UNIVERSITA' DEGLI STUDI DI PAVIA

FACOLTA' DI INGEGNERIA
DIPARTIMENTO DI INGEGNERIA INDUSTRIALE E DELL'INFORMAZIONE

DOTTORATO DI RICERCA IN BIOINGEGNERIA E BIOINFORMATICA
XXVII CICLO - 2014

CLOUD-BASED BIOINFORMATIC ANALYSIS OF NGS DATA IN CHRONIC MYELOMONOCYTIC LEUKEMIA

PhD Thesis by
ETTORE RIZZO

Advisor:
Prof. Riccardo Bellazzi

PhD Program Chair:
Prof. Riccardo Bellazzi

The roots of education are bitter,
but the fruit is sweet.

Aristotle

Abstract (English)

The high demand for low-cost sequencing has brought to develop high-throughput sequencing technologies, also known as next-generation sequencing, which process concurrently thousands or millions of sequences. After sequencing, researchers have to deal with raw sequencing data, representing short fragments of DNA or RNA, which have to be conveniently processed to quickly turn samples into results. However, few other scientific instruments generate as much data as next-generation sequencing ones and typical output volumes make sequencing a big data problem. Bionformaticians play therefore a lead role to unlock the full value of next-generation sequencing data sets and there is an increasing need for such specialists.

Typically, in order to extract biological information from the huge amounts of data produced, the basic steps in a bioinformatic workflow consist in translating raw data into short reads, mapping reads to a reference genome, performing protocol specific analysis and reporting results. Current informatic sources for next generation sequencing data are however extremely fragmented and often not well documented since most of the time are open-source tools made by researchers and rarely commercial. In order to define a valuable workflow, or pipeline, a bioinformatician has to choose between several tools that perform similar task and then accurately merge them to automate the end-to-end process. This can be done manually with a considerable amount of IT customization or through several integrative platforms developed to facilitate the pipeline assembly process, and furthermore, to record analysis metadata, including the tools, versions and parameter settings in order to enable reproducibility of experiments. As a rule of thumb, the whole pipeline of analysis for a sequencing run will require around three times the size of the instrument output but such a data deluge can adversely affect these platforms that may not necessarily be the best choice.

As well as storage consuming, next generation sequencing projects can be highly computational demanding and often sequencing laboratories aren't conveniently equipped to face both problems. Moreover computational needs are typically variable so the right sizing of high computing systems to allocate for next generation sequencing projects is not trivial. For a while now cloud

computing technologies have been used to overcome such limitations and are becoming every day more popular within the bioinformatic community.

Within this frame, this thesis work aims at studying the effects of Chronic Myelomonocytic Leukemia on a cohort of patients that have been characterized by high throughput sequencing on thirty-eight genes. In order to do that, all the informatic criticalities related with next generation sequencing, which have been previously introduced, will be addressed.
The first chapter starts with a general overview of such bioinformatic field and highlight the challenges that therein lies. Moreover the concept of targeted analysis is discussed and the sequencing experiment introduced.
Chapter 2 deepens the rational of the sequencing study by characterizing the myeloid neoplasm under investigation and then specifies how the used sequencing technology works.
Chapter 3 gets into file formats and algorithms selected to define the pipeline of analysis that has been used for the experiment. The pre-processing section details how reads are mapped against the human reference genome and processed to minimize the number of mismatching bases across them. Then the variant discovery section focuses on algorithms that scan samples for identifying regions that are different from the reference genome and therefore interesting. Finally functional annotation discusses how to enrich variants with important additional features.
In Chapter 4 two genomic workflow management systems that have been used to implement the pipeline are presented. The first one, GenePattern has been installed on a system of high performance computing while the second one, Cosmos, has been implemented on the cloud. A detailed comparison will report advantages and disadvantage of both platforms and a cloud approach will be compared with a classical one.
Chapter 5 and 6 will present two additional work developed during the PhD program. In particular Chapter 5 describes an efficient and cheap genotyping pipeline implemented in the cloud while Chapter 6 discusses an algorithm to classify NGS short reads by their allele origin.
Finally Chapter 7 summarizes the procedure to select the most interesting variants above the whole set of variants identified on the patient set and highlights the results while Chapter 8 draws the overall conclusion of this dissertation.

Abstract (Italian)

L'esigenza crescente di tecniche di sequenziamento a basso costo ha portato allo sviluppo di tecnologie definite high-throughput, o anche next-generation, che elaborano simultaneamente migliaia o milioni di sequenze. Una volta effettuato il sequenziamento, i ricercatori devono gestire opportunamente i dati grezzi in output, che rappresentano brevi frammenti di DNA o RNA, e processarli in accordo con gli obiettivi dello studio per trasformare i campioni analizzati in risultati. Tuttavia, solamente pochi altri strumenti scientifici generano quantita' di dati paragonabili ai sequenziatori di nuova generazione e i volumi tipici di produzione rendono il sequenziamento un problema di Big Data. In questo contesto i bioinformatici svolgono un ruolo guida per l'interprestazione dei dataset tipici del sequenziamento next-generation e vi e' una crescente necessita' di questi specialisti.

In generale, al fine di estrarre informazioni biologiche dalle enormi quantita' di dati prodotti, i passaggi fondamentali di un workflow di analisi bioinformatica consistono nel tradurre i dati grezzi in stringhe definite reads, mappare queste ultime ad un genoma di riferimento, eseguire analisi specifiche e presentare i risultati. Tuttavia gli strumenti informatici necessari per analizzare i dati di sequenziamento di nuova generazione sono estremamente frammentati e spesso non ben documentati, in quanto il piu' delle volte sono strumenti open source creati da ricercatori universitari e raramente prodotti commerciali. Per definire un flusso di lavoro robusto e accurato, un bioinformatico deve quindi scegliere tra diversi tool che svolgono funzioni analoghe e unirli in modo opportuno per formare una pipeline di analisi che automatizzi l'intero processo. Questo puo' essere fatto manualmente con un considerevole sforzo in termini di customizzazioni IT oppure e' possibile avvalersi di diverse piattaforme che facilitano il processo di creazione di worflow di analisi, e, inoltre, durante l'esecuzione registrano i metadati, come i tool utilizzati, le versioni specifiche e i parametri inseriti, il tutto per consentire la riproducibilita' degli esperimenti. Come regola generale, l'intera pipeline di analisi per una corsa sequenziamento richiede pero' circa tre volte la dimensione dell'ouput dello strumento, e una tale quantita' di dati puo' mettere in crisi le piattaforme sopracitate le quali non risultano essere necessariamente la scelta migliore.

Inoltre, oltre ad essere estremamente onerosi dal punto di vista dello storage necessario, i progetti di sequenziamento next-generation possono essere estremamente esigenti dal punto di vista computazionale e la maggior parte dei laboratori che si occupa di sequenziamento non e' attrezzata per affrontare entrambi i problemi. Per di piu' le esigenze di calcolo sono estremamente variabili e il corretto dimensionamento della strumentazione informatica da allocare per un progetto di sequenziamento non e' facilmente stimabile. Da un po' di tempo a questa parte le tecnologie di cloud computing sono state utilizzate per superare tali limiti e stanno diventando ogni giorno piu' diffuse nel campo della bioinformatica.

In questo contesto, il lavoro di tesi si propone di studiare gli effetti della leucemia mielomonocitica cronica su una coorte di pazienti di cui sono stati sequenziati tramite next-generation trentotto geni. Nel corso del lavoro verranno affrontate le criticita' informatiche precedentemente introdotte connesse a tale progetto.
Il Capitolo 1 fornisce una panoramica generale su questo campo della bioinformatica ed evidenzia le sfide tecnologiche ad esso correlate. Successivamente viene introdotto il concetto di analisi targettata poiche' su di essa si basa l'intero progetto di sequenziamento.
Il Capitolo 2 approfondisce il razionale dello studio caratterizzando la neoplasia della linea mieloide oggetto della ricerca e descrivendo il funzionamento della tecnologia utilizzata.
Il Capitolo 3 entra nel dettaglio dei formati di file e degli algoritmi selezionati per definire la pipeline di analisi utilizzata per processare i campioni. Nella sezione pre-processing viene approfondito il concetto di mappatura contro il genoma umano di riferimento e i passaggi necessari per ridurre il numero di basi che non coincidono con tale riferimento. In seguito la sezione variant discovery descrive gli algoritmi per la scansione dei campioni che si prefiggono di identificare regioni diverse dal genoma di riferimento e quindi interessanti. Infine tramite l'annotazione funzionale le varianti individuate vengono arricchite con importanti informazioni aggiuntive.
Nel Capitolo 4 sono presentati due sistemi di gestione del flusso di lavoro che sono stati utilizzati per realizzare la pipeline di analisi. Il primo, GenePattern e' stato installato su un su un sistema di calcolo ad alte prestazioni, mentre il secondo, Cosmos, e' stato implementato sul cloud. Un confronto dettagliato riporta quindi vantaggi e svantaggi di entrambe le piattaforme e confronta l'approccio cloud con quello classico.
Nel Capitolo 5 e 6 vengono presentati due lavori supplementari sviluppati durante il corso del dottorato. In particolare il Capitolo 5 descrive una pipeline di genotipizzazione efficiente ed economica implementata sul cloud, mentre il Capitolo 6 presenta un algoritmo per classificare le reads NGS sulla base della loro origine allelica.
Infine il Capitolo 7 riassume la procedura per selezionare, a partire dall'intero dataset di varianti individuate durante lo studio di sequenziamento, le varianti

a carattere somatico piu' interessanti ed evidenzia i risultati ottenuti mentre il Capitolo 8 trae le conclusioni del lavoro di tesi qui presentato.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Next-generation sequencing (NGS) is the latest technological milestone in the biological sciences and is changing the direction in biomedical science including cancer research. By definition it describes those recent technologies that provide a much cheaper and higher-throughput alternative to sequencing DNA than traditional Sanger sequencing.

Sanger sequencing, otherwise known as first generation sequencing, was the most widely used sequencing method since the late 70's [1] and through the Human Genome Project (HGP), which remains the world's largest collaborative biological project [2], led to determine the sequence of base pairs that define human DNA. However, despite this tremendous achievement and a great accuracy, Sanger sequencing has been replaced by second generation sequencing (SGS) techniques since early 2000s. These new technologies can indeed massively parallel sequence millions of DNA templates and overcome issues about tiny and costly throughput. On the horizon, other promising technologies defined as third generation sequencing (TGS), promise even higher throughput, faster turnaround time, longer read lengths, higher consensus accuracy, small amounts of starting material, and lower cost [3]. Both SGS and TGS are used synonymously to NGS and have been possible by innovations in sequencing chemistries, image processing, microfabrication and information technology [4].

These progresses have already changed the way we thought about DNA introducing the idea of population sequencing, an initial discovery phase that aims to uncover novel genes, pathways, and mutational processes implicated in a disease. Since 2004, the US National Human Genome Research Institute (NHGRI) has founded a series of project to reach the goal of sequencing a human genome for \$1000 or less. This target has been considered as a technological breakthrough to enable the huge sequencing studies that could lead to major discoveries in personalized medicine and cancer research. During the last decade the throughput of NGS sequencers has increased at a rate that outpaces Moore's law, more than doubling each year. Simultaneously the cost for raw megabase of DNA sequence has dropped from 1000\$ in 2004 to 0.1\$ in

2014 [5] and the latest Illumina product (HiSeq X Ten) seems to be able to beat the challenge previously defined. Nowadays, as a result, several genomes per week can be sequenced simultaneously and targeted DNA enrichment methods allow even higher genome throughput at a reduced cost per sample.

Scientists have therefore embraced NGS technology in order to better understand the malignant tumor genomes. Despite cancer initiation, promotion, and progression mechanisms are not yet fully understood, the fact that cancer is basically a genomic diseases is well established. As a consequence, several collaborative projects, such as The Cancer Genome Atlas (TCGA) [6] and the International Genome Consortium (ICGC) [7] aim to catalogue mutations in different cancer types. NGS could be then used for genetic diagnostic screening to detect inherited and somatic mutations in cancer genes [8, 9] while new biomarkers can accelerate the research process both identifying novel targets for drug development [10] and guiding the therapy using existing drugs. In order to detect somatic alterations such as nucleotide substitutions, insertions, deletions, copy number variations, and chromosomal rearrangements, several approaches can be adopted. While whole-genome sequencing (WGS) is likely to be few years away because costs are still high, whole-exome sequencing (WES) and transcriptome sequencing (RNA-Seq) as well as targeted sequencing of multiple specific genomic regions are extremely attractive. Nevertheless to get a significant statistical correlation between genomic variations and cancer phenotype, a relevant amount of samples is necessary. Targeted sequencing can be therefore a good compromise to define new insights over a sample representative of a population: once regions of interest are defined, basing on literature findings or other insights, it lowers costs and increases sensitivity compared to the whole-genome or exome approaches. Finally, the high complexity observed in cancer genomes suggests that the understanding of how cancer genes knit together, how they condition the disease evolution and possibly the clinical phenotype, can be only achieved through comprehensive analysis of large cohorts of well-characterized patients.

1.1 NGS data challenges in bioinformatics

The explosive growth of sequencing data gave life to several sequence archives that freely share sequencing data within the scientific community. An example is the Short Read Archive (SRA) that archives raw sequencing data since 2007 and nowadays stores more than 100TB of data [11]. However, despite this massive increase in available raw sequence data, there are several informatics challenges that must be addressed [12] in order to fulfill the growing cancer genomics promises.

Depending on the sequencing platform and instrument the output of an experiment can be up to hundreds of GB of raw data. Consider that a small project with 10 to 20 whole genome sequencing data can generate up to 4TB of raw data, the management of the amounts of data can therefore be seen as a “big data problem” [13] where storing and moving data is not a trivial task. Raw sequence data have then to be conveniently processed in order to identify causative variations or other interesting findings so the demand of statistical methods and bioinformatic tools to analyze NGS data has incredibly magnified. These tools span several categories including alignment of sequence reads to a reference, de novo assembly, base-calling and/or structural variant detection, annotation of detected variants, visualization of genomic data and many others.

Due to the availability of a wide array of data formats, software and analytical tools, there is no general solution to apply but the adoption of already defined pipeline should be strongly promoted. This would allow to better reproduce already published experiments, which could be combined into larger predictive studies. Unfortunately there are few current study that record exact details of their computational experiments [14]. The building process of a robust data analysis workflow to mine, analyze and interpret the NGS data, requires therefore to select the right computational tools for each step and technical informatics skills. Often, researchers of small biology labs have no computational experiences and the analysis of these data could be really challenging.

Finally, in order to store and process NGS data, sufficient computing facilities are needed. Despite a single lab usually owns some computing hardware, the computational demand of several bioinformatic tools can’t always be satisfied. As a consequence, high performance computing (HPC) resources, as cluster or server solutions, should be addressed to NGS analysis; this usually means additional expensive investments. Furthermore, their correct sizing can be very problematic because it requires to meticulously determine what is the expected sequencing throughput in order to define the storage capacity and what are the requirements (in terms of RAM or CPU) of software that will be run as a part of the analysis workflow.

Recently, cloud computing solutions have rapidly grown in number [15] to overcome these issues. Several processing frameworks that use cloud computation capacity have made many pipeline of analysis more accessible to the scientific community. Firstly, the possibility to “pay as you go” the required computation resources has partially reduced costs: this paradigm doesn’t require big initial investment. Additionally, a scrupulous sizing of the compute capacity is not anymore necessary because of the possibility to easily resize it depending on own needs.

1.2 NGS-based multigene mutational analysis

In 2012, motivated by this vibrating context, the Department of Hematological Oncology of Policlinico San Matteo (Pavia, Italia) has bought an Illumina MiSeq desktop sequencer. This choice was motivated both by technical and biological considerations. Illumina’s sequencing by synthesis (SBS) is indeed the state of art of sequencing technology because of its accuracy. This technology will be discussed in detail later but it is mainly based on fluorescently labeled reversible terminator that enable the identification of single bases as they are introduced into DNA strands. Furthermore, it guarantees excellent results in terms of run time, sample preparation's simplicity and high coverage throughput for targeted and small genome sequencing.

Although whole-genome or whole-exome applications reach the higher predictive power, the MiSeq instrument does not guarantee such an high throughput. Nevertheless an approach as targeted panels may offer further advance in routine molecular diagnostics of cancer by expanding the already existing diagnostic panels of several genes. Based on that, the first study conducted on MiSeq evaluated targeted next-generation sequencing as a new approach for testing a broad spectrum of mutations. Recently, recurrent somatic mutations have been indeed identified in Chronic Myelomonocytic Leukemia. As a consequence, a panel of genes implicated in myeloid malignancies was defined and it was enhanced with additional genes that were supposed to be related with this cancer type but not already well characterized. Hence, the aim of this targeted analysis on a selected subset of genes was to provide additional useful prognostic information and the results will be deeply discussed in the dissertation.

The research activity described in this thesis will deal with the bioinformatic aspects of this study. The first research direction is focused on the definition and implementation of an efficient workflow of analysis for NGS data. Usually referred as pipeline, it can be seen as a series of step that aims to transform raw sequencing data into useful information. Its definition has started off with different software comparison in order to select the most reliable ones. Once the most appropriate have been selected, the assembly was done through different workflow management systems: these implementing solutions allow for formal description of a pipeline in order to automate and serialize all the steps required to analyze these kind of genomic data. An efficient solution required several optimizations of the technical aspects regarding data transfer, management and elaboration. Subsequent efforts have then converged to technical solutions that deal with the sporadic availability of the data. MiSeq's sequencing rate has not indeed been constant over time. On the contrary, the amount of data to analyze has had critical peaks that stressed the modest compute resources at our disposal and resulted in long time for data processing. The potentialities of Cloud-based solutions have been therefore investigated and implemented. A rich part of this dissertation will focus on the result achieved by these computing infrastructures that overcome variable requirements of computational needs.

Chapter 2

Clinical and technological background

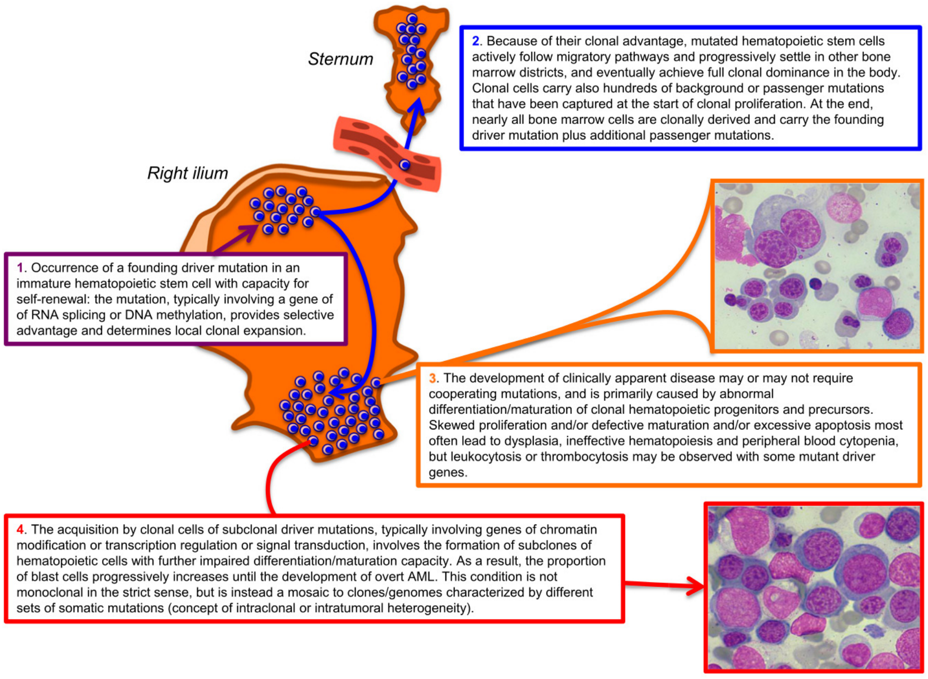
This chapter aims to give a general picture about the disease that has been investigated by the sequencing project and the technological aspects that characterize the MiSeq instrument. After a background section in which the myelodysplastic and myeloproliferative neoplasms will be introduced, details about Illumina's sequencing technology will be provided. A final section will concentrate on the ad-hoc target panel that was defined for the study and will examine the fully customizable, amplicon-based assay that was used to prepare sequencing samples.

2.1 Myelodysplastic/Myeloproliferative Neoplasms

According to the World Health Organization (WHO) classification of tumors of hematopoietic and lymphoid tissues [16] there are 5 subgroups of myeloid neoplasms: myeloproliferative neoplasms (MPNs); myeloid/lymphoid neoplasms with eosinophilia and abnormalities of PDGFRA, PDGFRB, or FGFR1; myelodysplastic syndrome (MDS); myelodysplastic/myeloproliferative neoplasms (MDS/MPNs); and acute myeloid leukemia (AML).

The NGS-based multigene mutational analysis that will be discussed focused on MDS/MPNs: this subtype was defined by Vardiman et al [17] as "clonal myeloid neoplasms that at the time of initial presentation have some clinical, laboratory or morphologic findings that support a diagnosis of myelodysplastic syndrome (MDS), and other findings more consistent with myeloproliferative neoplasm (MPN)". In particular, this study was performed on 214 patients affected by chronic myelomonocytic leukemia (CMML), a particular MDS/MPNs type characterized by a highly variable clinical course.

The interesting model shown in Fig. 2.1 summarizes the current concepts of



2. Because of their clonal advantage, mutated hematopoietic stem cells actively follow migratory pathways and progressively settle in other bone marrow districts, and eventually achieve full clonal dominance in the body. Clonal cells carry also hundreds of background or passenger mutations that have been captured at the start of clonal proliferation. At the end, nearly all bone marrow cells are clonally derived and carry the founding driver mutation plus additional passenger mutations.

1. Occurrence of a founding driver mutation in an immature hematopoietic stem cell with capacity for self-renewal: the mutation, typically involving a gene of of RNA splicing or DNA methylation, provides selective advantage and determines local clonal expansion.

3. The development of clinically apparent disease may or may not require cooperating mutations, and is primarily caused by abnormal differentiation/maturation of clonal hematopoietic progenitors and precursors. Skewed proliferation and/or defective maturation and/or excessive apoptosis most often lead to dysplasia, ineffective hematopoiesis and peripheral blood cytopenia, but leukocytosis or thrombocytosis may be observed with some mutant driver genes.

4. The acquisition by clonal cells of subclonal driver mutations, typically involving genes of chromatin modification or transcription regulation or signal transduction, involves the formation of subclones of hematopoietic cells with further impaired differentiation/maturation capacity. As a result, the proportion of blast cells progressively increases until the development of overt AML. This condition is not monoclonal in the strict sense, but is instead a mosaic to clones/genomes characterized by different sets of somatic mutations (concept of intraclonal or intratumoral heterogeneity).

Figure 2.1: **Schematic representation of the current understanding of the pathophysiology of myelodysplasia** Reproduced from [18]

myelodysplasia's pathophysiology [18]. Some steps of this model are still working hypotheses, but a similar clonal architecture has been recently observed also in patients with CMML [19]. A somatic driver mutations that occur in an immature hematopoietic stem cell can provide growth and survival advantages that lead to formation of a local clone (Fig. 2.1, step 1). The mutated stem cells must have additional advantages to let this clone become fully dominant in the whole body, however the mechanism by which hematopoietic stem cells settle in other bone marrow district is still unclear [20]. Once the neoplastic clone has become fully dominant in the bone marrow, the patient may or may not develop the disease. The development of clinically apparent disease is usually correlated with additional cooperating mutations.

An alternative representation that fit for CMML's evolution is shown in Fig. 2.2 [21]. During their lifetime, hematopoietic stem cells accumulate different age-dependant mutations represented by X. When initiating mutations occur, the affected cell gains a competitive advantage and the subsequent clonal expansion captures all of the preexisting mutations. Several progression events give the expanding clone other additional advantages; each progression mutation captures all the mutations that occurred between the initiating event and the progression event and all of them are designated as Y. Cells with appropriate progression events result in CMML which is defined by the founding clone, designated in red. Finally, subclones may acquire additional mutations represented as Z.

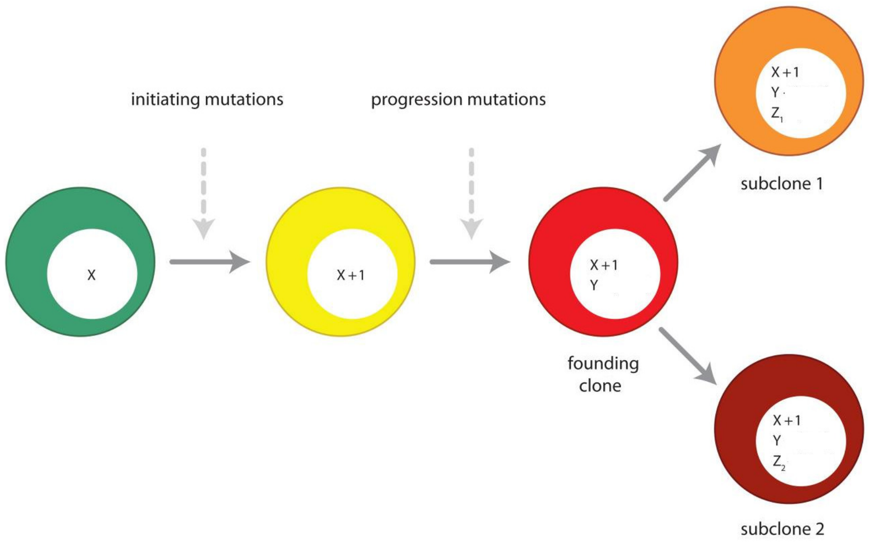


Figure 2.2: **Model for the origin of driver and passenger mutations during CMML evolution** X: age-dependant passenger mutations; Y: passenger mutations gained between initiating and cooperating mutations; Z: passenger mutations gained during progression to subclones

CMML affects mainly old adults which median age at presentation is ≈ 70 years. Its incidence is lower than 1 case per 100.000 persons per year so it can be considered a rare disorder [22]. There are several mutations involved in the disease that are well characterized and recent studies are adding new insight about the molecular basis of CMML. While NRAS or KRAS mutations have been known as molecular abnormality in CMML for a long time, TET2, CBL, ASXL1, RUNX1, EZH2 and SRSF2 mutations have been later identified in a significant number of CMML patients [23]. After all, despite some unclear details, the molecular basis of the disease can be considered relatively well-defined, involving primarily somatic mutations of TET2 and SRSF2: co-mutation of these genes is almost invariably associated with CMML, whereas the ASXL1 mutation involves poor outcome. These are indeed the most frequently mutated genes and the proportion of patients that carry mutations on these genes are respectively from 50 to 60% for TET2, from 40 to 50% for SRSF2 and from 30 to 40% for ASXL1 [24]. Table 2.1 is based on published studies [18] and reports a comprehensive list of several well known mutant genes in patients with MDS or MDS/MPNs. Several mutations are shared across the spectrum of subtypes so these genes have been added to the NGS targeted panel designed to screen a large cohort of CMML patients and some of the related controls.

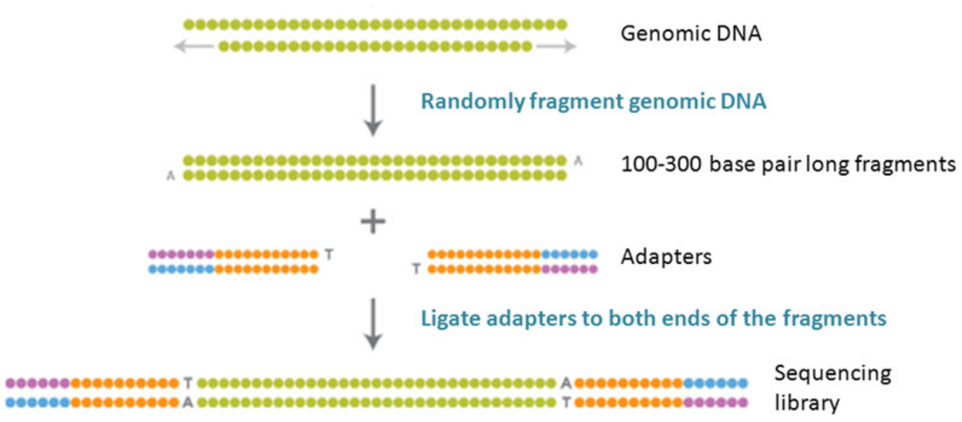

Genomic DNA
Randomly fragment genomic DNA
A
A
100-300 base pair long fragments
+
T
T
Adapters
Ligate adapters to both ends of the fragments
T
A
A
T
Sequencing library

Figure 2.3: **Illumina Library Preparation**

2.2 Illumina Sequencing Technology

Sequencing of CMML patients was performed by Illumina MiSeq. This desktop sequencer can produce 2 x 300 paired-end reads and generates approximately 9Gb or raw sequencing data on a single run. It is specifically designed for small genome sequencing or targeted resequencing studies and relies on Illumina sequencing technology.

2.2.1 Library preparation

NGS library preparation starts with shearing genomic DNA into a random library of fragments that are usually long from 100 to 300 base-pair. After repairing ends and adding A overhang, platform-specific oligonucleotide adapters, which are necessary for amplification and sequencing, are ligated to both ends of the DNA fragments. These ligated fragments are then size selected and purified to generate a sequencing-ready library. The overall process is displayed in Fig. 2.3.

2.2.2 Cluster Generation

The major innovation of the Illumina technology is the amplification of template fragments. This step, defined as cluster generation is performed on a proprietary solid surface, defined flow cell. Once the library is created, it is flowed across the flow cell and sequencing templates are immobilized by hybridizing to oligos on its surface that are complementary to the ligated adaptors. Following this, a solid phase amplification (Fig. 2.4), also called bridge amplification, creates approximately up to 1,000 identical copies of each single template in close proximity. These copies result in several million dense clusters of double-stranded DNA in each channel of the flow. Finally, the reverse

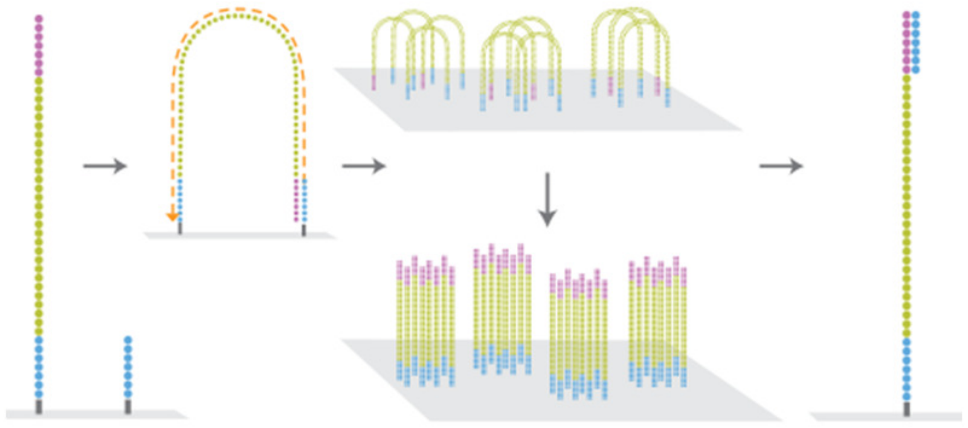


Figure 2.4: **Illumina Cluster Generation**

strands are cleaved and washed away and the sequencing primer is hybridized to the DNA-templates.

2.2.3 Sequencing

The sequencing reaction is conducted simultaneously on a very large number of different template molecules spread out on a solid surface. The sequencing cycle begins by adding four fluorescently-labeled reversible terminators, primers, and DNA polymerase. Because of the terminators, only a single base can be added by a polymerase enzyme to each growing DNA copy strand. After laser excitation, the emitted fluorescence from each cluster is detected by a camera (Fig. 2.5). The fluorescence label and the blocking group are then removed and the sequencing cycles are repeated to determine the sequence of bases one at a time.

2.2.4 Paired-end sequencing

The procedure that has just been described is referred as single-read sequencing and implicates sequencing DNA from only one end. This is the simplest way to utilize Illumina sequencing but also the less powerful. A slight modification to the standard single-read library preparation allows reading both the forward and reverse template strands of each cluster. This alternative sequencing procedure, or paired-end, can be really helpful and all Illumina NGS systems are capable of it.

Paired-end sequencing always improves the quality of the entire data set and several bioinformatic tools make use of this additional knowledge to fulfill different aims. The information about both ends of a sequence and the expected distance that exist between them can highly increase the specificity of the alignment compared to single end sequencing. For example, DNA repeats

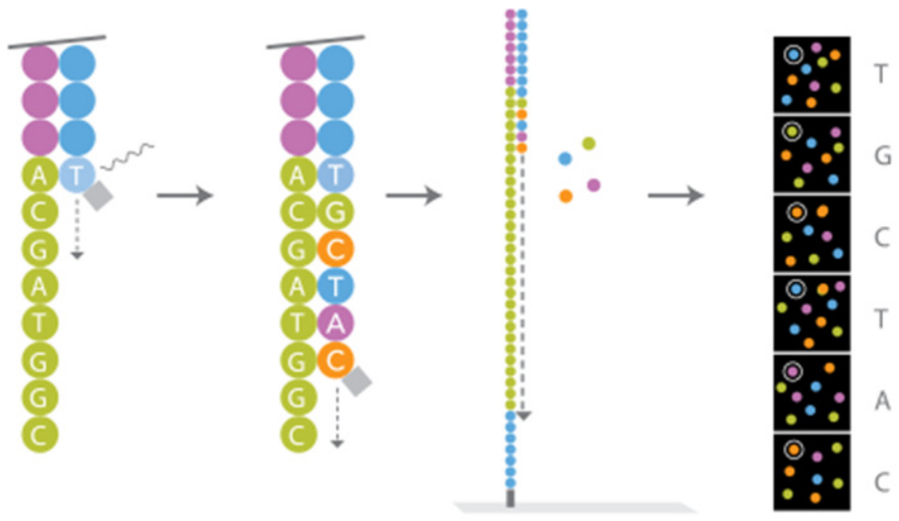


Figure 2.5: **Illumina Sequencing**

in the genome could be really challenging to align against and the additional information of pairing could resolve eventual un-univocal alignments. Additionally, paired-end distances that are different than expected could reflect genomic rearrangements or other structural variants.

2.3 Targeted resequencing

In order to dissect relationships between genotype and disease phenotype and to integrate somatic mutations into a clinical/molecular prognostic model, a focused screen of 38 cancer genes in cohort of 214 CMML patients was performed. Usually, gene discovery studies routinely screen matched tumor and constitutional DNA while large-scale gene resequencing is applied to tumor samples only. Despite this can be considered a large-scale resequencing study, for 74 patients the constitutional DNA was sequenced as well. Later on, the reason of this choice will be deeply discussed. In order to capture only 38 genes, a custom panel was designed through TruSeq Custom Amplicon, an amplicon sequencing solution for interrogating custom regions of interest. An online software (DesignStudio) allows researchers to design probes by entering target regions of the genome. Probes are then automatically defined by an algorithm that considers a range of factors like GC content or specificity and the typical design success is 90% or better. Once a custom design has been ordered, oligonucleotide probes are synthesized and pooled into a Custom Amplicon Tube (CAT).

The assay chemistry begins with hybridizing two custom-designed probes upstream and downstream of the region of interest (Fig. 2.6). Then an extension-ligation reaction extends across the region of interest and yield a library of new

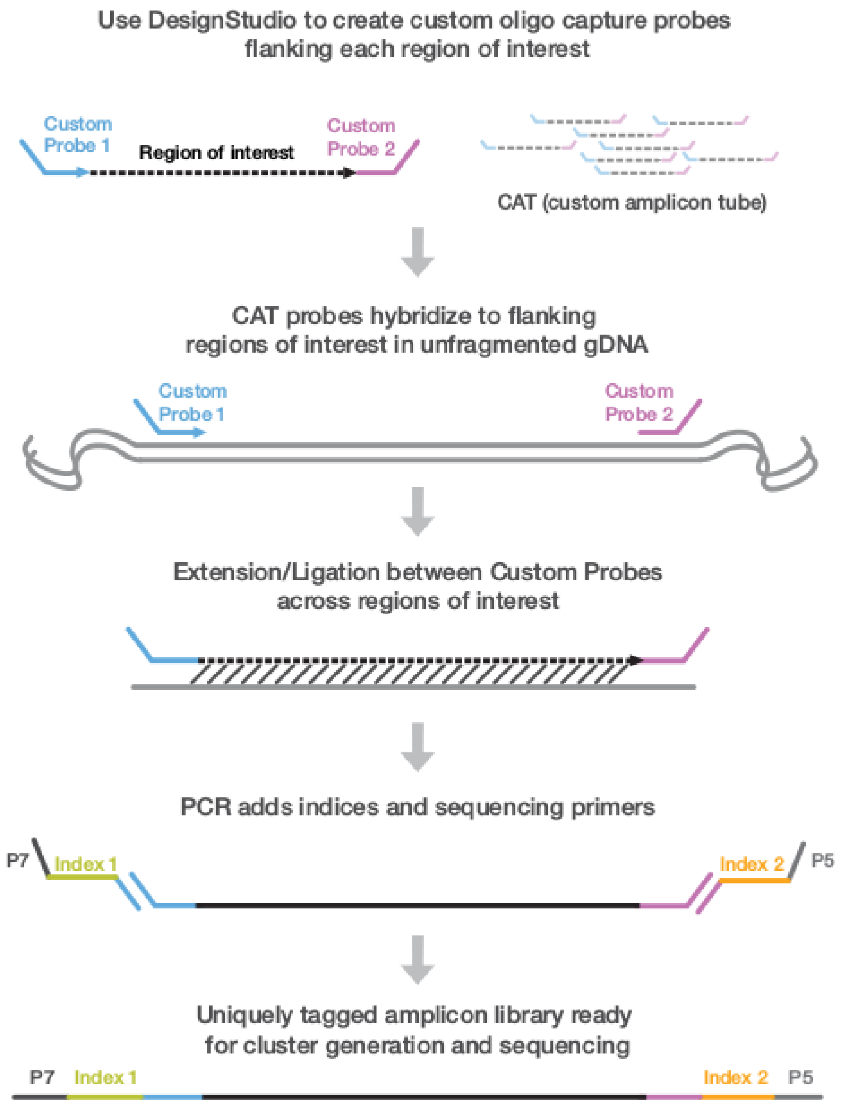

Use DesignStudio to create custom oligo capture probes flanking each region of interest
Custom Probe 1
Region of interest
Custom Probe 2
CAT (custom amplicon tube)
CAT probes hybridize to flanking regions of interest in unfragmented gDNA
Custom Probe 1
Custom Probe 2
Extension/Ligation between Custom Probes across regions of interest
PCR adds indices and sequencing primers
P7
Index 1
Index 2
P5
Uniquely tagged amplicon library ready for cluster generation and sequencing
P7
Index 1
Index 2
P5

Figure 2.6: **TruSeq Custom Amplicon Workflow Overview**

template molecules with common ends. Finally, sample-specific indices are added to each library by PCR and the pooled libraries are ready to be loaded into the MiSeq system for automated cluster generation and sequencing.

The customized TruSeq Amplicon panel that was defined for this study resulted in 886 pairs of probes designed to bind flanking genomic areas of interest. In particular, it allowed to capture all the exonic intervals for the 38 genes in Table 2.2. Library preparation and sequencing using MiSeq was finally performed according to the manufacturer's instruction for all 214 tumor samples and for 74 matched constitutional DNA.

Biological pathways and genes	**Frequency %***
RNA SPLICING	
SF3B1	15-30%
SRSF2	10-20%
ZRSF2	<10%
U2AF1	<10%
DNA METHYLATION	
TET2	20-30%
DNMT3A	$\approx$ 10%
IDH1/IDH2	$\approx$ 5%
CHROMATIN MODIFICATION	
ASXL1	15-20%
EZH2	$\approx$ 5%
TRANSCRIPTION	
RUNX1	$\approx$ 10%
BCOR	<5%
DNA REPAIR CONTROL	
TP53	$\approx$ 5%
COHESIN	
STAG2	<10%
RAS PATHWAY	
CBL	<5%
NRAS/KRAS	<5%
NF1	<5%
DNA REPLICATION	
SETBP1	<5%
RECEPTORS	
CSF3R	<1%

Table 2.1: **Somatic mutations of potential clinical relevance found in CMML patients** *Approximate proportion of patients with MDS carrying the mutant gene reported in studies published so far

List of genes detected by NGS
ASXL1, BCOR, CBL, CEBPA, CSF3R, CUX1, DNMT3A, EP300, ETNK1, ETV6, EZH2, FLT3, IDH1, IDH2, JAK2, KIT, KRAS, KMT2A, KMT2D, KMT2C, KMT2E, NF1, NPM1, NRAS, PHF6, PTPN11, RIT1, RUNX1, SETBP1, SF3B1, SRSF2, STAG2, TET2, TP53, U2AF1, KDM6A, WT1, ZRSR2

Table 2.2: **Targeted genes**

Chapter 3

Variant analysis workflow

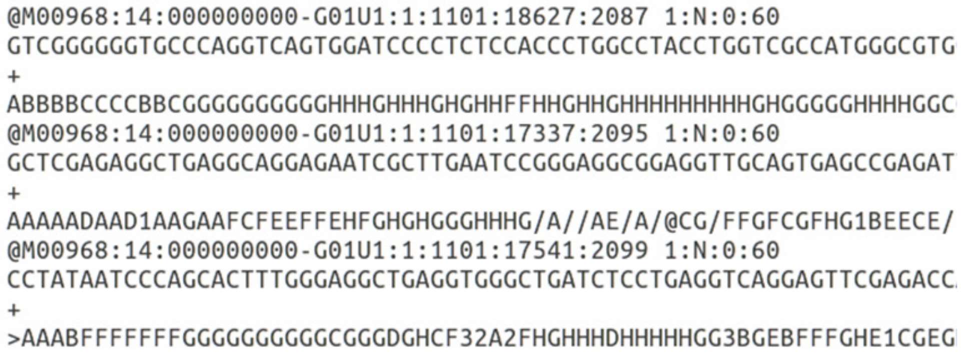
3.1 Data Pre-processing

Data pre-processing usually refers to a sequence of processes that converts raw sequencing data (FASTQ files) to an analysis-ready format (BAM files). A FASTQ file is indeed just a set of sequences that has to be processed in order to perform a downstream analysis that aspires to extract valuable results. Sequences have initially to be suitably merged to reconstruct the original genomic sequence from which they originated. Removing sequencing artifacts will subsequently make data more reliable and ready for the next part of the analysis. This section will therefore review data formats and analysis tools that characterize such a workflow.

3.1.1 FASTQ Format

The result of the sequencing process is a collection of strings representing the decoded bases of the genomic fragments generated during the library preparation step. The FASTQ file format has been established as a de facto standard for sharing sequencing data and combines both the sequences and the base quality scores [25]. Because of this ability to store a numeric quality score associated with each nucleotide in a sequence it can be considered as a simple extension of the FASTA format [26]. Each record of a FASTQ files is defined by four line; an example is shown in Fig. 3.1.

The first one begins with a '@' character and is followed by a record identifier. There is no length limit and an arbitrary comment can be included. The second line, as for FASTA format holds raw sequence letters. Again, there is no length limit but only IUPAC single letter codes are accepted and upper case is conventional. Third line begins with a '+' and marks the end of the sequence. Originally, the '+' character was followed by the same sequence identifier on the first line but not anymore. Finally, line 4 encodes the quality values for the sequence and it allows a subset of the ASCII printable characters. This



```
@M00968:14:000000000-G01U1:1:1101:18627:2087 1:N:0:60
GTCGGGGGGTGCCCAGGTCAGTGGATCCCCTCTCCACCCTGGCCTACCTGGTCGCCATGGGCGTG
+
ABBBBCCCCBBCGGGGGGGGGGHHHGHHHGHGHHFFHHGHHGHHHHHHHHHGHGGGGGHHHHGGC
@M00968:14:000000000-G01U1:1:1101:17337:2095 1:N:0:60
GCTCGAGAGGCTGAGGCAGGAGAATCGCTTGAATCCGGGAGGCGGAGGTTGCAGTGAGCCGAGAT
+
AAAAADAAD1AAGAAFCFEEFFEHFGHGHGGGHHHG/A//AE/A/@CG/FFGFCGFHG1BEECE/
@M00968:14:000000000-G01U1:1:1101:17541:2099 1:N:0:60
CCTATAATCCCAGCACTTTGGGAGGCTGAGGTGGGCTGATCTCCTGAGGTCAGGAGTTCGAGACC
+
>AAABFFFFFFFGGGGGGGGGGCGGGDGHCF32A2FHGHHHDHHHHHGG3BGEBFFFGHE1CGEG
```

Figure 3.1: **Example of a FASTQ file**

quality string is required to be equal in length to the sequence string; the scoring system used to assess quality is the so-called Phred [27] score. The quality of a base is defined in terms of the estimated probability of error:

$$Q_{PHRED} = -10 \log_{10} P_e \tag{3.1}$$

where P_e is the estimated probability of a base being wrong. Quality scores are then encoded as ASCII characters by adding 64 to the Phred values. Error rates typical range from a few tenths of a per cent to several per cent; 1% error rate corresponds for example to a phred score of 20. It is worthy to note that such measures can highly affect the assembly or alignment, the variant calling procedure and the downstream analyses.

3.1.2 FASTQ Quality Controls

Bioinformatics takes the lead role starting from here. Before any further analysis on sequencing data it is necessary to perform a set of quality controls on FASTQ data and give important feedbacks upstream to the wet-lab. This task is sometimes underestimated but it should be considered as a part of the data processing pipeline because it gives a first idea about any data problems that can reflect experimental or sequencing inaccuracies. The overall sequencing process requests indeed several requirements to be satisfied and errors can be introduced both during library and cluster generation or sequencing.

As a bioinformatic tool useful for this task we chose FastQC because of its simplicity and efficacy [28]. Moreover it can either run as a standalone interactive application or as a non-interactive application which can be integrated into a larger pipeline. FastQC implements twelve modules of analysis and generates graphs and tables to quickly assess sequencing data. Hereunder we will not discuss all them but only few interesting ones.

A first basic control checks the number of reads generated. A low number of sequences can be explained in terms of a little amount of DNA loaded during the library preparation or problems with imaging process or adaptors. In particular, adaptors can often be inaccurate to capture specific targeted regions.

Per base sequence qualities can then be checked. This check gives an overview about the range of quality values across all bases at each position in the FASTQ file. Usually a general degradation of quality over the length of reads is expected but, if it is too severe, it can be necessary to perform quality trimming based on average quality.

Further, it is fair to expect no difference between the proportion of different bases on a sequence run. Per base sequence content should not show an over-representation of any particular base. Strong imbalances at particular cycles can be indicative of the presence of adaptors, while skewness across all cycles may be indicative that a targeted region was not sequenced.

Finally it is important to check the level of duplicate sequences. An high level of duplication may indicate some kind of enrichment bias and should be avoided.

3.1.3 Alignment

The alignment or assembly of raw sequencing data is the most important step of data pre-processing. While quality controls ensure no technical errors during the sequencing process, the mapping or assembly procedure provides a first feedback about the success of an experiment.

Briefly, assembly algorithms need to compare every read with every other in order to align and merge them in a full-length sequence. This dissertation will not further discuss about assembly because this process takes place when the sequencing samples are derived from an organisms without a reference genome as small viruses or bacteria. On the contrary, over the last decade several version of a reference assembly have been generated for human and the latest one (GRCh38) was released on December 2013. When a reference genome is available, the alignment process expects to map raw sequences against it.

Mapping consists in determining the most likely source within the genome sequence for an observed sequence. The choice of the alignment algorithm has therefore a crucial role and strongly influences the variant detection process. A good aligner has then to endure incorrect alignments that are mainly due to sequencing errors or to real differences between the reference genome and the sequenced one as point mutations or indels. Furthermore, because of the size of the reference genome (approximately 3.2 GB) and the massive number of sequence to align, it should be fast enough and should guarantee an high accuracy.

The majority of the available methods relies on hash-based techniques or on the Burrows-Wheeler transform (BWT) [29]. Both these methods build an index for the human genome but the latter require half of the disk space to store it. Moreover, at the same sensitivity level, BWT-based ones are usually tenfold faster if compared to their hash-based counterparts and require less RAM memory. As a results of these considerations, the Burrows-Wheeler Aligner (BWA) was selected for this task [30, 31]. This a widely adopted aligner that was specifically designed for short read mapping and works for different sequencing platform. Nowadays BWA consists of three alignement algorithms: all of them support gapped alignment but they are tuned slightly differently in order to manage different type of sequencing data. The one implemented in the workflow is BWA-SW.

The algorithm uses an FM index data structure [32] created from the BWT of a genomic reference (the version used in the workflow was the GRCh37 version of the human genome). This data structure require firstly to modify the sequence order of the reference genome using the BWT (Fig. 3.2); in general, the BWT of a string S is a reversible permutation of it that enables the search for a pattern P in S to take linear time with respect to the length of P. Then the final index is created and is used for rapidly positioning a read on the genome.

Mapping qualities are then assigned by BWA to the aligned reads. These qualities are necessary for the reliability of the variant call procedures and are conditioned by several factors. The repeat structure of the reference genome is usually taken into account and reads that fall in repetitive regions of the reference are assigned low scores. Mapping qualities are then affected by read base qualities: low base qualities mean the observed sequence is wrong and can therefore lead to a wrong alignment. Sometimes, despite the algorithm performs gapped alignment, reads are not uniquely mapped and they are placed randomly with a mapping quality 0. These reported ambiguities, can be reduced by paired-end sequencing which additionally influence mapping qualities. If run with paired-end data, BWA searches indeed for suboptimal hits of pairs and reaches an higher alignment accuracy.

3.1.4 SAM/BAM Format

The result of the alignment step is a Sequence Alignment/Map (SAM) file [33]. The SAM format was generated to overcome the fact that alignment tools used to generate different output formats, complicating the downstream analysis. Nowadays it is the standard for storing read alignments against reference sequences and was defined in 2009 as a result of a collaborative effort involving scientists from many of the world's leading research institutions as the Broad Institute, the Wellcome Trust Institute and the Beijing Institute of Genomics. Alignment data from the 1000 Genomes Project have been released in this format making it widely used since then. It mainly consist of one header section and one alignment section that are distinguishable through the first line character.

Header section starts with '@' while alignment section do not. Each header line has then a two-letter record type code that is tab-spaced by a TAG:VALUE pair. All the TAGs can be grouped under 5 record types as defined in Table 3.1; this table then shows which tags are mandatory for each type. A detailed description can be found at http://samtools.github.io/hts-specs/SAMv1.pdf.

The alignment section contains the information about the alignment of all sequences against a reference genome. Each alignment line has 11 mandatory

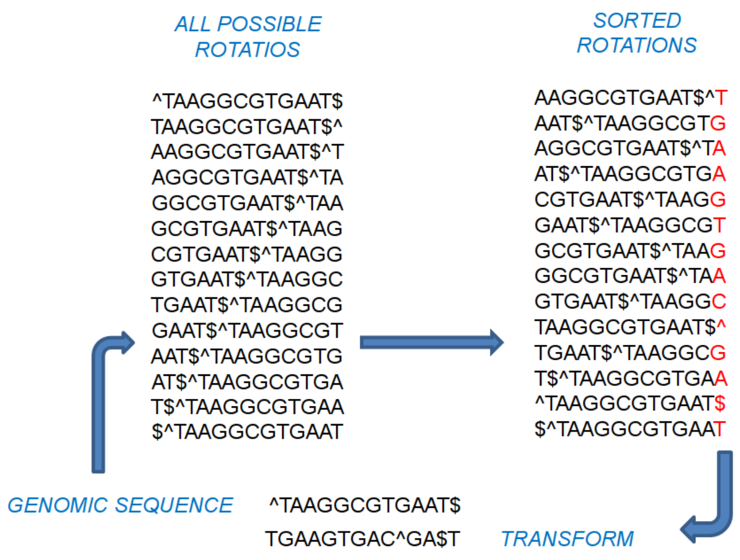


Figure 3.2: **Burrows-Wheeler transform for a genomic sequence** The characters ^ and $ mark the beginning and end of the sequence. To create a BWT of a 14-mer genomic sequence, all rotations of the given sequence needs to be constructed by taking the first character of the sequence and placing it at the end of the sequence. Then these sequences are sorted and the BWT is defined by the final column of the matrix

fields and a variable number of optional fields. All the mandatory fields must be present, but their values can be '0' or '*' if the corresponding information are unavailable. Table 3.2 gives an overview of these mandatory fields.

An alternative way to represent a SAM file is its binary representation. The Binary Alignment/Map (BAM) holds the same information but it is compressed by the BGZF library. This format relies on block compression and allows fast random access for indexed queries. However, in order to random access a position-sorted BAM file, the corresponding index file should be created; this process combines the UCSC binning scheme [34] and a simple linear indexing.

3.1.5 Duplicate reads removal

Looking at the basic steps of library preparation and sequencing, it is worth noting that between the adapter ligation step and the spread of DNA molecules across the flowcell exists a polymerase chain reaction (PCR) amplification that amplifies the fragments with ligated adapters. This step intentionally creates multiple copies of each genomic molecule in order to have enough of them. However once the library is flowed across the flow cell is impossible to get exactly one copy of the same molecule to get bridge amplified. Fragments are indeed randomly immobilized to folwcell's surface.

Tag		**Description**
@HD		The header line. The first line if present.
	*VN**	Format version
@SQ		Reference sequence dictionary.
	*SN**	Reference sequence name.
	*LN**	Reference sequence length.
@RG		Read group.
	*ID**	Read group identifier.
@PG		Program.
	*ID**	Program record identifier.
@CO		One-line text comment.

Table 3.1: **Header section in the SAM format** For each type only the required tags (defined by *) are reported

Col	**Field**	**Type**	**Brief description**
1	QNAME	String	Query template NAME
2	FLAG	Int	bitwise FLAG
3	RNAME	String	Reference 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 3.2: **Mandatory fields in the SAM format**

Duplicate reads occur when two copies of the same original molecule get on different primer lawns in a flowcell. This will result in a bias introduced during the sequencing procedure because two or more copies of the same fragment will not be treated as one. On the contrary, certain sequences will be represented in artificially high numbers. As a consequence, sequencing errors can be propagated by duplicates and can be confused with point variations. Removing duplicate reads is therefore a widely used practice to correct this bias when analyzing NGS data, otherwise it can badly affect results. The percentage of identical copies is usually around 5%. Higher rates of duplication arise when there is too little starting material and a greater amplification of the library is needed, or when there is a great a variance in fragment size. The polymerase chain reaction introduces bias in reproducing reads of different lengths and compositions [35], and smaller fragments, which are easier to amplify, can end up over-represented.

Duplicate reads removal step is implemented in the workflow of analysis by the MarkDuplicates tool. It is part of Picard (http://picard.sourceforge.net), a set of Java command line tools for manipulating high-throughput sequencing data and it can both mark the duplicates or remove them. The TruSeq Custom Amplicon kit generates indeed only duplicate reads because it aims to capture specific genomic regions. If sequencing data are generated by an experiment based on this kit, this step of analysis should be disabled because otherwise it will remove the majority of aligned reads.

3.1.6 Indel-based Realignment

Several realignment methods have been proposed to deal both with alignment errors and reference bias. This section will focus on the local realignment tool designed by the Genome Analysis Toolkit (GATK) [36] to minimize the number of mismatching bases across all the reads. It's the one implemented in the pipeline because reaches good realignment results and it's considered the state of art although the overall procedure can be very time consuming if compared with the initial alignment.

The majority of regions that require local realignment are usually near an insertion in the individual's genome with respect to the reference genome. Indels in reads can cause problems to the aligner and usually results in misalignment or mismatches. These mismatches can be considered artifacts and can erroneously be mistaken as SNP by most of the variant callers increasing the false positive rate. Since the alignment step is performed on each read independently, it is not possible to minimize these mismatches across all the reads unless by local realignment.

The step of the workflow here discussed aims therefore at realigning the results around indels to improve the accuracy of the downstream analysis. There are mainly three types of realignment targets: indels known to be frequent on the population, indels that are observed in the aligned reads and hidden indels that are suggested by some evidence. The realignment process is done by two step. Firstly, the RealignerTargetCreator tool determines small suspicious intervals which are likely in need of realignment and subsequently the IndelRealigner tool runs the realigner over those intervals. The RealignerTargetCreator takes the BAM file as input and possibly one or more lists of known indels. In particular, the following recommended sets of known sites have been used for the analysis:

- Mills_and_1000G_gold_standard.indels.b37.sites.vcf
- 1000G_phase1.indels.b37.vcf (currently from the 1000 Genomes Phase I indel calls)

The same files should be used for the IndelRealigner step.

In summary, the first step searches and identifies the most parsimonious alignment along all reads at a problematic site: if it is sufficiently better than the original it's accepted and the realignment process is performed by the second step. Several regions are finally transformed into clean reads containing a consensus indel suitable for standard variant discovery approaches.

3.1.7 Base Quality Score Recalibration

The last step of data preprocessing's pipeline modifies the base quality scores of each read in order to make them closer to the actual probability of mismatching the reference genome. Once again, a GATK's tool is widely considered the best solution to achieve this goal known as Base Quality Score Recalibration (BQSR). The BQSR's main function is therefore to reduce noise of quality scores caused by machine read errors by looking at more than just a single base at time, rather, every base in the BAM file.

Basically all reference mismatches are considered errors and indicative of poor base quality unless present in dbSNP [37]. dbSNP contains millions of known molecular variations and the used version (dbSNP_137) has more than 50 million variants. The fact that any mismatch not in dbSNP is an error is a statistically assumption that aims to reduce the number of false positives at the expense of an increased false negative rate. Because of the tool's ability to recalibrate also base insertion and base deletion, the following files have been used in addition to dbSNP in order to filter reference mismatches considered as errors:

- Mills_and_1000G_gold_standard.indels.b37.sites.vcf
- 1000G_phase1.indels.b37.vcf (currently from the 1000 Genomes Phase I indel calls)
- 1000G_omni2.5.b37.vcf

First pass of this process is performed by the tool BaseRecalibrator that analyzes covariation among several features as:

- reported quality score
- position within the read (machine cycle)
- preceding and current nucleotide (sequencing chemistry effect)

Given the particular covariates seen at each site it is possible to calculate the PHRED-scaled quality score as:

$$\frac{\#\text{ of reference mismatches} + 1}{\#\text{ of observed bases} + 2} \tag{3.2}$$

The output file is a table that reports several covariate values, the number of observations and mismatches and the new empirical quality score. A second run then applies these new quality scores on all reads in a BAM file.

3.2 Variant discovery

Next-generation sequencing mainly aims at discovering or confirming variations among populations of related samples. This task is commonly referred to as variant discovery and takes place after data-preprocessing. The following section will introduce the tools that have been implemented in the pipeline to perform this task. The topic will be preceded by several considerations in order to help its comprehension and to explain why these tools have been included in the analysis workflow.

Given accurately mapped and calibrated reads, variant discovery searches for differences between the sequenced genome and a reference. Human genetic variation ranges from the single base pair to large chromosomal events. Single Nucleotide Variants (SNVs) are variations of a single nucleotide with respect to the reference genome. SNVs are referred as SNPs (single nucleotide polymorphisms) if occur commonly (e.g. 1%) within or between populations. While not as common as SNPs, which are the main source of genetic human variation, indels are also widely spread across the genome. The term indel defines small (usually < 1 kb) INsertions or DELetions of bases in the DNA of an organism and the genetic variations caused by them is substantial.

It's probably not too surprising but next-generation sequencing has revealed that human genomes differ more as a consequence of structural variation than of single nucleotide differences. Structural variations were originally defined as insertions, deletions, inversions and translocations greater than 1 kb in size but due to the lack of a standard, sometimes also small indels go under this definition. A particular subtype of structural variation are then the copy-number variations (CNVs). These are variation in the number of copies of one or more sections of the DNA and includes insertions, deletions and duplications.

Usually the variant analysis consists in determining the presence of these variants in comparison to the human genome reference however, when this analysis refers to a somatic sample, variants can be called in comparison with a matched normal genome. Despite the latter approach has recently become a standard in cancer research, normal matched samples are not always present when a study is defined. Following these considerations different variant caller have been implemented in the pipeline and will be discussed in the next subsections. In particular, four variant caller will be reviewed: UnifiedGenotyper, a GATK's variant caller for point and indel germline variants; MuTect, a somatic point mutations variant caller; ExomeCNV, a method to detect copy-number variation and loss of heterozygosity from tumor-matched data; and finally SVDetect that identify structural variations from paired-end data.

Until recently tumor and matched samples were genotyped independently against the reference genome and the results were subtracted. Today this assumption of independence has been replaced by refined comparative analysis of tumor and normal samples that achieve higher performances with respect to the previous approach. [38, 39].

Even with normal sequence data, somatic calling can be very challenging because of tumor heterogeneity and normal contamination that add additional noise to the various sources of errors that have been introduced in the section about data-preprocessing.

3.2.1 Unified Genotyper

UnifiedGenotyper is a GATK's module that addresses SNP and genotype calling procedures. SNP calling aims at determine in which positions the aligned bases differ significantly from a reference sequence. Genotype calling then determines the genotype for only that positions in which a SNP or a variant has already been called.

Early methods for genotype and SNP calling rely only on counting the number of times each allele is observed at each site and using simple cutoff rules to perform SNP or genotype calls [40]. Cutoffs are empirically defined and usually lead to underestimate heterozygous genotypes. UnifiedGenotyper incorporates instead uncertainty by using a probabilistic framework that adds additional information regarding allele frequencies. As a result is it possible to provide measures of uncertainty about the genotype inference and scoring the results accordingly. This approach is very fast and relies on the assumption that bases are independent. SNP and indels are therefore called separately and each variant locus is considered independently.

Considering that D represents all the read for a particular individual at a particular site, is it possible to calculate the likelihood of the data $Pr(D|G)$ for a given genotype G. For each genotype, $Pr(D|G)$ is:

$$Pr(D|G) = \prod_j \left(\frac{Pr(D_j|H_1)}{2} + \frac{\mathrm{Pr}(D_j|H_2)}{2} \right) \tag{3.3}$$

where $G = H_1H_2$ because of the diploid assumption. It follows then:

$$Pr(D_j|b) = \begin{cases} 1 - \epsilon_j & \text{if } D_j = b \\ \epsilon_j & \text{otherwise} \end{cases} \tag{3.4}$$

where b is a specific allele and ϵ_j is the larger error rate between sequencing and alignment errors for the j-th base. If a prior probability is further supplied, the Bayes' formula can be used to calculate the likelihood of a genotype G:

$$Pr(G|D) = \frac{Pr(G)Pr(D|G)}{\sum_i Pr(G_i)Pr(D|G_i)} \tag{3.5}$$

Priors are applied when this calculation is performed simultaneously on different sample but in this case the algorithm is usually used in single sample mode and P(G) are set to 1.

SNP and genotype calling procedure can be summarized as follows:

- initially the algorithm computes the genotype likelihood for all the diploid genotypes (AA, AC, AG, AT, CC, CG, CT, GG, GT, TT) using only the pileup of bases and associated quality scores. These likelihood are however computed only on those bases satisfying minimum base quality, mapping read quality and pair mapping quality.

- the allele frequency distribution is therefore computed to determine the most likely allele count and a variant call is emitted if determined;
- if a variant will be emitted, the algorithm assigns a genotype: the genotype with the highest probability is generally chosen and a genotype quality is defined. Once all the likelihood have been calculated the likelihood of the best genotype is normalized to 1 and all the other likelihoods according to that scale. Then these values are phred-scaled and the genotype quality is the calculated as the likelihood of the most likely genotype minus the likelihood of the second most likely genotype.

Indel calling procedure relies on a different definition of genotype likelihood but the overall process is similar.

Recently, a new GATK's variant caller has been released. This tool, called HaplotypeCaller, calls simultaneously SNPs, indels, and some SVs by performing a local de-novo assembly. Both HaplotypeCaller and UnifiedGenotyper perform very well, but overall HaplotypeCaller provides more accurate calls, in particular for regions that are traditionally difficult to call. However it is currently computationally intensive, so UnifiedGenotyper is still very valuable and it was chosen as the one to implement in the pipeline.

3.2.2 MuTect

The diploid assumption made by UnifiedGenotyper makes this tool specific to detect germline variations in an individual's genome. Germline mutations may occur de novo or be inherited from parents' germ cells. On the contrary, somatic mutations accumulate in cells of the body during a person's lifespan and can lead to cancer. It is worth to remember that the implemented pipeline focus on tumors so a variant caller that doesn't treat somatic mutation is not enough. Somatic mutations however are more difficult to identify. Firstly, these mutations occur at lower frequency than germline mutations, usually from 0.1 to 100 per megabase; Secondly, they are only present on a small fraction of DNA molecules that are sequenced; tumor samples are nearly always contaminated by normal cells and additionally a somatic mutation can originate by a subpopulation of tumor cells.

MuTect [41] was therefore implemented in the pipeline. This additional variant caller reaches high sensitivity and specificity to detect somatic point mutations, given a tumor sample and its patient-matched equivalent. Preprocessed input sequences are scanned on each genomic locus independently by performing the next four steps:

1. removing low-quality sequence data;

2. performing variant detection through a Bayesian classifier;
3. filtering false positives;
4. designing the variants as somatic or germ-line through a second Bayesian classifier.

Each aligned read passes the first filter if the mapping quality score is greater than zero, if base quality scores are greater than or equal to five, if the sum of the quality scores of the mismatches is less than or equal to one hundred and if less than 30% of bases have been soft-clipped. Finally if there is an overlapping read pair, and both reads agree, the read with the highest quality score is retained, otherwise both are discarded.

In order to understand the variant detection process, we will review the main steps of the algorithm. For each site the reference allele is denoted as $r \in \{A, C, G, T\}$ while b_i and e_i represent the called base of read $i(i = 1...d)$ that covers the site and the probability of error of that base call (each base has an associated Phred-like quality score q_i where $e_i = 10^{-\frac{q_i}{10}}$). The algorithms exploit two models to call a variamt: (i) model M_0 in which there is no variant at the site and all nonreference bases are explained by sequencing noise, and (ii) model M_f^m in which a variant allele m truly exists at the site with an allele fraction f and, as in M_0, reads are also subject to sequencing noise. M_0 is equivalent to M_f^m with $f = 0$. The likelihood of the model M_f^m is given by:

$$L(M_f^m) = P(\{b_i\}|\{e_i\}, r, m, f) = \prod_{i=1}^{d} P(b_i|e_i, r, m, f) \tag{3.6}$$

assuming the sequencing errors are independent across reads. If all substitution errors are equally likely, that is, occur with probability $e_i/3$, it follows:

$$P(b_i|e_i, r, m, f) = \begin{cases} f^{e_i/3} + (1-f)(1-e_i) & \text{if } b_i{=}r \\ f(1-e_i) + (1-f)^{e_i/3} & \text{if } b_i{=}r \\ e_i/3 & \text{otherwise} \end{cases} \tag{3.7}$$

Variant detection is performed by comparing the likelihood of both models and if their ratio, that is, the LOD score, exceeds a decision threshold ($log_{10}\delta_T$) we declare m as a candidate variant at the site. LOD_T is therefore calculated as:

$$LOD_T(m, f) = log_{10}\left(\frac{L(M_f^m)P(m, f)}{L(M_0)(1 - P(m, f))}\right) \geq log_{10}\delta_T \tag{3.8}$$

and δ_T is set to 2 to ensure being at least twice as confident that the site is variant as compared to noise. LOD_T cab be rewrited as:

$$LOD_T(m, f) = log_{10}\left(\frac{L(M_f^m)}{L(M_0)}\right) \geq log_{10}\delta - log_{10}\left(\frac{P(m, f)}{(1 - P(m, f))}\right) = \theta_T \tag{3.9}$$

To determine $P(m, f)$, is assumed that $P(m)$ and $P(f)$ are statistically independent and that $P(f)$ is uniformly distributed (that is, $P(f) = 1$) and $P(m)$ is one-third of the expected mutation frequency for the studied tumor type (representing equal prior for all substitutions). A typical mutation frequency of $3 \cdot 10^{-6}$, yields $\theta_T = 6.3$.

Despite very useful as a threshold for detection, LOD score cannot be immediately translated into the probability that a variant is a real mutation rather than a sequencing error. Because the algorithm is based on the incorrect assumption of independent sequencing errors and accurate read placement, it usually underestimates the false positive rate. To eliminate these false positives MuTect applies six filters and possibly a panel of normal samples as controls to clear both germ-line events and artifacts.

Each variant detected is then classified as somatic or germline. To perform this task a second classifier similar to the one described is used. In this case, f is conservatively set to 0.5 for a germ-line heterozygous variant. Thus it follows:

$$LOD_N = log_{10}\left(\frac{L(M_0)P(m,f)}{L(M_{0.5}^m)P(germline)}\right) \geq log_{10}\delta_N \tag{3.10}$$

which can be rewritten as:

$$LOD_N = log_{10}\left(\frac{L(M_0)P(m,f)}{L(M_{0.5}^m)P(germline)}\right) \geq log_{10}\delta_N - log_{10}\left(\frac{P(m,f)}{P(germline)}\right) = \theta_N \tag{3.11}$$

This second classifier makes therefore sure the normal does not carry the variant allele. Once the LOD score for the normal sample is calculated it is compared with a cutoff determined by the log ratio of prior probabilities of the considered events. In particular, to calculate $P(germline)$ two cases were distinguished: (i) sites which are known to be variant in the population by dbSNP and (ii) all other sites. The default value are set to $\theta_{N|non-dbSNPsite} = 2.2$ and $\theta_{N|dbSNPsite} = 5.5$.

3.2.3 Variant Call Format

The output of the reviewed variant callers is a VCF file. VCF is the acronym of Variant Call Format and specifies a format used in bioinformatics for storing genome variations such as SNPs, insertions, deletions and structural variants, together with rich annotations [42]. It was originally developed for the 1000 Genomes Project and nowadays it is a standard that has reached the 4.2 version. It was designed to be scalable and because of this feature it can store millions of sites with genotype data and annotations from thousands of samples. Every VCF file has three parts that are ordered as follow:

- Meta-information lines.
- One header line.

- Data lines.

First VCF line specifies the version number and as all the other meta information lines begins with "##". Additional meta information lines provide a standardized description of tags and annotations used in the data section. In particular, these lines can be very useful to decode filters that have been applied to the VCF or additional fields that are not usual. Moreover these lines can store information about the software used to process the file and any other information relevant to its history. Data lines are also called VCF records and contain variants with the corresponding genotype data and annotations. These lines report all the information defined in the header line; the latter has eight mandatory columns that are defined in table 3.3. In addition, if

VCF mandatory fields	
CHROM	chromosome
POS	1-based position of the start of the variant
ID	unique identifiers of the variant
REF	the reference allele
ALT	a comma separated list of alternate non-reference alleles
QUAL	a phred-scaled quality score
FILTER	site filtering information
INFO	and a semicolon separated list of additional, user extensible annotation

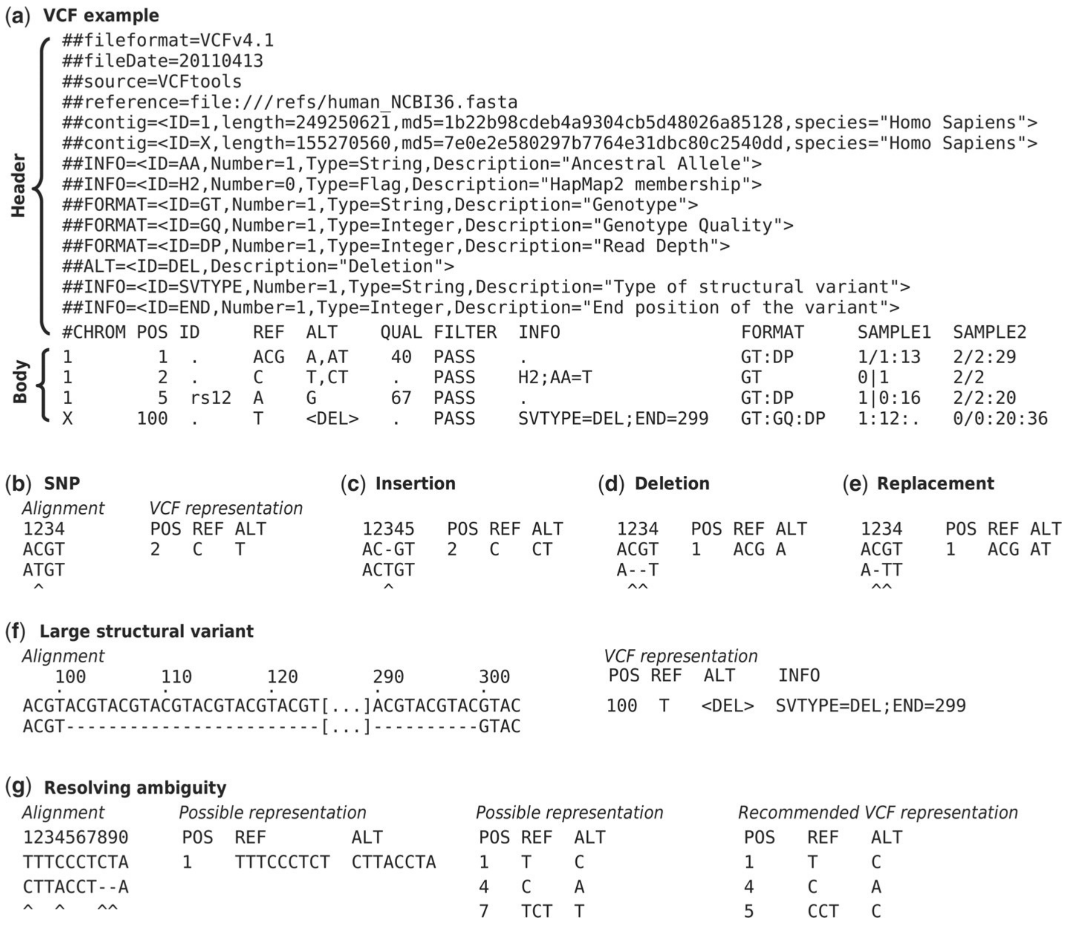
Table 3.3: **VCF mandatory fields**

samples are present in the file, the mandatory header columns are followed by a FORMAT column and an arbitrary number of sample IDs that define the samples included in the VCF file.

An example of VCF file is reported in Fig. 3.3 where the FORMAT field GT:GQ:DP indicates the genotype, genotype quality and read depth for each sample, respectively. All data lines are TAB delimited and it is strongly recommended that all annotation tags used are declared in the VCF header section. Moreover, Fig. 3.3 reports how to represent different sequence variants; it is strongly suggested to refer at http://samtools.github.io/hts-specs/VCFv4.2.pdffor a detailed description of VCF 4.2 specification.

3.2.4 Structural Variant Calling

As already stated, despite SNPs have long been considered to be the most common class of genetic variations, it is now recognized that CNVs embrace a greater proportion of the genome. In particular, an estimated 1.2% of a single genome differs from the reference human genome because of CNVs while only 0.1% because of SNPs [43].


Header
Body

(a) VCF example

```
##fileformat=VCFv4.1
##fileDate=20110413
##source=VCFtools
##reference=file:///refs/human_NCBI36.fasta
##contig=<ID=1,length=249250621,md5=1b22b98cdeb4a9304cb5d48026a85128,species="Homo Sapiens">
##contig=<ID=X,length=155270560,md5=7e0e2e580297b7764e31dbc80c2540dd,species="Homo Sapiens">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##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">
##ALT=<ID=DEL,Description="Deletion">
##INFO=<ID=SVTYPE,Number=1,Type=String,Description="Type of structural variant">
##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant">
#CHROM POS ID     REF  ALT    QUAL FILTER  INFO                FORMAT     SAMPLE1   SAMPLE2
1        1 .      ACG  A,AT     40 PASS    .                   GT:DP      1/1:13    2/2:29
1        2 .      C    T,CT      .  PASS    H2;AA=T             GT         0|1       2/2
1        5 rs12   A    G        67 PASS    .                   GT:DP      1|0:16    2/2:20
X      100 .      T    <DEL>     .  PASS    SVTYPE=DEL;END=299  GT:GQ:DP   1:12:.    0/0:20:36
```

(b) SNP

```
Alignment    VCF representation
1234         POS REF ALT
ACGT         2   C   T
ATGT
 ^
```

(c) Insertion

```
12345   POS REF ALT
AC-GT   2   C   CT
ACTGT
  ^
```

(d) Deletion

```
1234   POS REF ALT
ACGT   1   ACG A
A--T
 ^^
```

(e) Replacement

```
1234   POS REF ALT
ACGT   1   ACG AT
A-TT
 ^^
```

(f) Large structural variant

```
Alignment                                                   VCF representation
   100       110       120        290       300             POS REF  ALT    INFO
    .         .         .          .         .
ACGTACGTACGTACGTACGTACGTACGT[...]ACGTACGTACGTAC             100  T   <DEL>  SVTYPE=DEL;END=299
ACGT------------------------[...]----------GTAC
```

(g) Resolving ambiguity

```
Alignment    Possible representation       Possible representation    Recommended VCF representation
1234567890   POS  REF        ALT           POS REF  ALT               POS   REF   ALT
TTTCCCTCTA   1    TTTCCCTCT  CTTACCTA      1   T    C                 1     T     C
CTTACCT--A                                 4   C    A                 4     C     A
^  ^   ^^                                  7   TCT  T                 5     CCT   C
```

Figure 3.3: **Example of valid VCF** (a) Example of valid VCF. The header lines ##fileformat and #CHROM are mandatory, the rest is optional but strongly recommended. Each line of the body describes variants present in the sampled population at one genomic position or region. All alternate alleles are listed in the ALT column and referenced from the genotype fields as 1-based indexes to this list; the reference haplotype is designated as 0. For multiploid data, the separator indicates whether the data are phased (|) or unphased (/). Thus, the two alleles C and G at the positions 2 and 5 in this figure occur on the same chromosome in SAMPLE1. The first data line shows an example of a deletion (present in SAMPLE1) and a replacement of two bases by another base (SAMPLE2); the second line shows a SNP and an insertion; the third a SNP; the fourth a large structural variant described by the annotation in the INFO column, the coordinate is that of the base before the variant. (b-f) Alignments and VCF representations of different sequence variants: SNP, insertion, deletion, replacement, and a large deletion. The REF columns shows the reference bases replaced by the haplotype in the ALT column. The coordinate refers to the first reference base. (g) Users are advised to use simplest representation possible and lowest coordinate in cases where the position is ambiguous. Reproduced from [42]

Traditional methods as fluorescence in situ hybridization (FISH), SNP arrays and array comparative hybridization (aCGH) have been commonly used for detection of CNVs. However, these methods are usually affected by low resolutions that make difficult to detect short CNVs, and can't be used for a discovery analysis as they mainly target known CNV. NGS technology has recently brought several breakthroughs in SVs' detection process. The most promising methodologies that have been proposed to analyze NGS data for structural variants, adopt "Paired-End Mapping" or "Depth of Coverage" strategies.

Methods based on depth of coverage (DOC) usually partition the genome into non-overlapping windows and the counts in these windows are taken as a measure of DOC. This segmentation makes sure each window has the same read depth within it and contrasts with the adjacent ones. Candidate SVs are then determined by selecting the genomic windows in which the observed DOC is substantially different from the expected: a window can therefore indicate a gain, loss or no CNV event. In particular, it is assumed that the probability of any given base in the genome being sequenced is equal to any other base so the number of reads mapped to a region is assumed to follow a Poisson distribution and is proportional to the number of copies. As a result, a duplicated sequence in a sample have a number of reads higher than expected when mapping to a reference sequence. Similarly, a sequence that has been deleted would results in a lower read depth because fewer reads mapp to the reference. Although effective, DOC based methods cannot really detect where a CNV have been inserted and rely on segmentation processes that sometimes aren't robust enough.

PEM methods can detect more structural variants than DOC methods, including inversions and translocations by exploiting the mapping information of paired reads. Paired-end sequencing is required because these algorithms rely on the distribution of the insert sizes that occur between paired reads. If the end of a fragment maps at a distance longer than expected it could be indicative of a deletion while if the distance is shorter than expected, an insertion could have happened. Moreover once the alignment is done, reads can be investigated with respect to a set of possible configurations that imply certain SVs. These "signatures" are illustrated and described in Fig. 3.4.

3.2.5 ExomeCNV

ExomeCNV is a statistical method to detect CNV and LOH from exome sequencing data that relies on depth-of-coverage and B-allele frequencies from mapped sequence reads. This tool mainly works with paired samples and was integrated in the pipeline because, even if it was specifically designed for exome-sequencing data, it reaches good performances with targeted data if target regions are whole genes. The majority of depth-of-coverage techniques works with a continuous region as the whole genome or a specific target. On the contrary, by only taking into account the exons, the search space can

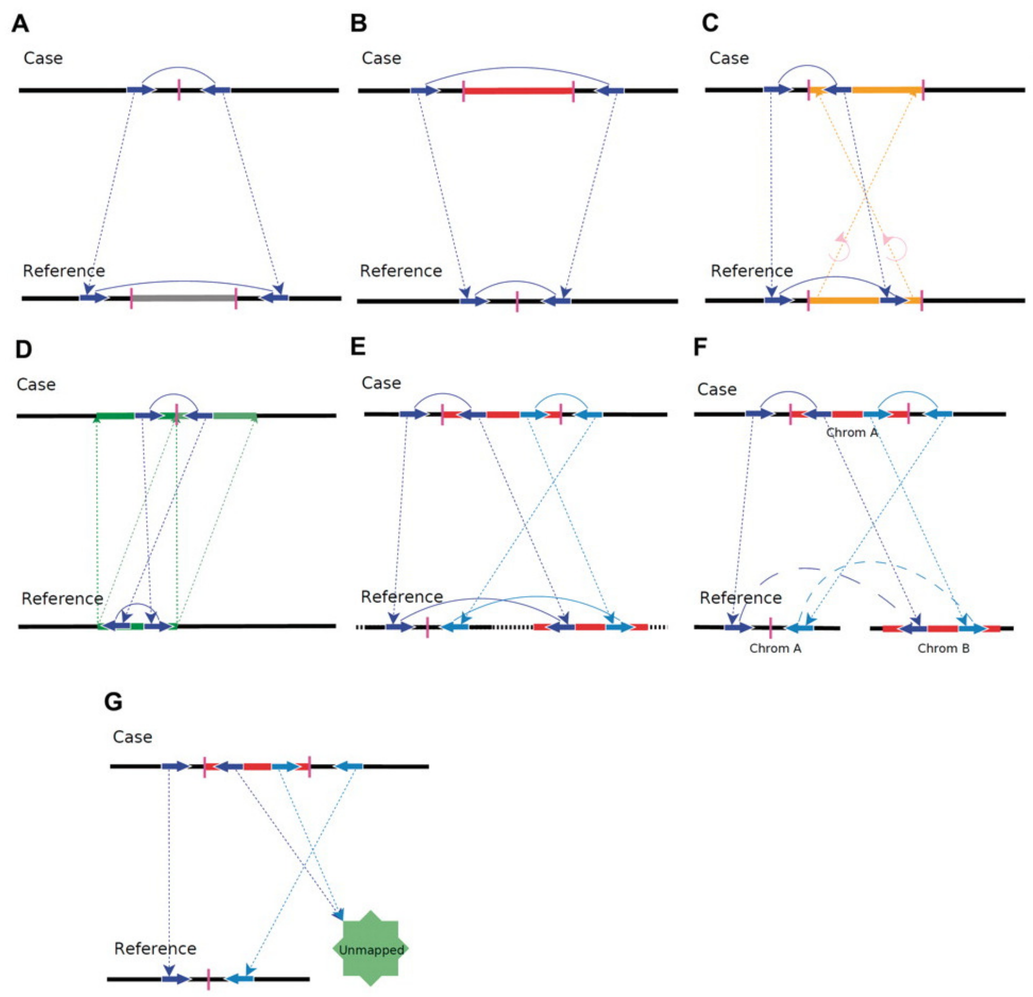


Figure 3.4: **Configurations of PEMs in various types of SVs** (A) Deletion. The paired-end read spans the breakpoint of a deletion. Thus, the mapped distance of the paired-end reads is significantly larger than the insert size. (B) Insertion. The paired-end reads spans an insertion, and the mapped distance significantly less than the insert size. (C) Inversion. The read pair encompasses one breakpoint of an inversion and the right end is mapped with incorrect orientation. (D) Tandem duplication. The read pair spans the middle breakpoint of a tandem duplication. The PEM will have correct orientation but with reverse order. (E) Intra-chromosomal translocation. Two read pairs span the two breakpoints of an intra-chromosomal translocation with one pair having a large mapped distance and the other having correct orientation but their ordering reversed. (F) Inter-chromosomal translocation. The two ends of the pair are mapped to different chromosomes. (G) One-end unmapped. The sequenced genome has a DNA segment that does not exist in the reference genome. One end of the pair is mappable but the other is not. Reproduced from [43]

be considered discontinuous. Moreover exome sequencing can be challenging because, due to a different efficiency of capture probes, the distribution of reads cannot be considered uniform.

Here in the following we will summarize the procedure to call both CNVs and LOHs [44]. Let consider an exon of length L, let X and Y denote the numbers of reads, each of length w, mapped within the exon of interest in case and control respectively. The depth-of-coverage is then Xw/L and Yw/L for case and control, respectively. Although the method is discussed in terms of depth-of-coverage, it is developed in terms of the count statistics X and Y. Let N_X and N_Y be the total numbers of aligned reads in case and control, respectively. The read count ratio is defined as:

$$R = \frac{X/N_X}{Y/N_Y} \tag{3.12}$$

Raw counts X and Y are divided by the total number of reads N_X and N_Y to mitigate the effect of overall increase in local counts due to the increase in total depth-of-coverage. The ratio is adjusted so that the exome-wide median is 1. Without lost of generality, it is assumed that $N_X = N_Y$ and it results that $R = X/Y$. Because X and Y follow Poisson distributions with parameters λ_X and λ_Y, respectively, with sufficient depth-of-coverage the Poisson distributions converge to normals with equal means and variances: $N(\lambda_X, \lambda_X)$ and $N(\lambda_Y, \lambda_Y)$. Under the null hypothesis of no CNV, $\lambda_X = \lambda_Y$, and under the alternative hypothesis, $\lambda_X = \rho\lambda_Y = \rho\lambda$. ρ indicates the copy-number ratio; for example, $\rho = 0.5$ for deletion and $\rho = 1.5$ for duplication. By Geary-Hinkley transformation, let:

$$t(\rho) = \frac{\mu_Y R - \mu_X}{\sqrt{\sigma_Y^2 R^2 + \sigma_X^2}} = \frac{\lambda_Y R - \lambda_X}{\sqrt{\lambda_Y R^2 + \lambda_X}} = \frac{\lambda R - \rho\lambda}{\sqrt{\lambda R^2 + \rho\lambda}} = \frac{(R-\rho)\sqrt{\lambda}}{\sqrt{R^2 + \rho}} \tag{3.13}$$

and $t(\rho)$ follows the standard normal distribution. Thus, the specificity and sensitivity are $1 - \alpha$ and $1 - \beta$ where

$$\alpha = \begin{cases} \phi(t(1)) & \text{if } \rho < 1 \\ 1 - \phi(t(1)) & \text{if } \rho \geq 1 \end{cases} \tag{3.14}$$

$$\beta = \begin{cases} 1 - \phi(t(\rho)) & \text{if } \rho < 1 \\ \phi(t(\rho)) & \text{if } \rho \geq 1 \end{cases} \tag{3.15}$$

These formulas describe the achievable specificity and sensitivity of a given cutoff ratio R. In calling CNV, is therefore necessary to identify a cutoff $r(\rho)$ by solving the above equations, which yields desired minimum specificity and/or sensitivity for testing a particular copy-number ratio ρ at a particular exon with some depth depth-of-coverage and length.

The CNV calling procedure can be summarized as follow:

1. for each exon the algorithm initially calculates the log coverage ratio between case and control;
2. given minimum values for specificity and sensitivity, the algorithm picks the cutoff that optimizes for these values basing on read coverage, exon length, and the estimated admixture rate;
3. a CNV call is made for each exon except for those without sufficient coverage or that cannot achieve the desired power/specificity;
4. exon are then merged into segments using circular binary segmentation;
5. CNVs are therefore called on each segment as described above;
6. the procedure over segment is repeated until the most sensitive segmentation is achieved.

Despite ExomeCNV was specifically used to detect CNV, it is also able to call LOH and the steps in LOH calling procedure are similar to those in CNV calling. In this case, the detection process starts by considering all polymorphic positions in the exome of the control sample, and for each of the positions, the B-allele count, B, is the number of reads with non-reference or B-allele at that position. For a polymorphic position i, N_i is the total number of reads mapped to that position; thus the B-allele count B_i follows a binomial distribution Binomial (p_i , N_i). A binomial that rejects the null hypothesis: $p_i = 0.5$ can be used to detect LOH at each polymorphic position. Then the algorithm will then perform circular binary segmentation and call LOH on each incrementally bigger segment.

3.2.6 SVDetect

SVDetect can be considered as a complementary tool to ExomeCNV in order to perform a complete structural variant analysis. As ExomeCNV, it can construct copy number profiles but it relies on paired end data and mainly searches for genomic rearrangements as large insertions-deletions, inversions, duplications and balanced or unbalanced inter-chromosomal translocations. In particular it uses the a priori information from paired-ends such as order, orientation and insert size as parameters to identify anomalously mapped pairs which can indicate potential genomic variations from the reference. This tool detects structural variations (SV's) by using sliding windows and clustering strategies, and also allows to visualize them at genome scale [45].

First step of SV's calling procedure consists on mapping all anomalous mapped paired-end reads onto a fragmented reference genome. Inputs are pair of reads that either have an incorrect orientation and/or the distance that exists between them is out of a typical range. The reference genome is partitioned into small genomic overlapped regions that are usually big as twice the maximum insert size between ends. SVDetect uses therefore a sliding-window strategy to identify all groups of pairs sharing a similar genomic location and assign each anomalous pair to at least one possible pair of two chromosomal region. The connections generated between two genomic regions are considered links and each link is characterized by a set of features as chromosomal location, number of pairs, orientation and order of the involved paired-ends. After removing redundant links, the remaining ones are characterized in terms of precise coordinates. Finally a sorting procedure occurs and close overlapping links are merged.

The next step consists in filtering all the generated links accordingly to a set of user-defined filtering parameters. Filtered links are considered as significant paired-end mapping clusters and used to call structural variants (SVs). Filters are mainly based on the number of pairs in each link, the consistency of the strand orientation between pairs, the order of reads and the distance insert size between intra-chromosomal ends. A threshold on the minimum number of paired-ends is one of the most important filter parameters and improves the confidence in the detection of SVs. Another option is filtering of pairs whose ends are not oriented in the same way as the ends of the majority of pairs in the two linked regions. These filtering procedure leads to structural variant calling. For example, if one of the two ends of filtered pairs has an unexpected strand orientation, the cluster is annotated as a potential inversion. Moreover, the order of paired-ends is used to annotate inter-chromosomal clusters and to point out if they are balanced or an unbalanced translocations. Insertions or deletions are instead called by evaluating how many standard deviations the insert size of links under analysis differs from the mean of the insert size distribution of aligned reads. Usually values of 2 to 3 standard deviations from the mean are considered enough to call deletions and insertions.

For each sliding windows along the genome, the log-ratio of depth-of-coverage between a sample and a control dataset is finally calculated using all pairs correctly mapped with the expected insert size. This ratio is then used to define copy-number profiles and potential loss or gain events.

SVDetect offers different output formats as the BED format or the Circos link format. In particular, the latter highly helps the subsequent analysis of reported paired-end clusters and a representation can be found in Fig. 3.5.

3.2.7 Variant Quality Score Recalibration

The Variant Quality Score Recalibration (VQSR) step is the last step of variant calling and aims at assigning accurate confidence scores to each SNP or indel call. This procedure is part of GATK and allows to filter variants basing on their accuracy in order to generate highly accurate call sets.

VQSR relies on the fact that mutation calling algorithms are usually very permissive so the resulting call sets need to be filtered. In theory, variants could be filtered by hand tuned filters that usually require a lot of expertise and time to be defined. In practice, it would be better to learn how to filter variants by the data itself. The idea is therefore to build a model of true genetic

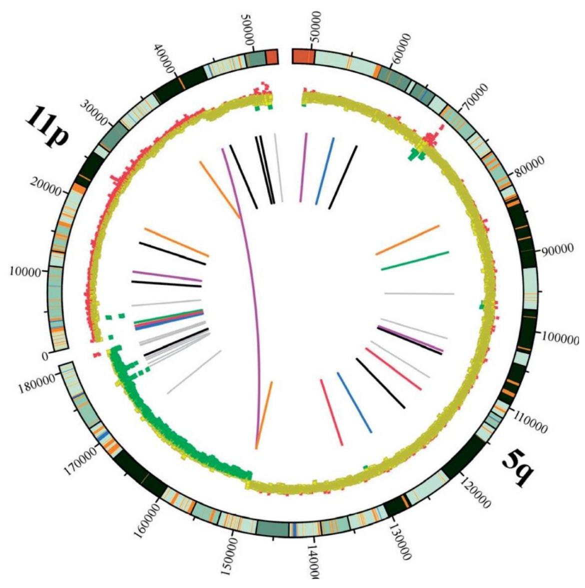


Figure 3.5: **Graphical visualization of predicted SVs** Genomic locations of inter- and intra- chromosomal links are shown using the Circos software. Starting from outside of the circle, the following features are displayed: chromosome ideograms, scatter plot of the copy-number profile and color-coded spans of chromosomal links [45]

variation and rank variants basing on their likelihood of being real. Such an approach should then enable analysts to trade off sensitivity and specificity depending on project goals.

As partially stated when the VCF format has been introduced, each variant has an associated set of statistics that are also called annotations. Variant annotations are different from the annotations that will be introduced in the next section. These annotations are reported in the INFO field of the VCF and some example are stated as follow:

- DP: Depth of coverage;
- AF: Allele frequency;
- AC: No. chromosomes carrying alt allele;
- QD: QUAL score over depth;
- MQ0: No. of MAPQ 0 reads at locus.

The interesting thing is that real variants tend to cluster together via these statistics and these clusters tend to be Gaussian distributed. As a result, the idea is to build adaptively a Gaussian mixture model basing on a subset of variants that overlap with training sites provided as input. A training on high-confidence known sites leads then to determine the probability that other sites

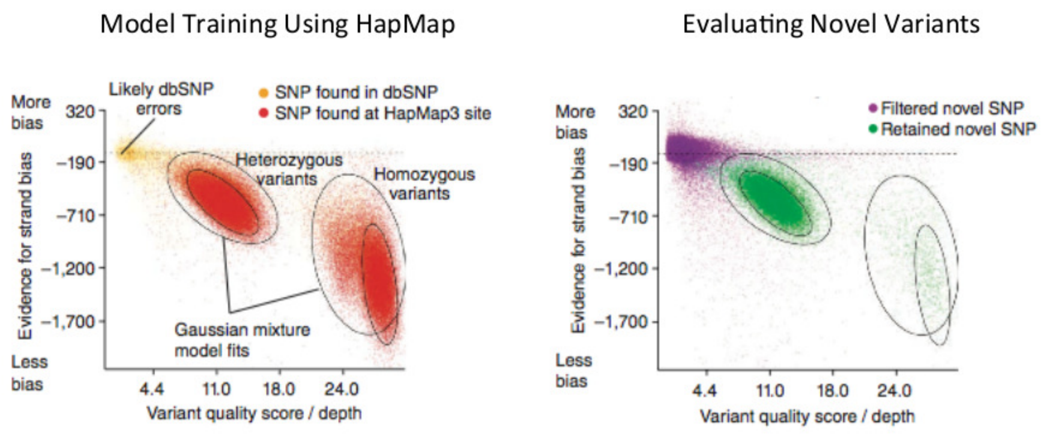


Figure 3.6: **VQSR training and evaluation of variants**

are true and new potential variants can therefore be evaluated against the defined model to evaluate the probability that each call is real.

VQSR can be summarized in two steps performed by two different tools. VariantRecalibrator is the first tools: it creates the Gaussian mixture model using the specified annotation values over a high quality set of training and attempts to group these variants into clusters. Then, it evaluates all input variants and assigns a score to each one of them Fig. 3.6. This score, also called VQSLOD, is the log odds ratio of being a true variant versus being false under the trained Gaussian mixture model.

The ApplyRecalibration tool finally applies user defined filters and write new annotated VCF. Usually these filters are expressed in terms of tranche sensitivity threshold. The recalibrated variant quality score allow indeed to partition the call sets into quality tranches that can be used to establish thresholds within your data that correspond to certain levels of sensitivity relative to the truth sets. Tranche sensitivity threshold are expressed as a percentage; if a 99.9% threshold is fixed, the programs looks at what is the VQSLOD value above which 99.9% of the variants in the training callset are included and uses it as a threshold to filter your variants. Variants that are above the threshold pass the filter, so theVCF's FILTER field will contain PASS. Variants that are below the threshold will be filtered out; they will be written to the output file, but in the FILTER field they will have the name of the tranche they belonged to.

It is worth noting that SNPs and Indels must be recalibrated separately. The VQSR's implementation considered the following databases as training sites to create the model for SNSs:

- dbsnp_137.b37.vcf
- hapmap_3.3.b37.vcf

- 1000G_omni2.5.b37.vcf

while the following have been used for Indels:

- dbsnp_137.b37.vcf
- Mills_and_1000G_gold_standard.indels.b37.vcf

3.3 Functional Annotation

The functional annotation of variants is the last step of the pipeline of analysis. This procedure is vital for the downstream analysis that looks for links between sequence variants and phenotype changes. Variant annotation aims indeed to assign functional information to DNA variants in order to predict their impact. In this section, the annotation procedure implemented in the pipeline will be discussed; moreover, the filtering and prioritizing procedures to find potential disease-causing mutations will be described.

There are several types of information that can be associated with variants; despite most of the available tools focus on the annotation of SNPs, there are few of them that are able to annotate indels and CNVs as well. In particular, the variant annotator implemented in the pipeline is Annovar [46]. Annovar works with SNPs, Indels and CNVs. Furthermore it is a very efficient command-line tool and can work with various genomes.

The first level of annotation implemented defines the relationship between a variant and coding sequences in the genome. Coding sequences are usually referred as genes or more specifically as transcripts and the majority of them encode a protein. There are also non-coding transcripts that, despite do not encoding proteins, can have important functions as the regulatory ones. In order to annotate DNA variants, a set of transcripts is therefore required. Ensembl [47], RefSeq [48] and UCSC [49] are the most popular databases of transcripts that can be used for variant annotation: Annovar can interrogate all of them and can annotate variants accordingly. However most of the time the information retrieved are redundant or discordant so it was chosen to annotate variants only against RefSeq.

A variant can often be shared by different transcripts that overlap in the same genomic position. Annovar reports, for each transcript, if the variant causes protein coding changes and possibly the amino acids that are affected. In accordance to the genomic location, a variant can be classified as reported in Table 3.4. The precedence defined in this table is used to decide what function to print out when a variant fits multiple functional categories; the user is allowed to change this default precedence rule.

The annotation against refSeq is defined as gene-based annotation and generates two different files. The first one adds to the original input file only two column representing the variant classification and the names of genes that are hit by the variant, or are neighbour to it. The second output file contains

Value	Default precedence	Explanation
exonic	1	variant overlaps a coding exon
splicing	1	variant is within 2-bp of a splicing junction
ncRNA	2	variant overlaps a transcript without coding annotation in the gene definition
UTR5	3	variant overlaps a 5' untranslated region
UTR3	3	variant overlaps a 3' untranslated region
intronic	4	variant overlaps an intron
upstream	5	variant overlaps 1-kb region upstream of transcription start site
downstream	5	variant overlaps 1-kb region downtream of transcription end site
intergenic	6	variant is in intergenic region

Table 3.4: **The possible values of gene annotation**

the amino acid changes as a result of the exonic variant and the possible values are reported in Table 3.5. Once the gene-based annotation is performed, the user can focus on a subset of variants depending on their classification. Most of the time interesting variants are considered relevant if they hit an exon and cause an amino acid change; in general, interestingness highly depends on the aim of the study.

Another fundamental level of annotation is based on the comparison between variants under analysis and variant databases. This kind of annotation aims at exacting from public databases variants with same start and end positions, and with same observed alleles to the processed ones. Two resources that are usually used are dbSNP and 1000 Genomes Project databases. If a variant is annotated as present in 1000 Genomes Project database, Annovar usually outputs the frequency of that variant in the population. Such a frequency can help to filter out variants that can be considered polymorphisms in the population and that are therefore not related to the disease of interest. The same process can be applied to variants that are reported in dbSNP with a MAF bigger than 1%, so that these variants are usually filtered for further analysis.

This kind of annotation is defined as filter-based and relies on different databases. Despite databases of frequencies are really helpful, there are other databases that are potentially more useful. In particulat, there are several databases developed for functional prediction and annotation of potential non-synonymous single-nucleotide variants (nsSNVs) in the human genome. Annovar for example uses dbNSFP [50] that compiles prediction scores from several prediction algorithms. In particular, the version we have used in our pipeline includes SIFT scores, PolyPhen2 HDIV scores, PolyPhen2 HVAR scores, LRT scores, MutationTaster scores, MutationAssessor score, FATHMM scores, GERP++ scores, PhyloP scores and SiPhy scores. These scores are computed based on various different approaches, such as protein structure, sequence homology, evolutionary conservation or statistical prediction based on known mutations. Moreover they can be then translated in categorical predictions as deleterious/tolerated or probably damaging/possibly damaging/benign and can be used to filter variants that are considered as neutral.

Another interesting features of Annovar is that it allows users to supply a custom-made annotation file. Annovar will then perform filter-based annotation on this annotation file. An example of a custom made annotation file will be provided in this thesis in the results section.

Annotation	**Default precedence**	**Explanation**
frameshift insertion	1	an insertion of one or more nucleotides that cause frameshift changes in protein coding sequence
frameshift deletion	2	a deletion of one or more nucleotides that cause frameshift changes in protein coding sequence
frameshift block substitution	3	a block substitution of one or more nucleotides that cause frameshift changes in protein coding sequence
stopgain	4	a nonsynonymous SNV, frameshift insertion/deletion, nonframeshift, insertion/deletion or block substitution that lead to the immediate,creation of stop codon at the variant site. For frameshift mutations, the creation of stop codon downstream of the variant will not be counted as ”stopgain”!
stoploss	5	a nonsynonymous SNV, frameshift insertion/deletion, nonframeshift, insertion/deletion or block substitution that lead to the immediate,elimination of stop codon at the variant site
nonframeshift insertion	6	an insertion of 3 or multiples of 3 nucleotides that do not cause frameshift changes in protein coding sequence
nonframeshift deletion	7	a deletion of 3 or mutliples of 3 nucleotides that do not cause frameshift changes in protein coding sequence
nonframeshift block substitution	8	a block substitution of one or more nucleotides that do not cause frameshift changes in protein coding sequence
nonsynonymous SNV	9	a single nucleotide change that cause an amino acid change
synonymous SNV	10	a single nucleotide change that does not cause an amino acid change
unknown	11	unknown function (due to various errors in the gene structure definition in the database file)

Table 3.5: **The possible values of exonic annotation**

Chapter 4

Genomic workflow management systems

Despite the development of next generation sequencing technologies has opened new scientific possibilities in biology and medicine, the management and analysis of such a flood of data has expanded the informatics needs for research laboratories. The current bottleneck is not anymore the sequencing itself but the storage and backup of the data as the computational infrastructure to perform the analysis. Most of the previously introduced algorithms can be indeed highly demanding in terms of computational resources and difficult to use because of several tuning parameters. Moreover a lot of programming and informatic expertise are necessary to install these tools and to combine them in a pipeline. As a consequence, in order to coordinate all the technological aspects related to NGS data a lot of IT skills, which span from backup strategies to the management of computing demand and from pipeline implementation to results sharing are needed. This chapter will initially discuss how an implemented solution allows scientists without programming experience to run the previously defined NGS workflow and to easily include new computational tools in this pipeline. Then, it will deepen an implementation of the analysis pipeline through a library for workflow management that can be run both on a traditional computing resource as well as on a cloud-based one.

4.1 GenePattern

After pipeline's definition, all the efforts were focused on making both the whole workflow of analysis and all the single tools accessible to experimental biologists and physicians. Although they possess the biological knowledge and experience that can led to novel discovery, without an informatic support they often cannot effectively use the huge amount of sequencing data because of the informatic challenges they pose. Moreover the challenges are even more relevant when multiple methods are combined in the analysis as for NGS pipeline.

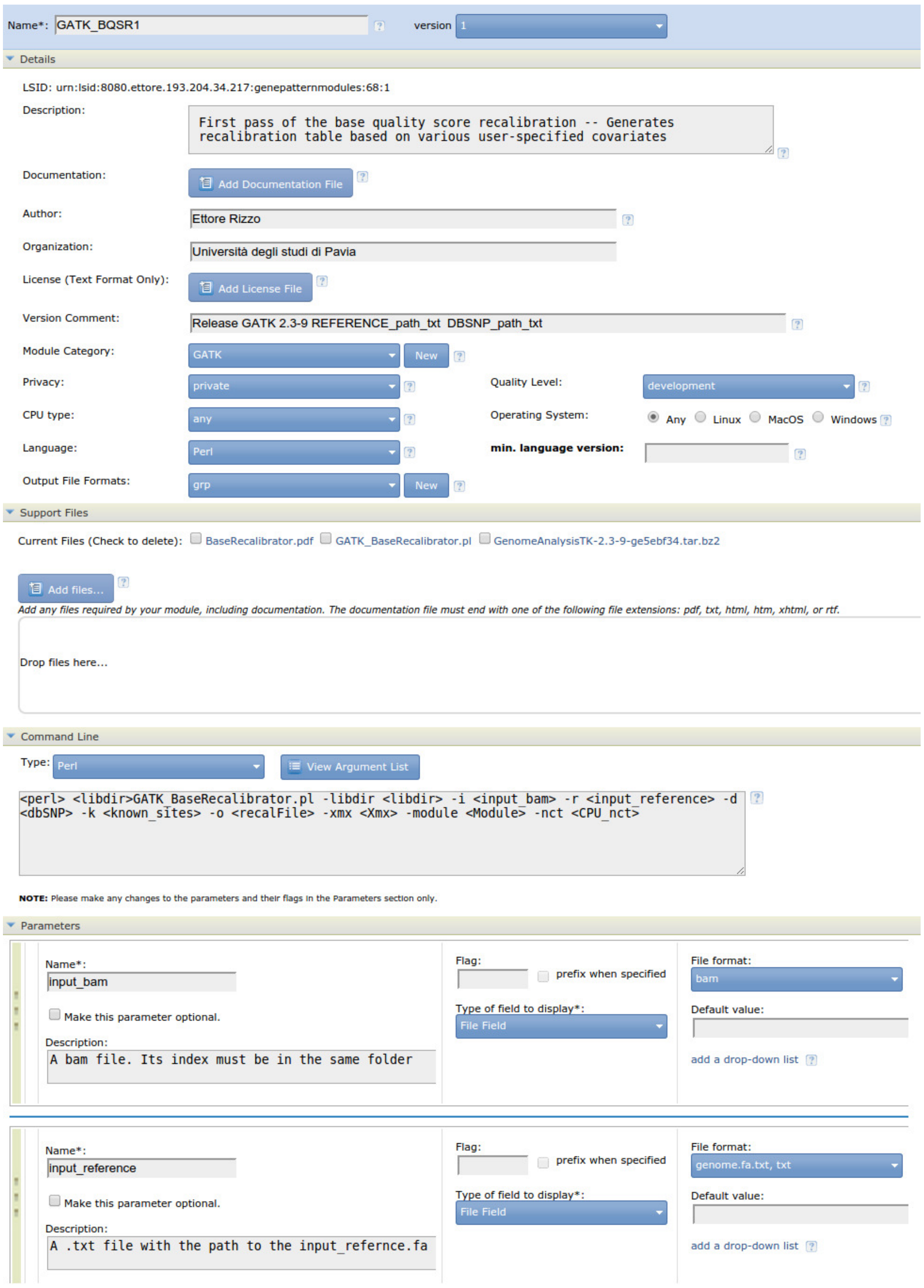
Simultaneously with the introduction of NGS technologies different platforms of analysis where developed to address the biomedical research challenges previously defined. These platforms provide simple interfaces to powerful tools and usually support users at all levels of computational experience by providing both web interfaces or downloadable package that can be deployed in individual laboratories. In particular GenePattern [51] and Galaxy [52] have become the most popular choices for integrated analysis that range from gene expression to next generation sequencing. As stated on the GenePattern website "it's a powerful genomic analysis platform that provides access to hundreds of tools for gene expression analysis, proteomics, SNP analysis, flow cytometry, RNA-seq analysis, and common data processing tasks. A web-based interface provides easy access to these tools and allows the creation of multi-step analysis pipelines that enable reproducible in silico research". Galaxy instead is defined as "an open, web-based platform for data intensive biomedical research. Whether on the free public server or your own instance, you can perform, reproduce, and share complete analyses". Both have a web-based interface for working with tools and provides a framework for adding new tools to the platform. Moreover they automatically generate metadata when tools are run and support repeatability. Currently, GenePattern and Galaxy are indeed the most mature platforms that support reproducible computational research and are often considered complementing each other. The concept of Reproducible Research System (RRS) is nowadays considered crucial and refers to the idea that the full computational environment used to produce scientific results such as the code and the data should be easily accessible in order to reproduce results and foster faster advances [53].

In the following we will detail the main concepts behind GenePattern, since this system was chosen to define the analysis workflow defined in the previous chapter.

4.1.1 Modules

Analysis modules are at the heart of GenePattern and implement computational methods and tools for analysis of genomic data. A module can be any software for which a command line invocation can be constructed. As already stated, the GenePattern public repository provides several modules of analysis and each one includes its own documentation, which is supplied by the module developer. However, this repository doesn't include the majority of the analysis tools used for NGS sequencing analysis described in the previous chapter. The platform, however, allows users to define new modules or to manipulate existing ones. In order to create a new module, GenePattern opens the Module Integrator window.

In Fig. 4.1 it is reported an example of an ad-hoc module to perform the first step of GATK base quality score recalibration. The *Details* section will ask for the following parameters that allow to better characterize the module:



Name*: GATK_BQSR1 version 1

Details

LSID: urn:lsid:8080.ettore.193.204.34.217:genepatternmodules:68:1

Description:
```
First pass of the base quality score recalibration -- Generates
recalibration table based on various user-specified covariates
```

Documentation: Add Documentation File

Author: Ettore Rizzo

Organization: Università degli studi di Pavia

License (Text Format Only): Add License File

Version Comment: Release GATK 2.3-9 REFERENCE_path_txt DBSNP_path_txt

Module Category: GATK New

Privacy: private

Quality Level: development

CPU type: any

Operating System: [✓ Any] Linux MacOS Windows

Language: Perl

min. language version:

Output File Formats: grp New

Support Files

Current Files (Check to delete): BaseRecalibrator.pdf GATK_BaseRecalibrator.pl GenomeAnalysisTK-2.3-9-ge5ebf34.tar.bz2

Add files...

Add any files required by your module, including documentation. The documentation file must end with one of the following file extensions: pdf, txt, html, htm, xhtml, or rtf.

Drop files here...

Command Line

Type: Perl View Argument List

```
<perl> <libdir>GATK_BaseRecalibrator.pl -libdir <libdir> -i <input_bam> -r <input_reference> -d
<dbSNP> -k <known_sites> -o <recalFile> -xmx <Xmx> -module <Module> -nct <CPU_nct>
```

NOTE: Please make any changes to the parameters and their flags in the Parameters section only.

Parameters

Name*: input_bam

Make this parameter optional.

Description:
```
A bam file. Its index must be in the same folder
```

Flag: prefix when specified

Type of field to display*: File Field

File format: bam

Default value:

add a drop-down list

Name*: input_reference

Make this parameter optional.

Description:
```
A .txt file with the path to the input_refernce.fa
```

Flag: prefix when specified

Type of field to display*: File Field

File format: genome.fa.txt, txt

Default value:

add a drop-down list

Figure 4.1: **Details of module integrator sections.**

- Name
- Description
- Documentation: it is strictly recommended to upload a pdf file that will represent the documentation for the module.
- Author
- Organization
- License
- Version Comment
- Module Category
- Privacy: ”public” allows all users connected to this server to view and run the module while ”private” means that only the creator or the administrator can view and run the module.
- Quality Level
- CPU type
- Operating System
- Language: user can choose C, C++, Matlab, Perl, Python, Java, R or any
- Output File Formats: a file format can be selected from the list of formats.

By the *Support Files* section the user then uploads the documentation file and all the files that are necessary to run a specific software. Fig. 4.1 module requires the GATK package and a perl file that is used to invoke it.

The *Command Line* field usually requires a combination of fixed text and variables defined by GenePattern. The command line defined in Fig. 4.1 uses several variables: <perl> represents the full path to the perl installation used by GenePattern while <libdir> represents the full path to the directory that contains the files for this module, including the program file. Then the perl script GATK_BaseRecalibrator.pl expects several parameters, as the <input_bam> file, the <input_reference> file and other parameters that have to be defined in the *Parameters* section. In this section all the parameters needed are defined in term of name, description, flag, type of field to display and file format.

All the workflow steps described in the previous chapter that weren’t available in the public repository, were therefore defined as new GenePattern modules.

4.1.2 Pipelines

NGS data processing as many other type of genomic analysis are multistep sequences of computational tools that should be integrated in unique workflows. In order to avoid writing programs that join these tools together, GenePattern provides a pipeline builder that can define reusable template of analysis. It is easy-to-use and allows to create a workflow from scratch by adding all the necessary tools and conveniently connecting them.

Fig. 4.2 shows how it works: by a graphical user interface users can add and connect modules to form a complete analysis. The pipeline designer comprises three main parts. From left to right it includes the module selection panel, the pipeline diagram and the editing panel. Initially the modules are browsed from the module selection panel in order to add the right ones (all the installed ones can be added to the workflow). Tools are then combined together by using modules' outputs as input for others and the pipeline builder verifies if these links are compatible. Finally additional parameters of each tool can be edited by using the editing panel. The result is a workflow that can be run repeatedly on different data without any changes.

Once a pipeline has been defined it can be easily exported. This process will generate a zip package that include the pipeline, all the modules that compose it and the associated parameters. This would allow to share analysis methods as single executable script, enabling in-silicio reproducible research.

NGS analysis modules were then assembled together to form a robust GenePattern pipeline that requires FastQ files as input and generates annotated files of variants.

4.1.3 Servers

GenePattern can run both on a publicly available server or on a local server. The publicly available GenePattern server is hosted at
http://genepattern.broadinstitute.org/gp/ and can be used without installing any software. This solution implies that users don't have to maintain the server but as a counterpart only the modules and pipelines hosted on this server can be used for an analysis. Moreover this solution can be adversely affected by time consuming data transfer from and to the server, in particular for NGS data.

On the contrary a local installation of the GenePattern server has several advantages. First of all, users can analyze their data without sending it over the internet but most importantly, modules and pipelines that are installed on the local server can be ad-hoc created and are not uniquely the ones hosted in the GenePattern's public repository (that can however be downloaded if needed). Moreover a local installation can be performed both on a standalone machine or on a networked machine. Finally, if there is a queuing system installed on the local server as LSF or SGE, a local installation allows to prioritize jobs and to assign the right amount of resources to each one of them. As a consequence,

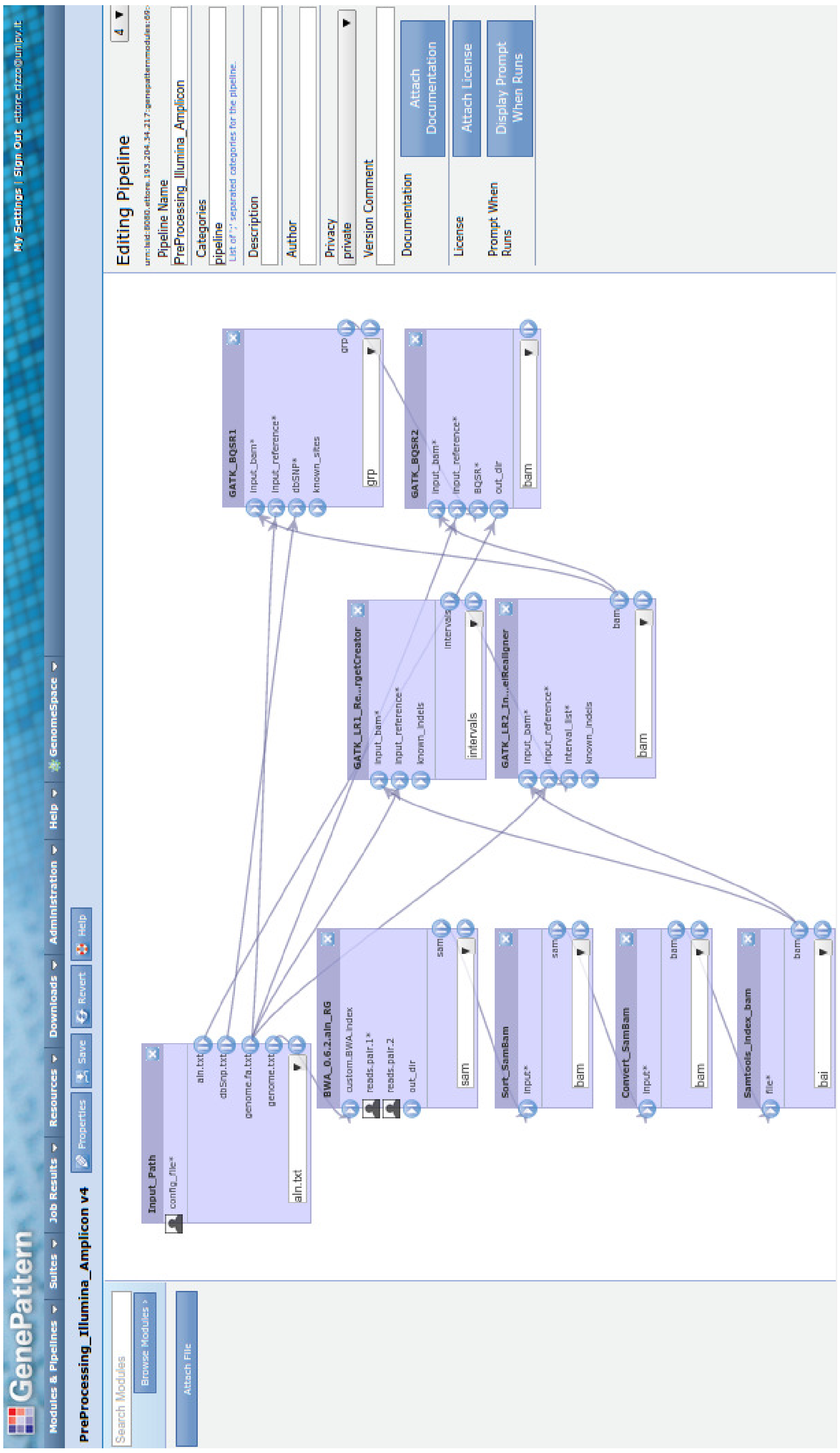


Figure 4.2: **GenePattern Pipeline editor.**

the local GenePattern installation was preferred above the publicly available server option.

4.1.4 Jobs

The execution of a module or a pipeline generates a specific job. In particular, the web browser sends a request to the GenePattern server to run the analysis and a job is created. Job results are then summarized in the Job Result Summary page and are usually stored on GenePattern server for a set period of time before deletion. If the GenePattern server is running, jobs are not interrupted even if the user exit the web browser. Otherwise, if the server is shut down during jobs execution, all the jobs are interrupted and restarted automatically when the server is restarted.

Finally it is worth noting that the GenePattern server can be configured to use an installed queuing system. In particular it includes support for Sun Grid Engine (SGE) and LSF. These queuing system strongly improves the execution performances. In order to configure the server with SGE or LSF, the user has only to properly edit the configuration file and restart the server. However GenePattern's interaction with the queuing system can be also configured either programmatically or with a command line prefix.

4.1.5 CABGen Server Install

A local GenePattern server was installed on a system of high performance computing implemented at the Center for Bioinformatic Analysis for Genomics (CABGen). This center was born in 2010 at the Institute of Molecular Genetics, Pavia, IT (an institute that belongs to the network of research institutes of the Italian National Research Council) and its computational hardware is constituted of a cluster IBM with six nodes, each of which characterized by two processors IntelXeon 4-core, 42 Gigabyte of memory capacity and equipped with two solid-state drives of 50 gigabytes.

This cluster is managed by a front-end IBM X3650 M3 with two CPUs IntelXeon from 4 cores, 36 gigabytes of memory, two 250 Gigabyte hard and communicates with the cluster through two switches with 10 Gigabit connections. Moreover it is equipped with 8 disks in RAID 5 of one terabyte each; such a storage is seen via NFS from the front-end to all nodes in the cluster and a backup unit guarantees against loss of data. The cluster is managed with the Linux oriented operating system CentOS vers. 5.5 that has been configured and optimized to use parallel computing, in particular MPI. Jobs are submitted to the system through the PBS software that allows the user to choose how many cores allocate to the job.

After an agreement between the CABGen staff and the Department of Hematology Oncology, the GenePattern server was installed on the front end node. However due to several ongoing project on this computational resource, 1 TB of storage was allocated for NGS analysis and it was allowed to submit jobs

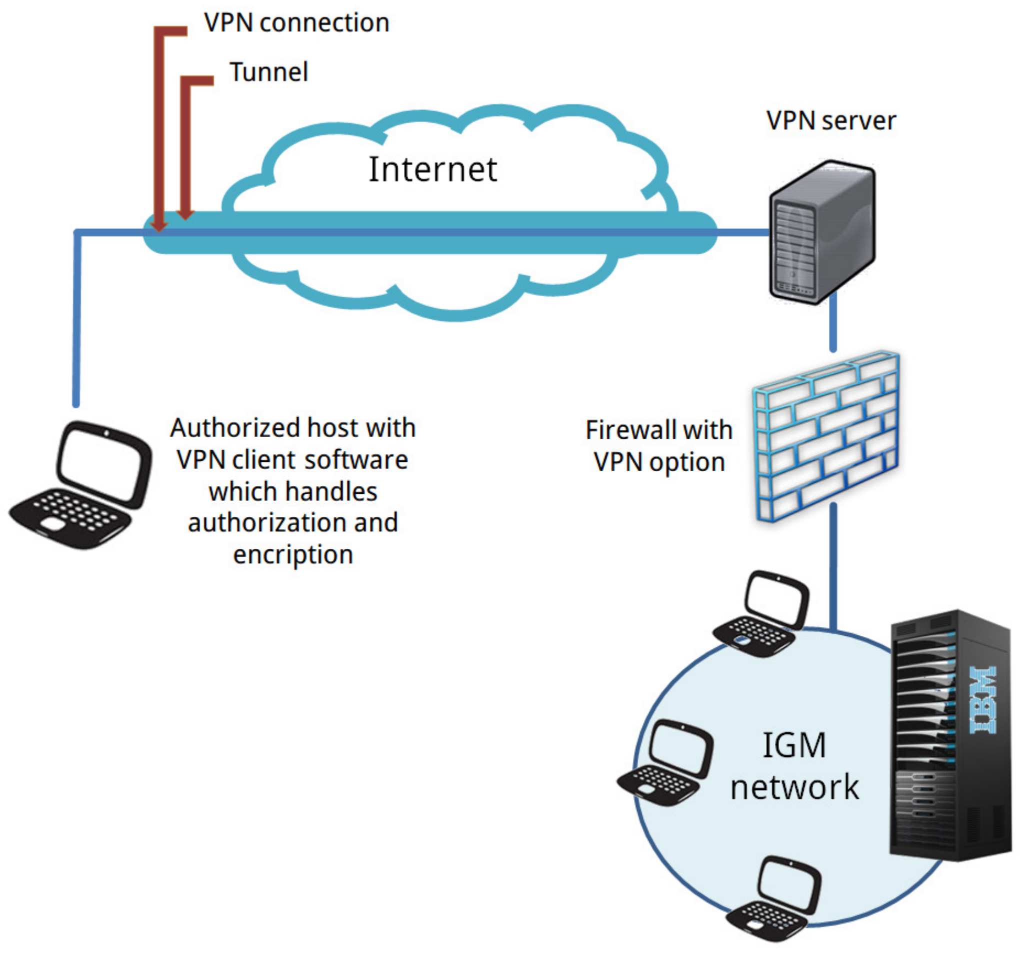

VPN connection
Tunnel
Internet
VPN server
Authorized host with VPN client software which handles authorization and encription
Firewall with VPN option
IGM network
IBM

Figure 4.3: **VPN tunneling to IGM's network.**

only to a specific node so GenePattern was configured accordingly. Finally a VPN was enabled to allow a workstation inside the Department of Hematology Oncology to reach the network of the Institute of Molecular Genetics where the computing system has been integrated (Fig. 4.3).

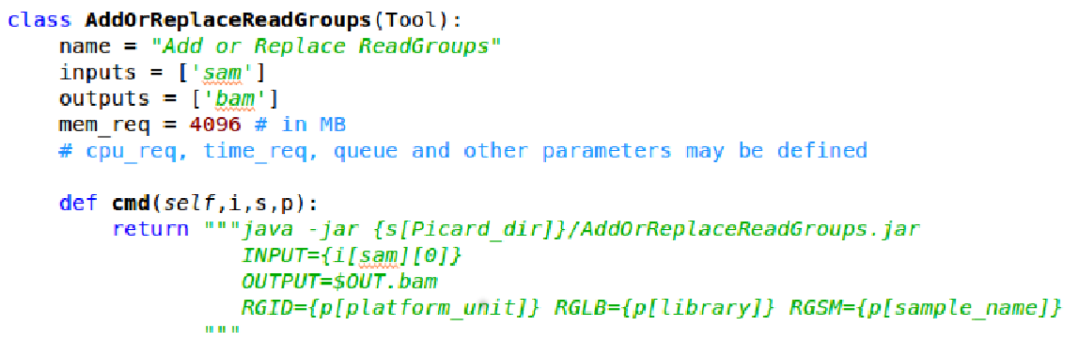
All the NGS modules and pipelines imported in this GenePattern server were previously developed on a personal computer where GenePattern was configured to submit job through SGE. However in this case, given that GenePattern doesn't support PBS job scheduling software and as the agreement set a limit on the amount of the available resources for NGS analysis, the interaction between GenePattern and the queuing system was configured with a command line prefix. Such a configuration was a good compromise to make biologists and physicians able to run the implemented pipeline, visualize the results and share analysis. Once connected to the server through the VPN, scientists can indeed upload their own datasets into the workspaces and use pipeline and workflow already defined. Moreover, as GenePattern automatically captures the history of any computational work being done, in order to facilitate publishing reproducible results the Microsoft Word add-in for the GenePattern was used to embed pipelines in Word documents.

Although this configuration has satisfied researcher needs, an integration with the queuing system by command line has significant drawbacks. In particular it is neither suitable for high volume of jobs nor for long running jobs. Command line prefix is indeed only appended to the specific command line defined in a module. Once started, each new job requires a dedicated server process that waits for the job to complete. However, if the user terminates a job, only the server process is terminated while the process launched on the queuing system is not terminated. This happens also when the server shuts down: once restarted the jobs are not restarted and any unfinished job must restart from the beginning. User jobs may be therefore only terminated but not restarted.

4.2 Cosmos

Software package as GenePattern have become popular within the scientific community because they allow the creation and execution of complex workflows. However NGS analysis, in particular exome or genome analysis, can potentially need hundreds of computing hours per sample and can be performed only by powerful computing clusters or cloud infrastructure. Despite powerful, GenePattern can be adversely affected by big amount of input data and, although it can be configured to interact with queuing system as LSF or Sun Grid Engine, it cannot easily deploy workflows onto these distributed resource management systems (DRMs) when clusters include thousands of computing cores. Therefore, in the following we will discuss COSMOS, a Python library that address these and other needs. COSMOS is a library for workflow management that allows formal description of pipelines and partitioning of jobs; workflows can be created both on traditional computing clusters or cloud-based services.

Thanks to a powerful syntax it allows to create complex pipelines and even programmer with limited experience can easily learn how to write workflows. The latter can be monitored during a run because of COSMOS ability to keep track of job information and resource utilization by a SQL database. Via a web interface is it possible to monitor all the step by looking at the information stored in the database and, thanks to such a fine-grained control over the whole process, it is possible to quickly debug new or existing workflows.



```
class AddOrReplaceReadGroups(Tool):
    name = "Add or Replace ReadGroups"
    inputs = ['sam']
    outputs = ['bam']
    mem_req = 4096 # in MB
    # cpu_req, time_req, queue and other parameters may be defined

    def cmd(self,i,s,p):
        return """java -jar {s[Picard_dir]}/AddOrReplaceReadGroups.jar
                  INPUT={i[sam][0]}
                  OUTPUT=$OUT.bam
                  RGID={p[platform_unit]} RGLB={p[library]} RGSM={p[sample_name]}
               """
```

Figure 4.4: **Cosmos tool definition.** Tools are defined by specifying input and output types, not files, and a cmd() function returns a string to be executed in a shell.

4.2.1 Tools

A Tool is a class which instance represents a command that is executed. A main feature of COSMOS is its ability to separate the definition of tools from the definition of the dependencies between them. Most of the workflow management software generally controls I/O by requiring specifically named input files and produce similarly specific output files. On the contrary, COSMOS controls tool I/O by specifying file types. Therefore I/O tools may expect a specific extension-typed input and generate a specific extension-typed output where both aren't related to file names or locations. For example, a tool in Fig. 4.4 expects a SAM-typed input and produces a BAM-typed output. Besides the expected input and output types, a tool also contains properties which define the resources that are required. Fig. 4.4 shows for example that 4096 MB are required as ram memory to run this tool.

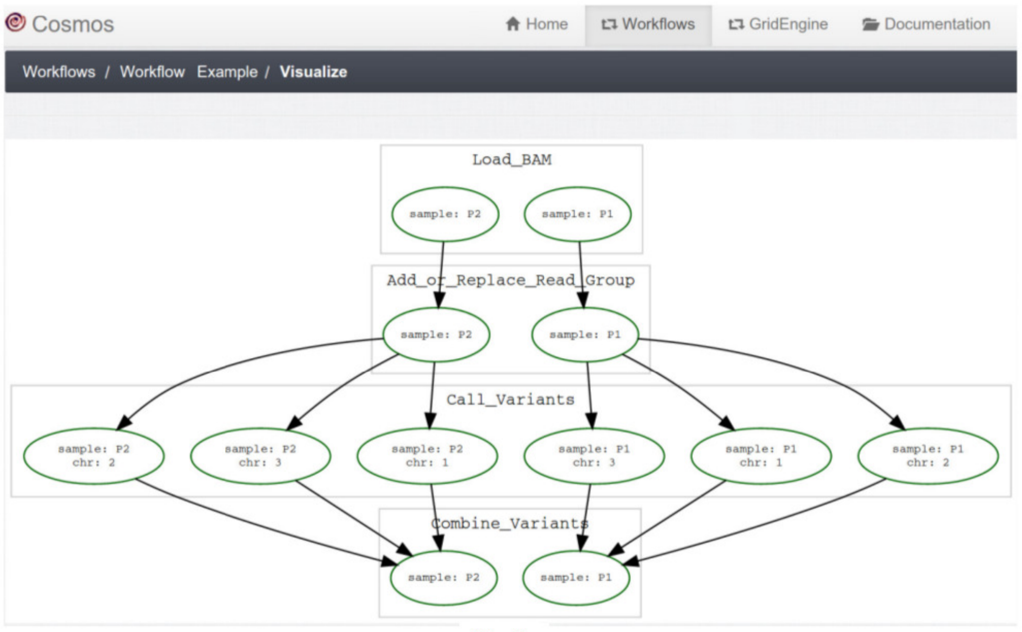
4.2.2 Workflows

A COSMOS workflow then formalizes tools' dependencies by using Python functions that support the map-reduce paradigm [54] (Fig. 4.5(a)). The *sequence_* and *apply_* primitive are respectively provided to execute tools in series or in parallel. In order to implement the map-reduce paradigm in large and branching workflows, COSMOS implements also a tagging system that associates a set of key-value tags to each task in the stage. This feature enables to formalize reduction over existing tag sets or to split by creating new combination of tags. When a workflow is executed, COSMOS generates a directed acyclic graph (DAG) that models the tools dependencies and links the inputs and outputs between them by recognizing file extension and types (Fig. 4.5(b)). In order to design a workflow and represent all the jobs and their dependencies in a DAG, five additional operators to *sequence_* and *apply_* can be used:

- The *add_* operator is always the first operator and simply adds a list of tool instances to the DAG, without adding any dependencies.

```
inputs = [ INPUT(file, tags={'sample':s}) for file,s in input_data ]
chrs = ('chr',range(1,3))
dag = DAG().sequence_(
    add_(inputs, stage_name='Load_BAM'),
    # Add or Replace Read Group to each BAM
    map_(tools.AddOrReplaceReadGroups, stage_name='Add_or_Replace_Read_Group'),
    # Separately call variant on chr 1,2 and 3
    split_(['chr'], tools.variantcall, stage_name='Call_Variants'),
    # Combine variants per sample
    reduce_(['sample'], tools.combinevariants, stage_name='Combine_Variants')
)
```

(a)



(b)

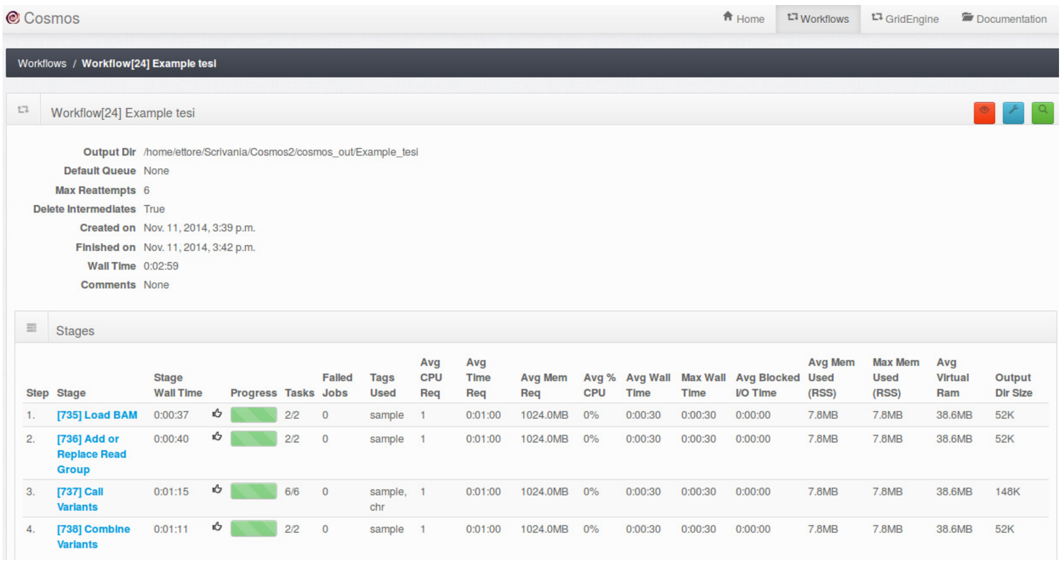
Figure 4.5: **(a) Cosmos Workflows definition. (b) Directed acyclic graph of jobs generated by the workflow.**

- The *map_* operator creates a one to one relationships for each tool in the stage last added to the DAG.

- The *split_* operator creates a one to many relationships for each tool in the stage last added to the DAG.

- The *reduce_* operator creates new tools with a many to one relationships to parent tools.

- The *reduce_split_* operator creates new tools by first reducing then splitting.



Workflow[24] Example tesi

Output Dir	/home/ettore/Scrivania/Cosmos2/cosmos_out/Example_tesi
Default Queue	None
Max Reattempts	6
Delete Intermediates	True
Created on	Nov. 11, 2014, 3:39 p.m.
Finished on	Nov. 11, 2014, 3:42 p.m.
Wall Time	0:02:59
Comments	None

Stages

Step	Stage	Stage Wall Time	Progress	Tasks	Failed Jobs	Tags Used	Avg CPU Req	Avg Time Req	Avg Mem Req	Avg % CPU	Avg Wall Time	Max Wall Time	Avg Blocked I/O Time	Avg Mem Used (RSS)	Max Mem Used (RSS)	Avg Virtual Ram	Output Dir Size
1.	[735] Load BAM	0:00:37	🟩	2/2	0	sample	1	0:01:00	1024.0MB	0%	0:00:30	0:00:30	0:00:00	7.8MB	7.8MB	38.6MB	52K
2.	[736] Add or Replace Read Group	0:00:40	🟩	2/2	0	sample	1	0:01:00	1024.0MB	0%	0:00:30	0:00:30	0:00:00	7.8MB	7.8MB	38.6MB	52K
3.	[737] Call Variants	0:01:15	🟩	6/6	0	sample, chr	1	0:01:00	1024.0MB	0%	0:00:30	0:00:30	0:00:00	7.8MB	7.8MB	38.6MB	148K
4.	[738] Combine Variants	0:01:11	🟩	2/2	0	sample	1	0:01:00	1024.0MB	0%	0:00:30	0:00:30	0:00:00	7.8MB	7.8MB	38.6MB	52K

Figure 4.6: **COSMOS web application.**

4.2.3 Additional features

COSMOS supports for DRMS such as SGE, LSF, PBS/Torque, and Condor by utilizing the DRMAA library to manage job submission, status checking and error handling. All COSMOS internal data as job queuing end execution data or workflows data are stored in a database through the Django framework: MySQL, PosgreSQL, Oracle and SQLite are supported. This information is then accessible by a web application in order to visualize the advancement of a workflow and debug it in case of failure. COSMOS Web (Fig. 4.6) can indeed show the whole workflow or it can focus on specific stage by interrogating the persistent database. The latter is also fundamental to restart failed pipelines and because of it a workflow can restart from the last successful stage. The web interface finally allows to draw the DAG as the one in Fig. 4.5(b).

4.2.4 Cloud Install

The GenePattern server installed on CABGen hardware was aimed at biologists and physicians without programming expertise and it has solved the majority of the problems related to the creation or execution of workflows as well as results management, thanks to its graphical user interface. However, despite this platform was very useful for researchers that weren't bioinformaticians, it has several power limitation. First of all, in order to parallelize an analysis step into several chunks, as COSMOS does, all the input stages must be manually created for each chunk the user want to implement. It means that chunks must be fixed a priori while COSMOS resolves this problem by creating the DAG at runtime. If this limitation can be moderate for small experiments, it becomes critical for big experiments that can drastically reduce runtime by using parallelization. As a result, the GenePattern server was good enough to analyze few samples but it was definitely too slow with an increasing number of samples. This is a consequence both of platform features and the limited computing resources available on CABGen server.

To overcome such computational restriction and scale the level of parallelization, the pipeline defined in the previous chapter was therefore implemented on top of COSMOS and the whole system was deployed on the cloud. Recently all the major tech giants have entered into the cloud computing market. Among different cloud platforms, Amazon Web Services (AWS) is definitely the most reliable because it has been the first one built and benefits from the experience gained during the years. Moreover throughout the year, AWS evaluates academic research support proposals from active faculty at accredited universities and grants are in the form of credits applicable to AWS services. In 2013, the university laboratory where this thesis was carried on, the Laboratory for Biomedical Informatics “Mario Stefanelli”, was granted by AWS and part of the credits have been used to perform NGS analysis on the cloud. In order to understand the configuration that will be explained later it is necessary to introduce some concept related to AWS.

Amazon EC2

The Amazon Elastic Compute Cloud (Amazon EC2) is the AWS web service that provides resizable compute capacity in the cloud. It is a virtual environment and allows to launch instances with different operating system, load instances with custom environment, control the network's access permission and scale the number of desired system. It gives user the complete control of the computing resources which are paid hourly.

Instances

Instances are the core of Amazon EC2 that provides a wide selection of them. Users can choose between different instance options that differ on CPU, memory, storage and networking capacity. In particular, instances are grouped in several instance types, each one optimized for different use cases as memory-intensive applications or highly computational demanding ones. Instance types then includes different instance sizes.

Finally, it is worth noting that instances come with templated Amazon Machine Image (AMI) to get up and running immediately. Users can also define their own AMI containing personal applications, libraries, data, and associated configuration settings.

Pricing

Besides the possibility to choose the number and the size of computing instances, users can choose different purchasing options. ”On-Demand” instances have a fixed hourly rate and users pay as long as they use them; ”Reserved” instances are paid upfront with a significant discount and are reserved for the whole time interval that was paid. Finally one of the most interesting solutions are ”Spot” instances that enable users to bid on the price they want to pay for the instance capacity; users can reach higher saving but if the Spot Price, that fluctuates based on supply and demand for instances, moves higher than a customer’s maximum price, the customer’s instance will be shut down by Amazon EC2.

Cluster Setup

In order to have an high level of parallelization, it is necessary to setup a cluster computing environment; here in the following we will show the main steps that are necessary to configure an AWS cluster that support COSMOS features and then we will describe the solution used to run the whole NGS project .

Firstly, in order to create a cluster, it is necessary to launch the required instances by using the AWS EC2 platform. All AWS instances come with private IP addresses that are used to let the instances in the same network communicate: once cluster instances are created, the hosts file of each instance should therefore be edited in order to map other instances hostnames to their IP addresses and let the cluster nodes able to communicate. Next, a queuing system should be installed to queue jobs that are generated during a workflow execution. The cluster should then be configured for folder and disk sharing. COSMOS requires indeed to be installed only on a cluster node but the install folder should be shared with the remaining nodes. Accordingly to that, COSMOS can be installed on the master node home directory which has then to be NFS shared with the other nodes in the cluster and mounted on the same path as in the master node. Finally, another folder has to be shared; this folder can be accessed by COSMOS to read inputs and write outputs and has to be accessible from all the nodes in the cluster.

The previous steps are necessary to create a cluster upon which to install COSMOS. In order to facilitate the cluster building process, an open source software called StarCluster can be used to automate the whole process. It simplifies the process of building, configuring, and managing clusters of virtual machines on Amazon’s EC2 cloud: user has to define a configuration file and a cluster will be created accordingly. In particular StarCluster launches the defined number of instances and names all the node in the cluster with a simple convention as master, node001, node002, etc. By default a non-root user account is created and the cluster is configured so that a secure ”ssh” connection may be used from any node in the cluster to connect to any other node in the cluster without using a password. Furthermore the /home directory on the master node is NFS-shared across the cluster and the Sun Grid Engine queueing system is configured for distributing tasks, or jobs, across the cluster.

Because of these features, which are strictly required by COSMOS, Star-

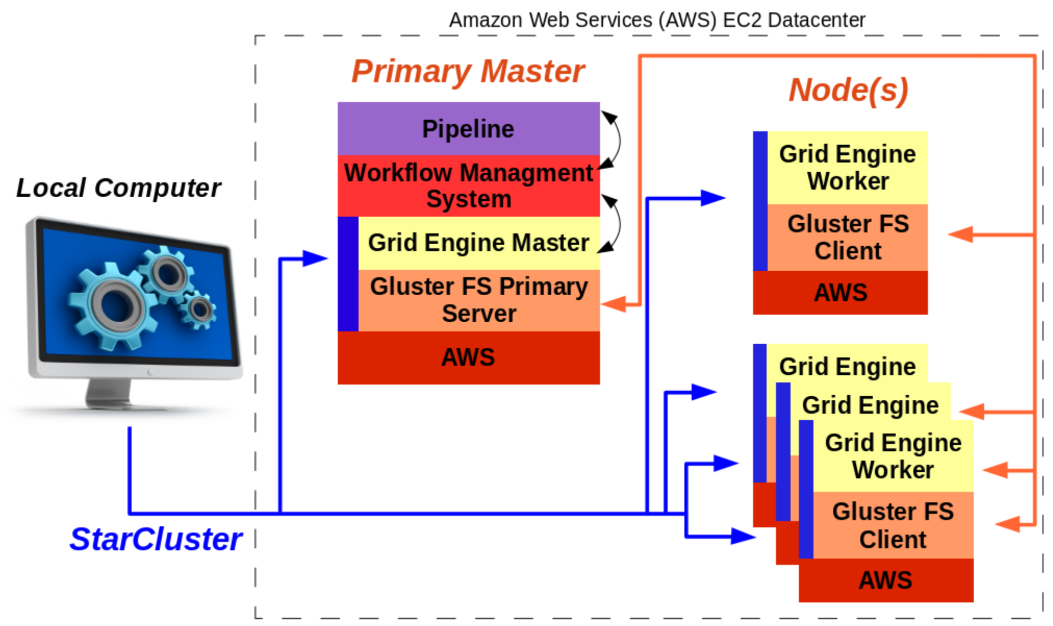


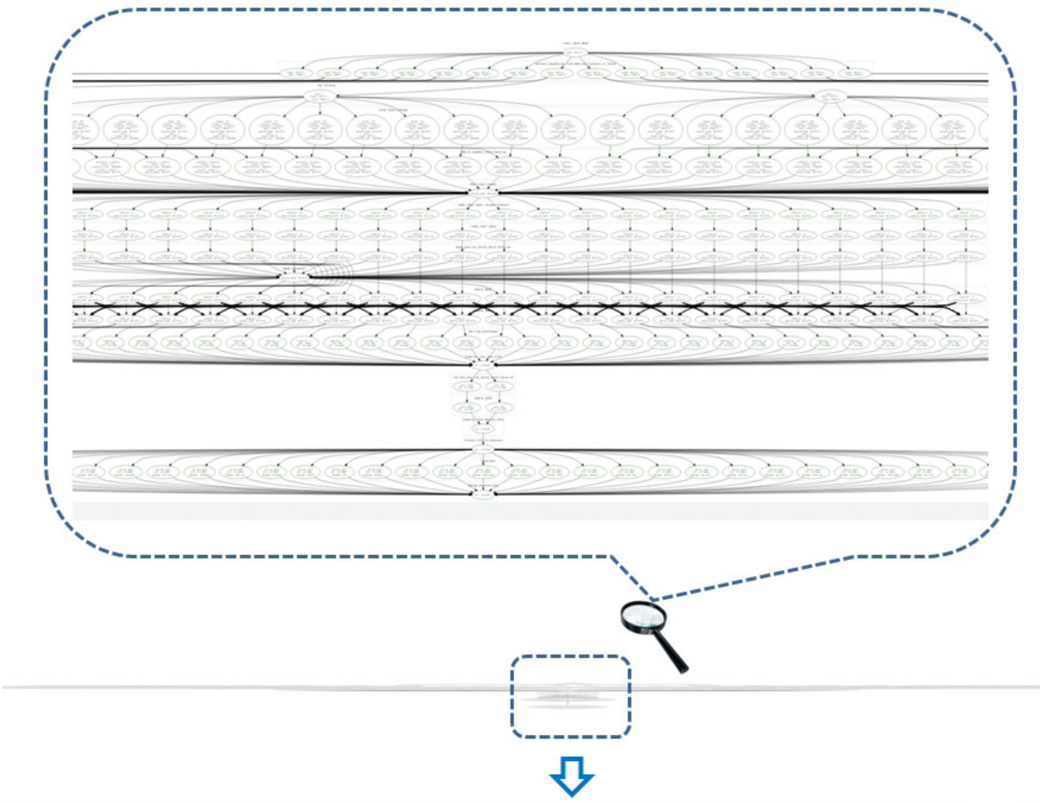
Figure 4.7: **Aws solution.**

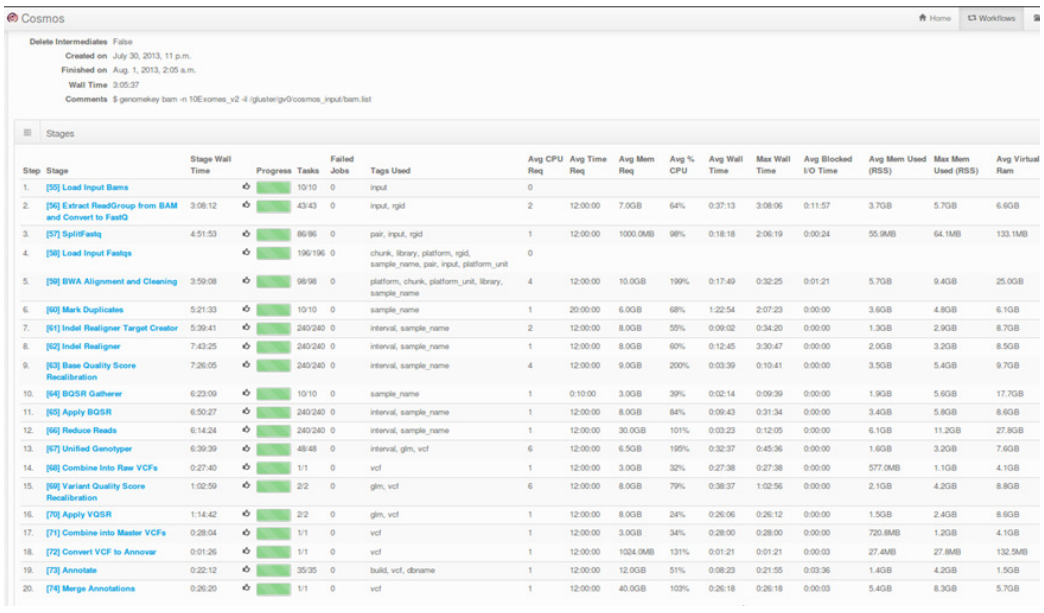
Cluster has been used to build the cluster that was used to run the whole NGS project: Fig. 4.8 shows the overall solution implemented on AWS. From a local computer StarCluster launches a cluster on AWS. Being the majority of workflow's software highly computational demanding, cc2.8xlarge instances were chosen in order to define the cluster. Such instance type is optimized for computing purposes and has 60 GB of memory, 32 virtual CPUs and 3.3 TB of ephemeral disk. Between all the possible operating systems available for this type of instance, the Ubuntu 12.04LTS os was chosen. A good compromise between timing and costs was set in a 5 cc2.8xlarge node cluster and Fig. 4.8 refers to this configuration. StarCluster is then responsible to NFS-share the /home directory on the master node and install the Sun Grid Engine queueing system. However as stated before, COSMOS requires an "input/output" folder that should be shared as well. Because such a folder requires high I/O performances, GlusterFS was preferred above other NAS because it is a file system designed to provide network storage that can be made redundant, fault-tolerant and scalable. Moreover it's particularly well suited to applications that require high-performance access to large files. So, once the cluster is up and running all the master node ephemeral drives are used as the bricks necessary to create the Gluster volume that will be the COSMOS input/output storage.

COSMOS, which in Fig. 4.7 is the workflow management system, is installed on the master node and the pipeline defined in the previous chapter can be finally executed. Fig. 4.8 shows how the pipeline steps are highly parallelized. When possible, the workflow split the input files both on read-group and chromosome. The upper part of Fig. 4.8 is indeed a zoom on a section of the overall dag which is then summirezed by the COSMOS web service in lower part of Fig. 4.8.

Finally, it is worth noting that this configuration can easily increase the number of computing instances if needed. StarCluster supports indeed a resizing procedures that allow for manually shrinking and expanding the size of the cluster. If the initial cluster sizing is incorrect, users can add or remove nodes to scale it accordingly to their needs. However despite the adding procedure will include new nodes in the cluster queuing system and the /home directory on the master node will be NFS-shared across them, user must mount the Gluster file system on new cluster nodes in order to share COSMOS input/output storage.

Although it is a manual procedure, the scaling feature makes the cloud configuration even more interesting. The suggested AWS solution allows indeed to set and use an HPC system according to real needs also because of the possibility to easily resize it. In conclusion, for this particular NGS project, the "pay as you go" scheme resulted in an overall cost reduction. An investment on a high performance computing solution like the one implemented on the cloud would have been much more expensive because except for some peaks of computation, it would have been underused.





Delete Intermediates False
Created on July 30, 2013, 11 p.m.
Finished on Aug. 1, 2013, 2:05 a.m.
Wall Time 3:05:37
Comments $ genomekey bam -n 10Exomes_v2 -il /gluster/gv0/cosmos_input/bam.list

Step	Stage	Stage Wall Time	Progress	Tasks	Failed Jobs	Tags Used	Avg CPU Req	Avg Time Req	Avg Mem Req	Avg % CPU	Avg Wall Time	Max Wall Time	Avg Blocked I/O Time	Avg Mem Used (RSS)	Max Mem Used (RSS)	Avg Virtual Ram
1.	[55] Load Input Bams		🟩	10/10	0	input	0									
2.	[56] Extract ReadGroup from BAM and Convert to FastQ	3:08:12	🟩	43/43	0	input, rgid	2	12:00:00	7.0GB	64%	0:37:13	3:08:06	0:11:57	3.7GB	5.7GB	6.6GB
3.	[57] SplitFastq	4:51:53	🟩	86/86	0	pair, input, rgid	1	12:00:00	1000.0MB	98%	0:18:18	2:06:19	0:00:24	55.9MB	64.1MB	133.1MB
4.	[58] Load Input Fastqs		🟩	196/196	0	chunk, library, platform, rgid, sample_name, pair, input, platform_unit	0									
5.	[59] BWA Alignment and Cleaning	3:59:08	🟩	98/98	0	platform, chunk, platform_unit, library, sample_name	4	12:00:00	10.0GB	199%	0:17:49	0:32:25	0:01:21	5.7GB	9.4GB	25.0GB
6.	[60] Mark Duplicates	5:21:33	🟩	10/10	0	sample_name	1	20:00:00	6.0GB	68%	1:22:54	2:07:23	0:00:00	3.6GB	4.8GB	6.1GB
7.	[61] Indel Realigner Target Creator	5:39:41	🟩	240/240	0	interval, sample_name	2	12:00:00	8.0GB	55%	0:09:02	0:34:20	0:00:00	1.3GB	2.9GB	8.7GB
8.	[62] Indel Realigner	7:43:25	🟩	240/240	0	interval, sample_name	1	12:00:00	8.0GB	60%	0:12:45	3:30:47	0:00:00	2.0GB	3.2GB	8.5GB
9.	[63] Base Quality Score Recalibration	7:26:05	🟩	240/240	0	interval, sample_name	4	12:00:00	9.0GB	200%	0:03:39	0:10:41	0:00:00	3.5GB	5.4GB	9.7GB
10.	[64] BQSR Gatherer	6:23:09	🟩	10/10	0	sample_name	1	0:10:00	3.0GB	39%	0:02:14	0:09:39	0:00:00	1.9GB	5.6GB	17.7GB
11.	[65] Apply BQSR	6:50:27	🟩	240/240	0	interval, sample_name	1	12:00:00	8.0GB	84%	0:09:43	0:31:34	0:00:00	3.4GB	5.8GB	8.6GB
12.	[66] Reduce Reads	6:14:24	🟩	240/240	0	interval, sample_name	1	12:00:00	30.0GB	101%	0:03:23	0:12:05	0:00:00	6.1GB	11.2GB	27.8GB
13.	[67] Unified Genotyper	6:39:39	🟩	48/48	0	interval, glm, vcf	6	12:00:00	6.5GB	195%	0:32:37	0:45:36	0:00:00	1.6GB	3.2GB	7.6GB
14.	[68] Combine Into Raw VCFs	0:27:40	🟩	1/1	0	vcf	1	12:00:00	3.0GB	32%	0:27:38	0:27:38	0:00:00	577.0MB	1.1GB	4.1GB
15.	[69] Variant Quality Score Recalibration	1:02:59	🟩	2/2	0	glm, vcf	6	12:00:00	8.0GB	79%	0:38:37	1:02:56	0:00:00	2.1GB	4.2GB	8.8GB
16.	[70] Apply VQSR	1:14:42	🟩	2/2	0	glm, vcf	1	12:00:00	8.0GB	24%	0:26:06	0:26:12	0:00:00	1.5GB	2.4GB	8.6GB
17.	[71] Combine into Master VCFs	0:28:04	🟩	1/1	0	vcf	1	12:00:00	3.0GB	34%	0:28:00	0:28:00	0:00:00	720.8MB	1.2GB	4.1GB
18.	[72] Convert VCF to Annovar	0:01:26	🟩	1/1	0	vcf	1	12:00:00	1024.0MB	131%	0:01:21	0:01:21	0:00:03	27.4MB	27.8MB	132.5MB
19.	[73] Annotate	0:22:12	🟩	35/35	0	build, vcf, dbname	1	12:00:00	12.0GB	51%	0:08:23	0:21:55	0:03:36	1.4GB	4.2GB	1.5GB
20.	[74] Merge Annotations	0:26:20	🟩	1/1	0	vcf	1	12:00:00	40.0GB	103%	0:26:18	0:26:18	0:00:03	5.4GB	8.3GB	5.7GB

Figure 4.8: **Workflow's parallelization.**

Chapter 5

GenomeKey: Scalable and cost-effective NGS genotyping in the cloud[1]

While next-generation sequencing (NGS) costs have plummeted in recent years, cost and computation time are still substantial barriers preventing the use of NGS in a clinical setting. For this technology to be adopted by clinicians, the turnaround time of genomic sequencing data analysis must be within hours and the cost of rendering to clinically actionable information must be conducted in the $10's of dollars. To that end, this chapter will show how COSMOS was used to develop GenomeKey, an NGS pipeline. COSMOS enables the implementation of complex NGS pipelines by taking advantage of parallelization and the resources of a high-performance compute cluster, either locally or in the cloud. Here following it is shown that the combination of COSMOS and GenomeKey on Amazon Web Services (AWS) results in timely and cost-effective processing of both known public benchmarking NGS datasets as well as large-scale heterogenous clinical NGS datasets in clinical timeframes ($<$ 1 d). The defined systematic benchmarking also reveals some useful guidelines for pipeline management, such as optimum batching of samples and efficient cluster configurations.

5.1 Introduction

Next-generation sequencing (NGS) costs have plummeted in recent years, rapidly outpacing the traditional benchmark for the decreasing cost of technology known as Moore's law. Routine clinical whole genome sequencing and analysis are now falling within the range of costs of medical testing. Mod-

[1]The content of this chapter is in proceeding to be submitted as *GenomeKey: Scalable and cost-effective NGS genotyping in the cloud, Y. Souilmi, A. K. Lancaster, J. B. Hawkins, JY. Jung, R. Powles, E. Rizzo, P. J. Tonellato, D, P. Wall*

ern sequencing platforms are now capable of sequencing approximately 5,000 megabases a day [55] at the cost of pennies per megabase. Sequencing centers such as the Broad Institute and the Beijing Genomics Institute are now capable of generating petabytes of sequencing data on a routine basis [56]. As a result of the increased efficiency and diminished cost of sequence generation, the adoption of NGS is increasing and generating large scale amount of data making the data analysis and interpretation of the sequenced genome quickly and within a reimbursable cost point the new technological barrier and price-limiting step of clinical genome usage [57, 58]. Furthermore, recent estimates suggest that the turnaround time for the clinical use of a sequenced genome should be on a scale of hours and cost in the $10's of dollars [58].

With the recent US Food and Drug Administration clearance of Sanger sequencing as a clinical diagnosis tool in January 2013, and its subsequent authorization of Illumina deep sequencing technology for similar purposes [59], cost efficient whole genome sequencing tools in the clinical space have become an absolute necessity. As a result of these recent regulatory developments, delivering a software solution that allows a whole genome to be sequenced within a day and rendered within hours at a cost of under $100 will be a breakthrough in clinical bioinformatics of great value to biomedical science, translational medicine, and healthcare as a whole. In a major step towards achieving this goal, we have developed a scalable, parallelizable workflow manager, COSMOS, capable of running on the cloud (e.g., Amazon Web Services - AWS) [60]. COSMOS is able to lower the cost of analyzing genomic data two ways: 1) implementing a highly parallelizable workflow that can be run quickly and efficiently on a large compute cluster, and 2) taking advantage of transient cloud-based instances that can be invoked and dismissed on-the-fly, enabling considerable reductions on per hour cost.

Since typical NGS pipeline involves the successive implementation of many computational tools into complex workflows, many analysis platforms [61, 62, 63] tend to focus on generating user-friendly, reproducible pipelines for biomedical researchers that may have very little computational background. More recently, efforts have also been made to leverage the power and speed of highly parallel computing in NGS workflows [64]. Unfortunately, few of these software packages have been demonstrated to have the speed and throughput efficiency necessary in larger clinical genomics projects, which typically involve the cost-effective processing of hundreds of genomes or exomes in a single study.

To leverage the power of cloud computing for NGS [65], we have developed a variant-calling workflow, GenomeKey, running on top of COSMOS. Our pipeline performs a thorough sequence analysis, including alignment, quality score recalibration, variant calling and optionally annotation, and can be run on both the cloud and local high-performance computing clusters. Our focus in choosing workflow tools is to take advantage of the runtime performance, cost and scalability of the GATK best practices standards established by the Broad Institute [66]. By using the COSMOS framework to process and ana-

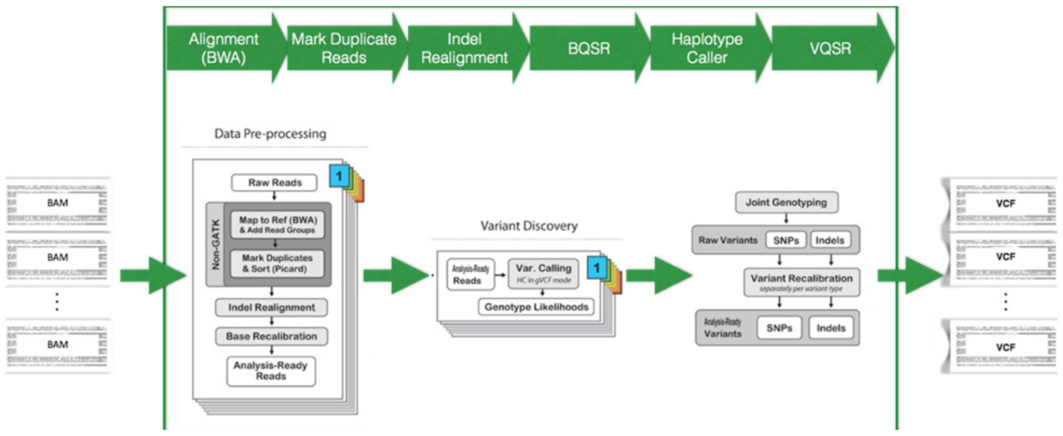


Figure 5.1: **GenomeKey workflow** GenomeKey workflow implements the GATK 3 best practices for genomic variant calling

lyze successively larger sets of genomic data, we have aggressively optimized the software for speed and cost performance. Here we perform a comprehensive benchmarking study of GenomeKey in the cloud using a heterogeneous combination of public and clinical NGS data to explore speed-cost tradeoffs. We show that we can reduce cost of the analysis of whole exome and whole genome data over 10-fold, from ≈ \$1000 [58] to under \$100.

5.2 Methods

The study design consists of four main components:

1. the GenomeKey workflow to identify genomic variants built upon the COSMOS engine Fig. 5.1;
2. the deployment of this workflow upon the Amazon Web Services (AWS) Elastic Compute Cloud (EC2) platform Fig. 5.2;
3. the collection of short-read sequencing data for both exomes and genomes;
4. the validation of this pipeline variant calls against previous variant calls.

Below, we describe each of them in turn.

5.2.1 Workflow

We created GenomeKey, a Python-based NGS-analysis pipeline that implements the Genome Analysis Toolkit's [67] version 3 of the best practice protocol [36, 66] including alignment, base quality scoring recalibration and joint variant calling to take advantage of the increased statistical power [40] and calibration. GenomeKey is built upon COSMOS, a Python library for

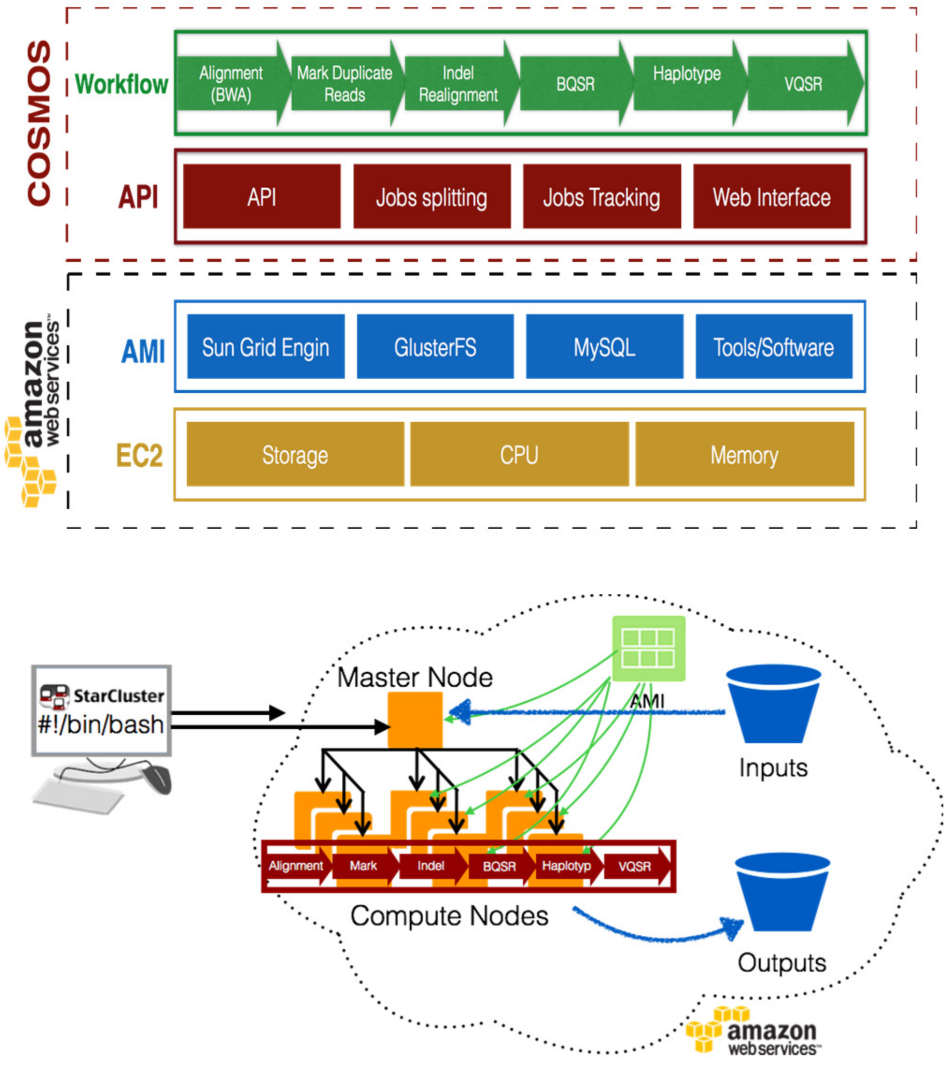


Figure 5.2: **Overall benchmarking study design** Deployment of the workflow upon the Amazon Web Services Elastic Compute Cloud (EC2) infrastructure using the COSMOS workflow management engine

workflow management that allows formal description of pipelines and partitioning of jobs [60]. GenomeKey uses COSMOS's workflow language to define the overall pipeline, and COSMOS, in turn, manages the overall computational workflow. After loading in genomic data, GenomeKey instructs COSMOS to break up each stage of the workflow into multiple tasks using COSMOS' tagging system, which are managed by COSMOS to be efficiently run in parallel. Note that although the focus in this article is on deployment on Amazon Web Services cloud, the COSMOS/GenomeKey combination runs equally well on local traditional HPC computing clusters.

The pipeline for variant detection proceeds in seven main stages, together with an optional eighth:

1. Re-alignment and mapping (BAM to BWA). To parallelize the realignment: previously aligned BAMs proceeds by first, splitting each BAM file into separate chromosomes, and optionally by read group (RG). This enables fuller use of cluster resources.

2. Indel realignment (IndelRealign)

3. Marking of read duplicates (MarkDuplicates)

4. Base quality score recalibration (BQSR)

5. Generate genomic VCFs (HaplotypeCaller): variants are called per sample and chromosome, generating "gVCF" files

6. Genotype samples (GenotypeGVCFs): this stage uses the previously generated gVCFs to call variants jointly across all samples in the original input

7. Variant quality score recalibration (VQSR)

8. Annotation (ANNOVAR): optional annotation stage with extended ANNOVAR databases

5.2.2 Deployment on AWS platform

Cluster configurations

All the runs were performed on a 21 cc2.8xlarge (60 GB of memory, 32 virtual CPUs and 3.3 TB of ephemeral disk) node cluster to be able to compare performance and scalability on the same cluster size. All clusters consisted of a master node and 20 worker nodes. The master node has installations of COSMOS and GenomeKey, all tools required by the workflow and all input files. Worker nodes can either be "spot-instances", subject to being eliminated due to price increases, or fixed price "on-demand" nodes (that is not subject to price increases during the runs). We placed initial bids at \$0.5/hr ($\approx$ \$0.27/hr) in the AWS us-east region. The primary master nodes are on-demand so that all data is preserved in case a node is eliminated by the spot bid. For initial runs, we used one master node, and 20 workers, for runs involving larger datasets (e.g., 20+ exomes) we converted some of the spot worker nodes to also be on-demand worker nodes, allowing us to incorporate their hard drive space into a common pool using GlusterFS, as described above (see Table 5.1 for the different configurations).

	Total size * (TB)	**Number of GlusterFS bricks**
Configuration 1	3.3	1
Configuration 2	6.5	2
Configuration 3	13	4

Table 5.1: **Different GlusterFS configurations used to increase the shared drive size**

Cluster management system

The cluster launching and management was performed using the open source software StarCluster (specifically tagged with v 0.95.4 on the github repository). Instances were launched with starcluster using the Ubuntu 12.04 LTS based AMI (Amazon Machine Image) **ami-5bd1c832** that greatly reduces the setup time because by default includes all the required software, COSMOS, GenomeKey as well as the tools needed by GenomeKey.

Job management and shared file system

The jobs created by COSMOS are submitted and managed across nodes in the cluster using version 6.2u5-4 of Sun Grid Engine. The compute nodes of the cluster share a common scratch space provided by one or more node(s). This shared filesystem is created and managed by version 3.4 of GlusterFS, a cluster shared file system.

Computing AWS cost

We used the AWS cli tools (available from http://aws.amazon.com/cli/) to compute the cost of "spot" instances using the start and stop timestamps of the workflow recorded by GenomeKey and our study driver automation script, specifically using the following commands:

```
aws ec2 describe-spot-price-history --start-time [start timestamp]
--end-time [end timestamp] --instance-types cc2.8xlarge
--availability-zone
```

The command returns the history of the spot price during the specified period and we use the mean of the returned values to compute the spot instances cost. In addition, the on-demand ($2 per hour per instance) price is fixed by AWS (listed here https://aws.amazon.com/ec2/pricing/). The total cost of the cluster for a given run was then computed by adding the costs of on-demand and spot nodes together.

5.2.3 Benchmarking data

To enable us to benchmark the effect of overall throughput and cost on an increasing number of samples on a fixed cluster, we created a range of groups (or "runs") of exomes and genomes of size, n, to be run simultaneously. For exomes, n ranged from n = 1, 3, 5, 10 ,25, 50. For genomes n = 1, 5, 10, 25.

Exomes

For control data, we used high coverage (≈150x) whole exome set of the CEU trio (NA12878, NA12891 and NA12892) from the Coriell CEPH/UTAH 1463 pedigree, sequenced at the Broad Institute. This set is recommended for reviewing or benchmarking purposes by the GATK team [http://gatkforums.broadinstitute.org/discussion/1292/which-datasets-should-i-use-for-reviewing-or-benchmarking-purposes.], because of the high coverage and availability of other genotype data sets from the same trio, including WGS data we will use in genome test runs. We included this trio data in each exome run, in order to compare and cross-check the quality of output variants (of the same input data) over different runs.

To round out our full exome runs dataset, we included biomedically disease relevant exomes originally generated by Christopher Walsh's group [68]. These data were chosen because: (i) the data are curated by National Database for Autism Research (NDAR); ii) mean read depth for proband data (≈158x) matches up with that of CEU trio; iii) extended family data are available including affected siblings; iv) VCF files are provided for a subset of probands; and v) phenotype information is available via NDAR or AGRE. The BAM files are renamed to group by families. This data set was also sequenced at Broad Institute, so we used the exome target regions (Agilent Sure-Select Human All Exon v2.0, 44Mb baited target) provided in the GATK bundle, with 100 bp padded at both ends, to extract targets for both control/case exome data.

Genomes

BGI Genomes: We selected 31 unique autism-associated genomes with coverage ranging from 31.5x to 42x with a mean coverage of 37x per run. These were originally sequenced by BGI on the Illumina platform. The genomes were selected to have trios in each batch to take advantage of the joint variants calling feature of GenomeKey.

Platinum Genomes: We selected a single high coverage genome (50x), sample NA12878 (corresponding to the same exome individual). The Platinum genomes also have "gold standard" variant calls in VCF format with variants called using different software and technology. Those VCFs enable quality control, as well as reference timings from Blue Collar Bioinformatics group bcbio-nextgen. (http://www.illumina.com/platinumgenomes/).

5.2.4 Variant validation

To validate GenomeKey, we downloaded the previously generated BAM files available for the trio of exomes from Phase I of the 1000 Genomes Project. For the 1000 Genomes trio, we were then able to compare our BAMs with these downloaded BAMs. The quality control consisted of the following steps:

- Compared the percentage of unmapped reads between our original mapped BAM and our re-mapped BAM. Although the number of mapped reads may be different to the Phase I output because we are using recent versions of BWA and the reference genome, we expect the number to be very close.
- Compared the distribution of phred base quality scores for each two BAM files using FastQC.

We then compared the results of our pipeline in terms of variant calls against available benchmark data [36]. The analysis was performed on the NA12878 sample which corresponding whole exome BAM file was originally aligned with MAQ on hg18. In order to compare variant calls we ran the same method (GATK v3 HaplotypeCaller) over the benchmark's BAM and our re-mapped BAM. The procedure was set with identical parameters except for the reference genome. An additional analysis was performed on sequencing data publicly available at www.platinumgenomes.org. This project provides whole genome sequence and variant call data for 17 members of the Coriell CEPH/UTAH, including NA12878 sample, in order to create a "platinum" standard comprehensive set of variant calls. We extracted raw sequencing data from NA12878's BAM files and re-mapped them against hg19 with GenomeKey. Once we had a genome vcf we evaluated the Genotype concordance against a benchmark whole-genome genotype calls dataset from Genome in a Bottle Consortium [69].

5.3 Results

Using a 21 node AWS cluster (each using 20 "worker" cc2.8xlarge nodes), we used the GenomeKey and COSMOS framework to process successively larger sets of genomic data. Using pre-aligned exomes and genomes to split and parallelize the input by chromosome, we measured both overall wall time for processing of all samples, as well as the overall cost for the entire workflow for each group of datasets.

Robust handling of heterogeneous datasets

We deliberately chose a mixture of datasets, including previously sequenced public datasets from the 1000 Genomes Project, as well biomedically-relevant autism disease datasets, with varying levels of coverage. This enabled assessment of the pipeline when confronted with relatively heterogenous data, as may be more common biomedical and clinical scenarios. We were able to smoothly process many different samples despite these differences: across the entire set we only observed two failed tasks (in the 25 genomes run) that led to a workflow stop. These failures resulted from the loss of a single AWS worker node and, a resume of the workflow enabled the entire workflow to complete successfully without the need to reprocess the successful tasks.

Accuracy

To evaluate the accuracy of GenomeKey in terms of alignment, we downloaded a trio of exomes from Phase I of the 1000 Genomes Project (The 1000 Genomes Project Consortium, 2010), remapped them all with GenomeKey and evaluated the difference of mapped reads. Although the number of mapped reads could have been different because of recent versions of BWA and the reference genome we observed only ≈0.25% increase in total mapped reads over the original BAMs. FastQC was then used to compare the phred base quality scores over the reads. Although the original BAMs were re-calibrated with GATK as well, the overall quality over the total sequence length is higher for the BAM files generated by GenomeKey.

To evaluate the accuracy of GenomeKey in terms of variant calls we compared the results of our pipeline against available benchmark data [36]. The analysis was performed on the NA12878 sample and compared a number of metrics of the called variants. Mapping differences between the aligners used, different reference builds and improvements in GATK methods account for missing calls on the benchmark BAM file. Overall, despite an increase on the total number of high quality SNPs, the consistency of the GenomeKey variant calls as compared with the previous SNP calls was high, in particular the genotype concordance as reported by the GATK Genotype Concordance module was 0.97.

We also evaluated the genotype concordance against a benchmark whole-genome genotype calls dataset from Genome in a Bottle Consortium (www.genomeinabottle.org). This reference dataset minimizes bias toward any method by integrating and arbitrating between 14 data sets from five sequencing technologies, seven read mappers and three variant callers [69]. GenomeKey reached a complete overall genotype concordance and a 0.997 sensitivity, confirming an high accuracy against benchmark dataset.

Fast processing of single exomes and genomes

GenomeKey processes single high-coverage genomic data extremely rapidly. The workflow processed a high-coverage (≈150x) exome from alignment to variant calling in 2 hr 16 min for a total cost of ≈\$23 on AWS (download from and backup to Amazon's S3 storage adds an additional cost of \$5). Four replicate runs for the 1, 3, 5 and 10 exome cases, established that the variability in these cost and run-times was small (the standard error was 2 min 21 s for the 1 exome run) (Fig. S4). A single high-coverage (≈42x) genome took 13 hr 52 mins for a total cost of ≈\$109 (additional cost \$16). The cost compares favorably with other tools. The use of spot instances, in particular, saves in overall AWS cost as compared with on-demand nodes (on-demand cost: \$126 for 1 exome run and \$588 for the 1 genome run) .

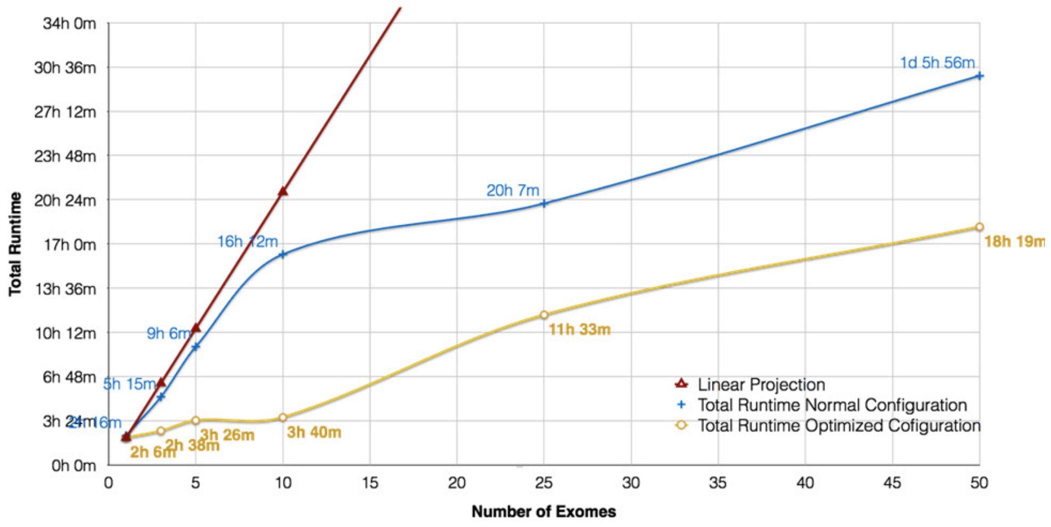
Parallelization reduces overall cost of multiple samples

This sample cost is significantly reduced as more samples are "batched" together, taking advantage of the parallelization, especially of the early alignment stages for both exomes (Fig. 2) and genomes (Fig. 3). As expected, cluster efficiency also improves with increasing sample size, as runs with more samples result in the cluster operating closer to its maximum capacity throughout the run (Fig. 4). However, increasing the number of samples, has an upper limit, as loading many samples can overtax the nodes for a given cluster size, leading to poorer overall performance (see 50 exomes). Batching multiple samples in this way does also result in a longer time-to-completion for any given sample, as all samples will be completed and ready at the total wall time of the entire workflow. However, a modest ≈12 hr processing time for 25 exomes (compared with ≈2 hrs for a single exome) has dramatic savings: reduces the cost per exome four-fold (from \$25.37 to \$5.81) (Fig. 2).

Grouping storage across multiple nodes speeds runtime

We explored the the role of grouping the storage required for output of each GenomeKey stage, across multiple nodes. In principle, this would speed runtimes because the network bandwidth costs are spread across multiple nodes, enabling parallel write operations and reduce latency. Using the GlusterFS file-system (see Methods) to pool hard-drive storage across multiple nodes dramatically improved overall runtime.

For example on the same input data, doubling the number of nodes participating in the GlusterFS from 2 to 4, almost halved the runtime in the 25 exome case (from ≈20 h to ≈11.5 h) (Fig. 2A). We also measured the gain per stage and found that the data transfer for the alignment stage (BWA) was 2.5 times faster using the 4 node GlusterFS configuration compared with the 2 node configuration. The tradeoff is that the nodes participating in the GlusterFS configuration, need to be persistent on-demand AWS instances and



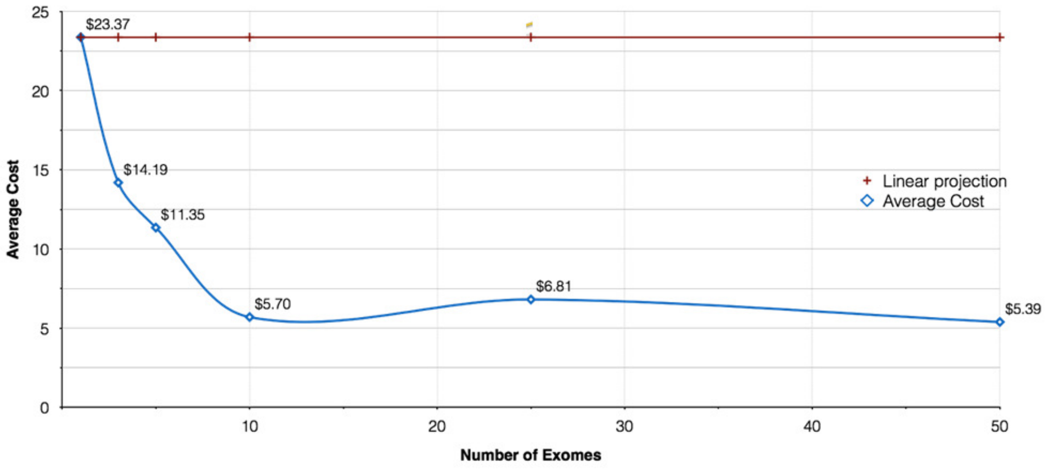


Figure 5.3: **GenomeKey workflow scales efficiently with increasing number of exomes compared on different gluster configurations** The blue curve represents the 1, 3, 5 and 10 exomes runs performed on a cluster with one GlusterFS brick; the yellow curve represents the scalability on a cluster with four GlusterFS bricks. Wall time (top) cost (bottom) as a function of exome and size for as compared to a linear extrapolation of a single exome

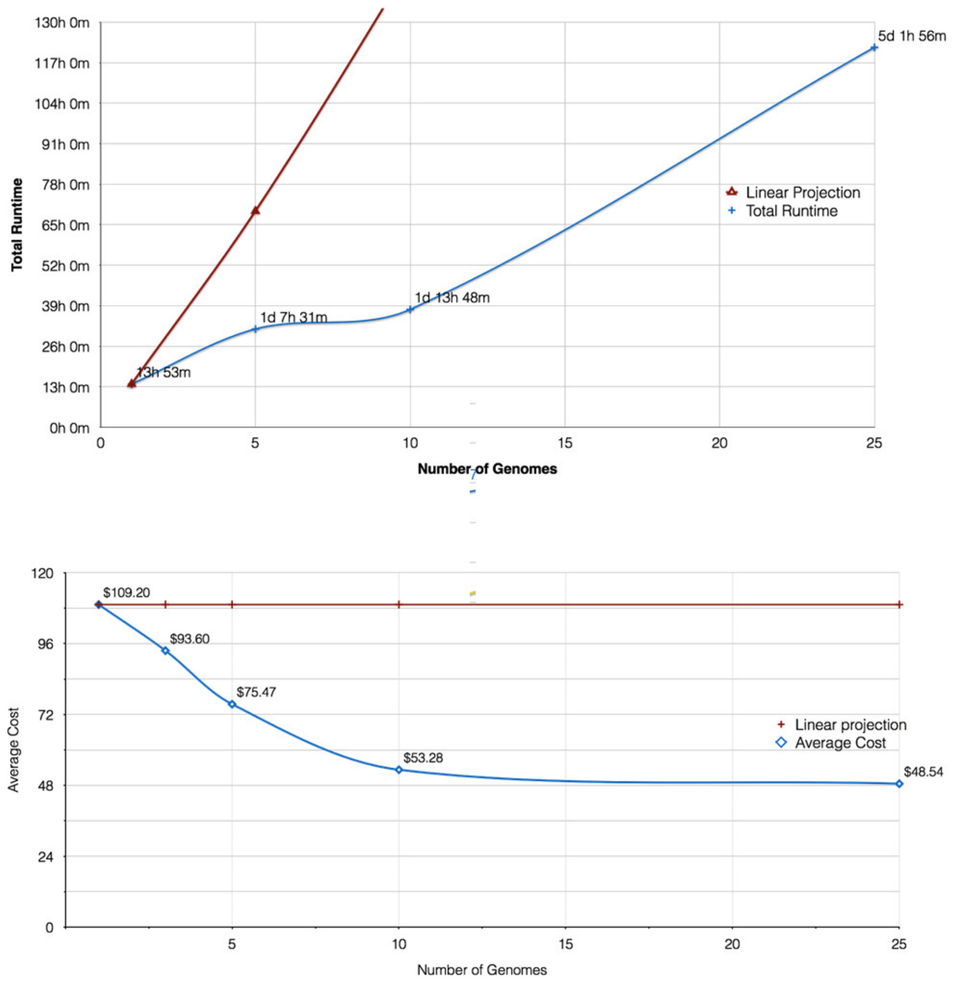


Figure 5.4: **GenomeKey workflow scales efficiently with increasing number of genomes** Wall time (top) cost (bottom)

thus are more expensive than transient spot instances, however this was more than offset by the increase in processing speed (Fig. S4).

Exploring upper limits to parallelization

We examined the effect of increasing the amount of parallelization even further by using GenomeKey to split input BAM files for the alignment stage ("BAM to BWA") on both read-group (corresponding to an individual "lane" for Illumina data) as well as chromosome. We found only a modest reduction in runtime using this approach, as compared to chromosome-only splits for a single exome, and for larger run sizes, the sheer number of tasks could overwhelm the queuing system (25 exomes and higher). Obviously, this result is dependent on the nature of the input data (in the case of the autism exomes, there were upwards of 10 read-groups per sample), but it clearly illustrates that there is an upper limit to parallelization of the alignment stage. We thus provide a command-line option to the GenomeKey workflow to split by chromosome only instead of the default chromosome and read-group.

5.4 Discussion

We have demonstrated that the combination of GenomeKey running on COSMOS presents a scalable, efficient solution to the increasing demand for effective genomic variant calling. Our pipeline allows for efficient parallelization of the early stages of the pipeline such as alignment. Where necessary later stages (i.e. variant calling) are run in serial to merge by chromosome. Our benchmarking has verified the overall robustness and scalability of GenomeKey in processing large data sets, established rapid runtimes and low per-exome and genome costs using cloud-based computing instances and provided some qualitative comparisons with existing methods. Our systematic approach to the study design also allows us provide some guidance for researchers using GenomeKey (or potentially other GATK-based workflows) by quantifying workflow configuration choices in two key areas: speed-cost tradeoffs in the number of samples processed and in the choice of cluster filesystem configuration. We discuss each of them below.

5.4.1 Run-time and cost comparisons with other cloud-based variant workflows

While streamlined, user-friendly variant-calling workflows has already seen a great deal of recent progress, GenomeKey and COSMOS provide an important service in efficient, low-cost variant calling. Community standards for performance benchmarking of full genomics workflows are relatively unestablished, so very few software packages include direct comparisons of speed or efficiency between competitors, opting instead for feature charts as seen in

SIMPLEX [70], or general qualitative comparisons, as seen in Mercury [71]. While the publishers of each piece of software do tend to report runtime statistics of their own system, these come from a variety of different data sets, and often differ in hardware specifications (Table 1).

In comparison, GenomeKey offers a fast, cost-efficient, accurate solution for alignment, cleaning, and variant-calling of both genomes and exomes compared to other software. STORMseq [62], for example, using BWA for alignment and GATK lite for quality control and variant calling, reports the processing of an exome in 10 hours and a genome in 176 hours, as compared to GenomeKey which can finish an exome run and a genome run in 2 hours and 14 hours, respectively. Most other software packages have been found to perform at similar levels to STORMseq, with exomes tending to take around 12 - 24 hours to be fully aligned, cleaned, and analyzed for variants (see Table 2 for more details). However, these are not direct comparisons - as mentioned previously, different data sets are used for benchmarking, and different computational resources are given for the published benchmarking of each software package. Additionally, many of these systems are designed for ease of use and small scale experiments [63]. The only large-scale benchmarking effort published so far is that of the cloud-based, genomics workflow Rainbow {zhao2013rainbow. Rainbow has been reported to have successfully processed 44 genomes in two weeks for \$5,800, amounting to ≈\$120 per genome - though this is still a higher price point and slower processing time than GenomeKey's \$50 genome capabilities. Although GenomeKey has not been used on the same size of dataset, it has been demonstrated to run efficiently at a similar multi-genome scale.

5.4.2 Building a road-map for NGS analyses

Beyond documenting the speed and robustness of GenomeKey, our systematic benchmarking has allowed us to build and investigate an initial "complexity roadmap" of NGS variant pipelines in the cloud in general. In particular, our results can provide guidance for workers building NGS pipelines in two keys areas: (1) choosing the optimum batch size of samples to run in parallel and (2) choosing cluster and shared storage configurations that speed runtimes.

Choosing batch size and cluster configuration

In a clinical context, an investigator or a hospital may have a fixed price point for genomic analysis of samples. Using our metrics of cost per exome, combined with overall wall time, we can provide some batch size estimates to achieve a time-to-sample analysis completion of under a day. In our particular case, we found a good balance to be between 10 to 15 exomes per batch, but sequencing centers will have different kinds of data and estimates will likely vary widely, so this is at best only a rough heuristic.

Choosing shared storage configurations

Even though our pipeline does as much processing as it can on each of the local nodes to avoid network latency, the output for each stage needs to be copied back to the shared storage to allow the dependent tasks to proceed to the next stage, and our benchmarking reveals that this can take a non-trivial amount of time. Our benchmarking clearly points towards the effectiveness of selecting multiple nodes to act as part of shared storage configuration. This becomes particularly important when there are many jobs performing intensive reads and writes, but needs to be weighed against the extra cost of having fixed nodes that are not low-cost transient "spot" instances.

Chapter 6

A kinetic model-based algorithm to classify NGS short reads by their allele origin[1]

Genotyping Next Generation Sequencing (NGS) data of a diploid genome aims to assign the zygosity of identified variants through comparison with a reference genome. Current methods typically employ probabilistic models that rely on the pileup of bases at eachlocus and on a-priori knowledge. Hereunder will be discussed a new algorithm, called Kimimila (KInetic Modeling based on InforMation theory to Infer Labels of Alleles), which is able to assign reads to alleles by using a distance geometry approach and to infer the variant genotypes accurately, without any kind of assumption.

The performance of the model has been assessed on simulated and real data of the 1000 Genomes Project and the results have been compared with several commonly used genotyping methods, i.e., GATK, Samtools, VarScan, FreeBayes and Atlas2. Despite the algorithm does not make use of a priori knowledge, the percentage of correctly genotyped variants is comparable to these algorithms. Furthermore, the method allows the user to split the reads pool depending on the inferred allele origin.

6.1 Introduction

Since the first release of the human genome consensus sequence, the impact of next generation sequencing (NGS) technologies is getting every day stronger and it is rewriting the rules of genomic research.

Current high-throughput DNA sequencing technologies can generate billions of short sequences (reads) whose length range from 50 to 400 bases. Once

[1]The contents of this chapter is published as *Marinoni, Andrea, Ettore Rizzo, Ivan Limongelli, Paolo Gamba, and Riccardo Bellazzi. "A kinetic model-based algorithm to classify NGS short reads by their allele origin." Journal of biomedical informatics (2014).*

available, these reads are mapped onto their corresponding reference genome to extract useful genomic information as known and/or novel variants. However, while the human genome sequence assembly for a chromosome is an arbitrary mix of the two haploid chromosomes, the DNA fragment library is usually derived from a diploid genome. This requires determining the zygosity of each variant detected by the variant calling process: genotype calling is aimed at classifying these variants as heterozygous or homozygous. Several genotyping methods have been proposed both for human and non-human genomes [72, 73]. Early methods rely on counting the number of times each allele of a single genomic locus(pileup) is observed. For example, VarScan [74] is based on an heuristic method that uses the number of aligned reads supporting each allele.

The most recent methods are often based on a probabilistic framework. Tools such as GATK UnifiedGenotyper (UG) [36] or Samtools [75] compute a genotype likelihood $p(D|G)$ for each genotype G, where D represents all the read data for an individual at a particular site. These genotype likelihoods incorporate errors that may have been introduced in base calling, alignment and assembly and are coupled with prior information, such as allele frequencies in a reference population (e.g. The 1000 Genome Project [76]) or databases of polymorphisms (e.g dbSNP [37]). The genotype with the highest posterior probability is chosen and it is associated with a measure of uncertainty.

Another recent method, Atlas2 [77] is based on a logistic regression model trained on validated whole-exome capture sequencing and uses features such as the ratio of the variant base to the reference sequence and the quality scores of the base calls in order to discriminate between true variants and sequencing, mapping, or alignment errors. These strategies rely on the single base pileup information derived from the aligned reads and are based on the assumption of variant locus and reads independence.

Haplotype-based methods are based on a different genotyping strategy. They make use of short haplotypes directly inferred from sequencing reads mapped to the reference. FreeBayes [78] is an algorithm that uses haplotype informative reads (i.e. reads spanning two or more variants) in combination with a Bayesian statistical method [79] to improve genotyping accuracy. Due to the promise of third generation sequencing technologies to increase reads length up to thousands of sequenced bases [80], we can expect haplotype-based methods to increase their accuracy and importance as well.

In this scenario was developed a new algorithm, called Kimimila (KInetic Modeling based on InforMation theory to Infer Labels of Alleles), that exploits a kinetic model approach relying on distance geometry. Kimimila, starting from a set of variant loci, infers the allele origin of each overlapping read by looking at reads similarity in terms of a non-euclidean distance and, as a consequence, can infer the zygosity of a variant through reads rather than through the single base pileup. Notably, the approach has been conceived to be fully data-driven. Therefore, it doesn't make use of a priori information and it does not rely on a statistical model. Furthermore, this algorithm provides a novel methodological contribution able to perform clustering with a non-euclidean metric. Kimimila was run on simulated data to evaluate the accuracy in classifying (or labeling) reads by the origin allele and on a sample of the 1000 Genomes Project in order to evaluate the genotyping and reads labeling accuracy as well, by using the relative genotype-chip data as validation set. Then, it was compared with several genotyping algorithms, namely GATK UG, VarScan, Samtools, Atlas2 and FreeBayes, and the percentage of correctly genotyped variants was comparable to the aforementioned tools. Furthermore, the method allowed the user to split the reads pool depending on the inferred allele origin. Such results can be further used for other interesting applications, such as haplotype reconstruction and to detect subclonal heterogeneity in tumor samples.

6.2 Methods

Labelling reads in NGS can be easily seen as a clustering problem. Standard clustering methods are typically based on some sort of Euclidean distance among the reads. Since mutation calling and base estimation are affected by noise and uncertain sequencing, the distance computation must take into account the reliability and the quality of the processed signals. Below will be described a new definition of the similarity metric able to cope with all these issues and to reach the desired clustering performance. It is worth noting that, in order to leverage the stress on the computational complexity, the bases that are considered are provided by the variant calling algorithm. The availability of a reliable variant calling algorithm is therefore assumed. The next subsection reports the notations and definitions that are used throughout the paper.

6.2.1 Definitions

Reliability can be identified as the likelihood of the given base over the whole dataset in a given locus. Therefore, let $M(i)$ be the set of the reads that map on the i-th base of the dataset and $|M(i)|$ the number of such reads. Let $\alpha(t_i) = \alpha_i$ be the nucleotide base that read α exhibits at the i-th position of the dataset, being $\alpha \in M(i)$. For sake of simplicity, let us assume that each nucleotide base is univocally mapped onto a natural integer in the set $\{1, 2, 3, 4\}$. Then, the aforesaid likelihood associated with the base at the i-th position for read α, $m_\alpha(t_i) = m_{\alpha_i}$ is defined as the fraction of reads in $M(i)$ that show the same base α_i in the same position, i.e., $m_{\alpha_i} = \frac{\sum_{\alpha' \in M(i)} \chi(\alpha'_i = \alpha_i)}{|M(i)|}$ where $\chi(\cdot)$ is the indicator function. In other terms, m_{α_i} measures the reliability (i.e., the probability of "noiseless" calls) of a given base for each read.

Furthermore, let $\varphi^b_\alpha(t_i) = \varphi^b_{\alpha_i}$ be the quality of the base b over read α in the i-th position as provided by the base caller. Then, $\varphi^b_{\alpha_i}$ can be written as a function of the probability of a base calling error for $b(P_{wrong}(\alpha_i, b))$ set by the base caller, i.e. $\varphi^b_{\alpha_i} = -10 log_{10}[P_{wrong}(\alpha_i, b)]$, that is the quality score value given by the base caller in Phred scale [27].

Then, let us set $v^b_\alpha(t_i) = v^b_{\alpha_i} = 1 - P_{wrong}(\alpha_i, b)$. The quantity $v^b_{\alpha_i}$ is a function of the quality of the given DNA base in the given read and it measures the likelihood (i.e., the probability of a correct call) of a given base for each read. By computing $v^b_{\alpha_i}$ for each b, it is possible to arrange a vector $\underline{v}_\alpha(t_i) = [v^b_{\alpha_i}]_{b=\{1,..,4\}}$.

Indeed, inferring allele labels over reads can be strongly affected by misalignment and error in base estimation algorithms, i.e., by the reliability and the quality of the information provided by the considered dataset. Hence, according to the aforementioned definitions, we can derive a properly designed distance geometry-based approach to improve the labeling method. The next subsection introduces the distance geometry-based approach.

6.2.2 A distance geometry approach

A reasonable metric to jointly consider the reliability and the quality of the given read over the given position is represented by the product of m_{α_i} and $v^b_{\alpha_i}$. Therefore, in order to compute the distance $d_{\alpha\beta}$ between two reads α and β in the set $M(i)$, we propose to use the following formula:

$$\sum_{t' \in \Delta_{\alpha\beta}} ||\underline{l_\alpha}(t') - \underline{l_\beta}(t')|| \tag{6.1}$$

where $\Delta_{\alpha\beta}$ is the overlap interval between the compared reads and $\underline{l_\tau}(t_i) = m_\tau(t_i)\underline{v_\tau}(t_i)$. It is important to note that the aforementioned definition of $d_{\alpha\beta}$ highlights the difference in variant calls among the reads, weighted according to their reliability degree.

A physics-based interpretation of such quantities is to assume them as the result of a kinetic process in the four dimensional space determined by the nucleotides bases. Specifically, each read can be seen as a time record of the movements of a particle over a given period. Each particle is characterized at each sampling time (t_i) by its mass (m_{α_i}) as well as by its velocity ($v^b_{\alpha_i}$) in each direction [81]. Furthermore, each particle at each time sample changes its mass and velocity as a result of an unknown collision. Thus, the global linear momentum of this hypothetical particle over a time interval Δ_η is:

$$\sum_{j \in \Delta_\eta} \underline{l_\eta}(t_j) = \sum_{j \in \Delta_\eta} \underline{m_\eta}(t_j)\underline{v_\eta}(t_j) \tag{6.2}$$

This quantity depends on the reliability and the quality of the information provided by the dataset that has been taken into account. Hence, the distance $d_{\alpha\beta}$ between two reads α and β is defined as the difference of their linear momenta over the “time interval” (i.e. the genome region) in which they overlap.

The basic idea of our method is to label the i-th base of the data set with a strategy that relies on the distances $d_{\alpha\beta}$ computed between all the reads of the set $M(i)$. The quantity $d_{\alpha\beta}$ can be actually considered as a distance

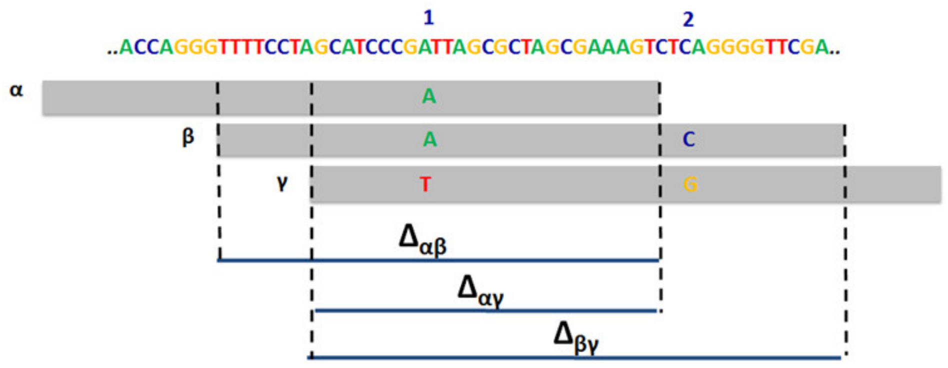


Figure 6.1: **Example of computation of distances: non Euclidean case**

since $d_{\alpha\beta} \geq 0$ and $d_{\alpha\beta} = d_{\beta\alpha}$. However, since the couples of reads typically have distinct supports (i.e., overlap on different intervals which can be affected by dissimilar kind of noise and mutations), the triangle inequality might not hold. Therefore, from our definition, the distance among the read set is non-Euclidean [82].

Given the provided definition of distance, Fig. 6.1 shows a little set of reads for which the triangle inequality does not hold. Reads are drawn as bricks. Mutations are expressed as colored letters in the brick. If the brick is grey, the given base in the read under analysis is homologous to the reference genome (which is reported in the string above the reads). Fig. 6.1 reports three reads α, β and γ mapping onto a genomic segment that shows two mutations (in positions 1 and 2). According to the definitions that have been previously introduced, is it possible to write $m_{\gamma_1} = 1/3$, $m_{\beta_1} = m_{\alpha_1} = 2/3$, $m_{\gamma_2} = 1/2$, $m_{\beta_2} = 1/2$. Furthermore, let assume that each base has been full quality sequenced, i.e., the velocity term is set to 1 for each base that is considered. Thus, the distance set is as follows:

$$d_{\beta\gamma} = \sqrt{\left(\tfrac{1}{3}\right)^2 + \left(\tfrac{2}{3}\right)^2} + \sqrt{\left(\tfrac{1}{2}\right)^2 + \left(\tfrac{1}{2}\right)^2}$$

$$d_{\alpha\gamma} = \sqrt{\left(\tfrac{1}{3}\right)^2 + \left(\tfrac{2}{3}\right)^2}$$

$$d_{\alpha\beta} = 0$$

Hence, it is easy to verify that the triangle inequality does not hold since $d_{\beta\gamma} \not< d_{\alpha\gamma} + d_{\alpha\beta}$. Therefore, from the definition, the distance among the read set in the NGS system is non-Euclidean.

Let us now consider the reads α and β that are the points at maximum distance (i.e., $d_{\alpha\beta} = d_{max}$), such as the ones in Fig. 6.2. In this case we can cluster the other reads on the basis of their similarity with either α and β. Comparisons using Euclidean distances basically rely on the paradigm that

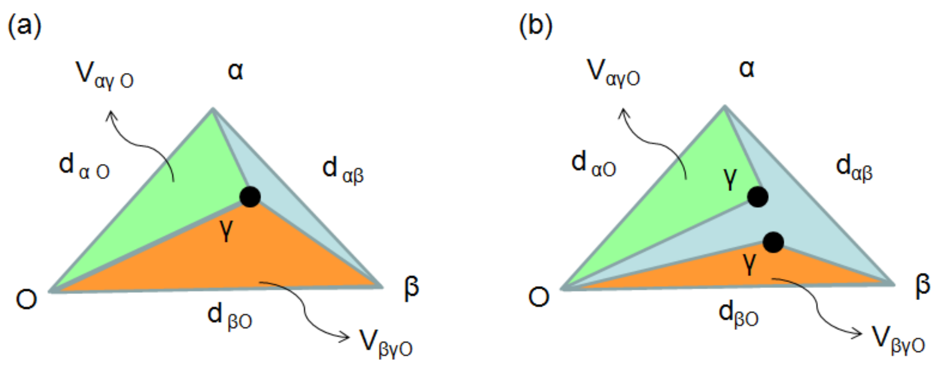


Figure 6.2: **Triangle inequality** (a) Example of Euclidean distances in non-coordinate system where the triangle inequality holds, i.e. $d_{\alpha\beta} < d_{\alpha\gamma} + d_{\beta\gamma}$. (b) Example of non-Euclidean distances in non-coordinate system where the triangle inequality does not hold. Vertex representing γ is split in two to stick with the constraint induced by the distances

states that the probability of a given point γ of the set to belong to the cluster identified by β is a function of the ratio $d_{\gamma\beta}/d_{max}$. Unfortunately, when the distance metric is non-Euclidean, this ratio can lead to inaccurate clustering performance, which may lead to wrong assignments. Therefore, it is necessary to set up a scheme that allows overcoming this issue.

Then, let us take another look to Fig. 6.2. Introducing a point (O) outside the space spanned by α, β and γ we can collect more information to get to a stable and reliable clustering metric. The use of point O allows considering triangles instead of distances and in general correspond to the use of the so-called "distance geometry" [83], which works on non-coordinate systems, delivering excellent performance in clustering and classification.

Specifically, it is possible to think that the similarity metric is a function of the areas induced by the triangles composed by O, γ and one between α and β (i.e., $V_{\alpha\gamma O}$ and $V_{\beta\gamma O}$ in Fig. 6.2, which are the green and orange areas, respectively). In other terms, the angles $\beta\hat{O}\gamma$ and $\alpha\hat{O}\gamma$ may replace the distances $d_{\beta\gamma}$ and $d_{\gamma\beta}$ and play a key-role in the definition of the similarity score.

The identification of the similarity metric as a function of the aforesaid areas implicitly relies on the simplicial decomposition paradigm, for which each point in a given simplex can be seen as a combination of its vertices [84]. In our case the simplex is a triangle (3-simplex), so that any point (i.e. read) γ lying within the triangle $\alpha O\beta$ can be written as a linear combination of the three vertices as long as such triangle contains them all, i.e. each read γ can be written as:

$$\gamma = \zeta_\alpha\alpha + \zeta_\beta\beta + \zeta_O O \tag{6.3}$$

where $\zeta_\alpha + \zeta_\beta + \zeta_O = 1$.

The weights (or abundances in the distance geometry literature) identify a reliable set of similarity metrics, so that if $\zeta_\alpha > \zeta_\beta$ the read γ can be clustered to α and viceversa. For this reason, O must be defined as the point that together with α and β spans the largest simplex. Specifically, the distances $d_{\alpha O}$ and $d_{\beta O}$ have to been set in order to fulfill this condition. Supplementary Section 3 provides the motivations and the actual setting for the aforesaid distances. Moreover, the next subsection introduces the proposed distance geometry-based labeling scheme.

6.2.3 Labeling using non-Euclidean metrics

As previously mentioned, the angles in the triangle $\alpha O \beta$ can play a key-role in read labeling. Specifically, the angle $\alpha \hat{O} \beta = \Gamma$ can be used to discriminate the maximum distance reads. According to Carnot's theorem, it is possible to draw an upper bound and a lower bound for Γ. Supplementary Section 3 provide a detailed description of their derivation. Briefly, we set the maximum value of Γ to $\Gamma_{max} = 60°$, whilst its minimum value Γ_{min} can be derived by the following equation:

$$cos\Gamma_{min} = 1 - \frac{1}{2|\Delta_{\alpha\beta}|^2} \tag{6.4}$$

Thus, if $\Gamma < \Gamma_{min}$, there is no reliable separation between the maximum distance reads, so that all the reads under analysis can be labeled uniformly. This implies that the two alleles are homologous or that only one allele has been processed. Thus, let two reads α and β be the points at maximum distance, as in Fig. 6.2. Then, the likelihood $L(\gamma \sim \alpha)$ that a given read γ belongs to the allele represented by read α is:

$$L(\gamma \sim \alpha) = 1 - \frac{V^2_{\alpha\gamma O}}{V^2_{\alpha\gamma O} + V^2_{\beta\gamma O}} \tag{6.5}$$

where V^2_{ABC} identifies the squared area of the triangle defined by vertices A, B, and C. Computing this area requires to first compute the matrix of the distances

$$D_{ABC} = \begin{bmatrix} 0 & d_{AB} & d_{AC} \\ d_{AB} & 0 & d_{BC} \\ d_{AC} & d_{BC} & 0 \end{bmatrix}$$

Then $V^2_{ABC} = -16det(CM_{ABC})$

where $CM_{ABC} = \begin{bmatrix} 0 & \underline{1} \\ \underline{1}^T & D^2_{ABC} \end{bmatrix}$

(see additional material for details). It is important to note that the computed likelihoods must be normalized to the sum $V^2_{\alpha\gamma O} + V^2_{\beta\gamma O}$ in order to fulfill the sum-to-one constraint, i.e., to make $L(\gamma \sim \alpha) + L(\gamma \sim \beta) = 1$. $L(\gamma \sim \alpha)$ and $L(\gamma \sim \beta)$ are proportional to the weights $\zeta_\alpha, \zeta_\beta$ (see eq. 6.3) and can be then used to cluster γ with β or α.

6.2.4 Labeling-based genotyping

Taking into account the process that has been introduced in the previous subsections, it is possible to efficiently perform genotyping of NGS reads by properly managing the outcomes of the distance geometry approach, summarized by the flowchart of Fig. 6.3. The first step of the algorithm involves applying eq. 6.4. The i-th mutation is called as homozygous if the following property holds:

$$\alpha\hat{O}\beta < \Gamma_{min} = \arccos[1 - \frac{1}{2|\Delta^2_{\alpha\beta}|}] \tag{6.6}$$

Otherwise, genotyping is performed taking into account the labeling process. Specifically, labeling splits $M(i)$ into two disjoint sets, $L_1(i)$ and $L_2(i)$. $L_1(i)(L_2(i))$ is defined as the set of the reads that are most likely belonging to the allele represented by read $\alpha(\beta)$. Then, we define $B_1(i)(B_2(i))$ as the base in the i-th position given by the reads in $L_1(i)(L_2(i))$. A majority rule is employed if the reads in $L_1(i)(L_2(i))$ show different bases on the i-th position.

The coverage of each variant plays a key-role in the genotyping procedure. Indeed, let us define Θ as the threshold value for the number of reads that map onto each variant. Thus, when genotyping the i-th variant, we check $|M(i)|$ w.r.t. Θ. Thus, if $|M(i)| < \Theta$, the i-th mutation is called as heterozygous as $B_1(i) \neq B_2(i)$: otherwise, a homozygosity is declared.

On the other hand, if $|M(i)| \geq 0$, further investigations on $L_1(i)$ and $L_2(i)$ have to be performed. Specifically, we consider the most represented base between $B_1(i)$ and $B_2(i)$ over $L_1(i)$ and $L_2(i)$, respectively. Then, we check whether the number of reads which show the most represented base over the i-th variant is less than a fraction ψ of the total reads mapping onto variant i. In case this condition is fulfilled and $B_1(i) \neq B_2(i)$, a heterozygosity is declared. Otherwise, a homozygosity is called. Supplementary Section 5 reports a detailed description of the aforesaid procedure.

Taking a look back to the genotyping process, we can say that the parameter Θ can be seen as the limit of the informativity associated with the reads in $\hat{B}$. In other terms, each read in $M(i)$ can provide useful information as it represents at least $1/\Theta$ of the global information delivered by $M(i)$. If this condition is not fulfilled, its informativity will not be discarded as $|\hat{B}| < \psi \cdot |M(i)|$. Throughout this paper we assume $\Theta = 20$ and $\psi = 0.8$ in order to achieve confident genotypes at moderate or high coverage (>20), as shown by literature [40].

Finally, the computational complexity of the proposed scheme is strongly dependent on the labeling process computational cost and on the steps needed to compute and compare weights (eq. 3).Thus, it is possible to draw an upper bound of the computational cost required by Kimimila for each variant as a function of the operations required by labeling and variant coverage: the upper bound can be written as $|M(i)| \cdot N \cdot \begin{pmatrix} |M(i) \\ 2 \end{pmatrix}$, where N is the maximum

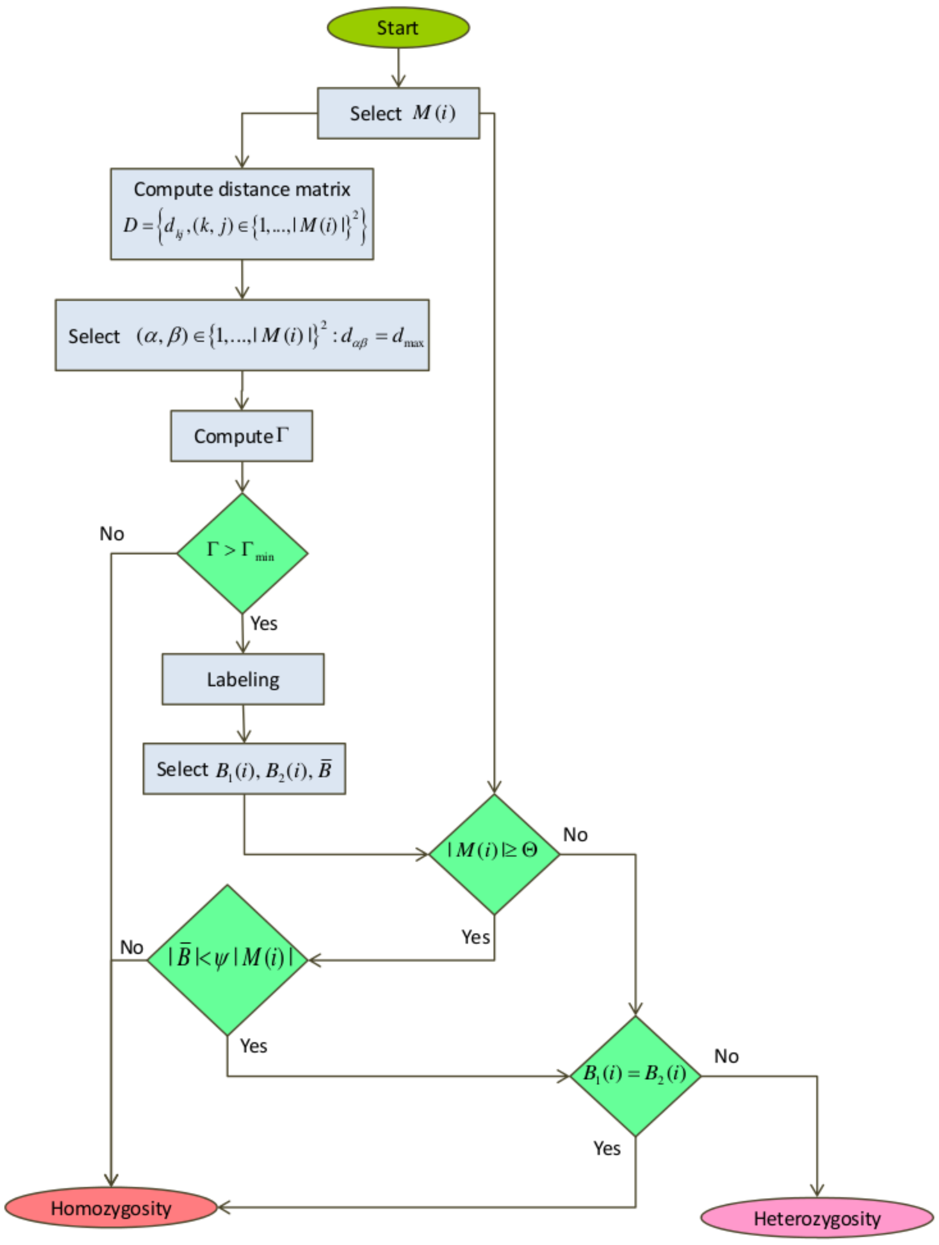


Figure 6.3: **The flowchart of the proposed algorithm**

number of variants two reads in $M(i)$ overlap on.

6.3 Datasets and Experimental Methods

The proposed method has been tested over in-silico and real dataset to exploit its performance in terms of read labeling and genotyping accuracy. The aforementioned datasets are introduced in the following Sections.

6.3.1 In-silico dataset

Simulated dataset consisted of a pool of sequence reads along with their base qualities and the mapping position on the genomic reference (GRCh37). The allele origin of each read was known. The dataset was built according to the following procedure:

1. Two copies of the human genomic reference GRCh37 corresponding to the same genomic region were taken to simulate the two possible alleles;
2. The two alleles sequences were modified according to a phased variation set on the above genomic region;
3. In order to simulate the allelic reads unbalance, the genomic region of interest was split in five intervals weighted in accordance to a pre-established allele frequency.
4. Sequence reads were generated by randomly choosing sequence snippets from the modified allele sequences. Snippets were generated from one of the two alleles according to the allele frequency of the interval.
5. For each read, base qualities were assigned following a linear function $Q = ((m - M + 2r)/L)x + (M - r)$ where x is the current base, m and M are the minimum and maximum quality scores (respectively set to 20 and 41 in Phred scale), L is the read sequence length and r is an integer random variable between 0 and 5.

To simulate next generation sequencing reads and alignment artifacts, we introduced noise by randomly choosing for each variant locus a set of overlapping reads (S) for which the corresponding base at the variant site within its quality Phred scores (Q) had to be changed. We distinguished between three combinations of S and Q to build the resulting datasets corresponding to:

1. Low noise dataset (S=[0,2], Q=[1,10])
2. Medium noise dataset (S=[0,3], Q=[1,20])
3. High noise dataset (S=[0,3], Q=[20,41])

A total of 960 reads were generated. Each read was 101 bases in length and mapped to one of the five genomic intervals taken under consideration and spanning 11 variations. The number of generated reads allowed an average coverage of about 30-fold.

6.3.2 Real dataset

A dataset was downloaded from the 1000 Genome Project public resource. In particular, the test of the algorithm was performed on Illumina alignment sequencing data (BAM files) of the HG00096 exome-sequenced sample belonging to the 2nd phase release. HG00096 SNPs array data obtained from the Illumina Omni 2.5M Chip, phased by SHAPEIT software [85] were used as validation set. By the intersection of above two data sources we obtained the set of reliable variants within their overlapping reads. Therefore, for each read we were able to infer the allele origin basing on the presence or absence of a reliable variant on it, except for those reads covering only one homozygous variant.

6.4 Results

The algorithm was tested on the simulated dataset in order to evaluate the performances of the algorithm in labeling reads according to their allele origin and on the real dataset to evaluate reads labeling and variant genotyping. In both cases Kimimila showed accurate results.

We finally compared Kimimila genotyping performances on the real dataset to GATK UG, VarScan, Samtools, Atlas2 and FreeBayes tools.

6.4.1 Results on simulated dataset

For the simulated dataset, the proposed method shows a reads labeling accuracy greater than 85% over 91% of the variants in case of low noise condition, 83% of the variants in case of medium noise condition and 81% of the variants in case of high noise condition (see Fig. 6.4). It is worth noting that the labeling accuracy is computed as the fraction of reads that have been correctly clustered w.r.t. their allele origin.

6.4.2 Results on simulated dataset - Reads labeling

As the real dataset is concerned, we tested the algorithm on a subset of 1094 reads overlapping 76 variants randomly chosen. The labeling accuracy results are shown in Fig. 6.5. We compared Kimimila against the performance of a standard clustering method such as the K-means algorithm. In order to get a fair comparison, the K-means algorithm has been fed with the distances that have been computed for each variant over the interval for which the maximum

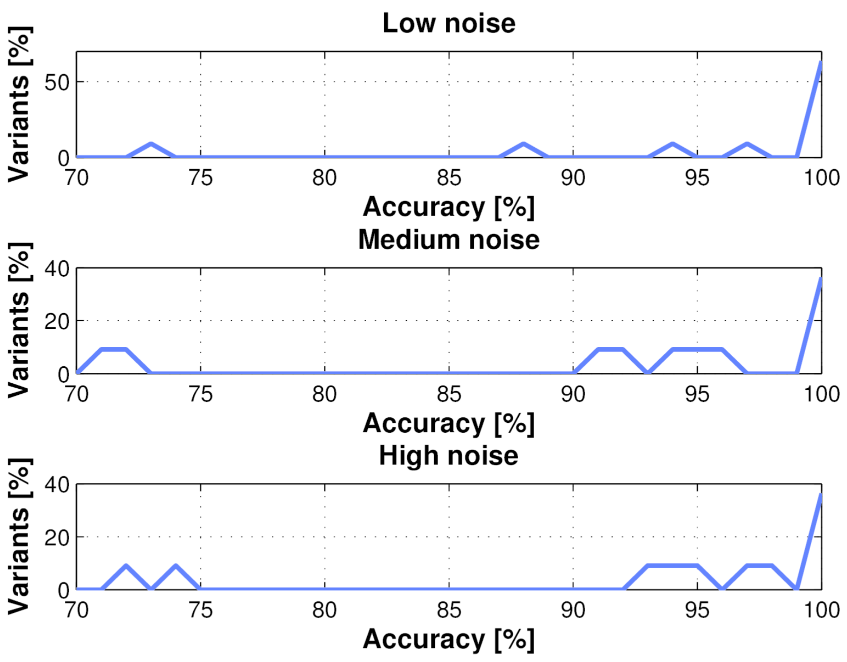


Figure 6.4: **Labeling performance over synthetic datasets characterized by different noise injection** The plot shows over the y-axis the percentage of variants for which the labeling has been performed with the corresponding accuracy that is reported over the x-axis

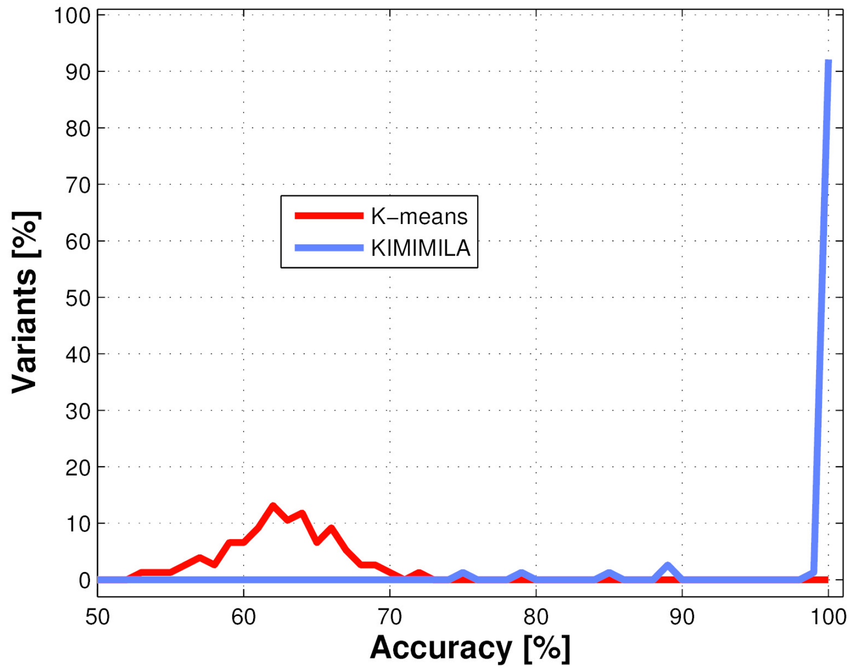


Figure 6.5: **Labeling performance over a real dataset having 76 variants** The plot shows over the y-axis the percentage of variants for which the labeling has been performed with the corresponding accuracy, reported over the x-axis. The performance of K-means and Kimimila are shown as a red solid line and a blue solid line, respectively

distance reads overlap. The performance of K-means and Kimimila are shown in Fig. 6.5 as a red solid line and a blue solid line, respectively. As previously mentioned, the labeling accuracy is computed as the fraction of reads that have been correctly clustered w.r.t. their allele origin. The labeling performance of K-means algorithm stands in the interval between 52% and 70%. On the other hand, Kimimila performs optimal labeling (i.e., is able to perfectly identify the allele origin) over 92% of the variants. Moreover, the proposed method shows a labeling accuracy greater than 85% over 97% of the variants.

6.4.3 Results on simulated dataset - Genotyping and comparison to the other algorithms

Genotyping accuracy was tested on all 69834 Omni Chip variants of HG00096 sample on all autosomes for which overlapping reads occurred. Results on genotyping were compared to those achieved by GATK UG, VarScan, Samtools, Atlas2 and FreeBayes. Table 6.1 shows the genotyping results over each chromosome for every level of coverage. Specifically, in Table 6.1 the percentages of variants that have been correctly genotyped by Kimimila and the other tools are provided. VarScan is not included here because it does not genotype variants with few supporting reads, therefore a comparison on the whole data set regardless of the coverage threshold would not have been fair. As a result Table 6.1 shows that the five considered methods provide approximately the same accuracy in genotyping throughout the chromosomes. The computational time required by Kimimila to perform genotyping can be estimated in an average of 6.7ms/variant over a 8GB RAM Intel i5 processor.

Chr	GatkUG	Samtools	FreeBayes	Atlas2	Kimimila
1	90.7	90.8	89.3	91.5	90.64
2	89.3	89.4	88	90.1	89.19
3	90	89.9	88.6	90.6	89.98
4	89.3	89.3	88.1	89.8	89.41
5	90.2	90.2	88.8	90.6	90.14
6	91.3	91.4	90.2	92.2	91.33
7	89.7	89.7	88.3	90.4	89.75
8	86.7	86.8	85.3	88	86.84
9	89.6	89.6	88.6	90.7	89.51
10	89.9	90	88.4	90.7	90
11	91.1	91	89.3	91.8	91.05
12	91.2	91.2	89.9	92.1	91.15
13	88.4	88.3	87.2	89.1	88.29
14	89.9	89.9	88.4	90.6	89.48
15	90.9	90.9	89.6	91.6	90.68
16	88.6	88.6	87	89.8	88.56
17	92.2	92.3	90.8	92.9	92.1
18	85.8	85.8	84.1	86.6	85.51
19	94.3	94.4	92.5	95	94.28
20	89.6	89.8	87.9	91	89.76
21	87.3	87.3	86	88	87.49
22	91.5	91.5	88.9	93	91.41
Average	**89.88**	**89.91**	**88.42**	**90.73**	**89.84**

Table 6.1: **Results on genotyping of the proposed methods** The table reports the percentage of variants (covered at least by 10 reads) over each chromosome that have been correctly genotyped by Kimimila, GATK, Samtools, FreeBayes and Atlas2 algorithms

Moreover, it is worth to note that a fraction of variants have been incorrectly genotyped because of insufficient coverage from one of the two alleles. As shown in Table 6.2, we stratified genotyping accuracy by reads coverage and we found that below a threshold of 10, the accuracy on heterozygous calls decrease to 50% or less for each tool, reflecting their tendency to easily call for homozygous variants in case of a low allele ratio. Indeed we observed a high accuracy (above 99%) on homozygous calls independently from the coverage. These results are compliant with previous literature findings in this field [86, 87] and to the recommendations that for diagnostic purposes an average coverage above 50-fold with at least 10 reads covering more than 90% of the genomic region of interest should be assured for quality assessment.

Tool	Coverage (C)	Accuracy	TP rate (Hom)	TN rate (Het)	FP rate	FN rate
GATK	C>50	.999	.999	.999	.0005	.0007
	C>10	.999	.998	.999	.0008	.0011
	10≤C≤50	.998	.998	.998	.001	.0017
	C<10	.808	.998	.500	.500	.0015
Samtools	C>50	.999	.999	.999	.0005	.0007
	C>10	.999	.999	.999	.0007	.0008
	10≤C≤50	.999	.999	.998	.001	.0007
	C<10	.807	.996	.501	.498	.0034
FreeBayes	C>50	.999	.999	.999	.0005	.0008
	C>10	.998	.999	.998	.0017	.0008
	10≤C≤50	.997	.999	.996	.0036	.0005
	C<10	.778	.999	.418	.582	.582
Atlas2	C>50	.999	.999	.999	.0005	.0008
	C>10	.998	.999	.998	.001	.001
	10≤C≤50	.998	.999	.997	.002	.0007
	C<10	.814	.998	.519	.480	.0014
VarScan	C>50	.999	.999	.998	.001	.0007
	C>10	.993	.999	.998	.009	.0006
	10≤C≤50	.987	.999	.998	.02	.0006
	C<10	-	-	-	-	-
Kimimila	C>50	.999	.999	.998	.0015	.0005
	C>10	.997	.995	.998	.001	.004
	10≤C≤50	.995	.992	.998	.002	.008
	C<10	.809	.996	.506	.493	.003

Table 6.2: **Comparison of genotyping accuracy between GATK Unified Genotyper, VarScan, Samtools, FreeBayes, Atlas2 and Kimimila** Here, homozygous and heterozygous variants are represented as positive and negative class respectively. True positive/negative rate and false positive/negative rate are reported as well. Results are stratified according to coverage (above 0, below 10, between 10 and 50 and above 50)

6.5 Discussion

Given the performance that have been previously reported, the algorithm is based on a different approach w.r.t. the other methods that have been considered. Indeed, it is able to assign and report the allelic origin for each read overlapping a variant locus. The increasing in length of reads produced by third generation sequencers will increase the number of haplotype informative reads as well, leading this approach to more accurate reads labeling and genotyping. Finally, it is important to note that Kimimila reached high accuracy in genotyping without making use of a priori knowledge, suggesting that the method can be suitable for those cases where lack of knowledge is present (i.e. genomes of other species).

Variant genotyping is only one of the possible applications of this method: further potential research directions are presented in the following. Haplotyping, which aims to link together (phase) two or more variants belonging to the same alleles, is one possible application. The majority of haplotype phasing algorithms are based on the correlation between alleles at specific SNPs in a population (so called linkage disequilibrium) [85, 88, 89, 36]. However,these approaches are usually less accurate if compared to other methods that make use of heterozygous variants on mapped reads for haplotype phasing [90, 91].

Although this method was aimed at defining the zygosity of a variant, the approach used to classify sample fragments could be easily reused to reconstruct the haplotype as well. If the overlapping rate between fragments is adequate and these fragments are long enough to encompass multiple variant sites (as expected by sequence technology improvement), a future development that aims to assemble these fragments to reconstruct long and reliable haplotype-blocks will be possible. Another potential research directionis related to polyploid domains as in the case with somatic samples. In order to fulfill this goal, the distance geometry approach would need to be adapted to a generic N-dimensional space, where N is the number of alleles of the sequenced sample. After this adaptation it will be possible to a) infer the number of sub-population genomes in the sample and b) genotype and phase variants belonging to each genome within the mixture.

6.6 Conclusion

This chapter presented a novel algorithm, Kimimila, able to classify reads that are mapped to a reference genome by their allele origin; as a consequence the algorithm can be used to assign the genotype of the genomic variant loci overlapped by the reads. A graphical abstract is showed in Fig. 6.6.

The algorithm is based on an original clustering technique based on a distance geometry approach for reads comparison; it makes use of both the nucleotide frequency for a single variant locus across the overlapping reads and the sequenced base qualities assigned by the base caller. Two sources of information were integrated into a single model that, dealing with reads rather than only single base pileup, is able to combine more variant loci when present on the same read.

The algorithm was tested on a simulated dataset in order to evaluate both the reads classification and variant genotyping. Kimimila is able to reach high performances despite high levels of noise. Finally Kimimila was compared to several algorithms for variant genotyping: GATK UG, Samtools, Atlas2, Free-

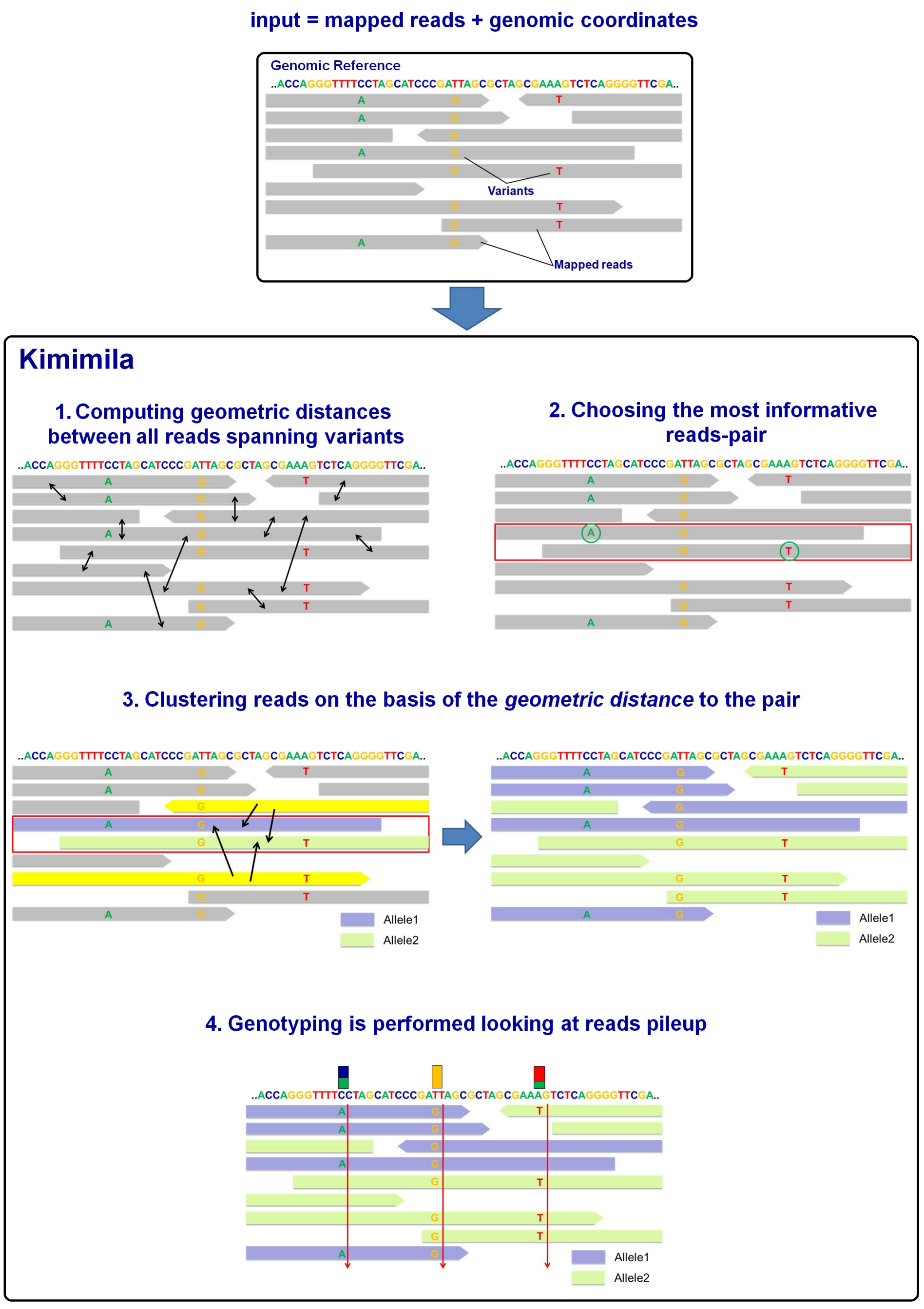

input = mapped reads + genomic coordinates
Genomic Reference
..ACCAGGGTTTTCCTAGCATCCCGATTAGCGCTAGCGAAAGTCTCAGGGGTTCGA..
Variants
Mapped reads
Kimimila
1. Computing geometric distances between all reads spanning variants
2. Choosing the most informative reads-pair
3. Clustering reads on the basis of the geometric distance to the pair
Allele1
Allele2
4. Genotyping is performed looking at reads pileup

Figure 6.6: **A graphical abstract of Kimimila**

Bayes and VarScan. These methods are comparable in terms of performances and suffer from the same limitations for shallow genome coverage.

Chapter 7

Variant analysis

This chapter will present the results achieved by the sequencing project[1]. In this study, we performed a comprehensive mutation analysis of genes implicated in myeloid malignancies in a large and clinically well characterized cohort of CMML patients with the aim to dissect relationships between genotype and disease phenotype and to integrate somatic mutations into a clinical/molecular prognostic model.

7.1 Patients

All investigations were approved by the Ethics Committee of the Fondazione IRCCS Policlinico San Matteo, Pavia, Italy, and other local Institutional Review Boards. The procedures followed were in accordance with the Helsinki Declaration of 1975, as revised in 2000, and samples were obtained after subjects provided informed consent.

Two-hundred-fourteen patients (pts) with CMML according to WHO classification (2008) were included in the study. Clinical and hematological features of patients at the time of mutational analysis are reported in Table 7.1. Myelodysplastic and myeloproliferative subtypes (CMML-MD and CMML-MP, respectively) were defined according to FAB criteria. CPSS was calculated according to [92].

7.2 Mutation analysis

Genomic DNA was obtained from bone marrow mononuclear cells or peripheral blood granulocytes by following standard protocols for human tissue. Bar-coded sequencing libraries were prepared and target enrichment for

[1]The content of this chapter has been partially reported for poster presentation at 56th ASH Annual Meeting and Exposition as *Somatic Mutations of ASXL1, RUNX1 and SETBP1 Improve Prognostic Stratification of Patients with Chronic Myelomonocytic Leukemia, C. Elena, A. Gallí, E. Such, U. Germing, E. Rizzo et al.*

the selected coding genes was performed before samples were pooled and analyzed by high throughput sequencing using Illumina MiSeq (Ilumina, San Diego, CA). The following genes were studied: ASXL1, BCOR, CBL, CEBPA, CSF3R, CUX1, DNMT3A, EP300, ETNK1, ETV6, EZH2, FLT3, IDH1, IDH2, JAK2, KIT, KRAS, MLL, MLL2, MLL3, MLL5, NF1, NPM1, NRAS, PHF6, PTPN11, RUNX1, SETBP1, SF3B1, SRSF2, STAG2, TET2, TP53, U2AF1, UTX, WT1, ZRSR2. Sequencing output data have been processed with the pipeline presented in Chapter 3 through a cloud cluster introduced in Chapter 4. Despite the defined pipeline is able to search for structural variants, the following results refer to point mutation, short insertions or deletions because target resequencing data have limited power in identifying structural variants when compared to the robustness of results in term of point or small mutations.

7.2.1 EmatoDB

An important goal to achieve was related to the selection of a well defined database of known genomic structural variations related to MDS/MPNs neoplasms that could help in identify and prioritize variants found in patients sequenced at the Department of Hematology Oncology of Policlinico San Matteo.

In order to populate such database, called EmatoDB, about 100 scientific publications from 2009 to 2014 have been included in the analysis. All the selected studies refer to mutational analysis of patients with MDS/MPNs syndromes characterized by NGS and report hundreds of indentified variants. However, despite all the resulting variants were referred to hg19, they had different formats, often not well documented, because there is not yet a standard nomenclature for variants. Discussions regarding the uniform and unequivocal description of sequence variants in DNA and protein sequences were initiated more than 10 years ago but although the Human Genome Variant Society (HGVS) nomenclature recommendations [93] are becoming largely accepted, a lot of researchers still use different description of variants.

HGVS has set standards for describing variants at DNA level and protein level but it recommends to describe all variants at the most basic level, the DNA level. Descriptions should always be in relation to a reference sequence, either a genomic or a coding DNA reference sequence. Despite a genomic reference sequence seems best, in practice coding DNA reference sequences are preferred. Accordingly to HGVS variants are reported at DNA level as shown in the following example:

- c.85A>T

where c. is a coding DNA sequence, 85 is the base affected and A>T the sequence change. The example encodes a substitution but there are recommendations for deletions, duplications, insertions, inversions, translocations and repeated sequences.

Sequence changes at the protein level are described like those at the DNA but should only be given in addition to a description at DNA level. In this case a “p.” preceding the change is used to indicate a description at the protein level.

The database creation process has therefore combined both manually and automated variant importing. Variants respecting HGVS recommendations were imported automatically while not standardized variants were revised manually, converted to HGVS and finally imported on EmatoDB. Table 7.2 shows the five mandatory field stored in EmatoDB for each variant and it is worthy to note that variants are stored in terms of genomic positions with respect to a reference genome.

The majority of variants reported in the investigated publications were however annotated in relation to coding DNA reference sequences, so even if they were in HGVS format an additional conversion step from coding coordinates to genomic coordinates was required before the upload.

Furthermore, several publications reported variants only in terms of protein changes. Such format often can’t be univocally associated to a coding DNA change but, on the contrary, can reflect a variable number of coding sequence changes. As a result, the protein changes that can be generated by a limited number of coding sequence changes have been translated in genomic changes and uploaded on EmatoDB while not solvable protein changes were discarded.

While the majority of publications only reported lists of variants unconfirmed by other sequencing techniques, few articles Sanger-sequenced the identified variants. As a result, variants that were confirmed by Sanger should have been considered somatic and therefore highly weighted with respect to other variants. Such additional information have been therefore stored in EmatoDB by adding to table 7.2 an additional field with value *tumor variant*. Variants in EmatoDB can be classified indeed as *tumor variant* or *polymorphism* if there are supporting evidences or none of two, otherwise. Tumor variants can be then detailed as *oncogenic*, *possibly oncogenic* or *sanger*.

The initial importing procedure on EmatoDB has only tagged as *tumor variant - sanger* variants that have been confirmed by Sanger while the others were uploaded without a tag.

7.2.2 Annotation

As discussed before, variants that have been found on processed data need to be annotated in order to make sense of them. In particular, annotation enriches variants with additional features that can be used to define what is the impact on the protein functions besides the classification of such changes as somatic or not. As already stated, the custom annotation process is performed by Annovar that firstly use RefSeq to enrich variants with information relating genomic positions to intron-exon boundaries, gene names and protein products. Then variants are matched and annotated accordingly with the following databases:

- snp138
- snp138NonFlagged
- cosmic68
- 1000g2012apr_all
- ljb23_all
- esp6500_all
- EmatoDB

The *snp138* annotation adds the corresponding dbSNP identifier to variants that are reported in dbSNP version 138.
The *snp138NonFlagged* database is a reduced version of dbSNP 138 that includes variants with minor allele frequency $> 1\%$. It results in annotating only variants that can be considered frequent in population.
Cosmic68 database is a Catalogue Of Somatic Mutations In Cancer (COSMIC) and the corresponding annotation reports the identifier of variants that are present in such database.
The *1000g2012apr_all* annotation adds the information about the frequency of a specific variant above the 1000 Genomes Project population, if present.
By annotating against *ljb23_all* variants are enriched with information derived by dbNSFP. The latter is a database developed for functional prediction and annotation of all potential non-synonymous single-nucleotide variants in the human genome. Single nucleotide variants are therefore enriched with several scores that reflect their effect on the protein. In particular such annotation includes score calculated with different prediction software as SIFT, PolyPhen2, LRT, MutationTaster, FATHMM, GERP++, PhyloP and SiPhy, but we included in the analysis the first four only.
ESP is an exome sequencing project aiming at identify genetic variants in exonic regions from over 6000 individuals so the *esp6500_all* annotations reports the frequency of a variant above this population, if present.
EmatoDB annotation finally adds the identifier and the journal reference if a variants is present in the database defined in the previous section.

Once variants have been annotated using Annovar, an additional enrichment step was focused on pathways and conserved domains. In order to find the pathways in which targeted genes are involved, the KEGG database [94] was consulted and each variant has been enriched by adding pathway's identifiers that include the corresponding gene as a component. To enrich variants accordingly to conserved domains the Conserved Domain Database [95] was consulted: conserved intervals within targeted coding nucleotide sequences were retrieved and variants that fall inside such intervals were enriched with the corresponding conserved domain identifier.

Finally, variants have been enriched with PROVEAN [96] annotation as well. As the others in silico predictors, this tool predicts whether an amino acid substitution or indel has an impact on the biological function of a protein.

7.2.3 Tumor/Normal analysis

As previously stated, 214 tumor samples were sequenced during the study. 74 matched normal samples have been sequenced as well. Once processed and annotated through the pipeline, these data have been manually revised by physicians in order to refine the filtering and tagging procedure of variants that will be detailed later. For each variant identified in a tumor sample this process evaluated the following parameter:

- presence in the corresponding normal sample;
- allelic fraction of reads supporting the variant;
- frequence across patients;
- presence in dbSNP, 1000 Genome Project, Cosmic and EmatoDB;
- effect on the protein according to in-silico predictors.

Variants supported by strong evidence have then been tagged as tumor variant or polymorphism and imported in EmatoDB. For variants already present in EmatoDB the classification tags were updated accordingly to this analysis.

7.2.4 Filtering procedure

Because the study aimed to confirm and discover new variants typical of patients affected by CMML cancer, a filtering procedure was required to remove variants that could have been considered as germline events. If samples have a normal counterpart, such filtering procedure can be easily done through the somatic variant caller implemented in the pipeline or by subtracting variants that have been called against the reference genome from tumor and normal samples. However this procedure didn't work for the majority of patients that have been considered in this study since a normal matched sample wasn't available and as a result of the variant calling procedure the final variant set included hundreds of variant for each patients. By applying the filters that will be discussed here following, 97% of the variants have been filtered out and the amount of variants per patient that can be considered somatic is comparable with somatic variants identified in tumor normal matched samples.

Variants were initially filtered according to their allelic fraction. The allelic fraction was calculated for each variant of all patients that support it and a filtering flag was assigned to each patient according to the following rule:

- $45\% <$ allelic fraction $< 55\%$ or allelic fraction $> 90\%$

because such frequencies are typical of germline variants, in particular when tumor cell purity is not 100%; we have applied the filtering when studied samples show levels of contamination around 10% at least. If the variant was flagged to be filtered for more than 50% of patients and it wasn't annotated as present in Cosmic or EmatoDB, it was filtered out for all patients.

Mean, standard deviation and coefficient of variation of allelic fraction were then calculated for each variant among patients. A statistical analysis on tumor normal matched samples showed that a cutoff of 0.12 on the coefficient of variation was a good compromise between sensitivity and specificity to separate germline and somatic variant. As a result a variant was filtered if its coefficient of variation was greater than 0.12 and was found in at least five patients. Variants that were annotated as tumor variant in EmatoDB were however rescued.

Finally the following self-explaining filters were applied to the remaining variants:

- variants that fall outside coding regions were filtered;
- variants with allele frequency greater than 1% in dbSNP were filtered unless present in Cosmic or EmatoDB;
- variants with allele frequency greater than 0.14% in 1000 Genome Project were filtered unless present in Cosmic or EmatoDB;
- variants with coverage lower than 30x and less than 10 supporting reads were filtered;
- variants with allelic fraction lower than 0.02 were filtered.

7.2.5 Variant tagging

The last step of variant analysis and interpretation has consisted in a tagging procedure for unfilterd variants in order to highlight oncogenic ones. In particular variants were tagged as *oncogenic* if:

- annotated in EmatoDB as *tumor variant - sanger*;
- annotated in Cosmic as *haematopoietic and lymphoid tissue*;
- more than 75% of in silico predictors were concordant about the damaging effect of the variant;
- more than 60% of in silico predictors were concordant about the damaging effect of the variant and it was inside a conserved domain;

while as *polymorphism* if annotated in EmatoDB as *polymorphism* and as *possible oncogenic* if annotated in EmatoDB as *tumor variant.*

Once this initial tagging procedure was performed, variants that have been tagged as *oncogenic* were imported in EmatoDB with the tag *tumor variant - oncogenic*. If a variant was already present in EmatoDB its tag was updated.

The final stage of tagging procedure had scan all the untagged variants with respect to EmatoDB's variants that were tagged as *tumor variant - oncogenic*. In particular, each variant have been classified as *oncogenic* if:

- at least two patients support the variant that is no more than three amino acids far from an oncogenic EmatoDB one

and as *possibly oncogenic* if:

- one patient supports the variant that is no more than three amino acids far from an oncogenic EmatoDB variant

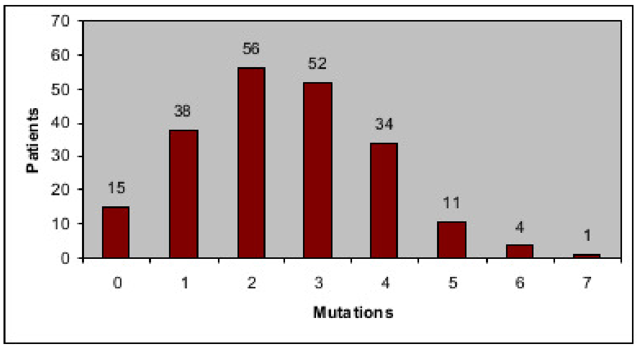
7.3 Statistical analysis

Survival analyses were performed with the Kaplan-Meier method. The cumulative incidence of acute myeloid leukemia (AML) evolution was estimated with a competing risk approach, considering death for any cause as a competing event. Multivariate survival analyses were performed by means of Cox proportional hazards regression. The effect of quantitative covariates on cumulative incidence of leukemic evolution was carried out using the Fine-Gray regression model. All analyses accounted for left censoring of the observations at the time of mutation assessment. The comparison of models with different types of covariates was carried out using Akaike's information criterion. Statistical analyses were performed using Stata 12.1 (StataCorp LP) software.

7.4 Results

7.4.1 Mutation spectrum and correlations between genotype and phenotype in CMML

Ninety-three percent of patients showed at least 1 oncogenic mutation (median number per patient: 2, range 0-7).The most frequently mutated genes are reported in Table 7.3 . Mutations in JAK2, NRAS and SETBP1 were significantly associated with CMML-MP (P values ranging from .03 to <.001), whereas TET2 and SF3B1 mutations were associated with CMML-MD (P=.007 and P=.024, respectively).



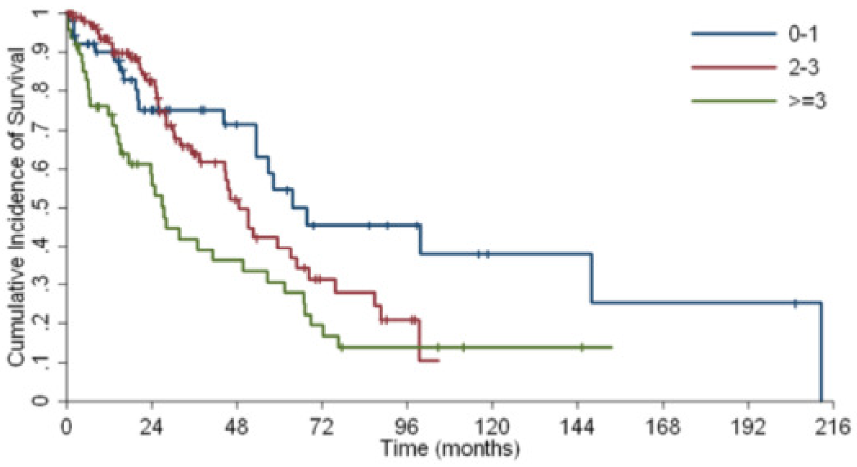
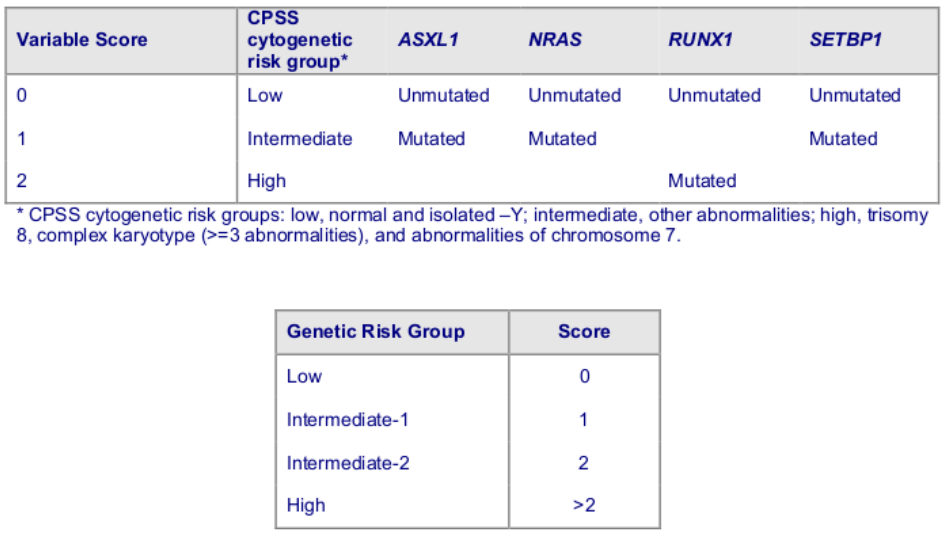


Figure 7.1: **Distribution of mutations per patient and prognostic impact on survival**

7.4.2 Prognostic value of genetic alterations in CMML and development of a clinical/molecular prognostic scoring system (CPSS-Mol)

The number of mutations per patient inversely correlated with overall survival (OS) (HR=1.23, P=.001) (Fig. 7.1). In univariate analysis, mutations in ASXL1 (HR=1.67, P=.032), EZH2 (HR=2.17, P=.021), NRAS (HR=2.04, P=.002), RUNX1 (HR=3.04, P<.001) and SETBP1 (HR=2.59, P=.002) significantly affected OS.

In order to investigate the additive value of somatic mutations to current prognostic assessment, we first performed multivariate Cox regression including CPSS cytogenetic risk categories and somatic mutations. The variables



Variable Score	CPSS cytogenetic risk group*	*ASXL1*	*NRAS*	*RUNX1*	*SETBP1*
0	Low	Unmutated	Unmutated	Unmutated	Unmutated
1	Intermediate	Mutated	Mutated		Mutated
2	High			Mutated	

* CPSS cytogenetic risk groups: low, normal and isolated –Y; intermediate, other abnormalities; high, trisomy 8, complex karyotype (>=3 abnormalities), and abnormalities of chromosome 7.

Genetic Risk Group	Score
Low	0
Intermediate-1	1
Intermediate-2	2
High	>2

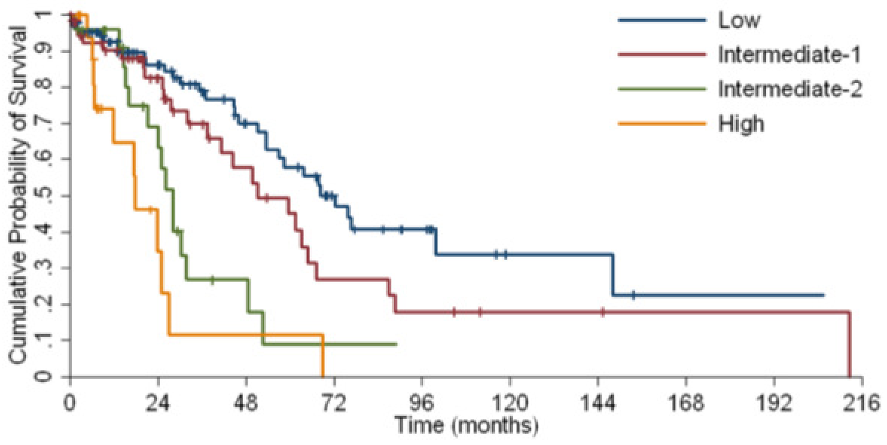
Figure 7.2: **Variables and scores of CMML-Genetic Prognostic Scoring System (CMML-GPS)**

that retained independent prognostic value were CPSS cytogenetic risk groups (HR=1.48, P=.026), and mutations in RUNX1 (HR=2.36, P=.018), NRAS (HR=2.17, P=.016), SETBP1 (HR=2.03, P=.05),and ASXL1 (HR=1.82, P=.022).

Based on regression coefficients, we defined a CMML-specific Genetic Prognostic Scoring System (CMML-GPS) that was able to identify 4 different groups with significantly different OS (HR=1.8, P<.001) and cumulative incidence of AML evolution (HR=2.62, P<.001). Fig. 7.2 summarizes variables and scores of CMML-GPS while Fig. 7.3 shows the survival and cumulative incidence of AML evolution according to CMML-GPS risk categories.

The Akaike information criterion showed that genetic risk score performed better than the original CPSS cytogenetic risk classification (AIC 664 vs. 684, respectively). According to the genetic risk score, 40.5% of patients had a shift toward a higher risk category compared with the cytogenetic classification.

Then, we performed multivariable regression analyses including new genetic risk categories, clinical and hematological variables. The following variables had a independent prognostic value: genetic risk groups (HR=1.5, P=.005), RBC transfusion-dependency (HR=2.74, P<.001), FAB subtype (HR=2.47, P<.001), WHO subtype (HR=2.12, P=.015). Based on regression coefficients, we defined a CPSS-Mol, which was able to identify 4 risk groups with different OS (HR=2.11, P<.001) and cumulative incidence of leukemic evolution (HR=2.58, P<.001). Fig. 7.4 summarizes variables and scores of CPSS-Mol while Fig. 7.3 shows the survival and cumulative incidence of AML evolution according to CPSS-Mol risk categories. The Akaike information criterion showed that the CPSS-Mol performed better than the original CPSS (AIC 590



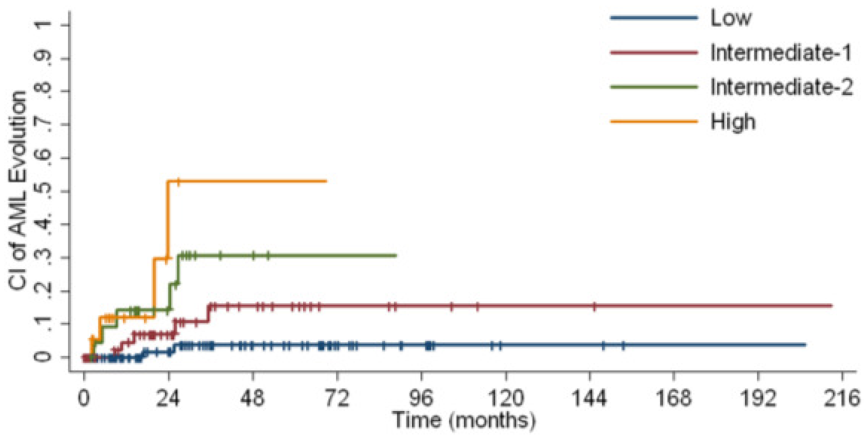
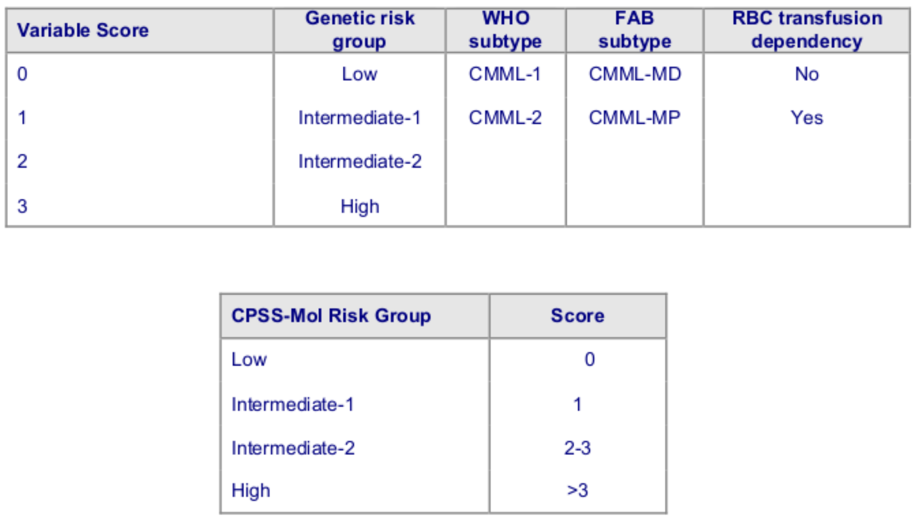


Figure 7.3: **Survival and cumulative incidence of AML evolution according to CMML-GPS risk categories**



Variable Score	Genetic risk group	WHO subtype	FAB subtype	RBC transfusion dependency
0	Low	CMML-1	CMML-MD	No
1	Intermediate-1	CMML-2	CMML-MP	Yes
2	Intermediate-2			
3	High			

CPSS-Mol Risk Group	Score
Low	0
Intermediate-1	1
Intermediate-2	2-3
High	>3

Figure 7.4: **Variables and scores of CPSS-Mol**

vs. 609, respectively).

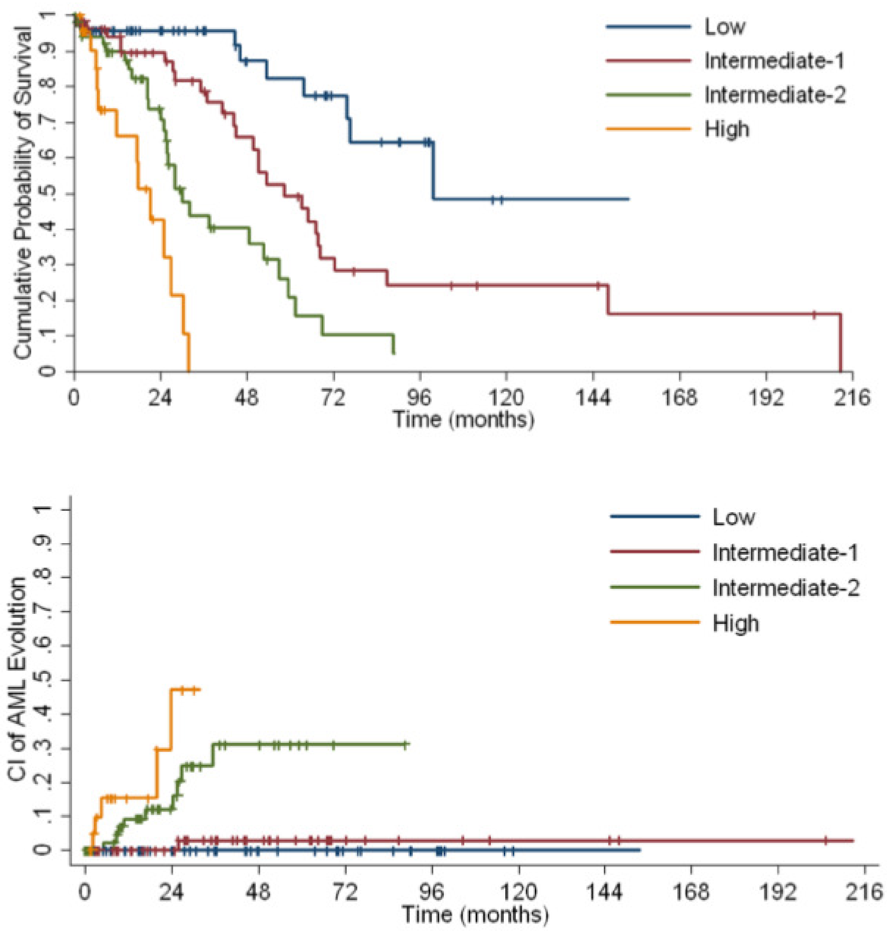


Figure 7.5: **Survival and cumulative incidence of AML evolution according to CPSS-Mol risk categories**

Patients, n.	214
Follow-up in months, median (range)	22 (0.4-212.6)
Age in years, median (range)	72 (28-99)
Males, n. (%)	151 (71)
FAB subtype, n. (%)	
CMML-MD	129 (60)
CMML-MP	85 (40)
WHO subtype, n. (%)	
CMML-1	176 (82)
CMML-2	38 (18)
WBC, x10ˆ9 /L, median (range)	9.8 (1.2-126)
ANC, x10ˆ9 /L, median (range)	5.1 (0.1-109.6)
AMC, x10ˆ9 /L, median (range)	2.1 (0.9-34)
Hgb, g/dL, median (range)	11.6 (6-16.6)
RBC transfusion dependency, n. (%)	54 (26)
PLT, x10ˆ9 /L, median (range)	124 (4-943)
PB blasts %, median (range)	0 (0-17)
BM blasts %, median (range)	3 (1-18)
BM ring sideroblasts, %, median (range)	0 (0-80)
Chromosomal abnormalities, n. (%)	44 (23)
CPSS cytogenetic risk group	
Low	150 (79)
Intermediate	21(11)
High	19 (10)
CPSS risk group	
Low	85 (46)
Intermediate-1	49 (27)
Intermediate-2	46 (25)
High	4 (2)

Table 7.1: **Clinical and haematological features of 214 patients with CMML at the time of mutational analysis**

VCF mandatory fields	
CHROM	chromosome
POS	1-based position of the start of the variant
REF	the reference allele
ALT	a comma separated list of alternate non-reference alleles
JOURNAL	reference to the scientific publication

Table 7.2: **EmatoDB fields**

Gene Mutation	CMML (%)	CMML-MD (%)	CMML-MP (%)	P value
TET2	44.4	51.6	32.1	.007
SRSF2	38.8	38.0	39.3	ns
ASXL1	29.0	25.0	34.5	ns
NRAS	11.7	4.7	22.6	<.001
SETBP1	9.4	3.9	16.7	.002
KRAS	8.9	9.4	7.1	ns
RUNX1	8.0	5.5	10.7	ns
CBL	8.4	7.8	8.3	ns
JAK2	7.0	3.9	11.9	.03
EZH2	7.0	5.5	8.3	ns
SF3B1	5.6	8.6	1.2	.03
U2AF1	4.2	4.7	3.6	ns
ZRSR2	4.2	5.5	2.4	ns

Table 7.3: **Relative frequency of gene mutations in the cohort of 214 CMML pts, considered as a whole, and subdivided according to FAB classification.**

Chapter 8

Overall conclusions

This dissertation was a comprehensive overview about the bioinformatics tasks that need to be addressed in order to perform an efficient next generation sequencing project. In this particular case, the overall procedure was referred to a targeted resequencing analysis of genes implicated in chronic myelomonocytic leukemia (CMML) in a large and well characterized cohort of patients but it can be easily scaled to exome or genome analysis. CMML is a myelodysplastic/myeloproliferative neoplasm characterized by a highly variable clinical course; since recurrent somatic mutations have been identified in such disease, the selected mutated genes were sequenced in order to provide useful prognostic information.

The defined pipeline of analysis was the result of a scrupulous comparison of different solutions that has brought to a robust combination of several state of the art tools to process next generation sequencing data. Such workflow implements the analyses usually referred as secondary and tertiary, while primary analysis is provided by the platform provider as part of the machine's function. Secondary analysis maps sequencing data to a reference genome and process them in order to be more reliable for the tertiary analysis, which mainly aims on identify all the differences between aligned reads and the reference genome. The pipeline's output is a combination of single nucleotide polymorphism, indel and structural variants that are then enriched with additional information in order to evaluate their impact and therefore their importance.

The amount of data produced by the sequencing project was however enormous and the processing procedures highly computational demanding. Such challenges were initially addressed by implementing the workflow through the GenePattern platform on a local server that was made available for the project. Despite powerful, this genomic analysis platform wasn't good enough to implement an high level of parallelization and deal with big amount of data. As a consequence, the pipeline was re-implemented with COSMOS, a library for workflow management that allows formal description of pipelines and partitioning of jobs. In addition to a remarkable time reduction to process sequencing data because of an increase of pipeline performances, that implementation resulted as a perfect solution to explore cloud-based services. In particular, a cloud-based cluster was defined and all the sequenced samples were analyzed accordingly to this solution.

The variant analysis summarized then the filtering and tagging procedure to identify new somatic mutations and confirm known ones. In conclusion, this study showed that mutations in CMML significantly correlate with disease phenotype and an additional important result relies on the definition of a database of mutation related to myelodysplastic/myeloproliferative. Such database will indeed play a central role in any future analyses conducted at the Department of Hematology Oncology of Policlinico San Matteo (Pavia, Italia). Moreover, it was shown that the integration of somatic mutations in current scoring systems significantly improves prognostic stratification of patients and a new clinical/molecular CMML-specific Prognostic Scoring System (CPSS-Mol), that is able to identify 4 risk groups with significantly different survival and risk of leukemic evolution, was defined.

Finally this thesis has presented two related works that were developed during the PhD program. The first one refers to an infrastructure for scalable and cost-effective NGS genotyping in the cloud aimed at reduce the turnaround time of data analysis and costs. The promising results shows that this technology, both in term of sequencing and bioinformatics analysis, is very close to be adopted by clinicians. The second one discusses a kinetic model-based algorithm to classify NGS short reads by their allele origin. Such approach has reached high accuracy in genotyping without making use of a priori knowledge, in contrast to the majority of genotyping software, and can be easily reused to reconstruct the haplotype as well.

Bibliography

[1] Sanger F, Nicklen S, and Coulson AR. Dna sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci USA*, 74(12):5463–5467, Dec 1977.

[2] International Human Genome Sequencing Consortium. Finishing the euchromatic sequence of the human genome. *Nature*, 431:931–945, Oct 2004.

[3] Schadt EE, Turner S, and Kasarskis A. A window into third-generation sequencing. *Hum Mol Genet*, 19(R2):R227–40, Oct 2010.

[4] Meldrum C, Doyle MA, and Tothill RW. Next-generation sequencing for cancer diagnostics: a practical perspective. *Clin Biochem Rev*, 32(4):177–195, Nov 2011.

[5] Wetterstrand KA. Dna sequencing costs: Data from the nhgri genome sequencing program (gsp). Available at www.genome.gov/sequencingcosts, Accessed September 27, 2014.

[6] Cancer Genome Atlas Research Network, Weinstein JN, Collisson EA, Mills GB, Shaw KR, Ozenberger BA, Ellrott K, Shmulevich I, Sander C, and Stuart JM. The cancer genome atlas pan-cancer analysis project. *Nature Genetics*, 45(10):1113–1120, Oct 2013.

[7] Hudson TJ International Cancer Genome Consortium, Anderson W, Artez A, Barker AD, and Bell C et al. International network of cancer genome projects. *Nature*, 464(7291):993–998, Apr 2010.

[8] Welch JS, Westervelt P, Ding L, Larson DE, Klco JM, and Kulkarni S et al. Use of whole-genome sequencing to diagnose a cryptic fusion oncogene. *JAMA*, 305:1577–84, Apr 2011.

[9] Link DC, Schuettpelz LG, Shen D, Wang J, Walter MJ, and Kulkarni S et al. Identification of a novel tp53 cancer susceptibility mutation through whole-genome sequencing of a patient with therapy-related aml. *JAMA*, 305:1568–76, Apr 2011.

[10] Frank R and Hargreaves R. Clinical biomarkers in drug discovery and development. *Nature Reviews Drug Discovery*, 2(7):566–80, July 2003.

[11] Kodama Y, Shumway M, Leinonen R, and International Nucleotide Sequence Database Collaboration. The sequence read archive: explosive growth of sequencing data. *Nucleic Acids Research*, 40:D54–D56, 2012.

[12] Koboldt DC, Ding L, Mardis ER, and Wilson RK. Challenges of sequencing human genomes. *Brief Bioinform*, 11(5):484–98, Sep 2010.

[13] Wandelt S, RheinlÃ¤nder A, Bux M, Thalheim L, Haldemann B, and Leser U. Data management challenges in next generation sequencing. *Datenbank-Spektrum*, 12(3):161–171, Nov 2012.

[14] Nekrutenko A and Taylor J. Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. *Nature Reviews Genetics*, 13:667–672, Sep 2012.

[15] Onsongo G, Erdmann J, and Spears MD et al. Implementation of cloud based next generation sequencing data analysis in a clinical laboratory. *BMC Research Notes*, 7:314, 2014.

[16] Swerdlow SH, Campo E, and Harris NL et al. *WHO Classification of Tumours of Haematopoietic and Lymphoid Tissues.* IARC Press, Lyon, France, 2008.

[17] Vardiman JW, Brunning RD, Arber DA, Le Beau MM, Porwit A, and Tefferi A et al. *Introduction and overview of the classification of the myeloid neoplasms.*, pages 18–30. IARC Press, Lyon, France, 2008.

[18] Cazzola M, Della Porta MG, and Malcovati L. The genetic basis of myelodysplasia and its clinical relevance. *Blood*, 122(25):4021–4034, Dec 2013.

[19] Itzykson R, Kosmider O, Renneville A, and et al. Clonal architecture of chronic myelomonocytic leukemias. *Blood*, 121(12):2186–98, Mar 2013.

[20] Papayannopoulou T and Scadden DT. Stem-cell ecology and stem cells in motion. *Blood*, 111(8):3923–3930, 2008.

[21] Welch JS, Ley TJ, Link DC, Miller CA, Larson DE, and Koboldt DC et al. The origin and evolution of mutations in acute myeloid leukemia. *Cell*, 150(2):264–78, Jul 2012.

[22] Orazi A and Germing U. The myelodysplastic/myeloproliferative neoplasms: myeloproliferative diseases with dysplastic features. *Leukemia*, 22(7):1308–1319, 2008.

[23] Cazzola M, Malcovati L, and Invernizzi R. Myelodysplastic/myeloproliferative neoplasms. *ASH Education Book*, 1:264–272, Dec 2011.

[24] Solary E. Chronic myelomonocytic leukemia (cmml). *Atlas Genet Cytogenet Oncol Haematol.*, 18(1):50–52, 2014.

[25] Cock PJ, Fields CJ, Goto N, Heuer ML, and Rice PM. The sanger fastq file format for sequences with quality scores, and the solexa/illumina fastq variants. *Nucleic Acids Res*, 38(6):1767–71, Apr 2010.

[26] Pearson WR and Lipman DJ. Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A*, 85(8):2444–8, Apr 1988.

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

[28] Andrews S. Fastqc. a quality control tool for high throughput sequence data. Available at http://www.bioinformatics.babraham.ac.uk/projects/fastqc/, 2012.

[29] Burrows M. and Wheeler D. J. *A block-sorting lossless data compression algorithm.* Technical report 124, Digital Equipment Corporation, 1994.

[30] Li H. and Durbin R. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, 25:1754–60, 2009.

[31] Li H. and Durbin R. Fast and accurate long-read alignment with burrows-wheeler transform. *Bioinformatics*, 26:589–595, 2010.

[32] Paolo Ferragina and Giovanni Manzini. Opportunistic data structures with applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 390–398. IEEE, 2000.

[33] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[34] W James Kent, Charles W Sugnet, Terrence S Furey, Krishna M Roskin, Tom H Pringle, Alan M Zahler, and David Haussler. The human genome browser at ucsc. *Genome research*, 12(6):996–1006, 2002.

[35] Daniel Aird, Michael G Ross, Wei-Sheng Chen, Maxwell Danielsson, Timothy Fennell, Carsten Russ, David B Jaffe, Chad Nusbaum, and Andreas Gnirke. Analyzing and minimizing pcr amplification bias in illumina sequencing libraries. *Genome Biol*, 12(2):R18, 2011.

[36] DePristo M.A., Banks E., Poplin R., Garimella K.V., Maguire J.R., Hartl C., and Philippakis A.A. et al. A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature genetics*, 43(5):491–498, 2011.

[37] Sherry S.T., Ward M.H., Kholodov M., Baker J., Phan L., Smigielski E.M., and Sirotkin K. dbsnp: the ncbi database of genetic variation. *Nucleic acids research*, 29(1):308–311, 2001.

[38] Rodrigo Goya, Mark GF Sun, Ryan D Morin, Gillian Leung, Gavin Ha, Kimberley C Wiegand, Janine Senz, Anamaria Crisan, Marco A Marra, Martin Hirst, et al. Snvmix: predicting single nucleotide variants from next-generation sequencing of tumors. *Bioinformatics*, 26(6):730–736, 2010.

[39] Erin D Pleasance, R Keira Cheetham, Philip J Stephens, David J McBride, Sean J Humphray, Chris D Greenman, Ignacio Varela, Meng-Lay Lin, Gonzalo R Ordóñez, Graham R Bignell, et al. A comprehensive catalogue of somatic mutations from a human cancer genome. *Nature*, 463(7278):191–196, 2009.

[40] Rasmus Nielsen, Joshua S Paul, Anders Albrechtsen, and Yun S Song. Genotype and snp calling from next-generation sequencing data. *Nature Reviews Genetics*, 12(6):443–451, 2011.

[41] Kristian Cibulskis, Michael S Lawrence, Scott L Carter, Andrey Sivachenko, David Jaffe, Carrie Sougnez, Stacey Gabriel, Matthew Meyerson, Eric S Lander, and Gad Getz. Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. *Nature biotechnology*, 31(3):213–219, 2013.

[42] Petr Danecek, Adam Auton, Goncalo Abecasis, Cornelis A Albers, Eric Banks, Mark A DePristo, Robert E Handsaker, Gerton Lunter, Gabor T Marth, Stephen T Sherry, et al. The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158, 2011.

[43] Pang AW, MacDonald JR, Pinto D, Wei J, Rafiq MA, Conrad DF, Park H, Hurles ME, Lee C, Venter JC, Kirkness EF, Levy S, Feuk L, and Scherer SW. Towards a comprehensive structural variation map of an individual human genome. *Genome Biol*, 5(11):R52, 2010.

[44] Jarupon Fah Sathirapongsasuti, Hane Lee, Basil AJ Horst, Georg Brunner, Alistair J Cochran, Scott Binder, John Quackenbush, and Stanley F Nelson. Exome sequencing-based copy-number variation and loss of heterozygosity detection: Exomecnv. *Bioinformatics*, 27(19):2648–2654, 2011.

[45] Bruno Zeitouni, Valentina Boeva, Isabelle Janoueix-Lerosey, Sophie Loeillet, Patricia Legoix-Né, Alain Nicolas, Olivier Delattre, and Emmanuel Barillot. Svdetect: a tool to identify genomic structural variations from paired-end and mate-pair sequencing data. *Bioinformatics*, 26(15):1895–1896, 2010.

[46] Kai Wang, Mingyao Li, and Hakon Hakonarson. Annovar: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic acids research*, 38(16):e164–e164, 2010.

[47] Paul Flicek, M Ridwan Amode, Daniel Barrell, Kathryn Beal, Simon Brent, Denise Carvalho-Silva, Peter Clapham, Guy Coates, Susan Fairley, Stephen Fitzgerald, et al. Ensembl 2012. *Nucleic acids research*, page gkr991, 2011.

[48] Kim D Pruitt, Tatiana Tatusova, Garth R Brown, and Donna R Maglott. Ncbi reference sequences (refseq): current status, new features and genome annotation policy. *Nucleic acids research*, 40(D1):D130–D135, 2012.

[49] Pauline A Fujita, Brooke Rhead, Ann S Zweig, Angie S Hinrichs, Donna Karolchik, Melissa S Cline, Mary Goldman, Galt P Barber, Hiram Clawson, Antonio Coelho, et al. The ucsc genome browser database: update 2011. *Nucleic acids research*, page gkq963, 2010.

[50] Xiaoming Liu, Xueqiu Jian, and Eric Boerwinkle. dbnsfp v2. 0: A database of human non-synonymous snvs and their functional predictions and annotations. *Human mutation*, 34(9):E2393–E2402, 2013.

[51] Michael Reich, Ted Liefeld, Joshua Gould, Jim Lerner, Pablo Tamayo, and Jill P Mesirov. Genepattern 2.0. *Nature genetics*, 38(5):500–501, 2006.

[52] Daniel Blankenberg, Gregory Von Kuster, Nathaniel Coraor, Guruprasad Ananda, Ross Lazarus, Mary Mangan, Anton Nekrutenko, and James Taylor. Galaxy: a web-based genome analysis tool for experimentalists. *Current protocols in molecular biology*, pages 19–10, 2010.

[53] Jill P Mesirov. Computer science. accessible reproducible research. *Science (New York, NY)*, 327(5964), 2010.

[54] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[55] Martin Kircher and Janet Kelso. High-throughput dna sequencing–concepts and limitations. *Bioessays*, 32(6):524–536, 2010.

[56] Michael C Schatz and Ben Langmead. The dna data deluge. *Spectrum, IEEE*, 50(7):28–33, 2013.

[57] Aarti N Desai and Abhay Jere. Next-generation sequencing: ready for the clinics? *Clinical genetics*, 81(6):503–510, 2012.

[58] Andrea Sboner, Xinmeng Jasmine Mu, Dov Greenbaum, Raymond K Auerbach, and Mark B Gerstein. The real cost of sequencing: higher than you think! *Genome biology*, 12(8):125, 2011.

[59] Francis S Collins and Margaret A Hamburg. First fda authorization for next-generation sequencer. *New England Journal of Medicine*, 369(25):2369–2371, 2013.

[60] Erik Gafni, Lovelace J Luquette, Alex K Lancaster, Jared B Hawkins, Jae-Yoon Jung, Yassine Souilmi, Dennis P Wall, and Peter J Tonellato. Cosmos: Python library for massively parallel workflows. *Bioinformatics*, page btu385, 2014.

[61] Mohamed Abouelhoda, Shadi A Issa, and Moustafa Ghanem. Tavaxy: Integrating taverna and galaxy workflows with cloud computing support. *BMC bioinformatics*, 13(1):77, 2012.

[62] Konrad J Karczewski, Guy Haskin Fernald, Alicia R Martin, Michael Snyder, Nicholas P Tatonetti, and Joel T Dudley. Stormseq: An open-source, user-friendly pipeline for processing personal genomics data in the cloud. *PloS one*, 9(1):e84860, 2014.

[63] Jeremy Goecks, Anton Nekrutenko, James Taylor, et al. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol*, 11(8):R86, 2010.

[64] Anton Nekrutenko and James Taylor. Next-generation sequencing data interpretation: enhancing reproducibility and accessibility. *Nature Reviews Genetics*, 13(9):667–672, 2012.

[65] Vincent A Fusaro, Prasad Patil, Erik Gafni, Dennis P Wall, and Peter J Tonellato. Biomedical cloud computing with amazon web services. *PLoS computational biology*, 7(8):e1002147, 2011.

[66] Geraldine A Auwera, Mauricio O Carneiro, Christopher Hartl, Ryan Poplin, Guillermo del Angel, Ami Levy-Moonshine, Tadeusz Jordan, Khalid Shakir, David Roazen, Joel Thibault, et al. From fastq data to high-confidence variant calls: The genome analysis toolkit best practices pipeline. *Current Protocols in Bioinformatics*, pages 11–10, 2013.

[67] Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, et al. The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome research*, 20(9):1297–1303, 2010.

[68] Timothy W Yu, Maria H Chahrour, Michael E Coulter, Sarn Jiralerspong, Kazuko Okamura-Ikeda, Bulent Ataman, Klaus Schmitz-Abe, David A Harmin, Mazhar Adli, Athar N Malik, et al. Using whole-exome sequencing to identify inherited causes of autism. *Neuron*, 77(2):259–273, 2013.

[69] Justin M Zook, Brad Chapman, Jason Wang, David Mittelman, Oliver Hofmann, Winston Hide, and Marc Salit. Integrating human sequence data sets provides a resource of benchmark snp and indel genotype calls. *Nature biotechnology*, 2014.

[70] Maria Fischer, Rene Snajder, Stephan Pabinger, Andreas Dander, Anna Schossig, Johannes Zschocke, Zlatko Trajanoski, and Gernot Stocker. Simplex: cloud-enabled pipeline for the comprehensive analysis of exome sequencing data. *PLoS One*, 7(8):e41948, 2012.

[71] Jeffrey G Reid, Andrew Carroll, Narayanan Veeraraghavan, Mahmoud Dahdouli, Andreas Sundquist, Adam English, Matthew Bainbridge, Simon White, William Salerno, Christian Buhay, et al. Launching genomics into the cloud: deployment of mercury, a next generation sequence analysis pipeline. *BMC bioinformatics*, 15(1):30, 2014.

[72] Li R., Li Y., Fang X., Yang H., Wang J., Kristiansen K., and Wang J. Snp detection for massively parallel whole-genome resequencing. *Genome research*, 19(6):1124–1132, 2009.

[73] Kumar S., Banks T.W., and Cloutier S. Snp discovery through next-generation sequencing and its applications. *International journal of plant genomics*, 2012:831460, 2012.

[74] Koboldt D.C., Chen K., Wylie T., Larson D.E., McLellan M.D., Mardis E.R., Weinstock G.M., Wilson R.K., and Ding L. Varscan: variant detection in massively parallel sequencing of individual and pooled samples. *Bioinformatics*, 27(17):2283–2285, 2009.

[75] Li H., Handsaker B., Wysoker A., Fennell T., Ruan J., Homer N., Marth G., Abecasis G., Durbin R., , and 1000 Genome Project Data Processing Subgroup. The sequence alignment/map (sam) format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[76] The 1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467:1061–1073, 2010.

[77] Challis D., Yu J., Evani U.S., Jackson A.R., Paithankar S., Coarfa C., Milosavljevic A., Gibbs R.A., and Yu F. An integrative variant analysis suite for whole exome next-generation sequencing data. *BMC bioinformatics*, 13:8, 2012.

[78] Garrison E. and Marth G. Haplotype-based variant detection from short-read sequencing. *ArXiv e-prints*, 1207:3907, 2012.

[79] Marth G.T., Korf I., Yandell M.D., Yeh R.T., Gu Z. Zakeri H., Stitziel N.O., Hillier L., Kwok P.Y., and Gish W.R. A general approach to single-nucleotide polymorphism discovery. *Nature genetics*, 23(4):452–456, 1999.

[80] Branton D., Deamer D.W., Marziali A., Bayley H., Benner S.A., Butler T., Di Ventra M., Garaj S., Hibbs A., and Huang X. et al. The potential and challenges of nanopore sequencing. *Nature biotechnology*, 26(10):1146–1153, 2008.

[81] Louis N Hand and Janet D Finch. *Analytical mechanics.* Cambridge University Press, 1998.

[82] Greenberg MJ. *Euclidean and non-Euclidean geometries.* W.H. Freeman, 2008.

[83] Blumenthal LM. *Theory and applications of distance geometry.* Chelsea Pub, 1970.

[84] Stephen Boyd and Lieven Vandenberghe. *Convex optimization.* Cambridge university press, 2009.

[85] Olivier Delaneau, Jonathan Marchini, and Jean-François Zagury. A linear complexity phasing method for thousands of genomes. *Nature methods*, 9(2):179–181, 2012.

[86] Verena Heinrich, Tom Kamphans, Jens Stange, Dmitri Parkhomchuk, Jochen Hecht, Thorsten Dickhaus, Peter N Robinson, and Peter M Krawitz. Estimating exome genotyping accuracy by comparing to data from large scale sequencing projects. *Genome Med*, 5:69, 2013.

[87] Jorge Duitama, Pramod K Srivastava, and Ion I Măndoiu. Towards accurate detection and genotyping of expressed variants from whole transcriptome sequencing data. *BMC genomics*, 13(Suppl 2):S6, 2012.

[88] Sharon R Browning. Multilocus association mapping using variable-length markov chains. *The American Journal of Human Genetics*, 78(6):903–913, 2006.

[89] Bryan Howie, Jonathan Marchini, and Matthew Stephens. Genotype imputation with thousands of genomes. *G3: Genes, Genomes, Genetics*, 1(6):457–470, 2011.

[90] Vikas Bansal and Vineet Bafna. Hapcut: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics*, 24(16):i153–i159, 2008.

[91] Derek Aguiar and Sorin Istrail. Hapcompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. *Journal of Computational Biology*, 19(6):577–590, 2012.

[92] Esperanza Such, Ulrich Germing, Luca Malcovati, José Cervera, Andrea Kuendgen, Matteo G Della Porta, Benet Nomdedeu, Leonor Arenillas, Elisa Luño, Blanca Xicoy, et al. Development and validation of a prognostic scoring system for patients with chronic myelomonocytic leukemia. *Blood*, 121(15):3005–3015, 2013.

[93] Johan T Den Dunnen, Stylianos E Antonarakis, et al. Mutation nomenclature extensions and suggestions to describe complex mutations: a discussion. *Human mutation*, 15(1):7–12, 2000.

[94] Minoru Kanehisa and Susumu Goto. Kegg: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, 28(1):27–30, 2000.

[95] Aron Marchler-Bauer, Shennan Lu, John B Anderson, Farideh Chitsaz, Myra K Derbyshire, Carol DeWeese-Scott, Jessica H Fong, Lewis Y Geer, Renata C Geer, Noreen R Gonzales, et al. Cdd: a conserved domain database for the functional annotation of proteins. *Nucleic acids research*, 39(suppl 1):D225–D229, 2011.

[96] Yongwook Choi, Gregory E Sims, Sean Murphy, Jason R Miller, and Agnes P Chan. Predicting the functional effect of amino acid substitutions and indels. *PloS one*, 7(10):e46688, 2012.