University of BRISTOL

Department of Computer Science

Human Genetic Variation
Single-Nucleotide Polymorphisms

Dionysios-Dimokratis Prinos

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering

September 2010 | CSMSC-10

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in Advanced Computing in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Dionysios-Dimokratis Prinos, September 2010

Executive summary

The aim and purpose of this dissertation was to investigate two particular types of mutations in the human DNA sequence, the non-synonymous single-nucleotide polymorphisms (SNPs) and the synonymous SNPs.

This project was divided into three main stages, the data extraction, the data mapping and the data analysis stages. The first stage required the extraction of data from the 1000 genome project web site, which was based on dynamic content and more specific in Ajax. The two main difficulties, which were the JavaScript generated content and the large amount of time needed to extract the data were overcome and the SNP data were collected successfully. Afterwards, in the data mapping stage, the SNP data were mapped to the superfamily database, which has all the proteins' information needed for the mappings. Each mutation was mapped to the corresponding superfamily, region and protein sequence. The final stage had to do with the analysis of the data and of the previous mappings. The results were presented graphically and were interpreted for biological meaning.

Through out the results biological concepts like natural selection, purification and evolution can be seen clearly. The analysis of the data showed that natural selection and evolution have taken place and that the mutations tend to be in balance.

Acknowledgements

I would like to express my deepest gratitude to Dr Julian Gough for his supervision and support from the start to the end of this project.

Also I would like to thank my family and friends for their support and encouragement through out this course and this dissertation.

TABLE OF CONTENTS

1 Introduction

Bioinformatics is a major field of computer science concentrating in molecular biology. Its main goal is to store and analyse a vast amount of biological data in order to help scientists to solve specific problems and to come to certain conclusions. More specific the study of genetics can be thought as the way of finding possible relations on information gathered from genes. The gene information is used to create data models, which will act as an efficient tool for distinguish and examine those relations. The study of genetics is divided into three main fields, the Mendelian genetic, the molecular genetics and the population genetics. We will concentrate in molecular genetics which provides a more extend analysis in the genetic information making possible for the researchers to examine the function, the structure and the composition of genes.

Over the past few years Bioinformatics has been focused in human genetic variation and more specific in mutations that occur in the DNA sequence between individuals within the same geographic region. These mutations are analysed for potential high percentage differences or similarities that may exist in the specific population. Afterwards, these differences and similarities are associated to major diseases such as cancer and diabetes and form a very significant solid tool which will help to understand their origin and to come to the desired cure.

These mutations are also called single nucleotide polymorphisms (SNPs). SNPs are variations in the DNA sequence that occur when in the genome sequence a single nucleotide is altered. SNPs can be distinguished into two main categories, harmless and harmful SNPs. The ones that occur in a non-coding region are of less significance and cause harmless effects than the ones in a coding region, which can be harmful. Moreover, the SNPs in a coding region which can cause harmful effects are characterised as non-synonymous and these are the ones in which scientists and this dissertation are most interested in.

A careful analysis of the relation between SNPs and individuals could reveal new locations (loci) in the DNA sequence, which contributes to several common human diseases. Although SNPs are not proved to cause a disease directly, they can be used as a mean of predicting the likelihood that a person could develop a particular disease. Until now, more than 50 loci have been identified which have been associated with diseases like diabetes and heart attack. A very simple example is the apolipoprotein E that is associated with the Alzheimer's disease and has two SNPs.

The purpose of this dissertation is to discover differences and similarities between individuals and relate these to differences in their proteins. The main task of this project is to process SNP data coming out of the 1000 genome project. The 1000 genome project, launched in January 2008, it's a very recent research and its main purpose is to establish the most detailed genome-wide catalog of human genetic variation, which will include most of the SNPs. The data process task is divided into three basic stages, data extraction, data mapping and data analysis. The first stage includes a web extraction of the latest SNP data from the 1000 genome project website. Afterwards, in the second stage, the previous acquired SNP data are mapped to the superfamily database and more specific to the corresponded protein sequence. The final stage is an analysis of the mappings and an interpretation of those findings in a more biological aspect. The main goal was first to answer some basic questions like, which domains in the superfamily have many or few SNPs, are these SNPs hydrophobic or hydrophilic and which SNPs occurred in regions of high conservation and secondly to give a biological meaning to these findings.

1.1 Aims and Objectives

This bioinformatics project aims to analyse mutations (SNPs) that occur in the human DNA sequence. It is divided into three main stages:

- The first stage was to extract the single-nucleotide polymorphisms data from the 1000 genome project web site. There was the need to overcome the dynamic nature of the site and find a efficient way to extract the data from the Ajax/JavaScript based tables.

- Secondly, these SNP data had to be mapped to the superfamily database, which has all the proteins' information, and more specific to the corresponding superfamily, region and protein sequence.

- The last stage was to analyse the data from the previous mappings and to interpret the results for biological meaning.

This project is the first step in understanding the meaning and the importance of DNA mutations, a very new area, which can be considered one of the most important in biological research.

2 Background and related work

2.1 Biological References

In order for someone to understand the importance and the main idea of this dissertation, first he or she must comprehend some basic concepts in biology, such as the SNP, the amino acid and the protein. It's very important to distinguish the dependence between those concepts and their significance in human nature.

2.1.1 Single nucleotide polymorphism (SNP)

The human DNA and the whole DNA sequence is consisted of pairs of nucleotides which are combined all together in a vertical chain structure. The single nucleotide polymorphism is a variation in the DNA sequence, occurring when a single nucleotide, A(Adenine), T(Thymine), G(Guanine), or C(Cytosine) differs between members of a species and in this case between individuals. These variations are a result of mutations that have happened in the DNA structure. There are several reasons for these mutations such as viruses, chemicals and errors that may take place in the DNA replication.

2.1.2 Amino Acids

The amino acids are molecules that are responsible for the creation of each protein found in the body. Every person has twenty amino acids, which are used as building blocks for the construction of the thousands proteins that the human body uses. The selection of each amino acid depends on the particular codon in the DNA sequence. A codon is a sequence of three adjacent nucleotides. For example, codon ATT corresponds to the Isoleucine amino acid. If a single nucleotide changes then the specific amino acid will also change and thus the type of protein that was to be created.

Amino Acid	*SLC*	*DNA codons*
Isoleucine	I	ATT, ATC, ATA
Leucine	L	CTT, CTC, CTA, CTG, TTA, TTG
Valine	V	GTT, GTC, GTA, GTG
Phenylalanine	F	TTT, TTC
Methionine	M	ATG
Cysteine	C	TGT, TGC
Alanine	A	GCT, GCC, GCA, GCG
Glycine	G	GGT, GGC, GGA, GGG
Proline	P	CCT, CCC, CCA, CCG
Threonine	T	ACT, ACC, ACA, ACG
Serine	S	TCT, TCC, TCA, TCG, AGT, AGC

Tyrosine	Y	TAT, TAC
Tryptophan	W	TGG
Glutamine	Q	CAA, CAG
Asparagine	N	AAT, AAC
Histidine	H	CAT, CAC
Glutamic acid	E	GAA, GAG
Aspartic acid	D	GAT, GAC
Lysine	K	AAA, AAG
Arginine	R	CGT, CGC, CGA, CGG, AGA, AGG
Stop codons	Stop	TAA, TAG, TGA

Figure 1.

The previous table shows that when a single nucleotide changes in a codon does not necessarily mean that there will be also a change in the amino acid.

2.1.3 Proteins

Proteins are large molecules which are composed of one ore more amino acids. If one of those amino acids changes, the produced protein will also be different. All the proteins are unique and have different functions from each other. Moreover, they are essential components of the human body and they are responsible for the proper structure and function of the body's cells and organs.

This logical relation between SNPs and proteins shows the great importance of the SNPs in the development of the human body and the need of investigating their effects as a whole.

2.1.4 Synonymous and non-Synonymous SNPs

Those SNPs that occur within a coding region of a gene can be characterised either as synonymous or non-synonymous ones. A non-synonymous SNP, which is also called a missense mutation, is the SNP that it is able to change a produced amino acid. For a synonymous SNP (silent mutation), despite the alteration that causes in a codon, the produced amino acid remains the same.

2.1.5 Hydrophobicity

Hydrophobicity is a biological concept that refers to molecules, which are not able to be dissolved in the water. The molecules that can be dissolved in water are described as hydrophilic and the ones that cannot as hydrophobic. Hydrophobicity is very important for the proteins because it provides stability to their structure. It forms a kind of physical shield for avoiding contact with water. A SNP can either have a hydrophilic or a hydrophobic effect depending on the change of the amino acid and on the hydrophobicity scale of that amino acid.

2.1.6 Substitution Matrix (BLOSUM62)

A substitution matrix describes the rate that a character changes to another through time. At bioinformatics these characters represent the amino acids whose differences in the sequences depend on the mutation rates.

At the final stage of the project the following substitution matrix, BLOSUM62, is used:

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	**9**	-1	-1	-3	0	-3	-3	-3	-4	-3	-3	-3	-3	-1	-1	-1	-1	-2	-2	-2
S	-1	**4**	1	-1	1	0	1	0	0	0	-1	-1	0	-1	-2	-2	-2	-2	-2	-3
T	-1	1	**4**	1	-1	1	0	1	0	0	0	-1	0	-1	-2	-2	-2	-2	-2	-3
P	-3	-1	1	**7**	-1	-2	-1	-1	-1	-1	-2	-2	-1	-2	-3	-3	-2	-4	-3	-4
A	0	1	-1	-1	**4**	0	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-2	-3
G	-3	0	1	-2	0	**6**	-2	-1	-2	-2	-2	-2	-2	-3	-4	-4	0	-3	-3	-2
N	-3	1	0	-2	-2	0	**6**	1	0	0	-1	0	0	-2	-3	-3	-3	-3	-2	-4
D	-3	0	1	-1	-2	-1	1	**6**	2	0	-1	-2	-1	-3	-3	-4	-3	-3	-3	-4
E	-4	0	0	-1	-1	-2	0	2	**5**	2	0	0	1	-2	-3	-3	-3	-3	-2	-3
Q	-3	0	0	-1	-1	-2	0	0	2	**5**	0	1	1	0	-3	-2	-2	-3	-1	-2
H	-3	-1	0	-2	-2	-2	1	1	0	0	**8**	0	-1	-2	-3	-3	-2	-1	2	-2
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	**5**	2	-1	-3	-2	-3	-3	-2	-3
K	-3	0	0	-1	-1	-2	0	-1	1	1	-1	2	**5**	-1	-3	-2	-3	-3	-2	-3
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	**5**	1	2	-2	0	-1	-1
I	-1	-2	-2	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	**4**	2	1	0	-1	-3
L	-1	-2	-2	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	**4**	3	0	-1	-2
V	-1	-2	-2	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	**4**	-1	-1	-3
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	**6**	3	1
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	**7**	2
W	-2	-3	-3	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	**11**

Figure 2.

This table shows how different the amino acid would be when a SNP causes the change. The smaller a negative number is the bigger the difference between the old and the new amino acid. If the number goes higher then we tend to have a smaller difference.

2.1.7 Natural Selection

In a population, in our case a human population, some times some characteristics (traits) are becoming more or less common over the generations. This is called natural selection, a process that nature uses in order to favour evolution and to permit and to preserve life within species.

Natural selection can be distinguished in two further types, negative and positive selection. Negative selection, also called purifying selection, is when a selective removal of deleterious alleles takes place in the human DNA. This means that nature, through the DNA repair mechanism, removes the harmful mutations in order to allow the evolution to continue. As described before the harmful mutations (non-synonymous) are those mutations that could change the amino acid and thus the corresponding protein that will be produced. Positive selection is very rare and it means exactly the opposite than the negative selection. In positive selection there is no removal of the deleterious alleles and thus the evolution is forced to go backwards.

2.2 Data Extraction

The first task in this project was to extract the SNP data from the 1000 genome project website (http://browser.1000genomes.org/index.html). We are focusing on the non-synonymous SNPs. The columns that were needed for this research were the 'ID', the 'Type', the 'Reference allele', the 'AA change' and the 'AA coordinance' of each non-synonymous SNP. The 'Type' field helps to distinguish the non-synonynous SNPs, the 'Reference allele' and the 'AA change' fields show the change of the amino acid, which is important for the hydrophobicity analysis and the 'AA coordinance' is used for the mapping task to the superfamily database.

Data extraction is the process in which required data could be retrieved from various and different data sources in order to be processed or stored. Generally this data comes out of unstructured data sources such as web pages, documents and e-mails and require some data transformation during the extraction. Although the act of the data extraction may seem an easy task, it has become quite a technical challenge due to the variety of data formats. Web data extraction or web scraping is the process that refers to the extraction of data out of the web.

2.2.1 Web data extraction – Web scraping

Day by day, due to the great amount of exchanged data, the Web has become the richest and biggest database in the world. However, this data is in HTML format and can only be viewed through a browser, having no other value in its current form. Web scarping is a technique based on software implementation for extracting unstructured data from web sites, which is mainly in HTML format, and transforms it into structured data, making it possible to be stored in a database for further analysis. There are several techniques for

Web scraping, scaling from very simple to very complex ones depending on the need of the current data extraction. As described in [7] those techniques are:

- Human copy-paste: This is a very raw technique, yet is useful when websites are structured in a way of not allowing web data extraction to be performed.
- Text grepping and regular expression matching: Another way to extract data from the Web is with the help of grep command, a text search utility for Unix, and with programming languages based on expression matching like Perl and Python.
- HTTP programming: Sending HTTP requests to the targeted Web server, using client-server programming.
- DOM(Document Object Model) parsing: Another solution is to embed a Web browser control in order for programs to retrieve the desired web content. This control can parse web pages into a DOM tree, which can provide information for the extraction of specific parts of the particular web pages.
- HTML parsers: Web data extraction is possible by using query-programming languages like XQL (XML query language) and HTQL (Hyper-text query language). These programming languages can retrieve the desired data by parsing HTML pages.
- Web scraping software: There are several software available for Web scraping, such as scripts for extracting Web data and database interfaces for storing this data to a database, that could be embedded in a Web scraping implementation.
- Semantic annotation recognizing: There can be several semantic annotations in a Web page, which can be used to locate specific data segments.

The data extraction task, since it is not so trivial, it can be divided into further subtasks. In each task, we could assign a different operator as described in [2],[4]:

- HTTP query building: An HTTP query is basically a construction that contains a query method, a URL and a set of keys with the corresponding values. The HTTP query operator creates those parts based on the given input.
- Fetching: This operator uses a URL or an HTTP request as an input and downloads the requested data. Its output is simply an HTTP response.
- Web service querying: A set of parameters is used as an input for this kind of operator; two of those parameters are the description of the Web service and the calling method. The output that the Web service-querying operator produces is a list, which contains the Web service's SOAP envelope.

- Parsing: The parsing operator creates a DOM object by parsing data in HTML or XML format. This DOM object can be used to easily distinguish and access specific parts of the data.
- Filtering: All input data are checked if they can verify a specified key value. Only the data that passes this verification procedure are returned.
- Extracting: A set of expressions is used by the extracting operator in order to choose which parts of the input are to be extracted.
- Transforming: The transforming operator is responsible for altering its input's format. An XSL stylesheet can be used to represent an HTML or an XML transformation.

The fact that the desired data, which need to be extracted, is in HTML pages, creates some significant challenges that, depending each time on the current page, require specific technical approach in order to result in an efficient data extraction. These challenges are explained in [3]:

- Update frequency: Internet is a very dynamic data source and its very difficult to keep data up to date, especially when they are continues altered and removed.
- Semi-structure: The HTML pages and generally the HTML data are in semi-structure format, which means that is not easy to find a standard way to extract Web data. For example, the schema may be too large and change very often or different objects may not have the same value type for the same attribute.
- Ill from page: An HTML page can be described as being ill form. This means that it has syntax errors, like missing attributes or closing tags, fact that make the extraction less feasible.
- Semantic Heterogeneous: The data could be retrieved from several different pages in which the semantic representations could be different, such as the variation of time and date representation between countries. These differences must be interpreted into a form that has the same meaning.
- Hidden Content: Many web pages, except from the standard HTML code, they may also use scripts like JavaScript and VBScript. These scripts can dynamically change the content of the page making the extraction task very hard.

In this project there was the need to overcome the hidden content problem. The web page from where the data have been extracted uses Ajax technology to represent the table with the information that we want to acquire creating a non-static content within the page. This can be seen in the next HTML code, which has been taken from this particular web page (http://browser.1000genomes.org/Homo_sapiens/Transcript/ Population?db=core;t= ENST00000215539):

```
<p class="invisible">.</p></div>
   <div class="panel"><div class="nav-heading">
         <div class="left-button print_hide"><a
href="/Homo_sapiens/Transcript/GO?db=core;g=ENSG00000099769;r=16:1780417-
1783735;t=ENST00000215539">&laquo;&nbsp;Gene ontology</a></div>
         <div class="right-button print_hide"><a
href="/Homo_sapiens/Transcript/Population/Image?db=core;g=ENSG00000099769;r=1
6:1780417-1783735;t=ENST00000215539">Comparison
image&nbsp;&raquo;</a></div><h2>Population comparison</h2>
         <p class="invisible">.</p></div>
      <div class="content"><div class="ajax"
title="['/Homo_sapiens/Component/Transcript/Web/TranscriptSNPTable?db=core
;t=ENST00000215539;_rmd=0110']"></div><div class="ajax"
title="['/Homo_sapiens/Component/Transcript/Web/TranscriptSNPInfo?db=core;t
=ENST00000215539;_rmd=0110']"></div>
      </div>
<p class="invisible">.</p></div>
```

Figure 3.

2.2.2 Web data extraction and Ajax

Ajax (Asynchronous JavaScript and XML) is a technique used for creating dynamic interfaces on web pages in order for them to load faster and consume less bandwidth. These web pages use Ajax to periodically exchange data in the background with the server. This helps web pages to update a specific content by just reloading that part and not the whole page. On the contrary, simple structured HTML pages need to reload the whole page for just updating a part of it. Although Ajax pages provide speed and bandwidth efficiency, due to their dynamic nature and their asynchronous loading, it is hard to extract data by just using a simple HTTP connection as in older classic methods of web data extraction.

A technique for extracting data from an Ajax based web page is described in [5]. The main concept is to use an embedded browser which can captures the states of the dynamic page by retrieving each time the DOM trees. Afterwards, based on the states of the page, the page content is fetched and the desired data are extracted from it. This technique is divided in two main tasks, fetching the content of the page and extract the data.

2.2.3 Fetching Ajax content

For this task a state repository is used where all the states of a page are stored. A fetcher utility gets, one by one, all the possible states of the page by using an embedded browser and stores them in the repository. This repository will be used in the second task for extracting the desired page content. The implementation design of the fetching process is shown in the next figure [5]:

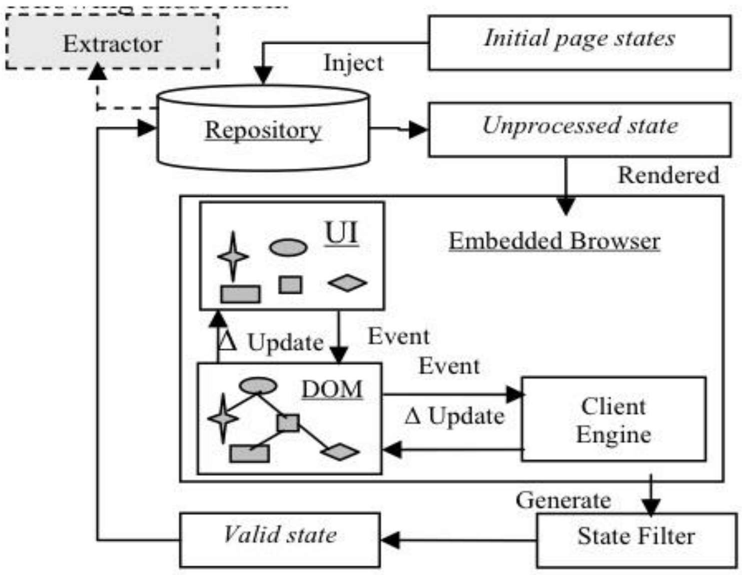


Figure 4 (taken from Ref.[5]).

The main concepts as can be seen in the previous figure are the states repository and the page state. The repository is a database, which stores all the various states of a page that the fetcher utility crawls and updates. It is a very important part of the whole fetching/extraction process because acts as a middleware between the fetcher and the extractor.

Since an Ajax based page is a dynamic web page, a single URL is not enough to describe the different states of a page. A page state is defined as a tuple of six elements, the URL of the initial state of the page, a text that describes the current state, the XpathList which tracks the path of each clicked element and helps to move from the root state to any other, the document object model (DOM) of the current state, a hash code based on that DOM for avoiding same states and a value which shows the processing state.

There are several challenges that need to be considered as concerning the fetching process. These challenges are explained in [6]:

- Client side execution: The Ajax technology uses a JavaScript engine that runs between the web server and the browser. Any implementation that wants to interact with the Ajax component must first support the JavaScript language.
- State changes and navigation: An Ajax page could only have one URL as its description, making difficult to distinguish and isolate a single state. Moreover, the task of the web page navigation is not appointed only to hypertext links but also to various events making the whole process even harder.

- Dynamic DOM: A simple extraction of the HTML code of a page would not show the current state and that is because the states are described dynamically in the DOM.

- Delta communication: The client and the server update any state change through delta communication type. In order for these updates to be retrieved and to be of some value, first they must be processed from the JavaScript engine and placed in the document object model.

- Internal state changes by elements: Hypertext links are not the only way that an Ajax page could change its state. Several events such as the onClick event could cause a page state to change, thing that traditional extraction techniques would ignore.

The next figure shows the fetching algorithm as described in [5]:

```
Procedure Fetcher:
repo = initStatesRepository();
injectInitialStates(repo);
browser = initBrowser();
fetching(repo);
extracting(repo);

Procedure fetching(repo):
while(repo has new state):
  cs = getNewState(repo);
  browser.goto(cs);
  dom = browser.getDom();
  E = getFilteredClickables(dom);
   for each e in E:
     browser.click(e);
     if browser.location<>cs.root:
        new_state = new state(browser.location, e);
     else if(sim(dom, browser.getDom()) <= τ)):
         new_state = new state(cs, e);
        repo.add(new_state);
        browser.goto(cs)
              cs.update("fetched", dom, md5(dom));
      repo.update(cs);
```

Figure 6 (taken from Ref.[5])

The algorithm starts with setting the initial page state into the repository and continues until it reaches the fetching method. In the fetching method the cs variable gets the first page state from the repository and the browser loads it. Then there is a call of the getFilteredClickables method, which acquires all clickable elements in the current DOM. This method implements a tag set that stores the clickable elements, which cause a page state to change. These elements, first are filtered based on the accept and deny rules. Only the elements that assert the accept rules are consider valid. Afterwards, all the valid elements are put into E and for each element e there is an execution on the click event. The page state is changed if the new address location is changed. The new DOM is compared with the previous one in order to see if the event caused a state change. If that is true, then a new state is created and stored in the repository. Afterwards the browser processes the cs, which now have a new state. When all elements have been executed the cs is labelled as “fetched”.

2.2.4 Extracting the data

The extraction of the data is possible by using a extraction rule file. This rule file shows the relation between the web page and the data to be extracted. The first step is to specify the URL in which the data exist and use an embedded browser to first load the page and then fetch its source code. Finally, a script must be imported in order to catch the mouse event. By selecting and producing a click event on the specific region of the page, an XPath is stored and after several times the rule file is created. Having the content of the page and the rule file, the extraction of the data can now be performed.

2.3 Data Mapping

After the extraction task was finished and the SNP data were gathered they were then mapped to the superfamily database and to the protein data. This information was used to the final step, which is the data analysis. The ‘AA coordinance’ field in the SNP data, is each time the key value for mapping the particular SNP to the corresponded protein sequence. The protein sequence is divided into domain regions, with each domain region having a specified range. We examine in which domain region does the value of the ‘AA coordinance’ field fells into and proceed to the mapping of the SNP to the protein sequence. The SNP data is stored in tab-delimited text files and to a MySQL database. The targeted protein sequences and all the protein information are stored in a local MySQL database (superfamily).

2.3.1 Mapping background theory

Describing the relations of two database schemas (source and target) has been the main problem in data integration and data exchange. The way to specify these relations is called shema mapping [8],[9],[10]. A shema mapping specifies how the data, coming from a source schema, will be converted in order to be compatible for integration to the target schema. The main problem with schema mappings is that their number could be quite big between the source and the target schema, making difficult to find the proper mapping for a given application.

A schema mapping is described as a triple (S, T, Σ) [8], where S is the source schema, T is the target schema and Σ is a set of possible mappings between the two schemas. Also, a schema mapping is defined in [9] as a source schema's query, which can give a subset of a relation to the target schema. In order for a mapping to be created, three tasks must be first completed. The first of the three task described in [10], is to determine how and where the data, coming from the source database, will appear in the target database. This is called determining correspondence and defines the combination and transformation of the source values to target values. It is very important to know where source data would appear in the target schema, which means in which specific attribute and how they will be represented. That is why a function called value correspondence is used to define the transformation of the source value to a new value in the target relation. The next step is data linking and describes how various tuples from different relations in the source schema can form a single tuple in the target schema. In order for the source tulpes to be combined properly for a target relation, query graphs are used to describe the possible links between the source tulpes. The last step is data trimming and defines which of the joined tuples from the source schema will be used to create a tuple for the target schema. After the query graph from the second step is created, all the links between the source tulpes are stored and only some of them are chosen because not all of them are useful for the target relation.

2.3.2 Database Backup

The main and obvious reason of a database backup is to recover and restore the data once a problem occurs, such as hardware failures. Although a backup could be used in a different way. In order to maintain the integrity of the original database and also to increase the speed of the data mapping and the data analysis procedures, we first backup the original database and then work on this duplicate. Due to the continues testing that is needed in order to develop the code for the previous two procedures, it is very possible to change or corrupt the data in the original database. That is why we backup the original database and create a duplicate of it locally, leaving the first intact. Another reason of taking such a step is because of the increase of the speed of the mapping and the analysis. When backing up a database, there is the choice of copying only the tables needed for the work and not the whole database. It is also possible to store those tables with just the necessary elements leaving the rest of them out. Having smaller tables with fewer elements makes the search and thus the whole procedure faster.

There are various backup and recovery types as described in [23]:

- Logical and Physical Backup: A logical backup has a structure of database statements in contrast with physical backup, which has raw copies of files containing the database elements.

- Online and Offline Backup: An online backup occurs when the database server is running and the offline backup when it is down.

- Local and Remote Backup: A backup is characterised as local when it is done from the same host where the database server runs. The remote backup is done from a different host.

- Snapshot Backup: This type of backup stores a copy of the file system in a specific point in time.

- Full and Incremental Backup: A full backup stores all the data from a database. The incremental backup consists of only the changes that occur in the database.

- Table Maintenance: Frequent checks for possible table corruption.

- Backup Scheduling, Compression and Encryption: This type of backup includes backup procedures that have been automated. They are also combined with compression and encryption for better speed and security.

The methods to backup a database are also mentioned in [23]:

- Copying Table Files: Each table has its own files in the system and by copying them we could create a backup of the database.

- Delimited-Text File Backup: A text file is created to store the table's data but not the table's structure.

- Backup with mysqldump: Mysqldump is a program which can backup any kind of tables. Also it can successfully copy and store InnoDB tables.

- Enabling the Binary Log (Incremental Backup): The binary log includes all the information that is needed to perform changes to the database from the point that the backup was taken.

- Using Replication Slaves: Due to the performance cost while making a backup on a master server, a backup could be performed on a slave server, which will have the entire original database and the master server's information.

- Recovering Corrupt Tables: There is also the option to recover the corrupted tables with a very high rate of success.

- File System Snapshot Backup: Another way to backup the data is to capture the current file state of the system and use it whenever a harmful change occurs in the database.

In this project mysqldump is used in order to copy the original database from the remote server and store it locally. With this way, we ensure the integrity of the original database and make the mapping and the analysis procedures run faster. The mysqldump program uses dump files to backup and reload the data. These files are used for various reasons as mentioned in [23]:

- Backup for recovery from possible data lose or corruption.
- Data source for creating replication slaves.
- Data source for creating a copy of a database and work on it rather than the original one.

The mysqldump program can give two types of output:

- If the --tab option is used, then it outputs Sql statements in order to backup all the objects of the database.
- If the –-tab option is missing, it outputs a text file for each table which contain all the table data. Also a Sql file is created for each table which has the statement for creating the table.

The basic command used to create a dump file, which will be in Sql format, is:

```
root> mysqldump --databases mydatabase > backup.sql
```

The basic command for reloading the dump file and creating the database is:

```
root> mysql mydatabase_copy < backup.sql
```

The mysqldump program is a very important tool for database experimentation, ensuring the integrity of the original information and data.

2.4 Data Analysis

The final task for this dissertation was an analysis of the data that arise after completing the previous task, which was to map the SNPs to the superfamily database. Several questions were answered, each of them having a specific biological significance. Moreover, there was a need of using statistic measurements like correlation, normalization and null hypothesis with ancova (analysis of covariance) in order to come to the desired results. Also, the use of 2D graphs was necessary for representing those results.

Data analysis [11],[12],[13] is the process in which data are organised and ordered so that they can be able to give the correct information that is needed to proceed to specific conclusions. It can be divided into four steps, inspecting, cleaning, transforming and modelling. There are several techniques in data analysis. The main techniques are, data mining, used more for predictive reasons, and the statistical techniques, which are the descriptive, exploratory and confirmatory statistical analysis. Two very important tools in data analysis are the charts and the graphs, which are used for the graphical representation of the results in order to form a more comprehensive outcome.

The most crucial factor in data analysis is the quality of the data. Before proceeding to the actual analysis of the data, their quality must be ensured. This can be achieved through several quality checks [13]:

- Frequency counts
- Descriptive statistics such as the standard deviation
- Normality such as the frequency histogram and the normal probability plot
- Associations like the correlation and the scatter plot.
- Data cleaning checks, which show if there is a change in the distribution of the sample after the data cleaning process, in which data are inspected and corrected in case of irrelevant and unmatched values in comparison with the sample.
- Analysis of missing and extreme observations where data are checked for having missing values or for causing a disturbance to the distribution.

2.4.1 Correlation

The correlation [11][12] is a statistic concept that shows the relationship between two variables. When a variable changes, this change may be associated to a change to a second variable. This means that these two variables are somehow related and each of them has a high or low dependency to the other. For example if a variable X increases, a second variable Y may increase as well. This relationship is characterised as positive. However, the second variable could decrease instead, forming a negative relationship. The general information (positive, negative or zero) on the relationship of two variables is called covariance and can be calculated with the next formula:

$$r_{xy} = \frac{\sum\left(\frac{x_i - \bar{x}}{S_x}\right)\left(\frac{y_i - \bar{y}}{S_y}\right)}{N - 1}$$

In order to calculate the strength of that relationship, the correlation coefficient must be used. In correlation coefficient the result is a value between -1 and 1. If its 0 or less, then there is a small or no relationship between the two variables. As it increases above 0, then the strength of that relationship becomes stronger.

$$r_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{(N-1)S_x S_y}$$

2.4.2 Normalization

Normalization [14] in its broader sense means to conform to a specific rule and come to an acceptable (normal) state. A way to return to a normal state could be by appointed to the entries of a data set values between 0 and 1 rather than leaving them with their current ones. The entries now will have the same range, making their analysis and comparison a more feasible task.

$\frac{\sigma}{\mu}$ (Coefficient of variation)

The values between different sets could be normalized by using the above formula, where μ is used as a measure of scale and can symbolise the size of the set and σ is the value or the positive distribution that we want to normalise in the given set.

2.4.3 Null hypothesis and ANCOVA (Analysis of covariance)

The null hypothesis is a statement that describes two entities as equal or as they do not have any significant difference with each other. Based on that hypothesis, we are trying to see if it is valid or not. An invalid null hypothesis often gives a more interesting conclusion rather than a valid one. This can help to understand the relation of the two entities and to come to more valid conclusions. For example, if we want to compare two regression lines and find if there is a significant difference between them, first we must take the null hypothesis that their slopes are equal:

$$H_0 : b_1 = b_2$$

Moreover, we can distinguish two types of null hypothesis [24]. The first one is the statistical hypothesis, which is based exclusively on numbers and on the comparison of averages and means. The second one is the biological null hypothesis, which uses biological theories and concepts in order to create a statement.

In order to see if the null hypothesis is valid fro two regression lines we must first calculate their slopes. The slope can be calculated by the following formula:

$$b = \frac{n(\sum XY) - (\sum X)(\sum Y)}{n(\sum X^2) - (\sum X)^2}$$

Afterwards, we must calculate and examine the t-test, which can show how different are the slopes:

$$t = \frac{\text{difference of regression slopes}}{\text{standard error of difference of regression slopes}}$$

so we have:

$$t = \frac{b_1 - b_2}{s_{b_1 - b_2}}$$

The $s_{b_1 - b_2}$ can be calculated by the next formula:

$$s_{b_1 - b_2} = \sqrt{s^2_{b_1} - s^2_{b_2}}$$

By having the value of t we can now find the P-value which is the possibility of how different are the two regression lines. In biological research the conventional significance level for the P-value is 0.05. This means that if the computed value is below 0.05, the null hypothesis is rejected. Otherwise, if it is equal or greater than 0.05, the null hypothesis stands and the two regression lines have no significant difference between them.

2.4.4 2D Graphs (Grace)

In this project there was the need to represent the results graphically, giving a more comprehensive view to the findings. Grace [15] is a tool for making two-dimensional graphs of numerical data. A graph could consist of a graph frame, a title, a subtitle, axes, a number of sets and additional annotative objects like text strings. The graph could be, either a XY graph, a XY chart, a polar graph, a fixed graph or a pie chart. In this project there was the need of using mostly pie charts, bar charts and scatter plots. The pie charts are used to show the general concentration of the calculated data, along with the bar charts, which are used to represent the spread of the data depending on a different property that they have each time. The scatter plots are used to show if a relation exists between the two targeted data and to compare this relation to a standard theoretical value in order to see if the calculations and all the work corresponds to a already valid theory and into which aspects of the specific theory the calculated data falls.

2.5 Biological Data Characteristics

Managing and processing biological data it is not an easy task since they have and share very complex and special characteristics. Especially in bioinformatics, which is the information security field that deals with DNA and protein sequence information, the complexity is of a bigger scale. These characteristics, as described in [16] must be taken into consideration before proceeding to any bioinformatics application:

- Biological data in contrast with the data of any other domain have a much more complex structure which must be carefully analysed and represented in a way that no information will be lost during the process.
- The data have a high degree of variability and many times its amount could be very vast. Putting constrains in the data may lead to an exclusion of values and a loss of important information.
- Since biological information is updated very often, the corresponded database schemas are changing at the same frequency making their management a very challenging task.
- The same data could be represented differently by biologists fact that makes difficult a possible linkage, comparison or confirmation of those data.
- The majority of the people using the database does not require more than a read-only access. The write access is only given to privileged users.
- Most biologists are unaware of the structure and the design of the database. Hence, the representation of the data must be in a very comprehensive way, such that it efficiently applies to the specific problem and gives a basic solid knowledge of the data structure.
- Many and various contexts may have been integrated to the data in order to give them a higher biological meaning and value increasing their complexity even more.
- Since the structure of the data is very complex, the related queries will have the same complexity level. This requires a very good knowledge in databases.
- Biological data before they are updated, they have their past values archived in case there is the need of verifying and comparing them with the new ones. This creates a great amount of information which management cannot be described as an easy task.

These characteristics of the biological data make any bioinformatics application and approach a quite challenging task that requires careful design and implementation.

2.6 Related Work

In [17] A.Cavallo and A.C.R Martin describe a method of mapping SNPs to protein sequence. They use two databases for this implementation, the HGVbase database and the PDB database. The first includes information about the SNPs and the genes and the second one has the protein sequences. The mapping process starts with the extraction of the SNP information from the HGVbase and continues with a gene analysis to determine if the mutation caused by the SNP is a missense mutation, which means that this particular SNP will cause a codon change and as a result a amino acid change of the particular protein. Afterwards there is an identification procedure to examine if that mutation occurs in a coding region and if so, the position number is retrieved. The mapping process finishes with the mapping of the SNP mutation (the changed nucleotide) to the protein sequence in the PDB database.

In this project as concerning the mapping task, the SNPs were extracted from the 1000 genome project website, which uses SNP information taken from the NCBI (National Centre for Biotechnology Information) and the EBI (European Bioinformatics Institute) databases. These two databases have the most recent and up to date SNP data. The data has already the category of each SNP mutation and the exact location in the protein sequence where that mutation occurs. This project uses the mapping procedure as a tool to extract crucial and vital information, which are used for the last and most important task of the data analysis. There, all the SNP information coming from the mappings are examined in order to give to each SNP a specific biological significance.

3.Project Design

This project has been divided into three stages. There is an absolute dependence between the stages because each stage uses the results from the previous one in order to be able to be completed. That is why it is crucial for each stage to finish on time and to give the correct results. The first and the most important stage was to acquire the SNP data. The only way to accomplish that was to extract the data from the 1000 genome project web site. The main difficulty was that the desired content was generated dynamically and more specific through Ajax thus the ordinary extraction techniques could not be used and there was the need of finding a programming language that could deal with such a challenge. The second stage was to map the extracted data to the superfamily database. In order to protect the integrity of the original database and to work faster, all the work was made in a copy of the database that was stored locally. The third and final stage was to make an analysis of the mappings and to interpret the results for biological meaning and mostly to have a better understanding of those DNA mutations.

3.1 Data Extraction – Web Scraping

As mentioned before the first stage of this project was to extract the SNP data from the 1000 genome project site. In order to do so, the web scraping technique was used and for every protein taken from the superfamily database there was the extraction of the corresponding mutations (non-synonymous, synonymous). The first and the most important obstacle to overcome was the fact that the mutation tables and all the information needed to be extracted ware populated through Ajax/JavaScript. This means that the tables could not be seen in the HTML code of the page, so the extraction and the parsing of the required data were not possible. For example, Perl with WWW:Mechanize is a very common tool for web extraction although it cannot work with dynamic content. Thus, there was the need of searching for new technologies that would be able to deal with this particular difficulty.

3.1.1 Technologies required

For this part, the Ruby scripting language was used along with two additional libraries, Watir [25] and Nokogiri [26]. Ruby on its own was not able to extract the dynamic generated data that is why Watir was used. Afterwards, the Nokogiri library was set to interact with Watir in order to parse the extracted data and retrieve the desired information. Nokogiri also used the XPath technology, which was essential for the parsing task. All these technologies were installed and used on a Mac OS X operating System.

The Ruby scripting language was used as a solid base in order for the Watir, Nokogiri and Xpath to work together. As we will see later, although the extraction was successful, the time factor was something that made this task more challenging.

Watir (Web Application Testing in Ruby), is an open source library, which helps automating web browsers. It is used mostly on web application development and when it is integrated with Ruby it can drive browsers like Mozilla Firefox, which is used in this project. The most important think about Watir, is its ability to extract dynamic content out of page. In our case, the dynamic content its generated through Ajax and Watir is able to capture the HTML code after the JavaScript runs and the tables are created. Moreover, Watir requires two additional things in order to work properly. Firstly, the browser that is going to be used by Watir must have a specific plugin installed. This plugin is called jssh and it is responsible for the interaction between the browser and the Watir library. Also a HTML inspector like Firebug needs to be installed on the browser to help browsing through the structure of the HTML pages.

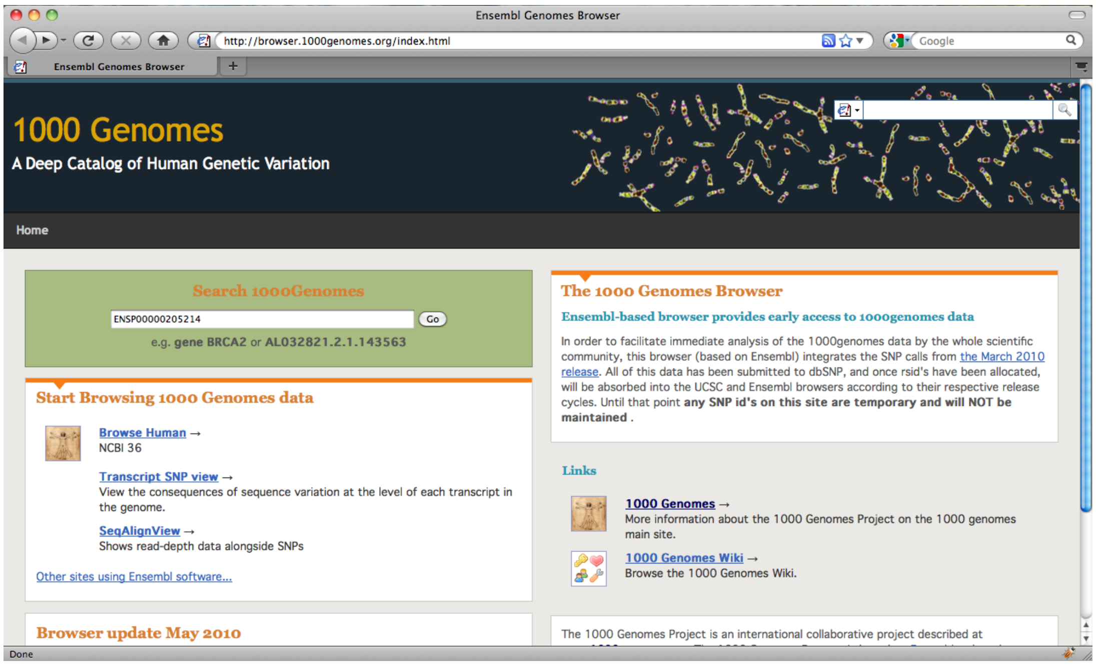
Nokogiri is also an open source library, which is designed for HTML, XML, SAX and Reader parsing. When Nokogiri library is used in a Ruby script along with the Watir library, it has the feature of interacting with Watir and reading the extracted HTML code from it. Since Watir has the HTML code with all the dynamic content of the page, Nokogiri with the help of Xpath searches and parses the desired information. Xpath, is a query language which is used to select elements from an XML document based on various criteria that the user sets. Nokogiri also needs for the MacPorts to be installed on the system, which are used to simplify the installation of open source software.

Although Watir and Nokogiri made the extraction of the SNP data possible, as two newly developed libraries, the information on their official tutorials was not sufficient enough making difficult to understand some of their features.

3.1.2 Extracting the data

The SNP data had to be extracted from the 1000 genome project web site and more specific from the mutation browser. For each protein the non-synonymous and synonymous SNP information had to be collected in order to be used later for the mapping stage. The names of all the human proteins were first retrieved from the superfamily database for extracting for each one of them the desired mutation data. The format of the proteins' name is ENSP00000123456 in which only the last six digits change each time. As we will see, some of the proteins do not have any SNPs or may have just one of the two types mentioned before.

The 1000 genome project mutation browser consists of four page steps before the SNP data could be available. The Ruby script could have navigated through all four pages in order to reach the actual page with the SNP data although this would have taken even more time making the whole process even more time consuming. Firstly, there had to be an examination of the mutation browser procedure. This is shown in the next figures in which there is an example of a protein search. The protein used for this example is ENSP00000205214, which has both non-synonymous and synonymous SNPs.


Ensembl Genomes Browser

http://browser.1000genomes.org/index.html

1000 Genomes

A Deep Catalog of Human Genetic Variation

Home

Search 1000Genomes

ENSP00000205214 Go

e.g. gene BRCA2 or AL032821.2.1.143563

Start Browsing 1000 Genomes data

Browse Human →
NCBI 36

Transcript SNP view →
View the consequences of sequence variation at the level of each transcript in the genome.

SeqAlignView →
Shows read-depth data alongside SNPs

Other sites using Ensembl software...

Browser update May 2010

The 1000 Genomes Browser

Ensembl-based browser provides early access to 1000genomes data

In order to facilitate immediate analysis of the 1000genomes data by the whole scientific community, this browser (based on Ensembl) integrates the SNP calls from the March 2010 release. All of this data has been submitted to dbSNP, and once rsid's have been allocated, will be absorbed into the UCSC and Ensembl browsers according to their respective release cycles. Until that point **any SNP id's on this site are temporary and will NOT be maintained** .

Links

1000 Genomes →
More information about the 1000 Genomes Project on the 1000 genomes main site.

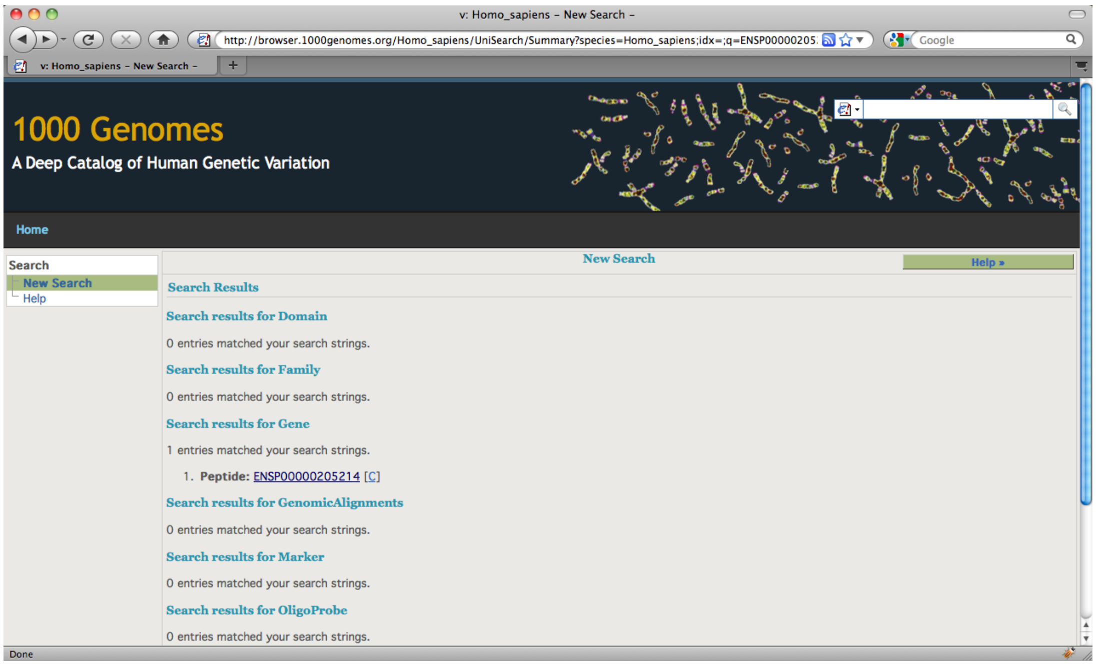
1000 Genomes Wiki →
Browse the 1000 Genomes Wiki.

The 1000 Genomes Project is an international collaborative project described at

Done

Figure 7 (taken from Ref. [27]).

The previous figure shows the first page of the mutation browser where the human protein is entered each time for starting the search for its possible SNPs.


v: Homo_sapiens – New Search –

http://browser.1000genomes.org/Homo_sapiens/UniSearch/Summary?species=Homo_sapiens;idx=;q=ENSP00000205

1000 Genomes

A Deep Catalog of Human Genetic Variation

Home

Search
New Search
Help

New Search

Help »

Search Results

Search results for Domain

0 entries matched your search strings.

Search results for Family

0 entries matched your search strings.

Search results for Gene

1 entries matched your search strings.

1. **Peptide:** ENSP00000205214 [C]

Search results for GenomicAlignments

0 entries matched your search strings.

Search results for Marker

0 entries matched your search strings.

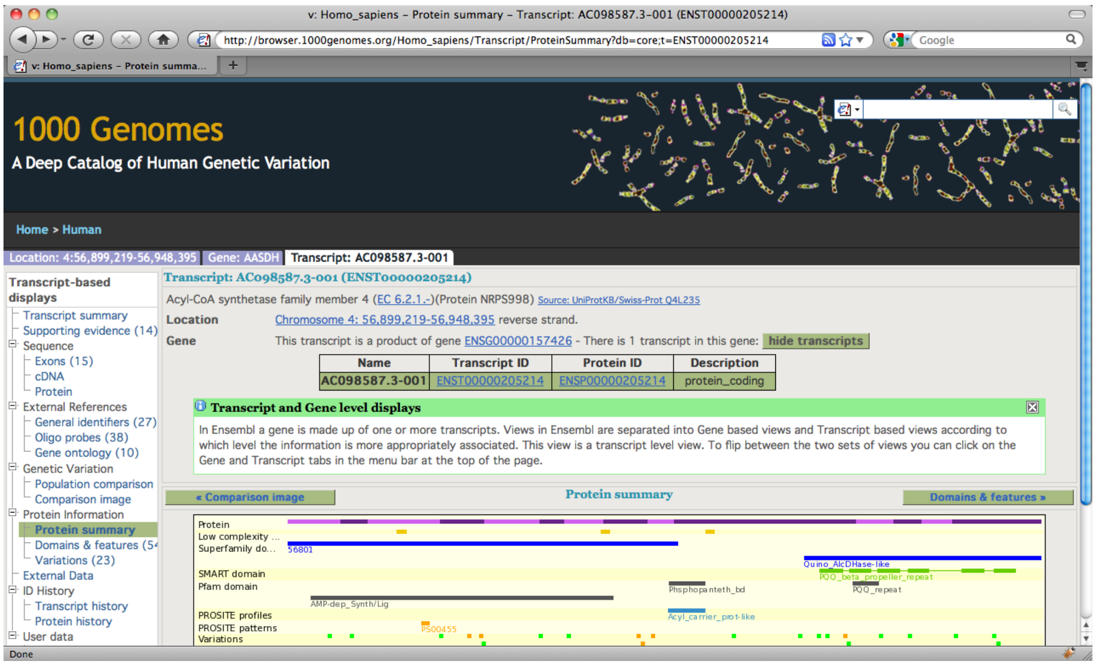
Search results for OligoProbe

0 entries matched your search strings.

Done

Figure 8 (taken from Ref.[27]).

In figure 8 we can see the first search result page which shows that the protein falls into the gene section of the browser.



Transcript: AC098587.3-001 (ENST00000205214)

Acyl-CoA synthetase family member 4 (EC 6.2.1.-)(Protein NRPS998) Source: UniProtKB/Swiss-Prot Q4L235

Location Chromosome 4: 56,899,219-56,948,395 reverse strand.

Gene This transcript is a product of gene ENSG00000157426 - There is 1 transcript in this gene: hide transcripts

Name	Transcript ID	Protein ID	Description
AC098587.3-001	ENST00000205214	ENSP00000205214	protein_coding

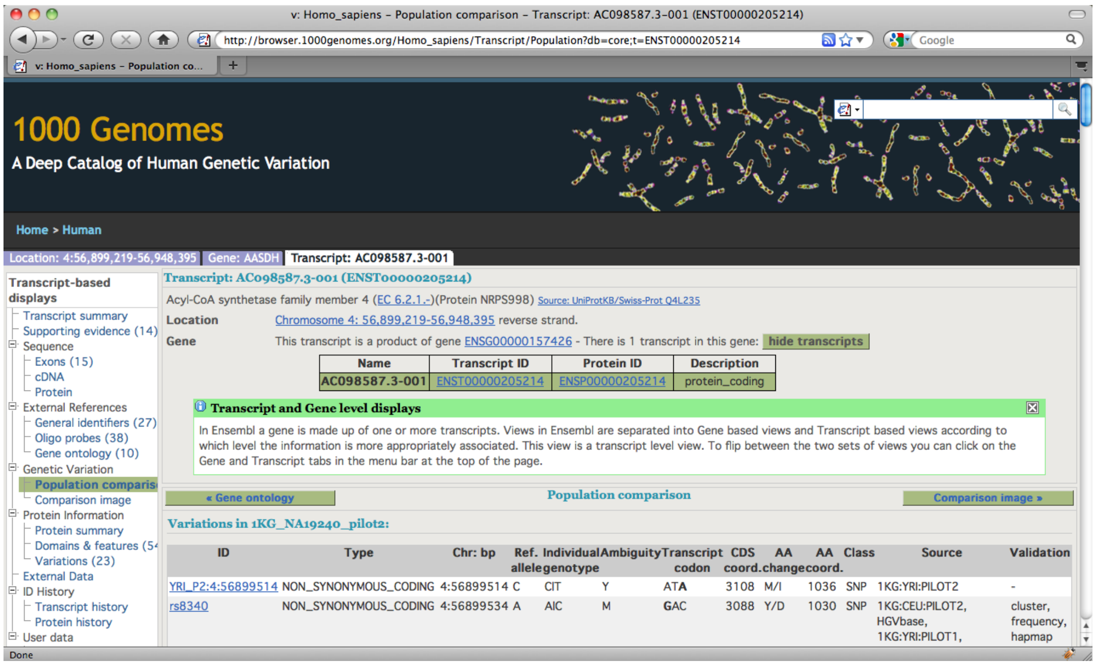
Transcript and Gene level displays

In Ensembl a gene is made up of one or more transcripts. Views in Ensembl are separated into Gene based views and Transcript based views according to which level the information is more appropriately associated. This view is a transcript level view. To flip between the two sets of views you can click on the Gene and Transcript tabs in the menu bar at the top of the page.

Protein summary

Figure 9 (taken from Ref.[27]).

Figure 9 shows the default page that loads after the previous search and presents the summary of the current protein.



Transcript: AC098587.3-001 (ENST00000205214)

Acyl-CoA synthetase family member 4 (EC 6.2.1.-)(Protein NRPS998) Source: UniProtKB/Swiss-Prot Q4L235

Location Chromosome 4: 56,899,219-56,948,395 reverse strand.

Gene This transcript is a product of gene ENSG00000157426 - There is 1 transcript in this gene: hide transcripts

Name	Transcript ID	Protein ID	Description
AC098587.3-001	ENST00000205214	ENSP00000205214	protein_coding

Transcript and Gene level displays

In Ensembl a gene is made up of one or more transcripts. Views in Ensembl are separated into Gene based views and Transcript based views according to which level the information is more appropriately associated. This view is a transcript level view. To flip between the two sets of views you can click on the Gene and Transcript tabs in the menu bar at the top of the page.

Population comparison

Variations in 1KG_NA19240_pilot2:

ID	Type	Chr: bp	Ref. allele	Individual genotype	Ambiguity	Transcript codon	CDS coord.	AA change	AA coord.	Class	Source	Validation
YRI_P2:4:56899514	NON_SYNONYMOUS_CODING	4:56899514	C	C|T	Y	ATA	3108	M/I	1036	SNP	1KG:YRI:PILOT2	-
rs8340	NON_SYNONYMOUS_CODING	4:56899534	A	A|C	M	GAC	3088	Y/D	1030	SNP	1KG:CEU:PILOT2, HGVbase, 1KG:YRI:PILOT1,	cluster, frequency, hapmap

Figure 10 (taken from Ref.[27]).

In figure 10 we see the page, after the population comparison option is clicked, with the generated tables that contain all the SNP data for the current protein.

The number of proteins for which the SNP information needed to be extracted was 46,591. This means that the time and the load of data would be extremely high. That is why there was the need to reduce, if possible, some of the steps required for the search to end up to the desired tables with the SNP data. As a first option was to use the last URL which had the desired information. The URL format of the last page (figure 10) that has the SNP data is:

http://browser.1000genomes.org/Homo_sapiens/Transcript/Population?db=core;t=ENST00000205214

The highlighted text is the transcript that corresponds to the ENSP00000205214 protein. A simple change to the last six characters of the URL could easily give us the targeted URL for each of the proteins. Although this would work for some of the proteins, not all proteins share the same number for both their ID name and their transcript. Thus, the extraction rule had to move a step backwards and to the previous page (figure 9). However the previous page had the same issue that is why the extraction had to start from the second page (figure 8) of the SNP browser each time.

The extraction rule first uses a URL link from the second page in order to navigate to the next one. The format of this URL is shown below:

http://browser.1000genomes.org/Homo_sapiens/protview?db=core;peptide=ENSP00000205214

The highlighted text shows the protein for which the SNP data will be extracted. All the proteins share the same URL format with the only change the highlighted ID name each time. Once this URL will be selected with a different ID protein name each time, the third page will be loaded.

The third page contains the protein summary and has also the “population comparison” image link which is going to be used in order to navigate to the forth and last page from where the SNP data will be extracted. A click event will be created to enable the “population comparison” image link. This was done with the next line of code, which uses the XPath:

```
browser.link(:xpath, ".//*[@id='local']/dd[5]/dl/dd[1]/a").click
```

The previous command describes a click event on the first item in a definition list, which is placed inside on the fifth item of a previous definition list. That is the place in the HTML code where the “population comparison” image link is.

Once the last page loads and after the JavaScript is executed, Watir retrieves the HTML code of the page with all the tables that contain the SNP data. The next figures show the HTML code before and after the JavaScript is executed:

```
<div class="content"><div class="ajax"
title="['/Homo_sapiens/Component/Transcript/Web/TranscriptSNPTable?db=core;t=ENST00000
205214;_rmd=7ec4']"></div><div class="ajax"
```

```
title="['/Homo_sapiens/Component/Transcript/Web/TranscriptSNPInfo?db=core;t=ENST000002
05214;_rmd=7ec4']"></div>
</div>
```

Figure 11 (taken from Ref.[27]).

The previous figure shows through the ajax class that the Ajax technology has been used in order to generate the tables. This is the HTML code that is available from all the browsers giving the structure of the page before the Javascript is loaded.

```
<div class=”content”><div class=”” title=””><p></p><h2>Variations in 1KG_NA19240_pilot2:
</h2><p>
</p><table class=”ss autocenter” style=”width: 100%; margin: 1em 0px;” cellpadding=”0”
cellspacing=”0”>
  <tbody><tr class=”ss_header”>
    <th class=”bottom-border left-border” style=”text-align: center; width: 0%;”>ID</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>Type</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>Chr: bp</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>Ref. allele</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>Individual genotype</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>Ambiguity</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>Transcript codon</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>CDS coord.</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>AA change</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>AA coord.</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>Class</th>
    <th class=”bottom-border” style=”text-align: center; width: 0%;”>Source</th>
    <th class=”bottom-border right-border” style=”text-align: center; width:0%;”>Validation</th>
  </tr>
  <tr style=”vertical-align: top;” class=”bg1”>
    <td class=” left-border”><a
href=”/Homo_sapiens/Variation/Summary?db=core;g=ENSG00000157426;r=4:56899219-
56948395;source=1KG:YRI%3APILOT2;t=ENST00000205214;v=YRI_P2:4%3A56899514;vf=
61489413”>YRI_P2:4:56899514</a></td>
    <td>NON_SYNONYMOUS_CODING</td>
    <td>4:56899514</td>
    <td>C</td>
    <td>C|T</td>
    <td>Y</td>
    <td>AT<b>A</b></td>
    <td>3108</td>
    <td>M/I</td>
    <td>1036</td>
    <td>SNP</td>
    <td>1KG:YRI:PILOT2</td>
    <td class=” right-border”>-</td>
  </tr>
```

Figure 12 (taken from Ref.[27]).

Figure 12 shows the table of one of the humans whose genome has been sequenced. It consists of several fields, including the ones that this project is interested in. Also, the first element in this table has been included which describes a non-synonymous mutation occurring in the 1036 amino acid position. This figure describes how the HTML code looks like after the JavaScript is executed and generally what Watir captures during the whole extraction process.

3.1.3 Parsing and Data Format

Once the Watir obtains the HTML code of the desired page, the Nokogiri with the XPath are used in order to parse the required data from the tables that contain the SNP data. During the parsing the needed data are saved to tab-delimited text files. The key commands that were used for the parsing are the following:

1. *page_html = Nokogiri::HTML.parse(browser.html)*
2. *parsed_element = page_html.xpath(".//*[@id='main']/div[2]/div[2]/div[1]/ table[#{i}]/tbody/tr[2]/td[2]").inner_text*

By using the first command, Nokogiri copies the content captured by Watir and based on the format of the second command it parses the SNP data. The second command is a simplified version of the actual commands used for the parsing part and it just saves the name of the first mutation. Afterwards, a check is performed to see if the name of the mutation corresponds to either a non-synonymous or a synonymous one, as this project is interested in only those two.

Only the SNPs with the name of “NON_SYNONYMOUS_CODING” or “SYNONYMOUS_CODING” in the field that describes the type of mutation are saved in the file. The format of the data can be seen in the next figure, which shows a very small sample of the data:

ENSP00000009606	NON_SYNONYMOUS_CODING	181	61	CGT	S/R
ENSP00000009606	NON_SYNONYMOUS_CODING	992	331	TTG	S/L
ENSP00000009606	SYNONYMOUS_CODING	702	234	AAT	N
ENSP00000010132	NON_SYNONYMOUS_CODING	181	61	ACG	A/T
ENSP00000010299	SYNONYMOUS_CODING	336	112	GCG	A
ENSP00000010338	NON_SYNONYMOUS_CODING	1057	353	GAA	Q/E
ENSP00000010338	NON_SYNONYMOUS_CODING	275	92	CAG	R/Q
ENSP00000010338	SYNONYMOUS_CODING	156	52	CAG	Q
ENSP00000010338	SYNONYMOUS_CODING	624	208	GCT	A
ENSP00000011292	SYNONYMOUS_CODING	321	107	TTT	F
ENSP00000011292	SYNONYMOUS_CODING	474	158	TAT	Y
ENSP00000011292	SYNONYMOUS_CODING	600	200	TAT	Y

Figure 13.

In the previous figure we can see five proteins with their SNP information. Some of the proteins have only non-synonymous or synonymous mutations and some others have both. In order to understand this data, it will first follow a brief description of each of the elements:

- The first element is the actual name of the protein. In the human proteins, for which this project is interested in, only the last six digits may change and the rest of the name remains the same. The range of the human proteins is from ENSP00000000233 to ENSP00000383878 and it has not a continues numbering.

- The next element describes the type of the SNP that the current protein has. In this project, as mentioned before, we examine two types of mutations, non-synonymous (missense) and synonymous (silent). Also, these SNPs have to occur at the coding regions of the human DNA in order to be of some significance and that is why the "coding" description at the end of each SNP.

- The third element of the data represents the CDS coordinates of the SNP. In general this can be considered as the position of the mutation in the whole human DNA sequence.

- The forth element describes the coordinates of the amino acid that is going to be changed. This number, always smaller than the previous one since it refers to a smaller area than the one that the CDS coordinates corresponds to, shows the position in the protein sequence in which the mutation occurs.

- The next element represents the new codon. After the SNP has occurred and one of the nucleotides in the old codon has changed a new codon is produced which may have the same or different chemical properties with the previous one. This in fact, plays a very important role for the produced amino acid, which is the next element in the data.

- The last element is the produced amino acid. As mentioned before, depending on the chemical properties of the new codon, the amino acid is either going to be changed or remain the same. For a non-synonymous mutation, as it is known from the theory and as it can be seen in the data, the amino acid changes to a different one. Moreover, a synonymous mutation does not cause any change to the current amino acid.

In this project, the first, the second, the fourth and the sixth elements are used for the mapping part. However, due to the vast amount of time required for the extraction, the other two elements had been included for the purpose of a future development.

Moreover, the extracted data consist of the SNP information of four different individuals. Thus, some of the mutations may be the same for some of the individuals. The next command that is used in the code removes any duplicate mutations from the file:

```
ruby -e 'puts STDIN.readlines.sort.uniq!.to_s' < dat.txt
```

3.1.4 Time factor

The most challenging part of the extraction was the time factor. Although Watir is capable of extracting dynamic content from a page, it can also be very slow. Every SNP search required two pages to load and also the JavaScript in the last one. All this process did not have a fixed execution time due to the unpredictable traffic and the current physical memory resources of the computer, which was doing the extraction. A way to deal with such an issue was to put timers inside the code and count the time needed for each execution to be completed. Afterwards, there was a check to see if that time was exceeding a specific time limit, which was 19 seconds. If it did, then the program would close in order to free the memory and start with a new traffic load. An average time for the extraction of the SNP information for one protein would be around 12 seconds, and if that number is multiplied by the total number of proteins, which is 46,591, it results in a total time of 559,092 seconds. This means that the extraction of all the data needed a full week to be completed.

3.1.5 Conclusion

Acquiring the data was the most essential part of this project because without the data the next two stages of the project, the mapping and the analysis, would not be possible. Since the data with all the SNP information were available only through the 1000 genome web site and through their browser all the previous steps had to be followed in order for the extraction to be successful. Although there were some technical difficulties, such as the dynamic content of the web site, the most important issue of all was the time despite the fact that some measures have been taken in order to reduce it.

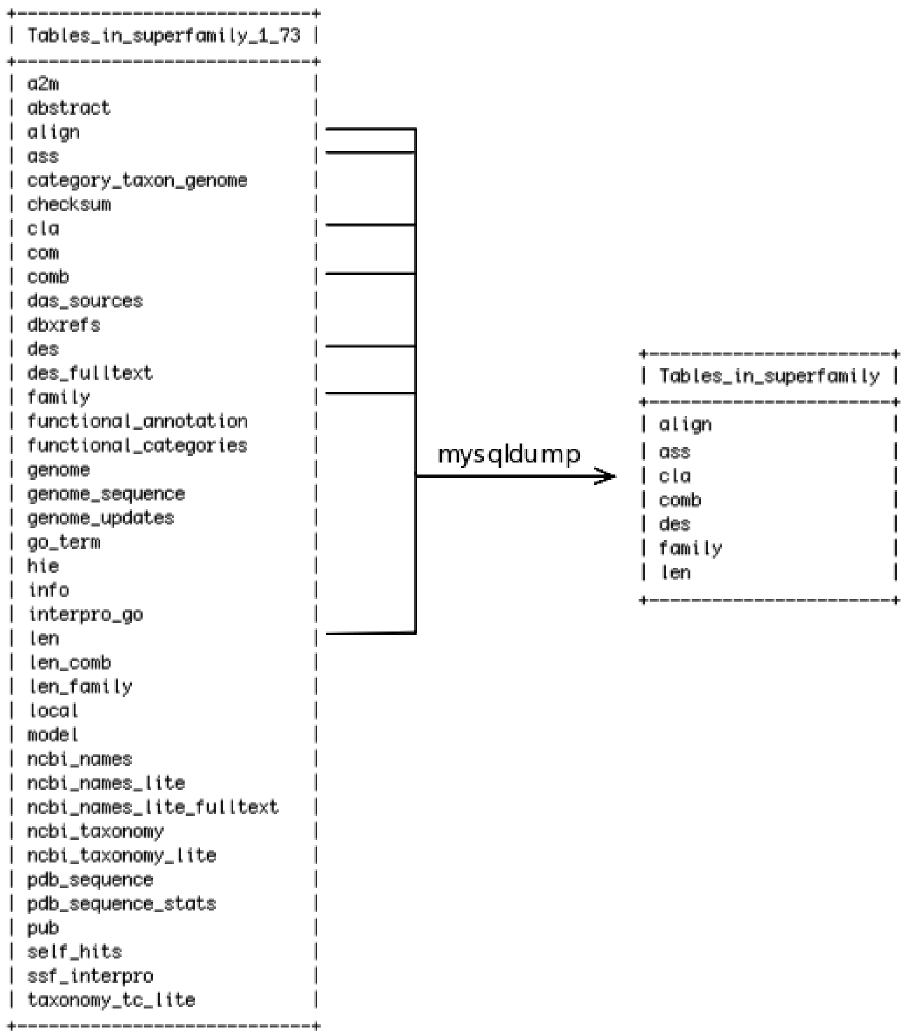
3.2 Data Mapping

After the extraction part was finished and all the SNP data were collected, they were then mapped to the superfamily database. The actual mappings were not made to the original database but to a copy of it. This database copy contained only the information needed for this research making the processes run faster and due to the large amount of data this was an efficient way to reduce the time required. There were two types of mappings that were made to the database. Firstly, the SNPs based on the amino acid position were mapped to the corresponding superfamily and to the specific region in which they belonged to. Moreover, by taking again the amino acid position and the amino acid change, the new amino acid was mapped to the corresponding protein sequence. For the whole mapping part two technologies were used, Java and MySQL. The code for the mappings was written in the Java programming language. Also, the JDBC module for MySQL was used in order for Java to interact with the MySQL database.

3.2.1 Database backup

A copy of the original database was made and stored locally in order to use it for the mapping and the analysis stages. This was implemented for two main reasons. Firstly, due to the large amount of tests that ware required in the mapping stage, the integrity of the data of the original database should have been ensured. Secondly, with this way, the time required for every process would have been significantly reduced. This is because the database copy would have included only a selection of tables and only a selection of the tables' data. Also, by storing and accessing the database locally helps avoiding the additional time required through a remote connection and for a possible heavy traffic and access to the original database. All these reasons could considerably detain the whole process due to the large amount of the data.

Furthermore, the mysqldump client was used to copy the tables and store them in the local database. The next figure shows the tables that have been used for the backup and loaded to the new database:



```
+----------------------------+
| Tables_in_superfamily_1_73 |
+----------------------------+
| a2m                        |
| abstract                   |
| align                      |
| ass                        |
| category_taxon_genome      |
| checksum                   |
| cla                        |
| com                        |
| comb                       |
| das_sources                |
| dbxrefs                    |
| des                        |
| des_fulltext               |
| family                     |
| functional_annotation      |
| functional_categories      |
| genome                     |
| genome_sequence            |
| genome_updates             |
| go_term                    |
| hie                        |
| info                       |
| interpro_go                |
| len                        |
| len_comb                   |
| len_family                 |
| local                      |
| model                      |
| ncbi_names                 |
| ncbi_names_lite            |
| ncbi_names_lite_fulltext   |
| ncbi_taxonomy              |
| ncbi_taxonomy_lite         |
| pdb_sequence               |
| pdb_sequence_stats         |
| pub                        |
| self_hits                  |
| ssf_interpro               |
| taxonomy_tc_lite           |
+----------------------------+
```

```
+------------------------+
| Tables_in_superfamily  |
+------------------------+
| align                  |
| ass                    |
| cla                    |
| comb                   |
| des                    |
| family                 |
| len                    |
+------------------------+
```

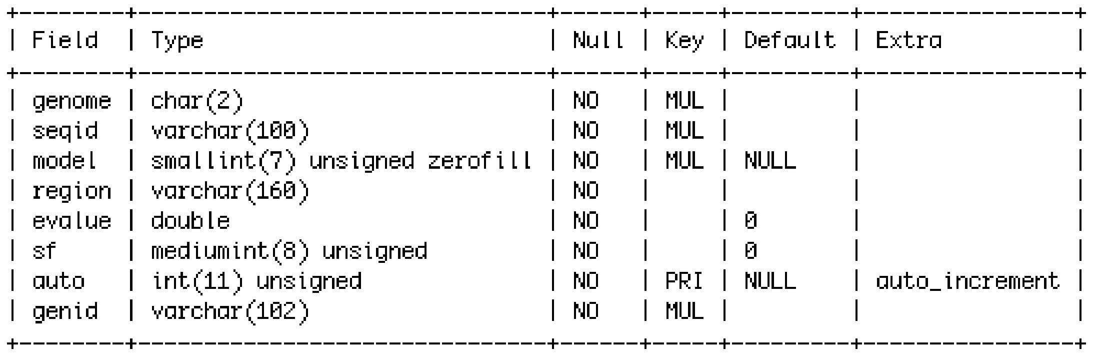
Figure 14 (Left table taken from Ref.[28]).

The format of the command that was used to backup the selected tables is given below:

mysqldump -u dp9493 -p -h supfam2.cs.bris.ac.uk superfamily_1_73 --where="genome = 'hs'" --single-transaction ass comb family align > backup.sql

From the previous command we can see that the current backup would consist of the ass, comb, family and align tables. The --where condition is set in order to select from the four tables only the data in which the genome field matches the human genome. Also the --single-transaction parameter is used to avoid putting locks to the database during the backup process and blocking other applications that use it.

In order to have a better understanding of the database structure it is necessary to have a brief description of the main tables, ass and align:



```
+--------+------------------------------+------+-----+---------+----------------+
| Field  | Type                         | Null | Key | Default | Extra          |
+--------+------------------------------+------+-----+---------+----------------+
| genome | char(2)                      | NO   | MUL |         |                |
| seqid  | varchar(100)                 | NO   | MUL |         |                |
| model  | smallint(7) unsigned zerofill | NO   | MUL | NULL    |                |
| region | varchar(160)                 | NO   |     |         |                |
| evalue | double                       | NO   |     | 0       |                |
| sf     | mediumint(8) unsigned        | NO   |     | 0       |                |
| auto   | int(11) unsigned             | NO   | PRI | NULL    | auto_increment |
| genid  | varchar(102)                 | NO   | MUL |         |                |
+--------+------------------------------+------+-----+---------+----------------+
```

Figure 15 (taken from Ref. [28]).

The previous figure shows the contents of the ass table, which has most of the proteins' information required for this research. The most important fields are:

- The genome field describes if the information refers to a human genome or to a different group. After the creation of the new database, this field will be related only to the human genome.

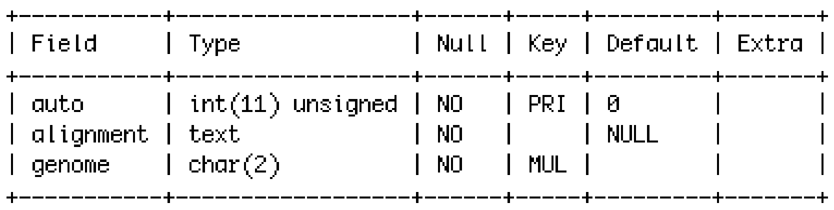
- The seqid field has all the human protein names. Every human protein name, as mentioned before, has the format of ENSP00000123456 with only the last six digits changing each time.

- In addition to the above, it is the region field. Each region is a different numeric range in which a protein falls into.

- Another field is the evalue field. An evalue represents in a way the error by which a region may fall into a superfamily.

- The sf field shows the name of the superfamily. A superfamily consists of several regions that share the same properties and have an evolutionary relationship.

- Another important field is the auto field. Every protein name and its corresponding region have a unique number as their identifier. This number is the auto number and it is used as a link value with the align table.



```
+-----------+------------------+------+-----+---------+-------+
| Field     | Type             | Null | Key | Default | Extra |
+-----------+------------------+------+-----+---------+-------+
| auto      | int(11) unsigned | NO   | PRI | 0       |       |
| alignment | text             | NO   |     | NULL    |       |
| genome    | char(2)          | NO   | MUL |         |       |
+-----------+------------------+------+-----+---------+-------+
```

Figure 16 (taken from Ref. [28]).

Figure 16 shows the align table which has three fields, two of which, the auto and the genome field, have been described previously in the ass table. The alignment field represents the protein sequence with all the amino acids from which a protein is consisted of.

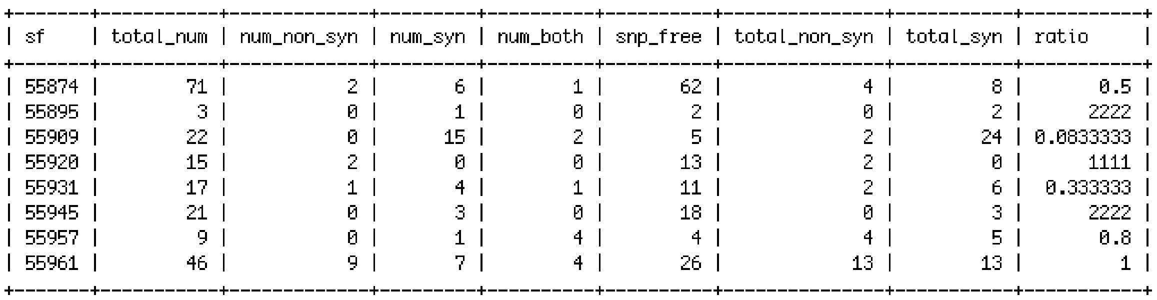
3.2.2 Mapping to the superfamily, region and protein sequence

Each SNP had to be mapped to the superfamily, to the region and to the protein sequence in which it belonged to. Every superfamily has a total number of non-synonymous mutations and a total number of synonymous ones. These numbers also provide a ratio of non-synonymous to synonymous mutations that is used later in the data analysis stage. Moreover, the regions of each superfamily have been separated into four categories depending on the mutations that fall into their range. There are regions that have only one of the two types of mutations or they have both. Also, there are regions that they do not have any mutations. Furthermore, each mutation is also mapped to the corresponding protein sequence. We will see that the last mapping only applies to the non-synonymous mutations because only they can cause an amino acid change.

The first task was to calculate the number of non-synonymous and synonymous mutations for every superfamily. Afterwards, there was the need to examine the regions of each superfamily. Thus, for every superfamily the total number of regions was divided into four categories, one for the regions that had only non-synonymous SNPs, a second one for the regions that had only synonymous SNPs, a third one for those regions having both types of SNPs and one more for the regions that were free of mutations.

For this specific mapping, the main concept was to use the position of the mutation and the name of the protein that has the current mutation in order to relate the SNP to the superfamily and to the region in which it occurred.

The following figure represents a small part of the main table (snps) that was used for the mappings and has most of the information that was required in the data analysis stage:



```
+-------+-----------+-------------+---------+----------+----------+---------------+-----------+-----------+
| sf    | total_num | num_non_syn | num_syn | num_both | snp_free | total_non_syn | total_syn | ratio     |
+-------+-----------+-------------+---------+----------+----------+---------------+-----------+-----------+
| 55874 |        71 |           2 |       6 |        1 |       62 |             4 |         8 |       0.5 |
| 55895 |         3 |           0 |       1 |        0 |        2 |             0 |         2 |      2222 |
| 55909 |        22 |           0 |      15 |        2 |        5 |             2 |        24 | 0.0833333 |
| 55920 |        15 |           2 |       0 |        0 |       13 |             2 |         0 |      1111 |
| 55931 |        17 |           1 |       4 |        1 |       11 |             2 |         6 |  0.333333 |
| 55945 |        21 |           0 |       3 |        0 |       18 |             0 |         3 |      2222 |
| 55957 |         9 |           0 |       1 |        4 |        4 |             4 |         5 |       0.8 |
| 55961 |        46 |           9 |       7 |        4 |       26 |            13 |        13 |         1 |
+-------+-----------+-------------+---------+----------+----------+---------------+-----------+-----------+
```

Figure 17.

A brief description is going to be given for each of the nine fields that can be distinguished in the previous table:

- The sf field, as mentioned before, stores the name of the superfamily, which is a six-digit number. The range of the superfamilies is from 46458 to 144292.

- The total_num field represents the total number of regions that a superfamily has. A superfamily may have one or more regions or even sometimes none.

- The num_non_syn field has the number of regions that have only non-synonymous mutations.

- The num_syn field has the number of regions that have only synonymous mutations.

- The num_both field has the number of regions in which it can be found both types of mutation.

- The next field is the snp_free field, which shows how many regions have no mutations at all.

- In the total_non_syn field it can be seen the total number of non-synonymous mutations that exist in the current superfamily.

- In addition to the previous field is the total_syn field, which stores the total number of synonymous mutations for a superfamily.

- The last field is the ratio field. Every value of this field refers to a non-synonymous to a synonymous mutations ratio. However the values of 1111 and 2222 are numeric flags. The 1111 value describes a superfamily that has only non-synonymous SNPs in contrast with the 2222 value, which is used when a superfamily has only synonymous mutations.

Moreover, by calculating the sum of the four categories of the regions and compare it with the total number of regions in each superfamily we can make a fast and accurate validation of the whole mapping procedure.

For the second task, which was to map each mutation to the corresponding protein sequence, the align table was used. Also a copy of the align table was created for having both the protein sequences, before and after the mapping. An example of a protein sequence from the align table is given below:

```
mdrkvarefrhkvdfliendaekdylydvlrmyhqtmdvavlvgdlklvinepsrlplfdairpliplkhqveydqltprrsr
klkevrldrlhpeglglsvrgglefgcglfishlikggqadsvglqvgdeivringysissctheevinlirtkktvsikvrhigli
pvksspdepltwqyvdqfvsesggvrgslgspgnrenkekkvfislvgsrglgcsissgpiqkpgifishvkpgslsaevgl
eigdqivevngvdfsnldhkeavnvlkssrsltisivaaagrelfmtdrerlaearqrelqrqellmqkrlamesnkilqeqqe
merqrrkeiaqkaaeeneryrkemeqiveeeekfkkqweedwgskeqlllpktitaevhpvplrkpksfgwfyrydgkf
ptirkkgkdkkkakygslqdlrknkkelefeqklykekeemlekekqlkinrlaqevseteredleesekiqywverlcqtrl
eqissadneisemttgppppppsvsplapplrrfagglhlhttdlddipldmfyyppktpsalpvmphpppsnpphkvpa
ppvlplsghvsassspwvqrtpppipippppsvptqdltptrplpsaleealsnhpfrtgdtgnpvedweaknhsgkptns
pvpeqsfpptpktfcpspqpprgpgvstiskpvmvhqepnfiyrpavksevlpqemlkrmvvyqtafrqdfrkyeEGF
DPYSMFTPEQIMGKDVRLLRIKKEGSLDLALEGGVDSPIGKVVVSAVYERGAAE
RHGGIVKGDEIMAINGKIVTDYTLAEAEAALQKAWNQGGDWIDLVVAVCPPKE
YDD-----elaslpssvaespqpvrklledraavhrhgfllqleptdlllkskrgnqihr
```

Figure 18.

The above sequence refers to the ENSP00000005226 protein. Every letter represents an amino acid, which is combined with all the others in order to create a protein structure. The actual sequence of the ENSP00000005226 protein is given by the lines with the capital letters and it is presented inside a small part of the general human sequence where it belongs to. For retrieving correctly this specific protein sequence, the protein's name and region were needed. These two values are necessary to find the unique identifier from the ass table and its auto field. Afterwards, the unique identifier is used to find the protein sequence from the align table and from its alignment field.

In order to get the sequence shown in the previous figure and to see the previous procedure through the MySQL syntax, the following commands must be used:

```
mysql> select auto from ass where region like '%735-847%' and seqid = 'ENSP00000005226';
```

This will give the unique identifier from the ass table, which in this example is 83858485.

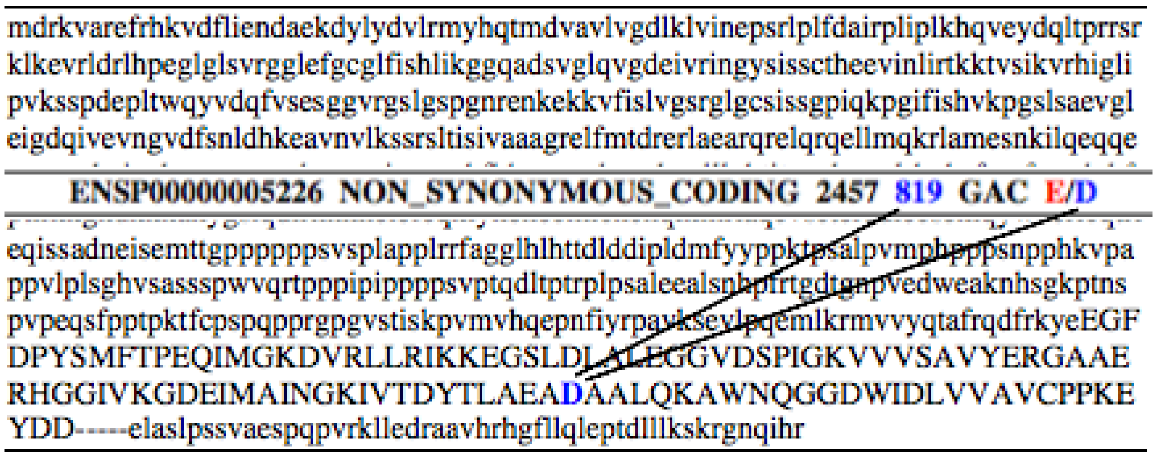
```
mysql> select alignment from align where auto ='83858485';
```

The last command gives the desired protein sequence.

The ENSP00000005226 protein from the previous example has one non-synonymous mutation at the position 819 that falls inside the specified region from 735 to 847. This information has been taken from the SNP data that were extracted at the first stage of the project:

ENSP00000005226 NON_SYNONYMOUS_CODING 2457 819 GAC E/D

As a non-synonymous mutation, it can cause a change in the amino acid that is located at the current position. The last element shows that the amino acid E has been change to D. In order to proceed to the mapping of the mutation to the protein sequence, the amino acid position and the amino acid change must be used. This becomes clearer in the next figure:



```
mdrkvarefrhkvdfliendaekdylydvlrmyhqtmdvavlvgdlklvinepsrlplfdairpliplkhqveydqltprrsr
klkevrldrlhpeglglsvrgglefgcglfishlikggqadsvglqvgdeivringysissctheevinlirtkktvsikvrhigli
pvksspdepltwqyvdqfvsesggvrgslgspgnrenkekkvfislvgsrglgcsissgpiqkpgifishvkpgslsaevgl
eigdqivevngvdfsnldhkeavnvlkssrsltisivaaagrelfmtdrerlaearqrelqrqellmqkrlamesnkilqeqqe
```

ENSP00000005226 NON_SYNONYMOUS_CODING 2457 819 GAC E/D

```
eqissadneisemttgpppppsvsplapplrrfagglhlhttdlddipldmfyyppktpsalpvmphpppsnpphkvpa
ppvlplsghvsassspwvqrtpppipippppsvptqdltptrplpsaleealsnhpfrtgdtgnpvedweaknhsgkptns
pvpeqsfpptpktfcpspqpprgpgvstiskpvmvhqepnfiyrpavkseylpqemlkrmvvyqtafrqdfrkyeEGF
DPYSMFTPEQIMGKDVRLLRIKKEGSLDLALEGGVDSPIGKVVVSAVYERGAAE
RHGGIVKGDEIMAINGKIVTDYTLAEADAALQKAWNQGGDWIDLVVAVCPPKE
YDD-----elaslpssvaespqpvrklledraavhrhgfllqleptdlllkskrgnqihr
```

Figure 19.

The previous figure is a combination of the SNP information and the sequence of the ENSP00000005226 protein. The number marked with blue is the amino acid position and points to where the change should occur. The letter marked with blue is the new amino acid, which also points to the position that the mutation occurs and with red is the changed amino acid.

As a conclusion to the above, knowing the SNP position and the protein's name, the specific region can be found. Also, the unique identifier can be retrieved since the protein's name and its region are known. The last step would be to find the protein sequence through the unique identifier and map there the mutation by combining the SNP position and the amino acid change information. In order to understand this better it must be seen through MySQL commands:

First the protein's regions must be found:

mysql> select region from ass where seqid = 'ENSP00000005226' and evalue <= 0.0001;

```
+---------+
| region  |
+---------+
| 66-191  |
| 735-847 |
| 209-289 |
+---------+
```

The above output shows that the protein has three regions. The mutation position of the previous example is 819, which falls into the range of the second region. Thus, the unique identifier can also be found:

mysql> select auto from ass where seqid = 'ENSP00000005226' and region like '%735-847%';

```
+----------+
| auto     |
+----------+
| 83858485 |
+----------+
```

This output gives the unique identifier and with that we can retrieve the protein sequence as it is shown in figure 18:

mysql> select alignment from align where auto = '83858485';

Afterwards the SNP position and the amino acid change will be used to successfully make the mapping in the specific protein sequence. A simple example of a key part of the Java application that is responsible for the mappings is given below:

```
.........
while ((text = rd.readLine()) != null){
.........
ResultSet r3 = s.executeQuery("select * from ass where genome = 'hs' and seqid =
'"+text+"' and evalue <= 0.0001");
while(r3.next()){
        String[] sf = r3.getString(4).split(",");
        for (String region : sf)  {
          String[] rg = region.split("-");
          reg1[i] = Integer.parseInt(rg[0]);
          reg2[i] = Integer.parseInt(rg[1]);
          i++;
        }
}
.........
if((x1 >= reg1[i]) && (x1 <= reg2[i])){
concat = reg1[i]+"-"+reg2[i];
.........
```

The above piece of code describes how the retrieved regions of the protein that is being read each time from the file, are stored in order for the mutation position to be checked if it falls inside of any of those regions.

3.2.3 Conclusion

In order to be able to develop and complete the mappings, except the programming skills that were required, also various biological concepts needed to be comprehended. This helped to understand the database structure and to decide which information should be used and how. Although the procedures and the mappings were checked for being valid and correct, a more efficient test, for such a large amount of data, would be through some specific biological theories that could be applied in the results of the next stage, which is the data analysis stage.

3.3 Data Analysis

The data analysis is the third and final stage of the project. Its main purpose and goal is to try and find a biological meaning in the previous mappings and more generally in the SNP data. It could also be considered as a validation tool for the two previous stages. Though out the data analysis we could see if there is a cohesion between the mutations, non-synonymous and synonymous ones, based on the biology theory and more specific on the theory concerning single-nucleotide polymorphisms. For this stage, as for the previous one, the Java programming language was used along with the JDBC module for interacting with the MySQL database. Afterwards, the results had been represented graphically by using the Grace plotting tool.

3.3.1 Processing the data

Once the mapping stage was finished, the SNP information was ready to be examined. Each time, the required data were retrieved from the database and from the snps table in order to be used for the current analysis. Moreover, statistical techniques were used like the normalization, the correlation and the null hypothesis with the analysis of covariance for transforming and modeling the data in a way such that the data analysis could be driven to a meaningful outcome.

Furthermore, two examples of the code that has been used for the data analysis stage is given below and describe two main functionalities of the whole process:

```
.........
ResultSet r2 = s1.executeQuery("select region from ass where sf = '"+r1.getInt(1)+"' and evalue <= 0.0001");
while(r2.next()){
        String[] sf = r2.getString(1).split(",");

        for (String region : sf)  {
          String[] rg = region.split("-");
          x = Integer.parseInt(rg[0]);
          y = Integer.parseInt(rg[1]);
          positions = (y - x) + 1;
          sum += positions;
        }
}
.........
```

The previous lines of code describe how the total number of positions is calculated each time for a superfamily. By knowing the length of the superfamilies we are able to normalize the distribution of the non-synonymous and synonymous SNPs for every superfamily.

```
.........
String[][] bl = new String[21][21];

        while ((blos = rd.readLine()) != null){
          String[] each = blos.split(",");

          for (j=0; j<21; j++){
                    bl[i][j] = each[j];
          }

          i++;
        }
.........
while ((change = rd1.readLine()) != null){
        String[] wrd = change.split("\t");

        if (wrd[1].equals("NON_SYNONYMOUS_CODING")){
          String[] alter2 = wrd[5].split("/");
          a1 = alter2[0];
          a2 = alter2[1];

          for (i=0; i<21; i++){
                    if (a1.equals(bl[i][0])){
                              flag1 = i;
                    }
          }

          for (j=0; j<21; j++){
                    if (a2.equals(bl[0][j])){
                              flag2 = j;
```

```
                }
        }
.........
```

The above code shows how the Blosum62 substitution matrix is integrated into the application. Moreover, in the second part, it can be seen the process by which each non-synonymous mutation is checked for its amino acid change. There is a comparison of the current change with the changes provided from the Blsum62 matrix in order to find the corresponding weight and to see how important is that change.

3.3.2 Graphical presentation

Every output of the data analysis stage is saved in a text file in order for the Grace to be able to process it and to visualize the results. The Grace has been used to create pie charts, bar charts and scatter plots. All the results are presented and explained in the next chapter.

4. Results and Biological Interpretation

4.1 General concentration of SNPs across superfamilies

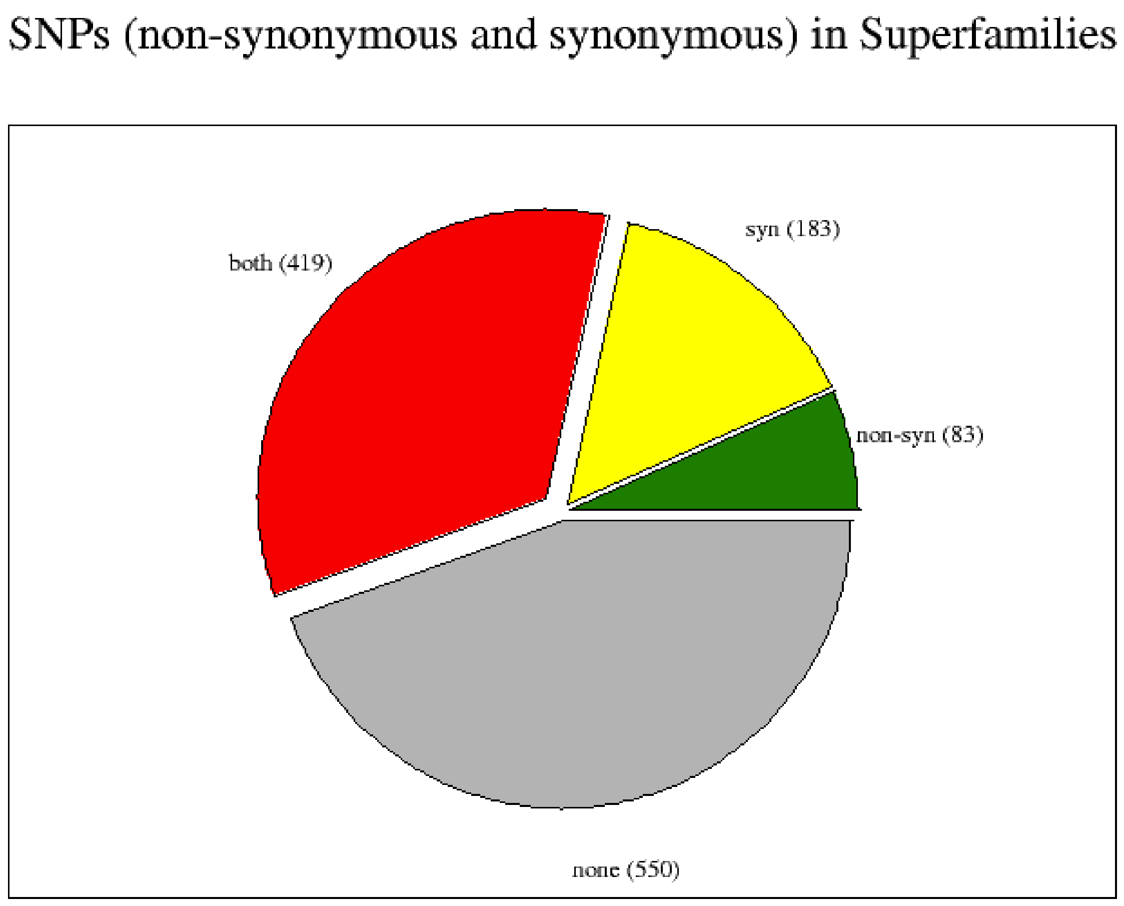


Figure 20.

The pie chart in the previous figure shows the general concentration and proportion of SNPs in the superfamilies. Furthrmore, the red sector represents those superfamilies that have both non-synonymous and synonymous SNPs, the yellow sector represents the superfamilies that have only synonymous mutations, the green sector represents the superfamilies that have only non-synonymous SNPs and the grey sector describes the superfamilies that have neither non-synonymous nor synonymous SNPs. The main reason that some superfamilies have not got a SNP is due to their small size. For example, a superfamily may have just one region, which can be to small in order to be possible for a mutation to occur. Also, some of the superfamilies that are shown having only non-synonymous or synonymous mutations are due to the fact that these also may have small sizes and may have just one non-synonymous or synonymous mutation, which is not of such significance.

4.2 Distribution of SNPs across superfamilies

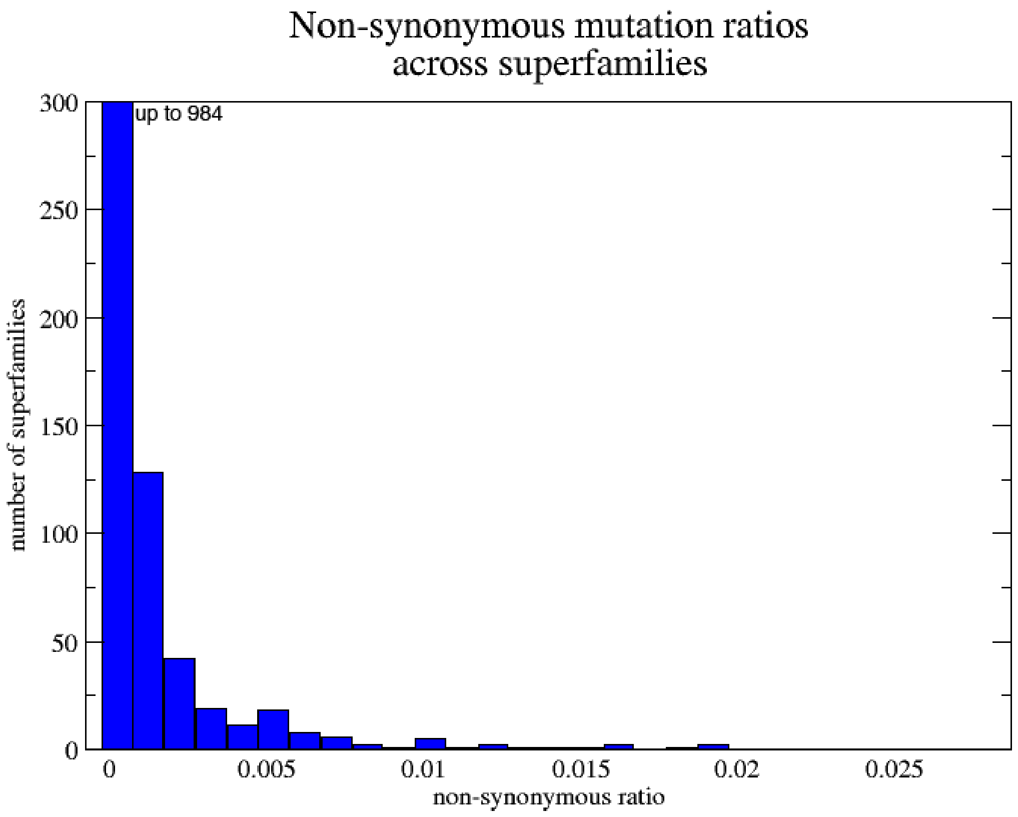


Figure 21.

In the previous figure we can see the distribution of non-synonymous SNPs across superfamilies. All the distribution has been normalized for the size of each of the superfamilies. The pie chart shows the number of superfsmilies that fall into a specific range of the non-synonymous ratio. Most of the superfamilies fall into the 0 to 0.001 and 0.001 to 0.002 ratio range. This in fact means that a large number of superfamiles seem to favour a low concentration of non-synonymous SNPs.

The next figure shows the distribution of synonymous SNPs across superfamilies. It is obvious that the superfamilies tend to have a slightly bigger ratio of synonymous mutations in comparison with figure 21 and the non-synonymous ratio. Moreover, this difference in the two distributions could be seen as a sign of natural selection and evolution, although these concepts will be more clear and explained better in the two last graphs of this chapter.

In order to have a more clear view of the ratio axes in both bar charts, the ranges at the X an Y axes have been reduced. That is why there is an explanation of the maximum value for the 0 to 0.001 ratio scales.

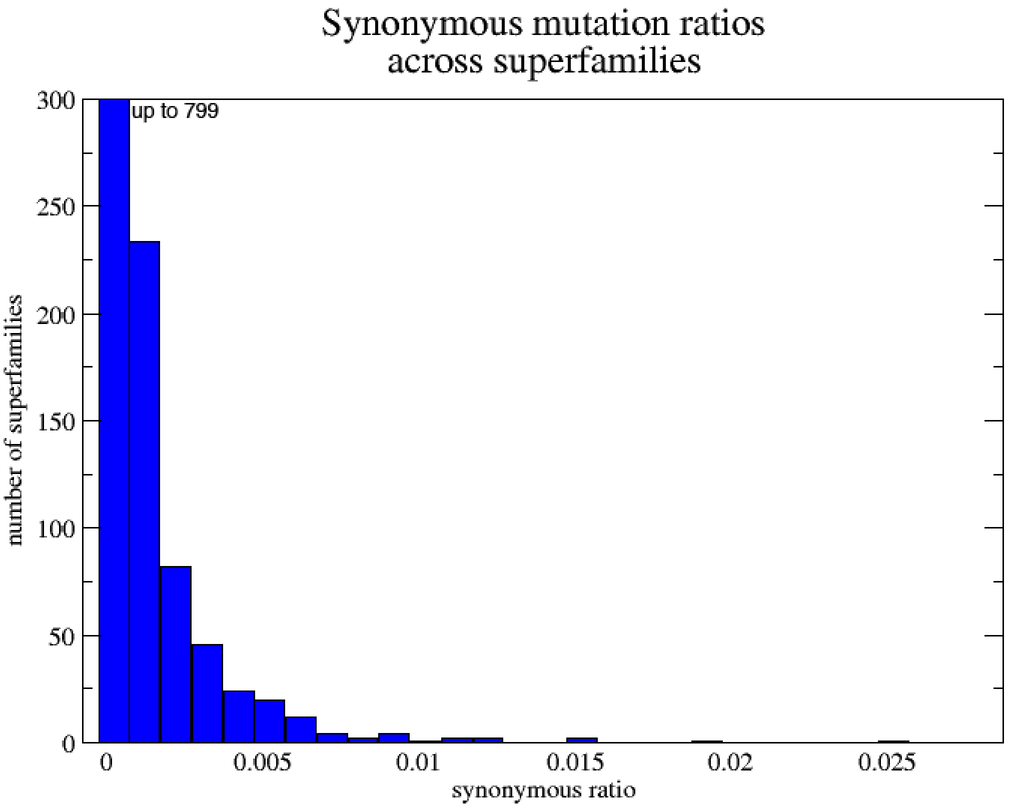


Figure 22.

4.3 Non-synonymous to synonymous SNP ratio

The non-synonymous to synonymous ratio, as it will be discussed in more details in the last graph, shows the type of natural selection that a superfamily has. The ratio from each superfamily could be compared with the neutral ratio in order for the natural selection and the evolution to be examined. The neutral ratio of non-synonymous SNPs to synonymous SNPs is approximately a 2:1 ratio. For example a superfamily that has two non-synonymous mutations and one synonymous mutation could be characterized as a non-evolving one. The superfamilies, which have a ratio smaller than the neutral ratio, are those that have passed through the purifying selection and have been evolved. Also, a very small number of superfamilies might have a ratio that could be greater than the neutral ratio, which means that not only there is no selective removal of deleterious alleles but also an increase of them.

The next bar chart shows the non-synonymous to synonymous SNP ratio across the superfamilies. It is shown that most of the superfamilies have a ratio smaller than the neutral ratio and only a few ones have a greater one:

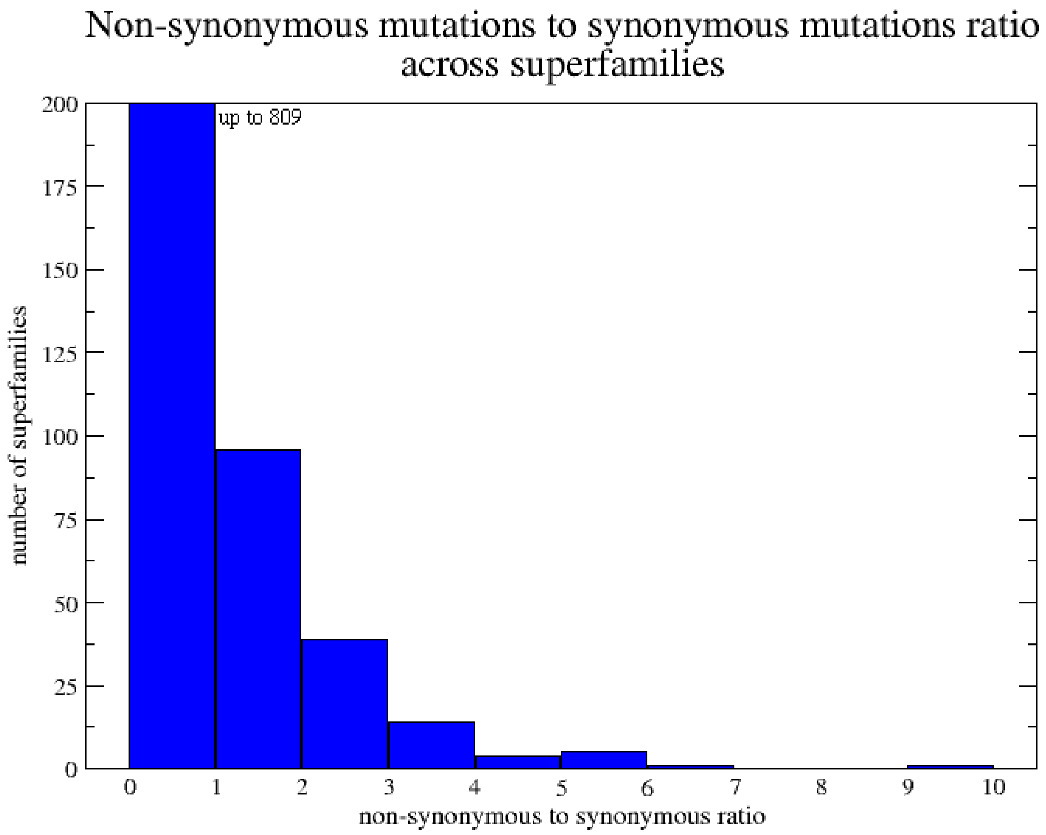


Figure 23.

4.4 Amino acid change weighted for each superfamily

As it is mentioned before, the non-synonymous SNPs are responsible for the amino acid change and thus for the alteration of the protein's structure. In order to investigate the importance of that type of mutation and its general impact to the human body, there must be first an examination in the amino acid change in each superfamily. Every superfamily, which has at least one non-synonymous SNP, is checked to see how important is the change that occurs in a specific amino acid. Afterwards, all the weighted amino acid changes for each superfamily are summed up and normalized for the number of members (proteins) that the non-synonymous mutations occur. For example, if a superfamily has five members and the non-synonymous mutations occur only in one member then the general amino acid weight of that superfamily will be just the sum of its amino acid weights. However, if the non synonymous SNPs occur in three of the members then a normalization will follow by dividing the sum of the amino acid weights with the number of the members in which the mutations occurred.

For this task the substitution matrix from figure 2 was used. The Blosum62 was altered in order to fit in a positive weight scale from 1 to 8. In all the amino acid changes that result in the same amino acid, the zero value was given. All the rest of the changes had the value of -4 subtracted from their original ones. The fixed Blosum62 is given below:

	C	S	T	P	A	G	N	D	E	Q	H	R	K	M	I	L	V	F	Y	W
C	**0**	5	5	7	4	7	7	7	8	7	7	7	7	5	5	5	5	6	6	6
S	5	**0**	3	5	3	4	3	4	4	4	5	5	4	5	6	6	6	6	6	7
T	5	3	**0**	3	5	3	4	3	4	4	4	5	4	5	6	6	6	6	6	7
P	7	5	3	**0**	5	6	5	5	5	5	6	6	5	6	7	7	6	8	7	8
A	4	3	5	5	**0**	4	5	6	5	5	6	5	5	5	5	5	6	6	6	7
G	7	4	3	6	4	**0**	6	5	6	6	6	6	6	7	8	8	4	7	7	6
N	7	3	4	6	6	4	**0**	3	4	4	5	4	4	6	7	7	7	7	5	8
D	7	4	3	5	6	5	3	**0**	2	4	5	6	5	7	7	8	7	7	7	8
E	8	4	4	5	5	6	4	2	**0**	2	4	4	3	6	7	7	7	7	6	7
Q	7	4	4	5	5	6	4	4	2	**0**	4	3	3	4	7	6	6	7	5	6
H	7	5	4	6	6	6	3	3	4	4	**0**	4	5	6	7	7	6	5	2	6
R	7	5	5	6	5	6	4	6	4	3	4	**0**	2	5	7	6	7	7	6	7
K	7	4	4	5	5	6	4	5	3	3	5	2	**0**	5	7	6	7	7	6	7
M	5	5	5	6	5	7	6	7	6	4	6	5	5	**0**	3	2	6	4	5	5
I	5	6	6	7	5	8	7	7	7	7	7	7	7	3	**0**	2	3	4	5	7
L	5	6	6	7	5	8	7	8	7	6	7	6	6	2	2	**0**	1	4	5	6
V	5	6	6	6	4	7	7	7	6	6	7	7	6	3	1	3	**0**	5	5	7
F	6	6	6	8	6	7	7	7	7	7	5	7	7	4	4	4	5	**0**	1	3
Y	6	6	6	7	6	7	6	7	6	5	2	6	6	5	5	5	5	1	**0**	2
W	6	7	7	8	7	6	8	8	7	6	6	7	7	5	7	6	7	3	2	**0**

Figure 24.

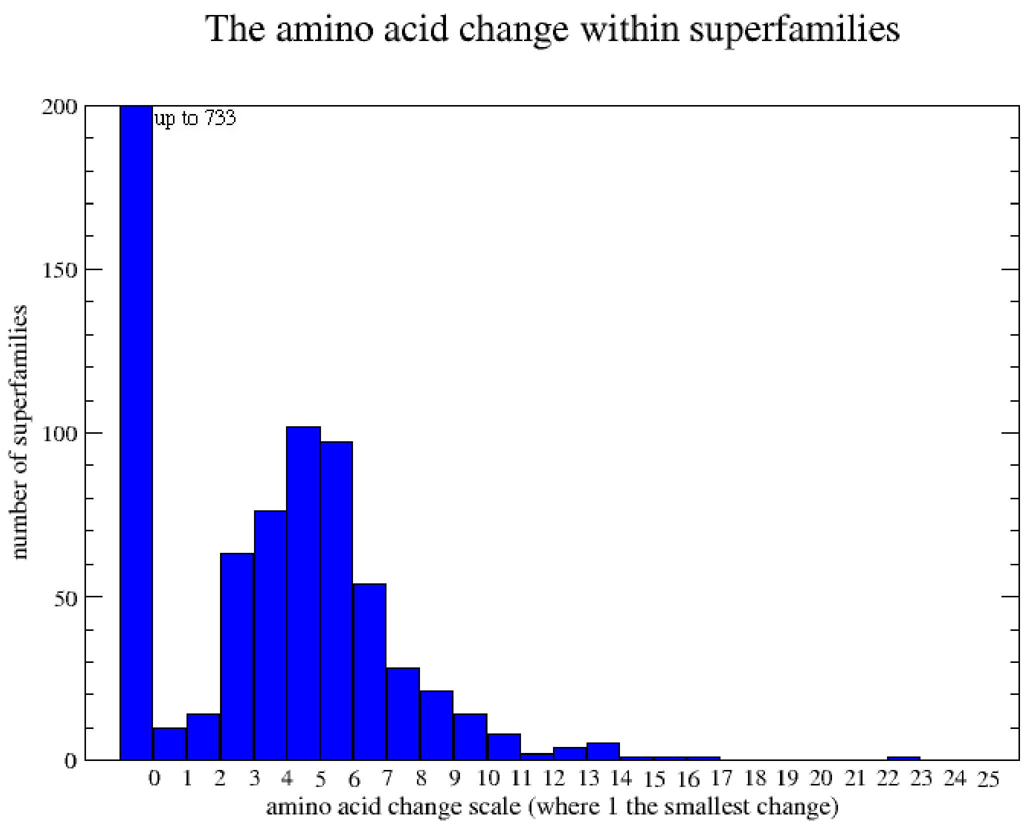
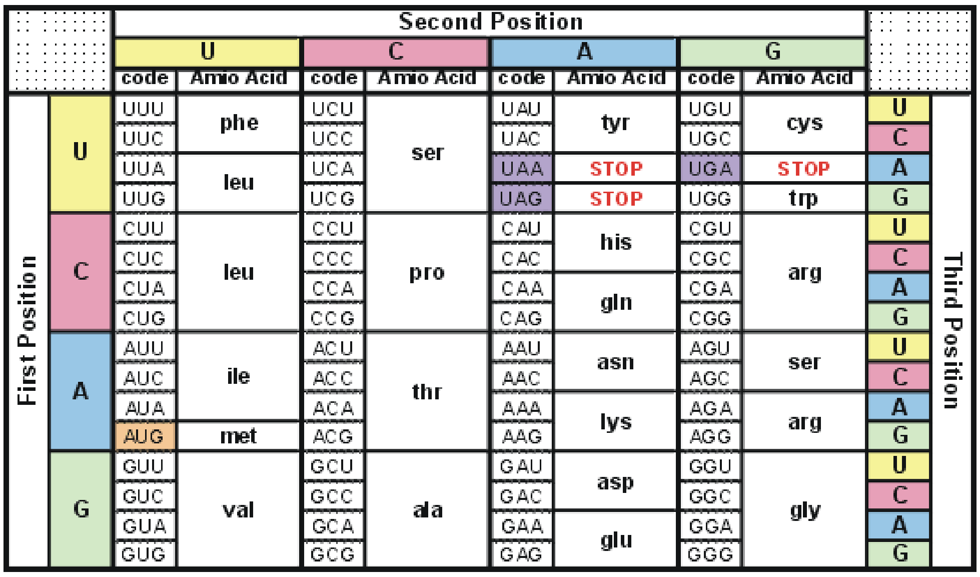


Figure 25.

In figure 25, we can see the general amino acid change weight of each of the superfamilies. The Y-axis represents the number of superfamilies that share the same weight and the X-axis shows the amino acid change scale. In this specific scale, the smaller the value of the weight the less important is the amino acid change. As it can be seen from the bar chart, most of the superfamilies tend to keep a balance as concerning their general amino acid change weight. A very large number of the superfamilies have a weight of 5 to 6, which means that the non-synonymous mutations that occur in these superfamilies seem to be in balance. Having a general amino acid change weight of 5 or 6, means that all the amino acids, which have changed to a different one, as an average conclusion, they are not significantly important in order to cause a serious alteration to the human body, although, they do have a valuable degree of participation in the evolution process.

4.5 Examination of natural selection

In order to examine the natural selection through the SNP data and the mappings, there must be first a comparison between the average ratio of non-synonymous to synonymous SNPs of the calculated data with the theoretical neutral ratio. The neutral ratio of non-synonymous to synonymous mutations can be found and calculated from the table shown in the next figure:



		Second Position								
		U		C		A		G		
		code	Amio Acid	code	Amio Acid	code	Amio Acid	code	Amio Acid	
First Position	U	UUU	phe	UCU	ser	UAU	tyr	UGU	cys	U
		UUC		UCC		UAC		UGC		C
		UUA	leu	UCA		UAA	STOP	UGA	STOP	A
		UUG		UCG		UAG	STOP	UGG	trp	G
	C	CUU	leu	CCU	pro	CAU	his	CGU	arg	U
		CUC		CCC		CAC		CGC		C
		CUA		CCA		CAA	gln	CGA		A
		CUG		CCG		CAG		CGG		G
	A	AUU	ile	ACU	thr	AAU	asn	AGU	ser	U
		AUC		ACC		AAC		AGC		C
		AUA		ACA		AAA	lys	AGA	arg	A
		AUG	met	ACG		AAG		AGG		G
	G	GUU	val	GCU	ala	GAU	asp	GGU	gly	U
		GUC		GCC		GAC		GGC		C
		GUA		GCA		GAA	glu	GGA		A
		GUG		GCG		GAG		GGG		G Third Position

Figure 26 (taken from Ref. [30]).

The previous figure describes how the amino acid changes when the first, the second and the third element of a codon are changed. Furthermore, when a codon changes in its first or second position, this results in a change of the current amino acid. However, if the codon changes in its third position there is a very high possibility that the current amino acid will remain the same. Thus, a 2:1 non-synonymous to synonymous mutations ratio can be considered as a neutral ratio by which there is no evolution. The next figure presents the calculated average non-synonymous to synonymous ratio in comparison with the neutral 2:1 ratio:

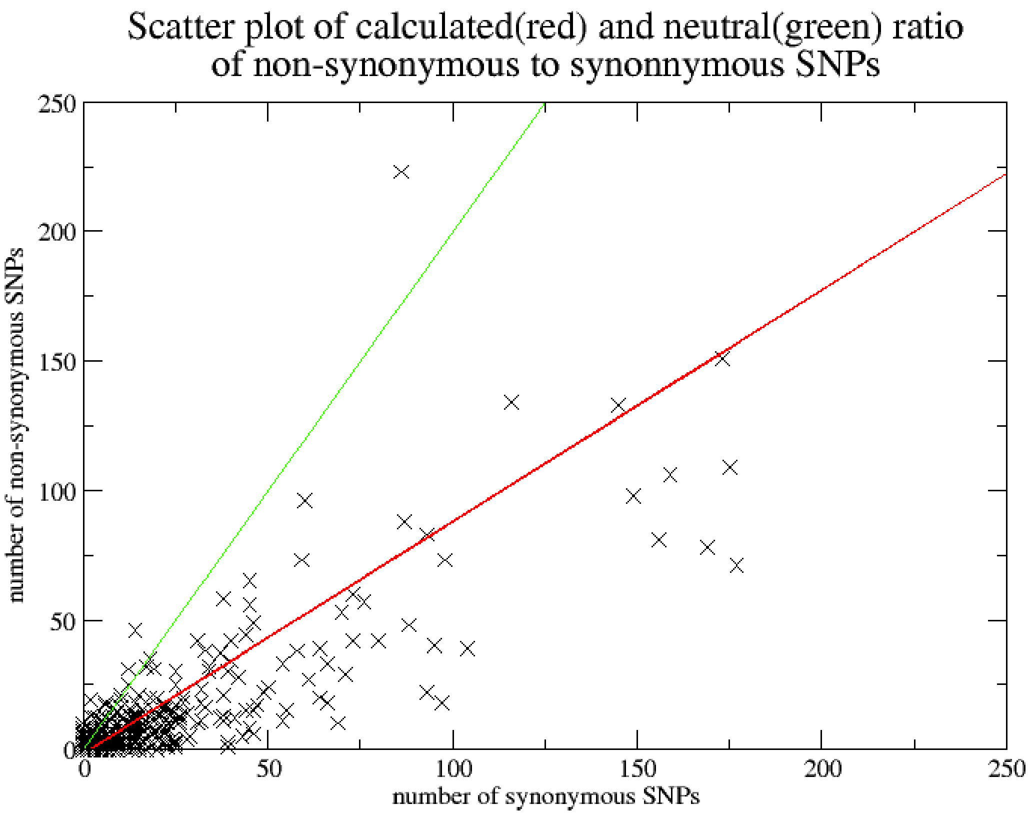


Figure 27.

The previous figure shows a scatter plot in which there are two ratio lines. The green ratio line represents the neutral ratio and the red one the average ratio of the calculated data. Every point in the graph describes the non-synonymous to synonymous SNPs ratio of each superfamily. A simple observation of the graph shows that the average ratio is smaller than the neutral ratio. This means that as concerning the natural selection, these data shows that a purifying (negative) selection took place. Due to the selective removal of deleterious alleles, there is a decrease in the number of non-synonymous SNPs and thus a decrease in the non-synonymous to synonymous mutations ratio. The superfamily ratios, which are below the neutral ratio line are those that are evolving, Those ratios that are under the green line and close to it are representing the superfamilies that are fast evolving in contrast with the ones that are closer to the X-axis, which are slow evolving. Also, the null hypothesis and the analysis of covariance have been examined for these two regression lines in order to see if there is a significant difference between them. The two sources [24], [31] which have been used for this examination, showed that these two regression lines are significantly different with each other. Their P-value was below the 0.05 reference limit, which shows if there is an actual difference.

5.Conclusion

The main aim of this project was to make an analysis of two certain types of mutations that occur in the human DNA sequence. The non-synonymous and synonymous SNPs where collected and then combined with the existed protein information in order for a further examination of their biological importance. Despite the main difficulty, which was the time factor, especially in the first stage of the data extraction, this project managed to fulfill its purpose. After the first stage, which was the extraction of the data from a web page that it was based on dynamic content, and the second stage, in which the extracted data were mapped in the local database, it was the analysis stage, which was the last one. The data analysis stage, except its role of the interpretation of the data and the mappings, it was also used as a tool in order to validate the previous two stages. Since the results correspond to certain biological theories, then the whole procedure before the data analysis should be to some degree valid and correct. The results showed that the natural selection had taken place and more specific the purifying (negative) selection, which allowed to the individuals to evolve. Moreover, the SNPs seem to be in balance and especially the non-synonymous ones. It is very important for the non-synonymous SNPs to be normally distributed and not to cause a serious alteration to the corresponding amino acids. On the whole, this project is considered to be the first step of a bigger research, which will try to find answers in the relation of the single-nucleotide polymorphisms (SNPs) with certain types of diseases.

6. Future Work

Due to the fact that this project is the first step of a further research of SNPs that occur in the human DNA sequence, it can be extended and continued in a variety of ways. In this chapter, two possible extensions of the current project are going to be discussed and proposed.

6.1 Mapping to the 3D protein structure

Since all the SNP data and more specific the mutations' positions have been collected, a very useful work would be to map those mutations to the 3D structure of the corresponding protein. An example, of a 3D protein structure is given below:

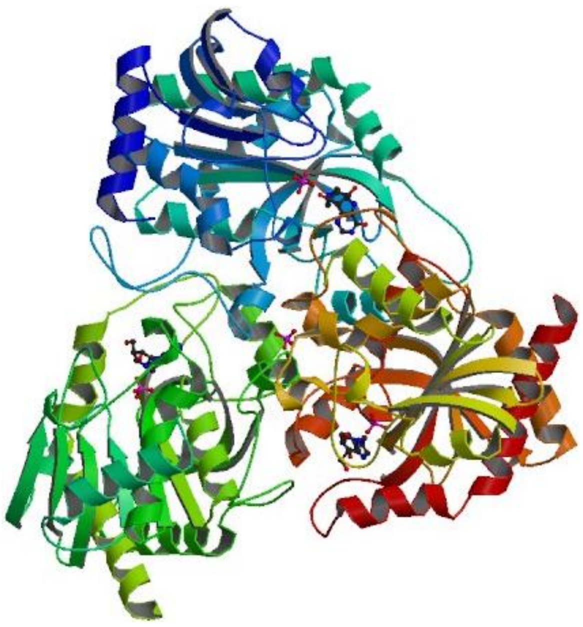


Figure 28 (taken from Ref. [29]).

In order for the SNPs to be mapped to a 3D structure the mutation amino acid position must be used along with the 3D coordinates, which are available in pdb files. The pdb (protein data bank) files can be retrieved from the RCSB PDB web site [29]. The next figure shows a part of a pdb file and how the positions in the protein sequence are related to the 3D coordinates:

```
ATOM    998  OD1 ASP A 219     25.535  -4.476  91.355  1.00 21.02          O
ATOM    999  OD2 ASP A 219     26.096  -5.601  89.555  1.00 19.56          O
ATOM   1000  N    HIS  A 220     27.635  -3.586  93.088  1.00 12.96          N
```

Figure 29.

The second element in the previous figure represents the position in the current protein sequence and the eighth, the ninth and the tenth elements are the 3D coordinates of that position. Thus, the mutation position can be related with the specific 3D coordinates. Having the mutations in the 3D structure of the protein, an analysis of the distance between the SNPs could reveal even more information about their importance and their general function.

6.2 Analysis based on same disease type

A very interesting direction of the analysis would be to examine a group of individuals that share the same type of disease. For example, the data from a group of ten persons that have the same type of cancer could be investigated in order to find trends, which could be used to understand new possible aspects of the current disease that have not yet been discovered. The first task would be to find similar SNPs between all the individuals and compare the superfamilies' ratios. If all of the ten persons share the same ratio and that ratio describes a fast evolving superfamily or even a superfamily that is placed above the neutral ratio, then that particular superfamily should be examined further. The second task could include the use of the Blosum62 subsitution matrix by which the general amino acid change weight of the specific superfamily could be checked. If the general weight is a very high number then a more advanced research of the current superfamily should follow in order to see what are its general function and its role to the human body.

Bibliography

[1] Novotny R., Vojtas P., Maruscak D., Information Extraction from Web pages, IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology – Workshops, 2009.

[2] Quafafou M., Jarir Z., Erradi M., Services orchestration for web information extraction, Proceedings of the Third International Conference on Next Generation Web Services Practices, pp. 85-89, IEEE Computer Society Washington, DC, USA, 2007.

[3] Lam M. I., Gong Z., Web Information Extraction, Proceedings of the IEEE International Conference on Information Acquisition, Hong Kong and Macau, China, 2005.

[4] Habegger B., Quafafou M., Web Services for Information Extraction from the Web, Proceedings of the IEEE International Conference on Web Services, IEEE Computer Society Washington, DC, USA, pp. 279-286, 2004.

[5] Xia T., Extracting Structured Data from Ajax Site, Proceedings of the First International Workshop on Database Technology and Applications, pp. 259-262, IEEE Computer Society Washington, DC, USA 2009.

[6] Mesbah A., Bozdag E., Deursen V. A., Crawling AJAX by Inferring User Interface State Changes, Eighth International Conference on Web Engineering, pp. 122-134, IEEE Computer Society Washington, DC, USA, 2008.

[7] Web Scraping, [online], Available from: http://en.wikipedia.org/wiki/Web_scraping

[8] Alexe B., Chiticariu L., Miller J. R., Tan W. C., Muse: Mapping Understanding and deSign by Example, Proceedings of the ACM SIGMOD international conference on Management of data, pp. 1281-1284, ACM New York, NY, USA, 2008.

[9] Chiticariu L., Tan W. C., Debugging Schema Mappings with Routes, Proceedings of the 32nd international conference on Very large data bases, pp. 79-90, VLDB Endowment, 2006.

[10] Yan L. L., Miller J. R., Haas M. L., Fagin R., Data-Driven Understanding and Refinement of Schema Mappings, Proceedings of the ACM SIGMOD international conference on Management of data, pp. 485-496, ACM New York, NY, USA, 2001.

[11] Lewis-Beck M. S., Data Analysis: An Introduction, Sage University Paper series on Quantitative Application in the Social Sciences, pp. 2-22, Thousand Oaks, CA: Sage, 1995.

[12] Nolan B., Data Analysis An Introduction, pp. 147-152, Polity Press, 1994.

[13] Data Analysis, [online], Available from: http://en.wikipedia.org/wiki/Data_analysis

[14] Normalization, [online], Available from: http://en.wikipedia.org/wiki/Normalization

[15] Grace tool, [online], Available from: http://plasma-gate.weizmann.ac.il/Grace

[16] Elmasri R., Navathe S. B., Fundamentals of Database Systems, 5th edition, pp. 1022-1034, Pearson International, Addison Wesley, 2007.

[17] Cavallo A., Martin A. C. R., Mapping SNPs to protein sequence and structure data, Boinformatics, Vol. 21, pp. 1443-1450, Oxford University Press, 2004.

[18] Ramensky V., Bork P., Sunyaev S., Human non-Synonymous SNPs: server and survey, Nucleic Acids Research, Vol.30, No 17, Oxford University Press pp. 3894-3900, 2002.

[19] Collins F. S., Brooks L. D., Chakravarti A., A DNA Polymorphism Discovery Resource for Research on Human Genetic Variation, Genome Res., pp. 1229-1231, Gold Spring Harbor Laboratory Research, 1998.

[20] Komar A. A., Single Nucleotide Polymorphisms Methods and Protocols, 2nd Edition, Humana Press, 2009.

[21] Genetic Variation (SNPs), [online], Available from: http://nci.nih.gov/cancertopics/understandingcancer/geneticvariation

[22] SNP Fact Sheet, [online], Available from: http://www.ornl.gov/sci/techresources/Human_Genome/faq/snps.shtml

[23] MySQL Backup and Recovery, Reference Manual, Oracle, 2010.

[24] Handbook of Biological Statistics, Analysis of covariance, [online], Available from: http://udel.edu/~mcdonald/statancova.html

[25] Watir, [online], Available from: http://watir.com/documentation/

[26] Nokogiri, [online], Available from: http://nokogiri.org/tutorials

[27] 1000 genome project browser, [online], Available from: http://browser.1000genomes.org/index.html

[28] Superfamily database, [online], Available from: http://supfam.cs.bris.ac.uk/SUPERFAMILY/

[29] RCSB protein data bank, [online], Available from: http://www.pdb.org/pdb/home/home.do

[30] Amino acid codon table, [online], Available from: http://citnews.unl.edu/hscroptechnology/html/printLesson.shtml?lessonID=990631834

[31] Statistical Tools, Comparing two regression lines, [online], Available from: http://amchang.net/StatTools/Comp2Regs_Exp.php

Appendix A: Source

extract.rb

```
require 'watir'
require 'nokogiri'

#calling firefox for new browser window
Watir::Browser.default = "firefox"
b = Watir::Browser.new
b.speed = :fast

total = 0

#opening file which has all the human proteins
prot = File.open(ARGV[0], "r+")

#checking for end of file and execution time limit
while ((line = prot.gets) && (total < 19.0))

        start = Time.now
        z = line.chomp
        #first page to load
        b.goto("http://browser.1000genomes.org/Homo_sapiens/protview?db=core;peptide=#{line}")

        wr = Nokogiri::HTML.parse(b.html)
        w = wr.xpath(".//*[@id='tab_location']/a").inner_text

          if (w != "Genome")

                #clicking element to load the desired page and watir captures the html code after the javascript
                has been executed
                b.link(:xpath, ".//*[@id='local']/dd[5]/dl/dd[1]/a").click

                #wait for the page to load
                sleep 4

                # calling nokogiri to parse the html code which has been captured by watir
                page_html = Nokogiri::HTML.parse(b.html)

                        for i in 1..10
                                for j in 2..20

#parsing the, mutation ID, CDS position, amino acid position, changed codon and changed amino amino acid
x = page_html.xpath(".//*[@id='main']/div[2]/div[2]/div[1]/table[#{i}]/tbody/tr[#{j}]/td[2]").inner_text
x1 = page_html.xpath(".//*[@id='main']/div[2]/div[2]/div[1]/table[#{i}]/tbody/tr[#{j}]/td[8]").inner_text
x2 = page_html.xpath(".//*[@id='main']/div[2]/div[2]/div[1]/table[#{i}]/tbody/tr[#{j}]/td[10]").inner_text
x3 = page_html.xpath(".//*[@id='main']/div[2]/div[2]/div[1]/table[#{i}]/tbody/tr[#{j}]/td[7]").inner_text
x4 = page_html.xpath(".//*[@id='main']/div[2]/div[2]/div[1]/table[#{i}]/tbody/tr[#{j}]/td[9]").inner_text
#parsing the ID of the specific human
x51 =
page_html.xpath(".//*[@id='main']/div[2]/div[2]/div[1]/table[#{i}]/tbody/tr[#{j}]/td[2]/preceding::h2[1]").inner_text
x5 = x51[14, 18]

                                #checking to see if the SNP is non-synonymous or synonymous
                if ((x == "SYNONYMOUS_CODING") || (x == "NON_SYNONYMOUS_CODING"))

                        #saving all the parsed elements, exccept the human ID, to a file
                        myfile = File.open("dat.txt", "a")
                        myfile.puts("#{z}\t#{x}\t#{x1}\t#{x2}\t#{x3}\t#{x4}\n")
                        myfile.close

                        #saving all the parsed elements, including the human ID, to a file
                        myfile2 = File.open("datper.txt", "a")
                        myfile2.puts("#{z}\t#{x}\t#{x1}\t#{x2}\t#{x3}\t#{x4}\t#{x5}\n")
                        myfile2.close

                end

                                end

                        end

        end

        #deleting the current protein whose data has been extracted from the protein file in order to proceed to the
        next protein
        `ruby -n -i.bak -e 'puts $_ unless $. == 1' #{ARGV[0]}`
        total= Time.now - start
        puts total

end

prot.close

b.close
```

```
#if the time when extracting the protein's data exceeds the limit of 19 seconds the scipt reloads in order to free the
memory and to reduce the data traffic
if (line != nil)

        sleep 19
        `irb extract.rb #{ARGV[0]}`

end

#removing dublicate entries and save the rest to a new file
nodupl = File.open("dat2.txt", "w")
result = `ruby -e 'puts STDIN.readlines.sort.uniq!.to_s' < dat.txt`
nodupl.puts("#{result}")
nodupl.close
```

from Mapping.java

```
..............................

//finding the SNP data of the current protein
while ((text1 = rd1.readLine()) != null){
        String[] words = text1.split("\t");

        //reading the mutation position
        if (words[0].equals(text)){
          x1 = Integer.parseInt(words[3]);

          if (words[1].equals("NON_SYNONYMOUS_CODING")){
                    String[] alter = words[5].split("/");
                    a1 = alter[0];
                    a2 = alter[1];
          }

          //checking if the SNP position falls into any of the regions
          for (i=0; i<1000; i++){
                    if((x1 >= reg1[i]) && (x1 <= reg2[i])){
                              concat = reg1[i]+"-"+reg2[i];

                              //mapping to the superfamily
                              if (args[0].equals("sf")) {

                              ResultSet r4 = s.executeQuery("select sf from ass where seqid='"+text+"' and
                              region like '%"+concat+"%'");

                                                  while(r4.next()){

                                        Statement s2=con.createStatement();

                                        if (words[1].equals("NON_SYNONYMOUS_CODING")){

                                          ResultSet r5 = s2.executeQuery("select total_non_syn from snps where sf='"+r4.getInt(1)+"'");


                                                  if(r5.last()){
                                                            cnt = r5.getInt(1)+1;
                                                            s2.executeUpdate("update snps set total_non_syn='"+cnt+"' where
                                                           sf='"+r4.getInt(1)+"'");

                                                  }


                                                  r5.close();

                                                  s2.close();
                                          }

                                        else if (words[1].equals("SYNONYMOUS_CODING")){

                                          ResultSet r5 = s2.executeQuery("select total_syn from snps where sf='"+r4.getInt(1)+"'");

                                                  if(r5.last()){
                                                            cnt = r5.getInt(1)+1;
                                                            s2.executeUpdate("update snps set total_syn='"+cnt+"' where
                                                           sf='"+r4.getInt(1)+"'");

                                                  }


                                                  r5.close();

                                                  s2.close();

                                        }

                              }
                              r4.close();
                              ..............................
```

```
//mapping to the regions
if (args[0].equals("sf")) {

        ResultSet r9 = s.executeQuery("select sf from snps");
        int sm1 = 0;
        int sm2 = 0;
        int sm3 = 0;
        int sm4 = 0;

        while (r9.next()){

           Statement s6=con.createStatement();
           ResultSet r10 = s6.executeQuery("select non_syn, syn from snp_reg where sf = '"+r9.getString(1)+"'");

           while (r10.next()){

                      Statement s8=con.createStatement();

                      if ((r10.getInt(1) > 0) && (r10.getInt(2) == 0)){

                                 sm1++;
                                 s8.executeUpdate("update snps set num_non_syn='"+sm1+"' where sf =
                                '"+r9.getString(1)+"'");
                      }

                      else if ((r10.getInt(2) > 0) && (r10.getInt(1) == 0)){

                                 sm2++;
                                 s8.executeUpdate("update snps set num_syn='"+sm2+"' where sf =
                                '"+r9.getString(1)+"'");
                      }

                      else if ((r10.getInt(1) > 0) && (r10.getInt(2) > 0)){

                                 sm3++;
                                 s8.executeUpdate("update snps set num_both='"+sm3+"' where sf =
                                '"+r9.getString(1)+"'");
                      }

                      else if ((r10.getInt(1) == 0) && (r10.getInt(2) == 0)){

                                 sm4++;
                                 s8.executeUpdate("update snps set snp_free='"+sm4+"' where sf =
                                '"+r9.getString(1)+"'");
                      }
                                 s8.close();

                      }

                      r10.close();
                      s6.close();

                      sm1 = 0;
                      sm2 = 0;
                      sm3 = 0;
                      sm4 = 0;
           }

           r9.close();
}

..............................
```

from Analysis.java

```
..............................

//checking if the mutation is a non-synonymous one and finding the amino acid change weight from the Blosum62
matrix
if (wrd[1].equals("NON_SYNONYMOUS_CODING")){
           String[] alter2 = wrd[5].split("/");
           a1 = alter2[0];
           a2 = alter2[1];

           for (i=0; i<21; i++){
              if (a1.equals(bl[i][0])){
                             flag1 = i;
              }
           }

           for (j=0; j<21; j++){
              if (a2.equals(bl[0][j])){
                             flag2 = j;
              }
           }
```

```
//fixing the Blosum62 matrix to fit to a positive scale
if (bl[flag1][flag2].equals("11")){
    x = 0;
}
else if (bl[flag1][flag2].equals("9")){
    x = 0;
}
else if (bl[flag1][flag2].equals("8")){
    x = 0;
}
else if (bl[flag1][flag2].equals("7")){
    x = 0;
}
else if (bl[flag1][flag2].equals("6")){
    x = 0;
}
else if (bl[flag1][flag2].equals("5")){
    x = 0;
}
else if (bl[flag1][flag2].equals("4")){
    x = 0;
}
else if (bl[flag1][flag2].equals("3")){
    x = 1;
}
else if (bl[flag1][flag2].equals("2")){
    x = 2;
}
else if (bl[flag1][flag2].equals("1")){
    x = 3;
}
else if (bl[flag1][flag2].equals("0")){
    x = 4;
}
else if (bl[flag1][flag2].equals("-1")){
    x = 5;
}
else if (bl[flag1][flag2].equals("-2")){
    x = 6;
}
else if (bl[flag1][flag2].equals("-3")){
    x = 7;
}
else if (bl[flag1][flag2].equals("-4")){
    x = 8;
}
```

```
z = Integer.parseInt(wrd[3]);
ResultSet r30 = s.executeQuery("select region from ass where genome = 'hs' and seqid = '"+wrd[0]+"' and
evalue <= 0.0001");
int reg11[] = new int[1000];
int reg22[] = new int[1000];
i = 0;

while(r30.next()){

    String[] sf2 = r30.getString(1).split(",");

    for (String region2 : sf2)  {
                    String[] rg2 = region2.split("-");
                    reg11[i] = Integer.parseInt(rg2[0]);
                    reg22[i] = Integer.parseInt(rg2[1]);
                    i++;

    }
}

r30.close();

//if the mutation position falls into a region then its amino acid change weight is added to the current sum of
the region
for (i=0; i<1000; i++){
    if((z >= reg11[i]) && (z <= reg22[i])){
                    concat2 = reg11[i]+"-"+reg22[i];

                    ResultSet r31 = s.executeQuery("select sf from ass where seqid='"+wrd[0]+"' and region like
                    '%"+concat2+"%'");
                    Statement s31=con.createStatement();
                    while(r31.next()){
                                    ResultSet r32 = s31.executeQuery("select weight, count from temp_blos where sf
                                    = '"+r31.getString(1)+"'");
                                    Statement s32=con.createStatement();

                                    while(r32.next()){
                                                    s32.executeUpdate("insert into temp (sf, seqid)
                                                    values('"+r31.getString(1)+"', '"+wrd[0]+"')");
                                                     sum = r32.getFloat(1) + x;

                                                     s32.executeUpdate("update temp_blos set weight='"+sum+"' where sf =
                                                    '"+r31.getString(1)+"'");
                                    }
                                    r32.close();
                                    s32.close();
                          }
                          r31.close();
                          s31.close();
                }
              }
  }
}
ResultSet r34 = s.executeQuery("select sf from temp_blos");
Statement s34=con.createStatement();
while(r34.next()){
        ResultSet r35 = s34.executeQuery("select count(distinct sf, seqid) from temp where sf='"+r34.getInt(1)+"'
        order by sf, seqid");
        Statement s35=con.createStatement();
        while(r35.next()){
          s35.executeUpdate("update temp_blos set count='"+r35.getInt(1)+"' where sf = '"+r34.getString(1)+"'");
        }
        r35.close();
        s35.close();
}
r34.close();
s34.close();
ResultSet r33 = s.executeQuery("select sf, weight, count from temp_blos");
Statement s33=con.createStatement();
FileWriter outwrite = new FileWriter("blos.txt");
BufferedWriter out = new BufferedWriter(outwrite);

FileWriter outwrite2 = new FileWriter("blos2.txt");
BufferedWriter out2 = new BufferedWriter(outwrite2);

//the general amino acid weight of the superfamily is normalized for the number of members(proteins) that had a
least one mutation
while(r33.next()){
        if (r33.getInt(3) > 0){
        sum2 = r33.getFloat(3);
        sum = r33.getFloat(2)/sum2;
          s33.executeUpdate("update temp_blos set average='"+sum+"' where sf = '"+r33.getString(1)+"'");
        }
        else {
          sum2 = 0;
          sum = 0;
        }
}
r33.close();
s33.close();
..............................
```