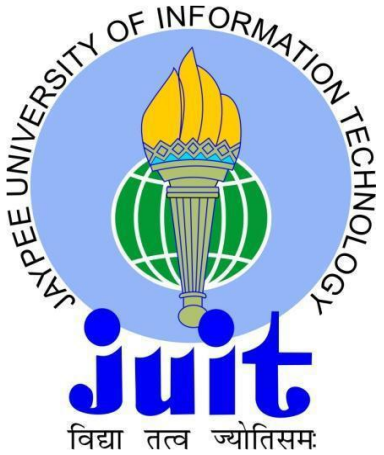
REMAP: A Web Server For Regulatory Elements Mapping And Prediction

Enrollment Number: 151502

Name of Student: Hitesh Thakur

Name of Supervisor: Dr. Tiratha Raj Singh



MAY 2019

Dissertation submitted in partial fulfilment of the requirement for the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF BIOTECHNOLOGY AND BIOINFORMATICS

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

WAKNAGHAT, SOLAN

CONTENTS

DECLARATION

I thus declare that the work introduced in this report entitled "REMAP: A Web Server for Regulatory Elements Mapping and Prediction" in partial satisfaction of the requirement for the award of the degree of Bachelor of Technology in Bioinformatics submitted in the Department of Biotechnology and Bioinformatics, Jaypee University of Information Technology, Waknaghat, Solan-173234, Himachal Pradesh is my very own genuine record work did over a period from July 2018 to May 2019 under the supervision of Dr Tiratha Raj Singh, Associate Professor (Senior Grade), Department of Biotechnology and Bioinformatics.

The content written in the report has not been submitted for the award of some other degree or diploma.

Signature of the student

(Hitesh Thakur)

CERTIFICATE

This is to affirm that project report entitled "REMAP: A Web Server for Regulatory Elements Mapping and Prediction", put together by Hitesh Thakur is in its partial satisfaction for the award of level of Bachelor of Technology in Bioinformatics Engineering to Jaypee University of Information Technology Waknaghat, Solan has been done under my watch.

This work has not been submitted incompletely or completely to some other college or institution so as to accomplish any award or some other degree.

Signature

Supervisor Name: Dr. Tiratha Raj Singh
Designation: Associate Professor
Jaypee University of Information Technology,
Waknaghat Solan, Himachal Pradesh

ACKNOWLEDGEMENT

I might want to offer my earnest thanks and gratitude to my project guide, Dr. Tiratha Raj Singh, whose patience, direction, consolation, and devotion had propelled me to do this project under him.

I might likewise want to express genuine gratitude to Dr. Sudhir Sayal (Head, Department of Biotechnology and Bioinformatics) for his command and backing to take a shot at this project.

Moreover, I might want to thank the Administration of the Department of Biotechnology and Bioinformatics for the majority of their specialized expertise and support.

I additionally appreciate the majority of the help my folks, companions and instructors have given me all through the educational curriculum, I couldn't have done this without the assistance and support of every one of these individuals.

Thank you

LIST OF FIGURES

LIST OF ABBREVIATIONS

miRNA	Micro Ribonucleic Acid
Bp	Base Pairs
RNA	Ribonucleic Acid
miRISC	Micro RNA-Induced Silencing Complex
UTR	Untranslated Region
TFBS	Transcription Factor Binding Site
SSRs	Simple Sequence Repeats
STRs	Short Tandem Repeats
MISA	Micrsatellite
FASTA	Fast Alignment
DNA	Deoxyribonucleic Acid
VNTR	Variable Number of Tandem Repeats
SNPs	Single Nucleotide Polymorphisms
mRNA	Messenger RNA
ORF	Open Reading Frame
CAI	Codon Adaptive Index
Nc	Effective Number of Codons
SDLC	Software Development Life Cycle
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
XML	Extensible Markup Language
GUI	Graphical User Interface
VCF	Variant Call Format
CROM	Chromosome
POS	Position
ID	Identifier
REF	Reference
ALT	Alternate
QUAL	Quality
REA	Regulatory Element Analysis
INFO	Information

ABSTRACT

Computational methods for regulative elements mapping and prediction are presently undergoing in depth review and analysis. There's however an implausible demand for development of these tools and bioinformatics approaches are looking towards high-throughput tries to approve expectations. The mix of large-scale techniques with computational tools won't solely give larger credence to computational predictions however conjointly result in the higher understanding of specific biological queries. Apart from all the individual tools required for mapping and predicting regulatory elements, there is need or a server that integrate all these tools and allow to perform analysis on one platform only.

Re-Map is a web server that integrate all the major tools required for analysis of regulatory elements in genome sequences. Its main applications are (i) miRNA prediction in the genome sequences, (ii) microsatellites or simple sequence repeats prediction and (iii) predicting transcription factor binding sites in genome sequences. All these putative regulatory elements will be analysed to provide meaningful information to the academicians and researchers.

Chapter 1 - INTRODUCTION

Knowing the regulative mechanisms responsible for gene expression stays one among the foremost fundamental difficulties for biomedical investigation. A regulative part can operate 1000 base pairs (bp) away from the template sequence [1], adding one more layer of much more complexness to transcriptional regulation. However, most genes are successfully annotated, our information of regulative components that controls such genes in numerous cell varieties, at varied time periods and in totally different environment conditions remains restricted. Recent studies show that mutations in several of the already familiar regulative components are related to health issues [2], pointing towards the vital role that regulative components may play in identifying disease and drug discovery.

Computational techniques for miRNA target expectation are as of presently encountering wide review and assessment. There's in any case a doubtful interest for improvement of such devices and bioinformatics approaches are attempting towards high-throughput examinations to approve expectations. The blend of huge scale system with computational devices won't just give a great deal of significant conviction to computational desires in any case moreover result in the higher understanding of specific biological queries. MicroRNAs (miRNAs) are very little, non-coding ribonucleic acid molecules that manage the expression of protein-coding genes at the post-transcriptional level. Since various essential formative and physiological procedures are entirely managed by miRNAs, it is not astounding that deregulation of miRNA work has been involved within the pathological process of diverse human diseases [3]. Understanding miRNA function has thus been a significant focus of biomedical analysis within the previous decade.

In the authoritative pathway, miRNAs direct a protein complex, named miRISC, to binding site that regularly reside in the 3′ untranslated area (3′ UTR) of target mRNA molecules. In this manner, miRISC starts inhibition of translation, deadenylation and rot of the target mRNA [4]. Learning of target mRNAs is basic to comprehend the job of a specific miRNA in both typical cell procedures and pathogenesis. Correspondingly, knowing the full complement of miRNAs controlling a specific mRNA is fundamental to grasp its dynamic direction that is firmly connected to its function.

Moreover, a main emphasis in cell science is to discover functional Transcription Factor Binding Sites in charge of the control of a downstream gene and microsatellite or simple sequence repeats present in the genomic arrangements. As wet-lab procedures are repetitive and expensive, it isn't reasonable to perceive TFBS for all uncharacterized genes in the genome

by basically test implies. Computational systems went for predicting potential regulatory regions can build the effectiveness of wet-lab analyses meaningfully. Microsatellites arose around 25 years back [5] and remain an ordinarily utilized genetic marker system in plant genetics, reproducing and forensics [6], where they are regularly stated as simple sequence repeats (SSRs) or short tandem repeats (STR), separately. The fundamental structure unit of a microsatellite is a short grouping theme (more often somewhere in the range of one and six base-pairs long) that is repeated in tandem. These attributes can be distinguished by the in-silico examination of nucleotide sequences gotten by conventional Sanger or high-throughput resequencing information. The MISA microsatellite discoverer is a device for finding microsatellites in nucleotide sequences. Despite the acknowledgment of flawless microsatellites, MISA is also prepared to find flawless compound microsatellites that are made numerous events out of more than one simple sequence motif [7].

1.1 Problem Statement

Regulatory elements prediction and mapping in various organisms is a bottleneck for seeing how biological procedures are sorted out, how they function, and how they developed in those species. For the prediction of such regulatory elements, a researcher needs to experience writing mining, or servers which predict single regulatory element at a time. For prediction of such regulatory elements, a researcher needs to experience bunches of tools (e.g. to predict or identify regulatory elements one must go multiple server as each server is specifically for only one regulatory elements) which takes more endeavours and are time-consuming processes.

1.2 Objective

To reduce such endeavours mentioned above, built up the asset by incorporating major tools and information, making enhanced forms of famous tools and built up another web server by conquering the execution of tools that were minimally utilized or worked inadequately keeping in mind the end goal to streamline the client encounter.

- All major tools for the analysis of regulatory elements at one place.
- Integration of tools for the analysis of genes at sequence level.
- Provide better visualization of important elements using dynamic structures.

1.3 Proposed Web Server

Here, we present a new user-friendly web server, called “Re-MAP” – Regulatory Elements Mapping and Prediction Server that will allow biologists to identify and predict major regulatory elements at a single platform with less endeavours. Also, facilitating user to run some small level sequence analysis quickly and providing them option to download their results.

This web server is freely accessible to each and every one all over the web, universally. The user can perform regulatory elements and sequence analysis on their uploaded files which should be in standard and most adequate FASTA file format. There is also a sample dataset which are available to download. The web server does not ask for any paid services for its working. The users will most likely get in touch with us through a form gave in the website, where they can give us feedback and recommendations about the adjustments or the updates required for the web server.

1.4 Applications

Regulatory Elements Mapping and Prediction:

- MicroRNAs prediction in the genome sequences

  MicroRNAs (miRNAs) are very small generally range from 18 – 28 nucleotides in size and are non-coding ribonucleic molecules. Their main job is in the post-transcription regulation to control the expression of the protein with specific function and their participation in traditional and pathological cellular processes has been demonstrated. miRNAs can be defined as "multivalent," with one miRNA prepared to target multiple genes, thus controlling the expression of numerous proteins. Many crucial cellular processes, such as cell differentiation from each other, cell cycle progression throughout its life span, and cell death [8][9]. Micro RNAs in pituitary differentiation have been shown to play relevant roles. Missing information about miRNA binding genes, however, delays full understanding of miRNAs' biological functions. More studies are therefore required to predict miRNA binding genes in pituitary adenomas for either down or up regulated miRNAs. Predictive miRNAs are likely to be useful diagnostic markers, increasing pituitary adenomas arrangement.

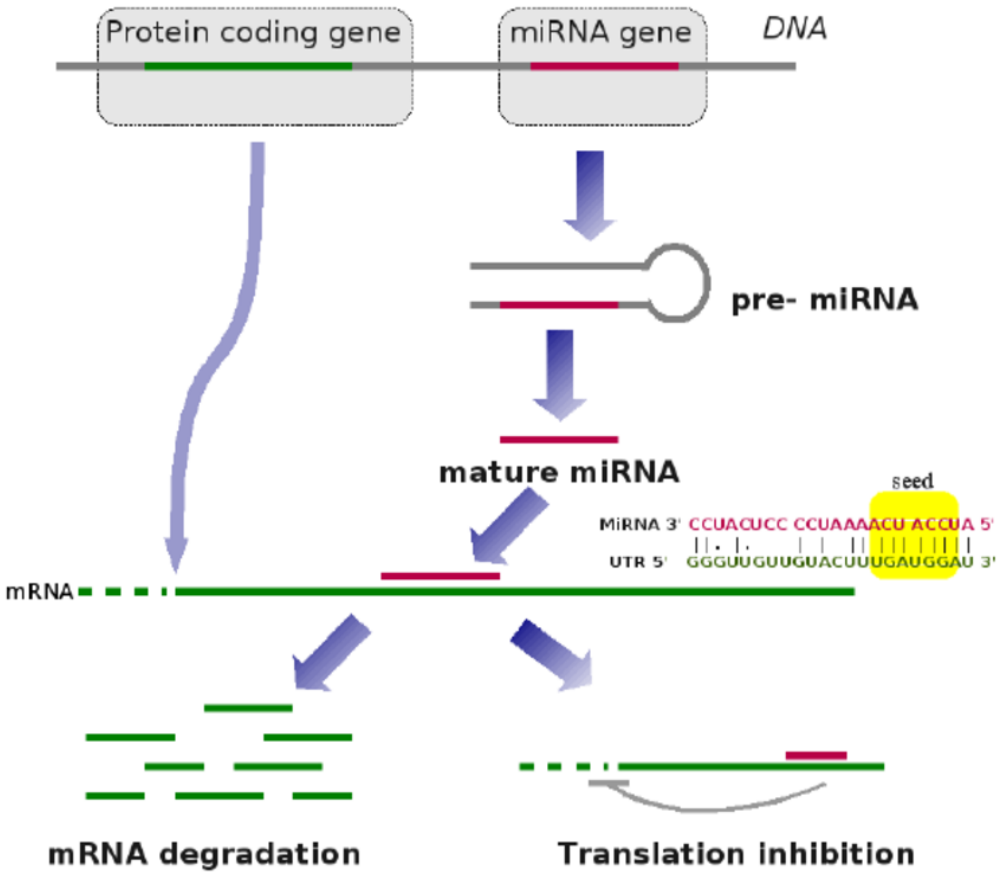


Figure 1: Illustration of miRNA function

[Image source: Researchgate, https://www.researchgate.net/figure/Simplified-illustration-of-miRNA-biogenesis-and-function-miRNA-genes-are-first_fig1_221197863, 1 May 2019]

- Micro-satellites (SSRs) prediction

  A microsatellite can be a repetitive deoxyribonucleic acid tract within which certain DNA patterns (ranging from 1 to 6 or additional base pairs) are recurrent, usually 5 to 50 times.[10] [11] Microsatellites occur at thousands of locations inside the genome of an organism. They have a larger mutation rate than alternative areas of deoxyribonucleic acid [12] leading in high genetic diversity. Microsatellites are typically spoken by forensic geneticists and in genealogy as short tandem repeats (STRs), or by plant geneticists as simple sequence repeats (SSRs) [13].

  Microsatellites are classified as VNTR (variable number of tandem repeats) deoxyribonucleic acid along with their longer cousins, the minisatellites. The name "satellite" deoxyribonucleic acid refers to the first observation that the centrifugation of genomic deoxyribonucleic acid in a test tube separates a distinguished layer of bulk

deoxyribonucleic acid from related "satellite" layers of repetitive deoxyribonucleic acid [14].

In cancer diagnosis, kinship analysis (especially paternity testing) and forensic identification, they are widely used for DNA profiling. In addition, they are utilized in genetic linkage analysis to find a gene or a mutation that is susceptible for a given trait or disease. Microsatellites are used in population genetic science to measure levels of connectivity between subspecies, groups and people.

- Identification of Single Nucleotide Polymorphism (SNPs)

  A single-nucleotide polymorphism could be a variation in the DNA sequence that happens when a single nucleotide adenine [A], thymine [T], cytosine [C], or guanine [G] within the genome (or alternate shared sequence) varies in associates of a species or paired chromosomes in a specific person.[15] For instance, there are two DNA fragments with known sequence from totally diverse people, AAGAGCGTGA to AAGAGCTTGA, contain a distinction only at one nucleotide position. We can say that we have 2 alleles in this case: G and T. There are only 2 alleles in the majority of common SNPs.

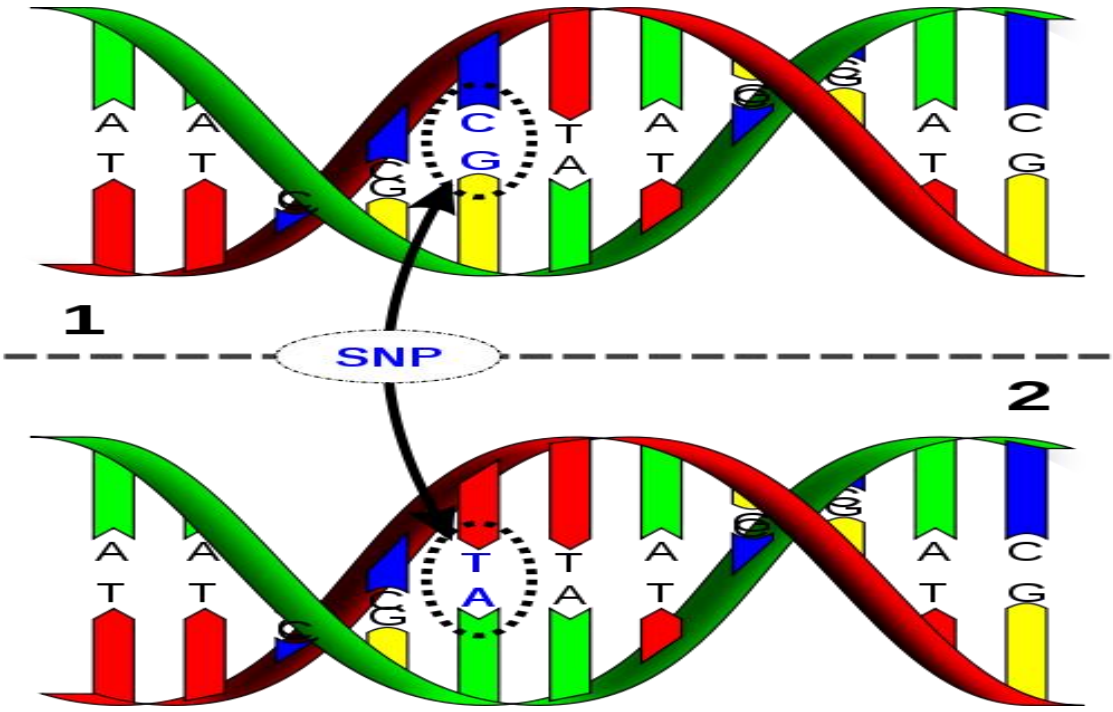


Figure 2: Single Nucleotide Polymorphism

[Image source: ISOGG, https://isogg.org/wiki/Single-nucleotide_polymorphism, 1 May 2019]

Single nucleotides could also be modified (substitution), removed (deletions) or added (insertion) to a polynucleotide sequence. Single nucleotide polymorphisms could fall inside coding sequences of genes, non-coding regions of genes, or within the intergenic regions between genes. SNPs inside a coding sequence won't essentially change the amino acid sequence of the protein that's created, thanks to degeneracy of the ordering. A SNP during which each form causes an equivalent peptide sequence is termed synonymous (sometimes known as a silent mutation) — if a distinct peptide sequence is created, they're nonsynonymous. A nonsynonymous amendment could either be missense or nonsense, wherever a missense amendment leads to a distinct amino acid, whereas nonsense changes leads to a premature stop codon. [16] SNPs that aren't in protein-coding regions should still have consequences for gene splicing, transcription factor binding, or the sequence of non-coding RNA.

Sequence analysis package:

- Translation of nucleotide sequences using multiple genetic code system

  The genes in deoxyribonucleic acid encode protein molecules, which are the cell's "workhorses," ending all the necessary functions forever. As an example, all proteins are enzymes, along with some that digest nutrients and helps in building new cellular elements. Additionally, there are some that create copies of deoxyribonucleic acid throughout cell division such as deoxyribonucleic acid polymerases and various enzymes [17].

  In the simplest sense, gene expression suggests that synthesizing its corresponding protein, and this process consist of basically two important steps. In the first step, the information which is present inside the deoxyribonucleic acid is passed to a messenger RNA (mRNA) fragment by the method which is commonly known as transcription. During the process when transcription is taking place, the deoxyribonucleic acid of a gene assists as a template for complementary base pairing, and a protein which is widely known as polymer enzyme II catalyze the construction of a pre-mRNA molecule, that is then further undergo some processes to create mature messenger RNA (Figure 3). At last, the mRNA that has been formed is only a single-stranded copy of the gene, that further need to be translated into a protein molecule.

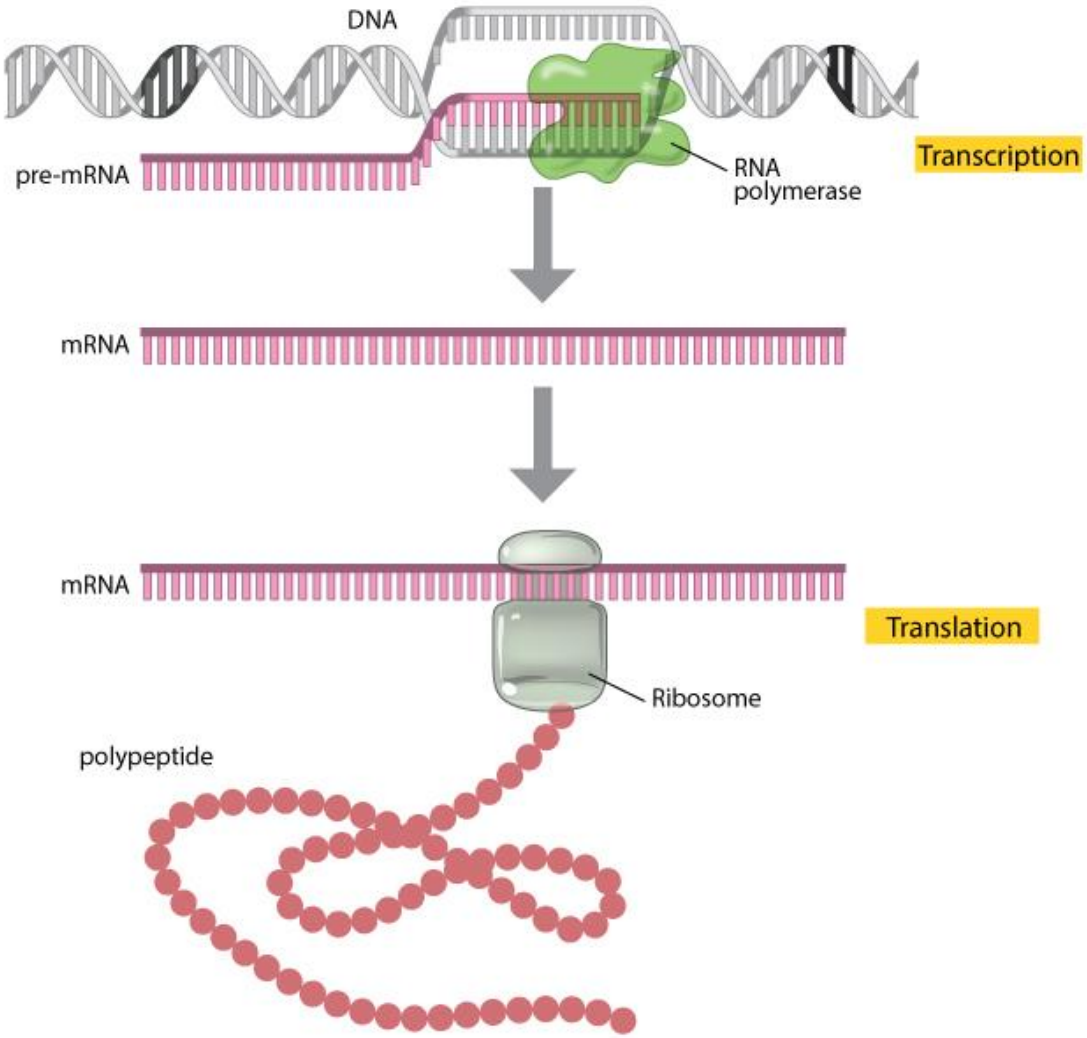


Figure 3: Translation of DNA into Protein

[Image source: Nature, https://www.nature.com/scitable/topicpage/translation-dna-to-mrna-to-protein-393, 1 May 2019]

The messenger RNA is "read" in accordance with the triplet formed by the three nucleotides where each triplet codes for particular amino acid, on the basis of which polypeptide chain is synthesized, which is the second major or last step in gene expression or translation. The sequence of ribonucleic acid is thus used as a base for generating a long chain of amino acid that forms a protein. Once the stop codon is recognized the chain synthesis stops.

- Determining the consensus sequence for the given multiple nucleotide sequences:
  A consensus sequence is a perfect promoter sequence in deoxyribonucleic acid [18] - in E. coli, for instance, two are found, a -35 sequence and a -10 sequence. The best

promoter sequence - the consensus sequence - isn't truly found in deoxyribonucleic acid, and a promoter's strength is judged by its similarity to the consensus sequence. The nearer a promoter is to the best sequence, the stronger it'll be and thus a lot of messenger RNA are made, which can result in a bigger yield of proteins. The -35 consensus sequence is TTGACA, and also the -10 consensus sequence is TATAAT.

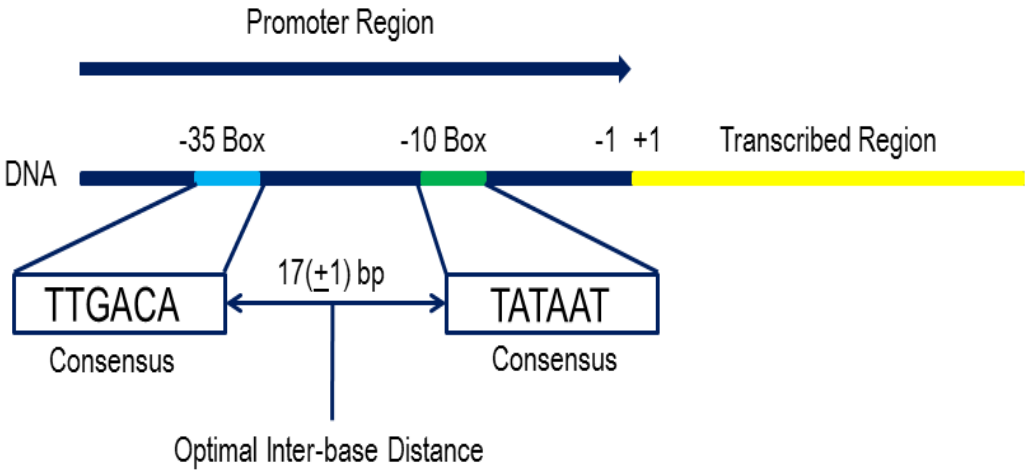


Figure 4: Consensus sequence in case of E. coli

[Image source: Biomedical Sciences, https://teaching.ncl.ac.uk/bms/wiki/index.php/Consensus_sequence, 1 May 2019]

Originally, the consensus sequence was determined by comparing already well-known promoter sequences and selecting the base that was most typical of each position. Any upstream sequence of the transcription starting site is given a negative sign in front of it as the starting site is + 1.

- Open Reading Frame (ORF) Finder:

  Open reading frame (ORF) in genetic science is the part of a nucleotide sequence which has initiation codon and has the ability to translate into a protein. An ORF may be a nonstop stretch of codons starting with a beginning nucleotide triplet (generally AUG) and ending with a stop codon (usually UAA, UAG or UGA) that do not code for any amino acid.[19] The ATG sequence (RNA AUG) inside the ORF may indicate wherever the translation starts. After the ORF, after the transcription stop codon, the transcription termination site is found. If transcription were to stop before the stop sequence, the entire translation would create an incomplete protein. [20] The introns

are removed in eukaryotic genes with the help of splicing and this is a post transcription process. Then the spliced exons are joined along after transcription step is completed to form the final protein translation RNA. Within the context of finding a gene, the start-stop explanation of an ORF thus solely applies to spliced mRNAs and this does not imply to genomic DNA as there may be a chance of appearing stop codon within the introns and/or shifts in reading frame. So, we do not consider start and stop codons while discussing genomic DNA.

Figure 5: Open reading frame with six-frame translation

[Image source: Wikipedia, https://en.wikipedia.org/wiki/Open_reading_frame, 1 May 2019]

- Summary of the nucleotide sequences:

  To obtain the count of A/T/G/C/N, total length, AT and GC content percentage in a given nucleotide sequences.

- To find K-mers of desired length for a sequence:

  In computational genetics, k-mers consult with all the potential sub sequences (with length as k) from scanning deoxyribonucleic acid sequencing. The quantity of k-mers potential given a string of size “L” is “L-k+1”, while the amount of potential k-mers

given “n” possibilities (4 in case of dealing with deoxyribonucleic acid e.g. ACTG) is “n^k”.

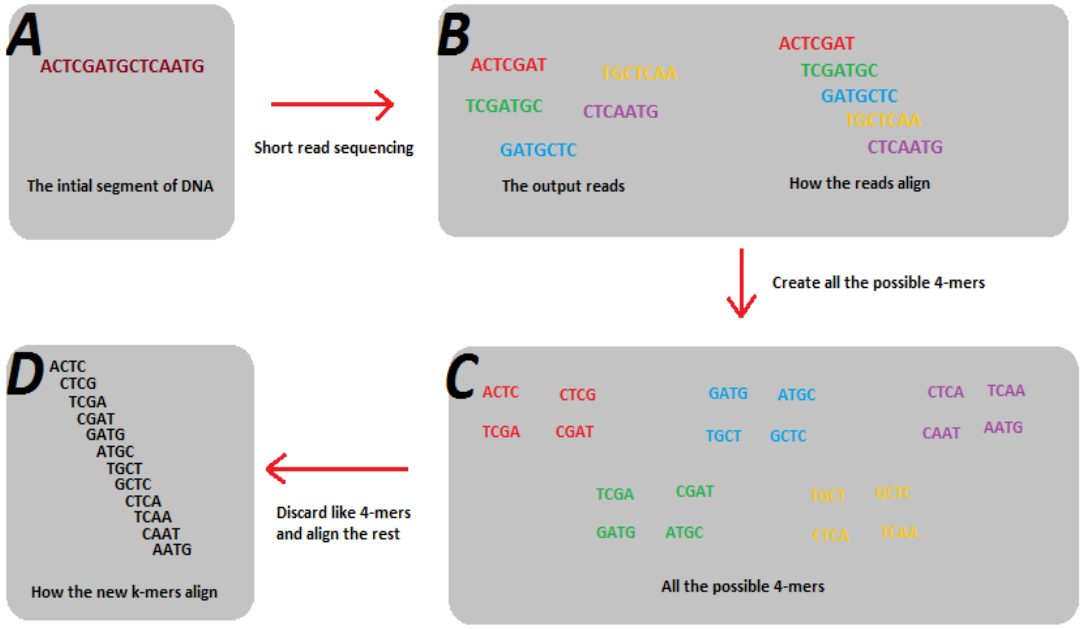


Figure 6: Process of splitting reads into smaller k-mers

[Image source: Wikipedia, https://en.wikipedia.org/wiki/K-mer, 1 May 2019]

K-mers are usually used while doing the assembly of the sequences, [21] however may be utilized in sequence alignment. Within the context of the human genome, k-mers of varied lengths are used to justify variability in mutation rates.

- Calculating the Codon Adaptation Index (CAI):

  The Codon Adaptation Index (CAI) [22] is the most common technique used to analyse bias in the use of codon. Unlike various codon-use bias measures, such as the “effective number of codons” (Nc), which measures direct change from a consistent bias (null hypothesis). CAI is used to measures the change of a given nucleotide sequence which codes for a protein from the already provided reference set of nucleotide sequences i.e. basically genes. CAI is used to predict the amount of expression of a gene based on its codon sequence as a quantitative technique.

Chapter 2 - MATERIALS AND METHODS

2.1 Explanation of the Project

Requirement: A web server for regulatory elements mapping and prediction that allow biologists to identify and predict major regulatory elements at a single platform.
Input: Nucleotide sequences either single or multiple in FASTA file format (.fa).
Output: Identify or predict regulatory elements in the given sequence(s) uploaded by client/operator.

2.2 Setting up Requirements

Machine that holds server: Installation of XAMPP or any other APACHE server for creating local server.
Machine required by the client: Working PC with internet connection to connect to the server.
Software: Perl and python needs to install on the server with necessary packages/modules installed (as required for the written scripts which runs in the background).

2.3 Software Development Life Cycle Model:

A Software Development Life Cycle (SDLC) relates to the indispensable stages that engineers need while making any new programming bundle, for example, planning, breaking down, structuring and actualizing. A life cycle of programming improvement covers all phases of programming bundle advancement from the earliest starting point, holding the need to keep up the product package.[23] Multiple SDLC models are out there like falls, iterative, V-shaped, coordinated, and so on. Every one of these models was considered, their properties, benefits, drawbacks were studied, and it was presumed that the waterfall model was the most suitable model for REMAP web server to hold the project.
There are certain qualities of Waterfall Model:

- It's an orderly model of development.
- The desires must be clear in this model before reaching the next section.
- Testing is permitted if the whole code is created.
- Improvement stages need to happen at the following stage and there is no intersection between two phases.
- Work advancement ought to be reported once each area has been finished.

- Testing takes place at the end of each section as well. This practice assists in maintaining the project standard.
- Each step should be made closed before moving to the next step, i.e. the need is frozen in advance, at that time only the coding and alternative implementation will take place.
- A time limit should be set for each section to be completed.

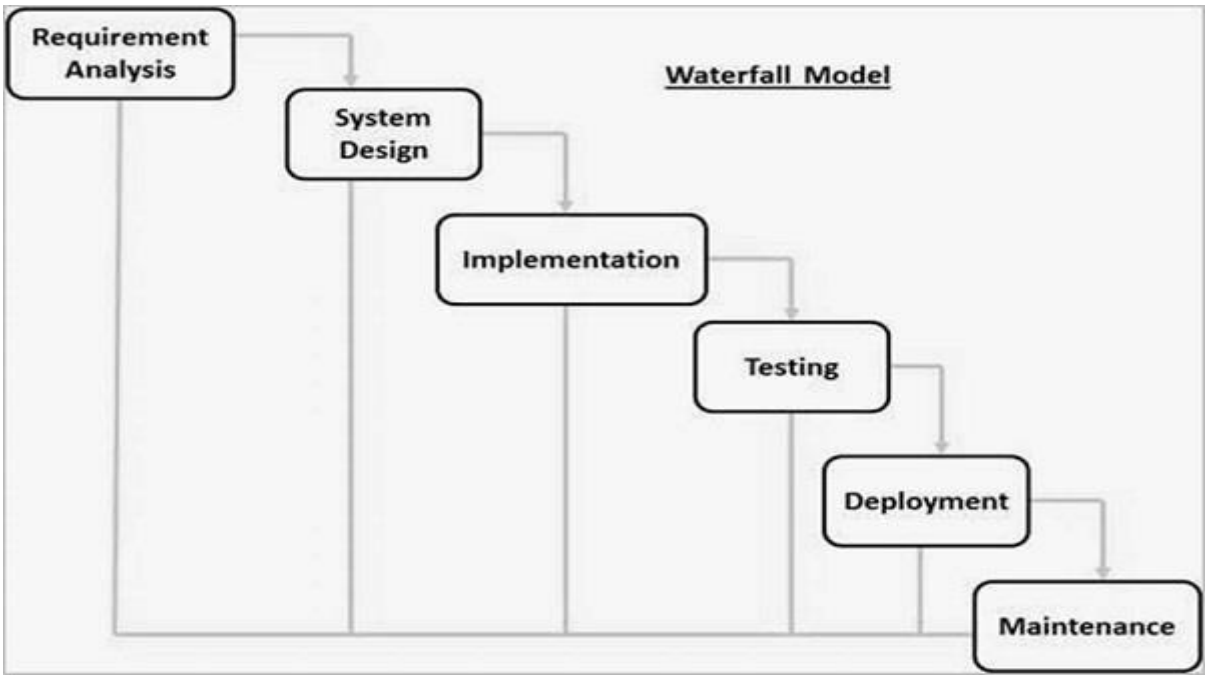


Figure 7: Waterfall Model

[Image source: tutorialspoint, https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm, 1 May 2019]

In this task, the means that have been pursued:

- Initially, examined all the data about the regulatory elements, its applications, previously existing scripts/tools, the usage of these tools and how to coordinate these instruments on a single platform i.e. web server.
- Next, endeavoured to plan a technique about how the site with server will be actualized, how the front end will look, how to deal with numerous clients demands, how to execute PERL/python contents in the backend, which programming languages and tools would be utilized, and so on.
- Next, in the improvement step we built up a website page utilizing HTML, Bootstrap, JavaScript, CSS and PHP.

- After the improvement is finished, we tried the server with numerous request and inputs, and inspect the time required by the server to create outputs.
- The subsequent stage is usage or launching our web server on the web.
- At last, we will continue updating our site. For instance, adding more regulatory elements prediction tools and providing more choices to the user. In more straightforward words, will attempt to stay aware of the support of the site.

The work flowchart is shown that indicates the various steps.

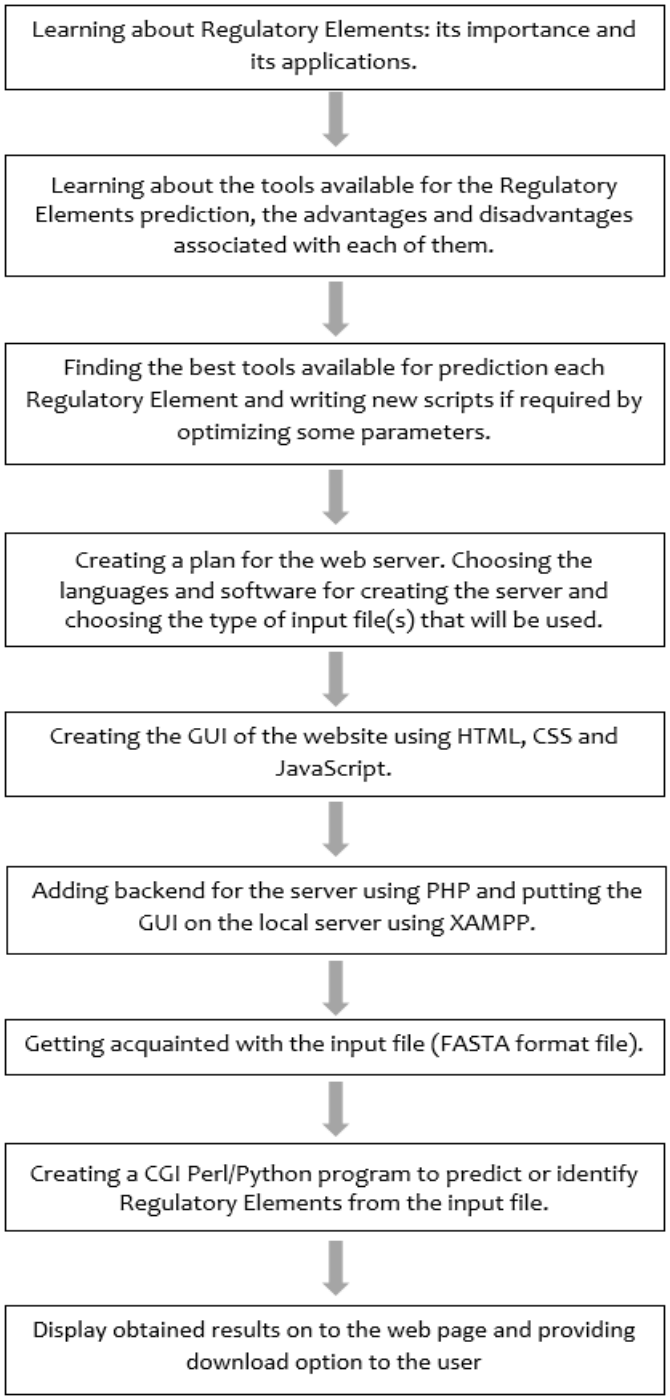


Figure 8: Workflow chart

2.4 Tools and Programming Languages

Multiple tools and languages will be required to create this tool. The decision of these tools and languages depended on their ease of use, how agreeable I am with them, and whether they can create the output we consider.

2.4.1 Local Server - XAMPP:

XAMPP, created by Apache Friends, furnishes a client with every one of the things that a web server needs to arrange. XAMPP is a cross-stage web server that proposes that it tends to be utilized similarly well to make a nearby Windows, Mac and UNIX framework. It incorporates Apache Server, MariaDB database, and scripting language mediators, for example, Perl and PHP (Figure 9). The progress from changing website currently on local server to later on online server is smooth as most web servers utilize indistinguishable components from XAMPP [24].

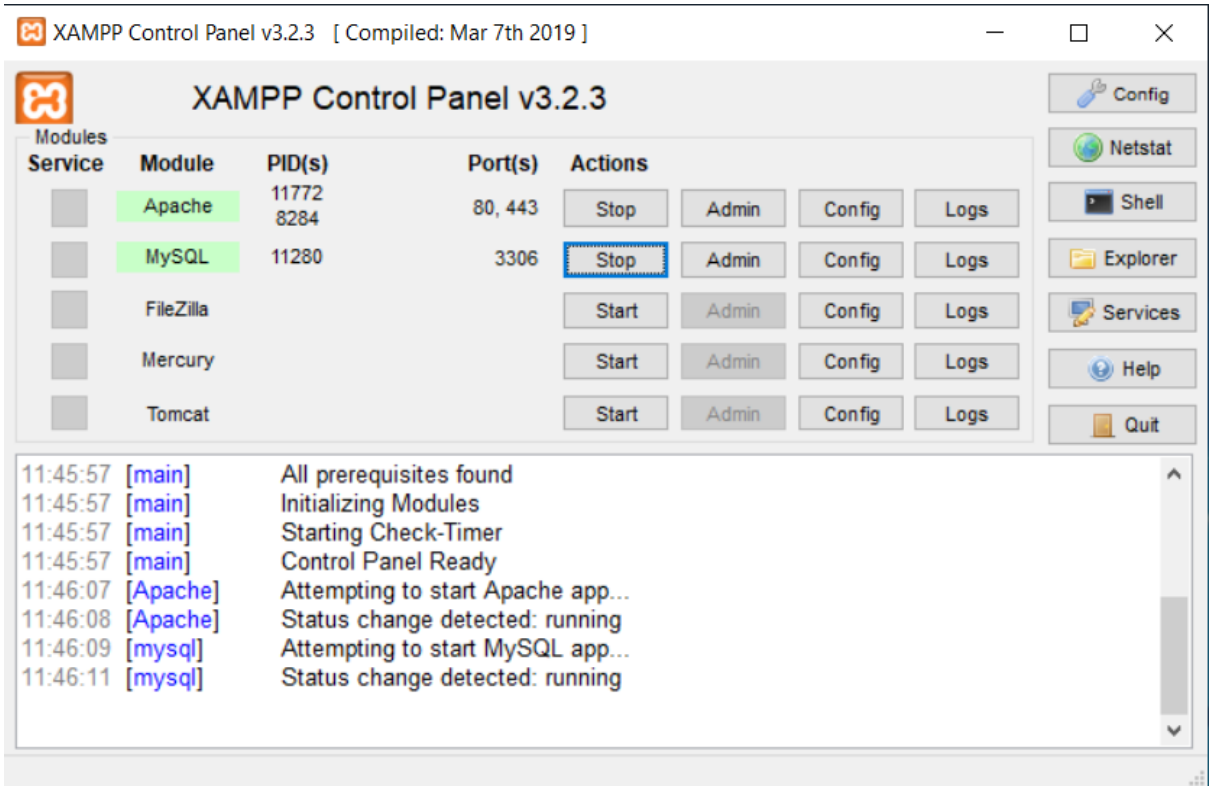


Figure 9: XAMPP control panel

2.4.2 Hypertext Mark-up Language:

HTML is utilized worldwide to make rich site and applications as an increase language. For instance, labels can be made in HTML tables, content can be styled utilizing labels, text style,

records can be created, hyperlinks can be given, pictures, recordings and different articles can be installed, and considerably more. HTML can be written in two linguistic uses, HTML being one and XML being the other. In spite of the fact that XML is quicker than HTML, however XML support for the program is restricted, for this undertaking HTML will be favoured over XML [25].

2.4.3 JavaScript

JavaScript is a language of scripting used to make dynamic website pages. It works with HTML and is for the most part used to make responsive pages, for example, moving pictures, responsive catches, slideshows, and so on [26].

2.4.4 Cascading Style Sheets (CSS)

CSS is a template language that is utilized in our hypertext increase language to set the looks of an increase language. CSS is the third language used to make site pages separated from hypertext increase language and JavaScript. CSS is for the most part used to isolate the content of the website page from its introduction, so the introduction of records is generally set as opposed to designing it each time some content is included or ever changed [27].

2.4.5 Bootstrap

Bootstrap is a front-end free and open-source library for sites and web applications. It incorporates increase language and layouts for typography dependent on CSS, catches, route and alternate interface parts, just as discretionary JavaScript extensions. It is expected to facilitate the dynamic sites and web applications event [28]. Bootstrap is a front-end web structure, that is, an easy to use, dislikes the server-side code on the "back end" or server. Bootstrap is GitHub's second most featured task, with more than 95,000 stars and more than 40,000 forks.

2.4.6 Hypertext Preprocessor (PHP)

PHP is a language of programming that permits web engineers to make dynamic substance that associates with databases. PHP is mostly used to create web package applications for software. PHP performs system functions, i.e. generating, opening, reading, writing, and shutting them from files on a system. PHP handles forms, i.e. collecting file information, saving information to a file, sending information via email, returning information to the user. Within your

information, you add, delete and modify components via PHP. Variables for accessing cookies and setting cookies. You can limit users to access some of your website's pages using PHP. It will be able to encode data.

2.4.7 Python

Python is a language of scripting that is high-level, understandable, interactive and object-oriented. Python is meant to be very clear. It frequently uses English keywords wherever punctuation is used as alternative languages, and it is less syntactic than alternative languages. The interpreter processes Python at runtime. You don't have to compile your program before you run it. This can almost be the same as PERL and PHP.

2.4.8 Perl

Perl is an artificial general-purpose language originally developed for text manipulation and currently used together with system administration, web development, network programming, interface development, and more for a wide range of tasks. Because of its content control capacities and fast improvement cycle, Perl used to be the most well-known web artificial language. Perl is broadly alluded to as "Internet duct tape". Perl can deal with scrambled web information, just as web-based business exchanges. Perl can be inserted in web servers to accelerate forms by up to 2000 percent.

2.5 Implementation

The Re-Map was created in three major steps involving data collection from various sources, data collection integration, and the development of web portals. Data collection involves gene and annotation, miRNA dataset, and most appropriate major tools in the form of scripts either PERL or Python. Dataset collected and its source are listed in the table below (Table1).

Table1: Data source for miRNA module used for creating Re-Map Server.

Dataset	Data source	Output
miRNA	miRbase	Predict the miRNA targets.

For the miRNA module, data collected was stored for each species in separate FASTA file and parsed using custom python program. Once the dataset is created, using some parameters of

similarity between query sequences and miRNA dataset, PERL program is used to identify the target. For the user's convenience, some already known model organism data set is provided to download them directly and use them as a test sample for analysis.

We have incorporated freely available microsatellite script in simple sequence repeats prediction module. A microsatellite investigation with PERL content of MISA joined in our web server requires just a single information document as a setup record ('MISA.ini') with three information parameters: 'SSR look parameters', 'compound SSR look through parameters' and 'yield document type parameters' are as of now predefined by us. The required information record is a FASTA document containing the nucleotide arrangement that will be dug for microsatellites.

Identification parameters used microsatellite prediction:

"definition (unit_size, min_repeats): 1-10 2-6 3-5 4-5 5-5 6-5"

"interruptions(max_difference_between_2_SSRs): 100"

misa.ini

"definition (unit_size, min_repeats): 1-12 2-6 3-5 4-3 5-3 6-2"

"interruptions(max_difference_between_2_SSRs): 100"

Identification parameters

"Monomer minimum 12 bp"

"Dimer minimum 12 bp"

"Trimer 15bp"

In single nucleotide polymorphism module, SNP sites detection python script is integrated which runs at the background whenever the user input the necessary files in the respected module and hit the button to identify or detect SNPs in the provided nucleotide sequences. The results produced by this module has been tested multiple times and confirmed to be correct.

In sequence analysis package, there was no need to have datasets/database. The sequence analysis package is fully dependent on scripts written in either PERL or python programming languages. This module only requires FASTA format nucleotide sequence file may be single or multiple and runs respected scripts in the background and provide instant result in the form of pop-up modals on the same page. Even download option is provided if user wants to download his/her results for the given inputs.

Chapter 3 - RESULTS AND DISCUSSION

In genetic or genomic studies biologists usually end up with a set of genes involved in different biological processes. Knowing the transcriptional factors of those genes is a challenging and time consuming for the researcher due to the vast amount of available data and as well as tools with different accuracy and complexities. Keeping this in mind we have developed a Re-Map server using the data integration, reproducible and highly scalable approaches. This tool is meant only for predicting or identifying the targets of these elements with the help of mapping. Integration of multiple tools related to regulatory elements has given us the advantage over other servers, as to our knowledge each server available for regulatory elements is specific to one regulatory element only. While a few servers for regulatory elements identification in genomic sequences are already publicly available, but we have added some new features like incorporating sequence analysis module where user can perform various tasks such as translation, consensus sequence, ORF finder, sequence summary, find K-mers and CAI-value.

3.1 GUI of REMAP Server

For the making of Graphical User Interface (GUI) the code was written in the HTML with Bootstrap and the content was styled utilizing CSS. The pages were made powerful utilizing JavaScript.

In the home page, there is an overview of the web server i.e. what is it all about with the option to directly go to the particular prediction tool by just one click. Also, a navigation bar is provided for the direct access to the home page, tools page, basic utility page, about page and contact page.

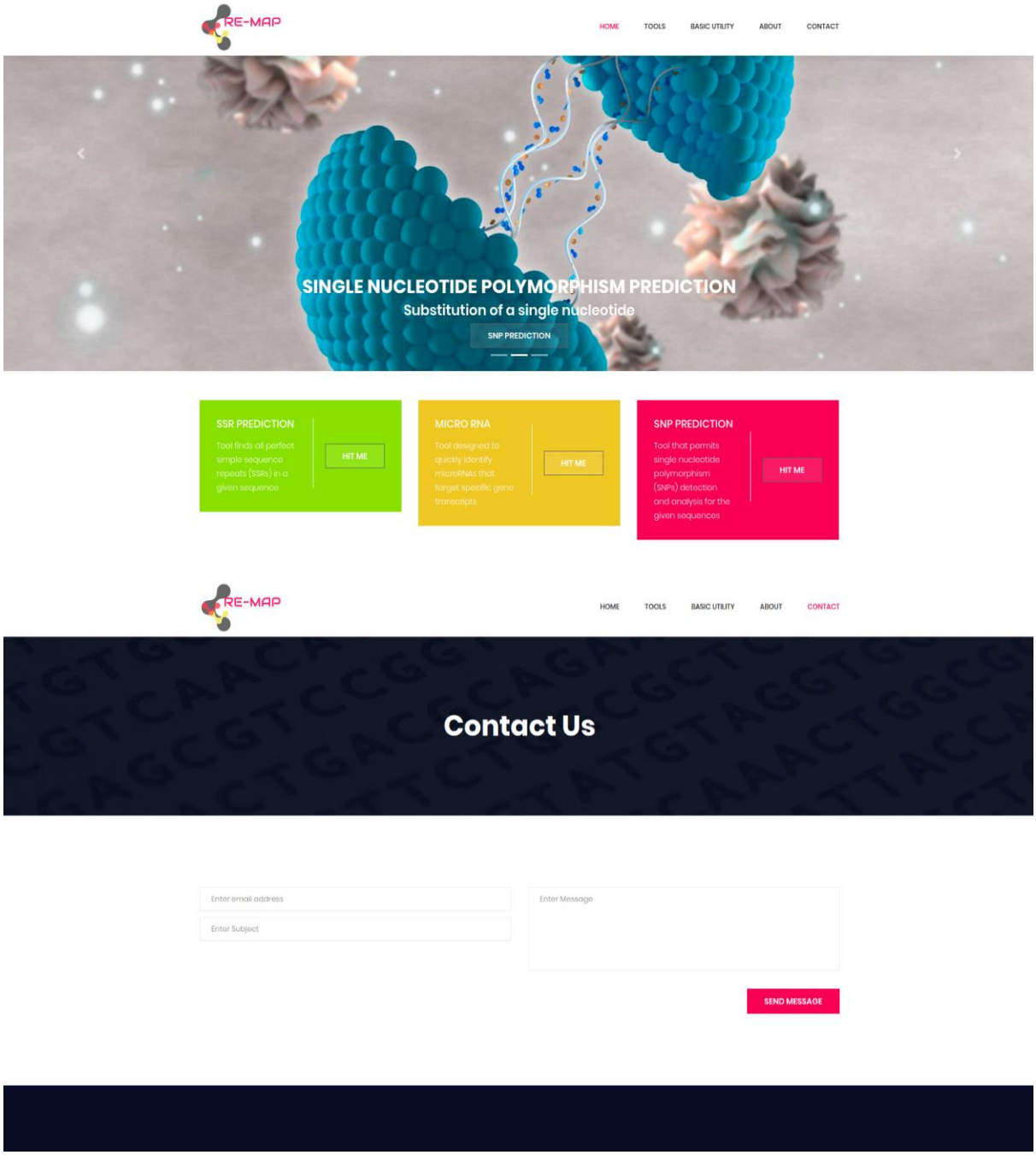


Figure 10: REMAP webserver graphical user interface

3.1.1 Micro RNA Module

In miRNA module, first user needs to select the database from dropdown menu and also need to upload or input his query sequence. Then miRNA targets are identified in the query sequence(s) and for each query sequence having miRNA targets sites, those sites are retrieved with the organism name and accession id. Moreover, the position of target sites (start and end) is provided in the results with the dynamic structure of stem loop of the miRNA. For annotation of the miRNA predicted we provided a direct link to miRBase.

3.1.1.1 Input GUI for miRNA module

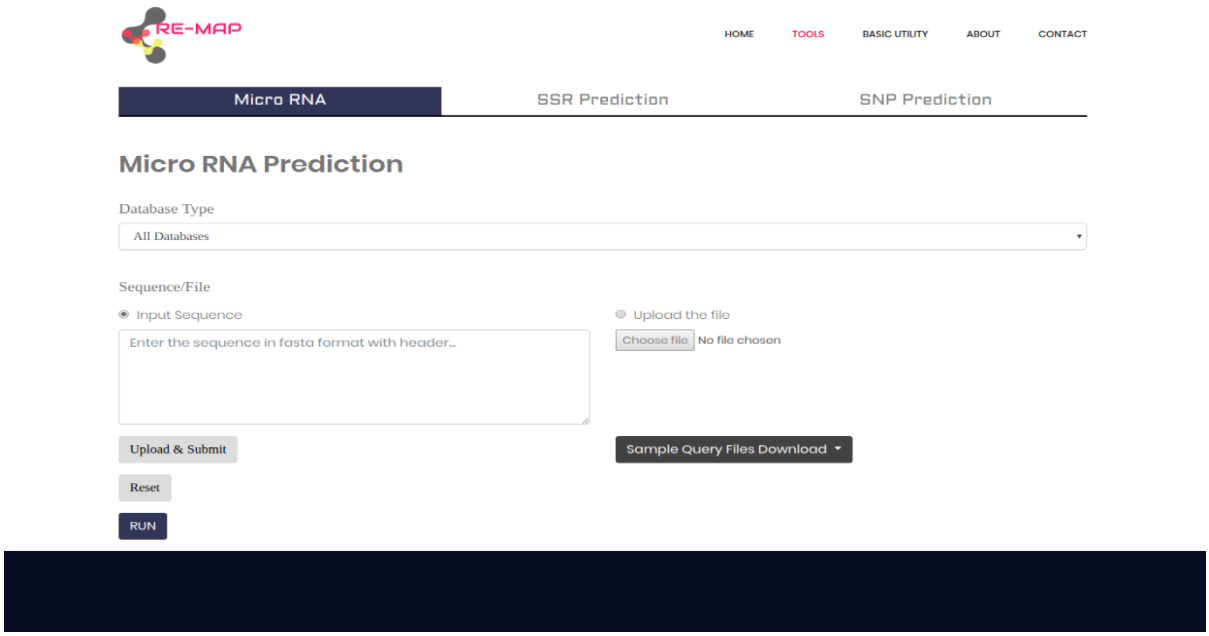


Figure 11: miRNA module

3.1.1.2 Result obtained from miRNA module

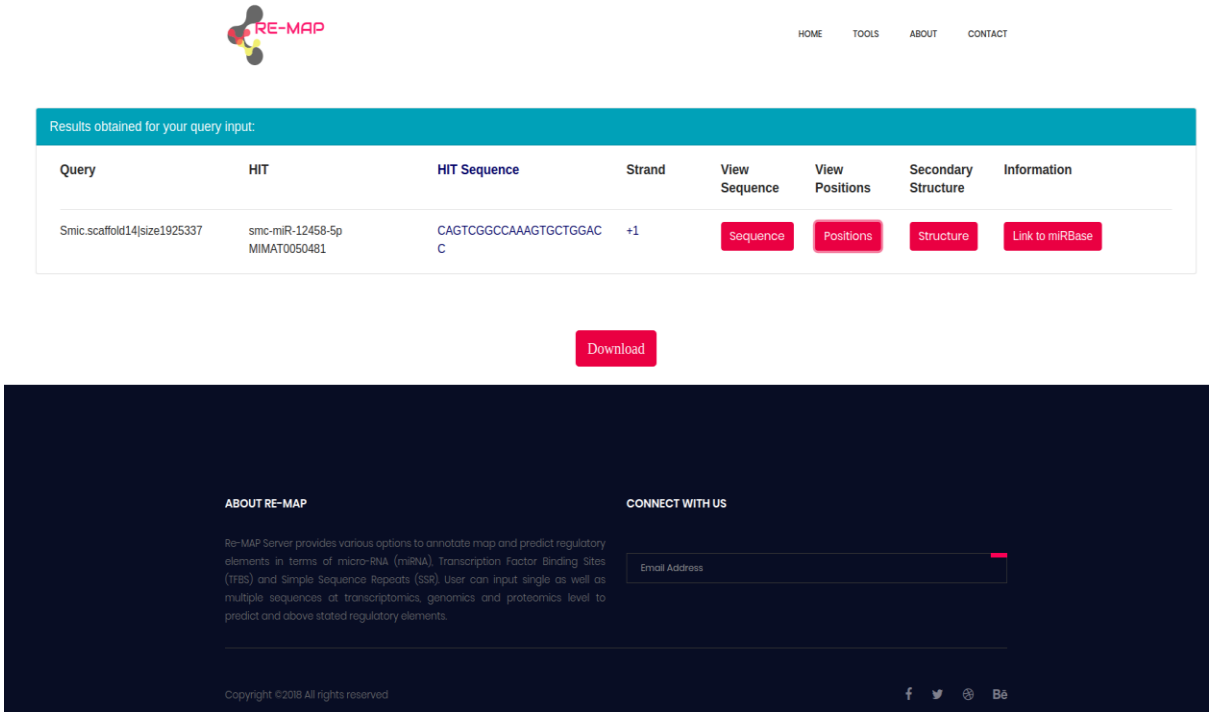
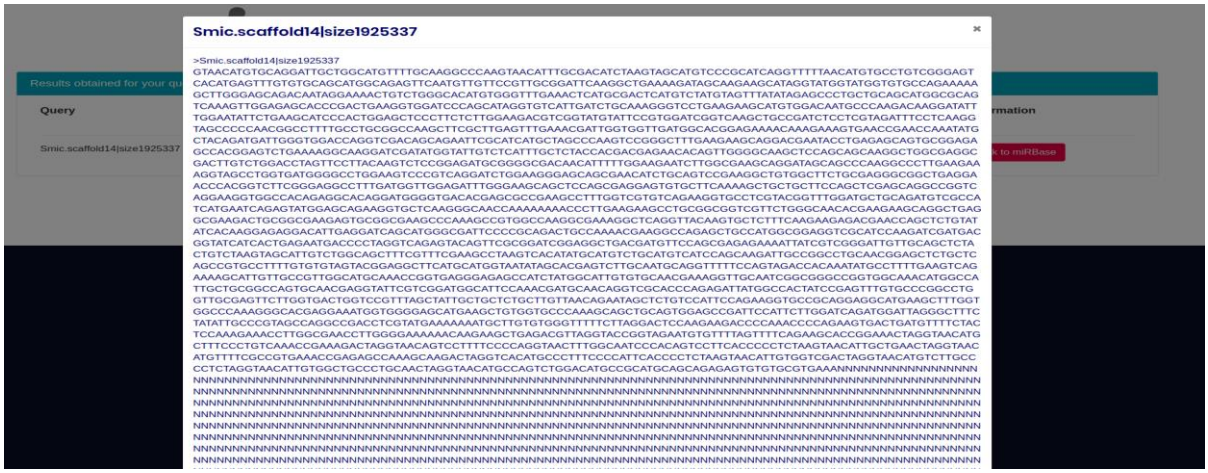
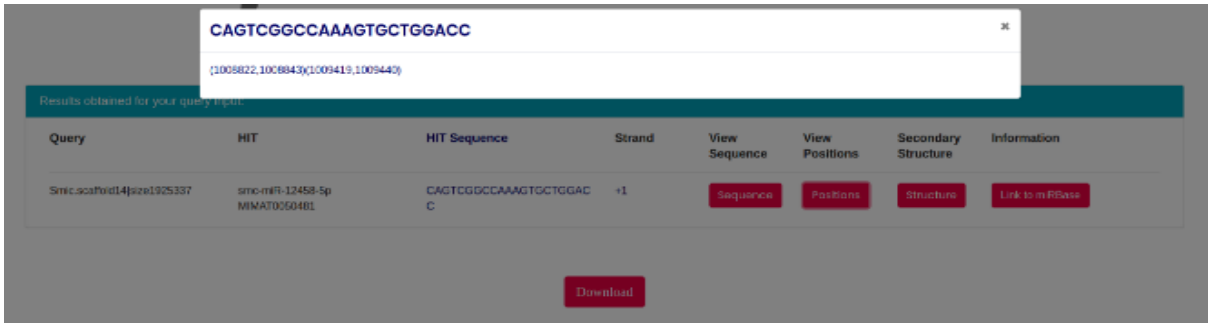


Figure 12: Results from miRNA module



(a)



(b)

(c)

Figure 13: (a) Sequence that contains miRNA target; (b) Start and end position of the miRNA in the query sequence; (c) miRNA structure visualization (Dynamic)

3.1.2 Simple Sequence Repeats Module

In SSR module, user needs to upload or input his query sequence FASTA format file (.fa extension named file). Once that is done user needs to hit upload and run button to fetch results for their input query. Simple sequence repeats Perl script is run in the background and output obtained regarding frequency and most abundant repeats present in the query sequence(s).

For mononucleotides, despite the fact that A, T, C and G are conceivable, A and T are assembled into a single classification, G and C as A reoccur on a strand is equivalent to a T reoccur on the contrary strand, and a C on a strand is equivalent to a G on the contrary strand, bringing about two one of a kind classes of mononucleotides, A/T and C/G all dinucleotide motifs were gathered into the accompanying four special classes I AT/The repeats of trinucleotides are assembled into 10 interesting AAG/TTC classes containing AAG/AGA/GAA/CTT/TTC/TCT SSRs.

3.1.2.1 Input GUI for SSR module

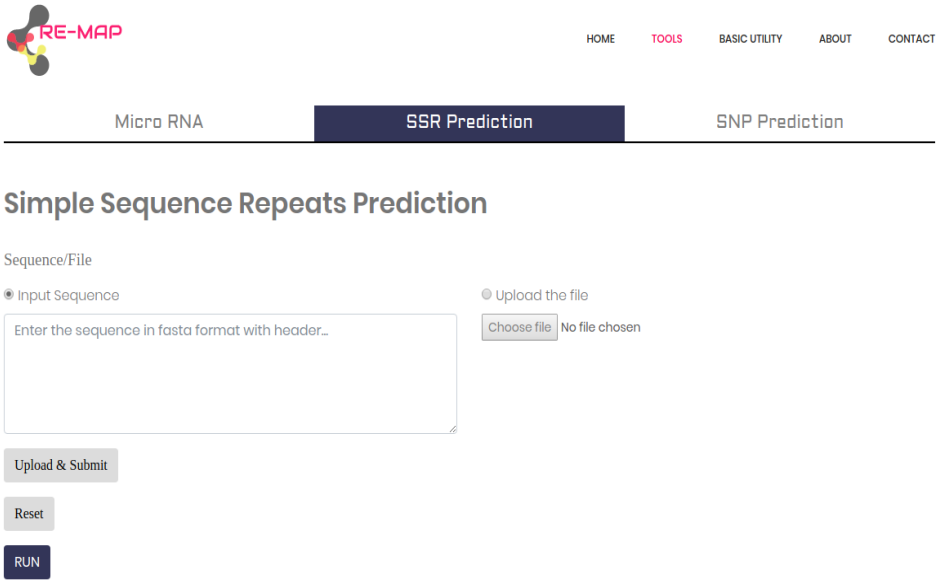


Figure 14: SSR module

3.1.2.2 Result obtained from SSR module:

ID	SSR Nr.	SSR Type	SSR	SSR Size	SSR Start	SSR End
seq1	1	c	(CT)8cgatcgagatcgatggc(CAT)6	51	21	71
seq2	1	c	(G)13cgctatacgcgctcggagagaga(TC)6ttatagagatcgatcgactagctagatataag(ACTC)6	104	23	126

(a)

(b)

Figure 15 (a) & (b): Results from SSR module

3.1.3 Single Nucleotide Polymorphism Module

In SNP module, user needs to upload or input his query sequence FASTA format file (.fa extension named file). Once that is done user needs to hit upload and run button to fetch results for their input query. Single nucleotide polymorphism python script is run in the background and output obtained in the form of VCF format. VCF is a text file format (the most packed way that is available). It incorporates meta-data lines, a header line, and in this way information lines each containing data about a genome position.

The header line names the 8 segments that are fixed and required. These are the following columns:

1. “CHROM”
2. “POS”
3. “ID”
4. “REF”
5. “ALT”
6. “QUAL”
7. “FILTER”
8. “INFO”

On the off chance that genotype information is available in the record, at that point a self-assertive number of test IDs is trailed by a FORMAT segment header. There is a tab-delimited header line.

1. CHROM chromosome: a reference genome identifier.
2. POS position: reference position, with position 1 of the a respectable starting point. Inside each reference arrangement CHROM, positions are arranged numerically, in expanding request.
3. Semi-colon isolated rundown of exceptional identifiers ID where accessible.
4. REFdatabase(s): Each base must be A, C, G, T, N.
5. Comma isolated ALT rundown of option non-reference alleles approached something like one of the examples. CHROM chromosome: an identifier from the reference genome.
6. QUAL phred-scaled quality score for the ALT guarantee. For example give - 10log 10 prob(it's inappropriate to call ALT). On the off chance that ALT is." "(no variation), it is - 10log 10 p(variant), and if ALT isn't" "- 10log 10 p(no variation). High scores of QUAL demonstrate high calls of trust.
7. FILTER channel: PASS if all channels have passed this position, for example at this position a call is made. Something else, a semicolon-isolated rundown of channel codes will fall flat if the site has not passed all channels. For instance, "q10;s50" could demonstrate that the quality at this site is underneath 10 and the quantity of information tests is beneath half of the all out number of tests.
8. INFO Additional Information: INFO fields are encoded as a semi-colon shorter key arrangement with discretionary configuration esteems: < key>=<data>[,data].

3.1.3.1 Input GUI for SNP module

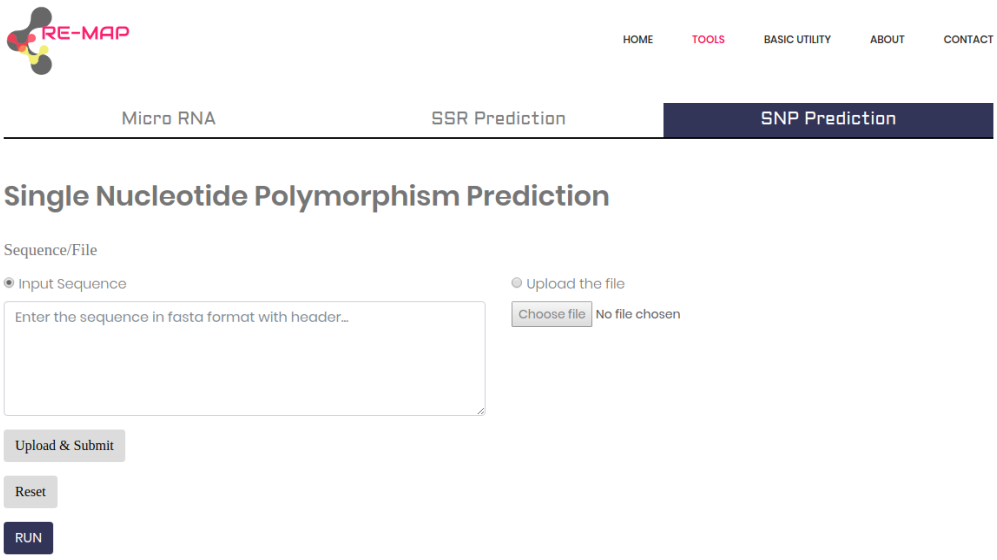


Figure 16: SNP module

3.1.3.2 Result obtained from SNP module:

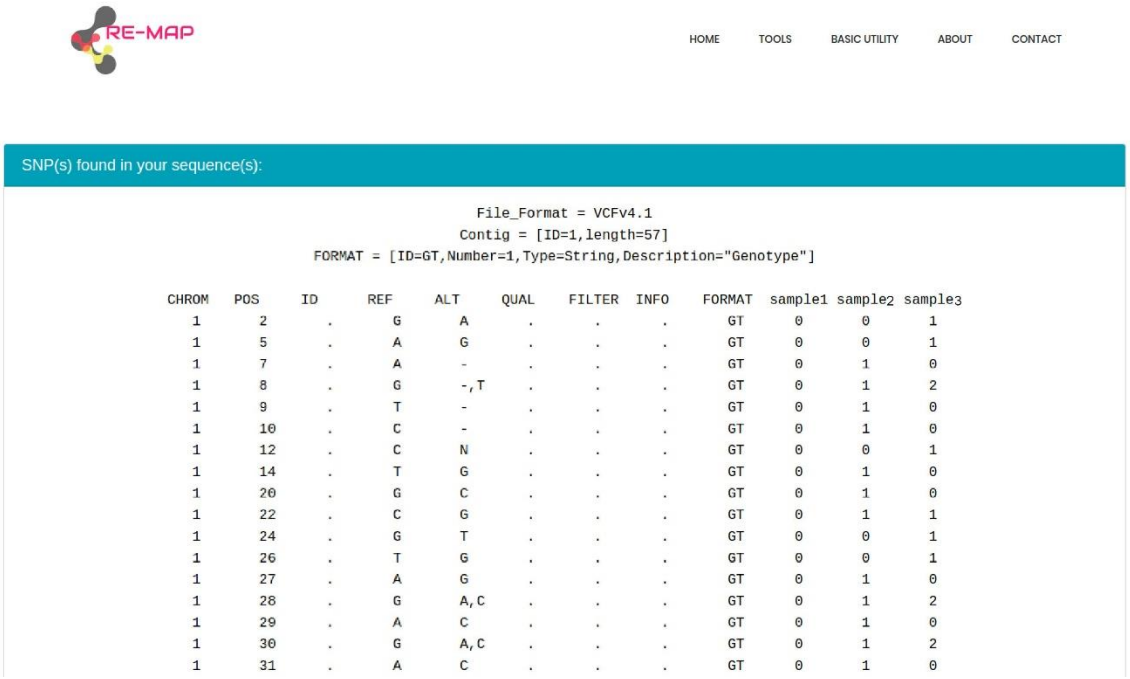
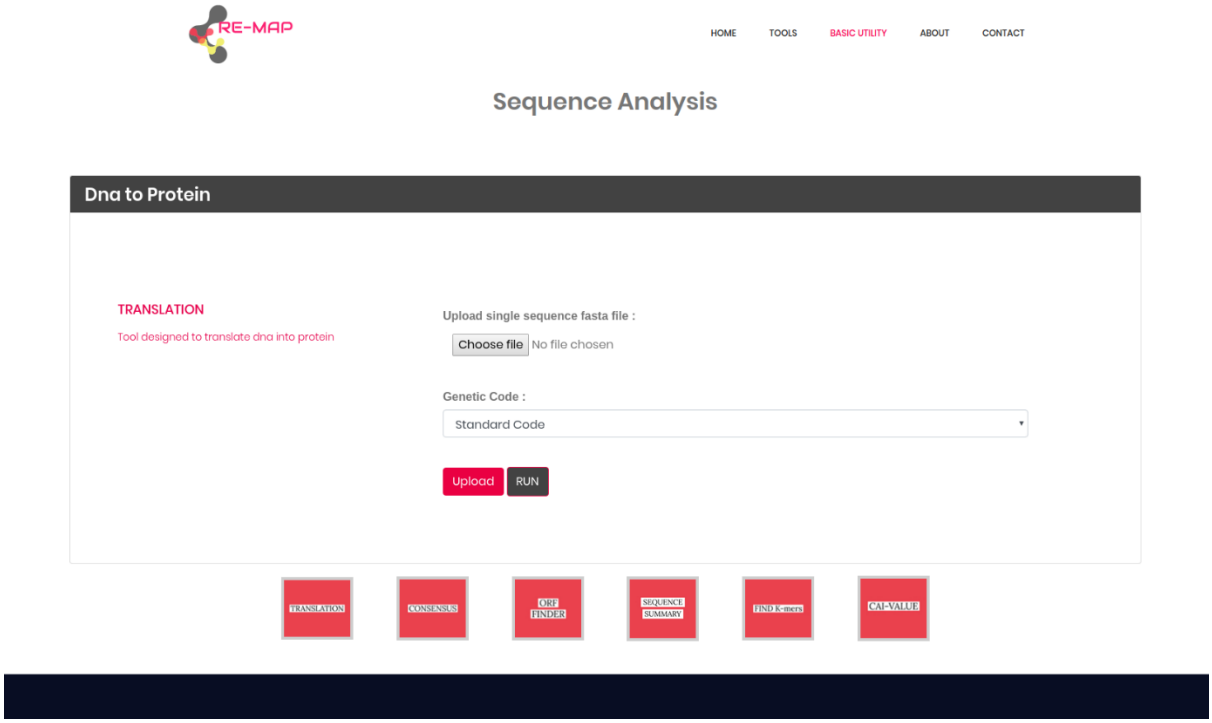


Figure 17: Results from SNP module

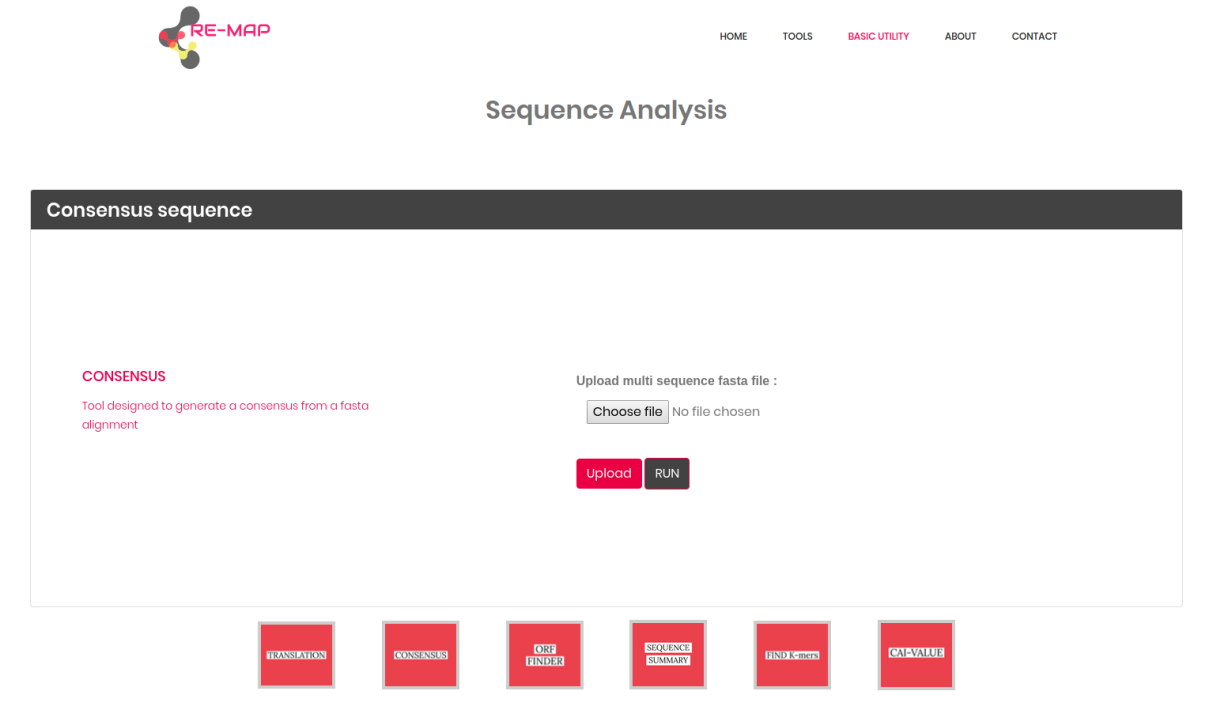
3.1.4 Sequence Analysis Module

In sequence analysis module, we have provided multiple tools for the analysis based on nucleotide sequence(s). User can select any tool he/she is interested in by clicking on the respected tool named button. Once it's done, user needs to upload or input his query sequence FASTA format file (.fa extension named file) as per the requirement by the tool and needs to hit upload and run button to fetch results for their input query. Python/Perl scripts are run in the background and output obtained is rendered on to the web page.

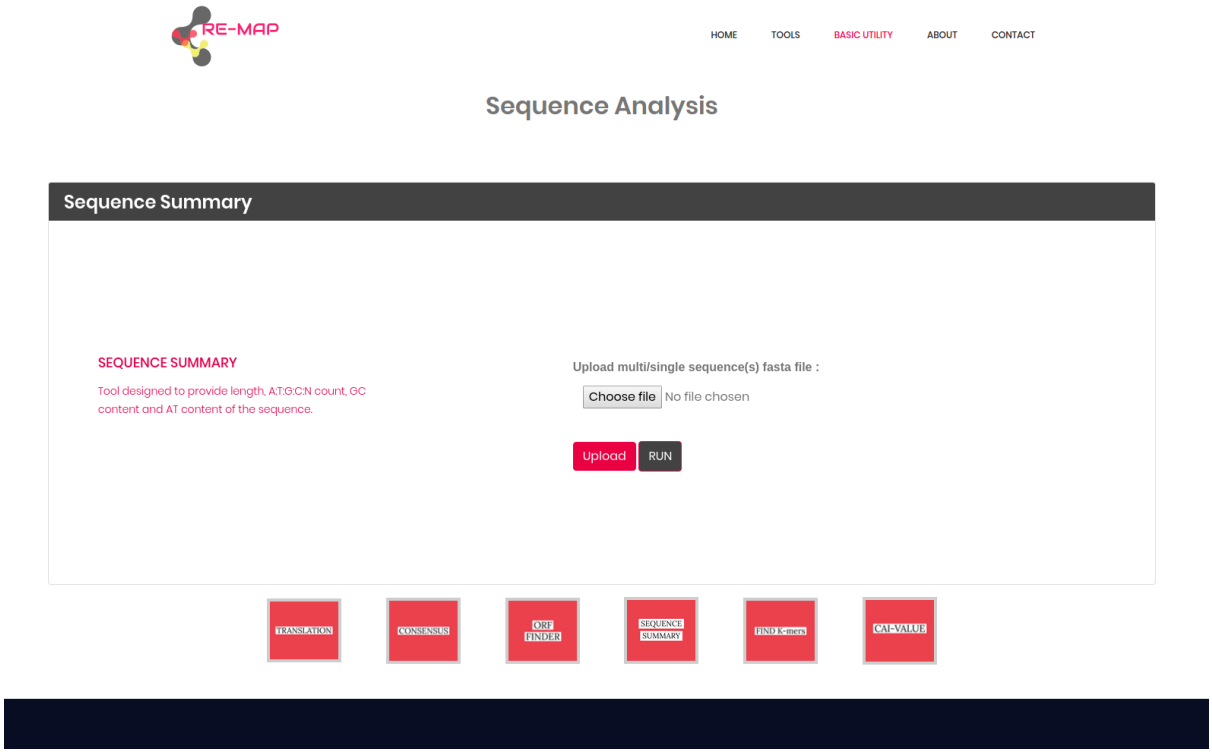
3.1.4.1 Input GUI for sequence analysis module



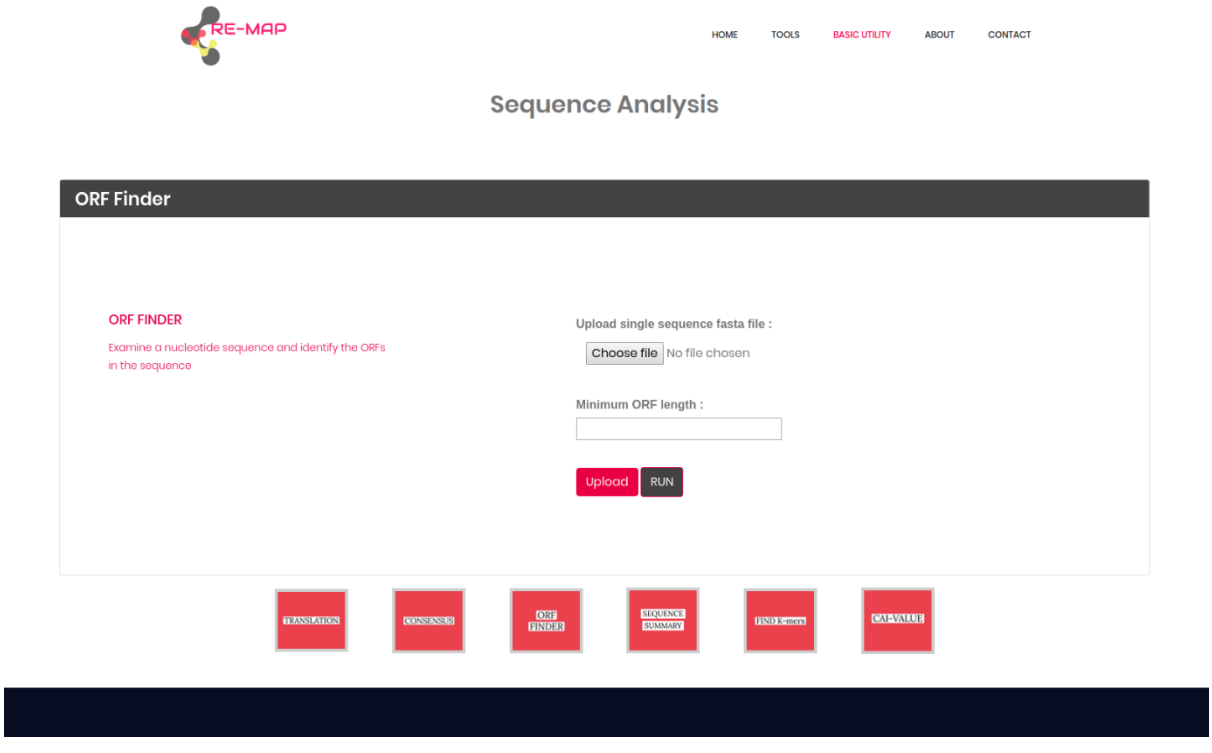
(a)



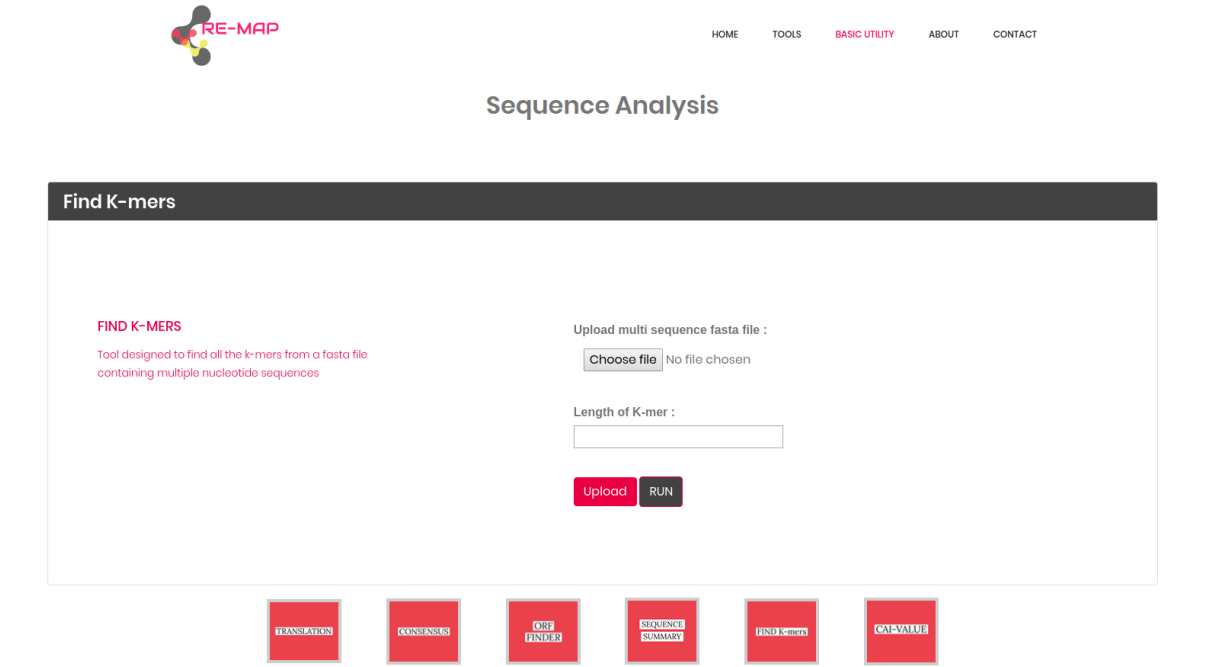
(b)


RE-MAP
HOME
TOOLS
BASIC UTILITY
ABOUT
CONTACT
Sequence Analysis
Sequence Summary
SEQUENCE SUMMARY
Tool designed to provide length, A:T:G:C:N count, GC content and AT content of the sequence.
Upload multi/single sequence(s) fasta file :
Choose file
No file chosen
Upload
RUN
TRANSLATION
CONSENSUS
ORF FINDER
SEQUENCE SUMMARY
FIND K-mers
CAI-VALUE

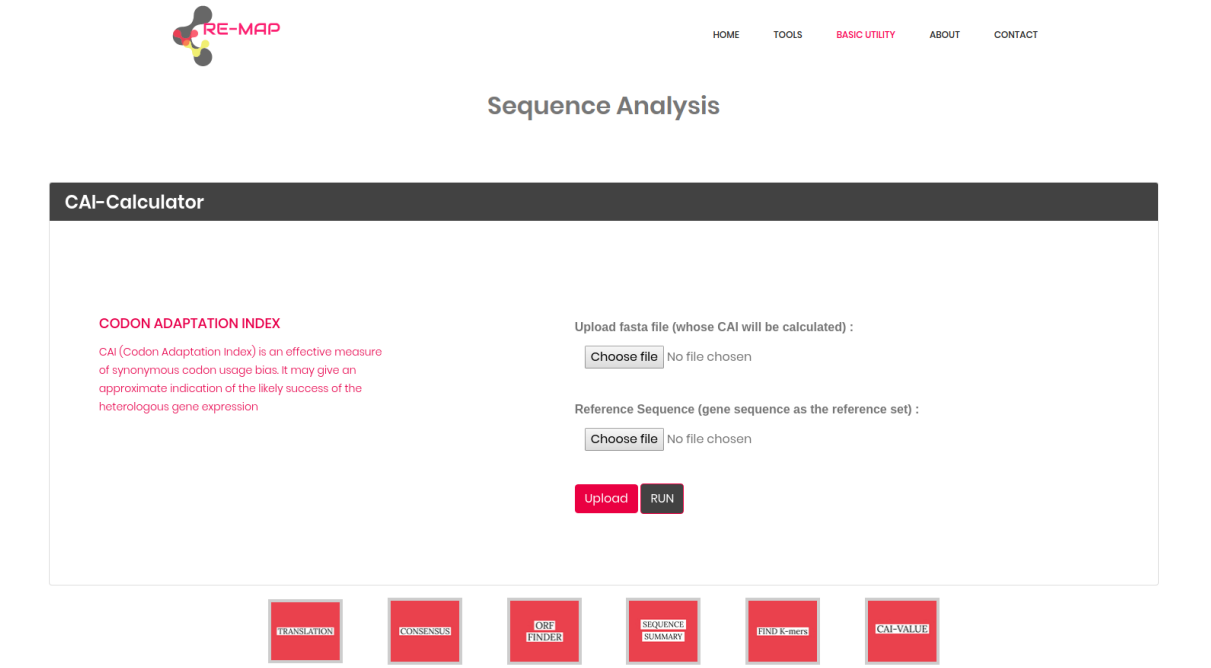
(c)


RE-MAP
HOME
TOOLS
BASIC UTILITY
ABOUT
CONTACT
Sequence Analysis
ORF Finder
ORF FINDER
Examine a nucleotide sequence and identify the ORFs in the sequence
Upload single sequence fasta file :
Choose file
No file chosen
Minimum ORF length :
Upload
RUN
TRANSLATION
CONSENSUS
ORF FINDER
SEQUENCE SUMMARY
FIND K-mers
CAI-VALUE

(d)



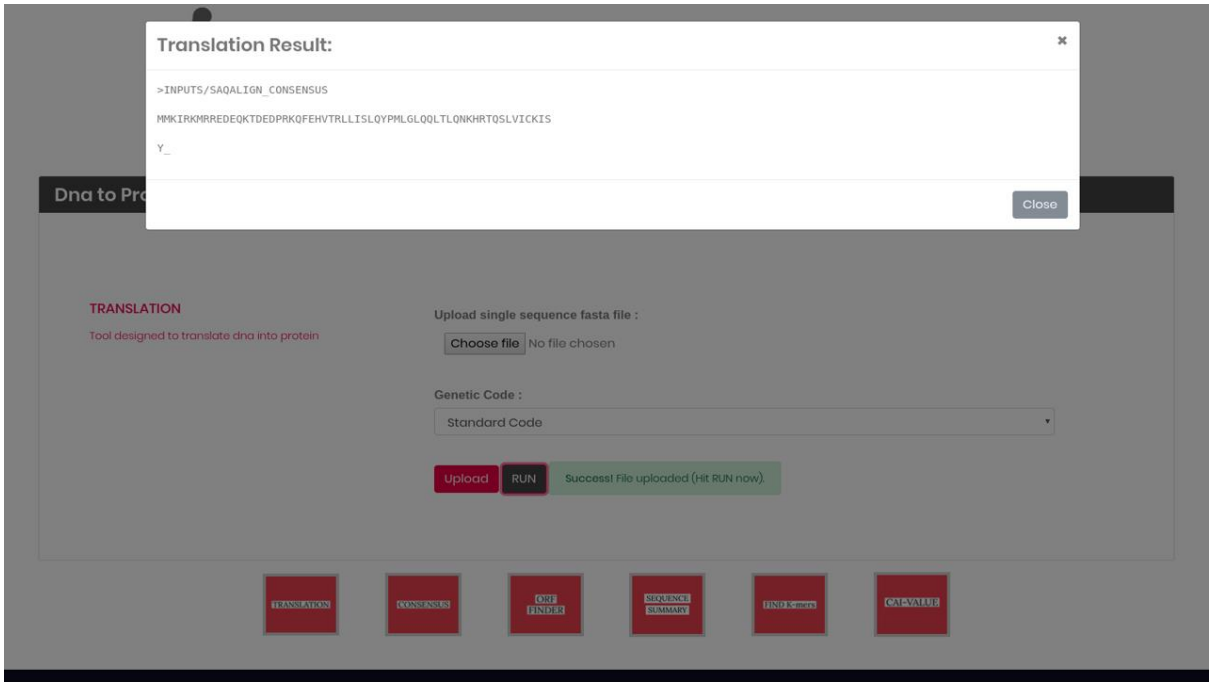
(e)



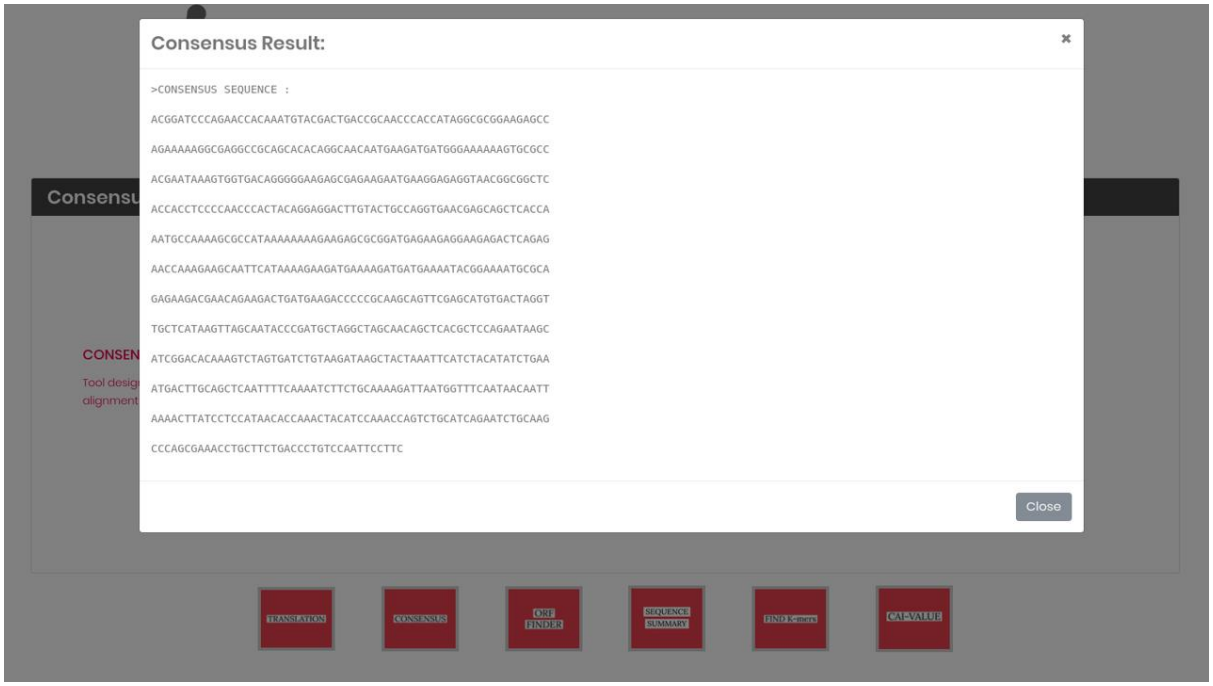
(f)

Figure 18: Showing GUI of different tools present on the web server for the analysis of nucleotide sequences (a) Translation; (b) Consensus; (c) Sequence Summary; (d) ORF Finder; (e) Find K-mers; (f) CAI – Calculator.

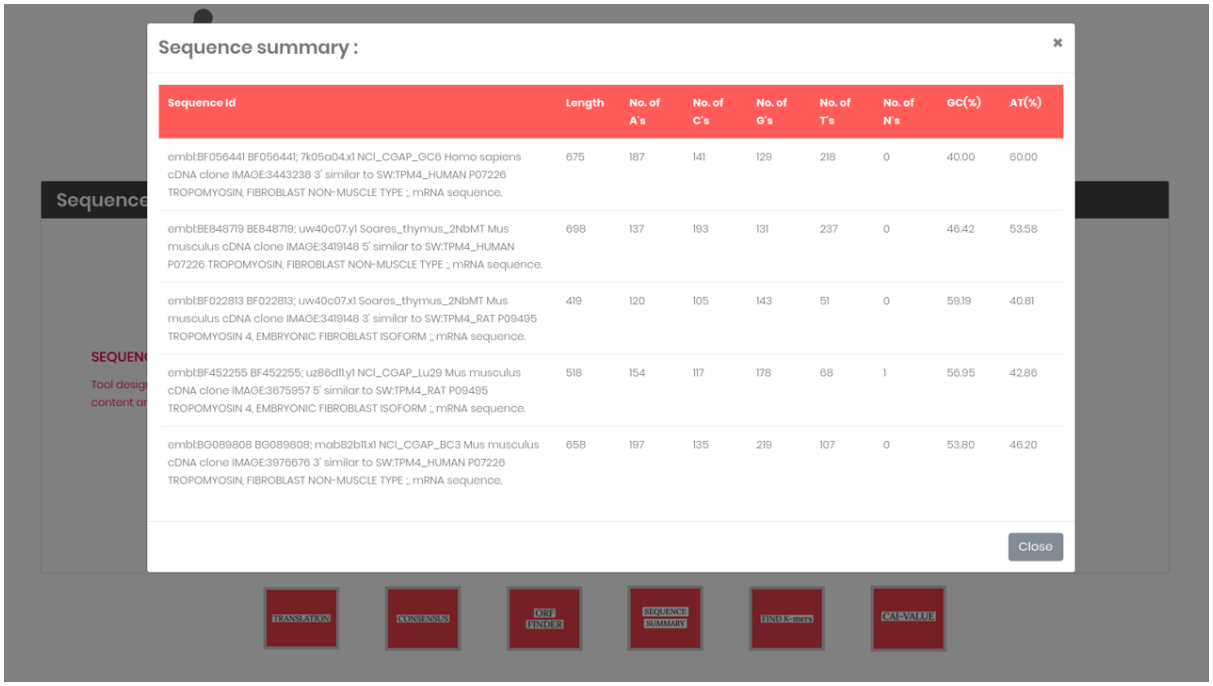
3.1.4.2 Result obtained from sequence analysis module:



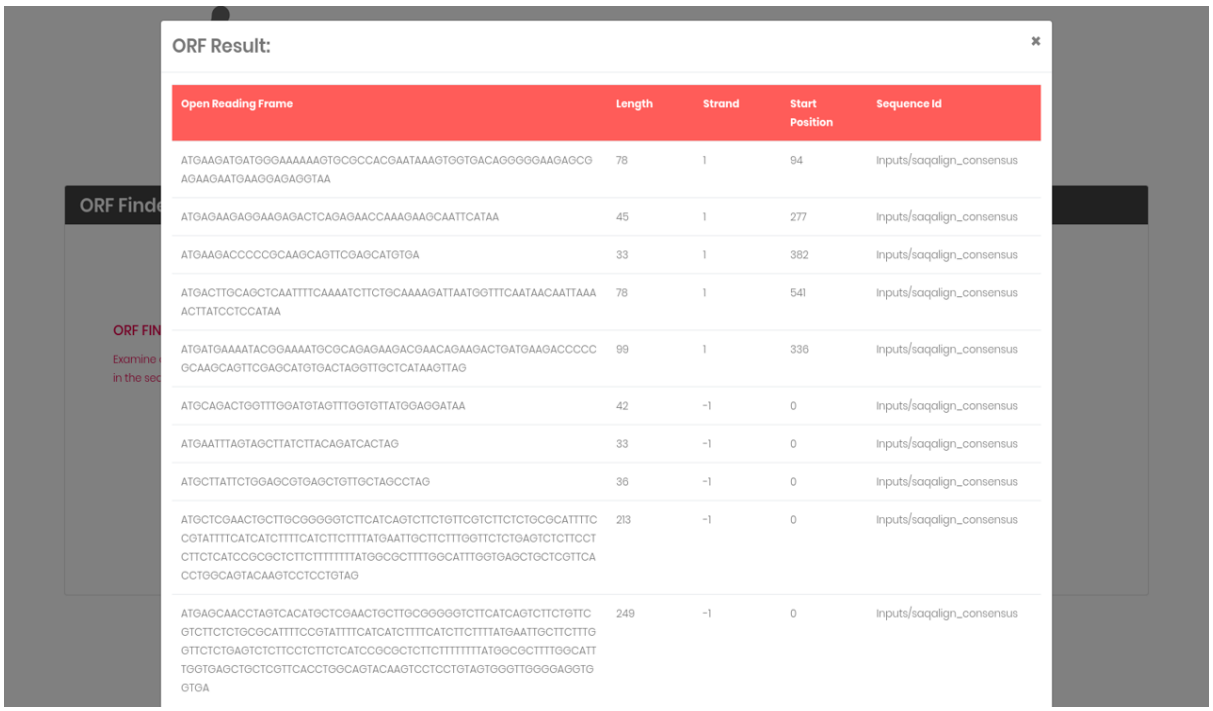
(a)



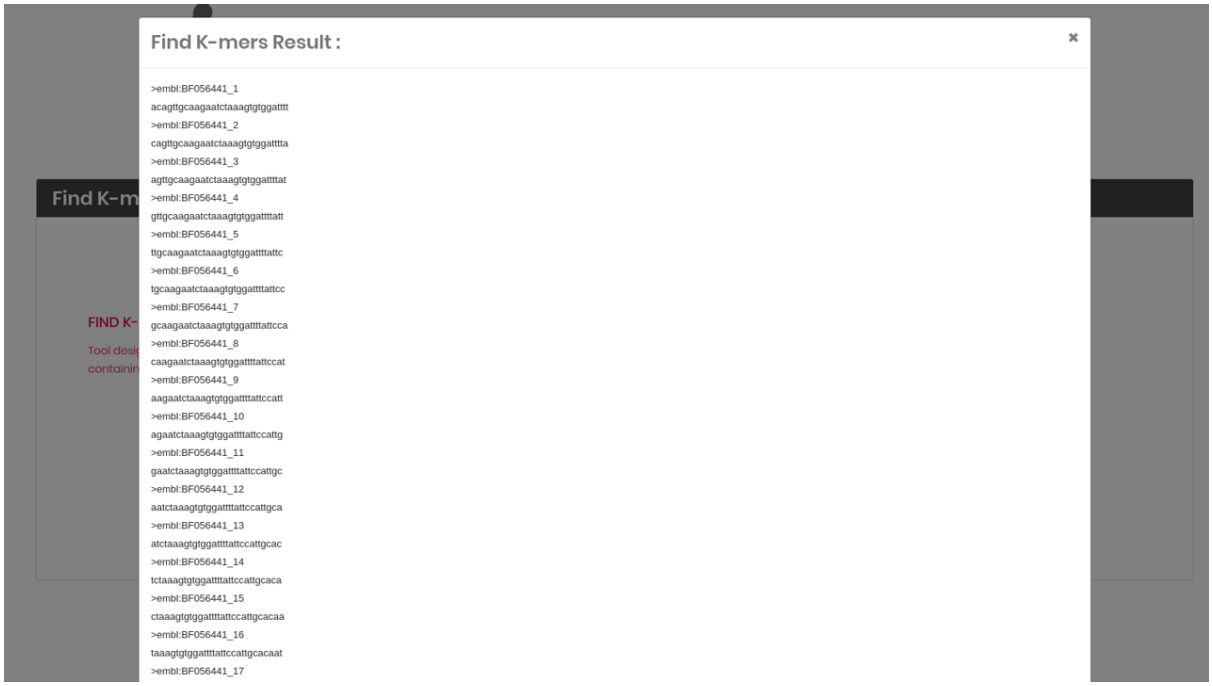
(b)


Sequence summary :
Sequence Id
Length
No. of A's
No. of C's
No. of G's
No. of T's
No. of N's
GC(%)
AT(%)
embl:BF056441 BF056441; 7k05a04.x1 NCI_CGAP_GC6 Homo sapiens cDNA clone IMAGE:3443238 3' similar to SW:TPM4_HUMAN P07226 TROPOMYOSIN, FIBROBLAST NON-MUSCLE TYPE ;, mRNA sequence.
675 187 141 129 218 0 40.00 60.00
embl:BE848719 BE848719; uw40c07.y1 Soares_thymus_2NbMT Mus musculus cDNA clone IMAGE:3419148 5' similar to SW:TPM4_HUMAN P07226 TROPOMYOSIN, FIBROBLAST NON-MUSCLE TYPE ;, mRNA sequence.
698 137 193 131 237 0 46.42 53.58
embl:BF022813 BF022813; uw40c07.x1 Soares_thymus_2NbMT Mus musculus cDNA clone IMAGE:3419148 3' similar to SW:TPM4_RAT P09495 TROPOMYOSIN 4, EMBRYONIC FIBROBLAST ISOFORM ;, mRNA sequence.
419 120 105 143 51 0 59.19 40.81
embl:BF452255 BF452255; uz86d11.y1 NCI_CGAP_Lu29 Mus musculus cDNA clone IMAGE:3675957 5' similar to SW:TPM4_RAT P09495 TROPOMYOSIN 4, EMBRYONIC FIBROBLAST ISOFORM ;, mRNA sequence.
518 154 117 178 68 1 56.95 42.86
embl:BG089808 BG089808; mab82b11.x1 NCI_CGAP_BC3 Mus musculus cDNA clone IMAGE:3976676 3' similar to SW:TPM4_HUMAN P07226 TROPOMYOSIN, FIBROBLAST NON-MUSCLE TYPE ;, mRNA sequence.
658 197 135 219 107 0 53.80 46.20
Close
TRANSLATION
CONSENSUS
ORF FINDER
SEQUENCE SUMMARY
FIND K-mers
CAI-VALUE

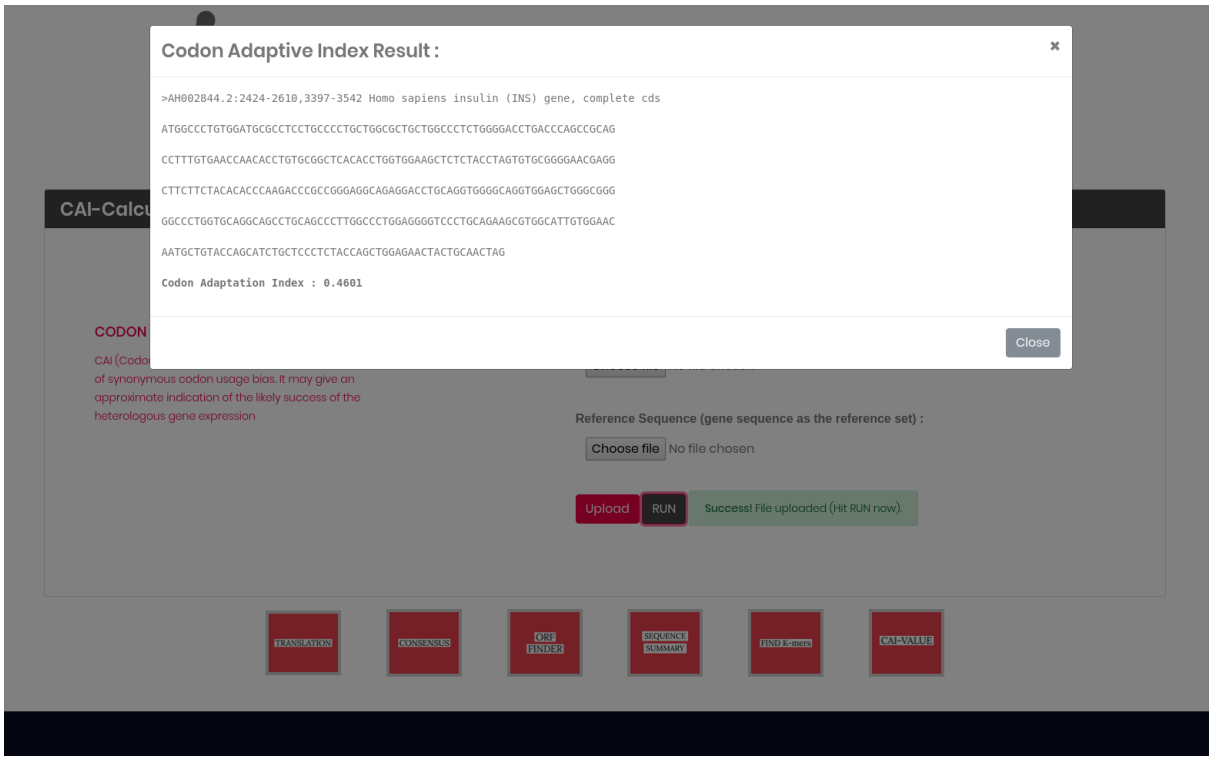
(c)


ORF Result:
Open Reading Frame
Length
Strand
Start Position
Sequence Id
ATGAAGATGATGGGAAAAAAGTGCGCCACGAATAAAGTGGTGACAGGGGGAAGAGCGAGAAGAATGAAGGAGAGGTAA
78 1 94 Inputs/saqalign_consensus
ATGAGAAGAGGAAGAGACTCAGAGAACCAAAGAAGCAATTCATAA
45 1 277 Inputs/saqalign_consensus
ATGAAGACCCCCGCAAGCAGTTCGAGCATGTGA
33 1 382 Inputs/saqalign_consensus
ATGACTTGCAGCTCAATTTTCAAAATCTTCTGCAAAAGATTAATGGTTTCAATAACAATTAAAACTTATCCTCCATAA
78 1 541 Inputs/saqalign_consensus
ATGATGAAAATACGGAAAATGCGCAGAGAAGACGAACAGAAGACTGATGAAGACCCCCGCAAGCAGTTCGAGCATGTGACTAGGTTGCTCATAAGTTAG
99 1 336 Inputs/saqalign_consensus
ATGCAGACTGGTTTGGATGTAGTTTGGTGTTATGGAGGATAA
42 -1 0 Inputs/saqalign_consensus
ATGAATTTAGTAGCTTATCTTACAGATCACTAG
33 -1 0 Inputs/saqalign_consensus
ATGCTTATTCTGGAGCGTGAGCTGTTGCTAGCCTAG
36 -1 0 Inputs/saqalign_consensus
ATGCTCGAACTGCTTGCGGGGGTCTTCATCAGTCTTCTGTTCGTCTTCTCTGCGCATTTTCCGTATTTTCATCATCTTTTCATCTTCTTTTATGAATTGCTTCTTTGGTTCTCTGAGTCTCTTCCTCTTCTCATCCGCGCTCTTCTTTTTTTTATGGCGCTTTTGGCATTTGGTGAGCTGCTCGTTCACCTGGCAGTACAAGTCCTCCTGTAGTGGGTTGGGGAGGTGGTGA
213 -1 0 Inputs/saqalign_consensus
ATGAGCAACCTAGTCACATGCTCGAACTGCTTGCGGGGGTCTTCATCAGTCTTCTGTTCGTCTTCTCTGCGCATTTTCCGTATTTTCATCATCTTTTCATCTTCTTTTATGAATTGCTTCTTTGGTTCTCTGAGTCTCTTCCTCTTCTCATCCGCGCTCTTCTTTTTTTTATGGCGCTTTTGGCATTTGGTGAGCTGCTCGTTCACCTGGCAGTACAAGTCCTCCTGTAGTGGGTTGGGGAGGTGGTGA
249 -1 0 Inputs/saqalign_consensus

(d)



(e)



(f)

Figure 19: Images showing results produced by different tools present on the web server for the analysis of nucleotide sequences (a) Translation; (b) Consensus; (c) Sequence Summary; (d) ORF Finder; (e) Find K-mers; (f) CAI – Calculator.

Chapter 4 - CONCLUSION

We have effectively built up a web server named as REMAP – Regulatory Elements Mapping and Prediction, where a client can upload a FASTA format file containing nucleotide sequence(s) to carry out Regulatory Elements Analysis (REA). The purposed web server is easy to use and results are produced in a sensible measure of time. Contrasting REMAP and other existing servers that perform regulatory element analysis, we reason that REMAP gives one stop arrangement by incorporating major regulatory elements predicting tools at one place with an addon sequence analysis package.

The main contributions of this work are as per the following: (i) the gathering of the datasets for miRNA prediction and mining literatures for additional data concerning regulatory elements; (ii) the incorporation of multiple regulatory elements identification tools; (iii) giving instinctive interface to encourage the presentation of abundant information provided within the proposed REMAP; (iv) giving a helpful response to thoroughly clarifying the nucleotide sequences with the assistance of sequence analysis module integrated within the purposed web server REMAP.

As of now there is no choice provided for choosing a particular tool in particular module e.g. list of multiple tools provided for miRNA prediction, among which user can select one of his own choice and can proceed further with it.

Thus, the future work may involve:

- Adding more tools in each section.
- Covering more regulatory elements.
- Keep miRNA database up to date as we don’t want our server to get outdated.

Appendix - I

Perl script for miRNA identification in query nucleotide sequence(s):

```
#!usr/bin/perl

# $f=$ARGV[0]; #Command line argument, give name of transcript file, example : perl sc
.pl PKS-15 transcripts.fasta
$f = $ARGV[0];
$database = $ARGV[1];

@array=split(/\//,$f);
open(FH,$database) or die "cannot open file"; #input mirna file, contains mirna from t
he selected plants
open(FH1,$f) or die "cannot open file1";

open (WH,">scriptoutput/$array[-
1] results.txt"); #Output file in the format "transcript.fasta results.txt"

@arr=<FH>;
@art=<FH1>;

print "Size of mirna file : ".$#arr."\n";
print "Size of transcript file :".$#art."\n";
$count=0;
foreach $s(@arr)
{
 print "current : ".$count."\n";
 $count++;
 #print $s;getc;
 #if($count%1000==0){print "current=$count\n";}
 if($s =~ /^>/){chomp($curt=$s);}
 else
 {
  @seq=();
  chomp($mir=$s);
  $mir =~ tr/U/T/;  #### replacing U with T
  $mirv= reverse($mir); #### reverse
  #print $mir;getc;
  foreach $str(@art)
  {
   #print $str;getc;
   if($str =~ /^>/){$cur=$str;}
   else
   {
    if($str =~ /$mir/)   ### Searching for exact match through regex
    {
     my @matches;
     @positionF=();
     while ($str =~ /$mir/g) {
    @temp=();
    push @matches, $1;
        my $startpositionF = $-[0]+1;
        my $endpositionF=length($mir)+$startpositionF-1;
    #print $startpositionF."\n";
    push @temp,"(";
    push @temp,$startpositionF;
    push @temp,",";
    push @temp,$endpositionF;
    push @temp,")";
    #print @temp;
    print "\n";
    push @positionF,@temp;
     }
```

```
58.      print "Found $curt---$cur";
59.      #print WH $curt."\t".$cur."\t".$mir."\t"."+1"."\t"."(".$startpositionF.",".$endpo
    sitionF.")"."\n";
60.      print WH $curt."\t".$cur."\t".$mir."\t"."+1"."\t";
61.      print WH @positionF;
62.      print WH "\n";
63.     }
64.     if($str =~ /$mirv/)  ### Same search for the reverse string
65.     {
66.      my @matches;
67.      @positionR=();
68.      while ($str =~ /$mirv/g) {
69.     @temp=();
70.     push @matches, $1;
71.         my $startpositionR = $-[0]+1;
72.         my $endpositionR=length($mirv)+$startpositionR-1;
73.     #print $startpositionF."\n";
74.     push @temp,"(";
75.     push @temp,$startpositionR;
76.     push @temp,",";
77.     push @temp,$endpositionR;
78.     push @temp,")";
79.     #print @temp;
80.     print "\n";
81.     push @positionR,@temp;
82.      }
83.      my $startpositionR = index($str,$mirv) + 1;
84.      my $endpositionR=length($mirv)+$startpositionR-1;
85.      print "Found $curt---$cur";
86.      #print WH $curt."\t".$cur."\t".$mirv."\t"."-
    1"."\t"."(".$startpositionR.",".$endpositionR.")"."\n";
87.      print WH $curt."\t".$cur."\t".$mirv."\t"."-1"."\t";
88.      print WH @positionR;
89.      print WH "\n";
90.     }
91.    }
92.   }
93.  }
94. }
```

Appendix - II

Perl script for microsatellite identification in query nucleotide sequence(s):

```
#!/usr/bin/perl -w
# Open FASTA file #
@new = split(/[\/]/,$ARGV[0]);
$last= $new[-1];
print $last;
open (InputFileHandle,"<$ARGV[0]") || die ("\nError: FASTA file doesn't exist !\n\n");

open (OutputFileHandle,">scriptoutput/$last.misa");
print OutputFileHandle "ID\tSSR nr.\tSSR type\tSSR\tsize\tbeginning\tterminate\n";

# Reading arguments #

open (SPECS,"SSR/misa.ini") || die ("\nError: Specifications file doesn't exist !\n\n"
);
 %typrep;
 $amb = 0;
while (<SPECS>)
   {
   %typrep = $1 =~ /(\d+)/gi if (/^def\S*\s+(.*)/i);
   if (/^int\S*\s+(\d+)/i) {$amb = $1}
   };
 @typ = sort { $a <=> $b } keys %typrep;


#$$$$$ CORE $$$$$#

$/ = ">";
 $high_occurence = 1; #count frequency
 $low_occurence = 1000; #count frequency
 (%count conserved,%count class); #count
 ($number nuc sequenceuences,$size nuc sequenceuences,%ssr containing nuc sequences);
#stores number and size of all nuc_sequenceuences examined
 $ssr in compound = 0;
 ($nuc ident,$nuc sequence);
while (<InputFileHandle>)
  {
  next unless (($nuc ident,$nuc sequence) = /(.*?)\n(.*)/s);
   ($nr,%beginning,@order,%terminate,%conserved,%frequency); # store info of all SSRs
from each nuc_sequenceuence
  $nuc_sequence =~ s/[\d\s>]//g; #remove digits, spaces, line breaks,...
  $nuc ident =~ s/^\s*//g; $nuc ident =~ s/\s*$//g;$nuc ident =~ s/\s/ /g; #replace wh
itespace with " "
  $number_nuc_sequenceuences++;
  $size_nuc_sequenceuences += length $nuc_sequence;
  for ($i=0; $i < scalar(@typ); $i++) #check each conserved class
    {
     $conservedlen = $typ[$i];
     $minreps = $typrep{$typ[$i]} - 1;
    if ($low_occurence > $typrep{$typ[$i]}) {$low_occurence = $typrep{$typ[$i]}}; #cou
nt frequency
     $search = "(([acgt]{$conservedlen})\\2{$minreps,})";
    while ( $nuc_sequence =~ /$search/ig ) #scan whole nuc_sequenceuence for that clas
s
      {
       $conserved = uc $2;
       $copies; #reject false type conserveds [e.g. (TT)6 or (ACAC)5]
      for ($j = $conservedlen - 1; $j > 0; $j--)
        {
         $redconserved = "([ACGT]{$j})\\1{".($conservedlen/$j-1)."}";
        $copies = 1 if ( $conserved =~ /$redconserved/ )
```

```
54.         };
55.       next if $copies;
56.       $conserved{++$nr} = $conserved;
57.        $ssr = uc $1;
58.       $frequency{$nr} = length($ssr) / $conservedlen;
59.       $terminate{$nr} = pos($nuc sequence);
60.       $beginning{$nr} = $terminate{$nr} - length($ssr) + 1;
61.       # count frequency
62.       $count conserveds{$conserved{$nr}}++; #counts occurrence of indivnuc identual co
    nserveds
63.       $conserved{$nr}-
    >{$frequency{$nr}}++; #counts occurrence of specific SSR in its appearing repeat
64.       $count class{$typ[$i]}++; #counts occurrence in each conserved class
65.       if ($high_occurence < $frequency{$nr}) {$high_occurence = $frequency{$nr}};
66.       };
67.     };
68.   next if (!$nr); #no SSRs
69.   $ssr containing nuc sequences{$nr}++;
70.   @order = sort { $beginning{$a} <=> $beginning{$b} } keys %beginning; #put SSRs in ri
    ght order
71.   $i = 0;
72.    $count_nuc_sequence; #counts
73.    ($beginning,$terminate,$ssrnuc sequence,$ssrclasses,$size);
74.   while ($i < $nr)
75.     {
76.      $space = $amb + 1;
77.     if (!$order[$i+1]) #last or only SSR
78.       {
79.       $count nuc sequence++;
80.        $conservedlen = length ($conserved{$order[$i]});
81.       $ssrclasses = "p".$conservedlen;
82.       $ssrnuc sequence = "($conserved{$order[$i]})$frequency{$order[$i]}";
83.       $beginning = $beginning{$order[$i]}; $terminate = $terminate{$order[$i++]};
84.       next
85.       };
86.     if (($beginning{$order[$i+1]} - $terminate{$order[$i]}) > $space)
87.       {
88.       $count_nuc_sequence++;
89.        $conservedlen = length ($conserved{$order[$i]});
90.       $ssrclasses = "p".$conservedlen;
91.       $ssrnuc sequence = "($conserved{$order[$i]})$frequency{$order[$i]}";
92.       $beginning = $beginning{$order[$i]}; $terminate = $terminate{$order[$i++]};
93.       next
94.       };
95.      ($interssr);
96.     if (($beginning{$order[$i+1]} - $terminate{$order[$i]}) < 1)
97.       {
98.       $count nuc sequence++; $ssr in compound++;
99.       $ssrclasses = 'c*';
100.             $ssrnuc_sequence = "($conserved{$order[$i]})$frequency{$order[$i]}($conse
    rved{$order[$i+1]})$frequency{$order[$i+1]}*";
101.             $beginning = $beginning{$order[$i]}; $terminate = $terminate{$order[$i+1]
    }
102.             }
103.           else
104.             {
105.             $count_nuc_sequence++; $ssr_in_compound++;
106.             $interssr = lc substr($nuc_sequence,$terminate{$order[$i]},($beginning{$o
    rder[$i+1]} - $terminate{$order[$i]}) - 1);
107.             $ssrclasses = 'c';
108.             $ssrnuc_sequence = "($conserved{$order[$i]})$frequency{$order[$i]}$inters
    sr($conserved{$order[$i+1]})$frequency{$order[$i+1]}";
109.             $beginning = $beginning{$order[$i]};  $terminate = $terminate{$order[$i+1
    ]};
110.             #$space -= length $interssr
111.             };
```

```
112.           while ($order[++$i + 1] and (($beginning{$order[$i+1]} - $terminate{$order[
    $i]}) <= $space))
113.             {
114.             if (($beginning{$order[$i+1]} - $terminate{$order[$i]}) < 1)
115.               {
116.               $ssr in compound++;
117.               $ssrnuc_sequence .= "($conserved{$order[$i+1]})$frequency{$order[$i+1]}
    *";
118.               $ssrclasses = 'c*';
119.               $terminate = $terminate{$order[$i+1]}
120.               }
121.             else
122.               {
123.               $ssr in compound++;
124.               $interssr = lc substr($nuc_sequence,$terminate{$order[$i]},($beginning{
    $order[$i+1]} - $terminate{$order[$i]}) - 1);
125.               $ssrnuc sequence .= "$interssr($conserved{$order[$i+1]})$frequency{$ord
    er[$i+1]}";
126.               $terminate = $terminate{$order[$i+1]};
127.               #$space -= length $interssr
128.               }
129.             };
130.           $i++;
131.           }
132.         continue
133.           {
134.           print OutputFileHandle "$nuc_ident\t$count_nuc_sequence\t$ssrclasses\t$ssrn
    uc sequence\t",($terminate - $beginning + 1),"\t$beginning\t$terminate\n"
135.           };
136.         };
137.
138.       close (OutputFileHandle);
139.       open (OutputFileHandle,">scriptoutput/$last.statistics");
140.
141.       #$$$$$ InputFileHandleFO $$$$$#
142.
143.       #$$$ Specifications $$$#
144.       print OutputFileHandle "Specifications\n==============\n\nSequence source file:
     \"$last\"\n\nDefinement of microsatellites (unit size / minimum number of frequency):
    \n";
145.       for ($i = 0; $i < scalar (@typ); $i++) {print OutputFileHandle "($typ[$i]/$typr
    ep{$typ[$i]}) "};print OutputFileHandle "\n";
146.       if ($amb > 0) {print OutputFileHandle "\nMaximal number of bases interrupting 2
     SSRs in a compound microsatellite:  $amb\n"};
147.       print OutputFileHandle "\n\n\n";
148.
149.       #$$$ OCCURRENCE OF SSRs $$$#
150.
151.       #small calculations
152.        @ssr_containing_nuc_sequences = values %ssr_containing_nuc_sequences;
153.        $ssr containing nuc sequences = 0;
154.       for ($i = 0; $i < scalar (@ssr_containing_nuc_sequences); $i++) {$ssr_containin
    g_nuc_sequences += $ssr_containing_nuc_sequences[$i]};
155.        @count_conserveds = sort {length ($a) <=> length ($b) || $a cmp $b } keys %cou
    nt conserveds;
156.        @count class = sort { $a <=> $b } keys %count class;
157.       for ($i = 0; $i < scalar (@count_class); $i++) {$total += $count_class{$count_c
    lass[$i]}};
158.
159.       #$$$ Overview $$$#
160.       print OutputFileHandle "RESULTS OF MICROSATELLITE SEARCH\n==================
    ===========\n\n";
161.       print OutputFileHandle "Total number of nuc_sequenceuences examined:
      $number_nuc_sequenceuences\n";
162.       print OutputFileHandle "Total size of examined nuc_sequenceuences (bp):
      $size_nuc_sequenceuences\n";
```

```
163.        print OutputFileHandle "Total number of nuc_idententified SSRs:
     $total\n";
164.        print OutputFileHandle "Number of SSR containing nuc_sequenceuences:
       $ssr_containing_nuc_sequences\n";
165.        print OutputFileHandle "Number of nuc_sequenceuences containing more than 1 SSR
    :  ",$ssr containing nuc sequences - ($ssr containing nuc sequences{1} || 0),"\n";
166.        print OutputFileHandle "Number of SSRs present in compound formation:    $ssr_i
    n_compound\n\n\n";
167.
168.        #§§§ Frequency of SSR classes §§§#
169.        print OutputFileHandle "Distribution to different repeat type classes\n--------
    ------------------------------------\n\n";
170.        print OutputFileHandle "Unit size\t\tNumber of SSRs\n";
171.         $total = undef;
172.        for ($i = 0; $i < scalar (@count_class); $i++) {print OutputFileHandle "$count_
    class[$i]\t\t\t$count class{$count class[$i]}\n"};
173.        print OutputFileHandle "\n";
174.
175.        #§§§ Frequency of SSRs: per conserved and number of frequency §§§#
176.        $outsnuc identevar=sprintf "%-17s","Repeats";
177.        print OutputFileHandle "Frequency of nuc_idententified SSR conserveds\n--------
    -------------------------\n\n$outsnuc_identevar";
178.        for ($i = $low_occurence;$i <= $high_occurence; $i++) {print OutputFileHandle "
    \t$i"};
179.        print OutputFileHandle "\ttotal\n";
180.        for ($i = 0; $i < scalar (@count_conserveds); $i++)
181.          {
182.           $typ = length ($count conserveds[$i]);
183.          $changevar=sprintf "%-17s",$count conserveds[$i];
184.          print OutputFileHandle $changevar;
185.          for ($j = $low_occurence; $j <= $high_occurence; $j++)
186.            {
187.            if ($j < $typrep{$typ}) {print OutputFileHandle "\t-";next};
188.            if ($count_conserveds[$i]-
    >{$j}) {print OutputFileHandle "\t$count_conserveds[$i]-
    >{$j}"} else {print OutputFileHandle "\t"};
189.            };
190.          print OutputFileHandle "\t$count_conserveds{$count_conserveds[$i]}\n";
191.          };
192.        print OutputFileHandle "\n";
193.
194.        #§§§ Frequency of SSRs: summarizing copies and reverse conserveds §§§#
195.        # Eliminates %count conserveds !
196.        print OutputFileHandle "Frequency of classified repeat types (consnuc_identerin
    g nuc sequenceuence complementary)\n-----------------------------------------------------
    ----------------------\n\n$outsnuc_identevar";
197.         (%red rev,@red rev); # groups
198.        for ($i = 0; $i < scalar (@count_conserveds); $i++)
199.          {
200.          next if ($count_conserveds{$count_conserveds[$i]} eq 'X');
201.           (%group,@group,$red rev); # store copies/reverse conserveds
202.           $reverse conserved = $actual conserved = $actual conserved a = $count conser
    veds[$i];
203.          $reverse_conserved =~ tr/ACGT/TGCA/;
204.          $reverse conserved = reverse $reverse conserved;
205.           $reverse conserved a = $reverse conserved;
206.          for ($j = 0; $j < length ($count_conserveds[$i]); $j++)
207.            {
208.            if ($count_conserveds{$actual_conserved}) {$group{$actual_conserved} = "1";
     $count_conserveds{$actual_conserved}='X'};
209.            if ($count_conserveds{$reverse_conserved}) {$group{$reverse_conserved} = "1
    "; $count conserveds{$reverse conserved}='X'};
210.            $actual conserved =~ s/(.)(.*)/$2$1/;
211.            $reverse_conserved =~ s/(.)(.*)/$2$1/;
212.            $actual_conserved_a = $actual_conserved if ($actual_conserved lt $actual_co
    nserved_a);
```

```
213.           $reverse_conserved_a = $reverse_conserved if ($reverse_conserved lt $revers
    e conserved a)
214.           };
215.         if ($actual_conserved_a lt $reverse_conserved_a) {$red_rev = "$actual_conserv
    ed a/$reverse conserved a"}
216.         else {$red_rev = "$reverse_conserved_a/$actual_conserved_a"}; # group name
217.         $red_rev{$red_rev}++;
218.         @group = keys %group;
219.         for ($j = 0; $j < scalar (@group); $j++)
220.           {
221.           for ($k = $low_occurence; $k <= $high_occurence; $k++)
222.             {
223.             if ($group[$j]->{$k}) {$red_rev->{"total"} += $group[$j]->{$k};$red_rev-
    >{$k} += $group[$j]->{$k}}
224.             }
225.           }
226.         };
227.       for ($i = $low_occurence; $i <= $high_occurence; $i++) {print OutputFileHandle
    "\t$i"};
228.       print OutputFileHandle "\ttotal\n";
229.       @red rev = sort {length ($a) <=> length ($b) || $a cmp $b } keys %red rev;
230.       for ($i = 0; $i < scalar (@red_rev); $i++)
231.         {
232.          $typ = (length ($red rev[$i])-1)/2;
233.         $againchangevar=sprintf "%-17s",$red rev[$i];
234.         print OutputFileHandle $againchangevar;
235.         for ($j = $low_occurence; $j <= $high_occurence; $j++)
236.           {
237.           if ($j < $typrep{$typ}) {print OutputFileHandle "\t-";next};
238.           if ($red_rev[$i]->{$j}) {print OutputFileHandle "\t",$red_rev[$i]->{$j}}
239.           else {print OutputFileHandle "\t"}
240.           };
241.         print OutputFileHandle "\t",$red_rev[$i]->{"total"},"\n";
242.         };
```

Appendix - III

Python script for SNPs identification in query nucleotide sequence(s):

```
1.  #!/usr/bin/env python
2.
3.  import argparse
4.  import json
5.  import pyximport
6.  import unittest
7.
8.  from collections import OrderedDict
9.  from cStringIO import StringIO
10.
11. pyximport.install()
12.
13. def write_row(row, output_file):
14.   output file.write("\t".join(map(str, row)) + "\n")
15.
16. def write_header(sequence_names, reference_length, output_file):
17.   args.output.write("""\
18. File Format = VCFv4.1
19. Contig = [ID=1,length=%i]
20. FORMAT = [ID=GT,Number=1,Type=String,Description="Genotype"]
21. \n""" % reference length)
22.   header row = ["CHROM", "POS", "ID", "REF", "ALT", "QUAL", "FILTER", "INFO", "FORMAT"
    ]
23.   header_row += sequence_names
24.   write row(header row, output file)
25.
26. def parse_fasta(input_fasta):
27.   for line in input_fasta:
28.      if line[0] == '>':
29.        break
30.   sequence name = line[1:].rstrip()
31.   sequence_lines = []
32.   for line in input_fasta:
33.     if line[0] == '>':
34.       yield (sequence_name, "".join(sequence_lines))
35.       sequence_name = line[1:].rstrip()
36.       sequence lines = []
37.     else:
38.       sequence_lines.append(line.rstrip())
39.   yield(sequence_name, "".join(sequence_lines))
40.
41. def update_snps(sequence_names, snps, ref_seq, sequence_name, sequence_seq):
42.   sequence_names.append(sequence_name)
43.   for i in xrange(len(ref_seq)):
44.     r,s  = ref seq[i], sequence seq[i]
45.     if r != s:
46.       snps.setdefault(i, []).append((len(sequence_names)-1, chr(s)))
47.
48. BUFFER SIZE = 10*1024*1024
49.
50. if __name__ == '__main__':
51.   parser = argparse.ArgumentParser()
52.   parser.add argument('input', type=argparse.FileType('r', BUFFER SIZE))
53.   parser.add_argument('output', type=argparse.FileType('w'),
54.                       default=open('scriptoutput/random.short.fa.vcf', 'w'))
55.   args = parser.parse args()
56.   sequences = parse fasta(args.input)
57.   ref_name,ref_seq = sequences.next()
58.   snps = {}
59.   sequence_names = []
```

```
60.   sequence names.append(ref name)
61.   for seq_name,seq_seq in sequences:
62.     update_snps(sequence_names, snps, bytearray(ref_seq),
63.                                       seq_name, bytearray(seq_seq))
64.   snps = OrderedDict([(posn, snps[posn]) for posn in sorted(snps.keys())])
65.
66.   write_header(sequence_names, len(ref_seq), args.output)
67.
68.   for row_idx, (posn, snp_in_posn) in enumerate(snps.items()):
69.     ref base = ref seq[posn]
70.     output_row = [1, posn+1, '.', ref_base]
71.     alts_set = set([seq_base for _,seq_base in snp_in_posn])
72.     alts = {a: i+1 for i,a in enumerate(alts_set)}
73.     output row.append(",".join(alts.keys()))
74.     output_row += ['.', '.', '.', 'GT']
75.     alts[ref base] = 0
76.     snp_index_in_posn = [(idx, str(alts.get(base, '0'))) for idx,base in snp_in_posn]
77.     indices_at_posn = []
78.     for (seq_index, snp_base) in snp_index_in_posn:
79.       indices at posn += ['0'] * (seq index - len(indices at posn)) + [snp base]
80.     indices_at_posn += ['0'] * (len(sequence_names) - len(indices_at_posn))
81.     output row += indices at posn
82.     write row(output row, args.output)
83.
84. class TestParseFasta(unittest.TestCase):
85.   def test_parse(self):
86.     input fasta = StringIO("""\
87. >foo
88. AAAAAAAAAAAAAAAAAA
89. >bar
90. GGGGGGGGGGGGGGGGGG
91. """)
92.     sequences = snp_sites_extensions.parse_fasta(input_fasta)
93.     self.assertEqual(sequences.next(), ('foo', 'AAAAAAAAAAAAAAAAAA'))
94.     self.assertEqual(sequences.next(), ('bar', 'GGGGGGGGGGGGGGGGGG'))
```

REFERENCES

1. G. Khoury and P. Gruss, "Enhancer elements", *Cell*, vol. 33, no. 2, pp. 313-314, 1983. Available: 10.1016/0092-8674(83)90410-5.
2. G. Maston, S. Evans and M. Green, "Transcriptional Regulatory Elements in the Human Genome", *Annual Review of Genomics and Human Genetics*, vol. 7, no. 1, pp. 29-59, 2006. Available: 10.1146/annurev.genom.7.080505.115623.
3. M. Esteller, "Non-coding RNAs in human disease", *Nature Reviews Genetics*, vol. 12, no. 12, pp. 861-874, 2011. Available: 10.1038/nrg3074.
4. E. Huntzinger and E. Izaurralde, "Gene silencing by microRNAs: contributions of translational repression and mRNA decay", *Nature Reviews Genetics*, vol. 12, no. 2, pp. 99-110, 2011. Available: 10.1038/nrg2936.
5. Tautz D and Schlotterer, “Simple sequences”, *Nature Reviews Genetics*, vol. 4, pp. 832–837, 1994.
6. G. Miah et al., "A Review of Microsatellite Markers and Their Applications in Rice Breeding Programs to Improve Blast Disease Resistance", *International Journal of Molecular Sciences*, vol. 14, no. 11, pp. 22499-22528, 2013. Available: 10.3390/ijms141122499.
7. T. Thiel, W. Michalek, R. Varshney and A. Graner, "Exploiting EST databases for the development and characterization of gene-derived SSR-markers in barley (Hordeum vulgare L.)", *Theoretical and Applied Genetics*, vol. 106, no. 3, pp. 411-422, 2003. Available: 10.1007/s00122-002-1031-0.
8. F. Amaral et al., "MicroRNAs Differentially Expressed in ACTH-Secreting Pituitary Tumors", *The Journal of Clinical Endocrinology & Metabolism*, vol. 94, no. 1, pp. 320-323, 2009. Available: 10.1210/jc.2008-1451.
9. M. Sivapragasam et al., "MicroRNAs in the Human Pituitary", *Endocrine Pathology*, vol. 22, no. 3, pp. 134-143, 2011. Available: 10.1007/s12022-011-9167-6.
10. G. Richard, A. Kerrest and B. Dujon, "Comparative Genomics and Molecular Dynamics of DNA Repeats in Eukaryotes", *Microbiology and Molecular Biology Reviews*, vol. 72, no. 4, pp. 686-727, 2008. Available: 10.1128/mmbr.00011-08.
11. J. Gulcher, "Microsatellite Markers for Linkage and Association Studies", *Cold Spring Harbor Protocols*, vol. 2012, no. 4, pp. pdb.top068510-pdb.top068510, 2012. Available: 10.1101/pdb.top068510.

12. B. Brinkmann, M. Klintschar, F. Neuhuber, J. Hühne and B. Rolf, "Mutation Rate in Human Microsatellites: Influence of the Structure and Length of the Tandem Repeat", *The American Journal of Human Genetics*, vol. 62, no. 6, pp. 1408-1415, 1998. Available: 10.1086/301869.
13. "5364759 DNA typing with short tandem repeat polymorphisms and identification of polymorphic short tandem repeats", *Biotechnology Advances*, vol. 14, no. 1, p. 81, 1996. Available: 10.1016/0734-9750(96)83416-1.
14. S. Kit, "Equilibrium sedimentation in density gradients of DNA preparations from animal tissues", *Journal of Molecular Biology*, vol. 3, no. 6, pp. 711-IN2, 1961. Available: 10.1016/s0022-2836(61)80075-2.
15. S. Barbaux and F. Cambien, "The single nucleotide polymorphism story", *Pharmacogenetics*, vol. 13, no. 8, pp. 443-444, 2003. Available: 10.1097/00008571-200308000-00001.
16. R. Gibbs et al., "A global reference for human genetic variation", *Nature*, vol. 526, no. 7571, pp. 68-74, 2015. Available: 10.1038/nature15393 [Accessed 1 May 2019].
17. Clancy, S. & Brown, W., "Translation: DNA to mRNA to Protein", Nature Education, 2008.
18. P. Ferrara, "Identification of protein consensus sequence", *Biochimie*, vol. 72, no. 12, pp. 898-899, 1990. Available: 10.1016/0300-9084(90)90014-8.
19. M. Gray, H. Colot, L. Guarente and M. Rosbash, "Open reading frame cloning: identification, cloning, and expression of open reading frame DNA.", *Proceedings of the National Academy of Sciences*, vol. 79, no. 21, pp. 6598-6602, 1982. Available: 10.1073/pnas.79.21.6598.
20. J. Foster, J. Foster and K. Gillen, *Microbiology: An Evolving Science*. New York: W.W. Norton & Company, 2017.
21. P. Compeau, P. Pevzner and G. Tesler, "How to apply de Bruijn graphs to genome assembly", *Nature Biotechnology*, vol. 29, no. 11, pp. 987-991, 2011. Available: 10.1038/nbt.2023.
22. P. Sharp and W. Li, "The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications", *Nucleic Acids Research*, vol. 15, no. 3, pp. 1281-1295, 1987. Available: 10.1093/nar/15.3.1281.
23. G. Everett and R. McLeod, *Software testing*. Hoboken, N.J: Wiley-Interscience, 2007.
24. Dvorski, Dalibor D. "Installing, configuring, and developing with Xampp." Skills Canada (2007).

25. Vaughan-Nichols, Steven J. "Will HTML 5 restandardize the web?" Computer, 2010.
26. K. Howell, "JavaScript Sourcebook: Create Interactive JavaScript Programs for the World Wide Web", *Internet Research*, vol. 8, no. 1, 1998. Available: 10.1108/intr.1998.17208aaf.007.
27. S. Baker, "Making It Work for Everyone: HTML5 and CSS Level 3 for Responsive, Accessible Design on Your Library's Web Site", *Journal of Library & Information Services in Distance Learning*, vol. 8, no. 3-4, pp. 118-136, 2014. Available: 10.1080/1533290x.2014.945825.
28. R. STINE, "An Introduction to Bootstrap Methods", *Sociological Methods & Research*, vol. 18, no. 2-3, pp. 243-291, 1989. Available: 10.1177/0049124189018002003.