Computational analysis of germline and somatic mutation spectra

An interactive web application

Øyvind Sørby

Master’s Thesis
Informatics: Programming and networks
60 credits

Department of Informatics
The Faculty of Mathematics and Natural Sciences
Computational analysis of germline and somatic mutation spectra

Øyvind Sørby

May 2017
Abstract

The speed and cost of genome sequencing have rapidly decreased during recent years owing to the development of several new technologies commonly known as the next-generation or high-throughput sequencing. DNA sequencing is currently a vital part in biomedical research, both when it comes to cancer research and research on rare genetic disorders. Comprehensive catalogs of DNA sequence variants in healthy individuals have been established, in addition to DNA aberrations found in tumors of cancer patients. An intriguing observation when considering public databases of germline and somatic DNA variation is the significant level of overlap, i.e. the numerous nucleotide variants that are recorded both as somatic and as germline. However, a limited number of studies have quantified the characteristic properties of coinciding DNA variation, for instance their particular distribution in relation to DNA variant types and DNA sequence context, allelic frequency of germline variants, and tumor types of somatic variants. In this project, we have gathered comprehensive datasets of single nucleotide variants in the human genome that are of both germline and somatic origin. Moreover, through various subgroupings of the somatic and germline variant datasets (e.g. by cancer type or germline allele frequency), we have quantified key mutation attributes for the coinciding set, and similarly for the non-coinciding sets that occur strictly within the germline and soma, respectively. These statistics have formed the basis for an interactive web application that permits several different analyses of coinciding DNA variation (http://coinciding-snv-analyzer.no). We demonstrate through various use cases how the application can shed light on the nature and possible mechanisms underlying coinciding DNA variation.
Acknowledgements

It was long journey with many rough paths, steep climbs and dead ends. Foremost I would like to thank my supervisor Sigve Nakken for his guidance, support and patience which helped me through the obstacles. I’d also like to thank the people at the 4th floor. At last I am deeply grateful for support of my family, especially my brother and Børre.
List of figures

Figure 1.1 Intersection of germline and somatic variant databases 3
Figure 2.1 Chromosome, DNA double helix, and nucleotide base pairs 5
Figure 2.2 Base pairs, Guanine, adenine, purines, cytosine and thymine 6
Figure 2.3 DNA sequence, RNA sequence and protein sequence 7
Figure 2.4 Structure of an average human gene 8
Figure 2.5 DNA replication error 15
Figure 2.6 mutations resulting in same variant type 17
Figure 2.7 variant type incorporated with the bases immediately 5’ and 3’ to the mutated base 18
Figure 3.1 Pairing of somatic and germline databases 23
Figure 3.2 Categorisation, unique somatic, unique germline, and coinciding 24
Figure 3.3 snapshot of the first few lines of a VCF file 26
Figure 3.4 consequence types that could be added to a VCF variant record 26
Figure 3.5 variants records of VCF file, annotation with vcfanno configuration file 26
Figure 3.6 Example, CyVCF 27
Figure 3.7 Input values (ui.R) => R code (server.R) => output values (ui.R) 28
Figure 3.8 Preprocessing of files 30
Figure 3.9 Perl script¹ (VEP) 31
Figure 3.10 VCF file and VEP annotation 31
Figure 3.11 The annotation process is run both ways 32
Figure 3.12 First pass: annotations from ExAC.vcf is added to... 32
Figure 3.13 Second pass: annotations from ICGC.vcf is added to... 33
Figure 3.14 snapshot of VCF: ICGC annotated with ExAC 33
Figure 3.15 example of variant record 34
Figure 3.16 snapshot of Python that defines variant types 35

¹ More information available at their web site (http://www.ensembl.org/info/docs/tools/vep/script/index.html).
Figure 3.17 snapshot of Python code that decides which variant type a record belongs to
Figure 3.18 variant context type
Figure 3.19 variants chromosome number and position
Figure 3.20 retrieve variant type
Figure 3.21 retrieve the consequence type
Figure 3.22 Only first consequence types retrieved
Figure 3.23 Different types of variants consequences and their position
Figure 3.24 COSMIC_CANCER_TYPE_GW
Figure 3.25 Recurrent/nonrecurrent
Figure 3.26 Parsed text, example 1
Figure 3.27 Parsed text, example 2
Figure 3.28 Minor Allele Frequencies, common, low frequency, rare, very rare
Figure 3.29 dbSNP Submission frequency
Figure 3.30 data table annotation
Figure 3.31 adding the occurrence to table
Figure 3.32 snapshot of CSV file
Figure 3.33 loading of datatable, COSMIC vs 1000Genomes
Figure 3.34 user input, ui.R
Figure 3.35 plotOutput to tab panel
Figure 3.36 renderPlot, ggplot object

Figure 4.1 Sidebar panel, main panel
Figure 4.2 Sidebar panel, detailed
Figure 4.3 Main panel analysis tab - overall statistics
Figure 4.4 Main panel analysis tab - Variant types
Figure 4.5 Main panel analysis tab - Sequence context
Figure 4.6 Main panel analysis tab: Variant consequence
Figure 4.7 Main panel analysis tab: Coinciding variants
List of tables

Table 2.1 Number of variants for each database 19
Table 3.1 Categorization of variant consequence types, coding or noncoding 38
Table 3.2 List of 21 cancer types in alphabetic order 40
Table 3.3 Amount of variant records quantified, comparison 45
Table 4.1 Coinciding variants: allele frequencies 62
Table 4.2 Coinciding variants: allele frequencies and cancer types 63
Table 4.3 Distribution of recurrent and nonrecurrent coinciding... 69
1 Introduction

2 Background
   2.1 DNA
   2.2 Genes and proteins
   2.3 The human genome
      2.3.1 Genome sequencing
   2.4 Germ cells and somatic cells
   2.5 Mutations and DNA variation
      2.5.1 Types of mutations - tissue of origin
      2.5.2 Types of mutations - effect on sequence
         2.5.2.1 Single nucleotide variants
      2.5.3 Types of mutations - effect on function
   2.6 Mutational spectra
      2.6.1 Variant types
      2.6.2 Variant sequence context
      2.6.3 Mutational signatures
   2.7 DNA variation databases
      2.7.1 The 1000 Genomes Project
      2.7.2 The Exome Aggregation Consortium (ExAC)
      2.7.3 The Database of Short Genetic Variations (dbSNP)
      2.7.4 The catalogue of somatic mutations in cancer (COSMIC)
      2.7.5 The International cancer Genomics Consortium (ICGC)

3 Methods & Implementation
   3.1 Introduction to the web application
   3.2 VCF
   3.3 Software libraries and computational frameworks
      3.3.1 VEP
      3.3.2 vcfanno
      3.3.3 Python libraries
         3.3.3.1 Parsing the VCFs variant records
      3.3.4 R and R packages
         3.3.4.1 deconstructSigs
         3.3.4.2 ggplot2
      3.3.5 Shiny framework
   3.4 Data preparation: Preprocessing and Quantification of variant attributes
      3.4.1 Step one – Determination of variant consequence/effect
      3.4.2 Step two – Determination of coinciding versus unique variants
3.4.3 Step three – Quantification of variant properties

3.4.3.1 Variant type
3.4.3.2 Variant context type
3.4.3.3 Consequence type
3.4.3.4 Variant consequence
3.4.3.5 Cancer type and Variant frequency across samples
3.4.3.6 Population and Minor allele frequency
3.4.3.7 dbSNP Submission frequency
3.4.3.8 dbSNP Validation
3.4.3.9 Datatable function
3.4.3.10 Building .CSV files

3.4.4 Variants - requirements for inclusion in quantification

3.5 Application implementation

3.5.1 Application layout
3.5.2 Sidebar panel
3.5.3 Charts, statistics and data tables

4 Results

4.1 Application layout
4.1.1 Sidebar panel
4.1.2 Main panel
4.1.2.1 Overall statistics
4.1.2.2 Variant types
4.1.2.3 Sequence context
4.1.2.4 Variant consequences
4.1.2.5 Coinciding variants
4.1.2.6 Mutational signatures

4.2 Case examples of functionality
4.2.1 Coinciding variants - coding versus non-coding
4.2.2 Coinciding variants - relation to germline allele frequency
4.2.3 Coinciding variants - mutation spectra versus germline allele frequency
4.2.6 Coinciding variants - DNA sequence context
4.2.7 Coinciding variants: recurrent versus nonrecurrent
4.2.8 Mutational signatures within coinciding variants

5 Discussion and Further work

5.1 Data preparation
5.2 Application implementation
5.3 Dataset limitations
5.4 Reproducibility
5.5 Case examples
5.6 Further work

6 Bibliography
1 Introduction

DNA, or deoxyribonucleic acid, is the hereditary material in humans and almost all other organisms. The information in DNA is stored as a code made up of four chemical bases: adenine (A), guanine (G), cytosine (C), and thymine (T). DNA sequencing refers to the process of determining the order of the four nitrogenous bases in a DNA molecule. Through DNA sequencing, it is possible to determine the sequence of protein-coding genes, chromosomes, and even entire genomes of individuals. The first complete genome of an organism was sequenced in 1977 (Sanger et al. 1977), it was a virus with a 5386 bases long DNA sequence. The human genome contains more than 3.2 billion basepairs. A major effort to sequence the entire DNA sequence of the human species was initiated by The Human Genome Project in 1990 and the first draft was published by on February 12, 2001 (Lander et al. 2001a). The speed and cost of genome sequencing have rapidly decreased during recent years owing to the development of several new technologies commonly known as the next-generation or high-throughput sequencing. Pending on the available computational resources, it takes now only days to weeks to sequence a complete human genome (Goodwin et al. 2016). DNA sequencing is currently a vital part in biomedical research, both when it comes to cancer research and research on rare genetic disorders. Importantly, the advances in DNA sequencing technologies have led to the discovery of large amounts genetic variation in the human population (1000 Genomes Project Consortium et al. 2012). Comprehensive catalogs of DNA sequence variants in healthy individuals have been established, in addition to DNA aberrations found in tumors of cancer patients (Forbes et al. 2011). Today, ongoing research projects around the world are continuously detecting novel sequence variants in human DNA samples, and registering them in international variant databases such as the database of short genetic variants (dbSNP) and the catalogue of somatic mutations in cancer (COSMIC).

The collection of human DNA variants can be divided in two major classes pending on their tissue of origin. Mutations that occur in somatic cells are not passed on to progeny, but carry the potential of inducing tumor development (Griffiths et al. 2007). Germline mutations refer to those mutations that can be passed on to progeny, specifically mutations that occur in cells
that take part in sexual reproduction, e.g. male and female gametes. Some germline variants
are causing rare genetic disorders, whereas the vast majority is present in human populations
at different frequency levels and with unexplored associations to phenotypes (Karki et al. 2015).

Current DNA sequencing regimes for the detection of somatic and germline variation
are generally based on the same principles and underlying technologies. Importantly also,
considering that any somatic tissue of a given individual carries the germline background of
the same individual, separation of acquired somatic mutations from existing germline variants
is a challenging and error-prone endeavour. Henceforth, when considering data deposited in
public databases of somatic and germline DNA variation, it is likely that the observed
variants will be subject to different levels of noise. This issue is however further complicated
by the fact that underlying biological mechanisms of mutation are shared across cells of
germline and somatic origin (Ivanov et al. 2011a). In summary, few attempts have so far been
made to compare the nature, location, and relative frequency of germline and somatic
mutations, in particular sites of coinciding DNA variation.

The aim of this Master’s thesis project is to characterize sites of coinciding DNA variation,
that is genomic loci in which the exact same variant has both been observed across
individuals (i.e. as a germline variant) and detected as an acquired somatic mutation in one or
several tumor samples. Do such sites of coinciding variation reflect mutational hotspots in
DNA? Or are the extracted sites of coinciding variants from database searches not truly
reflecting sites of coinciding variation in the germline and the soma due to impact of false
positive entries in either of the two contributing entities? Recent studies suggest that issues
such as DNA oxidation and DNA contamination in sequencing studies may have induced a
significant proportion of noise in public DNA variation databases (Chen et al. 2017; Sinha et
al. 2017). Considering these findings, it may be that the the patterns of coinciding variants
will depend upon germline variant frequency and/or specific tumor types. In order to
facilitate data exploration in the context of these questions, we have undertaken an extensive
computational analysis of germline and somatic mutation spectra, and made the results
accessible by means of an interactive web application. The application highlights
characteristic patterns of coinciding DNA variation through a comparison with the patterns
that can be found for variants that are unique to the germline or soma (see Figure 1.1). Specifically, I have quantified properties (variant consequences, variant effects, mutation types and sequence contexts) for huge sets of DNA variation data retrieved from several international sequencing consortia, including the Exome Aggregation Consortium (ExAC), 1000 Genomes Project (1000Genomes), database of short genetic variants (dbSNP), the International Cancer Genomics Consortium (ICGC) and Catalogue of Somatic Mutations in Cancer (COSMIC). An overall aim of my thesis is thus to highlight characteristic patterns in the set of coinciding DNA variation, and further to make the analyses available to other researchers through an interactive web application.

Figure 1.1: Intersection of germline and somatic variant databases reveal sites of coinciding DNA variants as well as variants unique to the soma and the germline.
2 Background

In this chapter, we will describe fundamental aspects of genetics and mutations that are tightly related to the issue addressed in this project. Initially, we will describe DNA, and its building blocks. Next, we will describe genes, which are made up of DNA and form instructions to make protein molecules. At last we will describe the resources which provide the data that is needed for the project. This chapter should contain sufficient background knowledge for an understanding of the main results presented in the thesis.

2.1 DNA

The molecule known as DNA (deoxyribonucleic acid) is the hereditary material in almost all organisms. An organism is made of one or more biological units called cells, these cells provide the organism structure, maintain necessary functions and can make copies of itself. Animals, plants and other multicellular higher organisms that belong to the taxon eukaryota have all their DNA structured into chromosomes and stored inside the cell nucleus.

Figure 2.1: Chromosome, DNA double helix, and nucleotide base pairs (National Institutes of Health)

The physical structure of the DNA molecule consists of two strands of individual subunits called nucleotides and together they are intertwined into a double helix (illustrated in Figure
Each nucleotide is composed of three components, a sugar, a phosphate group, and one of four possible nitrogenous bases; adenine (A), guanine (G), cytosine (C), and thymine (T). A phosphate group of one nucleotide binds to the sugar of the neighbouring nucleotide and thereby makes up the backbone of each strand. Based on the nitrogen base structure (Figure 2.2), guanine and adenine are collectively called purines (two rings) and cytosine and thymine are called pyrimidines (one ring). The two strands of the DNA are reverse complementary, meaning they have complementary base pairing and run in opposite directions of each other. A basepair is a unit consisting of two nucleotides in opposing strands that are bound together by hydrogen bonds. Adenine can only bond with thymine, and guanine can only bond with cytosine. Hence the DNA double helix strands consist of complementary nitrogenous base pairs that are either A:T or G:C.

![Figure 2.2: Guanine and adenine are purines (two ring structure) and cytosine and thymine are pyrimidines (one ring structure) (Lents 2009)](image)

An important property of DNA is that it can replicate itself. Cells have a limited lifespan, so in order for an organism to stay alive, cells have to pass on the DNA onto new cells. This is done through a cell division known as mitosis (which cells do to replace old cells and grow tissue/organs) where a cell duplicates all of its content, including all of the chromosomal DNA and splits to form two identical daughter cells. Each daughter cell have the exact same DNA present in the old cell. This is done by taking advantage of the DNA double helix structure, where each strand is used as a pattern for duplicating the sequence of nucleotide bases (Genetics Home Reference [GHR], 2017).
2.2 Genes and proteins

The information in DNA lies along sequences of nucleotides and there are specific stretches known as *gene sequences*. These sequences can be used by the cell to build protein which are large and complex molecules that play many critical roles in the body. In order to function correctly, each cell depends on a variety of proteins to do their jobs correctly. Proteins are essential for physiological processes, structure and regulation in cells and tissue (GHR, 2017a). They function as antibodies, enzymes, messengers, structural components and transporters of molecules. A gene can be defined as basic physical and functional unit of heredity (GHR, 2017b). Since each gene correspond to a specific sequence of nucleotides it is common to consider the whole DNA molecule as a book full of recipes for proteins that life depend on.

![DNA sequence, RNA sequence and protein sequence diagram]

*Figure 2.3: DNA sequence, RNA sequence and protein sequence.*

The process of building/synthesizing a protein from a gene sequence is known as *gene expression*. It has two stages, *transcription* and *translation* (*Figure 2.3*). In the first stage, a gene sequence is copied from the DNA molecule and to a similar molecule, which also consist of a chain of nucleotides, called RNA (ribonucleic acid). If this new RNA sequence molecule is going to be used for synthesizing a full protein, it is known as messenger RNA (mRNA). In contrast to DNA, the RNA nucleotide consist only of one strand of nucleotides where the base thymine (T) is replaced by uracil (U). After transcription, the RNA is brought out of the cell nucleus for the second stage, which is translation. During this stage the RNA is used in a process to build a protein. But before the process can begin it may be necessary
with variable splicing (in eukaryotes). The gene sequence in both RNA and DNA have parts called intron and exon and only the latter is used for encoding protein (Figure 2.4). By variable splicing, depending on the protein that is going to be built, introns are removed and exons are merged into a new sequence. Not all exons have to be used and they can be put together differently, so in this way, many slightly different proteins can be synthesized from the same gene sequence. After a possible splicing is done, the enzyme ribosome (also a protein) starts reading the new forward strand. While reading, the ribosome translate codons along the coding sequence and join together amino acids by adding peptide links between them. During both stages, DNA and RNA are always read 5' to 3' direction.

A codon is a sequence that consist of three nucleotides from a selection of the four bases A, G, C and U. Each codon represents a particular type of amino acid. For example AAG code for the amino acid Lysine. So a triplet of three nucleotides could code for up to 64 amino acids but there are only 20 different amino acids that will fit a given codon, therefore several codons may represent the same amino acid. The linear sequence of codons is called the reading frame. The coding sequences in DNA and RNA that belong to the same gene are virtually identical and known as the base sequence.

The part of the gene sequence that code for protein comprise reading frames (ORFs) consisting of a series of codons that specify the amino acid sequence. More specific, the ribosome construct a polypeptide by start beginning at a start codon (AUG) and reads codons one after another until it reach at a termination/stop codon (UAA/UAG/UGA). The DNA sequence determines the protein sequence and the protein sequence determines the protein structure. A protein structure can also consist of several protein sequences (polypeptides). In the last step, the protein is folded in a specific way which determines its properties and function. The DNA gene sequence determines protein sequence which determines the protein
The flow of genetic information from DNA to RNA is referred to as the central dogma of molecular biology (Brown 2002a)

In addition to gene-coding sequences, with exonic and intronic regions, the DNA also consists of non-coding regions (not coding for proteins) such as intergenic regions and regulatory regions. The intergenic regions are the regions between the gene sequences. Regulatory regions regulate the expression of genes, such as untranslated region (UTR), promoters, enhancers/silencers (La Trobe Institute for Molecular Science, Melbourne, Australia et al. 2017). Untranslated regions are transcribed to RNA but not translated to protein and are attached at the start and at the end of gene sequences. The UTR and gene are together transcribed as one sequence to RNA. It has two sites, 5'UTR and 3'UTR and the former have a sequence that allows the protein ribosome to bind and initiate translation. 3'UTR on the other hand, has a role in translation termination and is placed just behind the termination codon.

2.3 The human genome

A genome is the complete set of DNA sequence of an organism and contains all of the biological information needed for an organism to build and maintain itself (Brown 2002c). Inside the cell nucleus of almost every cell of the human body lies the genome distributed over 23 pairs of chromosomes where each chromosome is a linear molecule containing a very long sequence of nucleotides. One of these chromosome pairs are the sex chromosomes and defines the gender of the individual, XX for females and XY for males. Non sex chromosomes are known as autosomes. Most human cell types are diploid which means they have two copies of each chromosome (i.e. 46 chromosomes in total). Haploid cells, on the other hand, have one copy of each (23 chromosomes). In diploid cells, each of the gene copies are inherited from either father or mother. Such a pair of genes copies could either have two identical genes (homozygous) or two versions of the same gene (heterozygous). Different versions of the same gene are also known as alleles.

2.3.1 Genome sequencing

Genome sequencing is a method used to determine the exact nucleotide sequence of the genome. In 2001, the first draft sequence of a human genome sequence was published by the Human Genome Project (Lander et al. 2001b). Through next-generation DNA sequencing of a given individual, it is possible to determine genetic variants in their genome by comparing their DNA sequence to a reference genome (i.e. human genome assembly GRCh38, as mentioned above); DNA sequencing of the entire genome sequence of an individual is known as whole genome sequencing (WGS), which is a type of process where the complete set of and the total length of a person's DNA is scanned and aligned to a reference genome, including both coding gene regions as well as noncoding, intergenic sequences. This is done in order to discover novel and known genetic variants present in an individual/personal genome. In many instances it is most relevant to only sequence the gene-coding part of the genome, since these are directly related to the protein products that are encoded in the genome. When only the nucleotide sequences that contain genes are sequenced it is referred to as whole exome sequencing (WES). Since WES is both cheaper and a less time consuming process than WGS (approx 1.5% of the full human genome is sequenced in WES), it is also usually done at a higher coverage, which is the number of times a nucleotide sequence is read and aligned to a reference sequence (Rabbani et al. 2014).

2.4 Germ cells and somatic cells

In the human body one may distinguish between two major cell types: somatic cells and germ cells. Somatic cells are diploid, each of these cells contain two copies of the genome and together they make up the vast majority of cells in the body. These can be seen as normal body cells and function as building blocks (skin, hair, and muscle) and utilize the set of genes to produce necessary proteins. Somatic cells are duplicated by mitosis (explained earlier) which is the most common form of cell division. The DNA of somatic cells last only during the lifespan of the individual, in germ cells however, it can be passed on to the next generation. Germ cells are dedicated for sexual reproduction and undergo meiosis, which is a specialized type of cell division that produces gametes, also known as egg and sperm cells.
Gametes are haploid and contain (in additional to one of each autosome) only one of the sex chromosomes, either X or Y. As explained earlier, during meiosis, the chromosomes undergo genetic recombination which means each of the gametes end up with different combinations of parent DNA. The human species multiply by sexual reproduction and during fertilization, the gamete of each parent are fused into a single cell known as zygote. Further, the zygote multiply and evolves into an embryo with a vast variety of cells turning out to be a new individual with an unique set of DNA. In the early stages of the growing embryo, most of these cells are set to be somatic but a few become germ cells. Female germ cells produces a limited amount of eggs during this early embryo stage while males have their sperm cells produced during their whole lifetime. Hence the inherited genetic material present in germline cells (i.e. egg and sperm cells) are passed on to be the genomes of new generations. Moreover, it is only through the germline that for example a genetic disease can be inherited by an offspring.

2.5 Mutations and DNA variation

DNA molecules change over time by the result of sequence alterations, commonly known as mutations. A mutation is a permanent change in the DNA sequence. These changes could vary from small-scale sequence alterations to larger bits of DNA. DNA mutations may be caused errors during DNA replication or by other types of DNA damage, such as chemicals or radiation (Helleday et al. 2014). During the course of evolution, the process of DNA mutation has resulted in a number of genetic differences between human genomes, which we collectively refer to as DNA variation. In the following sections, we will outline different classifications of DNA variants, from their tissue-of-origin to their impact on the DNA sequence and how they could influence gene and protein function. A particular focus will be put on single nucleotide variants (SNVs), which have been the mutation class subject to analysis of coinciding DNA variation.

2.5.1 Types of mutations - tissue of origin

Germline mutations, commonly referred to as germline DNA variation, arises from mutational events in germ cells (egg and sperm) that can be transmitted between parent and
offspring, and that is present in virtually every cell. *Somatic mutations*, also known as somatic DNA variation, refers on the other hand to acquired mutations in tissues that is not passed on to the offspring. Some somatic mutations can be associated with growth advantage upon affected cells and tissues, and germline gene mutations can come to attention by causing inherited disease and disadvantage upon individuals and their offspring. Whereas inherited disease usually include one or two pathological mutations at specific locus, cancer is often characterized by multiple somatic mutations distributed genome-wide (Ivanov et al. 2011b). Genetic diseases caused by somatic cells can only cause damage to the individual and would not go to the next generation. Cancer is a common somatic disease and is driven by somatic mutations, also known as *acquired mutations*, which occur in and throughout the life cycle of somatic cells. These mutations accumulate over time because DNA of somatic cells is regularly damaged by mutagens of external and internal origins (Stratton et al. 2009). Most of the damage is repaired by DNA but some mutations avoid proofreading and the DNA replication itself has a small error rate (Ivanov et al. 2011b). As a consequence, every cell of the body turn out to have a slightly different genome sequence and as opposed to germline mutations, somatic cell mutations are only relevant for the individual and last only over one lifespan.

Most cancers arise as a result of changes caused by acquired somatic mutations in the DNA of cells, each cancer is also the outcome of Darwinian evolution occurring among cell populations within the microenvironments provided by the tissue of an multicellular organism (Stratton et al. 2009). During evolution among cells, the selection mechanism diminish cells that have acquired deleterious mutations and bring up cells that have the capability to multiply and survive more effectively than other cells. Sometimes, at rare occasions within the body, a single cell manages to acquire the right set of suitable advantageous mutations that allows it to proliferate autonomously, invade tissue and spread to other parts of the body (metastasize). Somatic mutations in cancer cells can be classified into two types after the their consequence for cancer development. *Driver mutations* confer to a growth advantage on the cell in which it occur and have been positively selected in the tissue in which the cancer arises. By contrast, *passenger mutations* have not been selected, have not conferred to growth advantage and happened to be present when the cell when it
acquired its first driver mutations (Stratton et al. 2009). There are mechanisms to stop a harmful cell from developing, such as apoptosis, where the cell undergo programmed cell death, and the immune system which step in and destroy abnormal looking cells. However, during the evolution of cancer, mutations may knock out the cells capability to undergo apoptosis and mutations may even manage to make the cell evade destruction by the immune system (Hanahan and Weinberg 2011).

There is an important distinction between germline variants and somatic variants in relation to the process of DNA sequencing and calling of variation. In order to detect somatic variants in a given individual, it is common practice to sequence both a DNA sample from the tissue of interest and a control sample (e.g. blood) from the same individual. This is followed by a variant detection step that identifies DNA differences in the sample sequences through a comparison/alignment with the reference genome. Finally, to detect acquired or somatic variants, the variants in the control sample is subtracted from the variants found in the somatic tissue. The last step will thus exclude germline variants from the set of acquired mutations. In many instances however, one is trying to detect somatic mutations by only considering the DNA sequence of the tumor/somatic sample. In this setting, the variants found through a comparison with the reference genome will constitute a mix of somatic and germline variants. In an attempt to enrich for somatic events, it is a common practice to intersect the initial call sets with public germline variant databases in order to remove previously described variants in the human population. A problem with this approach is that public SNP databases have over the years included somatic mutations that may be cancer related (most prominently dbSNP, 1000 Genomes and ExAC likely to much lesser extent). Such indiscriminate filtering can thus remove real somatic variation that may have a relevance for underlying disease biology (Jung et al. 2013). In conclusion, since filtering processes for the detection of somatic/cancer mutations is suboptimal in many cases, there is likely to be a significant number of registered variants that have been mislabeled (i.e. somatic instead of germline and vica versa).
2.5.2 Types of mutations - effect on sequence

There are different types of small scale mutations that occur on the DNA sequence, some of these types of mutations are substitution, insertion, deletion, inversion, duplication and repetition. A substitution is a single base-pair substitution, it occurs when one of the nucleotides is replaced by another nucleotide. Insertion is when one or a few nucleotides are inserted and deletion is when one or a few nucleotides are removed. Insertions and deletions are also commonly referred to as “indels”. When a sequence is reversed end to end, it is known as inversion. Duplication happen when one or more nucleotides are duplicated and repetition is the occurrence of a copied sequence that repeats itself throughout the DNA sequence, also known as repetitive DNA. Single nucleotide substitutions, point mutations and single nucleotide variants (SNVs) are different terms that all refer to the same, a single base substitution in DNA.

2.5.2.1 Single nucleotide variants

A spontaneous point mutation could occur during DNA replication (L A Loeb and Kunkel 1982)). For example a nucleotide of a DNA strand could be substituted for another, resulting in a mismatch/mispair between the two DNA strands since one base pair consist of wrong complementary bases. Usually a DNA repair mechanism called proofreading manage to remove such errors, but when it fails to do so, it will stay there. So if a mismatch is not fixed, the DNA of the resulting parents daughter cells (after next cell division) will end up having different sequences where one cell have the mutation as a permanent part of its DNA (Brown 2002b). Therefore, when a mutation occur in DNA it is important to consider the location and the reverse complement of a nucleotide sequence. A point mutation could also arise when a mutagen reacts with DNA and cause a structural change that alter the base pairing capability of an nucleotide and cause it to mismatch. The consequence of such a structural change, after the next DNA replication, could be that the altered base (as a base analogue) ends up base pairing with a different base than its parent (Figure 2.5).
As mentioned before, the four different nucleotide bases of DNA can be divided into *pyrimidines* (cytosine and thymine) and *purines* (guanine and adenine). Based on biochemical properties, nucleotide substitutions can be grouped into two types: *transversion* when a pyrimidine is substituted for a purine or vice versa, and *transition* when a pyrimidine is substituted for another pyrimidine or purine for another purine. Analyses of human gene mutations associated with disease indicate that the most common nucleotide substitutions are C>T:G>A and A>G:T>C (Krawczak et al. 1998). In other words, transitions seem to be much more common than transversions.

### 2.5.3 Types of mutations - effect on function

If a mutation occurs in a gene-coding sequence or in a regulatory region it could affect the function of one or more proteins. A mutation that causes an alteration in a gene sequence is known as a gene mutation. Such a mutation in could for instance lead to a change in the proper code for gene codon, its corresponding amino acid, and with a potential to alter the shape and function of a protein that plays a critical role in human cells. In regulatory regions, such mutations could for example lead to increase or decrease of the transcriptional activity of a gene. Mutations can interfere with any stage in the pathway of expression from gene activation to synthesis and secretion of the mate protein product (Antonarakis and Cooper 2010). Point mutations that occur within the protein-coding region of a gene may be classified into three major kinds, depending upon what the erroneous codon codes for:
Silent/synonymous mutation
A substitution that alter a base pair but does not lead to a change of the amino acid for another (i.e. the new codon encodes the same amino acid). In other words, if a point mutation in a gene sequence does not affect the protein significantly it is often termed synonymous mutation or silent mutation.

Missense/nonsynonymous mutation
A substitution that do result in a change of one amino acid for another. The missense mutation, also known as nonsynonymous mutation, can be further categorized into two types: conservative mutation, an amino acid changed for another with similar biochemical properties (transition). Nonconservative mutation, an amino acid changed for another with different biochemical properties (transversion).

Nonsense/stop-gained mutation
A substitution that causes the altered codon to prematurely signal the cell to stop building protein. This happens when a codon is altered to a termination codon. A nonsense mutation is also known as a stop-gained mutation and such a mutation leads to a shortened protein that may function improperly or not at all.

2.6 Mutational spectra
The distributions of single nucleotide substitutions observed among human germline variants and somatic mutations can be referred to as mutational spectra. A range of mechanisms underlie the rates and spectra of germline and somatic mutations, and they are thus responsible for the great variability in the types of variants generated in different settings. In this section, I will describe key concepts related to mutation analysis of single nucleotide substitutions, specifically the categories of nucleotide variants and the notion of DNA sequence neighbourhood or context (taking into account the reverse complementary nature of DNA), as well as mutational signatures, which represents a key framework for deciphering the underlying mutational processes in different tumor types.
2.6.1 Variant types

There is a limited number of single nucleotide mutations that could occur at position in a DNA sequence. Since there are four different nitrogenous bases and each of them could be replaced by the other, it means that there are 3*4 different types of single nucleotide variants. Variant types are represented by the following six groups of substitution: A>G:T>C, C>T:G>A, C>A:G>T, C>G:G>C, A>T:T>A and A>C:T>G. Each group represents two base pair mutations that are “equal” in that sense they are complementary to each other. For instance, a A>G mutation on the forward strand would have been recorded as a T>C mutation on the complementary strand, they are both grouped into the A>G:T>C category (Figure 2.6).

![Figure 2.6: Both mutations occur at the same location and results in the same variant type.](image)

2.6.2 Variant sequence context

An important property of nucleotide substitutions is their genomic DNA sequence context, which denotes the immediate sequence upstream (5’) and downstream (3’) of the site of mutation/variation. (Figure 2.7 illustrates the sequence context for a A>G:T>C mutation, where the context is T[A>G]C:G[T>C]A). Key processes in the introduction of mutations, i.e. DNA damage as well as the efficacy of DNA repair, are to a large extent dependent on the sequence context. Analysis of variant sequence context can thus aid the interpretation of the likely underlying mutational processes. Considering the six types of single nucleotide
variants, and the bases immediately upstream and downstream, a total of 96 different sequence contexts can be established (16 different contexts per variant type).

![Diagram of mutational signature](image)

**Figure 2.7: A variant type incorporated with the bases immediately 5’ and 3’ to the mutated base.**

### 2.6.3 Mutational signatures

Cells within the human body accumulate a certain number of somatic mutations during the course of their lifetime. Originating from a range of both endogenous and exogenous mutational processes, they leave characteristic footprints or patterns of mutations, which have been coined mutational signatures (Petljak and Alexandrov 2016). To put it slightly different, there are frequently multiple mutational processes operative in a single tumor, each contributing its own spectra of somatic mutations to the overall set of mutations that can be observed. The final catalogue will thus contain a mixed record of various past exposures from different mutational processes. Recently, huge datasets produced by sequencing of cancer genomes were coupled with mathematical models in order generate the first comprehensive map of mutational signatures in human cancer. These mathematical models take into account the immediate DNA sequence context of mutations when mutational signatures are delineated. Up to date, >30 distinct mutational signatures have been identified, and etiologies have been proposed for many of them (Alexandrov et al. 2013).

Some cancer types have a substantial proportion of somatic mutations that are known to be generated by exposures, such as tobacco smoking in lung cancers and ultraviolet light in skin cancers (Alexandrov et al. 2013). For example G>T transversions, which are particularly enriched at methylated CG sites (C is directly followed by a G in the DNA sequence) in the TP53 tumor suppressor gene, are characteristic for the smoking-associated lung cancers. The UVB-specific C>T transitions are characteristic in non-melanoma skin cancers (Pfeifer 2010). Somatic mutations leading to tumor development can also be caused by abnormalities
within DNA maintenance, such as defective DNA mismatch repair, which is the case with a significant portion of colorectal cancers (Peña-Diaz et al. 2012).

2.7 DNA variation databases

The DNA sequence variation data for this project is retrieved from five public databases. The Catalogue Of Somatic Mutations in Cancer (COSMIC) and International Cancer Genome Consortium (ICGC) both contain somatic mutations from tumor samples. Germline variants are retrieved from the 1000 Genomes Project (1000Genomes), Exome Aggregation Consortium (ExAC), and The Database of Short Genetic Variation (dbSNP). The variant datasets coming from ICGC, The 1000Genomes, and dbSNP contains variants from all parts of the genome (i.e. both coding and non-coding regions), while COSMIC and ExAC only contain variants from the coding regions of genome.

In the following sections, we will briefly describe the properties of the different DNA variation databases that have been analyzed in this project. Table 2.1 lists the number of variants present in each of them.

<table>
<thead>
<tr>
<th>Database</th>
<th>COSMIC</th>
<th>ICGC</th>
<th>ExAC</th>
<th>1000 Genomes</th>
<th>dbSNP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNV count</td>
<td>1.833.137</td>
<td>37.376.494</td>
<td>9.508.168</td>
<td>81.707.926</td>
<td>142.475.156</td>
</tr>
</tbody>
</table>

Table 2.1: Number of variants for each database. All databases have reference human genome version (GRCh37) as reference sequence.

2.7.1 The 1000 Genomes Project

The 1000 Genomes Project ran between 2008 and 2015 with an aim to characterize genome-wide variation in the human genome and build a better understanding of genetic
variation in different human populations. The project employed low-coverage whole-genome sequencing and deep-coverage exome sequencing in selected individuals, sequencing a total of 2,504 individual genomes from 26 distinct subpopulations across five continents (Africa, Asia, Europe, and America) (1000 Genomes Project Consortium et al. 2012). The subpopulations were further organized as five super populations, from which allele frequencies have been estimated. These include Africa (denoted AFR), mixed Americans (AMR), Europeans (EUR), East Asians (EAS) and South Asians (SAS). Using this setup enabled the detection of most genetic variants with frequencies above 1% in the major human populations. Today, the project is maintained by the International Genome Sample Resource (IGSR, http://www.internationalgenome.org/) that hosts a comprehensive repository of common human genetic variation for use by the scientific community. Each variant record in the 1000 Genomes Project database is associated with a set of minor allele frequencies (represented as as fraction between 0 and 1), each coming from a specific subpopulation.

2.7.2 The Exome Aggregation Consortium (ExAC)

According to the official website (http://exac.broadinstitute.org), the Exome Aggregation Consortium (ExAC) is a coalition of investigators seeking to aggregate and harmonize exome sequencing data from a wide variety of large-scale sequencing projects, and to make aggregated data (i.e. variant frequencies) available to the wider scientific community. Specifically, ExAC provides data consisting of exome variants from 60,706 unrelated individuals (of which 1,851 are contributed by the 1000 Genomes Project), and they claim to have generated so far (as of 17 August 2016) the largest catalogue of human variation in the human protein-coding regions (Lek et al. 2016). The data is derived from a total of 14 different studies, and is available both through online browsing via the ExAC Browser, and as raw VCF download. Similar to the 1000 Genomes Project, there are five major subpopulations, i.e. African (AFR), American (AMR), Non-finnish European (EUR), South Asian (SAS), and East Asian (EAS). In addition, there is also the Finnish population (FIN), and a population class termed ‘Other’ (OTH), which comprises individuals that did not unambiguously cluster with the major populations. A global minor allele frequency is also present for each variant.
2.7.3 The Database of Short Genetic Variations (dbSNP)

dbSNP is the world's largest database for nucleotide variants and is an online resource to aid research on genetic variation. dbSNP was created in 1998 to supplement GenBank and is today a public archive of all short sequence variation across multiple species. Its goal is to act as a single database that contains all identified genetic variation. The data is aggregated and submitted from research laboratories around the world and dbSNP provides access to variants of both germline (primary) and somatic origin that are clinically significant (Kitts et al. 2014). dbSNP is developed and hosted by the National Center for Biotechnology Information (NCBI) in collaboration with the National Human Genome Research Institute (NHGRI). For each variant in dbSNP, methods that have contributed towards variant validation are registered as well as the number of individual submissions for the given reference SNP cluster. These quantities could thus be used to investigate high-quality variants (i.e. validated by many approaches and several submissions) versus low-quality and likely more suspicious variants (i.e. non-validated, few submissions)

2.7.4 The catalogue of somatic mutations in cancer (COSMIC)

COSMIC is an expert-curated online database of somatically acquired mutations found in human cancers. COSMIC, which is hosted by the Wellcome Trust Sanger Institute, was launched in 2004 and contains today several millions of somatic mutations from several hundred thousands of tumor samples. COSMIC collects data for inclusion in the database from all kinds of sequencing studies of tumor material, and will thus include both targeted screens (i.e. a specific set of genes) as well as large-scale genome-wide screens. The database is hosted by the Sanger Institute and is freely available without restriction via its website
(http://cancer.sanger.ac.uk). For each variant recorded in COSMIC, the associated tumor samples along with their tissue site/histology (i.e. tumor type) are recorded. This implies that we can query variants within the database that are found in selected tumor types, as well as those that are occurring in multiple or single samples (i.e. recurrent versus nonrecurrent variants)

2.7.5 The International cancer Genomics Consortium (ICGC)

ICGC is a voluntary scientific organisation that provides a forum for collaboration among cancer and genomic researchers. ICGC was launched in 2008 to coordinate large-scale cancer genome studies in tumours. Today the aim is to provide a comprehensive description of the somatic genomic abnormalities present in a broad range of human tumors. ICGC is founded by participating nations, each of which focuses on one or more forms of cancer, with the goal of mapping genomes of at least 50 types of cancers. Catalogues are produced by ICGC members and are made freely available to researchers.
3 Methods & Implementation

3.1 Introduction to the web application

In short, the web application that has been developed is a tool for the analysis and comparison of the mutational properties of coinciding DNA variation in the germline and the soma (i.e. cancer). The application employs processed data from ExAC, 1000 Genomes Project, dbSNP, ICGC, and COSMIC. Somatic mutation variants are represented by ICGC or COSMIC while germline variants are represented by 1000Genomes, ExAC, or dbSNP. Here we do a thorough quantification and analysis of single nucleotide variants for each pair of somatic and germline database. This process yields six different data comparisons, COSMIC versus 1000Genomes, COSMIC versus ExAC, COSMIC versus dbSNP, ICGC versus 1000Genomes, ICGC versus ExAC, and ICGC versus dbSNP (Figure 3.1). Furthermore, variants within each comparison set are categorized into groups, first those that only occur as germline, second, those that occur only as somatic and third, those that occur both as germline and somatic (i.e. shared or coinciding). These three groups of variant datasets are hereby referred to as unique somatic variants, unique germline variants and coinciding variants, respectively (Figure 3.2).

![Figure 3.1: Each somatic database is paired with a germline database.](image)
this yields 6 somatic and germline comparisons.

Figure 3.2: Variants of each somatic and germline comparison are categorized into three groups: unique somatic, unique germline, and coinciding.

In order to make these new data combinations accessible in a more practical way, it was necessary to design and develop a web-based interface where the data could be queried, filtered, and subject to visualization and analysis. The overall aim is to offer a toolkit for the exploration of characteristic patterns at genomic sites of coinciding DNA variation.

The genomic variation data needed to be pre-processed in multiple ways so that the following functionality could be built into the application:

**Graphical functions:**

- *Venn diagram*
  
  Visualises the relative size of each database and the intersection/overlapping part.

- *Variant types*
  
  Bar plots displaying the relative frequency of variant types. These mutation types are the six different single nucleotide substitutions that can occur within DNA.

- *Variant context types*
  
  Bar plots displaying 96 variant types incorporated with the sequence context. The classification is needed for distinguishing mutational signatures that cause the same substitutions but in different sequence context.

- *Variant consequences*
  
  Bar plots displaying the relative frequency of variant consequence types.
Datatable functions:

- **Coinciding variant table**
  A table listing all the coinciding variant records for the chosen database comparison.

- **Mutational signatures**
  A plot and table displaying the reconstructed mutational signature retrieved from coinciding variants.

Filter functions:

- Cancer type and variant frequency across samples.
- Population region and allele frequency.
- dbSNP validation method.
- dbSNP submission frequency.
- Variant consequence.

In this chapter, we will initially give a brief introduction to the variant call format (VCF), which is central to many of the tools that have been used. Next, we will describe the main tools and methods that have formed the backbone of the application. We will further describe the stepwise process of preprocessing VCF files and quantification of variant attributes that are subsequently stored in simple text files (i.e. comma-separated values (CSV)). Finally, we will outline the technical layout of the interactive web application, with a particular emphasis on the Shiny framework. The practical work with this project can be divided into two key parts:

- Data preparation (preprocessing and quantification of variant data)
- Application implementation (development of interactive web application within the Shiny framework)

3.2 VCF

*The Variant Call Format (VCF)* is a tab-delimited text file format used in bioinformatics for
the specification of DNA sequence variations. The format was developed for large-scale genotyping and DNA sequencing projects, such as the 1000 Genomes Project. In the VCF format, the chromosome and the chromosomal position, along with the genomic sequence variant (reference and alternative allele) constitute the minimum amount of data required for the representation of a DNA variant. Usually, a VCF file contains a list of variants where each variant is specified in a single line of the text file. Each line can contain additional information about the variant (e.g. various annotations) and the list of lines are sorted according to the variant position in the chromosome. The VCF is structured into two parts, a header and a body. The header describes the file format version, the formats and descriptions of the annotation tags per variant (INFO, FILTER, QUAL, and FORMAT columns), and potentially also other metadata, such as data source and the reference genome. In this project, a particularly interesting column is the INFO column, which contains functional annotations about each variant. A VCF file contains numerous variants records and each line of the VCF body represents an mutational variant (Figure 3.3).

```plaintext
##fileformat=VCFv4.1
##fileDate=20160513
##source=ICGC_rel21
##INFO=<ID=ICGC_PROJECTS,Number=.,Type=String,Description="Variant frequency count in different ICGC Project IDs"> 
##INFO=<ID=ICGC_DONORS,Number=.,Type=String,Description="ICGC donor IDs that carry the somatic variant"> 
##INFO=<ID=VT,Number=1,Type=String,Description="Variant type (SNV, INdel, inDEL)"> 
#CHROM POS ID REF ALT QUAL FILTER INFO
1 10002 A T . . PASS VT=snv;ICGC_DONORS=00022731;ICGC_PROJECTS=MELA-AU:1
1 10073 T TA . . PASS VT=INdel;ICGC_DONORS=00022540;ICGC_PROJECTS=PACA-CA:1
1 10144 T T . . PASS VT=INdel;ICGC_DONORS=00035082;ICGC_PROJECTS=PACA-CA:1
1 13494 A G . . PASS VT=snv;ICGC_DONORS=005479;ICGC_PROJECTS=BRCA-US:1
1 13504 G A . . PASS VT=snv;ICGC_DONORS=0052381;ICGC_PROJECTS=LUSC-KR:1
```

Figure: 3.3: A snapshot of the first few lines of a VCF file (ICGC), five variant records are visible here.

### 3.3 Software libraries and computational frameworks

#### 3.3.1 VEP

The Ensembl Variant Effect Predictor (McLaren et al. 2016) (VEP) is a powerful toolset for the analysis, annotation, and prioritization of genomic variants in coding and non-coding regions. The VEP script is useful for adding extra information to the INFO column of each variant in a VCF file, which is done with the help of coordinates and alternative alleles of each mutational variant. With VEP it is possible to discover genes, transcripts, location, consequence, allele frequencies and so on. The consequence type added by VEP is
represented by a SO term (Eilbeck et al. 2005). For each variant that is mapped to the reference genome, each Ensembl transcript that overlap the variant is identified. Figure 3.4 shows the location of each display term relative to the transcript structure.

Figure 3.4: This figure lists all the different consequence types that could be added to a VCFs variant record (courtesy of Ensembl project, 2017).

3.3.2 vcfanno

vcfanno is an effective tool for extracting attributes from multiple annotation files and to integrate annotations within the INFO column of the original VCF file (Pedersen et al. 2016). vcfanno allows to quickly annotate a VCF with any number of INFO fields from other VCF files (Figure 3.5). It uses a simple configuration file to allow the user to specify the source of annotation and fields and how they will be added to the INFO field of a VCF.
3.3.3 Python libraries

Mutational variants needs to be retrieved from VCF files. The Python programming language and the library CyVCF was chosen for this task.

3.3.3.1 Parsing the VCFs variant records

In this project, a Python script written with CyVCF (fast Python library for VCF files using Cython for speed (Pedersen and Quinlan 2017)) is used for parsing through large VCFs and retrieve information from variant records (Figure 3.6).

```python
>>> import cyvcf
>>> vcf_reader = cyvcf.Reader(open('icgc.vcf.gz', 'rb'))
>>> for record in vcf_reader:
...     print record
1 10002   .   A   T   .   PASS
VT=snv;ICGC_DONORS=DO222731;ICGC_PROJECTS=MELA-AU:1;CSQ=T,upstream_gene_variant,MODIFIER ,DDX11L1,ENSG00000223972,Transcript,ENST00000456328,processed_transcript,,,,,,,,,,,1867,1,SNV,HGNC,37102,YES,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,1
1 10073   .   T   TA   .   PASS
VT=INdel;ICGC_DONORS=DO221545;ICGC_PROJECTS=PACA-CA:1;CSQ=A,upstream_gene_variant,MODIFIER ,DDX11L1,ENSG00000223972,Transcript,ENST00000456328,processed_transcript,,,,,,,,,,,1795,1,insertion,HGNC,37102,YES,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,1
1 10144   .   TA  T   .   PASS
```

Figure 3.6: This figure shows basic usage, it takes a VCF as input and prints out each variant line by line.

3.3.4 R and R packages

R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing.

3.3.4.1 deconstructSigs

deconstructSigs is an R package available in R which aims to determine the contribution of known mutational processes in a tumor sample (Rosenthal et al. 2016). In this project, we have not used the package in the standard way, that is, by analysis of a particular tumor sample. Rather, for a given combination of germline and somatic database, we mimic the tumor sample through the distribution of sequence contexts of coinciding variants (i.e. their
relative fraction). In essence, a tumor sample is substituted with a “coinciding variants sample”. Next, we apply the iterative procedure implemented by deconstructSigs (i.e. the method `whichSignatures()`) to delineate or estimate the number and weights of known mutational signatures (and associated underlying etiologies) that can be found in the established “coinciding variants sample”. The known mutational signatures that is used in this project is coming from COSMIC, specifically an analysis of 10,952 exomes and 1,048 whole-genomes across 40 distinct types of human cancer.

### 3.3.4.2 ggplot2

ggplot2 is a popular plotting system and data visualization package for the statistical programming language R (Wickham 2011). Graphical bar plots and pie charts presented in the web application have been developed through methods offered by ggplot2. These plots can be drawn in real time by the use of the render functions of Shiny framework.

### 3.3.5 Shiny framework

Shiny is an open-source web application framework for R developed by RStudio\(^2\). It offers a simple methodology and solution to convey comprehensive data analyses in R through interactive web applications. Importantly, Shiny does not require extensive experience with web development languages (i.e. HTML, CSS or JavaScript), it offers high-level methods that generate the necessary user interface components. The Shiny web framework is fundamentally about making it easy to wire up **input values** from a web page, making them easily available in R, and have the results of R code to be written as **output values** back to the web page (*Figure 3.7*). When input values are changed, output values are updated immediately to reflect those changes.

\(^2\) [https://shiny.rstudio.com/](https://shiny.rstudio.com/)
Shiny web applications are interactive which means that a user can change the input values at any time. Shiny have a also a reactive programming library that can be used to structure application logic. By using the libraries reactive expressions, changing input values will naturally cause only the relevant parts of R code to be re executed, which will in turn cause any changed outputs to be updated. This prevents unnecessary work since reactive expressions are used to control which parts of the app to updated when.

A Shiny application has two components: a user-interface definition (e.g. file named ui.R) where the different parts of the application’s frontend is defined and a server script (e.g. file named server.R) where the data is processed.

Shiny comes with a family of pre-built widgets (web elements that users can interact with), each created with a transparently named R function. Widgets provide a way for users to send messages to the Shiny app. By widgets, inputs are collected from the user. When a user changes the widget, the input will change as well. Two examples of such functions are radioButtons that creates a set of radio buttons and selectInput that creates a drop down box with choices to select from.

Shiny provides a set of functions that turn R objects into output for user-interface . Examples of such functions are plotOutput that creates a plot, tableOutput that creates a table and textOutput that could add an reactive line of text. These output objects can be added to the
user-interface in the same way that HTML elements and widgets are added. By placing them in ui.R tells Shiny where to display objects.

In the server side code, render functions capture input expressions from ur.R and pre-process the expressions. Examples of such render functions are renderPlot that render a plot, renderTable that render a data frame and renderText that render character strings. These functions are assigned to the outputs that appear in user interface.

In summary, new input values (caused by a user changes) will cause rendering expressions (e.g. renderPrint and renderTable) to return new objects (e.g. output$summary and output$view) to be output for user-interface (e.g. textOutput and tableOutput).

3.4 Data preparation: Preprocessing and Quantification of variant attributes

The data retrieved from the variant databases comes in the form of VCF files using the human genome version 19 (hg19) as reference. In additional, a FASTA file containing the whole reference sequence of the human genome, build 37 (GRCh37), are used for the purpose of retrieving the sequence context of each variant type.

The preprocessing of VCF files was done in three steps (Figure 3.8). The first step was to add VEP annotation to each variant of each of the five VCF files. The second step was to sort these files into pairs (each pair is referred to as one comparison) and to annotate both files in each comparison with each other (through the use of vcfanno). This process yields twelve new files (two for each comparison). The third and most time-consuming step was to quantify variant attributes for each comparison and store results in comma-separated values files.

3 Text file at fasta format (.fa)
3.4.1 Step one – Determination of variant consequence/effect

In the first step additional information about both biological effects and consequence for each variant is added. This was done by annotating all VCF files using the Ensembl Variant Effect Predictor (VEP). Variants represented within VCF files often have specific project related information stored in the INFO column, and if any additional information is needed for additional variant analyses, it can be appended here. The VEP script provides annotation for every genomic feature that a variant overlaps and appends it to the variant record. The script (Figure 3.9) was used to annotate every variant of each of the five VCFs. VEP appends a number of annotations in a ‘CSQ’ tag in the INFO column of the query VCF (Figure 3.10).

```
perl variant_effect_predictor.pl --cache --vcf --everything --pick --assembly GRCh37 --port 3337 --fork 6 ICGC.vcf.gz
```

Figure 3.9: In order to process the VCF files, the standalone perl script (VEP) was used to analyse the variation data, (this was done locally). VEP writes several a blocks of annotation to the INFO column if the variant overlap a gene with multiple alternate splicing variants (transcripts). The --pick argument makes sure only one block of annotation is picked for each variant using an ordered set of criteria. However, this option it could involve data loss and lead to the loss of biologically relevant information at some point.

---

4 More information available at their web site (http://www.ensembl.org/info/docs/tools/vep/script/index.html).
Figure 3.10: A snapshot of a VCF file where the added VEP annotation is highlighted in blue. The VEP block has the ID tag 'CSQ' and consists of annotations separated by pipes (vertical bars|).

3.4.2 Step two – Determination of coinciding versus unique variants

The second stage involved the extraction of the unique somatic, unique germline, and coinciding variants from each database comparison. With vcfanno it was possible to detect coinciding variants between two VCF files and annotate those variants in the first VCF with information from the second VCFs INFO field (Figure 3.11).

Figure 3.11: The annotation process is run both ways, resulting in one ICGC.vcf file where its coinciding variants are ExAC-annotated and one ExAC.vcf file where its coinciding variants are ICGC-annotated.

This is equivalent to an ‘left outer join operation’ on two sets of data, A and B, where all the unique variants of A (variant only present in A) and coinciding variants (variants present in both A and B) makes up a new set C (Figure 3.12). This operation was also performed the
opposite direction, *right outer join*, and through that we retrieved the unique variants of set B (Figure 3.13).

**Figure 3.12:** First pass: annotations from ExAC.vcf is added to the coinciding variants of ICGC.vcf, this yields coinciding variants (purple) and unique somatic variants for ICGC (blue).

**Figure 3.13:** Second pass: annotations from ICGC.vcf is added to the coinciding variants of ExAC.vcf, this yields unique germline variants for ExAC.vcf (red) (Coinciding variants are found for each file in a comparison (purple)).

The clue is that *vcfanno* illuminates those variants that are unique by being unannotated (Figure 3.14), which makes it possible to extract both coinciding variants and unique variants for each file. However, to retrieve both unique sets, we ran *vcfanno* in both directions (i.e. by using both the somatic and germline as the query VCF). *Figure 3.14* illustrates how the annotations have been appended to the VCF INFO column.
3.4.3 Step three – Quantification of variant properties

The third step of the data preparation workflow involves the quantification of SNV attributes for each group of variants (unique somatic, unique germline, and coinciding), and for each combination of somatic-germline database. The various quantities are stored in comma-separated values (CSV) files, a simple format that enables rapid data querying and filtering for visualization and analysis in the web application.

The analyses offered by the application require that three essential mutation properties are quantified; variant type (e.g. C>T:G>A etc.), variant context type (e.g. C[>T:G>A]A), and variant consequence (e.g. missense_variant). By quantification, we refer to counts of all individual categories that belong to each of the above-mentioned mutation properties. Moreover, the quantification will be performed according to various subcategories of the data, specifically: cancer type and cancer variant frequency across samples (for somatic databases

---

Figure 3.14: A snapshot of ICGC.vcf annotated with ExAC. Population and allele frequency is added to INFO column of records which are coinciding with records in ExAC.vcf.
only), minor allele frequency (for germline databases only), dbSNP submission frequency and validation method (dbSNP only), and variant consequence (all databases).

To achieve the quantification of mutation properties listed above, it was necessary to develop a number of programming routines. In the different subsections below, we have exemplified how attributes were determined and quantified through Python code snippets.

3.4.3.1 Variant type

All variant records of a VCF file have chromosome number, position, reference base and the altered base (Figure 3.15). Variant records are categorized into six variant types. The variant type is retrieved by parsing the reference and alteration base. Figure 3.16 shows the list of defined variant types written in Python code. Figure 3.17 shows how a variant type is retrieved from a variant record.

Figure 3.15: The essence of a variant record is that the reference base (A) have been altered to a new base (T) at certain position (10505) of a chromosome (1). Parsed text is highlighted in blue.

```python
self.mType = OrderedDict()

# Transitions
self.variantType["A>G:T>C"] = 0
self.variantType["C>T:G>A"] = 0

# Transversions
self.variantType["C>A:G>T"] = 0
self.variantType["C>G:G>C"] = 0
self.variantType["A>T:A"] = 0
self.variantType["A>C:T>G"] = 0
```

Figure 3.16: Snapshot of Python code that shows the list of variant types defined in this project.
def getVariantType(self, record):
    REF = record.REF.upper()
    ALT = record.ALT[0].upper()

    mutation = REF + ALT
    complement = Seq(mutation).complement()

    if (mutation + complement) in self.variantType:
        return (mutation + complement)
    else:
        return (complement + mutation)

Figure 3.17: Snapshot of Python code that decides which variant type a record belongs to. The record shown in Figure 3.15 would be counted as a ‘A>T:T>A’ variant type.

3.4.3.2 Variant context type

Variant records are categorized into 96 variant context types (Figure 3.18). With the variants chromosome number and position it is possible to locate the position of the identical reference base in the reference file (Figure 3.19). The sequence context is found by retrieving the left and right (5' and 3') base next to that position. The variant context type is found by incorporating the variants type with the sequence context (Figure 3.20).

self.mutationContext = OrderedDict()
for mut in ['C>A', 'C>G', 'C>T', 'T>A', 'T>C', 'T>G']:
    for b1 in 'ACGT':
        for b2 in 'ACGT':
            self.mutationContext[(b1 + '+' + mut + b2):'+' + Seq((b1 + '+' + mut + b2)).complement()] = 0

Figure 3.18: Code for generating the 96 different sequence context variant types.

CHROM: 1
POS: 10505
REF: A
chromFA: ..GGTCTCCAGAGCTGAGAAGAC


Figure 3.19: The by knowing the variants chromosomal position it is possible to retrieve the nearby nucleotide sequence by looking up the exact same position in the reference sequence file and retrieve the reference allele left and right nucleotide base (highlighted in blue).
def getMutationContext(self, refSeq, record):
    POS = record.POS
    REF = record.REF
    ALT = record.ALT[0]
    context = refSeq[POS-2].upper() + ["\+REF\+\+ALT\"] + refSeq[POS].upper()
    complement = refSeq(context).complement()
    if (context + "\+complement") in self.mutationContext:
        return (context + "\+complement")
    else:
        return (complement + "\+context")

Figure 3.20: Piece of code for retrieving the variant type including the sequence context.

3.4.3.3 Consequence type

By parsing the INFO column it is possible to retrieve the consequence type of each variant record (Figure 3.21). Variant records are categorized into 18 different consequence types depending on their genomic location (Figure 3.22, Table 3.1).

### Returns only the first of potentially several consequences for each sample

def getConsequence(self, vep):
    consequence = vep[\'Consequence\'].split(\'\&\')[0]
    if consequence in self.consequenceList:
        return consequence
    else:
        return None

Figure 3.22: Note that only the first of potentially several consequence types is retrieved. Variant records could be registered with several consequence types.

<p>| Coding consequence | Noncoding consequence |</p>
<table>
<thead>
<tr>
<th>Splice acceptor variant</th>
<th>Splice region variant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splice donor variant</td>
<td>5 prime UTR variant</td>
</tr>
<tr>
<td>Stop gained</td>
<td>3 prime UTR variant</td>
</tr>
<tr>
<td>Stop retained variant</td>
<td>Non coding transcript exon variant</td>
</tr>
<tr>
<td>Stop lost</td>
<td>Intron variant</td>
</tr>
<tr>
<td>Start lost</td>
<td>NMD transcript variant</td>
</tr>
<tr>
<td>Missense variant</td>
<td>Non coding transcript variant</td>
</tr>
<tr>
<td>Synonymous variant</td>
<td>Upstream gene variant</td>
</tr>
<tr>
<td></td>
<td>Downstream gene variant</td>
</tr>
<tr>
<td></td>
<td>Intergenic variant</td>
</tr>
</tbody>
</table>

Table 3.1: Variants are categorized into either coding or noncoding according to this table.  
http://www.sequenceontology.org/

3.4.3.4 Variant consequence

Here we define the category variant consequence. Variant records that have variants types located in the exome region of the sequence are set to be coding, those outside are set to be noncoding (Table 3.1) (Figure 3.23).

![Figure 3.23](image)

Figure 3.23: This figure shows the different types of variants consequences and their position. Consequence types with blue text are defined as ‘coding’ variants. This is a modified version of figure 3.4.
3.4.3.5 Cancer type and Variant frequency across samples

From variants records with COSMIC annotation it is possible to retrieve cancer type and variant frequency across samples (Figure 3.24). Here we define the category cancer type, a variant record (of COSMIC) must have one of the cancer types listed in Table 3.2. We also define the category variant frequency across samples. These variant frequencies are set to be non-recurrent if the number samples equals one \(f = 1\) and recurrent if it's greater than one \(f > 1\) (Figure 3.25). This category is bound to the cancer type, every record with a cancer type have a frequency.

Figure 3.24: The tag COSMIC\_CANCER\_TYPE\_GW tells which tumor type the variant is sampled from and the variant frequency is the number of independent samples (acute\_myeloid\_leukemia:1) with this variant.

### Table 3.2: Cancer types and frequencies

<table>
<thead>
<tr>
<th>Cancer Type</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>acute_myeloid_leukemia</td>
<td>1</td>
</tr>
<tr>
<td>leukemia</td>
<td>1</td>
</tr>
<tr>
<td>pancancer</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3.25: Variants are set to have either recurrent or nonrecurrent cancer frequency.

```python
def getCancer(self, record):
    cancer = {}
    if 'COSMIC\_CANCER\_TYPE\_GW' not in record.INFO:
        return {}
    for cancerType in record.INFO['COSMIC\_CANCER\_TYPE\_GW'].split('&')[1].split(':'):
        c = cancerType.split(':')
        if int(c[1]) > 1:
            cancer[c[0]] = 'recurrent'
        else:
            cancer[c[0]] = 'nonrecurrent'
```

#CHROM POS ID REF ALT QUAL FILTER INFO
1 35416 . A G . PASS

COSMIC\_CODON\_COUNT\_GW=28:pancancer:1,28:leukemia:1,28:acute\_myeloid\_leukemia:1;COSMIC\_SITE\_HISTOLOGY=haematopoietic\_and\_lymphoid\_tissue@haematopoietic\_neoplasm:1;COSMIC\_CANCER\_TYPE\_GW=acute\_myeloid\_leukemia:1,pancancer:1;COSMIC\_FATHMM\_PRED=unknown;COSMIC\_SAMPLE\_NAME\_ALL=CN\_-AML\_-CR\_-42\_-Dx;COSMIC\_MUTATION\_ID=COSM5428243;COSMIC\_GENE\_INFO=FAM138A:ENSG00000237613;COSMIC\_CANCER\_TYPE\_ALL=leukemia:1,acute\_myeloid\_leukemia:1,pancancer:1;COSMIC\_SAMPLE\_SOURCE=primary:1;COSMIC\_COUNT\_GW=1;COSMIC\_VARTYPE=snv;COSMIC\_CONSEQUENCE=substitution\_missense

Def get cancer (self, record):

cancer = {}
if 'COSMIC\_CANCER\_TYPE\_GW' not in record.INFO:
    return {}
for cancerType in record.INFO['COSMIC\_CANCER\_TYPE\_GW'].split('&')[1].split(':'):
    c = cancerType.split(':
    # c[0] = cancer type, c[1] = cancer frequency
    if c[0] in self.cancerTypes:
        if int(c[1]) > 1:
            cancer[c[0]] = 'recurrent'
        else:
            cancer[c[0]] = 'nonrecurrent'
```
<table>
<thead>
<tr>
<th>Acute lymphoblastic b-cell leukemia</th>
<th>Colorectal cancer</th>
<th>Oesophageal cancer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acute lymphoblastic leukemia</td>
<td>Diffuse large B cell lymphoma</td>
<td>Ovarian cancer</td>
</tr>
<tr>
<td>Acute myeloid leukemia</td>
<td>Glioma</td>
<td>Pancreatic cancer</td>
</tr>
<tr>
<td>Breast cancer</td>
<td>Kidney cancer</td>
<td>Prostate cancer</td>
</tr>
<tr>
<td>Cervical cancer</td>
<td>Liver cancer</td>
<td>Sarcoma</td>
</tr>
<tr>
<td>Cholangiocarcinoma</td>
<td>Lung cancer</td>
<td>Stomach cancer</td>
</tr>
<tr>
<td>Chronic lymphocytic leukemia</td>
<td>Malignant melanoma</td>
<td>Urothelial cancer</td>
</tr>
</tbody>
</table>

Table 3.2: List of 21 cancer types in alphabetic order. In addition comes ‘pancancer’ which represents the global category of cancer (i.e. any cancer type)

3.4.3.6 Population and Minor allele frequency

Variant records of 1000Genomes and ExAC are registered with population region and allele frequency. The category **population** for 1000Genomes and ExAC is defined in the following way:

- 1000Genomes annotation have allele frequencies in up to five population regions; EAS_AF_1KG, EUR_AF_1KG, AFR_AF_1KG, AMR_AF_1KG and SAS_AF_1KG which are tags for *East Asia, Europe, Africa, America and South Asia* respectively (Figure 26).
- ExAC annotation have up to eight allele frequency population regions as follows: AF_AFR_EXAC, AF_AMR_EXAC, AF_EAS_EXAC, AF_FIN_EXAC, AF_NFE_EXAC, AF_SAS_EXAC and AF_Adj_EXAC which are tags for *Africa, America, East Asia, Finnish, Non Finnish Europe, South Asia and Adjusted Global* respectively (Figure 3.27). In additional to the main regions of ExAC is the AF_OTH_EXAC which stands Other Allele Frequency which is not used in this project.
We also define the category **minor allele frequency** which is tied to a population category. The **minor allele frequency** of 1000Genomes and ExAC (the relative frequency of a non-reference allele at a particular locus in a population) is calculated in the range 0 to 1. Allelic frequencies are further categorized into *Common* for frequencies greater than 5%, those between 1% and 5% are defined as *Low Frequency*, those between 1% to 0.1% are defined as *Rare* while those below 0.1% are defined as *Very Rare* (Figure 3.28).

```python
def getPopulation(self, record):
    pop = {}
    for k in record.INFO.keys():
        if k in self.population:
            MAF = record.INFO[k]
            if MAF > 0.0:
                if MAF >= 0.05:
                    pop[k] = 'Common'
                elif (MAF < 0.05 and MAF >= 0.01):
                    pop[k] = 'LowFreq'
                elif (MAF < 0.01 and MAF >= 0.001):
                    pop[k] = 'Rare'
                else:
                    pop[k] = 'VeryRare'
    return pop
```

*Figure 3.28: MAF (Minor Allele Frequencies) larger than 5% are set to be “Common”, in range of 1% to 5% are “Low Frequency”, in range of 1% to 0.1% are “Rare” and those below 0.1% are “Very Rare”.*
These numbers have been generalized into four categories: 1, 2:5, 6:10 and >10. So let's say if a variant has a submission number 2, it would fall into the 2:5 category (Figure 3.29). Likewise, if a number is above 10, it would be the >10 category.

<table>
<thead>
<tr>
<th>#CHROM</th>
<th>POS</th>
<th>ID</th>
<th>REF</th>
<th>ALT</th>
<th>QUAL</th>
<th>FILTER</th>
<th>INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10150</td>
<td>rs371194064</td>
<td>C</td>
<td>T</td>
<td>.</td>
<td>.</td>
<td>DBSNP_MAPPINGS=unique-in-contig;DBSNPRSID=371194064;DBSNPBUILDID=138;WGT=1;VT=snv;DBSNP_SUBMISSIONS=2;SSR=0;DBSNP_VALIDATION=byCluster</td>
</tr>
</tbody>
</table>

**Figure 3.29:** The annotation **DBSNP_SUBMISSIONS**=2 tells the number of submissions is 2 and the annotation **DBSNP_VALIDATION**=byCluster tells that the validation method used for variant validation is byCluster.

### 3.4.3.8 dbSNP Validation

Variant records with dbSNP contains also information about which methods that have contributed towards variant validation (Figure 3.29). The **dbSNP Validation** category is defined the following way. Validation methods are put into the following categories: **byCluster**, **byFrequency**, **by1000G**, **byOtherPop**, **suspect**, **byHapMap**, **by2Hit2Allele**, **unknown** and **multiple**. The last category is made those variants that are validated by more than one method. If an variant is validated by byCluster (**DBSNP_VALIDATION**=byCluster) it would fall into the **byCluster** category, but if it is registered with another method also, for instance by1000G, it would go into the **multiple** category.

### 3.4.3.9 Datatable function

A datatable function that display coinciding variant records. The following information is presented for each variant record in the coinciding variant table; assembly version (GRCh37), gdna position, consequence type, gene, symbol, biotype, cosmic mutation id, cancer type and info. The gdna position is a string composed of the chromosome number, position, reference and alteration base (CHROM:POS:REF>ALT). By parsing the **INFO** column it is possible to retrieve the variants gdna position, consequence type, gene, symbol and biotype (Figure 3.30).

<table>
<thead>
<tr>
<th></th>
<th>771685</th>
<th>T</th>
<th>C</th>
<th>PASS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>COSMIC_CODON_COUNT_GW=94:liver_cancer:1894:pancancer:1:COSMIC_SITE_HISTOLOGY=liver@carcinoma:1;COSMIC_CANCER_TYPE_GW=liver_cancer:1&amp;pancancer:1;COSMIC_FATHMM_PRED=neutral;COSMIC_SAMPLE_NAME_ALL=RK305_C01;COSMIC_MUTATION_ID=COSM4943781;COSMIC_GENE_INFO=.:ENSG00000197049;COSMIC_CANCER_TYPE_AL...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
L=liver_cancer:1&pancancer:1;COSMIC_SAMPLE_SOURCE=NS:1;COSMIC_SOMATIC_STATUS=somatic_confirmed:1;COSMIC_COUNT_GW=1;COSMIC_SAMPLE_NAME_GW=RK305_C01;COSMIC_VARTYPE=snv;COSMIC_CONSEQUENCE=substitution_missense;CSQ=C,intron_variant&non_coding_transcript_variant,MODIFIER,LINC01128,ENSG00000228794,Transcript,ENST00000445118,lincRNA,,2/4,ENST00000445118.2:n.270+7201T>C,,,,,,,rs2519005,,1,,SNV,HGNC,49377,YES,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,Figure 3.30: The gnda position of this variant would be 1:771685:A>C. The cosmic mutation id (COSMIC_MUTATION_ID) is COSM4943781. The third instance of the CSQ block contains the gene symbol (LINC01128), fourth is gene id (ENSG00000228794) and the seventh is biotype (lincRNA). Parsed text is highlighted in blue.

3.4.3.10 Building .CSV files

Here, the quantified data is put in comma-separated values (CSV), files which stores tabular (numbers and text) data in plain text. Each line of a CSV file will represent a data record and each record consists of a set of fields which are separated by comma. So a record will represents a set of attributes which a mutational variant could have. The last field of each record contain an integer value that represents the number of variants that fit this particular combination of data.

Three Python scripts were written in order to read VCF files, quantify variant records and build CSV files. The first file quantifyVariants.py takes an VCF as input, iterates it and sort variant records into coinciding and unique variants. The second script, buildCSV.py, build and fill two Python dictionaries and writes two CSV files to disk, one for coinciding variants and one for unique variants. The third file, analyseRecord.py, is used for parsing and extracting information out of each variant record.

The Python dictionaries fits all kinds of variant records. For example, a record retrieved from a VCF file could contain annotation about variant type, cancer type, variant frequency across samples, population and minor allele frequency. For every record, this annotation is extracted, turned into a string and used as a dictionary key for variants with a particular combination of annotation (Figure 3.31). The value is the number of variants with equal keys. Figure 3.32 shows how a CSV file looks like.
def regVariant(self, record, recordVEP, referenceSequence):
    mutation = self.mt.getType(record)
    context = self.mc.getMutContext(referenceSequence, record)
    consequence = self.cq.getConsequence(recordVEP)
    coding = self.cq.getCodingType(recordVEP)
    cancer = self.ct.getCancer(record)

    if consequence:
        if context:
            if cancer:
                for cancerType, cancerFreq in cancer.items():
                    self.variantPlot[cancerType]+","+cancerFreq+","+coding+","+mutation] += 1
                    self.contextPlot[cancerType]+","+cancerFreq+","+coding+","+context] += 1
                    self.consequencePlot[cancerType]+","+cancerFreq+","+coding+","+consequence] += 1
            return True

Figure 3.31: For every input record, the mutation variant, variant context, consequence, coding (coding or noncoding) and cancer type is retrieved from parsing the variant record and used as credentials for adding the occurrence to the table.

plot,set,cancer,cancerFreq,coding,type,n
variantPlot,unique_somatic,acute_myeloid_leukemia,recurrent,coding,A>G:T>C,967
variantPlot,unique_somatic,acute_myeloid_leukemia,recurrent,coding,C>T:G>A,1389
variantPlot,unique_somatic,acute_myeloid_leukemia,recurrent,coding,C>A:G>T,197
variantPlot,unique_somatic,acute_myeloid_leukemia,recurrent,coding,C>G:G>C,253
variantPlot,unique_somatic,acute_myeloid_leukemia,recurrent,coding,A>T:T>A,128
variantPlot,unique_somatic,acute_myeloid_leukemia,recurrent,coding,A>C:T>G,196

Figure 3.32: This is a snapshot of the CSV file created by the table that was filled in figure 3.31. plot could be either variant plot, context plot or consequence plot. set is either coinciding, unique_somatic or unique_germline. type depends on the plot type, consist either of 6 variant types, 96 sequence context types or 18 consequence types. n is the number of variant records.

3.4.4 Variants - requirements for inclusion in quantification

Not all variants records have the necessary annotation and some variants are ignored during parsing. Variant records that are incomplete or miss annotation are also ignored. The total number of VCF variant records quantified in this project is visible in Table 3.3.

The following records are ignored/filtered out:

- Records with missing INFO column annotation (cancer type, population region etc.).
- Records with incomplete germline or somatic annotation
  (For example: A COSMIC record may have annotation such as ‘COSMIC_CANCER_TYPE_ALL’ and be missing ‘COSMIC_CANCER_TYPE_GW’).
- COSMIC records without any of the cancer types listed in Table 3.2.
- Records without any of the consequence types listed in Table 3.1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Unique somatic</th>
<th>Unique germline</th>
<th>Coinciding</th>
<th>Skipped variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>COSMIC_ExAC</td>
<td>1.199.273</td>
<td>8.800.314</td>
<td>357.211</td>
<td>276.653 cosmic 350.643 exac</td>
</tr>
<tr>
<td>COSMIC_oneKG</td>
<td>1.456.787</td>
<td>78.766.361</td>
<td>102.695</td>
<td>273.655 cosmic 2.838.870 oneKG</td>
</tr>
<tr>
<td>COSMIC_dbSNP</td>
<td>1.169.204</td>
<td>137.572.912</td>
<td>390.488</td>
<td>273.445 cosmic 4.511.756 dbSNP</td>
</tr>
<tr>
<td>ICGC_ExAC</td>
<td>35.966.033</td>
<td>8.829.400</td>
<td>380.024</td>
<td>1.030.437 icgc 298.744 exac</td>
</tr>
<tr>
<td>ICGC_oneKG</td>
<td>34.816.916</td>
<td>77.353.635</td>
<td>1.530.520</td>
<td>1.029.058 icgc 2.823.771 exac</td>
</tr>
<tr>
<td>ICGC_dbSNP</td>
<td>33.490.171</td>
<td>135.160.914</td>
<td>2.860.450</td>
<td>1.025.873 icgc 4.453.792 exac</td>
</tr>
</tbody>
</table>

Table 3.3: This table shows how many variant records that have been quantified for each comparison.

3.5 Application implementation

3.5.1 Application layout

The web application has been coded in three distinct files:
- a source file (ui.R) for defining the user-interface,
- a server script (server.R) for processing data
- a support script (global.R) for loading datatables (CSV files) into memory at application startup (Figure 3.33).
The R package flexdashboard\(^5\) was used to setup the layout of the user interface and publish groups of related data visualizations as a dashboard. ui.R is implemented with control widgets (e.g. selectInput) which receives input from user. The server.R file, with R code, captures input expressions from ui.R, process data (filtering variants etc.) and return R objects (renderPlot). In ui.R, there are functions that receive these R objects and turn them into output for user-interface (e.g. plotOutput).

```r
library(data.table)

## cosmic_oneKG
cosmic_oneKG = list(
  "coinciding" = fread("data/coinciding/COSMIC_oneKG_coinciding_genome.csv", header=T, sep="","", stringsAsFactors=F),
  "germline" = fread("data/germline/COSMIC_oneKG_unique_germline_genome.csv", header=T, sep="", stringsAsFactors=F),
  "somatic" = fread("data/somatic/COSMIC_oneKG_unique_somatic_genome.csv", header=T, sep="", stringsAsFactors=F)
)
```

Figure 3.33: In global.R, CSV files are loaded into memory. Here is a snapshot of the code that loads the datatable for the COSMIC versus 1000Genomes comparison. Each CSV file have tables for variant plot, context plot and consequence plot.

### Sidebar panel

The sidebar panel contains two subpanels that make up the somatic panel and the germline panel. The sidebar panel is implemented with widget functions in order to receive input from user. The following widgets were implemented with selectInput(): somatic database, germline database, cancer type, population region and minor allele frequency (example for choice of population shown in Figure 3.34). The following widgets were implemented with radioButtons(): variant consequence, variant frequency across samples, dbSNP validation and dbSNP submission frequency.

```r
selectInput(  
  "oneKpopulation",
  label = "Populations",
  choices = list(
    "African" = "AFR_AF_1KG",
    "Asian" = "ASIA_AF_1KG",
    "American" = "AMERICAN_AF_1KG"
  ),
  multiple = TRUE
)
```

3.5.3 Charts, statistics and data tables

This menu is divided into separate tab windows which illustrate the different data analyses with charts, statistics, and data tables. The server side has render functions to capture input expressions from ui.R and produce output objects such as plots, tables, and text.

Statistics and a Venn diagram

In his window, htmlOutput() functions displays statistics about the somatic database, germline database and the intersection/overlap. The relative sizes of each databases are also illustrated in a venn diagram implemented by a function plotOutput() in ui.R and a renderPlot() in server.R.

Variant types

Bar plots displaying the relative frequency of six variant types. Two bar plots are displayed, one for coinciding and unique dataset (which unique dataset is optional) and and a logarithm plot for comparison of coinciding and unique dataset. These plots are implemented by function plotOutput() in ui.R and function renderPlot() in server.R.

Variant context type

Bar plot displaying 96 variant types incorporated with the sequence context. Four of these bar plots are displayed, one for coinciding, unique somatic, unique germline and one logarithm plot. These plots are implemented by function plotOutput() in ui.R (Figure 3.35) and function renderPlot() in server.R (Figure 3.36).
**Variant consequence**

Bar plot displaying the relative frequency of up to 18 variant consequence types\(^6\). Four bar plots are displayed, one for coinciding, unique somatic, unique germline and logarithm plot. These plots are implemented by function `plotOutput()` in `ui.R` and function `renderPlot()` in `server.R`.

**Coinciding variant table**

A table listing all the coinciding variant records retrieved from the VCF files. Each variant lists the following: *Assembly (GRCh37)*, *gdna_pos (chromosome:position:variant)*, *consequence*, *gene*, *gene symbol*, *biotype*, *cosmic mutation id*, *cancer and additional information (attributes)*. Implemented by function `dataTableOutput("dataTable")` in `ui.R` and function `DT::renderDataTable()` in `server.R`.

**Mutational signatures**

The R library `deconstructSigs` have a function `whichSignatures()` that takes the weights of the sequence context plot of coinciding variants as input and output the weights of mutational signatures that are identified. These weights are used to reconstruct the mutational profile. A pie chart and a context plot based on these mutational signature are implemented by `plotOutput()`. Signature weights are rendered in both text and as a table which describes the mutational signatures. This is also displayed by `htmlOutput()` and `tableOutput()`.

```r

tabPanel(
  h4("Sequence context"),
  value = "context",
  wellPanel(
    id = "tPanel",
    style = "overflow-y:scroll; max-height: 1000px;background:white",
    tags$h3("Coinciding variants"),
    plotOutput("contextPlot_coinciding"),
  ),
),

Figure 3.35: The sequence context plot is received from `plotOutput` (Figure 3.36) and placed in a tab titled “Coinciding variants” in the tab panel.

\(^6\) The set of consequence terms are defined by the [Sequence Ontology (SO)](https://www.geneontology.org)
```
Figure 3.36: Function `renderPlot` return `ggplot` object which assigned to the output variable `contextPlot_coinciding`.
4 Results

In this chapter we will describe in more detail the functionality and usability of the web application for joint analysis of germline and somatic mutational spectra. Initially, we will explain how the user interface has been designed and how the user can apply several filters on the underlying variant datasets. Next, we will illustrate the output of the various types of analyses that can be explored for a selected combination of somatic and germline mutation dataset. In the last part of the chapter, we will go through examples of research questions that demonstrate how the web application can be used to reveal particular properties of coinciding variants, specifically the ways in which the various filtering options (tumor type, germline allele frequency etc.) will impact the observed patterns of coinciding variations, and how these could be interpreted.

4.1 Application layout

The interface to the web application consists of two key components, a sidebar panel and a main panel (Figure 4.1). The sidebar panel offers a variety of selection criteria to subset or filter the variant datasets. The main panel has a tabset panel that consist of different windows, where each window contains the results (frequently in the form of visualizations) for a particular analysis of mutation datasets. The user scenario is thus to allow the user to specify particular subsets of germline and somatic databases to compare within the sidebar panel, and view the output updated in real time within the main panel.
4.1.1 Sidebar panel

The database sidebar panel has two subpanels, a somatic data panel and a germline data panel (Figure 4.2). Beneath these panels, there are buttons for the selection of Variant consequence. The types of selections available in each data panel depends on the choice of database. The somatic data panel has a selection for Somatic database and two additional selections, Cancer type and Variant frequency across samples, these choices will only appear if COSMIC is chosen. The germline data panel permits the user to choose one of the three germline databases: 1000 Genomes Project, ExAC, or dbSNP. If 1000 Genomes Project or ExAC is selected, then options for population and minor allele frequency will appear. When choosing dbSNP as the germline database, filtering options for variant validation method and submission frequency will occur, criteria which potentially may indicate variants of different levels of confidence.
Figure 4.2: Sidebar panel is shown to the left with: Variant consequence (1), Somatic database (2), Cancer type (3), Variant frequency across samples (4), Germline database (5), Population (6), Minor allele frequency (7). To the right: If dbSNP is selected, then the germline datapanel will be displayed with dbSNP validation method (8) and dbSNP submission frequency (9).

Detailed description of the selection/filter criteria displayed in Figure 4.2:

1. **Variant consequence**: This filter option allows the user to view coding or noncoding variants separately, for both somatic and germline datasets (As described in chapter 3.4.3.3).

2. **Somatic database**: This option allows the user to choose one of the two somatic databases, either COSMIC or ICGC, for a comparative analysis with a germline database.

3. **Cancer type**: A filter option that allows the user to view a specific cancer type. The different cancer types are listed in Table 3.2.

4. **Variant frequency across samples**: A filter option that allows the user to distinguish between variants that are recurrent (i.e. observed in several tumor samples) or those
that are nonrecurrent (i.e. observed in single samples only) *(Described in chapter 3.4.3.6).*

5. **Germline database:** This option allows the user to choose between 1000 Genomes, ExAC and dbSNP. The choice of germline database is used for comparative analysis with somatic database.

6. **Population:** This filter option allows the user to limit the germline variant set to the set that has observed minor allele frequencies in a specific population *(Described in chapter 3.4.3.7).* 1000 Genomes Project has five populations and ExAC has seven populations. ExAC also has the population *Adjusted Global,* which refers to the variant set in which no particular population is selected (i.e. ‘Any population’).

7. **Minor allele frequency:** A filter option that allows the user to view common, low frequency, rare, very rare variants separately *(Described in chapter 3.4.3.7, Figure 3.28).*

8. **dbSNP validation method:** This filter option allows the user to only consider dbSNP variants that has been validated according to a specific method (described in chapter 3.4.3.8).

9. **dbSNP submission frequency:** A filter option that allows the user to limit the dbSNP variant dataset to those that carry a particular submission frequency (described in chapter 3.4.3.8).

### 4.1.2 Main panel

The main panel consists of a tabset panel in which each tab displays a window. A user can view information and output in the form of statistics, plots, and data tables at these windows. The tabset panel consists of the following tabs; *Overall statistics, Variant type, Sequence context, Variant consequences, Coinciding variants, Mutational signatures,* and *About.* I will in the next subsections describe the contents and analysis presented in each of the above-mentioned tabs.

#### 4.1.2.1 Overall statistics

The tab-window with overall statistics *(Figure 4.3)* displays coinciding variant statistics categorized by the following headlines:
1. **Somatic variant database**: <somatic database>
2. **Germline variant database**: <germline database>
3. **Results from intersection of germline and somatic variant loci**

![Figure 4.3: Main panel analysis tab - overall statistics](image)

The somatic variant database and germline variant database headlines (1 and 2) indicates which pair of databases that are subject to comparative analysis. The size (number of variants) of each dataset is visible under each headline. These numbers are dependent on the status of other side panel selections. For example, if a user choose to select a cancer type such as lung cancer, all the non-lung cancer variants will be filtered out and the number of somatic variants will be altered accordingly. Furthermore, every specific user selection (such as lung cancer) at the side panel is also visible under its respective headline.

The third headline, **Results from intersection of germline and somatic variant loci** (3), shows the following numbers of the intersection between the somatic and germline dataset:

1. Number of coinciding variants and the corresponding relative fraction (percentage) within the chosen somatic and germline datasets.
2. Number of unique somatic variants and the corresponding relative fraction (percentage) within the somatic dataset.
3. Number of unique germline variants and the corresponding relative fraction (percentage) within the germline dataset.
A Venn diagram is displayed in order to visualize the size of the coinciding variant dataset in relation to the whole datasets of germline and somatic variants.

4.1.2.2 Variant types

This page displays two bar plots and at the bottom left corner there is a dropdown menu for selection between unique germline and unique somatic dataset (Figure 4.4). The first bar plot called Variant type plot and shows two datasets, the relative frequency of coinciding variant types and the relative frequency of either unique somatic or unique germline. The second plot is called Enrichment/Depletion of coinciding variants. This barplot indicates the relationship between coinciding and unique variant types. An enrichment ratio is calculated by the following formula: log2(coinciding/unique). For both plots, a user can use the dropdown menu at the bottom left to choose the dataset for which the coinciding variant set should be compared against.

4.1.2.3 Sequence context

The sequence context analysis tab displays four barplots (Figure 4.5). Three plots shows the relative frequency of variant types, now expanded with their immediate DNA sequence context (i.e. DNA three-mers). The following sequence context plots are present: Coinciding variants, Unique somatic variants, Unique germline variants, and Enrichment/Depletion of coinciding variants. In the three uppermost plots, the X-axis consists of the 96 different mutational sequence contexts and the Y-axis displays their relative frequency. The fourth plot models the proportional difference between coinciding variants and either unique somatic or unique germline variants. The ratio/transformation is calculated by log2(coinciding/unique). A sequence context with log-ratio of 1 will thus indicate that there are twice as many coinciding compared to unique variants for this particular context. At the left-bottom part of the page there is a dropdown menu where the user can choose to perform the enrichment analysis of coinciding variants against the set of non-coinciding somatic mutations or non-coinciding germline variants.
4.1.2.4 Variant consequences

In a similar fashion as the sequence context tab, this tab window displays four barplots of variant consequence types (Figure 4.6). Only consequence types with an occurrence within the coinciding dataset is chosen for visualization. The following variant consequence plots are present: Coinciding variants, Unique somatic variants, Unique germline variants and
Enrichment/Depletion of coinciding variants. The three uppermost plots consists of up to 18 variant consequence types along the X-axis and the Y-axis displays their relative frequency. The fourth plot displays the enrichment of coinciding variants in either unique somatic or germline variants, calculated by a log-ratio, e.g. log2(coinciding/unique). At the left-bottom of the page there is a dropdown menu where the user can choose to perform the enrichment analysis of coinciding variants against the set of non-coinciding somatic mutations or non-coinciding germline variants.

4.1.2.5 Coinciding variants

This window displays a data table containing various functional annotations of all coinciding variants (Figure 4.7). Specifically, information related to consequence type, gene names, type of genes (e.g. protein-coding etc.), cancer types, and cross-references to COSMIC are provided. Many of the annotations are represented as links to external databases/sources. Due to data limitations by the underlying Javascript code, data tables are only available for comparisons with the COSMIC database.
4.1.2.6 Mutational signatures

This window displays a reconstructed mutational profile of the coinciding variant set, and the relative weights of known mutational processes within the set. The `whichSignatures()` function of the deconstructSigs package (see section 3.3.4.1) accepts as input the distribution of the sequence contexts of coinciding variants, and estimates the contributions of known mutational signatures. The sequence context plot visualizes the signatures that are present. A percentage of each signature is also displayed and a pie chart to show their relative size (Figure 4.8). A table with detailed information about each signatures is also present (more information about each signature is available at the COSMIC mutational signature website (http://cancer.sanger.ac.uk/cosmic/signatures)).
4.2 Case examples of functionality

In this section, we explore key functionality and various types of analyses offered by http://coinciding-snv-analyzer.no. Through use cases, we attempt to demonstrate how different research questions related to coinciding DNA variation can lend support from the interactive web application that has been developed.

4.2.1 Coinciding variants - coding versus non-coding

If we choose a comparison of IGCC versus ExAC and look at tab “Overall statistics”, we see that the major database ICGC have ~37.3 million variants and the minor database ExAC have ~9.2 million variants. We also see that the coinciding variants are overlapping 1.02% of somatic and 4.13% of germline. These numbers will change by selecting “Coding” under Variant consequence at the sidebar. Now only the coding variants are analysed. Now somatic have only ~2.1 million variants and germline have ~5.4 million variants (Figure 4.9). The coinciding overlap of somatic and germline has increased significantly, from 1.02% to 14.8%. This demonstrates how the nature of the databases selected for comparison can have a
major impact on the results. Here, with the awareness that ExAC is primarily a database of coding variations, while ICGC contains variants both from whole genomes and exome sequencing runs, the shift in the relative sizes of coinciding and non-coinciding variants is as expected..

**Figure 4.9:** These Venn diagrams visualize the relative size of each database and their overlap. The left figure shows both coding and noncoding variants. Right shows coding variants only.

---

**Figure 4.10** displays two context plots, the left plot have *Variant consequence* set to coding (39,878 coinciding variants) and the right have *Variant consequence* set to noncoding (603,866 coinciding variants). As can be observed from the figure, there are some obvious differences between the DNA sequence context of coinciding variations when it comes to the distinction between coding and non-coding variants.

**Figure 4.10:** Right: coding, left: noncoding (ICGC versus 1000Genomes, Population set to European).
Figure 4.11 comes from an analysis using the same settings as those used for Figure 4.10 (ICGC versus 1000Genomes, population = European, allele frequency = Any). The figure displays two consequence plots made of coinciding variants, those found by filtering on coding variants versus those found using non-coding variants. Unsurprisingly, the right plot with coding variants has a majority of missense variants and synonymous variants, while the left plot is enriched with intergenic and intron variants.

![Figure 4.11: Right: coding, left: noncoding (ICGC versus 1000Genomes, Population set to European).](image)

4.2.2 Coinciding variants - relation to germline allele frequency

If we select COSMIC as the somatic database, ExAC as germline database. Variant consequence is set to coding and noncoding. For ExAC, population is set to adjusted global. For COSMIC, cancer type is set to any cancer type and variant frequency across samples is set to both recurrent and nonrecurrent.

Now if we look at the number of coinciding variants for each the four minor allele frequencies (Table 4.1) we notice that:

- Minor allele frequencies have variant sizes in the following declining order: Very Rare, Common, Rare and Low Frequency.
- The coinciding share of germline variants have the following declining order: Common, Low Frequency, Rare and Very Rare

<table>
<thead>
<tr>
<th>Allele Frequency</th>
<th>Number of Coinciding variants</th>
<th>Coinciding share of somatic</th>
<th>Coinciding share of germline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common</td>
<td>28 017</td>
<td>1.17%</td>
<td>19.6%</td>
</tr>
<tr>
<td>Low Frequency</td>
<td>6 770</td>
<td>0.282%</td>
<td>9.36%</td>
</tr>
<tr>
<td>Rare</td>
<td>12 840</td>
<td>0.535%</td>
<td>4.76%</td>
</tr>
</tbody>
</table>
Now we repeat the same procedure for the cancer types acute myeloid, breast, breast, lung, kidney, malignant melanoma, sarcoma and prostate cancer (Table 4.2).

<table>
<thead>
<tr>
<th>Cancer type</th>
<th>Common</th>
<th>Low freq.</th>
<th>Rare</th>
<th>Very Rare</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>948</td>
<td>400</td>
<td>787</td>
<td>18 143</td>
</tr>
<tr>
<td></td>
<td>0.597%</td>
<td>0.252%</td>
<td>0.496%</td>
<td>0.199% of somatic</td>
</tr>
<tr>
<td></td>
<td>of somatic</td>
<td>of somatic</td>
<td>of somatic</td>
<td>germline</td>
</tr>
<tr>
<td></td>
<td>0.664%</td>
<td>0.553%</td>
<td>0.292%</td>
<td>of germline</td>
</tr>
<tr>
<td>Lung cancer</td>
<td>280</td>
<td>150</td>
<td>378</td>
<td>10 197</td>
</tr>
<tr>
<td></td>
<td>0.305%</td>
<td>0.163%</td>
<td>0.412%</td>
<td>0.112% of somatic</td>
</tr>
<tr>
<td></td>
<td>of somatic</td>
<td>of somatic</td>
<td>of somatic</td>
<td>germline</td>
</tr>
<tr>
<td></td>
<td>0.196%</td>
<td>0.207%</td>
<td>0.14%</td>
<td>of germline</td>
</tr>
<tr>
<td>Malignant melanoma</td>
<td>433</td>
<td>228</td>
<td>1 122</td>
<td>65 553</td>
</tr>
<tr>
<td></td>
<td>0.0891%</td>
<td>0.0469%</td>
<td>0.231%</td>
<td>13.5% of somatic</td>
</tr>
<tr>
<td></td>
<td>of somatic</td>
<td>of somatic</td>
<td>of somatic</td>
<td>germline</td>
</tr>
<tr>
<td></td>
<td>0.303%</td>
<td>0.315%</td>
<td>0.416%</td>
<td>of germline</td>
</tr>
<tr>
<td>Acute myeloid leukemia</td>
<td>11 977</td>
<td>2 125</td>
<td>2 051</td>
<td>3 863</td>
</tr>
<tr>
<td></td>
<td>8.03%</td>
<td>1.42%</td>
<td>1.37%</td>
<td>2.59% of somatic</td>
</tr>
<tr>
<td></td>
<td>of somatic</td>
<td>of somatic</td>
<td>of somatic</td>
<td>germline</td>
</tr>
<tr>
<td></td>
<td>8.39%</td>
<td>2.94%</td>
<td>0.76%</td>
<td>of germline</td>
</tr>
<tr>
<td>Prostata</td>
<td>1 190</td>
<td>319</td>
<td>778</td>
<td>16 774</td>
</tr>
<tr>
<td></td>
<td>1.07%</td>
<td>0.287%</td>
<td>0.7%</td>
<td>15.1% of somatic</td>
</tr>
<tr>
<td></td>
<td>of somatic</td>
<td>of somatic</td>
<td>of somatic</td>
<td>germline</td>
</tr>
<tr>
<td></td>
<td>0.833%</td>
<td>0.441%</td>
<td>0.288%</td>
<td>of germline</td>
</tr>
<tr>
<td>Kidney</td>
<td>858</td>
<td>291</td>
<td>542</td>
<td>7 407</td>
</tr>
<tr>
<td></td>
<td>1.09%</td>
<td>0.371%</td>
<td>0.69%</td>
<td>9.43% of somatic</td>
</tr>
<tr>
<td></td>
<td>of somatic</td>
<td>of somatic</td>
<td>of somatic</td>
<td>germline</td>
</tr>
<tr>
<td></td>
<td>0.601%</td>
<td>0.402%</td>
<td>0.201%</td>
<td>of germline</td>
</tr>
<tr>
<td>Sarcoma</td>
<td>850</td>
<td>267</td>
<td>333</td>
<td>5 665</td>
</tr>
<tr>
<td></td>
<td>1.45%</td>
<td>0.455%</td>
<td>0.567%</td>
<td>9.65% of somatic</td>
</tr>
<tr>
<td></td>
<td>of somatic</td>
<td>of somatic</td>
<td>of somatic</td>
<td>germline</td>
</tr>
<tr>
<td></td>
<td>0.595%</td>
<td>0.369%</td>
<td>0.123%</td>
<td>of germline</td>
</tr>
</tbody>
</table>

Table 4.2: Most variants goes into the Very Rare category. In average, Low Frequency have the fewest variants.

By plotting the number of coinciding variants into one graph and the percentages of Somatic overlap over germline variants into another graph we can view the following patterns:

The following trends are visible in Figure 4.12:
- Very Rare have the most variants for 6 out of 7 cancer types.
- Common have the second most variants for 5 out of 7 cancer types.
- Rare have the third most variants for 4 out of 7 cancer types.
- Low Frequency have the least variants for 6 out of 7 cancer types.

The following trends are visible in Figure 4.13:
- Common have the largest share for 5 out of 7 cancer types
- Low Frequency have the second largest share for 5 out of 7 cancer types.
- Rare have the third largest share for 6 out of 7 cancer types.
- Very Rare have the least share for 6 out of 7 cancer types.

Figure 4.12: This plot displays coinciding variants for different classes of allele frequencies for each of the following cancer types: acute myeloid leukemia, breast cancer, kidney cancer, lung cancer, malignant melanoma, prostate and sarcoma. Note that Acute myeloid leukemia stands out from the rest.
4.2.3 Coinciding variants - mutation spectra versus germline allele frequency

Here, we have investigated the distribution of variant types and sequence contexts across different germline allele frequencies. Figure 4.14 displays variant type plots for coinciding and unique somatic variants types of common, low frequency, rare and very rare variants in lung cancer.
Figure 4.14: *A COSMIC versus ExAC comparison which displays variant type plots for common, low, rare and very rare variants in lung cancer. The first row shows the relative frequency of coinciding variant types and unique somatic variants. The second plot visualizes the ratio between coinciding variant types and unique somatic.*

Figure 4.15 displays sequence context plots for common, low frequency, rare and very rare coinciding variants in colorectal cancer, indicating that the mutational or technological mechanisms that underlie coinciding DNA variation (both of which are frequently DNA sequence-dependent) is related to germline allele frequency.

**Figure 4.15: COSMIC versus ExAC comparison. Here you can see a context plot for each allele frequency of colorectal cancer.**
4.2.6 Coinciding variants - DNA sequence context

By looking at the context plots for the different cancer types it is quite clear that C>T variants are the most frequent and T>A are the least frequent. For each of the six variant types the following context variants are the most pronounced: G[C>A]G, T[C>G]T, G[C>T]G, C[T>A]G, A[T>C]A and G[T>G]G respectively. Single variants that stand out are T[C>G]T (15/21 cancer types), G[C>T]G (18/21 cancer types) and G[T>G]G (16/21 cancer types).

*Figure 4.16* demonstrates an analysis of the sequence contexts for coinciding variants found in ExAC and cervical tumor samples (*Figure 4.16*).

![Image of sequence context plots](image)

*Figure 4.16: Sequence context plot for coinciding (upper panel), unique somatic, unique Germline, and enrichment/depletion of coinciding variants in cervical cancer for the comparison of COSMIC versus ExAC.*
4.2.7 Coinciding variants: recurrent versus nonrecurrent

By collecting the numbers retrieved from the tab Overall statistics, a table that displays the distribution of recurrent and nonrecurrent coinciding variants have been set up (Table 4.3) and a plot based on a ratio of nonrecurrent against recurrent variants (Figure 4.17). Acute myeloid leukemia is the single tumor type that differs from the rest, for which the relative fraction of somatic overlap among non-recurrent variants is larger than corresponding value among recurrent variants.

<table>
<thead>
<tr>
<th>Cancer type</th>
<th>recurrent and nonrecurrent</th>
<th>recurrent</th>
<th>nonrecurrent</th>
<th>nonrecurrent/recurrent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breast cancer</td>
<td>20 278 (12.8% of somatic)</td>
<td>1 028 (15.4% of somatic)</td>
<td>19 250 (12.7% of somatic)</td>
<td>~0.83</td>
</tr>
<tr>
<td>Lung cancer</td>
<td>11 005 variants (12% of somatic)</td>
<td>372 variants (13.1% of somatic)</td>
<td>10 633 variants (11.9% of somatic)</td>
<td>~0.91</td>
</tr>
<tr>
<td>Malignant melanoma</td>
<td>67 336 variants (13.9% of somatic)</td>
<td>10 297 variants (16.1% of somatic)</td>
<td>57 039 variants (13.5% of somatic)</td>
<td>~0.84</td>
</tr>
<tr>
<td>Prostate cancer</td>
<td>19 061 variants (17.2% of somatic)</td>
<td>2 618 variants (17.8% of somatic)</td>
<td>16 443 variants (17.1% of somatic)</td>
<td>~0.96</td>
</tr>
<tr>
<td>Sarcoma</td>
<td>7 115 variants (12.1% of somatic)</td>
<td>992 variants (13.5% of somatic)</td>
<td>6 123 variants (11.9% of somatic)</td>
<td>~0.88</td>
</tr>
<tr>
<td>Kidney cancer</td>
<td>9 098 variants (11.6% of somatic)</td>
<td>545 variants (13.7% of somatic)</td>
<td>8 553 variants (11.5% of somatic)</td>
<td>~0.84</td>
</tr>
<tr>
<td>Acute Myeloid Leukemia</td>
<td>20 016 variants (13.4% of somatic, 0.209% of germline)</td>
<td>5 452 variants (13% of somatic, 0.0569% of germline)</td>
<td>14 564 variants (13.6% of somatic, 0.152% of germline)</td>
<td>~1.05</td>
</tr>
</tbody>
</table>

Table 4.3: Distribution of recurrent and nonrecurrent coinciding variants for a set of cancer types. Variants are also shown as percentage share of total somatic variants. Ratio of nonrecurrent against recurrent is shown in far right column.
4.2.8 Mutational signatures within coinciding variants

We investigated the contribution of known mutational signatures among coinciding variants. Here, we analyzed coinciding variant sets from COSMIC and ExAC (any allele frequency), one for each tumor type. Figure 4.21 shows how the different cancer types compared with respect to mutational signature contributions from coinciding variants. Three different mutational signatures are prominent for most cancer types; signature 1, which is a aging-related endogenous mutational process initiated by spontaneous deamination of 5-methylcytosine, and signatures 6 and 15, which are both attributed to a deficiency in the DNA repair system.
Figure 4.18: Relative frequencies of known mutational signatures for a selection of cancer types (COSMIC and ExAC).
5 Discussion and Further work

In this chapter, we will primarily discuss pros and cons related to choices of methods, programming environments, and toolsets, as well as the key file format for DNA variant data. In addition, we will point to some noteworthy limitations of the work that has been undertaken, in particular related to the statistical support within the data analyses of coinciding DNA variation, which we believe should be prioritized in follow-up work.

5.1 Data preparation

The commonly used file format for the representation of genomic variant data is the variant call format (VCF), and it was thus natural to work with this file format. VCF has a comprehensive set of specifications, and is particularly useful for storing a set of DNA variants that is shared or non-shared across a set of individuals/samples, keeping all data related to genomic locus and alleles non-redundant. The VCF format is however a text file format, and will thus the file size will increase rapidly with the number of variants that are represented (and inevitably make the file slower for parsing).

The original VCF files from 1000Genomes, dbSNP, ExAC, COSMIC and ICGC were fairly large and added up to a total of 5.39Gb (compressed by bgzip). The processed dbSNP VCF file took up 5.6Gb when compressed with bgzip and more than 50Gb in the uncompressed state. Iteration of variants in these files was thus time-consuming in the way things were set up. As an example, one loop had to run through more than 160 million variant records, which took about 80 minutes on the standard desktop computer that was used for data preprocessing. Considering also the additional preprocessing steps, i.e. quantification of variant properties and storing these in CSV files etc., the full preprocessing time for dbSNP added up to a total of 11 hours. In retrospect, when it comes to speed of data analysis, we see several rooms of improvement in the data preprocessing workflow, either through parallelization or through the implementation of more sophisticated and efficient iteration/parsing algorithms. It may also be that the Python language was not optimal for this particular purpose, and that choosing another (e.g. more low-level) programming language could have saved valuable processing time. An attempt was made to organise data according
to chromosomes, but this resulted in excessively large data files (the table size had to be multiplied by the number of chromosomes). Attempts were further made to reduce the workload by splitting the quantification process into three modules/scripts (though tightly connected together). The first for iteration and sorting through coinciding and unique variants, the second for setting up and filling CSV tables, and the third for parsing and extracting characteristics from each variant record.

In this project we used the CyVCF library to parse and retrieve variant data from the VCF files, in addition to custom code for analysis of different properties of the variant record (e.g. context, consequence etc.). Although the CyVCF library is considerably faster than the PyVCF, there was recently an update of CyVCF that could speed up things even further (Pedersen and Quinlan 2017). Future VCF parsing should thus preferably be done with the latest library, CyVCF2.

With respect to the determination of coinciding variants, we initially employed an approach that used bedtools intersect. However, owing to the fact that bedtools is not allele-specific, but rather considering genomic segments when it comes to overlap, this did not prove to be a robust approach. The use of vcfanno did however turn out to work extremely well for assessing variant overlap between VCF files, which take the reference and alternative allele fully into account. Moreover, vcfanno turned out to be a much faster alternative compared to bedtools.

In retrospect, we see that it would be useful to remove all non-SNP variants (insertion, deletions, etc.) from the VCF files as a preprocessing step before performing the different annotation processes (e.g. VEP and vcfanno). For instance, the dbSNP database contained approximately 9% non-SNP variants (see table Table 2.1) which were subject to annotation (but not used in other downstream analyses). A simple script that removes these non-SNP variants would thus likely reduce the overall annotation process time.

In this project, the quantities of mutation properties for different data subgroups were stored as comma-separated values. One obvious benefit of using the CSV format is that it will simplify the potential export or interaction with other tools. Furthermore, CSV files can be efficiently read into data frames in R. This particular data type can then be manipulated with a range of other R packages, particularly efficient filtering and extraction operations on rows.
and. The combined size of all the CSV files that were built constituted a total of 820Mb, which was satisfactory for storage purposes.

5.2 Application implementation

We chose the statistical programming language $R$ as the backbone for the application layer. $R$ has suitable libraries and functionality for reading, sorting, aggregating, and manipulating data tables stored within CSV files. The graphics and charting capabilities such as the powerful $ggplot2$ package was also a significant advantage, it made it possible to make highly customized plots for the application. Another advantage with $R$ is the fact that numerous packages have been developed for the analysis of DNA variant data, and in our case we utilized the functions offered by $deconstructSigs$ package to retrieve known mutational signatures from the coinciding variant datasets. One drawback we experienced was the fact that large datasets had to be stored in physical memory during application runtime. The processing of displaying large coinciding variant tables demanded the most memory resources (in-RAM data), and this made the application respond slowly. One example of this is one function where variant tables was meant to be available for download in the coinciding variants table, these tables were without html code for users interested in the table. This function turned out make the application slower when run from server, although this could be because of inefficient code. This function was less of a priority and was temporarily deactivated.

Initially the plan was to stick to using just one programming language, and since Python was heavily used in the data preparation workflow, we were looking for Python web frameworks that could offer interactive data analyses. However, at the time of investigation, we could not find any Python equivalent of R’s Shiny framework. A framework called $pyxley$ was tested but it turned out to be very unstable, and with a general lack of documentation.

The most appealing part of the Shiny framework is that it does not require extensive knowledge or experience in front-end development languages for the web, that is HTML, CSS or Javascript. Shiny turned out to cover all the needed functionality. RStudio also offers a hosting service, shinyapps.io, which simplified the deployment of the Shiny application.

---

7 https://github.com/stitchfix/pyxley
from a local development environment to a web server. Furthermore, Shiny has decent documentation, well-made tutorials, articles and a large community which makes it appealing for beginners. We did however experience that as the codebase for the application grew larger it became difficult to maintain a clear overview and management of the different interacting components. This was particularly challenging for the different inputs and outputs associated with the multiple plots. Understanding how Shiny applications could be developed in a more modularized fashion should thus be a major focus in an improved version of the application.

5.3 Dataset limitations

The user needs to be aware of the fact that some of the datasets have limitations, as an example when comparing COSMIC and ExAC, acute lymphoblastic b-cell leukemia have only have 8 recurrent coinciding variants and 1383 nonrecurrent coinciding variants. This is not enough data to make an accurate analysis about the sequence context variants. If a comparison has too few coinciding or unique variants, we will see that there are too little data to draw a proper enrichment depletion plot (Figure 5.1). In this case a warning will also be displayed stating that the dataset contain missing values.

An obvious further extension to the present analysis is to add statistical support for the variant attribute quantities and comparative analyses that is presented to the user. Having more rigorous statistical support (e.g. through confidence intervals of ratios/fractions etc.) would give better leads as to findings that could be biologically interesting.

![Figure 5.1: This Enrichment depletion plot with missing values.](image_url)
5.4 Reproducibility

It is of general importance that research results made by the software developed in this project can be reproduced by others, i.e. an equivalent application should be able to output the same results by using the same raw data and preprocessing steps in order to replicate the output. In order to ensure reproducibility in all steps of the project, links to all VCF files and application code has been made publically available through GitHub. The project will then be subject to a transparency so that anyone can replicate the methods of processing and go through the application code in order to reproduce the results. By making the source code available on the internet, it also makes it possible for people to come with suggestions for improvement or point out bugs. The source code can be accessed at the following repository: https://github.com/oysor/coincidingSNVAnalyzer

5.5 Case examples

Through several use case examples, we demonstrated that the distribution of coinciding DNA variation between germline and the soma is a complex matter, that depends not only on the tumor type, but also on germline allele frequency and sequence context. Considering that germline variant detection is commonly detected using DNA isolated from blood, it was intriguing that a hematological tumor type (acute myeloid leukemia) contained the largest fraction of somatic variants that coincided with common germline variants. We also discovered that the most common variant type, i.e. C>T:G>A in the CpG sequence context was most enriched in coinciding variants. This particular variant is associated with aging, and represents the most prominent mutational hotspot in the human genome, which may indicate that coinciding variants reflect independent mutational events in the germline and the soma (as opposed to false positives (e.g. noise from technological artefacts)). With respect to mutational signatures, we discovered that both the aging-related signature and signatures associated with DNA mismatch repair deficiencies were present for the coinciding variants in different tumor types. The underlying reasons or mechanisms for the presence of DNA repair-deficient signatures within coinciding DNA variants need to be explored further.
We realize that our interactive web application in its current implementation can give initial leads as to the underlying mechanisms of coinciding DNA variation, and that more sophisticated statistical support is necessary to reach potential conclusions.

5.6 Further work

We see several rooms of improvement and several areas of extension in the current implementation of the interactive web application, some of which we already have mentioned. The main priority will be to add statistical support of existing analyzes. Additionally, we may implement more interactive graphics, e.g. through plotly, which permits even more interactive exploration of the data. Inclusion of other datasets of somatic variants that have been subject to a uniform set of variant calling procedures, such as the The Cancer Genome Atlas, could potentially highlight tumor type differences in a more unbiased fashion. With the exception of the immediate DNA sequence context, the current set of variant attribute quantifications completely ignores the genomic loci of coinciding variants. A natural extension of the current analyses would investigate the chromosomal distribution/clustering of coinciding variants, and contrast the genomic features at these loci with those present at non-coinciding variants.
6 Bibliography

Mclaren, W. et al., 2016. The Ensembl Variant Effect Predictor. Genome Biology, 17(1).


