directory. The
        script output will be saved in the HRDetect_fullPipeline.out
         file while any errors in the HRDetect_fullPipeline.err file
```

During the third step, the ***recalibrate_HRD_C.sh*** script takes care of calculating HRD for all RANDOM directories. Specifically, for each directory in the form RANDOM-*.files, contained in the recalibrateA folder, the pipeline HRDetect_fullPipeline-hg38.AUTO.sh is run, passing it as parameters the sample name of the tumour (saved in $TUMOUR) followed by the directory. The script output will be saved in the HRDetect_fullPipeline.out file while any errors in the HRDetect_fullPipeline.err file, both contained in RANDOM-*.files directory.

4. **Recalculate HRD based on median values from data matrix**

```
I print "Finally, recalculate HRD based on median values from
    data matrix coming from RANDOM directories"
I define FIRST_RANDOM as the RANDOM-1.files directory
I define DATA_MATRIX as $FILES_DIR without the extension ".files
    " to which I add ".data-matrix" in the queue
I repeat for each "out" in HRDetect_fullPipeline.out files (
    contained in the folders in the form RANDOM-*.files)
Through the "awk" command, I look for the values corresponding
    to the fields of del.mh.prop, SNV3 e SNV8 for each file in
    the form RANDOM-*.files/ HRDetect_fullPipeline.out
Calculating the median (per column) of all the values
    corresponding to del.mh.prop, SNV3 and SNV8, extracted from
    RANDOM-*.files/ HRDetect_fullPipeline.out files, obtaining
    three numbers saved in variables MH, SNV_3, SNV_8.
    Through the "awk" command, I look for the values
    corresponding to the fields of del.mh.prop, SNV3,SNV8,SV3,
    SV5 and hrd, as well as the name of the sample, in
    $FIRST_RANDOM/HRDetect_fullPipeline.out .
I replace the "original" values corresponding to del.mh.prop,
    SNV3 and SNV8 with the median values calculated previously (
    ie MH, SNV_3, SNV_8)
I save in $DATA_MATRIX the column names corresponding to del.mh.
    prop, SNV3,SV3, SV5, hrd, SNV8, the name of the sample and
    the values associated with the previous columns, taking into
     account the replacements (sample, MH, SNV_3, SV3, SV5, hrd,
     SNV_8)
Finally, I launch the pipeline _HRDetect-score_from_data-matrix.
    R, contained in the $BIN_DIR folder (previously defined and
    containing the name of the directory where the script is
    executed) passing it as parameters the $DATA_MATRIX. The
    script output will be saved in the HRDetect_fullPipeline.out
     file while any errors in the HRDetect_fullPipeline.err.
```

During the fourth step, the ***recalibrate_HRD_C.sh*** script takes care of recalculating HRD based on median values from datamatrix. In particular, the HRDetect pipeline requires an input data frame "data_matrix", which contains a sample in each row and one of six necessary features in each column. The six features are:

- proportion of deletions at microhomology (del.mh.prop),
- number of mutations of substitution signature 3 (SNV3),

- number of mutations of rearrangement signature 3 (SV3),
- number of mutations of rearrangement signature 5 (SV5),
- HRD LOH (Loss of Heterozygosity) index (hrd),
- number of mutations of substitution signature 8 (SNV8).

Initially, for each of the HRDetect_fullPipeline.out files (contained in the folders in the form RANDOM-*.files), generated during the previous step, I look for the values corresponding to the fields of del.mh.prop, SNV3 and SNV8. Of the N values (in this case N is defined equal to 10) extracted for each field from the HRDetect_fullPipeline.out files (since the RANDOM-*.files folders are N and each contains only one HRDetect_fullPipeline.out file), the medians are calculated (per column), obtaining three numbers saved in variables MH, SNV_3, SNV_8. Through the "awk" command, look for the values corresponding to the fields of del.mh.prop, SNV3, SNV8, SV3, SV5 and hrd, as well as the name of the sample, in the RANDOM-1.files/HRDetect_fullPipeline.out. At this point the "original" values corresponding to del.mh.prop, SNV3 and SNV8 are replaced with the median values calculated previously (ie MH, SNV_3, SNV_8) and are saved in hg38_metanormal___*.data-matrix the column names corresponding to del.mh.prop, SNV3, SV3, SV5, hrd, SNV8, the name of the sample and the values associated with the previous columns, taking into account the replacements (sample, MH, SNV_3, SV3, SV5, hrd, SNV_8). Finally, the pipeline _HRDetect-score_from_data-matrix.R is launched, passing it as input the hg38_metanormal___*.data-matrix. The script output will be saved in the HRDetect_fullPipeline.out file while any errors in the HRDetect_fullPipeline.err, both contained in the recalibrateC folder.

C.9 Get_Indel_freq-distr_anyPerc.NORM.sh

```
1 if [ $# -ne 2 ]; then  #checking the correctness of the number of files
      supplied in input
2       echo "Usage: $0 Indel.vcf[.gz] bin_length" >&2 #printing of
      information relating to the type of input requested if those
      provided are not correct
3       echo "Output the normalised GSL-histogram output" >&2 #printing
      of information relating to the type of output returned by the script
4       exit 78
5 fi
6 INDEL=$1 #acquisition of the VCF file relating to the Indel variations
7 L=$2 #acquisition of the length of the bin by input
```

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the VCF file relating to the Indel variations has been provided followed by the length of the bin, saved respectively as INDEL and L.

```
1 eval $(echo $L | awk '{
```

```
                        L=$1 #acquisition of the first (and only) argument acquired in input by the "AWK" command that corresponds to the length of the bin
                        N=int(100/$1)+1;  #calculation of the total number of bins based on the length of the bin
                        start=-1*L/2; #definition of the extreme left of the first bin
                        end=100+L/2; #definition of the extreme right of the last bin
                        print "N="N";", "start="start";", "end="end";" #print of the values corresponding to the number of bins followed by the start and the end
                }')
zcat -f $INDEL | \
        grep -v '^#' | \        #extraction of all the lines of the VCF file taken as input with the exception of those preceded by the symbol "#"
        awk 'BEGIN{OFS="\t"}
                {
                        n2=split($(NF),t,":")  #splitting of the string of the last column of the VCF file into strings using the ":" as a separator, saving these strings in the "t" array
                        print 100*t[n2]/t[n2-1]
#extraction of the value in the tumour sample corresponding to the position of "FC" in the FORMAT, saved in t[n2]. Extraction of the value in the tumour sample corresponding to the position of "FD" in the FORMAT, saved in t[n2-1]. Calculation and print of the variant allele frequency.
                }' | \
        gsl-histogram -u -- $start $end $N
#use of the gsl-histogram command to save data in the required format
```

Through the use of the "awk" command, the length of the bin acquired in input is used to calculate the number of total bins contained in an interval from 0 to 100, which is saved in the variable N. The length of the bin is also used for the definition of the start and end, corresponding respectively to the left end of the first bin and the right end of the last bin outlined. Specifically, these do not correspond directly to the values 0 and 100 as it is advisable for graphic reasons to translate the two positions so that, once the data has been used to create a histogram, the latter is centred and not superimposed on the axis lines. At this point, the values corresponding to the number of bins, the start and the end are printed. Remember in particular that, as previously introduced, the VCF format includes a header, whose lines are identified by the "##" symbols, the 9 mandatory columns identified by the "#" symbol and a body which lists all the variants. In this regard, I proceed by extraction of all the lines of the VCF file taken as input with the exception of those preceded by the symbol "#" (therefore only the lines belonging to the body of the file); for each of these lines splitting of the string of the last column of the VCF file into smaller strings using the ":" as a separator, saving these strings in the "t" array. In fact, remember that to extract the frequencies of Indel variations, starting from a VCF format file relating to the Indel variations, it is necessary to consider two specific fields of the FORMAT column, described in the header of the file as follows:

```
## FORMAT = <ID = FD, Number = 1, Type = Integer,
Description = "Fragment depth">
## FORMAT = <ID = FC, Number = 1, Type = Integer,
Description = "Fragment calls">
```

I therefore proceed with the extraction of the value in the tumour sample corresponding to the position of "FC" in the FORMAT, saved in $t[n2]$ and with the extraction of the value in the tumour sample corresponding to the position of "FD" in the FORMAT, saved in $t[n2-1]$. Subsequently, the frequencies of Indel variations are obtained through the ratio between the two identified values taken in the order described. Therefore, the result of multiplying the quotient obtained by 100 is printed, in order to obtain a percentage value of the variant allele frequency. Finally, using the gsl-histogram command it was possible to organise the obtained data in the required output format. In fact, the command takes three arguments, gsl-histogram $[-u]$ $xmin$ $xmax$ $[n]$, specifying the upper and lower bounds of the histogram and the number of bins. It then reads numbers from 'stdin', one line at a time, and adds them to the histogram. If $-u$ is given, histogram is normalised so that the sum of all bins is unity.

C.10 Get_SNV_freq-distr_anyPerc.NORM.sh

```
1 if [ $# -ne 2 ]; then  #checking the correctness of the number of files
    supplied in input
2         echo "Usage: $0 SNV.vcf[.gz] bin_length" >&2 #printing of
    information relating to the type of input requested if those
    provided are not correct
3         echo "Output the normalised GSL-histogram output" >&2 #printing
    of information relating to the type of output returned by the script
4         exit 78
5 fi
6 SNV=$1 #acquisition of the VCF file relating to the Indel variations
7 L=$2 #acquisition of the length of the bin by input
```

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the VCF file relating to the SNV variations has been provided followed by the length of the bin, saved respectively as SNV and L. The algorithm proceeds similarly to what was observed in the case of the Indel variations, retracing step by step all the actions described for the Get_Indel_freq-distr_anyPerc.NORM.sh script.

```
1 eval $(echo $L | awk '{
2                         L=$1 #acquisition of the first (and only)
    argument acquired in input by the "AWK" command that corresponds to
    the length of the bin
3                         N=int(100/$1)+1;  #calculation of the total
    number of bins based on the length of the bin
4                         start=-1*L/2; #definition of the extreme left of
     the first bin
```

```
                        end=100+L/2; #definition of the extreme right of
    the last bin
                        print "N="N";", "start="start";", "end="end";"
   #print of the values corresponding to the number of bins followed by
    the start and the end
                }')
zcat -f $SNV | \
        grep -v '^#' | \        #extraction of all the lines of the VCF
   file taken as input with the exception of those preceded by the
   symbol "#"
        awk 'BEGIN{OFS="\t"}
                {
                        n2=split($(NF),t,":")  #splitting of the string
   of the last column of the VCF file into strings using the ":" as a
   separator, saving these strings in the "t" array
                        print 100*t[n2]
#extraction of the value in the tumour sample corresponding to the
   position of "PM" in the FORMAT, saved in t[n2]. Calculation and
   print of the percentage values of frequencies of SNV variations,
   obtained by multiplying the identified value by one hundred.
                }' | \
        gsl-histogram -u -- $start $end $N
#use of the gsl-histogram command to save data in the required format
```

Through the use of the "awk" command, the length of the bin acquired in input is used to calculate the number of total bins contained in an interval from 0 to 100, which is saved in the variable N. The length of the bin is also used for the definition of the start and end, corresponding respectively to the left end of the first bin and the right end of the last bin outlined. Specifically, these do not correspond directly to the values 0 and 100 as it is advisable for graphic reasons to translate the two positions so that, once the data has been used to create a histogram, the latter is centred and not superimposed on the axis lines. At this point, the values corresponding to the number of bins, the start and the end are printed. Remember in particular that, as previously introduced, the VCF format includes a header, whose lines are identified by the "##" symbols, the 9 mandatory columns identified by the "#" symbol and a body which lists all the variants. In this regard, I proceed by extraction of all the lines of the VCF file taken as input with the exception of those preceded by the symbol "#" (therefore only the lines belonging to the body of the file); for each of these lines splitting of the string of the last column of the VCF file into smaller strings using the ":" as a separator, saving these strings in the "t" array. Differently from what was observed for the ***Get_Indel_freq-distr_anyPerc.NORM.sh***, in this case to extract the frequencies of SNV variations, starting from a VCF format file relating to the SNV variations, it is necessary to consider only one specific field of the FORMAT column, described in the header of the file as follows:

```
##FORMAT= <ID= PM, Number= 1, Type= Float,
Description= "Proportion of mutated allele">
```

I therefore proceed with the extraction of the value in the tumour sample corresponding to the position of "PM" in the FORMAT, saved in $t[n2]$. Subsequently, the percentage values of frequencies of SNV variations are obtained by multiplying the identified

value by one hundred. Finally, using the gsl-histogram command it was possible to organise the data obtained in the required output format, as seen for ***Get_Indel_freq-distr_anyPerc.NORM.sh*** script.

C.11 Get_Indel_freq-distr_anyPerc.sh

The algorithm is identical to that described for Get_Indel_freq-distr_anyPerc.NORM.sh script. The only difference lies in the use of the gsl-histogram command, which was used in the previous cases with the $-u$ feature, which implies the creation of a normalised histogram so that the sum of all bins is unity. In the case of ***Get_Indel_freq-distr_anyPerc.sh*** the command is executed without this feature, consequently the obtained histogram is not normalised.

C.12 Get_SNV_freq-distr_anyPerc.sh

The algorithm is identical to that described for Get_SNV_freq-distr_anyPerc.NORM.sh script. The only difference lies in the use of the gsl-histogram command, which was used in the previous cases with the $-u$ feature, which implies the creation of a normalised histogram so that the sum of all bins is unity. In the case of ***Get_SNV_freq-distr_anyPerc.sh*** the command is executed without this feature, consequently the obtained histogram is not normalised.

C.13 _get_mirror_distribution.py

```
1  if len(sys.argv) != 1+1:  #checking the correctness of the number of
       files supplied in input
2          sys.exit("""
3          Usage: %s distribution #printing of information relating to the
       type of input requested if those provided are not correct
4          """ % sys.argv[0])
5  distribution = [list(map(float,i.strip().split())) for i in open(sys.
       argv[1])]
6  #For each line of the distribution file acquired as input, the line is
       first split using " " as separator; through the python function "map
       ()" the "float()" function is applied (which, starting from a number
        or a string containing decimal points, returns a floating point
       number) to the split line. The decimal numbers obtained are
       subsequently saved in the list called "distribution".
7  x_y = [((i[0]+i[1])/2., i[2], i[0], i[1]) for i in distribution]
8  #For each index in the distribution list the mean value of the bin, the
       frequency associated with the bin, the lower end of the bin and the
       upper end of the bin are indicated in order
9  for i,j in zip(x_y, sorted(x_y, reverse=True)):
10 #Through the zip function all the elements of x_y and of the ordered x_y
        list ( sorted in descending order) are associated obtaining a tuple
        with the elements of the objects according to their position order.
```

```
11          if i[0]<j[0]:  #If the first element of the i-th row of the list
     x_y is less than the first element of the j-th row of the list x_y
    in decreasing order
12                  print(i[2], i[3], j[1])
13 #it prints the second and third element of the i-th row of the list x_y
    followed by the first element of the j-th row of the list x_y
    ordered in decreasing order
14          else: #otherwise
15                  print(i[2], i[3], i[1])
16 #it prints the second, the third and the first element of the i-th row
    of the list x_y
```

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the distribution file relating to the Indel variations or to the SNV variations has been provided. In particular, the input requested by this python script corresponds to the output in the format obtained as a result of the Get_SNV_freq-distr_anyPerc.sh, Get_SNV_freq-distr_anyPerc.NORM.sh, Get_Indel_freq-distr_anyPerc.sh, Get_Indel_freq-distr_anyPerc.NORM.sh, scripts described above. For each line of the distribution file acquired as input, the line is initially split using " " as separator, through the split() function, after it has been deprived of spaces at the beginning and at the end using the strip() function. Through the python function map(), the float() function is applied to the split line (which, starting from a number or a string containing decimal points, returns a floating point number). The three decimal numbers obtained for each line of the distribution file are subsequently saved in the list called "distribution". At this point, for each index in the distribution list the mean value of the bin $(i[0] + i[1])/2$, the frequency associated with the bin $i[2]$, the lower end of the bin $i[0]$ and the upper end of the bin $i[1]$ are saved in order in the list x_y. Through the zip function all the elements of x_y and of the ordered x_y list (sorted in descending order) are associated, obtaining a tuple with the elements of the objects according to their position order. For each i, index of the list x_y, and for each j, index of the ordered list x_y, I ask if the first element of the i-th row of the list x_y is less than the first element of the j-th row of the list x_y in decreasing order: if the answer is yes, the second and third element of the i-th row of the list x_y followed by the first element of the j-th row of the ordered list x_y (corresponding respectively to the right end of the bin and to the frequency relative to the bin, extracted from the first list, and to the left end of the bin, extracted from the second list); otherwise, the second, the third and the first element of the i-th row of the list x_y are printed (corresponding respectively to the right end of the bin, the frequency relative to the bin and the left end of the bin, extracted from the first list).

C.14 _subtract_distribution.py

```
1 if len(sys.argv) != 1+2:   #checking the correctness of the number of
    files supplied in input
2         sys.exit("""
```

```
3         Usage: %s distribution1 distribution2  #printing of information
    relating to the type of input requested if those provided are not
    correct
4         """ % sys.argv[0])
5 d1 = [list(map(float,i.strip().split())) for i in open(sys.argv[1])]
6 d2 = [list(map(float,i.strip().split())) for i in open(sys.argv[2])]
7 #For each line of the distribution1 file acquired as input, the line is
    first split using " " as separator; through the python function map
    (), the float () function is applied  to the split line (which,
    starting from a number or a string containing decimal points,
    returns a floating point number). The decimal numbers obtained are
    subsequently saved in the list called "d1". The same is done for the
     distribution2 file.
8 x_y1 = [((i[0]+i[1])/2., i[2], i[0], i[1]) for i in d1]
9 x_y2 = [((i[0]+i[1])/2., i[2], i[0], i[1]) for i in d2]
10 #For each index in the d1 list the mean value of the bin, the frequency
    associated with the bin, the lower end of the bin and the upper end
    of the bin are indicated in order. The same is done for the list d2
11 if len(x_y1) != len(x_y2):
12         sys.exit("ERROR: distributions have different sizes!")
13 #performing a check on the length of the two lists to verify that the
    two distributions compared have the same size
14 for n,i in enumerate(x_y1):
15 #The enumerate() function is used to iterate through all the elements of
     a list, having both the indices (n) and the values of the list (i)
    as variables to manage.
16         x1 = i[0] # save in x1 the first value contained in the i-th
    value of the x_y1 list (corresponding to the average value of the
    bin of the i-th row)
17 # in x_y2[n][0] there is the first value contained in the n-th value of
    the x_y2 list (corresponding to the average value of the bin of the
    n-th row)
18         if x_y2[n][0] != x1:
19 #performing a check on the average value of the bin to verify that the
    bin exits in distribution2
20                 sys.exit("ERROR: bin not found in distribution2: %s-%s"
    % (i[2], i[3]))
21         diff = i[1]-x_y2[n][1]
22 #calculating the difference between the second value contained in the i-
    th value of the x_y1 list (corresponding to the value of the
    frequency associated with the bin of the i-th row) and the second
    value contained in the n-th value of the x_y2 list (corresponding to
     the value of the frequency associated with the bin of the n-th row)
23         if diff > 0:  #check if the difference is positive
24                 print(i[2], i[3], diff)
25 #print the third and fourth value contained in the i-th value of the
    x_y1 list (corresponding to the extremes of the bin of the i-th row)
     followed by the calculated difference).
26         else:
27                 print(i[2], i[3], 0)
28 #print the third and fourth value contained in the i-th value of the
    x_y1 list (corresponding to the extremes of the bin of the i-th row)
     followed by zero.
```

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that the distribution file relating to the Indel variations (or to the SNV variations), obtained as output of the script get_Indel_freq-distr_anyPerc.sh (or get_SNV_freq-distr_anyPerc.sh), has been provided, followed by the output obtained by _get_mirror_distribution.py by using the distribution file relating to the Indel variations (or to the SNV variations) as input. Once the required distribution files have been acquired as input, for each line of the distribution1, the line is initially split using " " as separator, through the split() function, after it has been deprived of spaces at the beginning and at the end using the strip() function. Through the python function map(), the float() function is applied to the split line (which, starting from a number or a string containing decimal points, returns a floating point number). The three decimal numbers obtained for each line of the distribution file are subsequently saved in the list called $d1$. The same procedure is carried out for each line of the distribution2 file and the results are saved in the $d2$ list. At this point, for each index in the list $d1$ the mean value of the bin $(i[0] + i[1])/2$, the frequency associated with the bin $i[2]$, the lower end of the bin $i[0]$ and the upper end of the bin $i[1]$ are saved in order in the list x_y1. The same procedure is performed for each index in the $d2$ list and the results are saved in the x_y2 list. The lengths of the two lists just described are then calculated and a check is made on them to verify that the two distributions compared have the same size; if I do not have the same dimension, an error is returned as it is not possible to calculate the difference between the distributions provided in input. The enumerate () function is used to iterate through all the elements of the list x_y1, having both the indices (n) and the values of the list (i) as variables to manage. In particular, it is saved in $x1$ the first value contained in the i-th value of the x_y1 list (corresponding to the average value of the bin of the i-th row) while in $x\_y2[n][0]$ there is the first value contained in the n-th value of the x_y2 list (corresponding to the average value of the bin of the n-th row). At this point a check is made to verify that the bin identified in distribution1 exists in distribution2, if this is not verified an error is returned as it is not possible to calculate the difference between the distributions provided in input for that bin. Finally, the difference between the second value contained in the i-th value of the x_y1 list (corresponding to the value of the frequency associated with the bin of the i-th row) and the second value contained in the n-th value is calculated of the x_y2 list (corresponding to the value of the frequency associated with the bin of the n-th row); the third and fourth value contained in the i-th value of the x_y1 list (corresponding to the extremes of the bin of the i-th row) are then printed, followed by the calculated difference if the difference is positive, otherwise by zero.

C.15 Get_freq-distr_anyPerc_from_distribution.NORM.sh

The algorithm proceeds similarly to what was observed in the case of Get_Indel_freq-distr_anyPerc.NORM.sh, retracing step by step all the actions described for the script. Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has

been supplied as input. Specifically, it is verified that the distribution file relating to the Indel variations or to the SNV variations has been provided followed by the length of the bin, saved in $DISTR and $L respectively. In particular, the first input requested by this script corresponds to the output in the format obtained as a result of the _get_mirror_distribution.py and _subtract_distribution.py scripts described above.

```
if [ $# -ne 2 ]; then  #checking the correctness of the number of files supplied in input
        echo "Usage: $0 distribution bin_length" >&2 #printing of information relating to the type of input requested if those provided are not correct
        echo "Output the normalised GSL-histogram output" >&2 #printing of information relating to the type of output returned by the script
        exit 78
fi
DISTR=$1 #acquisition of the VCF file relating to the Indel variations
L=$2 #acquisition of the length of the bin by input

eval $(echo $L | awk '{
                        L=$1 #acquisition of the first (and only) argument acquired in input by the "AWK" command that corresponds to the length of the bin
                        N=int(100/$1)+1;  #calculation of the total number of bins based on the length of the bin
                        start=-1*L/2; #definition of the extreme left of the first bin
                        end=100+L/2; #definition of the extreme right of the last bin
                        print "N="N";", "start="start";", "end="end";" #print of the values corresponding to the number of bins followed by the start and the end
                }')
cat $DISTR | \
        awk 'BEGIN{OFS="\t"}
                {
                        x = ($1+$2)/2 #the average value of the bin ((extreme right + extreme left)/2) is saved in x
                        y = $3 #the value of the variation frequency relative to bin is saved in y
                        for(i=1;i<=y;i++) print x #for each integer i ranging from 1 to y, x is printed
                }' | \
        gsl-histogram -u -- $start $end $N
#use of the gsl-histogram command to save data in the required format
```

As noted above, through the use of the "awk" command, the length of the bin acquired in input is used to calculate the number of total bins contained in an interval from 0 to 100, which is saved in the variable N. The length of the bin is also used for the definition of the start and end, corresponding respectively to the left end of the first bin and the right end of the last bin outlined. At this point, the values corresponding to the number of bins, the

start and the end are printed. Differently from what has been seen for the Get_Indel_freq-distr_anyPerc.NORM.sh script, using the "awk" command on the $DISTR file again, for each line of the file the average value of the bin $(extreme\_right + extreme\_left)/2$ is saved in x and the value of the variation frequency relative to bin is saved in y. At this point, x is printed y times and finally, using the gsl-histogram command it is possible to organise the obtained data in the required output format (normalised histogram).

C.16 Subtract_germline_distribution.SNV.sh

The script ***Subtract_germline_distribution.SNV.sh***, supplied in input the VCF file relating to the metanormal (in the form metaNormal.SNV.vcf [.gz]), a name chosen as the basename of the subsequent outputs of the script and the length of the bin, executes in succession the scripts described above in order to graphically represent the distribution relative to the metanormal, the "subtraction" distribution and the "mirror" distribution, both normalised and non-normalized.
The main body of the algorithm follows in the form of pseudocode.

```
1  I verify that the script has been launched correctly, by introducing a
       check which verifies that what is required for the functioning of
       the script itself has been supplied as input. Specifically, the
       script requests as input the VCF file relating to the metanormal (in
        the form metaNormal.SNV.vcf [.gz]), a name chosen as the basename
       of the subsequent outputs of the script and the length of the bin,
       respectively saved in META, OUT and L_BIN.
2  The name of the directory where the script is executed, saved in BIN_DIR
3  I run the script  $BIN_DIR/get_SNV_freq-distr_anyPerc.NORM.sh providing
       $META $L_BIN as input and saving the output in $OUT.meta.norm
4  I run the script  $BIN_DIR/get_SNV_freq-distr_anyPerc.sh providing $META
        $L_BIN as input and saving the output in $OUT.meta
5  I run the script $BIN_DIR/_get_mirror_distribution.py supplying $OUT.
       meta as input and saving the output to $OUT.mirror
6  I run the script $BIN_DIR/get_freq-distr_anyPerc_from_distribution.NORM.
       sh supplying $OUT.mirror and $L_BIN as input and saving the output
       in $OUT.mirror.norm
7  I run the script $BIN_DIR/_subtract_distribution.py providing $OUT.meta
       and $OUT.mirror as input and saving the output in $OUT.substracted
8  I run the script $BIN_DIR/get_freq-distr_anyPerc_from_distribution.NORM.
       sh providing $OUT.substracted as input and saving the output in $OUT
       .substracted.norm
9  Rappresento in SNV_distribution.norm.SNV.freq-distr_comparison_after-
       subtract.pdf i grafici normalizzati di $OUT.meta.norm $OUT.
       subtracted.norm $OUT.mirror.norm
10 Rappresento in SNV_distribution.raw.SNV.freq-distr_comparison_after-
       subtract.pdf i grafici di $OUT.meta $OUT.subtracted $OUT.mirror
```

C.17 Subtract_germline_distribution.Indel.sh

The script ***Subtract_germline_distribution.Indel.sh***, supplied in input the VCF file relating to the metanormal (in the form metaNormal.Indel.vcf [.gz]), a name chosen as

the basename of the subsequent outputs of the script and the length of the bin, executes in succession the scripts described above in order to graphically represent the distribution relative to the metanormal, the "subtraction" distribution and the "mirror" distribution, both normalised and non-normalized. The algorithm is identical to that described for Subtract_germline_distribution.SNV.sh script therefore it is not reported.

C.18 Recalibrate_HRD_D.sh

The algorithm associated with Strategy D is created by introducing some specific changes to the one seen for Strategy C. In particular, it is possible to notice how the entire flowchart coincides almost perfectly with the one described above for Strategy C, both as regards the Get_SNV_from_freq-distr.py script and Get_indel_from_freq-distr.py. In light of the fact that some changes have been made, in order to adapt the workflow to the new strategy introduced, the new script created in bash will take the name of ***Recalibrate_HRD_D.sh*** (differentiating from that relating to Strategy C, called Recalibrate_HRD_C.sh and from that relating to Strategy A, called Recalibrate_HRD_A.sh). Specifically, the only disparities that emerge between the two scripts made in bash are identified by:

- In the acquisition of input files, associated with the initial section of the algorithm, it is possible to observe for both scripts that the first step is to get the name of the directory where the script is executed, saved in BIN_DIR. From this directory it is possible to go back to the Indel_median-freq_from_N-T and SNV_median-freq_from_N-T files (Table 5.2), contained in the directory itself, corresponding to the reference distribution files in the format as the GSL-histogram output. These respectively represent the median distribution of the SNV and Indel variations resulting from the comparison between the tumour sample and the associated normal, corresponding to the red curve in the graphs previously illustrated. Exporting the described files is followed by verification that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, for the Strategy C script, it is verified that the directory.files has been provided followed by the name of the tumour sample considered, saved respectively as FILES_DIR and TUMOUR. On the other hand, for the script related to Strategy D, it is verified that the directory.files has been provided followed not only by the name of the tumour sample considered but also by $L_BIN, corresponding to the chosen length of the bin.

```
1 BIN_DIR=$(readlink -e $(dirname $0)) #get the name of the directory
    where the script is executed
2 INDEL_NT_DISTRIBUTION=$BIN_DIR/Indel_median-freq_from_N-T#extraction
     of the reference distribution file in the format as the GSL-
    histogram output (median distribution of the Indel variations
    resulting from the comparison between the tumour sample and the
    associated normal)
```

```
3  SNV_NT_DISTRIBUTION=$BIN_DIR/SNV_median-freq_from_N-T #extraction of
       the reference distribution file in the format as the GSL-
     histogram output (median distribution of the SNV variations
     resulting from the comparison between the tumour sample and the
     associated normal)
4  export INDEL_NT_DISTRIBUTION SNV_NT_DISTRIBUTION BIN_DIR
5  if [ $# -ne 3 ]; then  #checking the correctness of the number of
     files supplied in input
6          echo "Usage: $0 directory.files tumour_sample L_bin" >&2 #
     printing of information relating to the type of input requested
     if those provided are not correct
7          echo "Recalibrate SNV and Indel files and recalculate HRD."
     >&2
8          exit 78
9  fi
10 FILES_DIR=$1 #acquisition of the input directory.files (result of
     the Sanger pipeline and the application of filters)
11 TUMOUR=$2  #acquisition of the name of the tumour sample provided in
       input
12 L_BIN=$3  #acquisition of the chosen length of the bin
13 N=50 #definition of the sampling number
14
```

- Always corresponding to the initial section of the script, it is possible to observe for the script of Strategy C the definition of $N = 10$ which corresponds to the definition of the number of sampling performed. In fact, the Recalibrate_HRD_C.sh script foresees the execution of the sampling on the data that describe the distribution obtained from the comparison between the tumour sample and the normal equivalent. In particular, the sub-folders of the FILES_DIR, folder provided as the first input to the script are initially defined: specifically, the two paths FILES_DIR/recalibrateC/Indel-sampling and FILES_DIR/recalibrateC/SNV-sampling are generated. The first step is then performed inside the recalibrateC folder, separately for SNV and Indel. In particular, the sampling_indel function (defined previously) is launched N times receiving as parameters the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.RANDOM-$i.gz (contained in the Indel-sampling folder), where "i" represents the index that is updated at each iteration proceeding from 1 to N. Similarly, the sampling_SNV function (previously defined) is launched N times receiving as parameters the file in the form SNV.*.vcf.gz (contained in FILES_DIR) followed by the SNV.*.vcf.RANDOM-$i.gz file (contained in the folder SNV-sampling). On the other hand, Strategy D differs from the previous one since it sets the value of N equal to 50 (five times the value chosen for Strategy C). The sub-folders of the FILES_DIR folder, provided as the first input to the script, are initially defined: specifically, the two paths FILES_DIR / recalibrateD. $L_BIN/Indel-sampling and FILES_DIR/recalibrateD. $L_BIN/SNV-sampling are generated to differentiate it from recalibrateC, referring to the first strategy). I go to the recalibrateD. $L_BIN folder and the sampling_indel function is launched fifty times receiving as parameters the file in the form Indel.*.vcf.gz (contained in FILES_DIR) followed by the file Indel.*.vcf.RANDOM-1.gz (contained in

the Indel-sampling folder). Similarly, for the sampling_SNV function.

- As regards the central body of the script created for Strategy D, this differs from the one created for Strategy C as it includes a further step. Specifically, in addition to the steps Sampling on N-T distribution, Recreate an updated '.files' directory, Calculate HRD for all RANDOM directories, Recalculate HRD based on median values from datamatrix, previously described, the "step zero" is introduced.

  0. **Get SNV_NT_DISTRIBUTION & INDEL_NT_DISTRIBUTION**

```
I define "SNV" as the file in the form SNV.*.vcf.gz contained in
    the folder $FILES_DIR
I run the script $BIN_DIR/subtract_germline_distribution.SNV.sh
   passing it as parameters the file $SNV followed by "
   SNV_distribution" (name chosen as the basename of the
   subsequent outputs of the script) and the value $L_BIN.
I export SNV_distribution.subtracted saving it in
   SNV_NT_DISTRIBUTION
I define "INDEL" as files in the form Indel.*.vcf.gz contained
   in the $FILES_DIR folder
I run the script $BIN_DIR / subtract_germline_distribution.Indel
   .sh passing it as parameters the file $INDEL followed by "
   Indel_distribution" (name chosen as the basename of the
   subsequent outputs of the script) and by the value $L_BIN.
I export Indel_distribution.subtracted saving it in
   INDEL_NT_DISTRIBUTION
```

  In particular, the introduction of the new step involves the execution of the two scripts created in bash $BIN_DIR/Subtract_germline_distribution.SNV.sh and $BIN_DIR/Subtract_germline_distribution.Indel.sh. The latter, receiving in input respectively $FILES_DIR/SNV.*.vcf.gz, "SNV_distribution" (name chosen as basename relative to the script outputs), $L_BIN and $FILES_DIR/Indel.*.vcf.gz, "Indel_distribution" , $L_BIN, allows the exportation of the distribution (respectively named SNV_NT_DISTRIBUTION and INDEL_NT_DISTRIBUTION) resulting from the subtraction of the "mirror" distribution from the metaNormal one.

C.19 Ensemble_strategy.sh

```
if [ $# -lt 1 ]; then #checking the correctness of the number of files
    supplied in input
        echo "Usage: $0 directory.files [directory.files_2 ....]" >&2 #
    printing of information relating to the type of input requested if
    those provided are not correct
        echo "Collect score for Ensemble Strategy" >&2 #printing of
    information relating to the type of output returned by the script
        exit 6
fi
```

Initially, it is verified that the script has been launched correctly, by introducing a check which verifies that what is required for the functioning of the script itself has been supplied as input. Specifically, it is verified that at least one directory, in the form directory.files, containing the information relating to the application of the different strategies to the patient considered, has been provided.

```
for d in $* #I iterate over each directory.files introduced in input
do
        mkdir $d/recalibrate_Ensemble  #I create the directory in which the result relating to the execution of the Ensemble Strategy will be saved
        RECC=$d/recalibrateC#I save in the variable RECC the path in the form directory.files/recalibrateC, relative to the folder in which the HR score obtained through the execution of Strategy C is saved
        if [ -e "$RECC" ]; then
                ensemble=$(awk '{if(NF>0) score = $NF}END{ensemble=0; if(score<0.30) ensemble=1; if(score>0.70) ensemble=2; print ensemble}' $RECC/HRDetect_fullPipeline.out)
#I extract the HR score from the HRDetect_fullPipeline.out file contained in recalibrateC, I assign the value 2, 1 or 0 to ensemble depending on whether the score is respectively >0.70, <0.30 or between 0.30 and 0.70 (extremes included), according to the criterion explained previously.
        fi
        if [ $ensemble -ne 2 ]; then
#I check the value contained in the ensemble variable, if the value is different from 2 this implies that the Strategy C has predicted a doubtful value (between 0.30 and 0.70) or a value less than 0.30 (HRP), therefore it is possible to apply the Ensemble Strategy to try to get a better result
                RECB=$d/recalibrateB  #I save in the variable RECB the path in the form directory.files/recalibrateB, relative to the folder in which the HR score obtained through the execution of Strategy B is saved
                if [ -e "$RECB" ]; then
                        RB=$(awk '{if(NF>0) score = $NF}END{result=0; if(score<0.30) result=1; if(score>0.70) result=-1; print result}' $RECB/out_HRD/INDEL-SLIDE-5-35.SNV-SLIDE-5-35.HRDetect_fullPipeline.out)
#I extract the HR score from the HRDetect_fullPipeline.out file contained in Strategy B, I assign the value -1, 1 or 0 to RB depending on whether the score is respectively >0.70, <0.30 or between 0.30 and 0.70 (extremes included), according to the criterion explained previously.
                fi
                RECA=$d/recalibrateA #I save in the variable RECA the path in the form directory.files/recalibrateA, relative to the folder in which the HR score obtained through the execution of Strategy A is saved
                if [ -e "$RECA" ]; then
                        RA=$(awk '{if(NF>0) score = $NF}END{result=0; if(score<0.30) result=1; if(score>0.70) result=-1; print result}' $RECA/HRDetect_fullPipeline.out)
```

```
#I extract the HR score from the HRDetect_fullPipeline.out file contained in recalibrateA, I assign the value -1, 1 or 0 to RA depending on whether the score is respectively >0.70, <0.30 or between 0.30 and 0.70 (extremes included), according to the criterion explained previously
                fi
                RECD=$d/recalibrateD #I save in the variable RECD the path in the form directory.files/recalibrateD, relative to the folder in which the HR score obtained through the execution of Strategy D is saved
                if [ -e "$RECD" ]; then
                        RD=$(awk '{if(NF>0) score = $NF}END{result=0; if(score<0.30) result=1; if(score>0.70) result=-1; print result}' $RECD/HRDetect_fullPipeline.out)
#I extract the HR score from the HRDetect_fullPipeline.out file contained in recalibrateD, I assign the value -1, 1 or 0 to RD depending on whether the score is respectively >0.70, <0.30 or between 0.30 and 0.70 (extremes included), according to the criterion explained previously.
                fi
                echo $RB $RA $RD | awk 'BEGIN {FS = " "};{sum=$1+$2+$3+$4}END{if(sum>0) print "HRP";if(sum<0) print "HRD";if(sum==0) print "NA"}' > $d/recalibrate_Ensemble/HRDetect_fullPipeline.out
#I sum the values obtained previously by assigning the "HRD" score if the sum is negative, the "HRP" score if the sum is positive and "NA" (Not Available) otherwise, saving the result in the HRDetect_fullPipeline.out file in the $d/recalibrate_Ensemble folder.
        else #the ensemble value is equal to 2 and the score is >0.70 (HRD)
                        echo "HRD"> $d/recalibrate_Ensemble/HRDetect_fullPipeline.out
        fi
done
```

By iterating on each directory in the form directory.files (relative to a specific sample examined) provided as input, it was possible to obtain for each of them the HR score obtained by using the Ensemble Strategy. Specifically, through the use of the "awk" command, it was possible to extract from the files in the form HRDetect_fullPipeline.out, contained in the recalibrateB, recalibrateC, recalibrateA, recalibrateD folders, respectively, the HR scores obtained as a result of the execution of Strategies B, C, A, D on the samples of the patients considered. As anticipated, since Strategy C is associated with the lowest percentage error, for the execution of the new strategy introduced, it was decided to use the results obtained starting from Strategy C as a reference, in order to improve the predictions made by it. In this regard, the "ensemble" variable is introduced, which is associated with the HR score obtained by performing the Strategy C; in particular, the value 2, 1 or 0 to ensemble is assigned depending on whether the score is respectively >0.70, <0.30 or between 0.30 and 0.70 (extremes included). It follows that the Ensemble Strategy will be applied only if the value contained in the ensemble variable is different from 2, which implies that the score predicted by the Strategy C is doubtful, between 0.30 and 0.70, or that the score predicted is less than 0.30 (HRP). In the other case,

ensemble = 2, the script will simply provide the result obtained through Strategy C, used as a reference, printing that the sample is HRD. Assuming that ensemble takes the value 0 or 1 and therefore the Ensemble Strategy comes into play, the model assigns the value "1" to the HR proficiency (therefore to score values lower than 0.30), the value "-1 "to the HR deficiency (therefore to score values higher than 0.70) while assigning a null value to all scores between 0.30 and 0.70 (extremes included), as outlined above. In this way each strategy will be associated with a numerical value, relative to the prediction made by the strategy itself, which in this case have been named *RB*, *RA*, *RD* (referring respectively to strategies B, A and D). At this point, the values obtained are added and the decision phase takes place: if the result obtained for the single patient sample (ie for the current file directory) is positive, HR proficiency is assigned as prediction, if it is negative, HR deficiency is assigned, otherwise (the result is null) the situation is recognized as doubtful and assigns the value "NA" (Not Available).

Bibliography

[1] *Intramural Research Program*, [Online]. Available: https://irp.nih.gov/our-research/scientific-focus-areas/cancer-biology.

[2] *Cancer.net*, [Online]. Available: https://www.cancer.net/navigating-cancer-care/cancer-basics/what-cancer.

[3] *World Health Organization*, [Online]. Available: https://www.who.int/health-topics/cancer.

[4] "Understanding cancer", *National Institutes of Health (US)*, 2007. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK20362/.

[5] *Reference genome — Wikipedia, the free encyclopedia*, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Reference_genome&oldid=1064522388.

[6] *Gene — Wikipedia, the free encyclopedia*, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Gene&oldid=1070976525.

[7] J. J. Salk, E. J. Fox, and L. A. Loeb, "Mutational heterogeneity in human cancers: Origin and consequences", *PMC*, vol. 5, pp. 51–75, 2010. [Online]. Available: https://doi.org/10.1146/annurev-pathol-121808-102113.

[8] I. A. E. M. van Belzen, A. Schönhuth, P. Kemmeren, and J. Y. Hehir-Kwa, "Structural variant detection in cancer genomes: Computational challenges and perspectives for precision oncology", *npj Precision Oncology*, vol. 5, 2021. [Online]. Available: https://doi.org/10.1038/s41698-021-00155-6.

[9] "SNV", *National Cancer Institute at the National Institutes of Health*, [Online]. Available: https://www.cancer.gov/publications/dictionaries/genetics-dictionary/def/snv.

[10] "SNP", *Nature*, [Online]. Available: https://www.nature.com/scitable/definition/snp-295/.

[11] J. Schütte, J. Reusch, C. Khandanpour, and C. Eisfeld, "Structural variants as a basis for targeted therapies in hematological malignancies", *PubMed Central*, vol. 9, 2019. [Online]. Available: https://doi.org/10.3389/fonc.2019.00839.

[12] "What are single nucleotide polymorphisms?", *National Library of Medicine*, [Online]. Available: https://medlineplus.gov/genetics/understanding/genomicresearch/snp/.

[13] *Biorender*. [Online]. Available: https://biorender.com/.

[14] *Ensembl Genome Browser*, [Online]. Available: https://www.ensembl.org.

[15] *UCSC Genome Browser*, [Online]. Available: https://www.genome.ucsc.edu.

[16] *Chromosome — National Cancer Institute Dictionaries*. [Online]. Available: https://www.cancer.gov/publications/dictionaries/genetics-dictionary/def/chromosome.

[17] B. Weinhold, "Epigenetics: The science of change", *PMC*, vol. 114, A160–A167, 2006. [Online]. Available: https://doi.org/10.1289/ehp.114-a160.

[18] S. Nik-Zainal, P. Van Loo1, D. C. Wedge, L. B. Alexandrov, and C. D. Greenman, "The life history of 21 breast cancers", *Cell*, vol. 149, pp. 994–1007, 2012.

[19] L. B. Alexandrov, S. Nik-Zainal, D. C. Wedge, *et al.*, "Signatures of mutational processes in human cancer", *Nature Medicine*, vol. 500, pp. 415–421, 2013.

[20] *COSMIC,Catalogue Of Somatic Mutations In Cancer*, [Online]. Available: https://cancer.sanger.ac.uk/cosmic.

[21] S. Nik-Zainal, H. Davies, D. Glodzik, *et al.*, "Hrdetect is a predictor of brca1 and brca2 deficiency based on mutational signatures", *Nature Medicine*, vol. 23, no. 4, pp. 517–525, 2017.

[22] M. R Stratton, S. Nik-Zainal et al., H. Davies, *et al.*, "Landscape of somatic mutations in 560 breast cancer whole-genome sequences", *Nature Medicine*, vol. 534, pp. 47–54, 2016.

[23] K. Caldecott, "Single-strand break repair and genetic disease.", *Nature Reviews Genetics*, vol. 9, pp. 619–631, 2008. [Online]. Available: https://doi.org/10.1038/nrg2380.

[24] L. Nguyen, J. W. M. Martens, A. Van Hoeck, and E. Cuppen, "Pan-cancer landscape of homologous recombination deficiency", *Nature Communications*, 2020. [Online]. Available: https://doi.org/10.1038/s41467-020-19406-4.

[25] A. Sfeir and L. S. Symington2, "Microhomology-mediated end joining: A back-up survival mechanism or dedicated pathway?", *PubMed Central*, vol. 40, pp. 701–714, 2015. [Online]. Available: https://doi.org/10.1016/j.tibs.2015.08.006.

[26] *CaVEMan, Cancer Variants Through Expectation Maximisation*, [Online]. Available: http://cancerit.github.io/CaVEMan/.

[27] *Pindel 2.0*, [Online]. Available: http://cancerit.github.io/cgpPindel/.

[28] *BRASS, BReakpoint AnalySiS Algorithm*, [Online]. Available: https://github.com/cancerit/BRASS.

[29] J. Brownlee, *Ensemble Learning Algorithms With Python: Make Better Predictions with Bagging, Boosting, and Stacking*. Machine Learning Mastery, 2021. [Online]. Available: https://books.google.it/books?id=IUkrEAAAQBAJ.

[30] D. Sims, I. Sudbery, N. E. Ilott, A. Heger, and C. P. Ponting, "Sequencing depth and coverage: Key considerations in genomic analyses", *PubMed Central*, vol. 15, pp. 121–132, 2014. [Online]. Available: https://doi.org/10.1038/nrg3642.

[31] A. Bardelli, S. Arena, G. Corti, *et al.*, "A subset of colorectal cancers with cross-sensitivity to olaparib and oxaliplatin", *Clinical Cancer Research*, 2019. [Online]. Available: clincancerres.aacrjournals.org.

[32] P. Rawla, T. Sunkara, and A. Barsouk, "Epidemiology of colorectal cancer: Incidence, mortality, survival, and risk factors", *PubMed Central*, vol. 14, pp. 89–103, 2019. [Online]. Available: https://doi.org/10.5114/pg.2018.81072.

[33] E. Dekker, P. J Tanis, J. L A Vleugels, P. M Kasi, and M. B Wallace, "Colorectal cancer", *PubMed Central*, vol. 394, pp. 1467–1480, 2019. [Online]. Available: https://doi.org/10.1016/S0140-6736(19)32319-0.

[34] E. J. Kuipers, W. M. Grady, D. Lieberman, and T. Seufferlein, "Colorectal cancer", *PubMed Central*, vol. 1, 2015. [Online]. Available: https://doi.org/10.1038/nrdp.2015.65.

[35] J.-P. Gillet, S. Varma, and M. M. Gottesman, "The clinical relevance of cancer cell lines", *PubMed Central*, vol. 105, pp. 452–458, 2013. [Online]. Available: https://doi.org/10.1093/jnci/djt007.

[36] Q. Guo, E. Lakatos, I. Al Bakir, K. Curtius, T. A. Graham, and V. Mustonen, "The mutational signatures of formalin fixation on the human genome", *bioRxiv*, 2021. [Online]. Available: https://www.biorxiv.org/content/early/2021/03/12/2021.03.11.434918.

[37] "FASTQ format — Wikipedia, the free encyclopedia", 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=FASTQ_format&oldid=1039439009.

[38] "SAM format — Wikipedia, the free encyclopedia", 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=SAM_(file_format)&oldid=1020226558.

[39] "Variant Call Format — Wikipedia, l'enciclopedia libera", 2021. [Online]. Available: http://it.wikipedia.org/w/index.php?title=Variant_Call_Format&oldid=121487763.

[40] H. Li, B. Handsaker, A. Wysoker, *et al.*, "The sequence alignment/map format and samtools.", *Bioinformatics*, vol. 25, pp. 2078–2079, 2009. [Online]. Available: https://doi.org/10.1093/bioinformatics/btp352.

[41] M. Liverani, "Programmazione della shell Bash", 2011. [Online]. Available: https://www.aquilante.net/doc/bash_programming.pdf.

[42] D. Barlow Close, A. D. Robbins, P. H. Rubin, R. Stallman, and P. van Oostrum, "The AWK Manual", 1995. [Online]. Available: https://www.cs.unibo.it/~renzo/doc/awk/nawkA4.pdf.

[43] H.-P. Halvorsen, "Python Programming", 2019. [Online]. Available: https://www.halvorsen.blog/documents/programming/python/resources/Python%5C%20Programming.pdf.

Acknowledgements

In primo luogo vorrei ringraziare il mio relatore, il Professore Luigi Preziosi, per avermi offerto la possibilità di svolgere questo lavoro di ricerca mettendomi in contatto con l'Istituto di Candiolo IRCCS e soprattutto per la pazienza e disponibilità che ha dimostrato nei miei confronti.

Ringrazio il Professore Alberto Bardelli, Direttore del Laboratorio di Oncologia Molecolare dell'Istituto di Candiolo IRCCS, per avermi concesso questa grande opportunità, permettendomi di confrontarmi con un mondo completamente diverso da quello da cui provenivo.

In particolar modo ringrazio il Dott. Giorgio Corti, per avermi accompagnato in questo percorso, per l'attenta presenza e l'aiuto costante durante tutti questi mesi. Lo ringrazio per tutto il tempo che mi ha dedicato, per i fondamentali spunti e guide che mi hanno permesso di iniziare, continuare e concludere il lavoro.

Un ringraziamento speciale è rivolto a tutti coloro che mi sono stati accanto durante l'intero periodo universitario. Persone con cui ho condiviso questa esperienza sin dal primo giorno, persone incontrate durante il percorso ed altre che hanno intrapreso strade diverse.

Ringrazio Adele e Stefania, perché prima di incontrarle non sapevo cosa fosse davvero l'amicizia. Insieme abbiamo riso, pianto, ballato, frequentato tutti gli ospedali del circondario, pianto, imparato a dire "ti voglio bene" (o forse no), bevuto shottini troppo cari e vini troppo economici, ho già detto pianto? Qualcuno diceva che fosse fondamentale sedersi affianco alle persone giuste il primo giorno di università, perché quelle sarebbero state per la vita. Sei anni fa questa affermazione mi spaventò, oggi più che mai spero che sia vera e che questo sia solo l'inizio delle avventure de "Le sopravvissute".

Ringrazio Francesca, per avermi capita, anche più di quanto non abbia mai fatto io. Non c'erano mali che un piatto di pastina, un gelato alla Romana ed una canzone di Ana Mena non potessero curare. Grazie per avermi sempre fatto sentire a casa.

Ringrazio Matteo, Vincenzo, Gianmarco e Riccardo che, in qualità di migliori coinquilini di un'altra casa del mondo, hanno reso Corso Montevecchio 66 un posto indimenticabile.

Un ringraziamento particolare va ai miei compagni di corso, le persone con cui ho condiviso attimi di gioia e di tristezza, con il rammarico di non aver potuto concludere insieme gli ultimi semestri del nostro percorso.
Grazie a Emanuele, Giovanni, Erika e Luca, per aver alleggerito le giornate più pesanti, tra riso ammottato, chili di farinata e caffè discutibili.

Grazie a mia zia Marilena, per tutti i consigli e per le parole giuste al momento giusto. Grazie, perché in tutti questi anni c'è sempre stata.

Grazie a Francesco, per essere stato più e più volte la mia luce in fondo al tunnel. Penso e ripenso ma non trovo le parole giuste per descrivere quello che è stato, ed è, per me. Lo ringrazio per avermi assecondato e sostenuto in ogni mia folle decisione, per aver creduto in me senza se e senza ma, per aver condiviso ogni giorno con me gioie, sacrifici e successi e per avermi spronato in ogni momento a superare i miei limiti. Questo traguardo è anche un po' suo.

Un ringraziamento speciale è rivolto alla mia famiglia.
A mia mamma e a mio papà, perché a loro devo tutto. Per la fiducia che hanno riposto in me, per essermi sempre stati accanto e non avermi mai fatto mancare il loro sostegno.
A mia sorella Anna, per essere stata presente in ogni scelta importante, per avermi supportato, e soprattutto sopportato, nei momenti peggiori. Non mancherò di ripeterle quanto le voglia bene e quanto sia importante per me.

Infine, voglio dedicare questo grande traguardo ai miei nonni, Nitta e Batti, so che sarebbero fieri di me e di ciò che sono diventata. Avranno sempre una parte del mio cuore.