RELATÓRIO DE PROJECTO EM BIOLOGIA CELULAR E MOLECULAR

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Ciências da Vida

Roberto Costa Bullitta, Nº52962

MutantGene: A tool to assist in gene characterization from tumour samples.

Orientador: Ludwig Krippahl
Local: Faculdade de Ciências e Tecnologias
Ano Lectivo: 2020/2021

Novembro de 2020

Acknowledgments

I first want to thank my faculty, Faculdade de Ciências e Tecnologias, for teaching me so much over the past 3 years with fascinating subject matters and excellent Professors.

I would like to give a special thank you to my coordinator, Professor Ludwig Krippahl, for accepting to guide me through this project and giving me a taste of the vast world of bioinformatics, all whilst having already so much other work to get done. For that, and for helping to cement my passion for integrating programming within the field of biology, I am truly grateful!

Abstract

Cancer is a deadly disease that has always been highly problematic, prevalent, and incurable to this day. A wide range of sources, from chemical to parasitic, can give rise to neoplasms and cause the formation of tumoural subpopulations, known as tumour heterogeneity. This makes cancer treatment success harder and calls for submitting each patient to specialized and personal treatments. Correct patient diagnosis and adequate treatment strategies are far from flawless. The rise of bioinformatics in the 1960s was a fundamental step in better understanding cellular mechanisms and the effects of certain mutations on the progression of diseases such as cancer. In this study, a Python library named MutantGene was created and contains 13 functions. These serve mainly to link gene names from cancer mutation samples, stored in Mutation Annotation Format (MAF) files, with their respective products, which are characterized as Gene Ontology (GO) terms that are determined by extracting information from Gene Ontology Annotation (GOA) files and Open Biological and Biomedical Ontologies (OBO) files. The end result is a Python dictionary of processed gene names as keys and their retrieved GO terms as values, useful to better understand the purpose of each gene and possible consequences of mutations suffered, especially within tumours. The functions were created to be versatile and offer the possibility of filtering information, saving results in a file, and even converting information from a file to be used in Python. A workflow example is given to better understand the use of each function and how they interact with one another. This library is a tool to help researchers select specific groups of genes relative to cancer progression, which is especially useful in heterogenous tumours where different genes are affected by different mutations. In the future, MutantGene can be further developed to characterize the mutations themselves.

Resumo

O cancro é uma doença mortal que tem sido desde sempre altamente problemática, prevalente e incurável. Várias fontes, de química a parasítica, podem dar origem a neoplasias e causarem a formação de subpopulações tumorais, designado por heterogeneidade tumoral. Isto dificulta o tratamento bem-sucedido que requer tratamentos especializados e pessoais a cada paciente. O correto diagnóstico de cada paciente, bem como as estratégias adequadas de tratamento, estão longe da perfeição. A ascensão da bioinformática nos anos 60 foi um passo fundamental na compreensão dos mecanismos celulares e os efeitos de certas mutações na progressão de doenças como o cancro. Neste estudo, uma biblioteca de Python chamada MutantGene foi criada e contém 13 funções. Estes servem principalmente para cruzar genes de amostras de mutações cancerígenas, guardadas em ficheiros "Mutation Annotation Format" (MAF), com os respetivos produtos, caracterizados como termos "Gene Ontology" (GO) que são determinados através da extração de informação contida nos ficheiros "Gene Ontology Annotation" (GOA) e "Open Biological and Biomedical Ontologies" (OBO). O resultado final é um dicionário de Python contento os nomes dos genes processados como chaves e os seus termos GO recolhidos como valores, útil para compreender melhor o propósito de cada gene e possíveis consequências das mutações sofridas, especialmente nos tumores. As funções foram criadas para serem versáteis e oferecerem a possibilidade de filtrar informação, guardar resultados num ficheiro, e converter a informação de um ficheiro para ser utilizada em Python. Um exemplo de um procedimento é dado para compreender melhor a utilidade de cada função e como interagem um com o outro. Esta biblioteca é uma ferramenta para ajudar investigadores a selecionarem grupos de genes específicos relativos à progressão cancerígena, que é especialmente útil em tumores heterogéneos onde diferentes genes são afetados por diferentes mutações. No futuro, MutantGene poderá ser desenvolvida para caracterizar as próprias mutações.

Index:

Introduction

Discovery of cancer

The Edwin Smith Surgical Papyrus is a 17th century B.C. treatise, believed to be a copy of the earliest known text on trauma from 3,000-2,500 B.C., about injury and containing medical observations and treatments[1]. 48 trauma cases are exposed, of which 8, that have been described as untreatable, have only more recently been identified as breast tumours[1,2]. This is the oldest discovered description of what is now known as "cancer", with physical evidence such as fossilized human bone tumours dating as far back as about 1.7 million years ago[1,3]. However, the term "cancer" (that originates from the Latin word for "crab") was first used as "carcinoma" by Hippocrates due to the spreading pattern of the disease that formed a crab-like shape[2]. Since the quality improvement of microscopes during the 19th century, tumour cells have started to become better understood in terms of shape and behaviour, and, in the 20th century, the study of cancer known as oncology was concretely established as a scientific area of research[4].

What is cancer?

Cancer cells start off as regular and healthy cells that typically become tumorous when aberrations occur that damage their DNA by altering certain genes[5-7]. This can end up provoking behaviour abnormalities such as uncontrolled cell growth and division, evasion of growth suppression and death signals, destruction of surrounding tissue, and even the development of other tumours[5-7]. These characteristics are known as the hallmarks of cancer, as they define neoplastic behaviour as cancer develops and spreads[6]. In many cases, some tumour cells can break off from the primary tumour cell aggregate, invade blood vessels or lymph nodes, travel around the body, and either die along the way or become embedded in surrounding tissue[8]. Whilst there, they are free to multiply, forming metastatic cancer cell populations in different locations around the body that are of the same nature as the primary tumour. Metastatic cancer is the primary

reason for cancer mortality (the cause of more than 90% of cancer patients' death) due to how difficult it is to treat[9].

The main genes that originate tumours are genes that control the growth and survival of the cell, and are known as proto-oncogenes (becoming oncogenes when they suffer mutations)[10,11]. These genes produce proteins that often inhibit cell death and promote proliferation, therefore, when they become oncogenes, it is common to observe an uncontrolled increase in these proteins (or decrease in the case of tumour-suppressor gene products that repress cell growth and correct DNA mistakes[12]), causing uncontrolled cell proliferation and no cell death[10]. The proof of the existence of these genes was discovered in 1970 with the discovery of the *src* gene in a chicken virus named Rous Sarcoma Virus (more information on this in the next paragraph)[13,14]. About a decade passed until the discovery and isolation of the first human oncogenes in the early 1980s from bladder cancer cells, named *ras*, whose functions involve signal transduction essential for cellular proliferation[15-17]. This pioneered the discovery of others such as family of oncogenes named *MYC* (also first discovered in a viral genome) which regulate the expression of a wide variety of different genes associated to different cellular mechanisms (**Figure 2**) and were observed to be deregulated in more than half of the recorded cancer patients[18,19]. This serves as a perfect representation of the complex nature of cellular signalling and regulation, and how cancer can easily disrupt homeostasis by altering a single component.

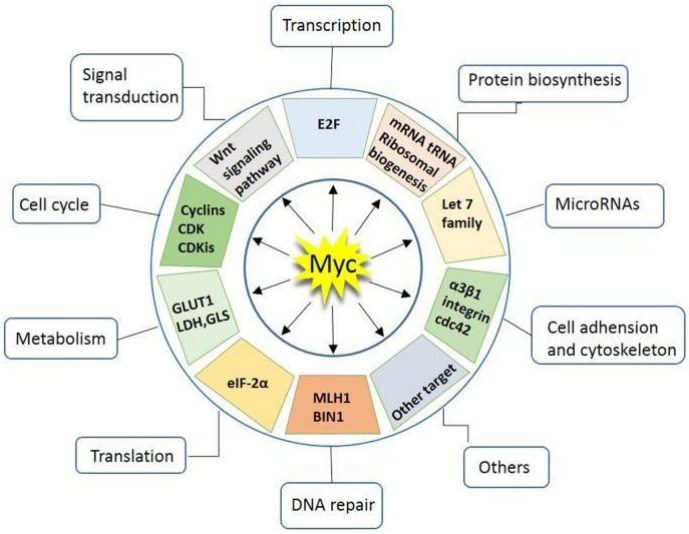

Figure 1: Representation of Myc proteins and their role in regulating a variety of different genes involved in distinct cellular processes[19].

Infectious origins of cancer

In 1908, 2 researchers (Vilhelm Ellermann and Oluf Bang) discovered that a type of leukemia, which is the designation given to the formation of neoplasms in hematopoietic stem cells stored in bone marrow, found in animals called leukosis can be transmitted between poultry via retroviruses known as Avian Leukosis Virus (ALV)[20-22]. This was the first evidence of cancer occurring via virus, however, leukemia was not considered cancer at the time[20]. 3 years later, in 1911, Peyton Rous discovered that solid tumours are also transmissible via virus[10,23]. A year prior, he had found that closely related chickens could contract tumours from one another through the transmission of tumour extracts, discovering a year later that the tumours were transmissible even after filtering the extracts to exclude animal and bacterial cells, now known to be caused by a retrovirus named Rous Sarcoma Virus (RSV)[10,23,24]. It was later discovered by Peter Vogt and Steven Martin that the RSV contains a gene termed *v-src* that, although does not contribute to retrovirus reproduction, allows the transformation of the avian cells it infects into neoplasms[11,14,25]. In terms of human cancer, 7 viruses have been associated to human neoplasm formation, with the first being reported in 1964, having been named Epstein-Barr virus[26-28] .

It is relevant to note that different viruses cause different types of cancer, and this is also observed within the same virus[29]. An example of this is the sexually transmitted Human Papillomavirus (HPV), which induces the production of E6 and E7 (2 oncoproteins that cooperate to induce tumour formation[30]) and can integrate its DNA in the host genome, leading to cancers such as cervical, anal or penile[29]. Another example is Kaposi's sarcoma-associated herpesvirus (KSHV), primarily transmitted via saliva[31]. Besides producing proteins that mimic human cytokines in order to overcome tumour suppression mechanisms and other proteins, KSHV produces the latency-associated nuclear antigen that inhibits lytic replication of the viral DNA and interacts with a number of cellular proteins to promote tumour formation, namely Kaposi's sarcoma (neoplasm formation adjacent to blood and lymph vessels[32]) and primary effusion

lymphoma (a rare tumour that originates from B-cells[33])[29]. Usually, cancer occurs due to chronic infections and other issues that can arise from viral infections and not directly due to the virus itself[34]. However, viral infection is necessary from both Kaposi's sarcoma and cervical cancer[34].

It has been estimated that 12% of cancers worldwide originate from viruses, the majority occurring in developing countries where the percentage of virus-related cancers can reach 50%[26,29,34]. Since 20% of all cancers originate from infection, that leaves 8% of cancers that occur due to bacteria and other infectious agents, such as *Aspergillus* moulds that produce aflatoxins (carcinogenic substances that can lead to liver cancer) and parasites that promote malignancy through chronic inflammation, production of oxidative stress inducing metabolites and direct tissue damage[35-38]. An example of a studied cancer-inducing bacteria is *Helicobacter pylori* (the first bacteria confirmed to be a cause of cancer), which contains modified lipid A molecules on its cell wall and modified flagellin molecules that avoid detection from Toll-like receptor 4 and 5 (responsible for initiating an immune response against pathogenic infections), respectively, thus avoiding immune responses[35,39]. Moreover, among other issues, *H. pylori* can induce the secretion of gastrin by gastric epithelial cells through signal transduction, activated by their association to CagL adhesins present on the surface of *H. pylori*[35]. The increased amounts of gastrin in the organism can lead to gastric adenocarcinoma (cancer originating in gland cells)[35].

Stages of cancer

Since the appearance of tumorous cells in the body, the cancer can go through different stages as it grows and, in some cases, spreads throughout the body[29,40]. The diagnosis of the stage by doctors is important to help predict the chances of survival of the patient and define the most adequate treatment for them[40].

As stated in “Cancer Staging” by the National Cancer Institute, the main method used by medical centres for cancer staging, despite the vast number of systems, is the TNM staging system[40]. The T stands for “tumour” (measuring the size and extent of the local spread of the primary tumour), the N stands for “nodes” (evaluating the amount of local lymph nodes that contracted tumorous cells from the primary tumour), and the M stands

for “metastasis” (indicating whether or not the tumour has metastasized to other organs)[40]. T can get attributed a number from 0 (meaning no primary tumour can be found) to 4, N can get a number from 0 (meaning the tumour hasn’t spread to nearby lymph nodes) to 3, and M can get either 0 or 1 depending on if the tumour has not or has metastasized, respectively[40-42]. Not all cancers are classified the same way, with some gaining additional attributes to classify the tumour, and the numbers attributed to the TNM system can also sometimes have different meanings[42]. For example, the T could classify the size of the primary tumour for some cancers, or if it has spread to nearby tissue for others, as described in “Cancer Staging” by the American Cancer Society[42].

As mentioned earlier, the TNM system is the one most widely used to grade certain tumours, however they can then be grouped into 5 stages for a more generalized description of the tumour[40,41]. Stage 0 is attributed to the appearance of abnormal cells located exclusively *in situ* (and is not considered yet a tumour), stage I is when the tumour has formed but is still localized, stages II and III refer to when the tumour has grown in size, number and has spread locally, and stage 4 is attributed to metastasized tumours[40,41]. The TNM staging system can be directly converted into these 5 stages[41].

What is heterogeneity?

A scientist called Johannes Muller noticed differences between tumour cells and normal cells by applying microscopy to neoplasm observations in 1833[43]. He, along with his student Rudolf Virchow, who linked the origin of tumour cells to normal ones, were considered one of the first scientists to note the heterogeneous nature of tumours[43]. Tumoural heterogeneity can be described as the phenotypic differences in tumour cells, being defined as intratumoural when these differences are observed within the same tumour[5,44]. This originates from genetic and non-genetic sources, the former referencing a wide array of genomic instability tumour cells suffer, from endogenous (like DNA replication/repair errors and oxidative damage) and/or exogenous origin (such as UV radiation or the various cancerogenic components contained in tobacco), and causing tumour treatment to be a more laborious task[5,44,45]. As for non-genetic sources, an

example is the presence of tumour stem cells[44]. Investigations have concluded the presence of tumoural hierarchy, where tumour stem cells that have the power of self-renewal and differentiation are present and give rise to new cell subpopulations with lesser proliferative power, thus contributing to heterogeneity[43,46,47]. These tumour stem cells are often hard to eliminate due to their resistant and adaptive nature, and are part of residual tumour cells after treatment, allowing cancer recurrence and offering investigators a great candidate for targeted cancer therapy[43,46,47]. Whether genetic or not, tumour heterogeneity can be gained over time, known as temporal heterogeneity where different subclones arise with the division of parental cells, or even in different locations in the body, known as spatial heterogeneity that happens due to the unique microenvironments neoplasms grow under within their site of growth (**Figure 2**)[5].

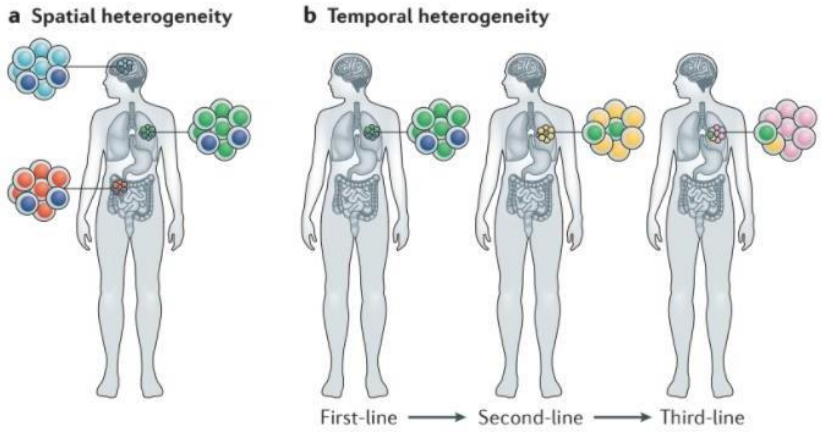

Figure 2: A representation of clonal evolution in tumours through location (spatial heterogeneity – **A**) and time (temporal heterogeneity – **B**)[5].

Tumour heterogeneity can be identified via the collection of neoplasm samples, which should be performed frequently and in different locations to consider temporal and spatial heterogeneity, respectively[5]. This process can be invasive, and so liquid biopsies to extract neoplasm derivatives circulating in the blood, although requiring major improvements, are a promising alternative with the potential for prognosis and even discovering certain mutations not detected with tissue biopsies[5,48,49]. Tumour sampling calls for sequencing techniques, which have matured from the initial Sanger method to the now more effective and cost-efficient second (requiring prior amplification of

sequencing libraries) and third (not requiring amplification) generation sequencing methods[50,51]. This is fundamental for the identification of different clonal populations within the tumour, including (with the help of single-cell sequencing as opposed to sequencing groups of cells from lesion samples) clones that harbour rare mutations, useful for heterogeneity quantification and revealing causes for treatment resistance[52,53]. These techniques, which are constantly evolving to achieve maximum efficiency, are crucial for unveiling the complex nature of tumours and accurately determining the best treatment strategies[44].

As mentioned before, clonal variations within a certain tumour can differ from patient to patient. In many cases these cells can harbour cancer driver genes, which are certain gene variants caused by mutations that confer a growth advantage to the suffered cells (along with other mutations present among tumour cell populations that do not contribute to cancer progression, known as passenger mutations)[54]. These driver mutations determine which clones prevail within the heterogeneous population and are therefore excellent targets for gene therapy by indicating possible treatment approaches[5,54]. An example of a cancer driver gene is the Epidermal Growth Factor Receptor gene which, after suffering specific mutations in its tyrosine kinase domain, suffers increased stimulation to induce cellular growth, survival and differentiation through signalling cascades in cancers such as lung, breast and brain[55,56]. Other examples of cancer drivers include the upregulation of the APOBEC3B cytidine deaminase (which converts cytosine to uracil during RNA editing) in breast cancer, which causes at times drug resistance mutations (proving to be a good target for chemotherapy[57,58]), and the introduction of mutations in small repetitive DNA sequences known as microsatellites due to faulty mismatch repair systems (seen in various cancer types such as colorectal cancer)[59,60].

The problem with eliminating specific cells that contain the targeted mutation is the induction of a “bottleneck” effect, which paves way for the growth and prevalence of different clones within the heterogeneous population, through temporal heterogeneity (as explained earlier), after a drastic reduction in size number[5,61]. This can also occur naturally, through processes of natural selection, where clones with advantageous mutations within their microenvironment survive and prevail, whereas the others die off[5]. Individual germline variations and other factors such as environmental conditions

can lead to the tumoural differences between different patients with the same cancer type, meaning the requirement of specialized and personal treatment towards each patient[5].

Treatment against cancer and heterogeneity

Back to the Edwin Smith Surgical Papyrus, under the 8 diseases deemed incurable (cancer), treatments were still mentioned. The tumours were said to be removed via cauterization using a fire-stick/fire-drill (an ancient tool used to start fires)[2,62]. Nowadays, of course, treatment methods and tools have grown and evolved, although tumour removal via cauterization (now known as electrocauterization) is still used in sensitive areas such as the brain, where electricity is used instead of fire[63]. This and other types of surgery are frequently used to detect, cure, and even prevent neoplasm formation[64]. However, since the 19th century, surgery is not the only treatment available, with options including immunotherapy, chemotherapy and, more recently, CRISPR-Cas gene editing.

The microsatellite instabilities mentioned earlier have been treated in the past with the blockade of the PD-1, a checkpoint protein part of the CD28 superfamily that down-regulates the immune response in order to prevent autoimmunity, thus allowing the immune system to kill cancer cells[65,66]. This is known as immunotherapy, which aims to boost the immune system to counteract tumour formations[67]. There are many forms of immunotherapy: the example given consists of immune checkpoint blockade (allowing the immune system to ignore signals to moderate its action) but others exist, including T cell transfer therapy (where t cells responsive to the patient's cancer are selected, grown, and reintroduced into the body) and monoclonal antibodies (artificially produced antibodies that bind to a specific tumour and mark it for an immune attack)[68]. Immunotherapy is used to treat many types of cancer, such as Breast and Prostate cancer, although it is not as frequently used for treatment as surgery and chemotherapy, according to the National Cancer Institute[67,68].

During the search for more potent chemicals for warfare in World War II, a mustard gas derivative called nitrate mustard was found to reduce an individual's number of white

blood cells, leading to a fascinating discovery[69,70]. This compound, like other mustard agents, adds alkyl groups to the DNA of rapidly dividing cells, such as cancer cells, which impedes their replication and, thus, kills them (because cancer cells divide quicker, they don't have time to repair the damages caused)[69-71]. Alfred Gilman and Louis Goodman were the first scientists to use this nitrate mustard to treat a patient with lymphoma, giving rise to a new form of cancer treatment known as chemotherapy[70]. It uses drugs to treat various types of cancer (such as leukaemia, breast cancer, colon cancer, etc.) and is advantageous in comparison to surgery due to the fact that it can treat metastatic tumours[69,72]. Treatment drugs differ depending on factors such as the type of cancer, its stage, the patient's health, among others[72]. Another example of a drug used (among hundreds of others) is methotrexate, which cured the very first metastatic cancer called choriocarcinoma (a rare trophoblast or germ cell cancer) in 1956[69,73]. Chemotherapy can, however, enhance and even cause cancer evolution and intratumoural heterogeneity, promoting, for example, the upregulation of APOBEC3B (mentioned earlier)[5]. This can cause cancers, such as is the case with Urothelial carcinoma, to become treatment-resistant, and may also offer other side effects such as hair loss (due to damaged hair follicles)[74,75]. To mitigate heterogeneity and minimize side effects, it is common for doctors to administer a combination of different drugs (called combination chemotherapy), which has proved very successful in curing many tumours, making chemotherapy a potent cancer treatment[69].

An article released in 1987 by a scientist named Yoshizumi Ishino exposed a strange repeated sequence that appeared downstream of the termination sequence of a gene responsible for isozyme conversion of the alkaline phosphatase in *E. coli*[76,77]. These repetitions, now known as Clustered Regularly Interspaced Short Palindromic Repeats (CRISPR), were soon found in several Bacteria and Archaea species[77,78]. During the 2000s, their function, along with their relationship with Cas (CRISPR associated) proteins, were revealed[77,78]. The CRISPR-Cas system functions as an immune mechanism against viral infections in the Bacteria and Archaea domains[77,78]. Cas 1 and 2 proteins are responsible for excising and integrating an area of an invading virus into the CRISPR locus, which then serves to "memorise" the attacker, and, by transcribing an RNA molecule of that same integrated spacer and associating it to another Cas protein, the bacteria is able to recognize the invading DNA through hybridization and

cut it, disabling the virus[79,80]. This system has proven useful for gene editing, since its discovery in 2012, when associated to synthetic RNAs that recognize specific gene segments to be used as guide RNAs[77,81,82]. These, when coupled with Cas9 proteins (or similar), bind to and cut the target DNA in living cells, granting the opportunity to alter the DNA sequence through insertions/deletions, thanks to the built-in DNA repair mechanisms Non-Homologous End Joining (NHEJ) and Homology Directed Repair (HDR), silencing or altering the product of the affected gene[82]. Due to its precise, effective, and relatively simple nature, it is not difficult to imagine the use of CRISPR techniques to treat human diseases, including cancer. In 2019, the first CRISPR treatment against cancer, which involved the genetic modification of T cells to better attach themselves to a cancer antigen called NY-ESO-1 and kill the associated cells, was administered to 3 cancer patients and, although it wasn't very effective, it managed to strike a solid tumour in a patient with sarcoma, which is difficult to achieve with cellular therapies[83]. From cancer genotyping to cancer treatment, CRISPR is a state-of-the-art tool that shows promise in the struggle against cancer, although more research will need to be conducted to improve its effectiveness and mitigate problems such as off-target DNA cutting and precise (and safe) introduction of CRISPR components *in vivo*[83].

Overall, a wide array of cancer treatments is available and rapidly evolving. Cancer rates remain high, however, and was considered the second leading cause of death worldwide in 2018[84]. A major issue that is being faced is the recurrence of heterogeneity which occurs in pretty much all circumstances in cancer patients, regardless of the administered treatment[5]. Currently, the best approach is to alternate both between and within treatment types[5]. In order to continue evolving and reach the ideal solution against cancer, through refined data analysis and heterogeneity measurements, aid must be provided by advanced computation models via the use of bioinformatics[5,85,86].

Origin of bioinformatics

From the conceptualization of first mechanical computer in 1822 by the English Mathematician Charles Babbage, named Difference Engine, that was to automatically

perform multiple calculations and print them in table form, to the currently reigning electronic supercomputer named Fugaku, with a memory space of 486.6 terabytes (explained at https://www.top500.org/system/179807/), computers have been continuously evolving over the years[87]. With the increasing capabilities computers have gained (such as solving complex mathematical problems and programming artificial intelligence able to evolve with real-time experience), it's not difficult to imagine the utility computers bring to other domains outside mathematics, which also includes biology.

The marriage between computation and biology is known as bioinformatics, and it officially originated in the 1960s with the usage of computers to analyse protein sequences[88]. Bioinformatics is a broad area of study that combines computation and biostatistics with the various fields of biology, bringing together both formal and natural sciences[89,90]. It is an essential tool used when dealing with large amounts of data and assists in the conclusion and interpretation of scientific experiments[89]. The term "bioinformatics" was first publicly stated by Ben Hesper and Paulien Hogewen in an article in Dutch in 1970, defining it as "the study of informatic processes in biotic systems"[91]. These two scientists took into consideration the vast amount and importance of biological information, and the need to study it as an individual scientific field[91]. Over the years, with advances in computer technology and power, and the increasing amount of data from genome projects and "omics" studies, this definition began to transfer towards the use of computational methods to acquire and compare information from said data[91,92].

The uses of bioinformatics are many and range from gene therapy (the alteration of specific genes' expression to treat or even cure diseases such as cancer) to drug development (helping identify target protein classes and predicting protein interactions to help make an educated guess towards selection of the right compounds for pharmaceutical production, as opposed to using the trial-and-error method)[93]. For these purposes, tools are required that assist in a wide range of biological studies and industries. An example is BLAST (Basic Local Alignment Search Tool: https://blast.ncbi.nlm.nih.gov/Blast.cgi), which accepts a query DNA or amino acid sequence and aligns it with similar sequences from a wide range of species stored in various databases[90]. The aligned sequences allow for sequence comparisons, where

differences can easily be located and inferences (such as mutation impacts on certain genes/gene products or phylogenetic relations between species) can be made[90].

Python

Ada Lovelace is credited for being the first person to write a computer program, after she explained in an article (in 1843) a set of instructions for the Analytical Engine (Babbage's second and more powerful computer concept, which he did not finish manufacturing) to print out the Bernoulli numbers[87,94]. Since then, various computing languages have been created. It started with Assembly, where the code is written in a way that is directly translated to code understood by the CPU (Central Processing Unit), which preceded the arrival of many others, including the language released in 1991 by Guido Van Rossum called Python[95,96].

Python is currently the most popular programming language worldwide (based on the number of Google searches on Python tutorials, registered at http://pypl.github.io/PYPL.html), including within the field of bioinformatics, due to its simple syntax, which makes it relatively easy to both write and read its code, and also its vast array of third-party tools that can be used for more specific tasks, such as Biopython which offers tools to work with biological computation[97]. Although there are other languages (such as R) that are great for statistical analysis, python was used for this study due to its highly multi-purpose nature and the familiarity of the student with this language.

Using gene characterization to evaluate cancer mutation impacts

Cancer mutations can affect different genes in different ways, as previously discussed. By altering genetic expression, different outcomes may ensue depending on the severity of the mutation or the actual gene affected[56,58]. In this study, a Python library named MutantGene containing 13 functions was created with the purpose of bridging the gap between information on gene variants stored in Mutation Annotation Format (MAF) files and gene product characterization through Gene Ontology (GO) terms, which define the general functionalities and locations of action of each gene product

(explained later). Using Gene Ontology Annotation (GOA) files as an intermediate, it is possible to select specific groups of gene names and associate them to their corresponding GO terms, allowing each user to develop a unique workflow and filter information using very few lines of code. From there, it is possible to assess the implications of each mutation and create assumptions on its severity and role in cancer progression. This comes in handy especially when dealing with heterogenous tumours due to the different combinations of mutated genes, offering organized documentation of each gene product's varied functions which can help guide doctors and researchers towards the ideal targets for patient-specific therapy.

State of the art

Genomic Data Commons

The National Cancer Institute (NCI) is the lead agency regarding cancer research support (including towards cancer diagnosis, clinical trials, laboratory constructions, etc.), and appropriate training programs (to spread interest in cancer prevention and train upcoming cancer researchers), monetarily assisting both fields[98]. In order to organize and manage the large amounts of data generated from different research groups and institutions such as The Cancer Genome Atlas (TCGA), which in turn focuses on characterizing cancer samples from various different cancer types[99], the Centre for Cancer Genomics (CCG) was created by the NCI[100]. The information gathered by the CCG is stored in the Genomic Data Commons (GDC), a unified cancer data repository that supports data sharing among researchers and bioinformaticians[101].

The GDC obtains tumour-normal DNA alignments from different research groups to evaluate the mutations suffered and stores the information in Variant Call Format (VCF) files[101,102]. The information from these files can be further processed through pipelines, which are a series of bioinformatic transformations in a file to process and select specific genetic data[103], in order to remove less relevant and/or confidential information such as germline variants, so as to occult each patient's genotype[104]. This

processed information is then stored in MAF files, as mentioned earlier, which can be accessed by anyone as long as the file in question is somatic, that is, it does not contain genetic information that could potentially identify a patient[101,104] .

Each MAF file contains unique tumour-normal alignment pairs from each sample which is processed so that only certain variants that fall under certain conditions are saved in the file (such as being somatic, being the most affected transcript, etc.)[101,102,104]. Each variant in a MAF file is saved with a variety of different associated information, including the Human Genome Organization (HUGO) symbol, the Human Genome Variation Society (HGVS) nomenclature, among others.

HUGO Gene Nomenclature Committee

The HUGO Gene Nomenclature Committee (HGNC), under the support of the Human Genome Organization (HUGO), is where all official gene names and symbols (abbreviations of the gene names) are decided and approved, in order to guarantee the attribution of unique nomenclature for each gene for unambiguous scientific communication[105]. These attributed names are stored in the HGNC database and each can be used to retrieve specific information in a variety of other databases[105].

Human Genome Variation Society

The Human Genome Variation Society (HGVS) nomenclature provides an official designation for each DNA, RNA or protein variant and also works in association with HUGO[106]. Each variant description is presented in the following manner: (**Reference):(Description)**[106]. The reference contains an accession number followed by a version number[106]. The description starts with the type of reference sequence used (c., g., p., or r. when referring to a coding DNA reference, genomic reference sequence, protein level consequence or RNA level consequence, respectively), followed by the nucleotide/protein coordinates and finally de mutation that occurred[106]. For example:

NM_004006.2:c.4375C>T

which points towards a mutation in the 4375th nucleotide of a coding sequence where a cytosine was converted to a thymine. The "NM" in the example informs the reader that the mutation occurred in a protein coding sequence[106]. Besides this example, there are other (sometimes more specific and elaborate) representations of certain mutations[106,107].

Gene Ontology Consortium

The GO Consortium consists of a unification of genome databases and researchers that specialize in attributing universal terminology to different eukaryotic gene products, that is, terminology that defines each gene product in an interspecies manner[108,109]. New information regarding gene function is analysed by the Consortium which proceeds to associate new GO terms (which classify gene function classes) to that gene, along with the term's unique GO identifier (GO ID)[108,109]. These GO terms are found in the GO resource (or knowledgebase) and are a crucial part of the ontologies, which consist of formal descriptions of each gene's functions and relations between them[108,109]. Each GO term is placed into one of 3 categories: **Biological Process** (the global biological function a certain gene product is involved in), **Molecular Function** (the specific, molecular level activity of a gene product) or **Cellular Component** (the location where a gene product's action within a cell takes place)[108,109]. They are then linked to other GO annotations according to the relationship between the terms through a phylogenetic organization, connecting parent terms (more generalized) to child terms (more specific) and offering a clearer representation of their relations[108,109].

Ontologies can be saved into files in 3 formats: JSON, OWL and OBO[108,109]. This last format is human-readable and created using the Open Biological and Biomedical Ontologies (OBO)-Edit software[110]. It stores a representation of a subset of (or general) ontologies, organized by an initial "header" block and followed by "stanzas" (which are blocks of text that describe a certain object or concept through consecutive lines of tag-value pairs)[110]. Three types of stanzas exist, one of which consists of "term" stanzas and were a focus in this report[110]. These stanzas describe the actual GO terms themselves and are made up of a line containing the stanza type ("[Term]"), a line with its unique

GO ID, and an undefined number of lines containing additional information, such as the term name and/or relations to other terms[110].

The association between genes and GO terms is known as an annotation[109]. These annotations are stored in the Gene Ontology Annotation (GOA) database that works in collaboration with the UniProt knowledgebase (UniProtKB - where a wide range of information regarding protein sequences are stored and maintained) by attributing annotations to the UniProtKB protein entries[111]. GOA files store several attributed GO IDs to each protein mentioned in the file through their UniProtKB accession number, which are new protein entries' identifiers attributed to automatic (stored in the TrEMBL database) or manual (stored in the Swiss-Prot database) entries[112,113]. These files' parameters consists, at minimum, of a gene product, a GO ID, a reference citation to support the annotation and an evidence code[108,109]. There can be many different GO IDs for the same protein if it has different attributes, such as what it does, where it's located in the cell, etc[108,109]. Furthermore, individual GO terms and annotations from the GOA database can be accessed using the QuickGO browser[114,115].

Implementation

Retrieving GO terms for selected genes that suffered mutations

Using Python, 13 functions were created in order to collect gene names related to selected mutation information and fetch their corresponding GO terms, and are the makeup of the MutantGene library. These functions were separated into 3 main steps to better explain in this report the main workflow from start to finish. The first step (2 functions) deals with selecting variants stored in a specified MAF file, the second (3 functions) with fetching the GO IDs attributed to the product of each corresponding gene from a chosen GOA file, and the third (2 functions) with converting them to their GO terms using an OBO file. 6 auxiliary functions also included serve to filter out information in any given step of the process and interconvert them between saved text files and lists/dictionaries to be used in Python.

The first step was to download a MAF file from the GDC Data Portal and analyse its structure and attributes. Due to the existence of 120 attribute types, the indexation of each one can be determined using a short auxiliary Python function (named *maf_index*) that returns a dictionary of all attribute titles as keys and the corresponding attributes of a random entry as values. From there, the desired attributes regarding information on the suffered mutations were selected to be extracted by default, although the parameters retrieved can be changed by the user.

After that, the focus shifted exclusively to the genes themselves. In order to establish the impact a certain mutation has on the cell/microenvironment, there needs to be an understanding of the function of the affected gene's product. For this purpose, a GOA file should be retrieved from the GOA database and used. GO IDs concerning proteins (therefore excluding non-translating RNA) were obtainable here as an example, but the corresponding terms had to be obtained elsewhere. A general OBO format file containing all available GO terms is used by default for this purpose, which can be changed by the user. This way, an easy visualization of each gene product's functions is obtained, which can then be compared to the mutation suffered for an adequate impact prediction.

These functions were documented in the library file to explain the functions' objectives, the parameters they take in, and what they return. These will also now be explained in further detail, organized by the steps previously mentioned.

Step 1: Mutation samples

The first part of this workflow aims to retrieve mutation sample information from MAF files and process them. To achieve this, a function named ***mutation_samples*** was created and takes in as parameters a MAF file directory, a maximum number of variants to be processed (optional), a variant starting point index (optional), and a list of attribute indexes to select the information to be retrieved (optional). The initial annotation lines of the file (that start with "#") and the attribute titles line were considered and are excluded from the information extraction process. If no variant limit and/or starting index is given, the function will process all variants in the given MAF file. This

function offers a default attribute index list that will be used in case no list is specified by the user. This list is comprised of the HUGO symbol ([0]), HGVSc (coding DNA reference description) ([34]), HGVSp (protein level consequence description) ([35]), variant class (type of mutation) ([95]) and impact ([93]). This information is extracted from the given number of variants and saved into individual lists that are added to a global list (the end product of this function).

In case the user decides to select different attributes to be retrieved, an auxiliary function named ***maf_index*** was created to assist the user in locating the desired information, as previously mentioned. This function takes in a MAF file as the sole parameter and returns a dictionary containing the attribute titles as keys and their corresponding index as values. The attributes appear in the same order as their titles, meaning this function can be used to locate the former in each variant.

Step 2: GO IDs

This part deals with finding the GO IDs linked to the genes of the selected mutation samples. For this, gene names will have to be one of the attributes extracted from the variants. The ***get_genes*** function takes in as parameters a list of mutation sample information lists (resulting from the *mutation_samples* function) and the index where the gene names reside within these lists (optional). This function assumes the gene names are the first elements in the lists by giving a default index of "0", meaning the index should be specified if this is not the case. The return is a set of all unique genes from the selected variants.

Before attributing each gene their GO IDs, these must first be retrieved from a GOA file containing the genes relative to a certain species (in this case relative to humans). The function named ***all_goa_id*** does this, taking in as a parameter the directory of a GOA file. This function grabs every gene ID ([2]) present in the selected GOA file and stores them as keys in a dictionary with a set of each of their associated GO IDs ([4]) as values.

Having now isolated the selected gene names into a set and created a dictionary containing a wide range of GO IDs associated to different genes, these can be used to

get the GO IDs of the desired genes. The function that achieves this was named ***get_goa_id*** and takes in as parameters a set/list of gene names and a dictionary with gene names as keys and a set of their GO IDs as values. The selected HUGO gene symbols are searched for in the dictionary and, if correspondence is found, the selected gene names will be stored in a final dictionary (which will be returned by the function) as keys with their associated GO IDs as values.

Step 3: GO terms

The final steps require converting the GO IDs to their respective terms. Similar to what was achieved with *all_goa_id,* a function that collects all (or a wide range of) GO terms is needed. This is done with the ***get_ontologies*** function which takes in an OBO format file or URL (optional). This function separates each stanza, processes the ones titled "[Term]" and stores the contained GO IDs as keys and their associated GO term as values, present in indexes [1] and [2] within each stanza, respectively. If no OBO file/URL is specified, the terms will be retrieved from an OBO file containing all currently known and used terms through its URL.

Finally, the GO IDs gathered relative to the collected genes in the previous steps can now be converted to their corresponding terms by searching through the generalized dictionary created with the *get_ontologies* function. The function that pulls this off was named ***id_to_term*** and takes in as parameters a dictionary containing the desired genes as keys and their GO IDs as values, and a dictionary with GO IDs as keys and their GO term as values. The list of GO IDs for each gene is iterated and evaluated for correspondence in the general GO term dictionary. If found, the GO ID is converted to its term and a dictionary containing the gene names as keys and their associated GO terms as values is returned.

Auxiliary functions

It may be the user's desire to further filter the information retrieved. Knowing this, 2 functions were created to allow this at almost any given step in the workflow: one

named ***filter_list*** can be used to filter lists and takes in as parameters a list to be filtered, a filter list containing elements to consider for filtering, and a character to give the user the option to keep elements of the list to be filtered that include ("y") or not (not "y") the elements of the filter list (optional); the other named ***filter_dict*** can be used to filter dictionaries and takes in as parameters a dictionary to be filtered, a filter list of elements to consider for filtering, a character that functions the same way as in the *filter_list* function ("y" or not "y") (optional), and also a character to decide if the dictionaries are to be filtered by comparing their keys ("k") or values (not "k") (optional). Both functions assume by default that the list/dictionary to be filtered must include at least 1 element from the filter list, and the *filter_dict* also assumes the user wants to filter out key-value pairs according to the keys. Moreover, these functions consider the possibility that they are filtering lists of lists (in the case of the *filter_list* function) or dictionaries with lists as values (in the case of the *filter_dict* function).

Another decision the user may want to make is to save the results from each function in a text file, or even retrieve information from text files to be used and/or processed. For the former, 2 functions were created: one for lists, called ***list_to_file***, which stores each element in the list in a single line in the file, and the other for dictionaries, called ***dict_to_file***, which stores each key-value pair in the same line separated by ": ". Both take in as parameters the list/dictionary to save in a file, and a name for the file to be created. The reverse procedure (retrieving information from files to be used in Python) also has 2 associated functions: ***file_to_list*** which creates a list from each line of the file given; and ***file_to_dict*** which creates key-value pairs from each line that are stored in dictionaries. They both take in a file name for the information retrieval, and the *file_to_list* function also takes in a character that specifies if each line should be added to the final list as individual lists ("n") or should be extended (not "n") (optional – the function assumes the users want each line to be individual lists within the final list).

Function interactions and optimization

The *mutation_samples* function kicks off the process, gathering specified information from any given MAF file including the genes stored and the specific type, location, and impact of the corresponding mutations the genes suffered (by default). This is the start

of the workflow, where the subsequent functions serve to characterize the functions of the products of the collected genes. These functions can act independently, meaning that if the user already has information saved in a file, for example, they can convert it to be used by a certain function in the library. Dictionaries were mostly used for the purpose of being able to easily grab information specific to a desired gene, apart from the *get_genes* function which contains only gene names, and the *mutation_samples* function because some genes are repeated throughout the file due to having suffered different mutations (using a dictionary would redefine the values of a repeated key instead of adding a new key-value pair). The interactions between functions and the direction of information transfer are represented in **Figure 3**.

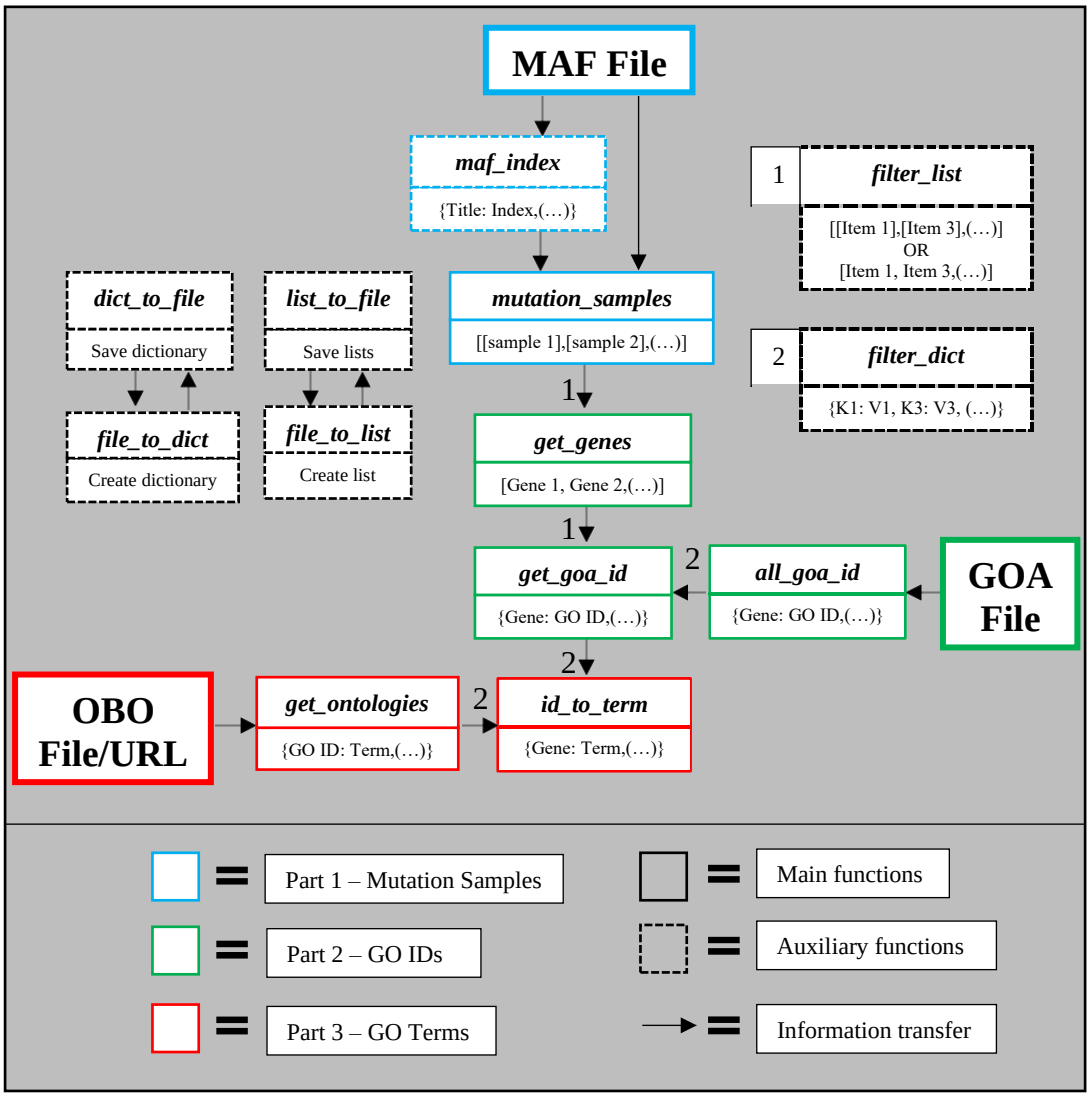

Figure 3: A schematic representation to the functions created, the source of the information they process and their interactions with one another.

Some functions were optimized to be able to consider different possibilities and applications. The "filter" functions are great examples of this optimization, in that they were created to be able to further process the information retrieved throughout the workflow (specified in **Figure 3**). The same can be said for the functions that save/extract information to/from files, respectively, as they can be used alongside practically all functions in the MutantGene library. *file_to_list* also considers that some files contain sets, converting them to lists in Python.

Function execution time and computer performance were factors considered whilst creating this library. Since *mutation_samples* collects repeated genes, there is no interest in searching for the same GO IDs in the *get_goa_id* function, which is why a set of the retrieved genes is created with the *get_genes* function, eliminating repetitions and saving time. The same goes for the GO IDs collected from the GOA file using *all_goa_id*, with some genes containing repeated GO IDs. This issue was also solved by using a set, which also offers cleaner results instead of some genes having many repeated GO IDs. Because MAF files can contain thousands of entries, it may not be optimal (or even of the user's interest) to retrieve information on every single one, hence the possibility of limiting the number of variants processed with the *mutation_samples* function. Adding a "break" to FOR loops in the *filter_list* and *filter_dict* functions, after determining the desired elements to keep/discard, was another time-aware addition that speeds up the functions' execution by skipping unnecessary line searching.

Workflow example:

To exemplify what each function does and how they interact with one another, the formerly described functions were used in a workflow example which was saved in a file named "examples.py". The main functions were used sequentially, and the auxiliary functions were introduced in between steps to better understand what each function

does (by saving the results of each function in a text file), utilize information saved in files, and filter certain results according to a desired outcome (with the filter functions). The file also contains commentary to explain the thought process of the workflow and why each function is used, which will now also be explained here. The code used in the workflow will also be shown throughout. Before anything, the library module must be imported (in this case it was also reassigned to its initials "mg" for practical reasons) and the personal MAF and GOA file directories were defined as maf and goa, respectively.

```
import MutantGene as mg

maf = 'C:\\Users\\Roberto Bullitta\\Desktop\\PROJETO BCM\\Estágio\\MAF\\MAF_file_1.txt'
goa = 'C:\\Users\\Roberto Bullitta\\Desktop\\PROJETO BCM\\Estágio\\goa_human.gaf'
```

The first step in this example will be to collect mutation sample information that is stored in a MAF file. The file chosen came from the TCGA-LAML genome project, which collects data from patients suffering with Acute Myeloid Leukemia (file retrieved at https://portal.gdc.cancer.gov/files/27f42413-6d8f-401f-9d07-d019def8939e)[101]. To understand where each attribute is stored in each line of the MAF file (in terms of index), the *maf_index* function was called to generate a dictionary with the title of each parameter as keys and their corresponding index as values, which was then saved in a file named "maf_index.txt" with the *dict_to_file* function (**Figure 4A**).

```
index = mg.maf_index(maf)
mg.dict_to_file(index, "maf_index.txt")
```

Knowing what each index stores, it was then possible to retrieve the desired information from the MAF file. For this, the *mutation_samples* function was called to retrieve the specified information (the list of indexes given was equal to the default indexes, but in a slightly different order) on 100 variants starting with the 40th in the file, saving the results in a file named "mutation_samples.txt" using the *list_to_file* function (**Figure 4B**).

```
raw_samples = mg.mutation_samples(maf, 100, 39, [93,0,34,35,95])
mg.list_to_file(raw_samples, "mutation_samples.txt")
```

The results indicated that some of the retrieved samples contained information on genes that suffered low impact mutations, meaning they were likely not heavily affected. This was a great opportunity to call the *filter_list* function to filter out all lists regarding low impact mutations by using "LOW" as the criteria for filtering. The results were saved in a file named "filtered_samples" using the *list_to_file* function (**Figure 4C**). This eliminated 23 lists, leaving a total of 77 in the global list.

```
samples = mg.filter_list(raw_samples, ["LOW"], "n")
mg.list_to_file(samples, "filtered_samples.txt")
```

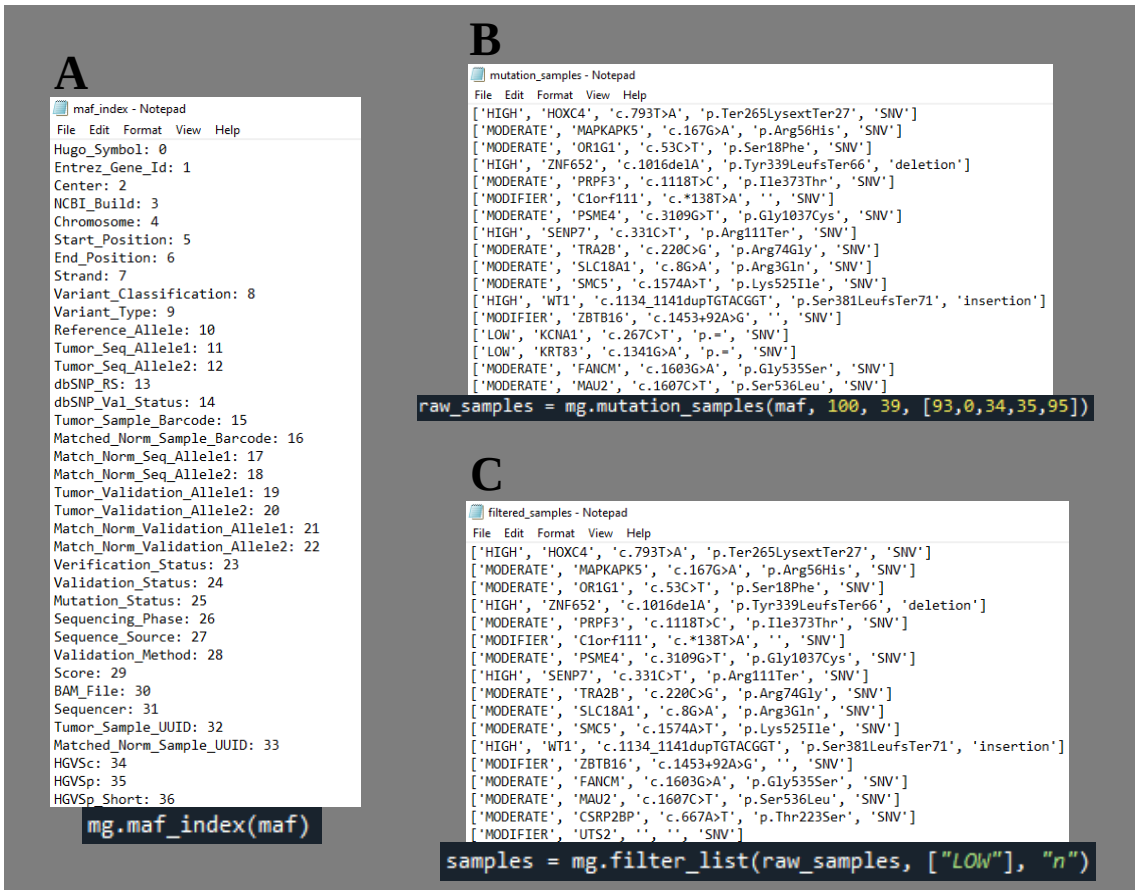

Figure 4: Extraction and manipulation of the information stored in a MAF file. (A) A dictionary created by calling the *maf*_index function stored in a file ("maf_index.txt") showing all 120 of the MAF file's attribute titles as keys and the index in which they appear in each variant as values (not fully represented). (B) The lists created by calling the *mutation_samples* function stored in a file ("mutation_samples.txt") containing the chosen information from 100 variants starting with the 40th variant (index [39]) (not fully represented). (**C**) The

resulting variants that were filtered from the results described in (B) by calling the *filter_list* function and saved in a file ("filtered_samples.txt") (not fully represented). This was used to filter out any lists containing "LOW" in them, meaning information on all retrieved samples containing genes that suffered a low impact mutation are removed from the global list, leaving 77 variants.

Now, the genes within each of the 77 variants can be characterized by first determining their GO IDs. To achieve this, the genes retrieved must be isolated and stored in a set (which disregards any repeated genes). The *get_genes* function was thus called, and the resulting list was saved in a file called "genes.txt" with the *list_to_file* function (**Figure 5A**). It is to note that 76 genes appeared in this list, meaning there was a gene that appeared twice in the samples processed. Moreover, because the gene names in the lists generated in this example appear in index [1], this index must be specified in the *get_genes* function.

```
genes = mg.get_genes(samples, 1)
mg.list_to_file(genes, "genes.txt")
```

This process also requires retrieving a generalized selection of genes and their associated GO IDs from a GOA file. For this example, a GOA file regarding human proteins was chosen (downloadable from ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/HUMAN/goa_human.gaf.gz, 06/10/2020). The *all_goa_id* function was called to gather all genes saved in the file and all the GO IDs attributed to each of them, storing them as key-value pairs in a dictionary, respectively. The results were saved in a file named "all_goa_id.txt" using the *dict_to_file* function (**Figure 5B**).

```
all_ids = mg.all_goa_id(goa)
mg.dict_to_file(all_ids, "all_goa_id.txt")
```

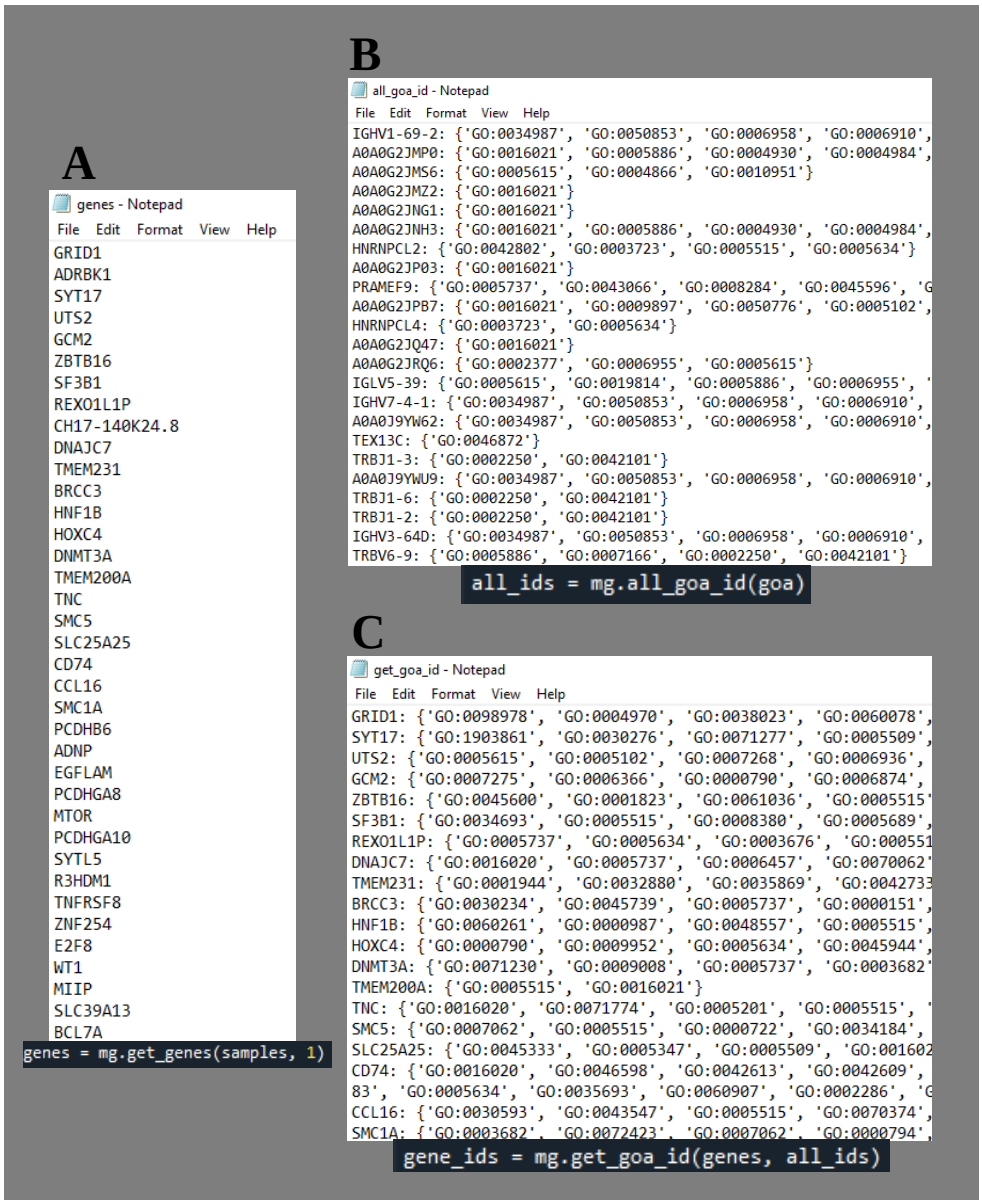

Figure 5: Retrieving the processed genes' GO IDs from a GOA file. (**A**) A set of the genes (76 in total) isolated from the processed variants using the *get_genes* function and saved in a file named "genes.txt" by calling the *list_to_file* function (not fully represented). (**B**) A dictionary containing all the saved genes in the chosen GOA file as keys and their GO IDs as values, created using the *all_goa_id* function, and saved in a file named "all_goa_ids.txt" by calling the *dict_to_file* function (not fully represented). (**C**) A dictionary created using the *get_goa_id* function that searches for each processed gene in the dictionary with all genes in the GOA file. The resulting dictionary with 72 gene-

IDs pairs (meaning 4 genes were not found in the GOA file chosen) is saved in a file named "get_goa_id.txt" by calling the *dict_to_file* function (not fully represented).

The *get_goa_id* function brings together the results of these last 2 functions by comparing the gene names processed from the samples to the gene names retrieved from the GOA file of choice. If the gene name is found within the dictionary, the gene is added to a final dictionary as keys and the corresponding GO IDs are assigned as their values. In this case, 72 genes had found correspondence, meaning 4 didn't and were not attributed GO IDs. The final dictionary was saved to a text file named "get_goa_id.txt" using the *dict_to_file* function (**Figure 5C**).

```
gene_ids = mg.get_goa_id(genes, all_ids)
mg.dict_to_file(gene_ids, "get_goa_id.txt")
```

A great opportunity arises at this point. Now that the processed genes have their GO IDs associated to them in a dictionary, we can further filter the results according to the gene product's functions. Child terms of the cytoplasm were found with the QuickGO browser (at https://www.ebi.ac.uk/QuickGO/term/GO:0005737), which assists in locating information on specific terms[115], and saved into a file named "cytoplasm_ids.txt" (**Figure 6A**). They were stored into a list using the function *file_to_list* and used to filter out genes that did not contain at least one of the GO IDs in the list (meaning their products don't have any known function or direct association regarding the cytoplasm) by using the *filter_dict* function. The result of this filtering process was a dictionary containing only 24 genes (which means 48 gene-IDs pairs were removed), which was saved to a file named "cyto_genes.txt" using the *dict_to_file* function (**Figure 6B**).

```
cyto_ids = mg.file_to_list("cytoplasm_ids.txt", "y")
cyto_genes = mg.filter_dict(gene_ids, cyto_ids, "v", "y")
mg.dict_to_file(cyto_genes, "cyto_genes.txt")
```

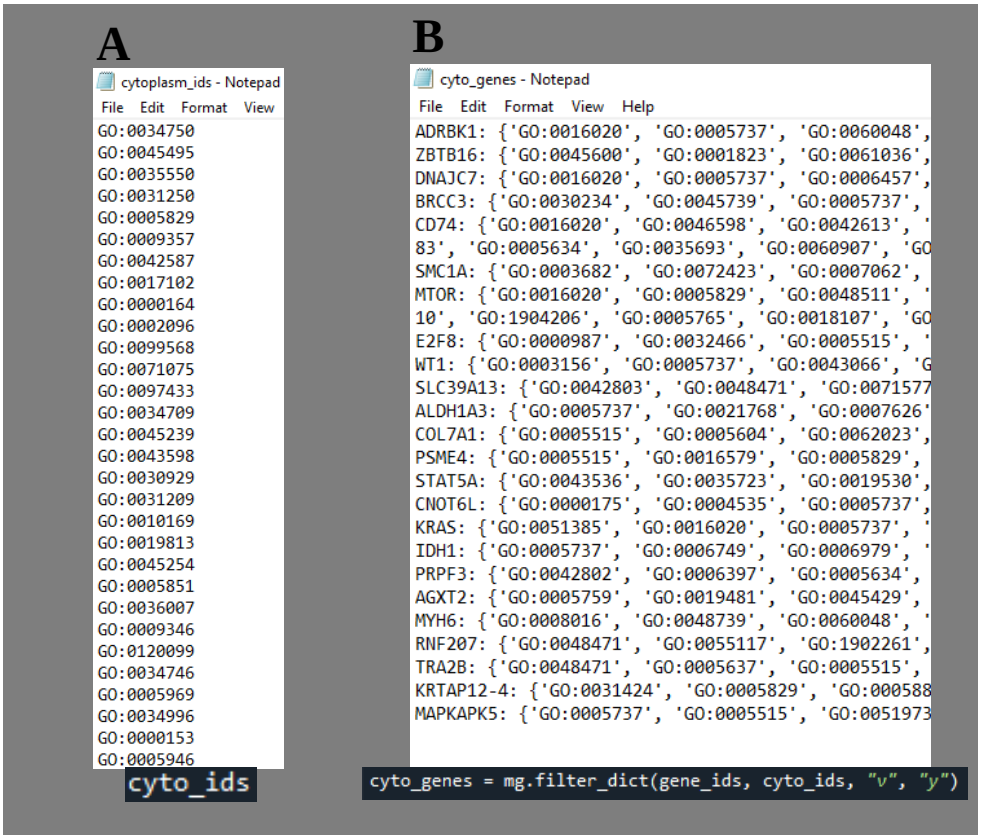

Figure 6: GO IDs of terms related to the cytoplasm and the use of these IDs to filter the processed genes. (**A**) A file created manually named "cytoplasm_ids.txt" and containing the GO IDs of all child terms of the cytoplasm (not fully represented). (**B**) The remaining 24 genes (in some way connected to the cytoplasm) that resulted from the filtration process using the *filter_dict* function. The resulting dictionary was saved in a file named "cyto_genes.txt" by calling the *dict_to_file* function (not fully represented).

The final steps of the workflow consist of converting the GO IDs retrieved to GO terms. Similar to the GO IDs, a wide array of GO terms should also be fetched so as to then be linked to the processed genes. This was achieved with the *get_ontologies* function which, in this example, was used with the default general OBO URL, so as to fetch all known and used GO terms. This function returned a dictionary with all GO IDs as keys and their associated GO terms as values, which was then saved into a file named "all_terms.txt" using the *dict_to_file* function (**Figure 7A**).

all_terms = mg.get_ontologies()
mg.dict_to_file(all_terms, "all_terms.txt")

This dictionary could then be used in the *id_to_term* function as a repository to obtain the GO terms synonymous to the retrieved GO IDs and to convert the latter to the former. The end product was a dictionary with the processed genes as keys and their linked GO terms as values, which was then saved to a file named "gene_terms.txt" using the *dict_to_file* function (**Figure 7B**).

gene_terms = mg.id_to_term(cyto_genes, all_terms)
mg.dict_to_file(gene_terms, "gene_terms.txt")

Figure 7: Converting the genes' GO IDs to terms. (**A**) A dictionary with all known and used GO IDs as keys and their respective terms as values, retrieved using the *get_ontologies* function and saved in a file named "all_terms.txt" by calling the *dict_to_file* function (not fully represented). (**B**) The resulting dictionary after converting each gene's GO IDs to terms using the *id_to_term* function and being saved in a file named "gene_terms.txt" by calling the *dict_to_file* function (not fully represented).

During the workflow, the *file_to_dict* function was the only one not used. However, the "examples.py" file contains an extra section where the "gene_terms.txt" file was converted back to a dictionary to exemplify its usage.

terms_dict = mg.file_to_dict("gene_terms.txt")

Discussion

Retrieving GO terms

Out of the 120 elements in each MAF file line (offering a great range of information that can be retrieved with the help of the *maf_index* function which locates the index of each 120 attributes), 5 are extracted by default with the *mutation_samples* function. The HUGO gene symbol is used to identify the related GO IDs and terms in the *get_goa_id* and *id_to_term* functions, respectively, and the other 4 elements are used to characterize the mutation that occurred in each extracted gene, including the type of mutation and how big is its predicted impact. The HGVS description explains what mutation took place in the coding sequence of the gene (HGVSc) and how it affected the gene product (HGVSp)[106]. In some cases, no HGVS description is given because, for example, the mutation occurred upstream/downstream of the gene and has no way to be described in reference to coding DNA[106].

By using the *all_goa_id* function, all the gene symbols, along with their respective GO IDs, are extracted from a selected GOA file and stored in a dictionary. This makes it easy to then use the desired gene symbols chosen from a MAF file (and gathered into a single list with the *get_genes* function) to find their associated GO IDs by using the *get_goa_id* function. Most genes have a variety of different GO IDs associated to them. Some of the genes were synonymous to others and were stored in index [10] of the GOA file where synonyms of a main gene (stored in index [4]) are listed and separated by the character "|". These gene names were taken into consideration and were attributed the same GO IDs as the main gene name. However, some retrieved genes may not be found depending on the GOA file chosen and thus no GO IDs are attributed to that gene.

Because some variants saved in the MAF file (although note different mutations) refer to the same genes, the *get_genes* function filters repeated genes by adding them to a set, which prevents the *get_goa*_id function from attributing the same GO IDs to the same genes, which would be more time consuming. Some GO IDs saved in the GOA file are also repeatedly attributed to the same genes, resolved by the *all_goa_id* function that also uses sets to eliminate GO ID repetitions for the same gene and prevents dictionary values containing lists of repeated GO IDs.

Converting the GO IDs to their terms was then achieved in the *id_to_term* function. The *get_ontologies* function aids this process by saving all encountered GO IDs in the given OBO file/URL as keys and their associated GO terms as values (by finding each stanza titled "[Term]") to then be used by the *id_to_term* function, which extracts the desired terms and associates them to their respective genes in a dictionary through their GO IDs. These terms describe the gene product's function and/or site of action in relation to the cell and its components. This information is crucial to understand the impact that the mutation suffered by each gene has on the global performance of the cellular machinery and, thus, the overall health of the patient.

"Generalization" was a keyword in the making of MutantGene and heavily influenced the functions. While generalization is visible in all created functions as a whole (such as the fact that users can choose which information to extract from MAF file variants and in which order), it is most noticeable in the auxiliary functions. These were made to be used in as many steps of a workflow as possible. The functions that save retrieved information to files can be used after getting results by any of the main functions, depending on if the results are lists (*list_to_file*) or dictionaries (*dict_to_file*). The filter functions (*filter_list* for lists and *filter_dict* for dictionaries) also do this to the results of pretty much any of the functions (except for filtering the values of the dictionary resulting from the *maf_index* function, as they are integers which is not supported by the function created). The functions that grab information from saved files and stores them in lists (*file_to_list*) or dictionaries (*file_to_dict*) can also be used to process information (such as the filter list that was given to the *filter_dict* function in the workflow example) or be itself processed. However, the *file_to_dict* function requires the information in the file to be saved in a specific format, where each key-value pair

must be saved in a unique line in the file and separated from each other by the characters ": ".

The workflow example given has the objective of explaining how each function can be used, what they return and how they interact with each other. Moreover, it exemplifies how the results start narrowing down after each step of the workflow. To clarify, 100 variants were chosen to extract information from in the MAF file, and in the end (after filtering low impact mutations, removing repeated genes, eliminating genes with no GO IDs in the GOA file chosen, and filtering the genes who's products' functions are not associated to the cytoplasm) only 24 genes remained in the final dictionary with their GO terms. This process is dynamic and depends on each user's goal.

MutantGene library evolution

For the first drafts of the library, an additional function named *protein_id* was created to link the MAF file to an ID Mapping file, which lists different identifiers from different cross-reference databases for each protein[116]. This was used in order to retrieve both Swiss-Prot and TrEMBL UniProtKB accession numbers of the genes collected using the *mutation_samples* function. This function would return a dictionary that stored the gene names as keys and a list of their corresponding protein IDs as values. From there, the *protein_id* function would be used to collect the protein IDs to then retrieve the GO IDs of each gene (instead of directly using the gene symbol for this, which is the method used and described in this project).

The *id_to_term* function was also initially different. To obtain the GO terms of each GO ID, a website called Gene Ontology Normal Usage Tracking System (GONUTS) wiki was used, which is a community-based browser, created by users at Texas A&M University, that complements information from the GO consortium[117]. The GONUTS wiki was used by the old function to obtain GO terms instead of the official QuickGO browser because the HTML script of the latter contains no information that can be extracted or used, contrary to the former. However, this function was split into 2 parts (*get_ontologies* and *id_to_term*) and now uses an OBO file URL to collect all terms instead of connecting to a unique website for each new term.

Lastly, the semi-final version of the library contained just 5 functions overall and offered little workflow variations. The latter functions would call all former functions before it, which meant that the final function would require an input of all parameters used in the 4 before it (such as a MAF file, variant limit, variant starting point, GOA file, etc.) These were finally divided into 13 simpler functions to allow different workflow approaches and personalized variant and gene selections. Further improvements can be made to the library in the future to better organize the genes extracted and possibly even characterize the mutations themselves.

Conclusion

Cancer is a severe issue that doctors and researchers have been attempting to battle for thousands of years. It can arise through many means, from UV radiation to viruses, and can affect any part of the body. The difficulty faced in the combat against cancer is in large part due to its constant mutability and the problems faced with identifying different clones within tumour cell populations. Although various cancer treatments are now being used and are continuously evolving, cancer recursion is frequent and mortality rates remain high. Bioinformatics arrived to tackle many different issues faced in the study of biological sequences, which, among other things, helped with better understanding and visualizing DNA sequences. Therefore, variations to these sequences can be swiftly caught, facilitating its characterization and, as is the case with cancer, its repair.

A plethora of tools has been generated, thanks to bioinformatics, which can and should be used when tackling issues such as cancer. The GDC data repository is one of them, and it contains information gathered from various cancer research groups which can be accessed between them in order to unify efforts to better understand how cancer functions and eventually perfect cancer treatment. Another is the GO Consortium, which has the potential to be a great help as it identifies unique genes and characterizes them. The marriage between the information retrievable from these data repositories may very well be a great step towards the abolition of cancer and was the primary goal of this study.

The MutantGene library contains 13 functions overall, with 7 forming the main structure of the library, in that they deal with the extraction and correlation of information stored in MAF, GOA and OBO files. The other 6 functions were created to assist the user in filtering desired/undesired results obtained from the extraction process, save their results in files to be shared (for example) or even obtain information from said files to be processed in Python. A workflow example was demonstrated in this report, showing how each function could potentially be used to obtain GO terms for each extracted and processed gene saved in a MAF file.

MutantGene suffered many alterations throughout this project, such as which files to extract information from, but ultimately resulted in the functions presented and leaves room for further development. This python library aims to assist cancer researchers in the quest to perfect both known and upcoming cancer treatments by associating specific gene mutations to their functions. When the cancer in question is made up of heterogenous sub-populations, this library can be helpful to visualize each gene suffered and their different roles. This can be used to quickly assess whether certain mutations are detrimental to cellular activity, pointing towards possible targets for targeted therapy.

References

1. van Middendorp, J. J., Sanchez, G. M. & Burridge, A. L. The Edwin Smith papyrus: a clinical reappraisal of the oldest known document on spinal injuries. *Eur Spine J.* **19**, 1815-1823 (2010).
2. American Cancer Society. Early History of Cancer. (2018). at <https://www.cancer.org/cancer/cancer-basics/history-of-cancer/what-is-cancer.html#references>
3. Strauss, M. Earliest Human Cancer Found in 1.7-Million-Year-Old Bone. (2016). at <https://www.nationalgeographic.com/news/2016/07/oldest-human-cancer-disease-origins-tumor-fossil-science/>
4. SEER Training Modules. Cancer: A Historic Perspective. at <https://training.seer.cancer.gov/disease/history/>

5. Dagogo-Jack, I. & Shaw, A. Tumour heterogeneity and resistance to cancer therapies. *Nat Rev Clin Oncol* **15**, 81–94 (2018).
6. Hanahan, D. & Weinberg, R. A. Hallmarks of cancer: the next generation. *Cell* **144**, 646–674 (2011).
7. Institute for Quality and Efficiency in Health Care. How do cancer cells grow and spread?. (2013). at <https://www.ncbi.nlm.nih.gov/books/NBK279410/>
8. National Cancer Institute. Metastatic Cancer. (2017). at <https://www.cancer.gov/types/metastatic-cancer>
9. Rankin, E. B. & Giaccia, A. J. Hypoxic control of metastasis. *Science* **352**, 175–180 (2016).
10. Chial, H. Proto-oncogenes to oncogenes to cancer. *Nature Education* **1**, 33 (2008).
11. Cooper, G. M. The Cell: A Molecular Approach. 2nd edition. (Sinauer Associates, 2000). at <https://www.ncbi.nlm.nih.gov/books/NBK9840/>
12. American Cancer Society. Oncogenes and tumor suppressor genes. (2014). at <https://www.cancer.org/cancer/cancer-causes/genetics/genes-and-cancer/oncogenes-tumor-suppressor-genes.html>
13. Bister, K. Discovery of oncogenes: The advent of molecular cancer research. *Proc. Natl. Acad. Sci.* **112**, 15259–15260 (2015).
14. Duesberg, P. H. & Vogt, P. K. Differences between the ribonucleic acids of transforming and nontransforming avian tumor viruses. *Proc. Natl. Acad. Sci.* **67**, 1673–1680 (1970).
15. Pierotti, M. A., Sozzi, G. & Croce, C. M. in *Holland-Frei Cancer Medicine. 6th edition* (Kufe, D. W., Pollock, R. E., Weichselbaum, R. R. et al.ed.) (BC Decker, 2003). at <https://www.ncbi.nlm.nih.gov/books/NBK13714/>
16. Fernández-Medarde, A. & Santos, E. Ras in cancer and developmental diseases. *Genes Cancer* **2**, 344-358 (2011).
17. Tsuchida, N., Murugan, A. K. & Grieco, M. Kirsten Ras* oncogene: significance of its discovery in human cancer research. *Oncotarget* **7**, 46717-46733 (2016).
18. Dang C. V. MYC on the path to cancer. *Cell* **149**, 22–35 (2012).

19. Chen, H., Liu, H. & Qing, G. Targeting oncogenic Myc as a strategy for cancer treatment. *Sig. Transduct. Target. Ther.* **3,** 5 (2018).
20. Ellermann, V. & Bang, O. Experimentelle Leukämie bei Hühnern. II. *Zeitschr. f. Hygiene* **63**, 231–272 (1909).
21. Justice, J., 4th & Beemon, K. L. Avian retroviral replication. *Current opinion in virology* **3**, 664-669 (2013).
22. Davis, A. S., Viera, A. J. & Mead, M. D. Leukemia: an overview for primary care. *Am Fam Physician* **89**, 731-738 (2014).
23. Rous, P. A TRANSMISSIBLE AVIAN NEOPLASM. (SARCOMA OF THE COMMON FOWL.). *J Exp Med.* **12**, 696-705 (1910).
24. Rous, P. A SARCOMA OF THE FOWL TRANSMISSIBLE BY AN AGENT SEPARABLE FROM THE TUMOR CELLS. *J Exp Med.* **13**, 397-411 (1911).
25. Martin, G. S. Rous Sarcoma Virus: a Function required for the Maintenance of the Transformed State. *Nature* **227**, 1021–1023 (1970).
26. Chang, Y., Moore, P. S. & Weiss, R. A. Human oncogenic viruses: nature and discovery. Philos. Trans. R. Soc. Lond. B Biol. Sci. **372**, 20160264 (2017).
27. Epstein, M. A., Achong, B. G. & Barr, Y. M. VIRUS PARTICLES IN CULTURED LYMPHOBLASTS FROM BURKITT'S LYMPHOMA. *Lancet* **1**, 702-703 (1964).
28. Morales-Sánchez, A. & Fuentes-Pananá, E. M. Human viruses and cancer. *Viruses* **6**, 4047-4079 (2014).
29. White, M. K., Pagano, J. S. & Khalili, K. Viruses and human cancers: a long road of discovery of molecular paradigms. *Clin. Microbiol. Rev.* **27**, 463-481 (2014).
30. Yim, E. K. & Park, J. S. The role of HPV E6 and E7 oncoproteins in HPV-associated cervical carcinogenesis. *Cancer research and treatment : official journal of Korean Cancer Association* **37**, 319–324 (2005).
31. Dow, D. E., Cunningham, C. K. & Buchanan, A. M. A Review of Human Herpesvirus 8, the Kaposi's Sarcoma-Associated Herpesvirus, in the Pediatric Population. *J. Pediatric Infect. Dis. Soc.* **3**, 66-76 (2014).

32. American Cancer Society. What is Kaposi Sarcoma?. (2018). at <https://www.cancer.org/cancer/kaposi-sarcoma/about/what-is-kaposi-sarcoma.html#references>
33. Narkhede, M., Arora, S. & Ujjani, C. Primary effusion lymphoma: current perspectives. *OncoTargets and therapy* **11**, 3747–3754 (2018).
34. Mui, U. N., Haley, C. T. & Tyring, S. K. Viral Oncology: Molecular Biology and Pathogenesis. *J Clin. Med.* **6**, 111 (2017).
35. van Elsland, D. & Neefjes, J. Bacterial infections and cancer. *EMBO Rep.* **19**, e46632 (2018).
36. Barrett, J. R. Liver Cancer and Aflatoxin: New Information from the Kenyan Outbreak. *Environ. Health Perspect.* **113**, A837-A838 (2005).
37. Hamid, A. S., Tesfamariam, I. G., Zhang, Y. & Zhang, Z. G. Aflatoxin B1-induced hepatocellular carcinoma in developing countries: Geographical distribution, mechanism of action and prevention. *Oncol. Lett.* **5**, 1087-1092 (2013).
38. van Tong, H., Brindley, P. J., Meyer, C. G. & Velavan, T. P. Parasite Infection, Carcinogenesis and Human Malignancy. *EBioMedicine* **15**, 12-23 (2017).
39. Parsonnet, J. Bacterial infection as a cause of cancer. *Environmental health perspectives* **103**, 263–268 (1995).
40. National Cancer Institute. Cancer Staging. (2015). at <https://www.cancer.gov/about-cancer/diagnosis-staging/staging>
41. Rosen, R. D. & Sapra, A. TNM Classification. *StatPearls Publishing*, (2020).
42. American Cancer Society. Cancer Staging. (2020). at <https://www.cancer.org/treatment/understanding-your-diagnosis/staging.html#references>
43. Zellmer, V. R., Zhang, S. Evolving concepts of tumor heterogeneity. *Cell Biosci* **4,** 69 (2014).
44. Marusyk, A., Almendro, V., Polyak, K. Intra-tumour heterogeneity: a looking glass for cancer?. *Nat Rev Cancer* **12**, 323–334 (2012).
45. Roberts, S. A. & Gordenin, D. A. Hypermutation in human cancer genomes: footprints and mechanisms. *Nat Rev Cancer* **14**, 786–800 (2014).

46. Ayob, A. Z., Ramasamy, T. S. Cancer stem cells as key drivers of tumour progression. *J Biomed. Sci.* **25,** 20 (2018).
47. Yu, Z., Pestell, T. G., Lisanti, M. P. & Pestell, R. G. Cancer stem cells. *The international journal of biochemistry & cell biology* **44**, 2144–2151 (2012).
48. Thierry, A. R. *et al.* Clinical utility of circulating DNA analysis for rapid detection of actionable mutations to select metastatic colorectal patients for anti-EGFR treatment. *Ann. Oncol.* **28**, 2149–2159 (2017).
49. Mattox, A. K. *et al.* Applications of liquid biopsies for cancer. *Sci. Transl. Med.* **11**, 1–4 (2019).
50. Azim, F. S., Houri, H., Ghalavand, Z. & Nikmanesh, B. Next Generation Sequencing in Clinical Oncology : Applications , Challenges and Promises : A Review Article. **47**, 1453–1457 (2018).
51. Kulski, J. K. in *Next Generation Sequencing - Advances, Applications and Challenges* (Kulski, J. K.ed.) (InTech, 2016).
52. Bedard, P., Hansen, A., Ratain, M. *et al.* Tumour heterogeneity in the clinic. *Nature* **501,** 355–364 (2013).
53. Navin, N. E. Cancer genomics: one cell at a time. *Genome Biol* **15,** 452 (2014).
54. Nono, A. D., Chen, K. & Liu, X. Comparison of different functional prediction scores using a gene-based permutation model for identifying cancer driver genes. *BMC Med Genomics* **12**, 22 (2019).
55. Luo, S.Y. & Lam, D.C. Oncogenic driver mutations in lung cancer. *Transl Respir Med* **1**, 6 (2013).
56. Sigismund, S., Avanzato, D. & Lanzetti, L. Emerging functions of the EGFR in cancer. *Molecular oncology* **12**, 3–20 (2018).
57. Burns, M. B., Lackey, L., Carpenter, M. A. *et al.* APOBEC3B is an enzymatic source of mutation in breast cancer. *Nature* **494**, 366–70 (2013).
58. Swanton, C., McGranahan, N., Starrett, G. J. & Harris, R. S. APOBEC enzymes: mutagenic fuel for cancer evolution and heterogeneity. *Cancer Discov.* **5**, 704–712 (2015).
59. Lee, V. et al. Mismatch repair deficiency and response to immune checkpoint blockade. *Oncologist* **21**, 1200–1211 (2016).

60. Le, D. T., Uram, J. N., Wang, H. *et al*. PD-1 Blockade in Tumors with Mismatch-Repair Deficiency. *New England Journal of Medicine* **372**, 2509–2520 (2015).
61. University of California Museum of Paleontology. Bottlenecks, BRCA, and Breast Cancer. (2013). at <https://evolution.berkeley.edu/evolibrary/news/131211_bottlenecks>
62. Goren-Inbar, N., Freikman, M., Garfinkel, Y. *et al*. The earliest matches. *PloS one* **7**, e42213 (2012).
63. Underwood, C. Electrocauterization: Purpose, Procedure & Risks. (2018). at <https://www.healthline.com/health/electrocauterization>
64. American Cancer Society. How Surgery Is Used for Cancer. (2019). at <https://www.cancer.org/treatment/treatments-and-side-effects/treatment-types/surgery/how-surgery-is-used-for-cancer.html>
65. Jin. H. T., Ahmed, R. & Okazaki, T. in *Negative Co-Receptors and Ligands*. (Ahmed, R. & Honjo, T.ed.) 17 (Springer-Verlag, Berlin, Heidelberg, 2010).
66. Llosa, N. J., Cruise, M., Tam, A. *et al*. The vigorous immune microenvironment of microsatellite instable colon cancer is balanced by multiple counter-inhibitory checkpoints. *Cancer discovery* **5**, 43–51 (2015).
67. American Cancer Society. Evolution of Cancer Treatments: Immunotherapy. (2014). at <https://www.cancer.org/cancer/cancer-basics/history-of-cancer/cancer-treatment-immunotherapy.html>
68. National Cancer Institute. Immunotherapy to Treat Cancer. (2019). at <https://www.cancer.gov/about-cancer/treatment/types/immunotherapy#what-are-the-types-of-immunotherapy>
69. American Cancer Society. Evolution of Cancer Treatments: Chemotherapy. (2014). at <https://www.cancer.org/cancer/cancer-basics/history-of-cancer/cancer-treatment-chemo.html>
70. Mandal, A. History of Chemotherapy. (2019). at <https://www.news-medical.net/health/History-of-Chemotherapy.aspx>
71. National Institute of Diabetes and Digestive and Kidney Diseases. Alkylating Agents. (2015). at <https://www.ncbi.nlm.nih.gov/books/NBK547849/>

72. American Cancer Society. How Is Chemotherapy Used to Treat Cancer?. (2019). at <https://www.cancer.org/treatment/treatments-and-side-effects/treatment-types/chemotherapy/how-is-chemotherapy-used-to-treat-cancer.html>
73. Bishop, B. N. & Edemekong, P. F. Choriocarcinoma. *StatPearls Publishing* (2020).
74. Faltas, B. M., Prandi, D., Tagawa, S. T. *et al*. Clonal evolution of chemotherapy-resistant urothelial carcinoma. *Nat Genet.* **48**, 1490-1499 (2016).
75. American Cancer Society. Chemotherapy Side Effects. (2020). at <https://www.cancer.org/treatment/treatments-and-side-effects/treatment-types/chemotherapy/chemotherapy-side-effects.html>
76. Ishino, Y., Shinagawa, H., Makino, K., Amemura, M. & Nakata, A. Nucleotide sequence of the iap gene, responsible for alkaline phosphatase isozyme conversion in Escherichia coli, and identification of the gene product. *J Bacteriol* **169**, 5429–5433 (1987).
77. Ishino, Y., Krupovic, M. & Forterre, P. History of CRISPR-Cas from Encounter with a Mysterious Repeated Sequence to Genome Editing Technology. *J. Bacteriol.* **200**, e00580-17 (2018).
78. Mojica, F. J. M., Díez-Villaseñor, C., García-Martínez, J., Soria, E. Intervening sequences of regularly spaced prokaryotic repeats derive from foreign genetic elements. *J Mol. Evol.* **60**, 174–182 (2005).
79. MedlinePlus. What are genome editing and CRISPR-Cas9?. (2020). at <https://medlineplus.gov/genetics/understanding/genomicresearch/genomeediting/>
80. Dyda, F. & Hickman, A. B. Mechanism of spacer integration links the CRISPR/Cas system to transposition as a form of mobile DNA. *Mobile DNA* **6**, 9 (2015).
81. Charpentier, E., Doudna, J. A. *et al.* A Programmable Dual-RNA–Guided DNA Endonuclease in Adaptive Bacterial Immunity. *Science* **337**, 816-821 (2012).
82. Reiss, A. *et al.* CRISPR/Cas9 & Targeted Genome Editing: New Era in Molecular Biology. (2014).

83. National Cancer Institute. How CRISPR Is Changing Cancer Research and Treatment. (2020). at <https://www.cancer.gov/news-events/cancer-currents-blog/2020/crispr-cancer-research-treatment>
84. World Health Organization. Cancer. (2018). at <https://www.who.int/news-room/fact-sheets/detail/cancer>
85. Noorbakhsh, J., Kim, H., Namburi, S. et al. Distribution-based measures of tumor heterogeneity are sensitive to mutation calling and lack strong clinical predictive power. *Sci Rep* **8**, 11445 (2018).
86. Brenner, C. Applications of Bioinformatics in Cancer. *Cancers* **11**, 1630 (2019).
87. Hemmendinger, D. et al. Computer. (2020). at <https://www.britannica.com/technology/computer>
88. Gauthier, J., Vincent, A. T., Charette, S. J & Derome, N. A brief history of bioinformatics *Briefings in Bioinformatics* **20**, 1981–1996 (2018).
89. Moore, J. H. Bioinformatics. *J Cell Physiol* **213**, 365-369 (2007).
90. Bayat, A. Science, medicine, and the future: Bioinformatics. *BMJ (Clinical research ed.)* **324**, 1018–1022 (2002).
91. Hogeweg, P. The Roots of Bioinformatics in Theoretical Biology. *PLoS Comput Biol.* **7**, e1002021 (2011).
92. Can, T. Introduction to bioinformatics. *Methods in molecular biology (Clifton, N.J.)* **1107**, 51-71 (2014).
93. Gupta, O. P. & Rani, S. Bioinformatics Applications and Tools: An Overview. *Biometrics and Bioinformatics* **3**, 107-110 (2011).
94. Gregerson, E. Ada Lovelace: The First Computer Programmer. at <https://www.britannica.com/story/ada-lovelace-the-first-computer-programmer>
95. Encyclopædia Britannica. Assembly Language. at <https://www.britannica.com/technology/assembly-language>
96. JournalDev. History Of Python Programming Language. at <https://www.journaldev.com/34415/history-of-python-programming-language>
97. Ekmekci, B., McAnany, C. E. & Mura, C. An Introduction to Programming for Bioscientists: A Python-Based Primer. *PLoS Comput Biol.* **12**, e1004867 (2016).

98. National Cancer Institute Overview and Mission. (2018). at <https://www.cancer.gov/about-nci/overview>

99. The Cancer Genome Atlas Program. at <https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga>

100. GDC Overview. at <https://gdc.cancer.gov/about-gdc/gdc-overview>

101. Grossman, R. L., Heath, A. P., Ferretti, V. *et al*. Toward a Shared Vision for Cancer Genomic Data. *New England Journal of Medicine* **375**, 1109-1112 (2016).

102. GDC VCF Format. at <https://docs.gdc.cancer.gov/Data/File_Formats/VCF_Format/>

103. Leipzig, J. A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics* **18**, 530-536 (2017).

104. GDC MAF Format v.1.0.0. at <https://docs.gdc.cancer.gov/Data/File_Formats/MAF_Format/>

105. Povey, S., Lovering, R. C., Bruford, E. A. *et al*. The HUGO Gene Nomenclature Committee (HGNC). Nomenclature Recommendations. *Human genetics* **109**, 678-80 (2002).

106. den Dunnen, J. T., Dalgleish, R., Maglott, D. R. *et al*. HGVS Recommendations for the Description of Sequence Variants: 2016 Update. *Human Mutation* **37**, 564-569 (2016).

107. den Dunnen, J. T., & Antonarakis, S. E. Mutation nomenclature extensions and suggestions to describe complex mutations: a discussion. *Human mutation* **15**, 7–12 (2000).

108. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature genetics* **25**, 25–29 (2000).

109. Ashburner, M., Ball, C. A., Blake, J. A. *et al*. The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Research* **47**, D330–D338 (2019).

110. Mungall, C. & Ireland, A. The OBO Flat File Format Guide, version 1.4. (2020).

111. Huntley, R., Dimmer, E., Barrell, D. *et al.* The Gene Ontology Annotation (GOA) Database. *Nat Prec* (2009).

112. The UniProt Consortium. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research* **47**, D506–D515 (2019).

113. The UniProt Consortium. UniProt: the universal protein knowledgebase *Nucleic Acids Research* **45**, D158–D169 (2017).

114. Huntley, R. P., Sawford, T., Mutowo-Meullenet, P. *et al*. The GOA database: gene Ontology annotation updates for 2015. *Nucleic acids research* **43**, D1057–D1063 (2015).

115. Binns, D., Dimmer, E., Huntley, R. *et al*. QuickGO: a web-based tool for Gene Ontology searching. *Bioinformatics (Oxford, England)* **25**, 3045–3046 (2009).

116. Uniprot Consortium. README. (2002). https://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/idmapping/README

117. Renfro, D. P. et al. GONUTS: the Gene Ontology Normal Usage Tracking System. *Nucleic acids research* **40**, D1262-9 (2012).