Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

Algorithms for identifying statistically significant progression patterns in cancer genomes

Student:
Federico Alberton

Advisor:
Prof. Fabio Vandin

Academic Year 2015/2016

Padua, 12/12/2016

Abstract

Recent advances in DNA sequencing methods now allow the collection of mutation information from large number of patients affected from a given cancer type. The identification of patterns of mutations in these patients has been previously shown to identify mutations that are important for the development of the disease. However, such patterns usually ignore information regarding the progression status of mutations in patients.
This work formalizes the problem of the identification of significant progression patterns from cancer genomics data and proposes algorithms to find such patterns. The proposed algorithms are based on Monte Carlo methods to find statistically relevant patterns using a probabilistic model that use tail distribution bounds, like the Chernoff bound and the Azuma-Hoeffding bound, to focus the computation on the most promising patterns. The proposed approach has been tested on real cancer data from The Cancer Genome Atlas project.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Cancer is an highly adaptable process that takes the advantages of the Darwinian evolution in order to resist to therapy and propagate in other sites of the afflicted organism. Nowell [14] provided the interpretation of such defining what nowadays is know as the Clonal Theory of cancer development where each cancer cell can be viewed as coming from a common ancestor cell where the first driver mutation happened then by a Darwinian like process multiple population of cancer cell, called subclones, are created and compete between them in order to take the control of the future cancer development.
Classifying mutations according to their clonal status can provides information about the time line of the cancer evolution giving indications on the timing of the mutational process. The presence of subclonal mutations can interfere with the drug therapy used as certain subclonal driver mutations lead to an increased resistance to specific drugs. Subclonal mutations can also be targets for a sort of immunotherapy. Hence the needs to also adjust the drug therapy properly.

Starting from a criterion used to classify one genetic mutation as clonal or subclonal [11] we have developed a greedy algorithm based on Monte Carlo methods that is used to find statistically significative cancer progression patterns including clonal information of multiple genes starting from cancer genomic reads.
Firstly we defined a way to summarize the information of such reads using a mutation matrix. After that we provided a way to find statistically significative couple of mutated genes that are classified as clonal and subclonal respectively using a statistical algorithm. Then we generalized such method in order to be applicable to couple of sets of mutated genes where the first set is classified as clonal and the second and subclonal proposing different efficient approaches. The first approach try to reduce the computation time applying interval logic considerations in order to avoid useless computations while also introducing a methodological framework to better characterize the computational problem. The second approach use the previous framework while using a probabilistic model that use tail distribution bounds like the Chernoff bounds and the notion of martingales and hence the Azuma-Hoeffding bound.
We then tested the derived approaches on a dataset of patients afflicted by a brain tumor called Glioblastoma Multiforme (GBM) extracted from The Cancer Genome Atlas project [2].

1.1 Structure of the thesis

The work follows the following structure:

- In Chapter 2 we give an introduction to certain basic molecular biology notion. We illustrate some important structure of a cell that are involved in the synthesize of protein, we then see how a cell know what components are need in order to create a protein and how it is actually created. After that we introduce what is a genetic mutation while introducing some genomic's terminology that is commonly used in bioinformatics. We then distinguish *passenger* and *driver* mutations while introducing the Clonal Theory. We also briefly introduce how from a set of cancer genomic reads information about somatic mutations are extracted.

- In Chapter 3 the mathematical instrument used in Chapter 4 are briefly introduced. We start introducing the concept of statistical significance relative to the hypothesis test, then we see how randomness is exploited to solve problems that are otherwise not trivially solvable with a deterministic approach and finally some important probability results that are at the base of randomized algorithms analysis are presented.

- In Chapter 4 we try to respond to the necessity of finding significative mutation patterns according to the Clonal Theory providing a statistical algorithm based on Monte Carlo methods that try address the exponential complexity in the number of gene that are taken in consideration by adopting a greedy approach.

- Finally in Chapter 5 we are going to provide and comment the results gathered from the application of the approaches previously illustrated.

Chapter 2

A biology primer

In this section we are going to give a very brief introduction to molecular biology in order to understand genomics based of the work of [7, 6].

Human's life depend on three types of molecules: DNA, RNA and proteins. The DNA holds information regard how such cell works while RNA is used to transfer part of such information in other cell places where they are used to synthesize proteins.
After years of biological experiments is now known that DNA, RNA, proteins are encoded by particular alphabets: the DNA is coded by a four letter alphabet (A, C, T, G) where each letter represents a small molecule called *nucleotide*, the RNA share the same alphabet of DNA except the fact that the nucleotide *thyamine* that represents the symbol T is substituted by the nucleotide *uracil* symbolized as U, finally proteins are encoded by a twenty letters alphabet representing *amino acids*.

2.1 Structure of a cell

Cells of Eukaryotes (organisms where each cell has a Nucleus) are constituted by various parts, the most interesting ones for our purposes are briefly introduced.

Ribosomes

Ribosomes are cell structures where proteins are assembled starting from more elemental molecules called amino acids and they are composed by proteins and nuclei acids. They synthesize protein starting from the a set of instruction (the list of amino acids that form the protein) given by the the mRNA by linking specific amino acids.
They are composed by two parts, the first one's role is the translation of the instruction contained in the mRNA while the second one gather the required amino acids and link them together.
Is interesting to notes that the difference between the ribosomes of Bacterials and Eukaryotes are exploited to make antibiotics such as Streptomycin that prevent the synthesize of proteins in Bacterial's ribosomes while not affecting the Human's ones.

Nuclei acids

All the genetic information of any individual are stored in nuclei acids, the DNA and the RNA, which are polymers of four different acid units called nucleotides.

Each nuclei acid consist of a three parts: the first part is a sugar, the second part is made off a purine or a pyrimidine, and the last one by a phosphate group which contains the element phosphorus.
In the DNA the purine nucleotides, (adenine, cytosine, guanine, thymine), are the elements that characterize every DNA molecule. For such fact nucleotides are called *bases* and are abbreviated in letters: A,C,G,T.
The RNA differs from the DNA by having a little different bases A,C,G,U and by the type of sugar, ribose in RNA and deoxyribose in DNA.
The length of a sequence of DNA is measured in its number of bases, usually the length is measured in thousand of bases abbreviated as *kb*.

Nowadays we know that the DNA forms a double helix constituted by two helical strands of polypeptide that runs in opposite direction. Is fundamental to note that the information that are in one strand are duplicated in the second strand. This property is given by the fact that adenines can bond only with thyamines and guanines can bond exclusively with cytosines. This fact is also fundamental in the protein synthesis as it is used to copy the part of the genetic information that are used to make the specific protein.

DNA molecules can also vary their shape for example from the classic double helix they can reverse the direction of its twist. Such shapes can have a role in turning on and off genes.

Proteins

Proteins are molecules that do a multitude of function in a cell: they give structural support in order to hold a creature together, the are used as *enzymes* in chemical reaction controlling it, they can be used to turn on or off a particular gene, and many more. They make up between the 15% and 20% of the cell [6].

Proteins are all constituted by amino acids small molecules constituted by a central atom of carbon (C), an amino group (NH_3) and a carboxyl group (COOH) and variable region. Chains of amino acids are created by a particular chemical reaction. In the DNA we can find which particular sequence of amino acids composes such protein, this sort of amino acid signature is called *primary structure*. This can be done because the nucleotides that form a DNA molecule identify unambiguously a specific amino acid. For each amino acid we know its chemical properties and such information are commonly used to predict the structure of protein. Is interesting to note that in all eucaryotic proteins that were synthesized in a ribosome they start synthesizing with the amino acid methionine.

Generally a protein contains a number of amino acids that can vary, most part of proteins are composed by more than 100 amino acids. After looking the structural contribute given by the amino acids to a protein we can look at the *secondary structure* of a protein as the three dimensional form of a few amino acid sequence. Given this an interesting open problem is to predict the structure and the function of a protein starting from its *primary structure* as there was investigated that the amino acids in the *primary structure* of a protein determine the protein final role.

In computational biology an interesting problem is the *protein folding problem.* As we said that in the DNA we can only find information about the *primary structure* nothing we know about the shape that a protein is going to assume after being synthesized by a ribosome. Usually after being synthesized a protein takes a few seconds to folds up into a specific shape that influences its biochemical function and when the protein take the particular shape that permit the accomplishment of its native function the protein is said to be in an *active conformation* while being in another shape the protein is said to be *denatured.* The *protein folding problem* can then summarized as finding a way to map the *primary structure* of a protein to a *tertiary structure* that represents the location of the atoms that constitute such protein.
This is also complicated by the fact that not all proteins can fold themselves without any help, some proteins need the help of other molecules called *chaper-ones* in order to be folded up correctly. Also some proteins in order to become active they need to bind with copy of themselves or with other molecules called *prosthetic groups*, for example the protein hemoglobin needs the *prosthetic group* heme in order to binds the oxygen that is in the blood. Such additions are refereed as the *quaternary structure* of the protein.

An interesting fact is that a protein assumes a form that minimize the total *free* energy. Given the position of the atoms is possible to compute the *free* energy. We can then solve the *protein folding problem* as an equivalent optimization problem where given the atoms of the protein we wold like to find the disposition that yields the minim *free* energy, than this solution can be used as the predicted *tertiary structure* of the protein. Unfortunately the input size of this optimization problem is very large and calculating the *free* energy of a disposition is a computationally demanding task.

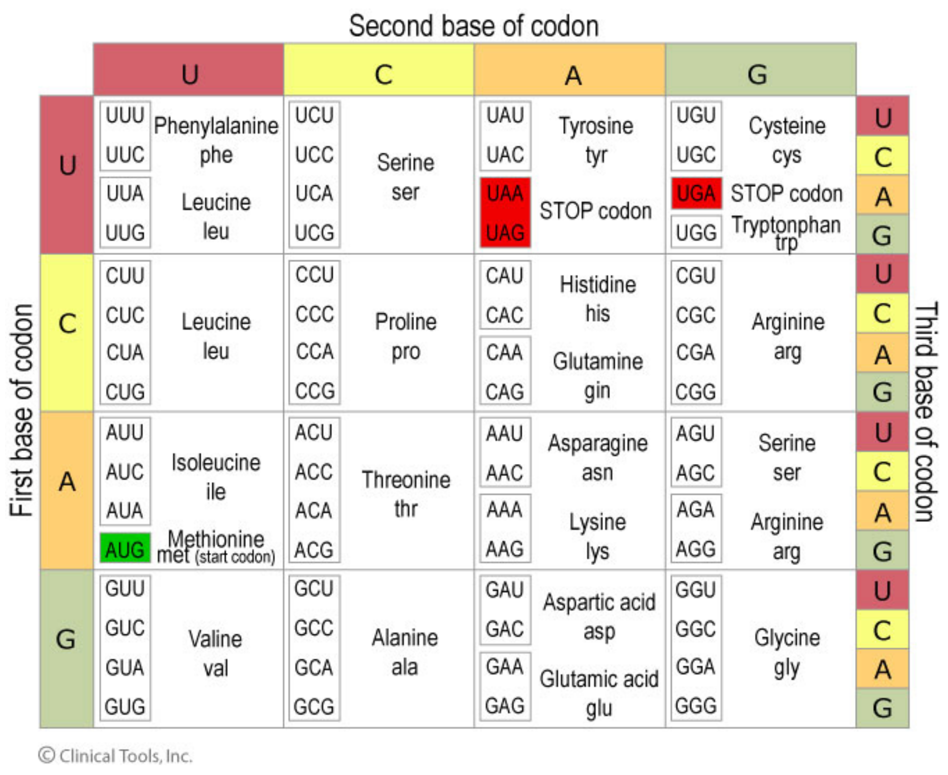
2.2 The Central Dogma

Francis Crick in the 1958 coined the term *Central Dogma* in order to refer to the flow of genetic information and more specifically to the fact that the flow of information start from the DNA then pass to the RNA and finally from RNA to proteins. With the term *genome* we are referring to all such information.

2.2.1 The code of life

In this section we will illustrate how the information contained in the DNA is encoded.

The genetic information of an individual are stored in DNA molecules called *chromosome.* In humans and other organism called *diploids* each chromosome contains two similar DNA molecules one for each parent while in other called *haploid* each chromosome contains a single DNA molecules. Humans have 23 pairs of linear chromosome. Each triplet of DNA string is called *codon*, such triplet is used as a map to one of the twenty amino acids. Given this we have that with a 3 sequence of nucleotides we have $4^3 = 64$ different triplets but we have only 20 amino acids to map, this fact translates that we have some redundancy as for example the amino acid arginine is coded by four different codons (CGU, CGC, CGA and CGG) that differs only by the last letter of the triplet, this property is know as the *degeneracy* of the code. Also the triplets UAA, UAG and UGA do not represent any amino acid, they are know as *stop codons* because they are used to represent the end of a protein sequence. In eucaryotes all proteins that are synthesized in a ribosomes start with the amino acid methionine whose coding triplet is AUG and it is called *start codons* because we can interpreter AUG as the starting code of a protein. The figure 2.1 contains the DNA to amino acid map.



First base of codon	Second base of codon: U		C		A		G		Third base of codon
U	UUU	Phenylalanine phe	UCU	Serine ser	UAU	Tyrosine tyr	UGU	Cysteine cys	U
	UUC		UCC		UAC		UGC		C
	UUA	Leucine leu	UCA		UAA	STOP codon	UGA	STOP codon	A
	UUG		UCG		UAG		UGG	Tryptonphan trp	G
C	CUU	Leucine leu	CCU	Proline pro	CAU	Histidine his	CGU	Arginine arg	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	Glutamine gin	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	Isoleucine ile	ACU	Threonine thr	AAU	Asparagine asn	AGU	Serine ser	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	Lysine lys	AGA	Arginine arg	A
	AUG	Methionine met (start codon)	ACG		AAG		AGG		G
G	GUU	Valine val	GCU	Alanine ala	GAU	Aspartic acid asp	GGU	Glycine gly	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	Glutamic acid glu	GGA		A
	GUG		GCG		GAG		GGG		G

Figure 2.1: The genetic map from a triplet of nucleotides to an amino acid. Source: le.ac.uk

In most eucaryotes the DNA sequences for a specific protein have some noncoding subsequence called *introns*, the coding ones are called *exons*. While most of the eucaryotes have introns most bacteria do not have any.

2.2.2 Transcription and Translation

In this section we illustrate how the information that comes from DNA are transferred to ribosomes and how ribosomes collect and "fuse" the required amino acids for a given protein.

After having decoded how the DNA map a triplet of nucleotides to an amino acid we need a way to copy the sequence of triplets for a given protein in order to give such instruction to the ribosome. This part is called *transcription* and a specific portion of DNA is copied into a messenger RNA molecule called *m-RNA* (messenger-RNA). This process start with a molecule called *RNA polymerase* that bind to the DNA where it catalyzes a reaction where the DNA is used as a template in order to create a complementary strain of RNA called *primary transcript* that contains exons and as well introns. In the next step called splicing introns are removed from the primary transcript and exons are spliced together in order to form what is called *mature RNA*. Finally the mature RNA is transported out of the nucleus in order to bind to a ribosome.

The next step takes place in the ribosomes and is called *translation*, the mature mRNA is used as a fingerprint in order to synthesize proteins. In order to assembly a given protein we need to collect the required amino acids encoded by the codons in the mRNA, this part is done by other molecules called *transfer-RNA* or *tRNA*. A tRNA molecule is constituted by two parts: the first part called *anti-codon* attach to a specific codon of the mRNA and the second part bring the corresponding amino acid into physical proximity. After that the required amino acid form a peptide bond with an another of the required ones. How the tRNA find the correct amino acid depends on the three dimensional structure of the RNA. And finally we obtain the required protein which will folds up on its own or requiring the help of other molecules, the folded protein may need to bind with other molecules in order to become active as seen before when speaking about the *quaternary structure*. After being activated the protein it is transported in the parts of the cell where its function is needed.

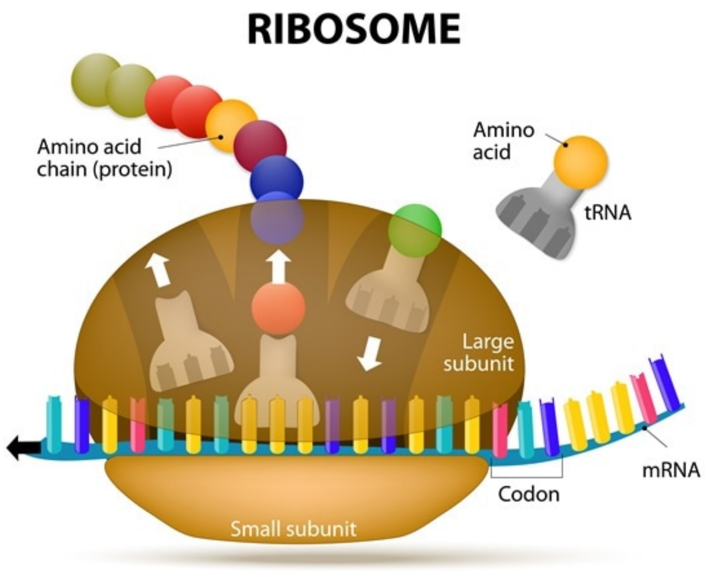


Figure 2.2: How a protein is assembled in a ribosome starting from the mRNA. Source: news-medical.net

2.3 Genetic mutations

In the previous sections we have seen that most of the important function of an individual are performed by proteins and the connection that there is between the coded DNA and them. We now investigate what happens when during the process of cell reproduction the DNA of the daughter cell differs from the one of its progenitor. Clearly sometimes this can comport to alterations in important function while other times there are not important consequences.

A genetic mutation is a permanent alteration of the DNA of a cell. Mutations are one of the main cause of diversity among organisms [13].
Such alteration can alter the structure of the DNA in various method for example given a sequence of DNA: a single nucleotide can be substituted with an another one (substitution mutations), one or more nucleotides can be added to the sequence (insertions), one or more nucleotides can be removed from the sequence (deletions).This alteration are called single-nucleotide variants (SNVs) Other more more complex mutations lead to genome rearrangements and are called structural variants (SVs) while other called copy-number aberrations (CNAs) can lead the genome to be constituted by repeated parts.
Generally mutations can be classified in two types:

Germline mutations are mutations that are genetically acquired from parents. They are called germline mutations because they are present in the germ cells of parents (egg or sperm cells). If the fertilized egg has a mutation all the cell of the resulting individual carry such mutation.

Somatic mutations are mutations that do not involve all the cell of the individual and only if a germ cell is affected they can be passed to the offspring. This type of mutations arise during the individual's life.

Given a mutation we can also give a classification based on the effect that the mutated gene has on its encoded protein [25]:

Frameshift mutation is caused by an insertion or a deletion of a number of nucleotides that is not a multiple of three. This can completely change the sequence of codons (given a sequence of nucleotides imagine to start reading the codons from the second nucleotides instead of the first one of the sequence) arising the synthesize of a different protein. If the number of insertion or deletions is a multiple of three the mutation is called in-frame mutation.

Nonsense mutation is the results of mutation that cause the appearance of a premature stop codon in the sequence giving life to a truncated protein.

Missense mutation are single nucleotide variations where a mutated codon codes a completely different amino acid giving the synthesize of not functional protein.

Neutral mutation is like a missense mutation but the mutated codon codes a chemically similar amino acid and such similarity does not disrupt the protein's functionality.

Silent mutation do not changes the *primary structure* of the protein, they can occur in non coding zones like *introns* or they do not alter any amino acid in *exons*.

Germline mutations can be the cause of certain inherited diseases for example Cystic fibrosis but somatic mutations can give birth to cancer phenomenons. The source of somatic mutations are multiple as they can arise from environmental source or from the chemical instability of the DNA: due to oxidative stress or due to error in the DNA repairing process. Environmental source are for example tobacco smoke, UV light and ionizing radiation.
Most somatic mutations that arise during a person's life are innocuous or even benign, single nucleotide substitution for example occur at a rate of 10^{-9} [21].
Somatic mutations are important as according to the *Knudson hypothesis*[1] cancer is interpreted as the results of an accumulated number somatic events.

2.3.1 From reads to mutations

In practice [17] we start from a set of *reads* extracted from a cancer genome, a sequence of a certain number nucleotides (that can be as little as 50 to tens of thousands of nucleotides in based to the sequencing technology used, usually reads small is size are used), then such reads are aligned to the human reference genome and difference from the reference genome are identified. In order to distinguish a mutation from being somatic or germline in parallel a normal sample is also analyzed.
Usually a tumor sample contains both normal cells and different type of cancerous cells. The tumor purity of a sample gives an indication about the composition of the tumor sample as it is defined to be the fraction of cells in the sample that are cancerous. Now a problem can rise in identifying somatic mutations as lowering the sample's tumor purity (equivalently increasing the sample's number of normal cells) reflects in a reduced number of reads that come from the cancer cells and hence a reduction in the efficacy to identify a somatic mutation.

2.4 Driver mutations and the Clonal Theory

Given a generic cancer genome we can find hundreds to thousands of somatic genetic alterations, most of them are just *passenger* mutations while only a small part of them called *driver* mutations are important for the development of a cancer mass. Generally a number ranging from 10 to 20 mutations per tumor can be considered as *driver*.

One of the most significative work in the cancer research is the article[14] published by Peter Nowell in the 1976 that create the *clonal evolution* theory of the cancer development where cancer can be viewed as an evolutionary process similar to the Darwinian selection where somatic mutations create a pool of cancer subclones competing for their evolution and where the *fittest* subclone can have an advantage over other subclone

[1]Firstly proposed by Carl O. Nordling in 1953 and later formulated by Alfred G. Knudson in 1971. This was inferred from the fact that there is a correlation between the cancer frequency f and the age a raised to a certain power x, mathematically $ln(f) = ln(a^x) + c$ with c a fitting constant.

predominating in the tumor mass development.
The evolution of such subclones is driven by selectively advantageous *driver* events and *passenger* mutation. Also cancer development takes place in a limited resource environment and this fact explain why the cancer cell doubling time is order of magnitude faster than the tumor doubling time as probably the resource limited environment limit the survival of the cancer cell, resource are used in a competitively way between various subclones. Such enviroment is not static it responds to the cancer evolution like a sort of a feedback system and can reduce or augment the advantage of certain subclones respect to other ones, for example protecting some subclones against therapy. This sort of cancer ecosystem can be altered by chemotherapy or by radiotherapy and this give to possibility for some cancer cells that were on a selective disadvantage to emerge.

McGranahan et. all. [11] defined a way to classify a gene mutation as clonal or subclonal and then studied the evolution of cancer on nine different types of tumors based on data from The Cancer Genome Atlas. We are using this classification criterion in Chapter 4.

2.5 Further topics

There are other important topics that need to be addressed such as: how a cell pass its DNA to the offspring, how a cell know the quantity of protein that is needed, how proteins accomplish their roles (which is the role of an *enzymes* in chemical reactions), how multiple protein can collaborate in order to perform important functions in organisms (*pathway*).
For further reading we remand to [6] and [7] for the computational aspect of biology and to [3] for a college level book on biology. We conclude with figure 2.3 giving a summary of the first two sections.

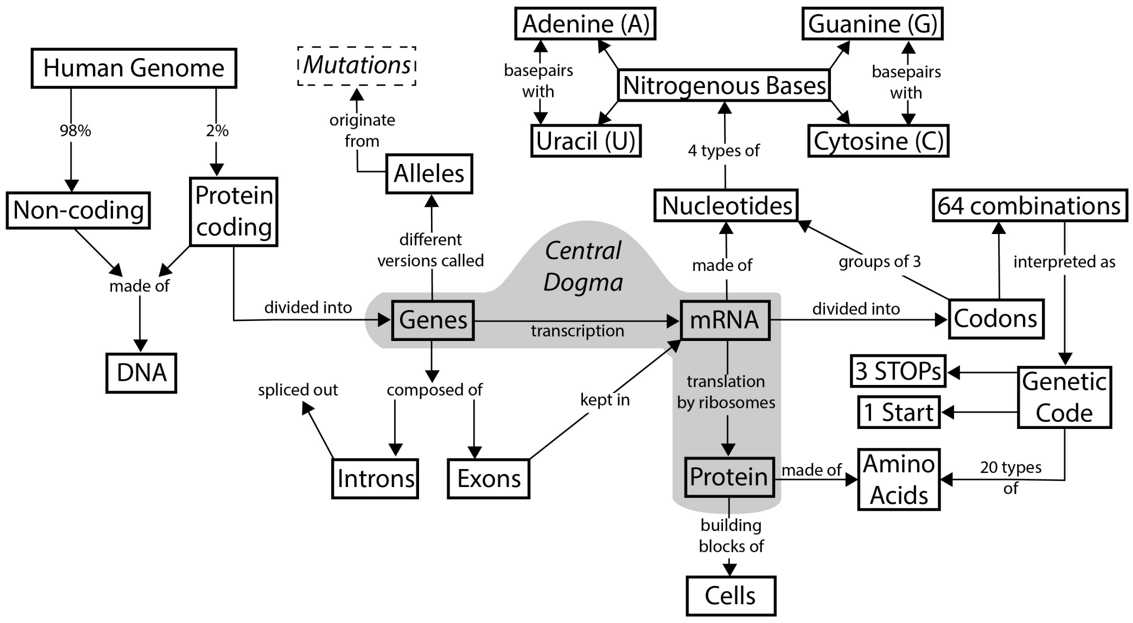


Figure 2.3: Concept map giving a summary of the first two sections of this Chapter. Source: coursera.org

Chapter 3

Mathematical tools

In this chapter we are going to introduce some mathematical concept that are used in Chapter 4 in order to develop our algorithmic methodology: the *hypothesis testing*, the *Monte Carlo methods*, the *Chernoff bound* and *Martingales*.

3.1 Statistical testing

In this section we introduce a method that uses data in our possession in order to refuse an hypothesis in the case the data induces us to think that probably such hypothesis is false. The statistical testing is a common used tool in a vast range of disciplines from medicine to psychology.

A statistical test provide a way to asses the significativity (in a statistical meaning) of a given starting hypothesis usually called *null hypothesis* and permits to refuse such hypothesis in order to accept an alternative one called *alternative hypothesis* [19].

Generally this type of test involve the following steps:

1. Make an initial claim.

2. Collect evidence from the set of data.

3. From the previous collected evidence decide, if it is legit, to refuse the initial claim.

Going a little more in depth we conduct a hypothesis testing following the follow procedure[8]:

1. We start conjecturing a fact that can be either true or false.

2. Next we setup the *null hypothesis* and the *alternative hypothesis.* This part must be done with care as misstating such hypothesis can invalid the entire process.

3. After that we can make statistical assumption about the sample like statistical independence or about the form of the distribution of the samples.

4. At this point we can choose an appropriate test and choose a *test statistic* T which is a value derived from the sample that can be viewed as a numerical summary of the information expressed by the observed data.

5. Now under the *null hypothesis* stated in the point 2 we derive the distribution of the test statistic T, "some times" the distribution of T follows a *normal distribution* or a *t-distribution* or others well know distributions.

6. We then select a threshold α generally called *significance level* that represents the probability level where below we would like to reject the *null hypothesis.* Usually in literature a value of $\alpha = 0.05$ or $\alpha = 0.01$ is chosen.

7. Now after choosing α and under the *null hypothesis* we have that the distribution of the values of T can be partitioned in two parts: the first part is constituted by the set values such that we accept the *null hypothesis*, the second part usually called *critical region* consists in the values for which we are going to reject the *null hypothesis* and accept the *alternative* one. This value partition is a function of α and represents the probability that for a given value we reject the *null hypothesis* and accept the *null hypothesis.* Lowering the parameter α restrict the *critical region.*

8. We then calculate the observed value t of the test static T.

9. And finally we can now decide if we can refuse our *null hypothesis* and accept the *alternative* one. We decide to reject the *null hypothesis* if the observed value t falls into the *critical region* otherwise we can conclude that there are not sufficient evidence for reject *null hypothesis.*

This approach is usually labeled as the *critical value approach* and is usually favored in old texts when only the tables of test statistic distribution at common probability threshold were usable.

Usually in journals, in statistical software or more generally in research an alternative methods is preferred, such method introduces the notion of p-value.
The *p-value approach* can be resumed in the following steps:

1. Given again a test statistic T we find out the observed value t from the observations.

2. We then setup a *significance level* α similarly as done before.

3. For the next point we need to calculate the p-value which is the probability of obtaining a value equal or more extreme than the observed value when the *null hypothesis* is assumed to be true.

4. If the p-value is not higher than the *significance level* α we then can reject the *null hypothesis* and accept the *alternative* one otherwise there are not sufficient information that induce us to think that the *null hypothesis* is false.

This latter *approach* has the advantage to give us more information rich results but usually requires a computational support.
Generally is common to valuate the data respect to the obtained p-value in an empirical way, more specifically:

if p-value ≥ 0.05 than data are claimed to be *not significant*,

if p-value ≤ 0.05 than data are claimed to be *significant*,

if p-value ≤ 0.01 than data are claimed to be *highly significant*.

Usually the *null hypothesis* correspond to the not interesting event and the *alternative hypothesis* correspond to the interesting one. For example if given a coin we are interested in investigating if the coin is biased toward head we then setup the *null hypothesis* to be "the coin is not biased toward head" and the *alternative hypothesis* to be the complementary event "the coin is biased toward head". In such way the lower the p-value the more we are inclined to refuse the fact that the coin is fair and accept the fact that instead it is biased.
Generalizing we have that the smaller is the p-value and larger is the significance as the data induce us to contradict our *null hypothesis*.
Is important to note that the *significance level* α is not derived from the data and does not depend on the hypothesis chosen as such value is derived from the consensus of the researcher in the specific research area.

Previously we omitted to define when "a value is equal or more extreme then the observed value". Based on how we interpreter such propriety we have different cases. Let t to be observed value and let X the random variable associated to the test statistic than this is related to the following events: $\{X \leq t\}$ called *left-tail event* or $\{X \geq t\}$ called *right-tail event* or we are referring to the *smaller* event among $\{X \leq t\}$ and $\{X \geq t\}$ called *double tail event*.

Let H_0 to be the *null hypothesis* and p to be the p-value then we have that this latter can be defined for each of the previous events as:

- $p = P[\{X \leq t\}|H_0]$,
- $p = P[\{X \geq t\}|H_0]$,
- $p = 2 \times min(P[\{X \geq t\}|H_0], P[\{X \geq t\}|H_0])$.

Figure 3.1 illustrate the relation between the p-value and the observed value t.

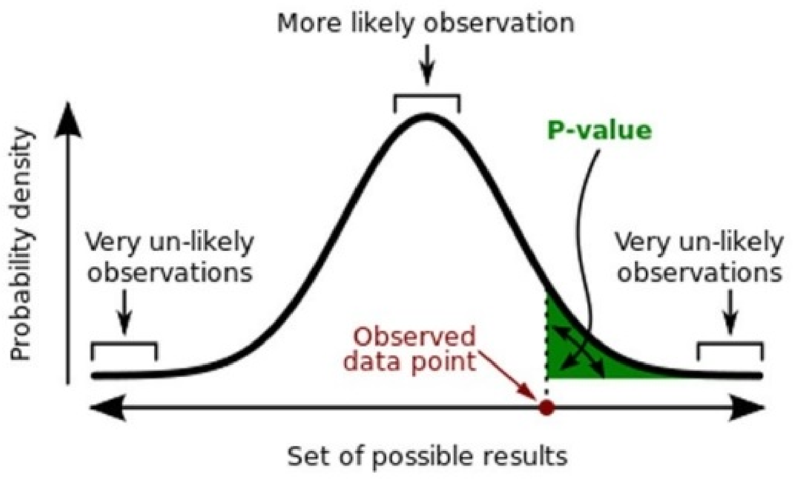


Figure 3.1: How the p-value is calculated.

On the x-axis we have the possible outcomes where in the y-axis we have probability density of an outcome when the *null hypothesis* is true. Source: wikipedia.org

Following such procedures we have not proved that the *null hypothesis* is true or either that the *alternative hypothesis* is true as we make our decisions based on the evidence instead of certain guarantied proof. In other terms if we reject the *null hypothesis* we do not prove that the *alternative hypothesis* is true and if we do not reject the *null hypothesis* we do not prove that the *alternative* hypothesis is false. From the above considerations we can see that there is always a chance that the conclusion we draw is erroneous, the table 3.1 illustrate the decision landscape involved.

	H_0 is true	**H_1 is false**
Do not reject H_0	OK	Type II error
Reject H_0	Type I error	OK

Table 3.1: Types of error in a statistical test.

With H_0 and H_1 we are referring to the *null hypothesis* and respectively to the *alternative hypothesis* as in literature those nomenclature is often used.

As we can see from the table 3.1 we can commit two different errors, in statistics they are know as:

Type I error: the *null hypothesis* is rejected when it is true.

Type II error: the *null hypothesis* is not rejected when it is false.

We then conclude this section providing an example on where the statistical test is used.

Coffee aroma tasting

In 2014 a team of researchers composed by members from the University of Oxford and from the Federation University of Australia published an article where they investigated if the aroma of a cup of coffee was influenced by the color of the coffee cup used to drink [20, 18]. The coffee's aroma was rated on a hundred point scale. 12 people were randomly selected to drink a cup of coffee from a cup with a white sleeve and other randomly selected 12 people drunk the coffee from a cup with a blue one. The first group rated the coffee 57.33 on average with a standard deviation of 16.27 points while the average rating from the second group was 35.57 with a standard deviation of 25.34. We would like to see if the color of the cup influences the perceived aroma of the coffee. Firstly we setup our *null hypothesis* H_0 and *alternative hypothesis* H_1:

H_0 The color of the cup does not influence the average aroma rating (the two means are the same).

H_1 The color of the cup influences the average aroma rating (the two means are not the same).

Now we choose an appropriate test statistic T, we need to compare two means and we do not know nothing about the variance so in this case we use a t-test. Let w and b to be the average from the samples and let s to be standard error of the difference between the two means under the hypothesis that samples are independent and distributed like a normal distribution we can choose T:

$$T = \frac{w - b}{s}. \tag{3.1}$$

T follows a t-student distribution which is a well know probability distribution. Now we calculate the observed statistic t. We can calculate s as $s = \sqrt[2]{w_e^2 + b_e^2}$ were w_e and b_e are the standard error of the mean calculated as the standard deviation divided by the square root of the number of samples, more specifically $w_e = \frac{16.27}{\sqrt[2]{12}}$ and $w_e = \frac{25.34}{\sqrt[2]{12}}$ which give $s = 8.7$ and then $t = \frac{57.33-35.57}{8.7} = \frac{21.76}{8.7} = 2.5$.
Now we decide to use $\alpha = 0.05$ as significance level and we consider a *double tail event* as the difference of means can vary with in two different direction. Let p to be our p-value we have that:

$$p = 2 \times min(P[T \leq 2.5] + P[T \geq 2.5]) = 2 \times min(0.01, 0.01) = 0.02 \leq a = 0.05. \tag{3.2}$$

We then decide to reject H_0 and accept H_1 as the data do not sufficiently support H_0. Then we can conclude that from the given data emerges an association between the color of the cup used and the perceived aroma coffee.

3.2 Monte Carlo methods

Monte Carlo methods [1] [12, 10, 15] refers to a class of computational algorithms that rely of repeated random sampling in order to solve computational problems that in a

[1]The Monte Carlo method was developed in the '40 in the context of the atomic bomb at the Los Alamos National Laboratory and had a critical importance in the Manhattan Project.

deterministic way are not easy to approach as finding a numeric or an analytical solution is difficult or impossible. They are used in vast range of scientific and engineering problems due to their flexibility and ease of implementation.
They are commonly used in optimization problem where they can permit to avoid to get local optimum solution by random exiting from the local optimum in order to find the global one. They are used in solving numerical analysis problems such in the numeric integration were random points are used to approximate the integral of a given function in a specific interval. We can also use them when we need to get sample from a given probability distribution.
Generally to apply a Monte Carlo method for statistical analysis following these steps:

1. We determinate the statistical properties of the possible inputs.
2. Then generate many sets of possible inputs that follow the previous statistical properties.
3. We then can performs some deterministic computation on those data.
4. And at the end we aggregate the previous results usually performing a statistical analysis.

A classical example in the numeric integration usage is the estimation of the value of π using a Monte Carlo method.

Let X and Y two uniform random variable in the interval $[-1, 1]$ we then represents a point that belong to a 2×2 grid center in the origin $(0, 0)$ as (X, Y). We now consider a circle with radius $r = 1$ centered in $(0, 0)$, we have that this circle is inside the 2×2 square and has a total area of π. Now we can define an indicator random variable Z associated to the point (X, Y) that indicates if (X, Y) is inside the circle or not:

$$Z = \begin{cases} 1 & \text{if } \sqrt{X^2 + Y^2} \leq 1, \\ 0 & \text{otherwise }. \end{cases} \tag{3.3}$$

Now we have that if (X, Y) is chosen uniformly at random we that:

$$P[Z = 1] = \frac{\pi r^2}{(2r)^2} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} \tag{3.4}$$

which is the ratio of the area of the circle to the area of the square. After choosing n random points (X_i, Y_i) and defining Z_i as the indicator random variable Z relative to (X_i, Y_i) we have that:

$$E[\sum_{i=1}^{n} Z_i] = \sum_{i=1}^{n} E[Z_i] = \frac{n\pi}{4}. \tag{3.5}$$

Then we can conclude that we can obtain an estimation of *pi* using the following formula supposing we know $z = \sum_{i=1}^{n} z_i$, where z_i is the value of Z_i:

$$\pi = 4\frac{z}{n}. \tag{3.6}$$

Is important to note that the simulation relies on a large number of sample and on uniformly distributed random numbers, whose generation is not a trivial task, in order to be reliable. Is interesting to note that we can obtain a more precise approximation of π as more sample we use, this fact can be analytically proved using the Chernoff bound [12] presented in the next section.

3.3 Tail distribution bounds

Given a set of random variables $X_0, ..., X_n$ we are sometimes interested in a random variable that is a function of the previous value $\bar{X} = f(X_0, ..., X_n)$.
In the next subsection we firstly introduces a classic bound on the probability of $\bar{X}$ where $\bar{X} = X_0 + ... + X_n$ with X_i an indicator random variable and assuming that such variables are independent. We then introduce the notion of martingales where trough the Azuma-Hoeffding inequality we can give a more general bound that do not require the independence of the random variables that are involved.
Analytic bounds on the tail distribution are fundamentals in the analysis of randomized algorithms.

3.3.1 Chernoff bound

This family of bounds [5, 12] on the tail probability is derived from using the Markov's inequality on the moment generating function of a random variable. For a random variable X the associated moment generating function is $M_X(t) = E[e^{tX}]$. Such function have some interesting qualities that are exploited in order to derive such bound for example under specific conditions the nth derivate of $M_X(t)$ evaluated at 0 is equal to the nth momentum of the random variable X, $E[X^n]$. A Chernoff bound give exponentially decreasing bounds on the tail distribution. We used the term family as there are different bounds all derived from using the moment generating function under specific hypothesis.

We define a Poisson trials to be a series of independent experiments where each experiment has a probability of success that can be different from the other experiments. Under such definition a Binomial random variable can be seen as a Poisson trials where each experiment has the same probability to success.

Let $X_1, ..., X_n$ be independent Poisson trials such that $P[X_i] = p_i$. Let $X = \sum_{i=1}^{n} X_i$

and $\mu = E[X]$. Than the following Chernoff bound hold:

$$P[X \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mu} \text{ for any } \delta > 0. \tag{3.7}$$

$$P[X \leq (1-\delta)\mu] \leq \left(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\right)^{\mu} \text{ for any } 0 < \delta < 1. \tag{3.8}$$

$$P[X \geq (1+\delta)\mu] \leq e^{-\frac{2\delta^2\mu^2}{n(b-a)^2}} \text{ for any } \delta > 0, \text{ if } a \leq X_i \leq b.\ \forall i = 1, ..., n. \tag{3.9}$$

$$P[X \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2\mu^2}{n(b-a)^2}} \text{ for any } \delta > 0 \text{ if } a \leq X_i \leq b.\ \forall i = 1, ..., n. \tag{3.10}$$

For example referring to the Monte Carlo Methods section where we have calculated an approximation of π to we can now apply a Chernoff bound in order to obtain an upper bound on the probability that the approximated value found differs from the true value of π by some amount.

3.3.2 Martingales

The term martingale [12, 16] derives from a set of strategies used by a group of french gamblers during the 1700s [24]. One of those strategies applies to a game where if a coin shows up an head the gambler win an amount of money otherwise he loose its bet. The strategy consist in whenever the gambler loose a game to double his bet so when the first victory approaches he can repay back all the money lost in the previous game and win an amount of money equal to his initial bet. Apparently this strategies seems to work as in the long run the probability of scoring a victory approaches to 1, but if the gambler doubles the initial bet every time eventually he will reach a point where the gambler cannot play anymore because the bet will be greater than the his gaming budget. Much of the work on martingales was done in order to prove that there aren't advantageous betting strategies.

A sequence of random variables $Z_0, Z_1, ...$ is a martingale with respect to a sequence $X_0, X_1, ...$ if, for all $n \geq 0$ we have that:

- Z_n is a function of $X_0, ..., X_n$;
- $E[|Z_n|] < \infty$;
- $E[Z_{n+1}|X_0, ..., X_n] = Z_n$.

Note that the sequence $X_0, X_1, ...$ can be equal to $Z_0, Z_1, ...$ in such case $Z_0, Z_1, ...$ is a martingale respect itself and the following conditions must apply:

- $E[|Z_n|] < \infty$;
- $E[Z_{n+1}|Z_0, ..., Z_n] = Z_n$.

Sometimes we need to create a martingale from a given sequence of random variables let $X_0, ..., X_n$ such sequence. We can use a general procedure to create a martingale

$Z_0, ..., Z_n$ respect to $X_0, ..., X_n$ such martingale is called a Doob martingale [2] .
We start choosing a random variable Y which can depend on $X_0, ..., X_n$ such that $E[|Y|] < \infty$, we then define the martingale $Z_0, ..., Z_n$ for i=0,1,...,n as:

$$Z_i = E[Y|X_0, ..., X_i]. \tag{3.11}$$

The created sequence is a martingale respect to $X_0, ..., X_n$ as:

$$E[Z_{i+1}|X_0, ..., X_i] = E[E[Y|X_0, ..., X_{i+1}]|X_0, ..., X_i] = E[Y|X_0, ..., X_i] = Z_i. \tag{3.12}$$

In the large part of the applications Z_0 is set in order to be equal to the expected value of Y, note that this requires X_0 to be independent of Y. Generally this defined martingale can be used when we want to predict the value of Y and such value is a function of $X_1, ..., X_n$, $Y = f(X_1, ..., X_n)$. We have that the martingale $Z_0, ..., Z_n$ can be viewed as a sequence of estimates on the value of Y by adding step by step more information on the value of $X_0, ..., X_n$. Then we have that according to (3.11) $Z_0 = E[Y]$, Z_i for $1 \leq i \leq n$ Z_i is equal to the expected value of Y when we know the value of $X_1, ..., X_i$, then accordingly $Z_n = Y$. Having a martingale can be useful as it is possible to apply inequality that are similar to a Chernoff bound, but in the case of martingale no hypothesis on the dependence of the random variables is made.

If $X_0, ..., X_n$ is a martingale where the values between two consecutive X_k are bounded by a value c_k, more formally such that for $k \geq 1$:

$$|X_k - X_{k-1}| \leq c_k. \tag{3.13}$$

Then for any $\lambda > 0$ and for $t \geq 0$ we have that:

$$P[|X_t - X_o| \leq \lambda] \leq 2e^{-\lambda^2/(2\sum_{i=1}^{t} c_i^2)}. \tag{3.14}$$

For any $\lambda > 0$, for $t \geq 1$ and if $c_i = c$ for $0 \leq i \leq n$:

$$P[|X_t - X_o| \leq \lambda c\sqrt{t}] \leq 2e^{-\frac{\lambda^2}{2}}. \tag{3.15}$$

A more strict bound can be obtained when for the martingale $X_0, ..., X_n$ exists a random variable $B_k = f(X_0, ..., X_{k-1})$ and a constant d_k such that $B_k \leq X_k - X_{k-1} \leq B_k + d_k$ then for $t \geq 0$ e for any $\lambda > 0$:

$$P[|X_t - X_0| \geq \lambda] \leq 2e^{-\lambda^2/\sum_{k=1}^{t} d_k^2}. \tag{3.16}$$

The set of such inequalities are called Azuma–Hoeffding inequalities, is interesting the fact that a prof can be done using the Chernoff bound. Also note that the term 2 in the above inequalities is due to the symmetry of $|X_t - X_0|$.

[2] Named after the american matematician Joseph Leo Doob considered the father of martingale's theory.

Chapter 4

Identifying progression patterns in cancer

In this section we propose a way to find statistically significative clonal and subclonal mutation patterns, using the work of [11] for the clonal and subclonal classification of single genes.
We introduce a way to summarize information from cancer reads using a mutation matrix, then we apply an approach to classify genes according to their clonal status [11] and from this we construct statistically significative couple of genes where one gene is classified as clonal and the other as subclonal. Successively we generalize such approach in order to find couple of stastically significative sets of genes where the genes in one set are classified as clonal and the others as subclonal respectively. We then put effort in finding an efficient approach in order to do that providing firstly a deterministic approach and then a probabilistic one that rely on the usage of bounds on the tail distribution.

4.1 The mutation matrix

A way to summarize cancer information that come from a dataset of genomic reads is to use a *mutation matrix*. This mutation matrix M uses as row the set of patients that are involved in the study while mapping in the columns the set of genes used. The matrix's role is to store if a given patient has a mutation in a specific gene.
More formally from a mutation matrix M where a single row represents the generic patient i while a column stands for the gene j we have that the position $M_{i,j}$ of the matrix stores the value relative to the gene j of the patient i that is going to be a 0 if gene j is not involved in a mutation or 1 otherwise. Let P to be the set of patients and let G to be set of genes than we have that the number of elements that are in the matrix is equal to $|P||G|$, we also expect to see a rather *sparse* matrix as from the biological point of view the number of mutated genes in a cancer patient is much smaller than |G|. This form of *binary* mutation matrix is used in the algorithm HotNet2[9]

We are going to introduce a *generalized* mutation matrix as this matrix use real values in the range $[0, 1]$. This variation is needed in order to provide a more realistic model of the biological problem. We would like to model the fact that not all the cancer cells

for a given patient carry a mutation in a specific gene as we can find families of cancer cells that involve mutations in different genes creating a rich mutational landscape, such aspect is ignored when we hypothesize that all the tumor cells in a patient share a mutation in the same set of genes.
We then have that $M_{i,j}$ represents for the patient i the fraction of cancer cells that carry a mutation in the gene j. An example of generalized mutation matrix can be found in table 4.1.

	Genes														
	0	0	0.7	0	0	0	0	0	0	0	0	0.2	0	0	0
	0.3	0	0	0	0.8	0	0	0	0	0	1	0	0	0	0
	0	0	0	0	0	0.9	0	0	0	0	0	0	0	0.6	0
Patients	0	0	0	1	0	0	0	0.1	0	0	0	0	0	0	0
	0.7	0	0	0	0	0	0	0	0.4	0	0	0	0.5	0	0
	0	0	0	0	0.5	0	0	0	0	0.3	0	0	0	0	1
	0	0.2	0	0	0	0	0.4	0	0	0	0	0	0	0	0

Table 4.1: An example of generalized mutation matrix used in our work.

4.2 Datasets

In this section we are going to define how we can create a generalized mutation matrix from a dataset that consist of cancer genomics data and then we are going to propose a further generalization based on the notion of *confidence interval.*

Each dataset consists of genomic information regarding a set of patients that are affected by a specific cancer. The datasets consist of different records and every record is relative to a specific read, there are multiple reads for each patient. There are also several headers but we are interested in only a subset of them in order to create the real matrix discussed in the previous section, we consider the following list of headers in the LGG dataset: `CASE.ID`, `HUGO_SYMBOL`, `n_alt_count`, `n_ref_count`, `VARIANT CLASSIFICATION`.
The `CASE.ID` header identifies the specific patient involved in the study, the `HUBO_SYMBOL` represents the gene name following the standard HUGO nomenclature[1], `n_alt_count` and `n_ref_count` indicate the number of the reads extracted from the cancer biopsy that are different and equal to the reference in the read mapping process and `VARIANT CLASSIFICATION` classifies the effects produced by the involved mutation.

After extracting the value of only those needed headers for every record we calculate a score, let s_r to be the score of the record r:

$$s_r = \frac{r[n\_alt\_count]}{r[n\_alt\_count]+r[n\_ref\_count]} \tag{4.1}$$

where with $r[n\_alt\_count]$ and $r[n\_ref\_count]$ we are referring to the value of the record r at the header `n_alt_count` and at the header `n_ref_count` respectively.

[1]HUGO stands for Human Genome Organization and has different committees one of them the HGNC (HUGO Gene Nomenclature Committee) aims to assign to every different human gene a specific and unique name and symbol.

The score s_r is going to be a real number in the range [0,1] so using the features `CASE.ID` and `HUBO_SYMBOL` as record identifiers and the definition of s_r for the generic record r we can map into a matrix, where the rows represent the set of patients and the columns represent the set of genes, the score obtained for every record in a specific dataset. If some records are related to the same patient and the same gene we use the maximum score obtained from these records and if the patient or the gene in a record are unknown (value `NA`) we decided to exclude that record from the tests. In this way we have obtained our real mutation matrix.

4.2.1 Binomial Proportion Confidence Interval

Referring to (4.1) if we see the event "the read is different from the reference" (we have a mutation) as a success and the complementary event "the read is equal to the reference" (we not have any mutation) as an unsuccess than s_r is equal to the point estimate of the parameter p of a binomial variable $B(p, n)$ that counts the number of reads that differ from the reference where n is the number of trials and is equal to the sum of the two events.

Instead of using a point estimate we can use a *binomial proportion confidence interval* [23, 1], an interval of values such that the probability that the real parameter p of the binomial variable fall in to this interval is equal to $1 - \alpha$.
There are numerous way to calculate such value, the normal approximation interval gives the simplest formula. After estimating $\bar{p}$ as described above the interval is:

$$\bar{p} \pm z\sqrt{\frac{\bar{p}(1-\bar{p})}{n}} \tag{4.2}$$

where z is the $1 - \frac{\alpha}{2}$ quantile value of the standard normal distribution, α is the error threshold and n is the number of experiments.

Unfortunately this approximation applies poorly when the sample size is less than 30 and when $\bar{p}$ is near to 0 or 1.
In order to provide a more precise approximation in the above cases and in general is preferable to use the more sophisticated approximation such as the Wilson score interval:

$$\frac{1}{1+\frac{1}{n}z^2}\left[\hat{p} + \frac{1}{2n}z^2 \pm z\sqrt{\frac{1}{n}\hat{p}\left(1-\hat{p}\right) + \frac{1}{4n^2}z^2}\right] \tag{4.3}$$

where z is the $1 - \frac{\alpha}{2}$ quantile value of the standard normal distribution.
In the case of $\alpha = 0.05$ the Wilson score interval is equal to the binomial proportion interval (4.2) using $\bar{p} = \frac{2+k}{4+n}$ where k is the number of successes among n trials. Also note that both intervals are centered around $\bar{p}$.

This can be a useful alternative score, as in the next section we are going to use a $1 - \alpha$ confidence interval real matrix for characterizing the clonal and subclonal classification, in particular we are going to use the Clopper-Pearson interval [1] that use the Beta distribution $B(x; y, z)$ [22]:

$$\left[B\left(\frac{\alpha}{2}; x, n-x+1\right), B\left(1-\frac{\alpha}{2}; x+1, n-x\right)\right] \tag{4.4}$$

where x are the number of success and n the number of trials. There are also other formulas that used different considerations, refer to [23].

4.3 Clonality and progression patterns

We use the clonal and subclonal classification criteria discussed in [11].
After estimating the cancer cell fraction (CCF) defined as the fraction of tumor cells with a specific mutation, a mutation is classified as clonal if the upper band of the 95% CCF confidence interval overlaps 1. This classification rule was introduced in order to not overestimate the number of subclonal mutations. In the supplementary material of [11] a new definition of clonal is provided in order to not force the classification of a gene to be clonal or subclonal introducing the fact that some mutations can be labeled as unclassified, under this new assumption their results remained consistent with the previous one. We have then decided to use the classification version that forces every gene to be either clonal or subclonal.
We use the terms *early* and *late* referring to the clonal and subclonal classes.

It is possible to define an approximative version of the CCF starting from our definition of the score s_r. Given the fact that for every gene we have two equal copies (humans are *diploid* organisms) and usually if a gene is mutated only one of this copy is mutated (is not common to find a *somatic* mutation that target both copy of the gene) then we have that only half of the reads are mutated. So we can relate the score s_r for a specif gene-patient tuple to CCF as the CCF being $2s_r$.
Starting from the real score matrix created at the beginning we can create a new matrix in order to obtain the point estimate of CCF for every patient-gene couple, let Q a real score matrix where in each cell we have the $1-\alpha$ binomial proportion confidence interval of s_r we can find a $1-\alpha$ binomial proportion confidence interval matrix for the CCF by multiplying each element of the matrix Q by 2, as we are diploid organisms.

After creating a new confidence interval matrix I let $I_{i,j}$ to be the $1-\alpha$ confidence interval for the score s_r relative to gene j and the patient i we create for every patient two sets E_p and L_p in order to classify as Early or Late every mutated gene for that patient. Let S to be the sample set and G the gene set, E and L two arrays of sets than algorithm 1 classify the entire dataset.
Note that when the 1-α confidence interval matrix I is relative to the score s_r then the input parameter u used a classification criterion is equal to $\frac{1}{2}$ as we have previously said that the CCF is defined to be $2s_r$.

We decided to use the Clopper-Pearson interval as it's more conservative than other binomial proportion confidence interval approximation formula [1]. Given the fact that is more conservative the interval itself is larger than other formulas and so some genes

```
input  : A 1-a confidence interval matrix $I$ for $s_r$ of size $|S| \times |G|$, u $\in \mathbb{R}$.
output: Two vectors $E$ and $L$ of sets.
for $i \in S$ do
    for $j \in G$ do
        if the gene j is mutated in the patient i then
            if upper bound of $I_{i,j} > u$ then
                $E_i \leftarrow E_i \cup j$
            else
                $L_i \leftarrow L_i \cup j$
            end
        end
    end
end
return $E$, $L$;
```

Algorithm 1: Pseudocode for classify for every patient in the sample set their mutated genes according to the Early-Late definition.

that with the other formula are going to be classified as Late using the Clopper-Pearson formula are instead classified as Early. Generally the upper bound given by other formulas is smaller than the one given using Clopper-Pearson.

After obtaining the sets E and L for all possible genes, given a couple of two genes (a, b) we can find in how many patient the gene a is classified as Late and the gene b is classified as Early. What we obtain is a statistic about the relationship between two specific genes in the real data.
Given a couple of genes and a patient a relation between two genes in such patient is a function $f\colon G \times G \to D$ where D is a set that consist in the different types of relation considered, for example if given (a, b) we find that in a specific patient the gene a is classified as Early and the gene b as Late then the couple (a, b) is in an Early-Late relationship in this patient. Then for every different couple of genes and for every different relation we obtain the number of patients with that specific couple of genes in that specific relation.

We consider interesting studying the relation between a couple of genes while both genes are mutated. Let (a, b) to be a couple of genes where $a, b \in G_t$ under the previous assumption we consider the following 4 cases representing 4 different relations:

Early-Late a is classified as Early and b as Late.

Late-Early a is classified as Late and b as Early.

Early-Early a is classified as Early and b as Early.

Late-Late a is classified as Late and b as Late.

We only consider these 4 relations but there are others possible types of relations, for example considering the couple of gene (a, b) the cases where one gene in classified as Early or Late but the other one is not mutated in a given patient or even when the genes a and b are not mutated in the given patient.

Now we are interested in seeking statistically significant relations so we created a Monte Carlo simulation, we are going to use this simulation in order to apply a statistical test using the p-value approach. Let S to be the sample set and G_t the subset of genes considered, E and L two arrays of sets than we can create an initial random distribution following Algorithm 2 using as initial domain the non simulated dataset and then we can defined a metric for the p-value that aggregate the results from the different Monte Carlo experiments.

After running Algorithm 1 and Algorithm 2 we obtain a simulated dataset with the

```
input : $R$ stores the number of patients where a specific gene g appears as
        Early or Late.
output: Two vectors $E$ and $L$ of sets representing the sets $E_p$ and $L_p$ for
        every patient.
for $g \in G_t$ do
    Let $R_{g,e}$ the number of patients in which the gene g appear as Early and
     $R_{g,l}$ as Late.
    $SIM \leftarrow R_{g,e} + R_{g,l}$ patients uniformly at random from $S$
    $SIM_e \leftarrow R_{g,e}$ patients uniformly at random from $SIM$
    $SIM_l \leftarrow SIM \setminus SIM_e$
    for $p \in SIM_e$ do
        $E_p \leftarrow E_p \cup \{g\}$
    end
    for $p \in SIM_l$ do
        $L_l \leftarrow L_l \cup \{g\}$
    end
end
return $E$, $L$;
```

Algorithm 2: Pseudocode for generating the initial random distribution for one Monte Carlo simulation instance.

same number of patients of the real dataset and two new sets E and L for every patient that permit for a given couple of genes (a, b) to find out the number of patients where the couple of genes is in a specific relation, for example where a is classified as Early and b is classified as Late for the relation Early-Late.
For a given relation R and given couple of genes we can define a *permutational* p-value: let n the number of experiments done and t the number of datasets where c appear in the relation R in more or the same number of patients that the real dataset:

$$p_{val}(c, R) = \tfrac{t+1}{n+1} \tag{4.5}$$

A pseudocode that describe how the Monte Carlo simulation is structured is given by Algorithm 3.

```
input  : $n$ the number of simulation to run, a $1 - \alpha$ confidence interval matrix $I$
         for $s_r$ of size $|S| \times |G|$, u $\in \mathbb{R}$, a matrix $R$ which stores the number of
         patients where a specific gene g appears as Early or Late.
output: $U$ a matrix storing the $p$-value for every couple of genes for different
         relations, $U_{c,R}$ stores $p_{val}(c, R)$.
$U \leftarrow 0$
/* Let $C_{c,R}$ to be the number of simulated datasets where for the couple of genes
  $c$ and in the relation $R$ where the number of occurrence in the simulated
  dataset is not less that the number of occurrences in the real dataset. */
$C_{c,R} \leftarrow 0$ for every distinct couple of genes and for every different relation
$E, L \leftarrow$ Algorithm 1$(I, \frac{1}{2})$
for $i$ :=1 to $n$ do
    $Es, Ls \leftarrow$ Algorithm 2$(R)$
    for every distinct couple of genes $(a, b)$ from $G_t$ do
        for every different relation $R$ do
            Let $u$ and $q$ to be the number of patients where the couple (a,b)
              appears in the relation $R$ in the sets $E, L$ and respectively in $Es, Ls$.
            if $u <= q$ then
                $C_{c,R} \leftarrow C_{c,R} + 1$
            end
        end
    end
end
for every distinct couple of genes $c = (a, b)$ and for every different relation $R$ do
    $U_{c,R} \leftarrow \frac{C_{c,R}+1}{n+1}$
end
return $U$;
```

Algorithm 3: Pseudocode for running the Monte Carlo experiment.

4.4 Aggregating the results

Starting from the output of the Algorithm 3 we are interested in aggregating and exploiting the knowledge given by the Early and Late classification.
Until this point we have information only about statistically significative couples of genes. We are interested in generalizing and finding statistically significative couple of sets of genes that are large in size.

We could try to provide a statistical test in order to find statistically significant couple of sets but we must first define a method to calculate the p-value between two gene sets.
In a way that is similar to the how we assign the p-value for a couple of genes we can use a Monte Carlo experiment with a similar initial random distribution as we done previously.
Given two gene sets A and B we define a patient p to be *interesting* if $|E_p \cap A| \geq 1$ and $|L_p \cap B| \geq 1$ where E_p, L_p are the sets of mutated genes classified as Early and respectively as Late in the patient p. Another way to see the previous definition is that the patient p is *interesting* if and only if at least one gene of $E_p \in A$ and at least one gene of $L_p \in B$. Given A and B we define $c_{A,B}(T)$ the number of *interesting* patients in the patient set T in relation to the gene sets A and B. The definition of the p-value for a given couple of genes A and B follows. Let T to be a simulated sample set, let S to be the real sample set we define T to be *countable* if $c_{A,B}(T) \geq c_{A,B}(S)$ than t is the number of the *countable* simulated sample sets in the Monte Carlo simulation:

$$p_{val}(A,B) = \tfrac{t+1}{n+1} \tag{4.6}$$

where n is the number of experiments done.
Algorithm 4 for generating the p-value of two sets of genes follows.

Giving two sets of genes then we can provide a test in order to verify if those sets are statistically significant, given A and B two sets of genes they are statistically significant if $p_{val}(A,B) \leq \alpha$ where α is the test parameter, we commonly use $\alpha = 0.05$.

We could try to create all the different couples of statistically significative sets in a naive-enumerative way following for example this approach. Firstly we create all the different couples of sets of genes from the gene set G_t. This first part can be done in the following way, we imagine to create one couple of sets using a systematic approach let A_g and B_g the generic couple of gene sets: initially from the gene set G_t we can choose $k = |A_g \cup B_g|$ genes from the G_t, from those k genes we choose k_A genes to part of the set A while the remaining $k - k_A$ genes are part of the set B. According to the previous method the number of different couples of set of genes is: $\sum_{k=1}^{|G_t|} \binom{|G_t|}{k} (\sum_{u=1}^{k-1} \binom{k}{u})$. This number is $\Omega(3^{|G_t|})$ as for every gene in G_t we can decide if the considered gene belongs to A, B or if it does not belong to the solution. Then for every generated couple we can test if the couple is statistically significative using directly the definition given.

Clearly the computational complexity of the enumerative approach for finding the set of the couple of statistically significative sets is too high as it is exponential in the number of the genes considered hence we must consider a different approach.

```
input  : $n$ the number of simulation to run, $A$ and $B$ two gene sets, $S$ the real
         patient set.
output: The estimated $p$-value for the couple of gene sets $A$ and $B$.
$r \leftarrow c_{A,B}(S)$
$t \leftarrow 0$
for $k := 1$ to $n$ do
    For every patient p $\in S$ let $E_p = \emptyset$ and $L_p = \emptyset$ the set of genes classified as
     Early and as Late respectively.
    for every gene $g$ in $A \cup B$ do
        Let $e$ and $l$ the number of patients $\in S$ where the gene $g$ is classified as
         Early and as Late respectively.
        $P_s \leftarrow$ select $e + l$ random patients from $S$
        $E_s \leftarrow$ select $e$ random patients from $P_s$
        $L_s \leftarrow P_s \setminus E_s$
        for every patient $p \in E_s$ do
            $E_p \leftarrow E_p \cup \{g\}$
        end
        for every patient $p \in L_s$ do
            $L_p \leftarrow L_p \cup \{g\}$
        end
    end
    /* Let $T$ to be the above simulated dataset */
    if $c_{A,B}(T) \geq r$ then
        $t \leftarrow t + 1$
    end
end
return $\frac{t+1}{n+1}$;
```

Algorithm 4: Pseudocode for calculating $p_{val}(A, B)$.

Now that we have a method to calculate the p-value for a couple of sets of genes we can create an algorithm to find statistically significant couple of sets starting from the previous calculated couple of genes.
We can try to create two statistically significant sets A, B in a *greedy* way. We start from a couple of statistically significative genes (a, b) assuming the gene a classified as Early and the gene b classified as Late we then initialize $A = a$ and $B = b$. Let (A, B) to be the current solution we obtain a candidate solution (A_c, B_c) by adding to A or to B a gene g such that $g \notin A$ and $g \notin B$ from those candidate solutions we then select the solution with the smaller p-value. The solution at the (i+1)-iteration differs from the solution at the previous state only by the gene g that is chosen to be inserted into A or into B, consequently for any candidate solution $(A_c,\ B_c)$ we have that $|A \cup B| < |A_c \cup B_c|$. We continue then expanding the solution adding genes until we can find a gene that decrements or maintains the p-value of the solution. So for every iteration the cardinality of the solution is expanded and its p-value is decremented at most by an amount d. The convergence of the solution into a final solution is given by the fact the set of genes G_t is finite. Furthermore we can limit the cardinality of the two genes sets in the solution by upper bounding the size of their union by an integer parameter s_{max}, this limit can be view also as an alternative stopping criterion. Algorithm 5 apply this greedy approach in creating a solution it also use the p-value calculated by Algorithm 4. We have omitted what we should do when we find more than one candidate solution that can be promoted as the new current solution by using the term "apply a Choice". Given a set of "eligible" candidate solutions we can consider the following Choices:

- LAST: We choose the last added one.
- RANDOM: We select one uniformly at random.
- BALANCE: Given a solution (A, B) we try to keep the size of the set of genes A similar to the size of the set B, promoting as current solution a solution that increments the size of the set of genes (A or B) with the lowest size. If more than one solution accomplish this we select one uniformly at random. If $|A| = |B|$ we select one uniformly at random.

If we take two sets of genes A and B is possible to observe that if we start from a couple of statistically significant sets of genes if we add a generic gene to one set obviously the resulting couple of sets is not always statistically significative.
Given a current solution (A, B) with $|A| = n$ and $|B| = m$ and $|G_t| = k$ the pool of genes considerated then if from a current solution a single gene is inserted in one set we have that the resulting number of different candidate solution generate is $2(k - n - m) = O(k)$ but if instead a current solution is enriched adding couples of genes than the number of candidate solution rise to $(k - n - m)^2 = O(k^2)$.

```
input  : $U$ the set of different couples of genes and their $p$-value relatives to different
         relations, $G_t$ the set of genes considered.
output: $A$ and $B$ sets of genes.
Let $(a, b)$ to be the couple of genes with the minimum $p$-value looking at the relations
  Early-Late and Late-Early such that $a$ is classified as Early and $b$ as Late.
$A \leftarrow a$
$B \leftarrow b$
$pv \leftarrow p_{val}(A, B)$
$f \leftarrow true$
while $f$ is true do
    $f \leftarrow false$
    $candidates \leftarrow \emptyset$
    for every gene $g \in G_t \setminus (A \cup B)$ do
        $u \leftarrow p_{val}(A \cup \{g\}, B)$
        $t \leftarrow p_{val}(A, B \cup \{g\})$
        if $u \leq pv$ then
            $candidates \leftarrow candidates \cup \{(A \cup \{g\}, B), u\}$
        end
        if $t \leq pv$ then
            $candidates \leftarrow candidates \cup \{(A, B \cup \{g\}), t\}$
        end
    end
    if $|candidates| > 0$ then
        $f \leftarrow true$
        From the candidate solutions in candidates find the ones that yield the
          minimum $p$-value, if there is more than one apply a "Choice"
        Let $(A^*, B^*)$ such candidate solution
        $A \leftarrow A^*$
        $B \leftarrow B^*$
        $pv \leftarrow p_{val}(A^*, B^*)$
    end
end
return $A, B$;
```

Algorithm 5: Pseudocode for the greedy algorithm.

4.4.1 A deterministic bound on the p-value

In the previous pages we have created an algorithm that find the p-value first for two couples of genes and then for two sets of genes using a Monte Carlo algorithm. In all cases we have generated a set of random data starting from the real data using different approaches. The first approach is used in Algorithm 3 and generate a complete simulated dataset involving all the genes of G_t and then for every couple of genes it does some computation, this task is repeated n number of times, where n is the number of Monte Carlo experiments to run. If the first approach generate an entire simulated dataset the second approach generate simulated input information considering only a subset of genes. For example in Algorithm 4 we generate the E_s and L_s only looking at the genes that are in $A \cup B$ as only this set of genes is involved for calculating the p-value, we then generate those two simulated sets n times. If the sets A and B change then we need to recalculate or expand E_s and L_s in order to reflect the change.
We could have used also the first approach in Algorithm 4 : firstly we can create n full simulated dataset involving all the genes in the pool G_t, then for two sets A and B we check for each one of the n datasets the number of *interesting* patients and compare this number with number of *interesting* of the real dataset, finally we calculate the p-value following the definition provided. In this way we can use those simulated datasets regardless of the genes that are in $A \cup B$ instead of expanding o recalculating the sets of the simulated datasets wherever $A \cup B$ changes. Using this approach we can pre-initialize n different random simulated datasets and then start finding $c_{A,B}(T)$ for each simulated dataset T and then calculate the p-value of (A, B), in the next iteration we just reuse those datasets again this permits to give some particular observations.

On the example of Algorithm 5 we are now interested in finding a way to exclude a gene from being added to the current solution, this can reduce the number of candidate solutions to be considered. Providing a lower bound to the p-value of a candidate solution can be used as a pruning criterion on the set of genes that can be added.
For the following considerations we assume to be in the case of pre-initialized random simulated datasets.
For a given dataset the number of *interesting* patient increase or remain the same iteration by iteration because a patient that is *interesting* remain *interesting* by definition while a patient can become *interesting* if while adding a set of genes G_a to the current solution (A, B) we can find at least two genes (v, t) such that v is classified as Early in that patient and t is classified as Late respectively. Clearly this consideration apply both to the real dataset and to the simulated ones. On the top of this observation is possible to provide an upper and a lower bound to the p-value of a candidate solution.

Remember that a patient p with E_p and L_P, the sets of genes classified as Early and as Late in p, is *interesting* if for a given solution (A, B) we have that $|E_p \cap A| \geq 1$ and $|L_p \cap B| \geq 1$. Let (A, B) be the current solution and let (A^*, B^*) be the candidate solution obtained by adding one gene to A or to B.
We indicate with k^i the number of *interesting* patient in the real dataset and with $k_1^i, ..., k_n^i$ the number of *interesting* patient in the simulated datasets at the i-iteration. We define the indicator function $\mathbb{1}(k_x^i)$ relative to a simulated dataset x, $1 \leq x \leq n$, as follows:

$$\mathbb{1}(k_x^i) = \begin{cases} 1 & \text{if } k_x^i \geq k^i, \\ 0 & \text{otherwise .} \end{cases} \tag{4.7}$$

A simulated dataset x is *countable* if $\mathbb{1}(k_x^i) = 1$. Then rewriting the definition of the p-value for two sets of genes (A, B) using the new terminology we have that:

$$p_{val}(A, B) = \frac{\sum_{x=1}^{n} \mathbb{1}(k_x^i) + 1}{n + 1} = \frac{1}{n + 1} + \frac{\sum_{x=1}^{n} \mathbb{1}(k_x^i)}{n + 1}. \tag{4.8}$$

Clearly a very simple lower bound for the p-value is given by $\frac{1}{n+1}$. The p-value increase as the number of *countable* datasets rise while it decrements otherwise, is then directly proportional in the number of *countable* datasets.
Also if the p-value of a solution decrements or remains the same iteration by iteration we have that for two consecutive iteration i and $i + 1$ the follow relation apply:

$$\sum_{x=1}^{n} \mathbb{1}(k_x^{i+1}) \leq \sum_{x=1}^{n} \mathbb{1}(k_x^i). \tag{4.9}$$

Now supposing to extend the current solution (A, B) adding the gene g that is classified as Early in g_E patients and as Late in g_L. For a given solution (A, B) a patient p can be *interesting* or not, but among the not *interesting* patients we can find a patient where $|E_p \cap A| \geq 1$ or where $|L_p \cap A| \geq 1$ those particular sets of patients can become *interesting* adding a particular gene.
A patient p is *E-almost interesting* if $|E_p \cap A| \geq 1$ and $|L_p \cap B| = 0$.
A patient p is *L-almost interesting* if $|E_p \cap A| = 0$ and $|L_p \cap B| \geq 1$.
Then for example an *E-almost interesting* patient p_e can become *interesting* if we add a gene $g \in L_{p_e}$ to B archiving that $|L_{p_e} \cap B| \geq 1$ then such patient becomes by definition *interesting.*
Let:
qE the number of *E-almost interesting* patients in the real dataset given the solution (A, B);
qL the number of *L-almost interesting* patients in the real dataset given the solution (A, B);
qE_x the number of *E-almost interesting* patients in the simulated dataset x given the solution (A, B);
qL_x the number of *L-almost interesting* patients in the simulated dataset x given the solution (A, B).

Now adding the gene g to the solution we define $\triangle k^{i+1} = |k^{i+1} - k^i|$ and $\triangle k_x^{i+1} = |k_x^{i+1} - k_x^i|$ indicate the number of patient that become *interesting* in the real dataset and in the x-simulated dataset respectively, if we add the gene g to the solution. Such values are bounded by the following quantities:
if $g \in A^*$ (if g is added to set A): $\triangle k^{i+1} \leq min(qL, g_E) = \triangle^* k^{i+1}$ and $\triangle k_x^{i+1} \leq min(qL_x, g_E) = \triangle^* k_x^{i+1}$ for every simulated dataset x,
if $g \in B^*$ (if g is added to set B): $\triangle k^{i+1} \leq min(qE, g_L) = \triangle^* k^{i+1}$ and $\triangle k_x^{i+1} \leq min(qE_x, g_L) = \triangle^* k_x^{i+1}$ for every simulated dataset x.

After adding a gene g the number of simulated dataset that were *countable* can vary and the number of interesting patient can varies also in the real dataset. In figure 4.1 we can see an example on the range of values taken by k_x^{i+1} and k^{i+1} in the case of not *countable* dataset x. We now define formally when a dataset can vary or maintain its interestingness after expanding the solution.

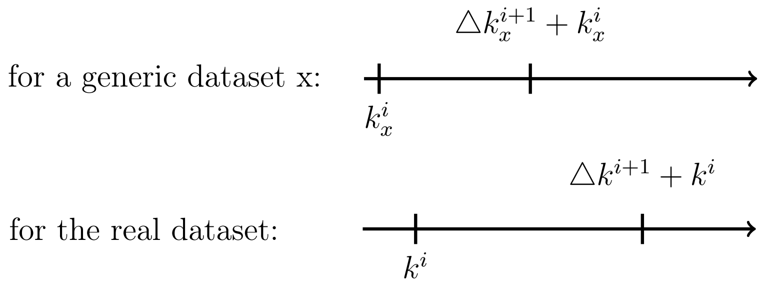


Figure 4.1: Interval for the number of *interesting* patient in the (i+1)-iteration given the number in the i-iteration.

A generic dataset x can be either *countable* or not, then:
x can become not *countable* if: $k^i \leq k_x^i$ but $\triangle k_x^{i+1} + k_x^{i+1} < \triangle k^{i+1} + k^{i+1}$;
x can become *countable* if: $k^i > k_x^i$ but $\triangle k_x^{i+1} + k_x^{i+1} \geq \triangle k^{i+1} + k^{i+1}$.
Then aggregating the inequalities we have that a simulated dataset x is *at risk*:

$$\begin{cases} \text{if } x \text{ is not } \textit{countable}\text{:} & \triangle k_x^{i+1} + k_x^i \geq k^i, \\ \text{otherwise:} & k_x^i < \triangle k^{i+1} + k^i. \end{cases} \tag{4.10}$$

The set of datasets that are *at risk* for two different genes g_1 and g_2 are generally different as we take in consideration g_E and g_L for a particular gene.
Now it is possible to use datasets that are *at risk* in order to obtain a bound to the p-value of (A^*, B^*).
Let r_{nc} and r_c to be the set of the not *countable* datasets that are *at risk* and the set of the *countable* datasets that are *at risk*, then the p-value of (A^*, B^*) is bounded by:

$$p_{val}(A^*, B^*) = [\frac{1}{n+1}(1 + \sum_{j=1}^{n} \mathbb{1}(k_j^i) - r_c),\ \frac{1}{n+1}(1 + \sum_{j=1}^{n} \mathbb{1}(k_j^i) + r_{nc})]. \tag{4.11}$$

This bound is obtained assuming the fact that all the r_c datasets will become not *countable* while the status of the r_{nc} ones remains unchanged if we then expand the solution (A, B) by adding the gene g obtaining the candidate solution (A^*, B^*) this give a lower-bound for the p-value of the candidate solution, when we assume that all the r_{nc} datasets will become *countable* leaving the other r_c that are *at risk countable* we obtain respectively an upper bound. With such hypotheses the above bound is based on a worst-case analysis.

The considerations used to derive this bound can also be used to provide an alternative way to calculate the p-value of a candidate solution given the set of the datasets that are at *at risk* in relation to gene g. Supposing that we start from a current solution (A, B) while adding the gene g to (A, B) to obtain the candidate solution (A^*, B^*), we have that:

$$\begin{aligned} p_{val}(A^*, B^*) =& \frac{1}{n+1}(1 + \sum_{j=1}^{n} \mathbb{1}(k_j^i) + r_{NC} - r_{CN}) = \\ =& p_{val}(A, B) + \frac{1}{n+1}(r_{NC} - r_{CN}), \end{aligned} \tag{4.12}$$

where r_{NC} are the dataset *at risk* that from being not *countable* become *countable* and r_{CN} are the dataset *at risk* that become not *countable* from being *countable*. Clearly we have that $r_{NC} \leq r_{nc}$ and $r_{CN} \leq r_c$ and if we do again the worst case analysis done before we obtain obviously the previous results. Also if we look at the first two terms of the sum we can recognize that they are exactly the p-value of the current solution (A, B). So we can find the p-value of a candidate solution starting from the value obtained from the current solution considering only the set of dataset that are *at risk* at that time.

If we look at the set of patients relative to a generic dataset D we have that the number k_D^i of interesting patient at the iteration i is less or equal than the number of interesting patient at the successive iteration so $k_D^i \leq k_D^{i+1}$ as a patient that is *interesting* remain *interesting* over the future iterations. This can be useful while we need to update k_D^i to become k_D^{i+1} because we just need to look at the subset of the patients of D that were not *interesting*, reducing the computational cost.

Also if we suppose to know the value k^{i+1} for the real dataset we have that for a subset of the simulated datasets that are *at risk* it may be possible to note directly that their status is going to change, specifically let x to be a simulated dataset then we have that:

$$\begin{aligned} &1)\text{If } x \text{ is } \textit{at risk}, \textit{ not countable} \text{ and } k^{i+1} > k_x^i + \triangle^* k_x^{i+1} \implies x \text{ remains } \textit{not countable}. \\ &2)\text{If } x \text{ is } \textit{at risk}, \textit{ countable} \text{ and } k^{i+1} > k_x^i + \triangle^* k_x^{i+1} \implies x \text{ become } \textit{not countable}. \\ &3)\text{If } x \text{ is } \textit{at risk}, \textit{ countable} \text{ and } k^{i+1} \leq k_x^i + \triangle^* k_x^{i+1} \implies x \text{ remains } \textit{countable}. \end{aligned} \tag{4.13}$$

Note the asymmetry of the conditions, a third condition like is applicable only if x is not *at risk* and in this case x maintains its status by definition.
We can use (4.13) directly in order to speed up the p-value calculations, as we increase the number of simulations to do calculating k^{i+1} can be advantageous in term of computations avoided.

4.5 A probabilistic approach

4.5.1 Approach based on Chernoff bounds

In the previous section while calculating the bound we defined $\triangle^* k_x^{i+1}$ and we used this value to provide a deterministic bound. Generally not all the mutations relative to

a gene g are going to affect a specific subset of patients, in our case the set of the patients that are *E-almost interesting* or *L-almost interesting*, since mutations are placed randomly in a simulated dataset.
Given (A, B) the current solution at the iteration i we then consider the candidate solution (A^*, B^*) obtained by adding, for example, to the gene set A the gene g obtaining then $A^* = A \cup \{g\}$ and $B^* = B$, we indicate gE and gL the number of patients with the gene g classified as Early and as Late. Now considering a generic simulated dataset x we have that respect to the current solution x has k_x^i interesting patients while we have that some patients are *E-almost interesting* and *L-almost interesting*, let Eq_x and Lq_x to be the sets of those patients. A patient p that is *L-almost interesting* can become *interesting* if while considering the candidate solution the gene g is classified as Early in such patient. Considering that while creating a simulated dataset for each gene we select uniformly at random a pool of patients from the patients set, we can now define a new model. Now considering gE mutations to be distributed among $|x|$ patients where m of them are *L-almost interesting* we are interested in the number of patients that become *interesting*. Let X_i the indicator random variable defined is this way:

$$X_i = \begin{cases} 1 & \text{if the mutation } gE_i \text{ involve a patient that is } \textit{L-almost interesting}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.14}$$

Clearly X_i is Bernoulli random variable and $P[X_i = 1] = p_i$ is a function of $(X_1, ..., X_{gE})$ more specifically $p_{i+1} = f(X_i)$ for any $1 < i \leq g_E$. Now let n_i to be the number of patients that after assigning the first i-mutations do not have a mutation in the gene g and let m_i to be the number of patients that are also *L-almost interesting*. Now assuming $n_i \geq m_i$ and if $m_i > 0$ and $n_i > 0$ we have that:

$$p_{i+1} = \begin{cases} \dfrac{m_i - 1}{n_i - 1} & \text{if } X_i = 1 \text{ and } m_i > 1, \\ 0 & \text{if } X_i = 1 \text{ and } m_i \leq 1, \\ \dfrac{m_i}{n_i - 1} & \text{if } X_i = 0, \end{cases} \tag{4.15}$$

This dependency between the various X_i is given by the fact that a patient cannot have "more mutations" involving the same gene, this constraint represents an extraction without replacement model. Let n to be the number of patients that before assigning any mutation do not have a mutation in the gene g and let m to be the number of patients that are also *L-almost interesting*, we decided to set the probability $p = \frac{m}{n}$ for each indicator random variable X_i, as mutation are assigned one after the other and given this the random variables are now independent. This can be viewed as an approximation of the sampling without replacement model.
Now let X to be the random variable that count the number of *L-almost interesting* patients that are affected by a mutation in the gene g while g is classified as Early. Now we have that $X = \sum_{i=1}^{g_E} X_i$ from which we can compute the expected value of X as:

$$E[X] = E[\sum_{i=1}^{g_E} X_i] = \sum_{i=1}^{g_E} E[X_i] = g_E \frac{m}{n} = \mu. \tag{4.16}$$

We have that $\mu = 0$, according to the (4.16), when $g_E = 0$ or when $\frac{m}{n} = 0$, more specifically when $m = 0$. This can happen when we cannot find any patient in the real

dataset where g is classified as Early, respectively as Late, or when for the simulated dataset in question we cannot find any *L-almost interesting patients*, respectively *E-almost interesting patients*. In such cases the number of *interesting* patients for such simulated dataset respect to candidate solution remains the same as respect to the current solution, we can then immediately find if the dataset x change its status or not.
Now assuming to know the number k^i of interesting patients according to the current solution in the real dataset and the number $\triangle k^{i+1}$ of patients that were *L-almost interesting* but according to the candidate solution are now *interesting* in the real data set. Assuming now that the dataset x is not *countable* we would like to find out the probability that such dataset become *countable* under the candidate solution. If x is not *countable* then x become *countable* if:

$$k_x^i + X \geq k^i + \triangle k^{i+1} = k^{i+1} \tag{4.17}$$

from which it follows that:

$$P[x \text{ become } \textit{countable}] = P[X \geq k^{i+1} - k_x^i]. \tag{4.18}$$

Now we can provide an upper bound to this probability using a Chernoff bound.
Given that $X_1, ..., X_{g_E}$ are independent we can apply the Chernoff bound choosing a specific δ such that $X \geq k^{i+1} - k_x^i = (1+\delta)\mu$, now solving in respect to δ we have that

$$\delta = \frac{k^{i+1} - k_x^i}{\mu} - 1 \tag{4.19}$$

which is applicable to Chernoff if:

$$k^{i+1} > k_x^i + \mu \tag{4.20}$$

and now using the δ defined as in (4.19), applying the relation (4.16) and reordering the inequality:

$$P[X \geq k^{i+1} - k_x^i] \leq \left(\frac{e^{\frac{k^{i+1} - k_x^i}{g_E \frac{m}{n}} - 1}}{\left(\frac{k^{i+1} - k_x^i}{g_E \frac{m}{n}}\right)^{\frac{k^{i+1} - k_x^i}{g_E \frac{m}{n}}}} \right)^{g_E \frac{m}{n}} \tag{4.21}$$

This bound is applicable only to the set of simulated dataset where (4.20) is satisfied. This condition tend to afflict more the datasets with $k_x^{i+1} \geq k^i$: such subset of datasets is the set of the *countable* ones which iteration by iteration, if the p-value lowers, is going to lowers in size. For all the dataset where the Chernoff bound is not applicable we then compute k_x^{i+1} simulating the expansion of the current solution with the addition of the gene g. We then know exactly if such datasets become or remain *countable.*

In the datasets where (4.20) is satisfied (4.21) give us an upper bound on the probability that a set that is not *countable* turn out to be *countable* if we consider the candidate solution instead of the current one. In the same way if we consider a dataset x that is *countable* and considering when it remains *countable* upon the new candidate solution than the probability that x maintain its status is the same as in (4.21) and hence Chernoff provide again an upper bound to for such probability.
Clearly if the dataset x is not *at risk* then the probability that such dataset become or remains *countable* is:

$$P[x \text{ become or remains } \textit{countable} \mid x \text{ not } \textit{at risk}] = \begin{cases} 1 & \text{if } x \text{ is } \textit{countable}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.22}$$

We can interpreter such probability as a score that indicates for every simulated dataset x how likely the dataset contribute to the p-value of the current solution. We then define for every simulated dataset x the score $s(x)$:

$$s(x) = P[x \text{ become or remains } \textit{countable}]. \tag{4.23}$$

If x is not *at risk* then $s(x)$ is equal to (4.22) if the otherwise is equal to the upper bound provided by Chernoff in (4.21) if applicable. Otherwise for the datasets where (4.20) is not satisfied we consider the candidate solution as current we then find out k_x^{i+1} and if the dataset stays or become *countable* then $s(x) = 1$ otherwise $s(x) = 0$. Now let D to be the set of the simulated datasets, we can now definite two different types of random variables. The first one is relative to the datasets that are *countable* while the second one involve only the not *countable* ones, their role is to indicate if the dataset stay or become *countable* if we consider the candidate solution instead of the current one. Let X_x the random variable associated to the simulated dataset x, then:

$$\begin{gathered} \text{if } x \text{ is } \textit{countable} \text{ than } X_x = \begin{cases} 1 & \text{if } x \text{ remains } \textit{countable}, \\ 0 & \text{otherwise.} \end{cases} \\ \text{if } x \text{ is not } \textit{countable} \text{ than } X_x = \begin{cases} 1 & \text{if } x \text{ become } \textit{countable}, \\ 0 & \text{otherwise.} \end{cases} \end{gathered} \tag{4.24}$$

Clearly X_x is an another Bernoulli random variable $B_e(p)$ such that the $P[X_x = 1] = s(x)$ then we have that $p = s(x)$.
Now we define a new random variable $\bar{X}$ that count the number of datasets that become or stay *countable*, we have than that:

$$\bar{X} = \sum_{x \in D} X_x \text{ and } E[\bar{X}] = E[\sum_{x \in D} X_x] = \sum_{x \in D} E[X_x] = \sum_{x \in D} s(x) = s \tag{4.25}$$

If a dataset x is not *at risk* or if we cannot apply the Chernoff bound (4.21) due to (4.20) not holding we then know the value k_x^{i+1} as we are forced to calculate it in order to find the p-value of the candidate solution. We can separate the contribute that such datasets give to $\bar{X}$ as we can note that $\bar{X}$ is constituted by a deterministic part given by such datasets and a probabilistic part given by the remaining datasets. Let d the deterministic contribute and let $\hat{X}$ the random variable that count the number of datasets that are *at risk* and where 4.20 holds that stay or become *countable* respect to the candidate solution:

$$\bar{X} = d + \hat{X}. \tag{4.26}$$

Now we have that for the candidate solution (A^*, B^*) its p-value is:

$$p_{val}(A^*, B^*) = \frac{1 + \bar{X}}{|D| + 1} = \frac{1 + d + \hat{X}}{|D| + 1}. \tag{4.27}$$

Let $p_{val}(A, B)$ to be the p-value of the current solution and let k the number of datasets that are *countable* in the current solution, we are now interested in the probability that the p-value of the candidate solution (A^*, B^*) is not greater than the p-value of the current solution:

$$P[p_{val}(A^*, B^*) \leq p_{val}(A, B)] = P\left[\frac{1 + d + \hat{X}}{|D| + 1} \leq \frac{1 + k}{|D| + 1}\right] = P[\hat{X} \leq k - d], \tag{4.28}$$

where given the fact that $\hat{X} \geq 0$ we must have that:

$$k \geq d. \tag{4.29}$$

If $k < d$ we have that:

$$\frac{1 + k}{|D| + 1} \leq \frac{1 + d + \hat{X}}{|D| + 1} \Longrightarrow p_{val}(A, B) < p_{val}(A^*, B^*) \tag{4.30}$$

we then can conclude immediately that whenever we see $k < d$ the p-value of such candidate solution can only increase as $\hat{X} \geq 0$, we can then discard such solution directly.
Now assuming $E[\hat{X}] = \hat{s} > 0$ we have that applying a Chernoff bound again and using:

$$\delta = 1 - \frac{k - d}{\hat{s}} \tag{4.31}$$

which is applicable to Chernoff if:

$$d < k < \hat{s} + d \tag{4.32}$$

we obtain that:

$$P[\hat{X} \leq k - d] \leq \left(\frac{e^{\frac{k-d}{\hat{s}} - 1}}{\left(\frac{k-d}{\hat{s}}\right)^{\left(\frac{k-d}{\hat{s}}\right)}} \right)^{\hat{s}}. \tag{4.33}$$

According to the (4.32) we have that p-value of the actual solution need to be between the lower bound that the p-value can reach and between the p-value calculated considering a solution with s *countable* datasets.
Given the equation 4.8 we have that the p-value of two set of genes assume value that

are multiple of the lower bound $\frac{1}{|D|+1}$ hence the p-value of a solution is a discrete value that ranges from $\frac{1}{|D|+1}$ to 1, the value that can be taken in this interval can be seen as a finite succession of value:

$$a_0, a_1, ..., a_i, ..., a_{|D|-1}, a_{|D|} \text{ with } a_i = \frac{i+1}{|D|+1}. \tag{4.34}$$

Then we can see that using:

$$\epsilon = \frac{1}{a}\frac{1}{|D|+1} \text{ with } a > 1 \tag{4.35}$$

and defining V as a random variable that assume a value that is the succession 4.34 we have that for such succession:

$$P[V \leq a_i + \epsilon] = P[V \leq a_i]. \tag{4.36}$$

This observations can be used in order to generalize the use of the Chernoff bound when we are in the case $k = d$, this arises when the p-value of the current solution is equal to the lower bound $\frac{1}{|D|+1}$ and we found a candidate solution that maintains the p-value to such value.
Assuming the general case:

$$\begin{aligned} P[p_{val}(A^*, B^*) \leq p_{val}(A, B) + \epsilon] =& P[\frac{1+d+\hat{X}}{|D|+1} \leq \frac{1+k}{|D|+1} + \frac{1}{a}\frac{1}{|D|+1}] = \\ =& P[\hat{X} \leq k - d + \frac{1}{a}]. \end{aligned} \tag{4.37}$$

again we must have that:

$$k \geq d - \frac{1}{a}. \tag{4.38}$$

Now if $\hat{s} > 0$ we can use a Chernoff bound with:

$$\delta = 1 - \frac{k - d + \frac{1}{a}}{\hat{s}} \tag{4.39}$$

which is applicable to Chernoff if:

$$d - \frac{1}{a} < k < \hat{s} + d - \frac{1}{a}. \tag{4.40}$$

Now we can see that for $k = d$ we have that:

$$\delta = 1 - \frac{1}{a\hat{s}} \tag{4.41}$$

and the 4.40 translates to:

$$\hat{s} > \frac{1}{a}. \tag{4.42}$$

And finally we obtain that:

$$P[\hat{X} \leq \frac{1}{a}] \leq \left(\frac{e^{\frac{1}{a\hat{s}}-1}}{(\frac{1}{a\hat{s}})^{(\frac{1}{a\hat{s}})}} \right)^{\hat{s}}. \tag{4.43}$$

When applicable we can then use 4.43 when $k = d$ instead of calculate the p-value of the candidate solution.
When $\hat{s} = 0$ we have that we cannot apply the previous Chernoff bounds but again we can note that $\hat{s} = 0$ only if there is no probabilistic contribute to s then we known directly the p-value of the candidate solution as it only depends on the amount d which is already available.

Now we have obtained an upper bound to the probability that the p-value of the candidate solution is less or equal that the p-value of the current solution.
We can use such bound to define a new greedy method to expand a valid solution. We could start with the couple of genes that provide a valid solution with the lowest p-value and then add to this initial solution the gene g with the highest upper bound (we decided also to not considerate any gene g were the bound given by (4.21) is less than a certain threshold), obtained as in (4.21), to $P[p_{val}(A, B) \geq p_{val}(A^*, B^*)]$. After this we need to calculate the p-value of solution created by expanding (A, B) with g, if the calculated p-value is greater than the current p-value we then consider the gene with the second highest upper bound, we can then reiterate such considerations until we found a valid gene. The previous consideration is valid only when for all the genes considered the condition (4.32) is applicable otherwise we first start looking at the subset of genes where due to the not applicability of (4.32) we need to find out the p-value of the candidate solution respect to such genes and then only when we cannot find a valid candidate solution constructed upon such genes we start looking the remaining genes.
If we found a valid gene we will promote the candidate solution, created starting from the current solution by adding such gene, to be the new current solution and then we then start seeking a new valid gene. We can then reiterate such approach until we cannot find any valid gene or until we have expanded the solution in order to reach a specific maximum size.

4.5.2 Approach based on martingales

We can model how a gene affect the patients in a simulated dataset using the notion of martingale.
We have a prefixed amount of gene mutations to assign uniformly to a set of patients and where we would like to find the number of mutations of a gene in respect to a specific subset patients, the *almost interesting* ones.
Supposing again that the candidate solution is obtained by adding the gene g to A in the current solution (A, B), let Y the random variable that count the number of *L-almost interesting patients* affected by a mutation in the gene g classified as Early. Let again X_i to be the indicator random variable relative to the gene mutations as defined in (4.14). Note that $Y = f(X_0, ..., X_n) = \sum_{i=0}^{n} X_i$, when $X_0 = 0$. Now we have that applying the Doob's procedure as described in (3.11) we obtain the martingale $Z_0, ..., Z_n$ where we set $Z_0 = 0$. Now, as already generally mentioned, Z_i represents the expected number of *L-almost interesting patients* when we know the value of $X_0, ..., X_{i-1}$ then supposing tho know the outcome of X_i we have that adding such information to Z_i we obtain that:

$$Z_{i+1} = Z_i - E[X_i] + X_i \tag{4.44}$$

from where it follows that:

$$|Z_{i+1} - Z_i| \leq 1. \tag{4.45}$$

We have then that $c_k = c = 1$. Now assuming to know $\triangle k^{i+1}$, k^i and k^i_x where x is a simulated dataset, note that $\triangle k^{i+1} + k^i = k^{i+1}$. Supposing now x to be an not *countable* dataset we are interested in the probability that such dataset turn out to be *countable* respect to the candidate solution:

$$\begin{aligned} P[k^i_x + \triangle k^{i+1}_x \geq k^i + \triangle k^{i+1}] &= P[\triangle k^{i+1}_x \geq k^i + \triangle k^{i+1} - k^i_x] = \\ = P[Y \geq k^i + \triangle k^{i+1} - k^i_x] &= P[Y - \mu \geq k^i + \triangle k^{i+1} - k^i_x - \mu] = \\ = P[Z_n - Z_0 \geq k^i + \triangle k^{i+1} - k^i_x - \mu] &= P[Z_n - Z_0 \geq k^{i+1} - k^i_x - \mu] \end{aligned} \tag{4.46}$$

We used the fact that Y is positive random variable, then is possible to apply the Azuma-Hoeffding bound by taking:

$$\lambda = k^{i+1} - k^i_x - \mu \tag{4.47}$$

when:

$$k^{i+1} > k^{i+1}_x + \mu. \tag{4.48}$$

we obtain that:

$$\begin{aligned} P[Z_n - Z_0 \geq k^{i+1} - k^i_x - \mu] =& P[Y - \mu \geq k^{i+1} - k^i_x - \mu] \leq \\ & \leq e^{-(k^{i+1} - k^i_x - \mu)^2/2n}. \end{aligned} \tag{4.49}$$

A stronger bound is given using (3.16), recalling how Z_i is defined we have that:

$$\begin{aligned} Z_i = E[Y|X_0, ..., X_i] = E[\sum_{i=0}^{n} X_i | X_0 = a_0, ..., X_i = a_i] = \sum_{k=0}^{i} a_k + E[\sum_{k=i+1}^{n} X_k] = \\ = \sum_{k=0}^{i} a_k + \sum_{k=i+1}^{n} E[X_k], \end{aligned} \tag{4.50}$$

we have that $Z_k - Z_{k-1}$ is bounded by:

$$0 - E[X_k] \leq Z_k - Z_{k-1} \leq 1 - E[X_k], \tag{4.51}$$

which is obtained conditioning on the value of X_k, we can then derive the following relation:

$$P[Z_n - Z_0 \geq k^i + \triangle k^{i+1} - k^i_x - \mu] =\leq e^{-2(k^{i+1} - k^i_x - \mu)^2/n}, \tag{4.52}$$

clearly the upper bound obtained by this approach is more strict.
Note that according to the condition (4.48) and according to the condition (4.21) the subset of datasets where either the Azuma-Hoeffding or the Chernoff bound is applicable is the same as the two condition are mathematically identical.
Then similarly as we done previously we can define a score $s(x)$ for any simulated dataset x equal to the probability that x remains or becomes *countable* respect to the current solution. On the dataset *at risk* where the condition (4.48) is satisfied we use the upper bound in (4.49) as the score of $s(x)$, otherwise we calculate k^{i+1}_x and the we set $s(x) = 1$ if $k^i_x \leq k^i$ otherwise we set $s(x) = 1$. For any dataset not *at risk* $s(x) = 1$

if x is *countable* while for the not *countable* ones $s(x) = 0$.
We can then define X_x the indicator random variable associated to the dataset x that indicate if the dataset x becomes or remains *countable* or if it turns out to be not *countable*. We have that $P[X_x = 1] = s(x)$, and X_x is a Bernoulli random variable.
Supposing now X to be the random variable that count the number of *countable* dataset respect to the candidate solution. Again we would like to find the probability that the p-value of the current solution is lower or equal to the p-value of the current solution. We can then applying the same consideration done in the previous section use a Chernoff bound where with δ as in (4.31) when the condition (4.32) is valid we obtain again:

$$P[\hat{X} \leq k - d] \leq \left(\frac{e^{\frac{k-d}{\hat{s}} - 1}}{\left(\frac{k-d}{\hat{s}}\right)^{\left(\frac{k-d}{\hat{s}}\right)}} \right)^{\hat{s}}. \tag{4.53}$$

Even that such bound is the mathematically the same as in (4.33) the way on how the value $\hat{s}$ is derived is different as we used a completely different approach in order to derive it.
We can now use the same greedy approach described as in the previous section in order to create and expand a valid solution to our problem.

Chapter 5

Results

In this section we describe the preprocessing part done on data and some characteristics of the dataset used.
Then we illustrate some of the results obtained running an implementation of the algorithms described in Chapter 4 and provide an interpretation of them.

5.1 Datasets

We have considered two different datasets extracted from the Cancer Genome Atlas.

The first dataset comes from [4] and is relative to a set of patients that has a specific type of brain tumor abbreviated as LGG (Low Grade Glioma) the second dataset comes from the same study but is relative to a set of patients that has another type of brain tumor abbreviated as GBM (Glioblastoma Multiforme).

We decided to exclude from the study every record of the datasets where the value of `VARIANT CLASSIFICATION` is equal to `Silent` as the effects produced by those mutation are clinically irrelevant.
A script was created in order to filter only a set of specific headers, exclude invalid records and filter only clinically significant `VARIANT CLASSIFICATION` classes.

The GBM dataset consists of 36467 different records with 139 headers and involves 276 patients and a set of 11382 genes while the LGG one has 11533 records and 108 headers and the number of patients and genes are respectively 504 and 5554.
The `VARIANT CLASSIFICATION` classes are not standard, each experiment use its own nomenclature, in the GBM dataset classes are more specific respect to the LGG dataset.

In the tables 5.1 and 5.2 are summarized the different `VARIANT CLASSIFICATION` classes, the number of records that fall in a specific class and the number of records with a score s_r greater than 0. We see that the resulting matrix from the LGG dataset is very sparse as there are only 562 record whose score are greater than 0 instead in the GBM every entry has a significant score. So we decided to exclude from our next tests the LGG dataset because it is not sufficiently rich in data.

Looking at figure 5.1 that shows how many patients have a given number of mutations, we can see that from the GBM dataset the number of mutated genes in a patient range from about 7000 to about 5 and the most part of patients have a number of mutated genes in the range from 20 to 100. Figure 5.2 shows an histogram for the the cumulative score of a patient, the sums of the scores relative to all the genes in a specific patient, in the number of patients where we can see that the cumulative score is concentrated in the range from 0 to 100. Tables 5.3 and 5.4 show the 10 patients with the most number of mutated genes and with the least number of mutated genes respectively, with their number of mutated genes and their cumulative score. Is possible to note that the number of mutated genes for a GBM afflicted patient range from as little as 5 genes to thousands of mutated genes.

The number of genes with at least a mutation in the GBM dataset consists of 9859 genes and given the fact that we can choose from $\binom{|G|}{2}$ different couples we considered only the genes that are mutated in at least in the 2% of patients let G_t to be this subset of G, this subset is 747 in size and consequently we considered 278631 different couples.

For the clonal and subclonal classification part we have that figure 5.3 shows for every gene the fraction of patients where such gene is classified as Early and as Late respectively. We see that there is a rather low correlation between the fraction of patients where a gene is classified as Early and as Late. In table 5.5 we can see for various relation the number of statistically significative couple of genes, their average p-value, and the standard deviation on the p-value. Referring to the Early-Late and Late-Early relations the average p-value is generally high as, also the standard deviation, we expect that most of the p-values to be high as the number of true relations tends to be rather small. Figure 5.4 shows the distribution of the p-values for some relations.

Finally if we consider the full set of genes we have that in the average patient about 53 genes are classified as Early and about 35 as Late.
In the full set of genes on average for a generic couple of genes (a, b) the 98.635% of patients are missing the mutation in both genes and 1.2226% of patients do not have a mutation in the gene a or in the gene b while in the remaining 0.1424% of patients both genes are mutated.

`VARIANT CLASSIFICATION class`	`#records`	`#records with` $s_r > 0$
De_novo_Start_InFrame	6	6
De_novo_Start_OutOfFrame	10	10
Frame_Shift_Del	814	814
Frame_Shift_Ins	363	363
In_Frame_Del	329	329
In_Frame_Ins	38	38
Indel	1	1
Missense	6	6
Missense_Mutation	23565	23565
Nonsense_Mutation	1747	1747
Nonstop_Mutation	17	17
Silent	8955	8955
Splice_Site_DNP	11	11
Splice_Site_Del	102	102
Splice_Site_Ins	28	28
Splice_Site_ONP	1	1
Splice_Site_SNP	469	469
Start_Codon_Del	2	2
Start_Codon_Ins	1	1
Stop_Codon_Del	2	2
`Total`	36467	36467

Table 5.1: `Variant Classification` classes statistics for the GBM dataset.

`VARIANT CLASSIFICATION class`	`#records`	`#records with` $s_r > 0$
Frame_Shift_Del	467	15
Frame_Shift_Ins	122	4
In_Frame_Del	164	6
In_Frame_Ins	7	0
Missense_Mutation	7149	359
Nonsense_Mutation	529	36
Nonstop_Mutation	6	0
Silent	2622	124
Splice_Site	467	18
`Total`	11533	562

Table 5.2: `Variant Classification` classes statistics for the LGG dataset.

Patient ID	Number of mutation	Cumulative score
'TCGA-06-5416-01'	6969	2415.38
'TCGA-32-2616-01'	817	295.80
'TCGA-06-1802-01'	569	313.04
'TCGA-19-1787-01'	503	290.46
'TCGA-16-0848-01'	246	86.00
'TCGA-12-0829-01'	228	111.38
'TCGA-19-1389-01'	223	90.91
'TCGA-28-1760-01'	203	85.17
'TCGA-14-0866-01'	176	63.03
'TCGA-06-2566-01'	163	59.77

Table 5.3: Top-k patient in mutation numbers terms, $k = 10$.

Patient ID	Number of mutation	Cumulative score
'TCGA-06-0139-01'	5	1.89
'TCGA-32-1980-01'	6	1.90
'TCGA-06-0240-01'	8	2.83
'TCGA-06-0178-01'	9	3.18
'TCGA-06-0189-01'	15	2.81
'TCGA-06-0881-01'	16	4.55
'TCGA-14-1821-01'	19	9.58
'TCGA-06-0132-01'	20	3.93
'TCGA-28-2510-01'	21	4.57
'TCGA-06-0151-01'	22	8.13

Table 5.4: Min-k patient in mutation numbers terms, $k = 10$.

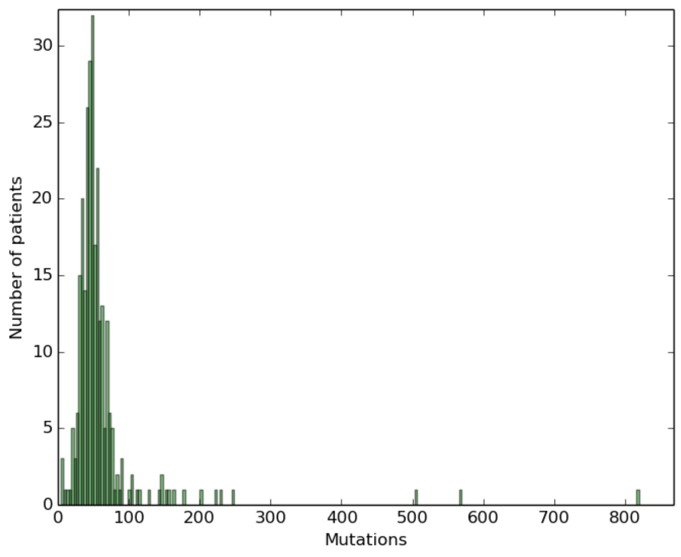


Figure 5.1: Histogram for the number of mutations in the number of patients.

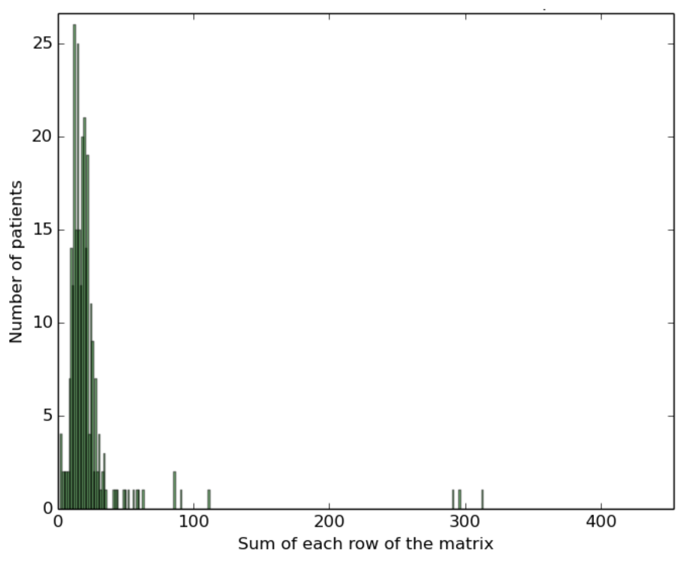


Figure 5.2: Histogram for the cumulative score in the number of patients.

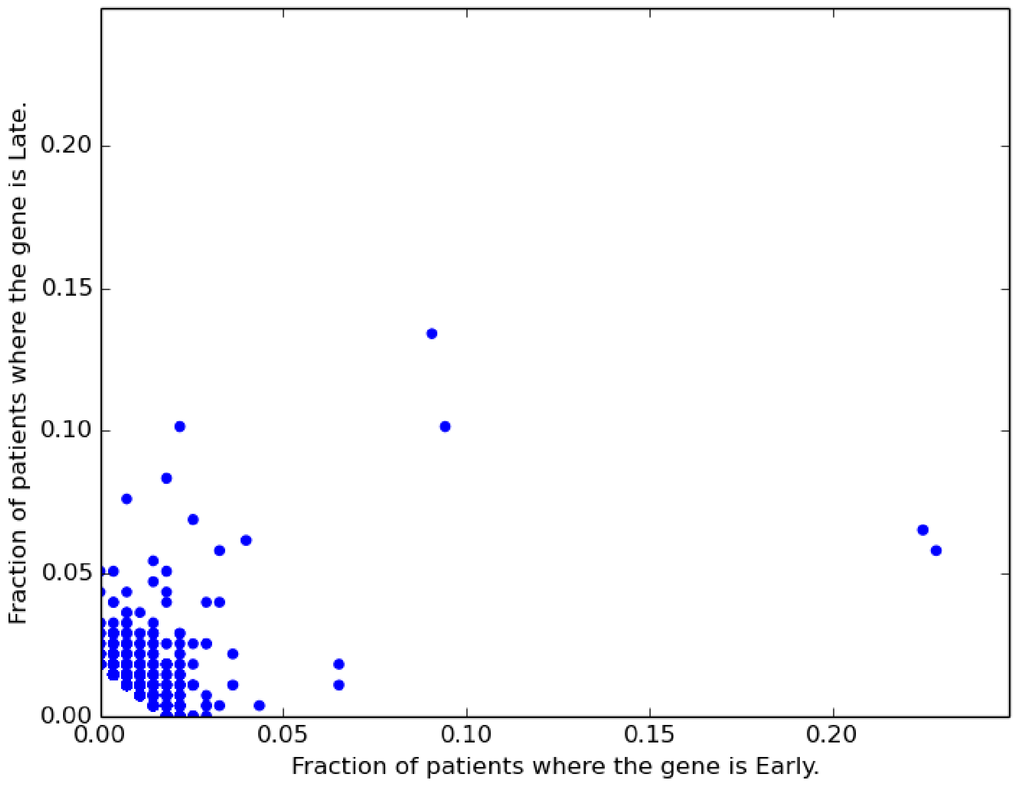
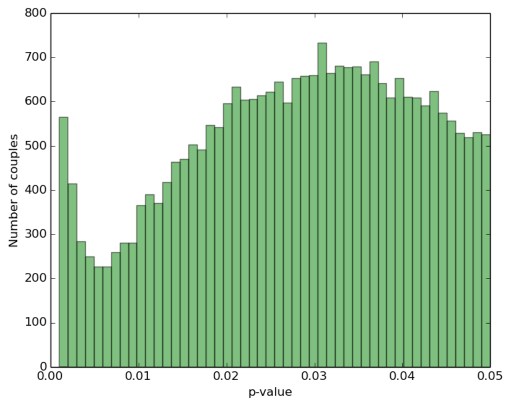


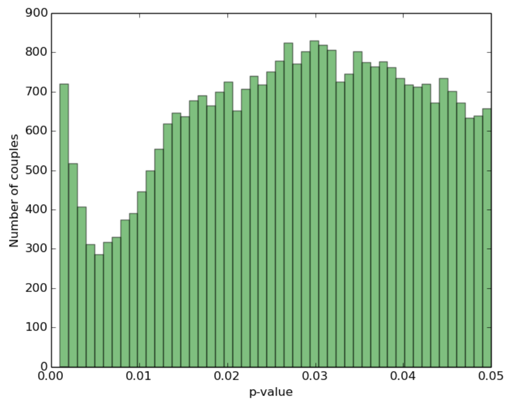
Figure 5.3: Scatter for the percentage of patient with a specific gene classified as Early or Late, Pearson correlation of 0.2297.
Each point represents a gene involved in the study, the coordinates of a point are the fraction of patients where such gene is classified as Early, on the x-axis, and respectively the fraction patients where it is classified as Late, on the y-axis.

`Relation`	`Stat.sign.couples`	`Average` p`-value`	p`-value std.dev.`
Early-Late	26577	0.8338	0.3585
Late-Early	32649	0.8027	0.3840
Early-Early	24392	0.8708	0.3255
Late-Late	41622	0.6996	0.4373

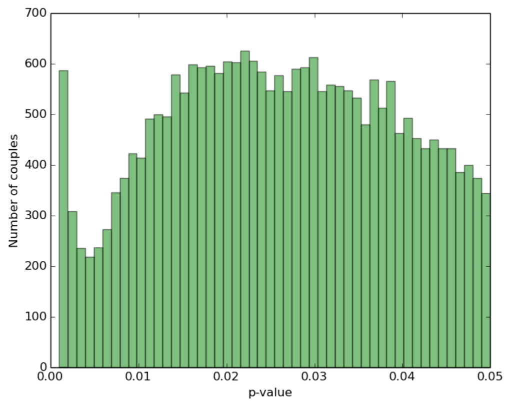
Table 5.5: Statistics for the different relations using n=1000 and α=0.05.



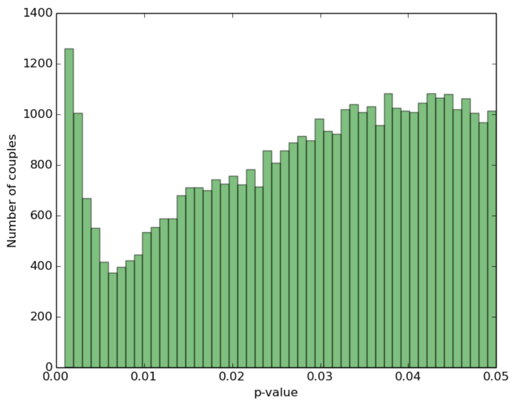
(a) Early-Late



(b) Late-Early



(c) Early-Early



(d) Late-Late

Figure 5.4: Histogram for the p-value of the statistically significant couples in various relations, $\alpha = 0.05$, $n = 1000$.

5.2 Description of the experiments

We have different implementations that differs by small variations that yield particular improvements.

The different implementations of the algorithm following the consideration that are presented in the previous chapter are:

- **Greedy**: This implementation initialize a random simulated pool of dataset in order to perform a Monte Carlo simulation every time we need to calculate the p-value of a candidate solution. Each simulated dataset only contains the genes that are in the candidate solution. As the pool of simulated dataset is recreated continuously the optimizations that are discussed in the subsection 4.4.1 are not applicable.

- **Fixed**: In this implementation the pool of random datasets used in order to calculate the p-value are initialized at the start of the program and are keep the same for every needed computation, we refer to such fact using the *fixed datasets* terminology. Hence such pool of datasets contains all the considered genes regards of any candidate solution. In this implementation we can also skip to check patients that were already *interesting* in a previous iteration's solution.

- **Bound**: Like the previous implementation we use *fixed datasets* but this time we also apply the optimizations discussed in the section 4.4.1 while ignoring the observation given by the equations in 4.13.

- **Bound***: This implementation is like *Bound* implementation except for the fact the we decided to remove from the considered genes every genes that form a couple of genes with a p-value equal to the lower bound $\frac{1}{n+1}$, where n is the number of simulated datasets. Every algorithm with an * after its name implements this decision/option.

- **Imp. B.***: In this implementation *fixed datasets* are used and the optimizations discussed in the section 4.4.1 are fully applied.

- **Chernoff***: From this implementation we start using the probabilistic approach described in section 4.5 while also using the optimization of 4.4.1. In particular we apply the method described in 4.5.1.

- **Azuma***: Like the *Chernoff** implementation but this time we are going to use the martingale's approach as described in the subsection 4.5.2.

- **Hybrid***: Take the advantages given by both of the previous implementations by using the tail distribution bound that gives more information (takes the lowest one) between the Chernoff bound and the Azuma-Hoeffding one.

For every implementation we also provide two alternative starting options:

- **Min**: The implementation start from a couple of genes that yields the minimum and statistically significative p-value ($\alpha \leq 0.05$) while taking in consideration all the distinct couple of genes from the considered ones (if * option is also chosen it also exclude the genes that form a couple with a p-value of $\frac{1}{n+1}$).

- **Max**: Like the *Min* option but starting from the couple with the maximum but still statistically significative p-value ($\alpha \leq 0.05$) (while also being compatible with the * option).

Every time we have more than one candidate solution with the lowest p-value we must decide which one is going to be promoted as the new current solution. We have 3 different choices:

- **L**: We decide to choose among the candidate solutions that provide the lowest p-value the last one (as every gene is tested following a specific order).

- **R**: We choose one solution uniformly at random.

- **B**: Given a solution (A, B) we try to keep the size of the set of genes A similar to the size of the set B, promoting as current solution a solution that increments the size of the set of genes (A or B) with the lowest size. If more than one solution accomplish this we select one uniformly at random. If $|A| = |B|$ we apply the criterion defined by R.

A more interesting choice is discussed in the conclusions.

In all the following tests we used the same simulation parameters:

- we consider the GBM dataset
- $n = 250000$ (the number of simulated random datasets used)
- $\alpha \leq 0.05$ (the significance level used)
- maximum solution size $(|A \cup B|) = 10$
- the set of the considered genes G_t is equal to the set of genes that appear in at least the 4% of the patients ($|G_t| = 57$, if * is chosen then $|G_t| = 55$)

We also excluded from the benchmarks the time needed in order to initialize *fixed dataset* (generally for $n = 250000$ is less than 30 minutes).

5.3 Discussion of the results

A first fact that can be noted is that not all the experiments end reaching the maximum size limit, some solution are lower in size. On this fact we have that all the experiments that start with the MAX couple according to table 5.7 reach the maximum size limit while some of the experiments that start from the MIN couple according to table 5.6 end with smaller solution in size.
An explanation could be in the observation that starting from the MIN couple that usually has a much lower p-value than the MAX couple restrict the size of the possible candidate solution as we have less possible genes (a candidate solution is constructed starting from the current solution by adding a candidate gene) that can be used to improved a p-value that is already rather small resulting in the fact that we have less chance to find admissible candidate solutions.

The use of the Choice B used in order to provide solutions that are balanced in size yields the desired result in all the implementations that do not use the probabilistic approach of section 4.5 that start from the MAX couple and in most part of the ones that start with the MIN couple. Again the previous observation could be the explanation regards to the fact that some of the deterministic implementations that start from the MIN couple and use the Choice B are not balanced.

We note that the Choice B do not have the desired effect on most part of the probabilistic implementation experiments. This could be an effect of the fact that when we have a set of candidate solutions where the Chernoff bound is applicable we use such upper bound as ulterior greedy choice criterion that permits to avoid the explicit computation of the p-values of the solution that are in such set by choosing the "most promising" one and then verifying if such promising solution is admissible (has a p-value that is actually not greater than the current solution), otherwise we skip to the one with the second lowest upper bound reiterating the approach. Considering only one candidate solution at time and taking the "most promising" and admissible one can be viewed as a sort of "a priori" choice before the application of the Choice B (only if we find more than one candidate that "is promising" we can apply the Choice B) and hence we have that the choices of balancing solutions are less (there can be a more balanced in size candidate solution that is not very "promising" while actually yielding a p-value that can be lower or equal than the "most promising" ones). This can also explain why some solution of the experiments that use the probabilistic approach do not reach the lower bound of the p-value, which in the case of $n = 250000$ is equal to $3.999e - 06$.

Looking at 5.5 we see that generating a set of random datasets at each p-value computation raise consistently the average iteration respect to the *fixed datasets* approach. We also see that using the observations of section 4.4.1 provide a tangible computation time advantage respect to not using them. Referring to figure 5.9 that shows the percentage of skipped p-value computation of the bound given by the equation 4.11 we see that the bound tends to be not very strict providing only a minor computational advantage that is completely nullified after the first iterations. Generally we note that more we reach the lower bound of the p-value more the bound 4.11 is useless. For the the previous observation ("a priori" choice) we have that the probabilistic approach is order of magnitude more rapid than the deterministic one.
These observation remains consisted also in the case of L and R as choices. If the add to the experiments that use the *fixed datasets* approach the time needed for generate such datasets divided by the number of iteration than: the difference between using and not using *fixed datasets* remains tangible, the implementations that have less iterations see an increment of their average iteration time that is much higher than the implementations with a higher number of iterations (see A-H* and Hybrid* from table 5.6).

For each experiment after deriving a final solution we have recalculated the p-value of such solution using a set of random datasets that is different from the one used during the experiment. Given this fact the recalculated value can differ from the one calculated during the execution of one implementation. In tables 5.6 and 5.7 we can find such value under the column "$p\#$". In figure 5.6 we can see that the difference from the p-value of the starting solution and the $p - value$ of the final solution calculated during one experiment (p-START - p-END) and the difference between the p-value of the starting solution and the p-value of the final solution recalculated on a different set of random datasets (p-START - $p\#$) are most of times similar (using B as choice), as generally p-START >> p-END and p-START >> $p\#$. The only exceptions involve the solutions of the experiments that start from the MIN couple, in particular from the implementations: Hybrid*, Imp. B* and Bound. In figure 5.7 we have mapped with a scatter plot the above discussed quantities.

In figure 5.8 we can see how the p-value of the current solution iteration by iteration stabilize rapidly at the p-value's lower bound except for the solution of the experiment that use the configuration (MIN, Hybrid*, B) where we have a size that is well less than the maximum threshold (the size of such solution is 4, the maximum threshold is 10). Generally for all deterministic implementations we a have a rapid stabilization of the p-value on its lower bound. For the probabilistic implementations the stabilization is more slow and sometimes the small size of the final solution do not permits to reach the p-value's lower bound.

In figure 5.10 we can see for how many candidate solution, iteration by iteration, the Chernoff bound can be used and for how many candidate solutions we need to explicit calculate the p-value due the not applicability of the Chernoff bound. If we focus on the MAX couple case, that is more representative than MIN couple one (we have data about only 3 iteration), we can see that after some iteration the number of candidate solution where we can use Chernoff as metric rise rapidly to slowly decrements in the next iterations. This was generally observed also in the A-H* and Chernoff* implementations when enough iterations are granted.

Finally we note that the set of random dataset from where we have calculated the statistic that are used in order to choose the starting couple and hence the starting solution differs from the one used in every implementation, this reflect into the fact that p-value of such starting solution can vary and particularly in the case of starting solutions that yields a p-value that is near to significance level α we have that respect to a different set of random dataset such starting solution can not be significative anymore (see table 5.7).

Implementation	**Choice**	**|SetA|**	**|SetB|**	***p*-START**	***p*-END**	*p*#
Greedy	L	6	1	1.679e-04	3.999e-06	1.199e-05
	R	7	2	2.319e-04	3.999e-06	1.199e-05
	B	5	3	1.719e-04	3.999e-06	7.999e-06
Fixed	L	8	2	1.639e-04	3.999e-06	3.999e-06
	R	6	4	1.639e-04	3.999e-06	2.399e-05
	B	7	3	1.639e-04	3.999e-06	3.999e-06
Bound	L	5	3	1.919e-04	3.999e-06	1.999e-05
	R	8	1	1.719e-04	3.999e-06	1.999e-05
	B	5	5	2.159e-04	3.999e-06	2.799e-05
Bound*	L	7	1	1.959e-04	3.999e-06	7.999e-06
	R	8	2	1.919e-04	3.999e-06	1.999e-05
	B	5	5	1.679e-04	3.999e-06	3.999e-06
Imp. B.*	L	8	1	1.439e-04	3.999e-06	2.399e-05
	R	2	8	1.919e-04	3.999e-06	1.199e-05
	B	5	5	1.839e-04	3.999e-06	3.199e-05
Chernoff*	L	2	2	2.079e-04	7.999e-06	3.599e-05
	R	9	1	1.879e-04	3.999e-06	5.199e-05
	B	8	1	2.359e-04	7.999e-06	7.999e-06
A-H*	L	8	2	2.079e-04	3.999e-06	1.599e-05
	R	7	3	1.879e-04	3.999e-06	3.199e-05
	B	2	1	1.839e-04	1.319e-04	1.319e-04
Hybrid*	L	8	2	2.039e-04	3.999e-06	2.399e-05
	R	2	2	2.239e-04	1.999e-05	2.399e-05
	B	2	2	1.919e-04	1.199e-04	3.199e-05

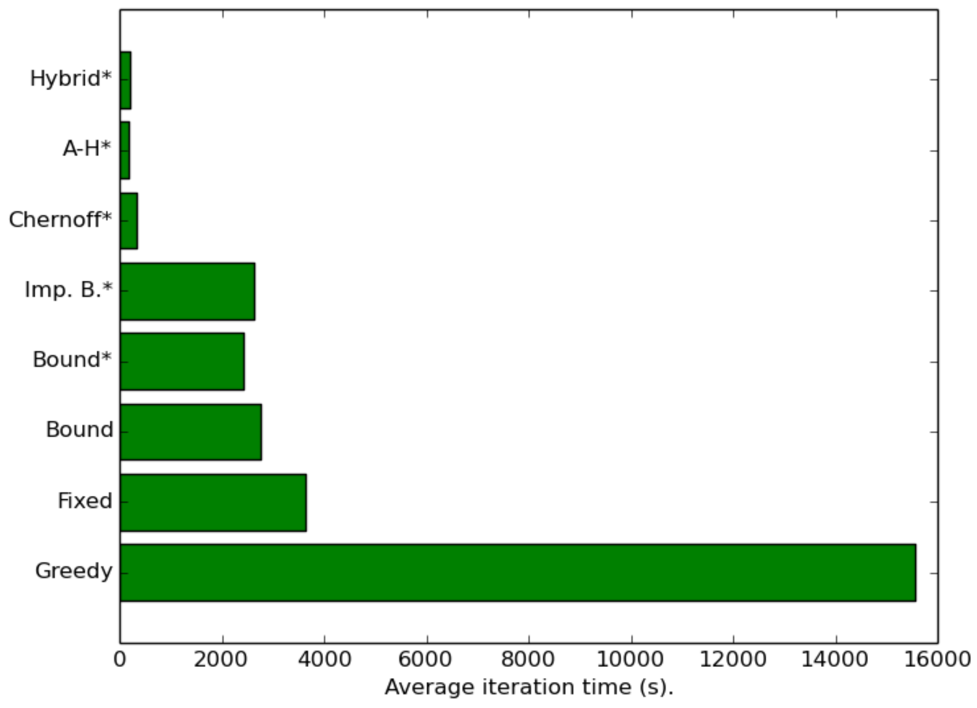
Table 5.6: Results from different implementations starting from MIN couple.

We have listed in order: the implementation used (Implementation), the choice used (Choice), the size of the final solution (|setA| and |setB|), the *p*-value of the starting solution (*p*-START), the calculated *p*-value of the final solution by a given experiment configuration (*p*-END), the recalculated *p*-value of the final solution under a different set of random datasets (*p*#).

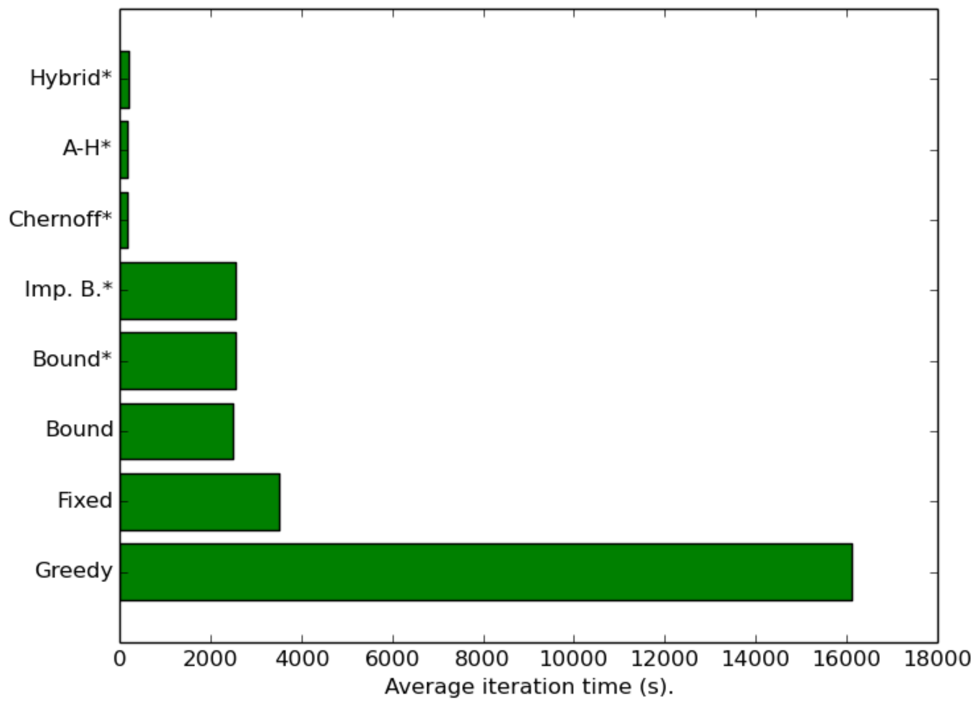
Implementation	**Choice**	**|SetA|**	**|SetB|**	***p*-START**	***p*-END**	*p*#
Greedy	L	7	3	5.013e-02	3.999e-06	3.999e-06
	R	6	4	5.049e-02	3.999e-06	3.999e-06
	B	5	5	4.985e-02	3.999e-06	1.199e-05
Fixed	L	8	2	4.977e-02	3.999e-06	3.999e-06
	R	6	4	4.985e-02	3.999e-06	2.799e-05
	B	5	5	4.985e-02	3.999e-06	1.199e-05
Bound	L	6	4	5.009e-02	3.999e-06	7.999e-06
	R	8	2	5.027e-02	3.999e-06	1.199e-05
	B	5	5	5.028e-02	3.999e-06	3.999e-06
Bound*	L	8	2	5.019e-02	3.999e-06	2.399e-05
	R	8	2	5.085e-02	3.999e-06	7.999e-06
	B	5	5	4.964e-02	3.999e-06	7.999e-06
Imp. B.*	L	6	4	5.024e-02	3.999e-06	1.199e-05
	R	7	3	4.997e-02	3.999e-06	1.599e-05
	B	5	5	4.944e-02	3.999e-06	3.999e-06
Chernoff*	L	7	3	5.039e-02	3.999e-06	1.599e-05
	R	7	3	5.023e-02	3.999e-06	3.999e-06
	B	7	3	5.089e-02	3.999e-06	6.399e-05
A-H*	L	8	2	5.072e-02	3.999e-06	6.799e-05
	R	7	3	5.026e-02	3.999e-06	7.999e-06
	B	7	3	5.107e-02	3.999e-06	5.599e-05
Hybrid*	L	7	3	5.024e-02	3.999e-06	6.399e-05
	R	7	3	5.027e-02	3.999e-06	7.999e-06
	B	7	3	5.019e-02	3.999e-06	5.199e-05

Table 5.7: Results from different implementations starting from MAX couple.

We have listed in order: the implementation used (Implementation), the choice used (Choice), the size of the final solution (|setA| and |setB|), the *p*-value of the starting solution (*p*-START), the calculated *p*-value of the final solution by a given experiment configuration (*p*-END), the recalculated *p*-value of the final solution under a different set of random datasets (*p*#).



(a) Starting from MIN couple.



(b) Starting from MAX couple.

Figure 5.5: Average iteration time for different implementations. Choice = B.

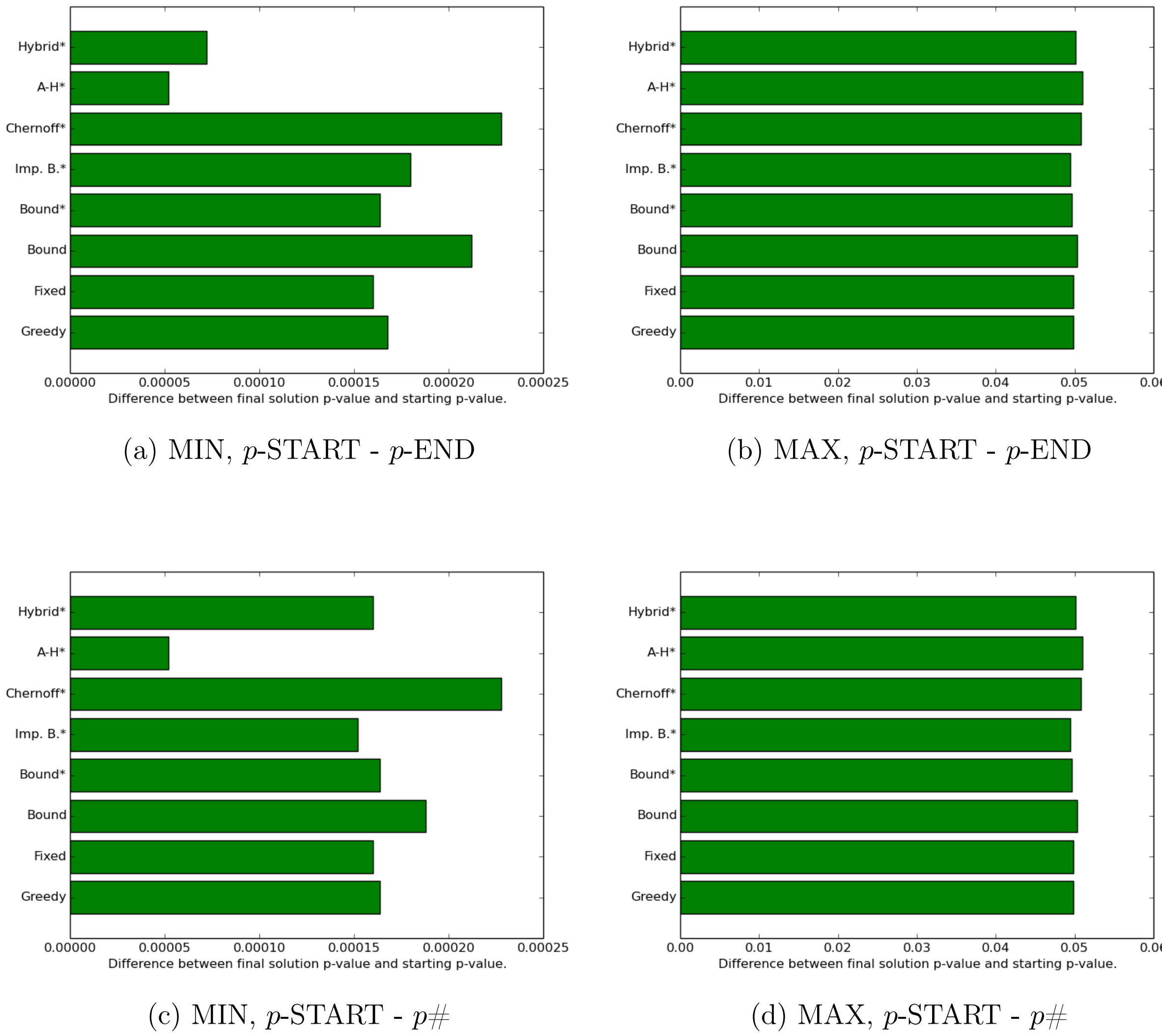
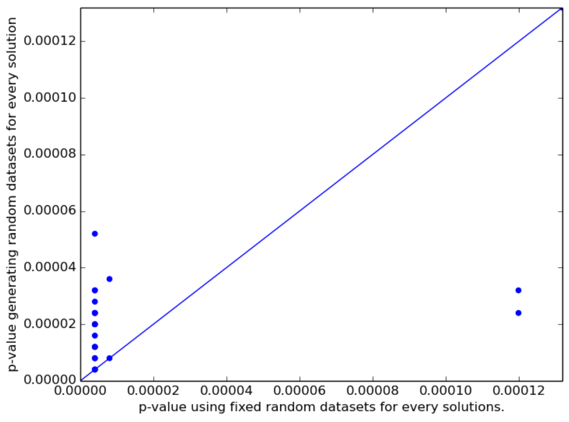
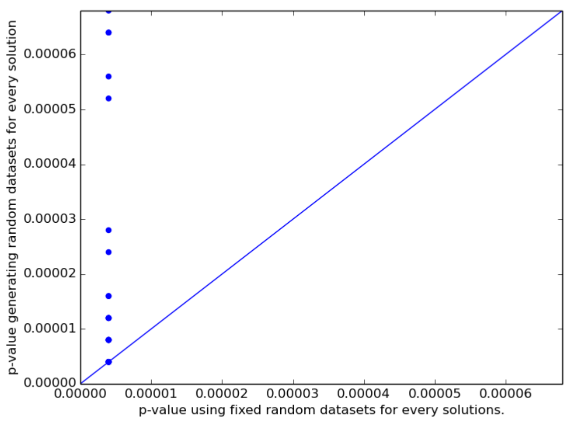


Figure 5.6: Histograms for the difference of the starting solution p-value from the final solution p-value and for the difference of the starting solution p-value from the p-value of the final solution recalculated using a different set of random dataset. Choice = B.

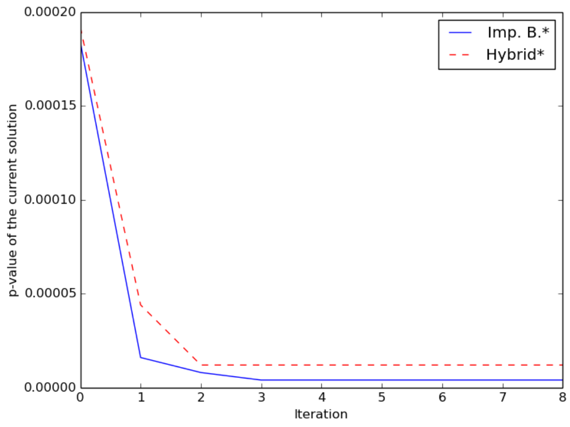


(a) Starting from MIN couple.

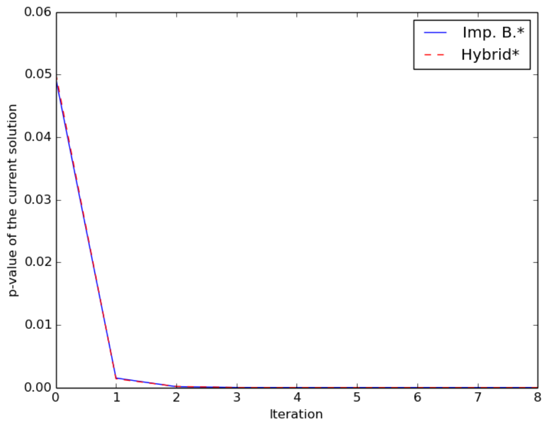


(b) Starting from MAX couple.

Figure 5.7: Scatter for the p-value of each final solution (p-END) and for the recalculated p-value of each final solution ($p\#$). Choice = B.



(a) Starting from MIN couple.



(b) Starting from MAX couple.

Figure 5.8: p-value of the current solution iteration by iteration for *Imp.B** and *Hybrid**. Choice = B.

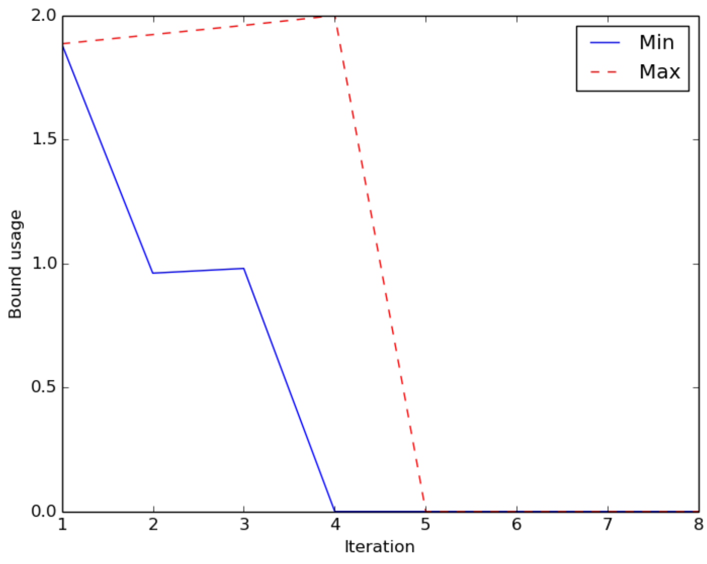
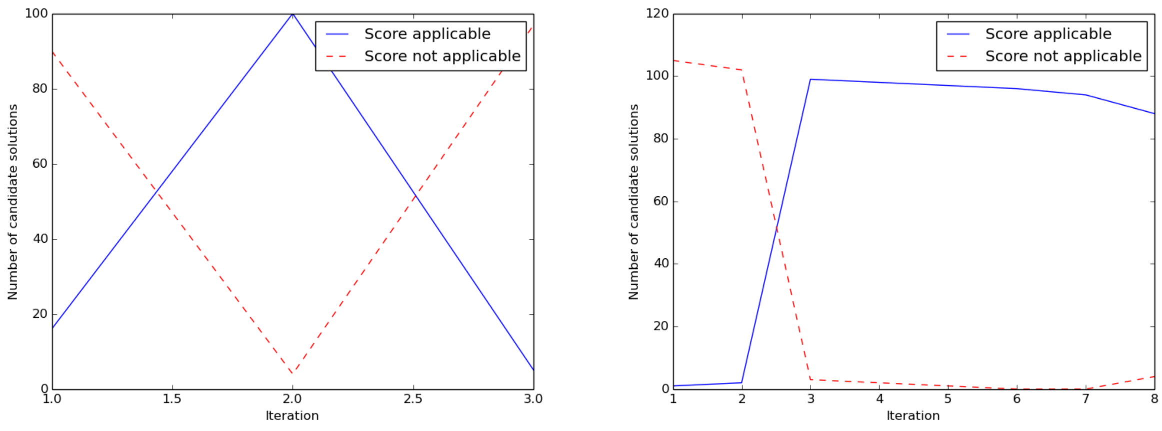


Figure 5.9: Deterministic bound usage in percentage of skipped p-value computations iteration by iteration using Imp. B.* method. Choice = B.



(a) Starting from MIN couple.

(b) Starting from MAX couple.

Figure 5.10: Applicability and not applicability of (4.33) or of (4.43) using the Hybrid* method. Choice = B.

Starting from a current solution (A, B) we obtain a candidate solution by adding a gene that is not in the current solution to A or to B. For each of such candidate solutions we can have a "score" based on the Chernoff bound only if it is applicable.

Chapter 6

Conclusions

In this work we have proposed an algorithmic methodology used to find statistically significative progression patterns on cancer development firstly classifying genes according to their clonal status and then expanding such classification to couple of sets of genes. Then we focused on improving the computational performance of the proposed algorithm by observing that if the Monte Carlo simulation is based on the usage of *fixed datasets* is possible to derive an alternative way to calculate the p-value of two sets of genes skipping computations on datasets that are not *at risk* while also providing an upper and lower bound on the p-value of a candidate solution constructed from a given one. From experimental results the time performance gained by using *fixed datasets* approach is very high, then refining such approach introducing the alternative p-value computation method and using the discussed bound an ulterior gain in computation time performance is obtained. After that we concentrated in providing an ulterior enhancement to the algorithm providing a probabilistic model on the dynamic of the Monte Carlo simulation while using *fixed datasets* from where we defined a new greedy choice based on tail distribution bounds as an indicator of the presumed quality of a candidate solution that permits to avoid a large part of the p-value computations for the possible candidate solutions. This new algorithm provide an ulterior high gain in computation time over the deterministic approach while not significantly impacting the quality of the computed solution.

Future work is needed in order to provide a more suitable greedy choice for the deterministic approach. In particular we can try to promote as current solution the solution that maximize the variability of the status off the datasets that are *countable* while maintaining the status of the datasets that are not *countable*, in order to not increment the p-value of the next solution. Clearly this new greedy choice can also be formalized in a way to boost the predictive power of the probabilistic approach.
Looking at the results gathered we can also note that, except for the B choice, given a final solution $(setA, setB)$ the size of the set the $setA$ is usually greater than the $setB$. More research is needed in order to see if the modern methods of detecting cancer genes are biased in identifying clonal mutations instead of subclonal ones.
More work can be concentrated in studying how much the proposed approach is susceptible to over fitting dynamics.
Then is possible to add to our approaches the biological information that come from known Protein-Protein Interaction networks (PPI) and validate if such information

provide a tangible advantage over the non usage.
Finally we can test our algorithms in datasets relative to different tumor types providing a sort of Pan-Cancer study.

Appendix A

Floating point limit

In order to calculate the bound provided by 4.21 we need to be aware that *float* and *double* number follows an encoding based on the IEE 754 standard that is able to represent only a subset of the infinite real number. Under such encoding is then impossible to represent certain number, for example trying to represent a number that is greater than the maximum number that can be encoded raise an *overflow* error.
Even that the bound provided by 4.21 appertains to the real interval $(0, 1)$ the numerator or especially the denominator of such formula can exceeds the maximum encodable number raising an overflow error. One trick to avoid this problem follows.
Let:

$$b = \frac{k^{i+1} - k_x^i}{g_E \frac{m}{n}}, t = g_E \frac{m}{n} \tag{A.1}$$

now we have that 4.21 is equal to:

$$P[X \geq k^{i+1} - k_x^i] \leq \left(\frac{e^{b-1}}{b^b} \right)^t = w \tag{A.2}$$

Now when w is more than 0 we can apply the logarithmic function:

$$\begin{aligned} log(w) = log \left(\frac{e^{b-1}}{b^b} \right)^t = t(log(\frac{e^b}{b^b}) + log(\frac{1}{e})) = t(log(e^b) - log(b^b) - 1) = \\ = t(b(1 - log(b)) - 1). \end{aligned} \tag{A.3}$$

Now we again obtain w by exponentiation:

$$w = e^{log(w)} = e^{t(b(1-log(b))-1)}. \tag{A.4}$$

This new definition permits to avoid the exponentiation of b^b at the denominator that can potentially cause an overflow error.
Same considerations can be applied to 4.33 and 4.43.

Bibliography

[1] Brent A. Coull Alan Agresti. Approximate is better than "exact" for interval estimation of binomial proportions. *The American Statistician*, 52(2):119–126, 1998.

[2] Cancer Genome Atlas. Home page.

[3] Neil A Campbell, Jane B Reece, Lisa Urry, Michael L Cain, and Steven A Wasserman. *Biology: a global approach*. Pearson Higher Ed, 2014.

[4] Michele Ceccarelli, Floris P Barthel, Tathiane M Malta, Thais S Sabedot, Sofie R Salama, Bradley A Murray, Olena Morozova, Yulia Newton, Amie Radenbaugh, Stefano M Pagnotta, et al. Molecular profiling reveals biologically discrete subsets and pathways of progression in diffuse glioma. *Cell*, 164(3):550–563, 2016.

[5] Michel Goemans. Lecture notes on chernoff bounds.

[6] Lawrence Hunter and Joshua Lederberg. Artificial intelligence and molecular biology. In *AI Magazine*, 1993.

[7] Neil C Jones and Pavel Pevzner. *An introduction to bioinformatics algorithms*. MIT press, 2004.

[8] E.L. Lehmann and J.P. Romano. *Testing Statistical Hypotheses*. Springer Texts in Statistics. Springer New York, 2008.

[9] Mark DM Leiserson, Fabio Vandin, Hsin-Ta Wu, Jason R Dobson, Jonathan V Eldridge, Jacob L Thomas, Alexandra Papoutsaki, Younhun Kim, Beifang Niu, Michael McLellan, et al. Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes. *Nature genetics*, 47(2):106–114, 2015.

[10] David JC MacKay. Introduction to monte carlo methods. In *Learning in graphical models*, pages 175–204. Springer, 1998.

[11] Nicholas McGranahan, Francesco Favero, Elza C de Bruin, Nicolai Juul Birkbak, Zoltan Szallasi, and Charles Swanton. Clonal status of actionable driver events and the timing of mutational processes in cancer evolution. *Science translational medicine*, 7(283):283ra54–283ra54, 2015.

[12] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.

[13] Nature.com. Genetic mutation.

[14] PC Nowell. The clonal evolution of tumor cell populations. *Science (New York, NY)*, 194(4260):23, 1976.

[15] University of Geneve. Slides on monte carlo simulation.

[16] Mark Pinsky and Samuel Karlin. *An introduction to stochastic modeling*. Academic press, 2010.

[17] Benjamin J Raphael, Jason R Dobson, Layla Oesper, and Fabio Vandin. Identifying driver mutations in sequenced cancer genomes: computational approaches to enable precision medicine. *Genome medicine*, 6(1):1, 2014.

[18] PennState university. Lecture notes on statistics.

[19] PennState university. Review on basic statistics.

[20] George H. Van Doorn, Dianne Wuillemin, and Charles Spence. Does the colour of the mug influence the taste of the coffee? *Flavour*, pages 3–10, 2014.

[21] Fabio Vandin, Eli Upfal, and Benjamin J. Raphael. Algorithms and genome sequencing: Identifying driver pathways in cancer. *IEEE Computer*, 45(3):39–46, 2012.

[22] Wikipedia. Beta distribution.

[23] Wikipedia. Binomial proportion confidence interval.

[24] Wikipedia. Martingale (probability theory).

[25] Wikipedia. Mutation.