Universitat Politècnica de Catalunya

ETSEIB

MASTER'S THESIS

DESIGN AND DEVELOPMENT OF A WEB ACCESIBLE DATABASE FOR SATELLITE DNA

José Castillo Rabazo

Computer Science Department

Tutor: Xavier Messeguer Peypoch

January 2020

General Index

Figure Index

Acknowledgements

I would like to thank Xavier Messeguer and Joan Subirana for the opportunity to work with them and for making this project possible.
To Alex Perera, for his attention and dedication.
To Gabriel Verdejo and all the people at the RDlab, for their patience.
To the, so much needed, voice of my parents on the phone. Thank you.

Abstract

The evolution of computing technology has reached levels that were unimaginable not so long ago. Nowadays, everyone has technology with a substantial amount of computing power literally at the reach of their hands. This technological evolution has led to serious advantages in every branch of scientific and industrial research, and it even has allowed the creation of new disciplines, as it is the case of bioinformatics.

The idea of this project is to take advantage of these new technologies and the high computing power available in today's computers in order to develop a web accessible database for satellite DNA. In genomics, satellite DNA is defined as tandemly repeated sequences of DNA. Since its discovery in the late 60's, the way researchers have tried to find them has changed, ranging from pure biotechnological methods to more advanced ones based on computers. For this application in particular, satDNA data has been computed with the help of SATFIND, a satellite computing program developed by the ALGGEN group at the UPC. All the data has later been treated and stored in a large database located at the Research and Development lab (RDlab) servers. In addition, a web application has been developed in order to make the data available to the public and to the different research groups that are involved nowadays in tandemly repeated genome sequences.

For this master's thesis, a simple, easy to use repository with satDNA data computed from tens of thousands of different species, including both eukaryotic and prokaryotic genomes, has been developed. This document collects all the relevant information about the development process as well as more information about the current state of the art of satellite DNA research and other similar technologies available today.

Key words
Genomics, satellite DNA, tandem repeats, database, postgreSQL, Django, bioinformatics, SATFIND, Genbank, genome computing.

Glossary of terms

AWK The AWK language is a data-driven scripting language consisting of a set of actions to be taken against streams of textual data. 8, 13, 14

Bash Bash is a Unix shell and command language written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell. 8, 13, 14

Django Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. 8, 14, 15

FASTA The FASTA format is a text-based format for representing either nucleotide sequences or amino acid (protein) sequences, in which nucleotides or amino acids are represented using single-letter codes. Every file has two parts, the name, and the sequence, separated by a ">" symbol (See Figure A.1). iii, 10, 12, 19

full-stack In software engineering, a full-stack development is referred to as a process in which the engineer handles all the specific parts of the project such as databases, servers, clients and software engineering. 1

GenBank GenBank is a state of the art database provided by the NCBI that contains public data about genomes, chromosomes and may more genomics related data of a wide range of species. iii, 5, 7, 13, 19

indels In genomics, indel refers to the mutations of either insertion or deletion that can appear in a genome sequence. 5

PlantSat Database that integrates sequence data available for satDNA in plants. iii, 4, 6

Python Python is a very powerful object oriented, general purpose, interpreted programming language that is very popular for its immediacy and for all the available tools that can be found for it on the internet. 6, 8

SATFIND Satfind is a program developed by the ALGGEN group that computes satDNA from a given genome sequence. i–iii, vii, 1, 4, 7, 8, 10–14, 19, 20

Glossary of acronyms

ALGGEN ALGGEN is the acronym for the Algorithmics and Genetics Group, part of the ALBCOM Research Group in the Computer Science Department of the Universitat Politècnica de Catalunya (BarcelonaTech).. vii, ix, 1–4, 7, 17

bp Base pair. 3, 5, 13

CSRF Cross-site Request Forgery. 7, 8

CSS Cascading Style Sheets. 6

DBMS Database Management System. 6

GUI Graphic user interface. 10

HTML Hypertext Markup Language. 6

Kb 1 Kb = 1000 bp. 3

NCBI National Center for Biotechnology Information. ix

NGS Next-Generation sequencing. 4

PHP PHP: Hypertext Preprocessor. 6

RDlab Research and Development lab. vii, 17

SQL Structured Query Language. 6

TAREAN Tandem Repeat Analyzer. 4

TRDB Tandem Repeats Database. iii, 4, 5, 7, 17

TRF Tandem Repeat Finder. 4

UPC Universitat Politècnica de Catalunya.. vii, 3, 7, 17

Chapter 1

Introduction

In genomics, satellites are defined as tandemly arranged repeat sequences present in many eukaryotic genomes. The topic of satellite DNA (satDNA) has been evolving since the discovery of highly repetitive DNA in the 1960s [1], with a lot of scientific publications discussing their structure, organization, function and evolution of such sequences.

The study of satDNA has been found to be very promisingg, as it can help explain the adaptation of species to environmental factors [2], the formation of the so called *de novo* genes [3] and the C-value enigma [4]. In this master's thesis, the computation of satDNA from whole genome sequences has been done, as well as the full-stack development of a web-accessible computer database for storing the data.

1.1 Objective and motivations

The main objective and motivations of this master's thesis are academic, professional, technical and personal.

From the academic point of view, it has been intended to design **DataSat**, a state of the art web-accessible database that will store a big amount of satDNA sequences and metadata in order to have a proper amount of data as well as including the work in the context of bioinformatics.

The technical part of the thesis, also conforming its larger volume, has been the full-stack development of the database, the batch computation of satDNA sequences with the help of the SATFIND program [5] and all the other technical components of the process such as file and data treatment that had to be done and that will be explained in detail later in the document.

From the professional point of view, this master's thesis is part of the work realized by the student as a research assistant in the ALGGEN group and its result will be later used both by the student and by the other members of the group in further stages of the project.

Lastly, form a personal perspective, the main objective of this thesis is to learn information about the use and implementation of large databases, the way that a development and deployment process of a web application works, and gather information about satDNA and other aspects on the field of bioinformatics.

1.2 Structure of the document

The present document is organized in five chapters.

The first chapter is the introduction, in which the project and its main objectives are presented, as well as the motivations that led to it.

The second chapter explains the current state of the art of satDNA research, computation and data storage, a brief explanation of the technology involved in the development of web applications, and a glimpse of some of the database engines available nowadays.

The third chapter explains the analytical process behind this master's thesis, as well as a summary of the requirements and the thinking process that led to the current design of the application.

The fourth chapter is dedicated to the rundown of all the implementation process. Everything regarding the actual web application and the file and data treatment plus satellite computation is explained in this chapter.

The fifth and last chapter includes the conclusions made during the development process of this master's thesis and the future work that will be done with the application on the ALGGEN group.

To conclude, there are a series of Annexes that provide useful additional information that the reader might need throughout the reading of the document as well as all the used bibliography.

Chapter 2

State of the art

In this chapter, the current state of the art of the main components of the thesis is going to be presented. First of all, a more thorough explanation about satDNA is going to be done as well as a brief summary of the current state of the art of satDNA research and computation. Then, I will dive with more detail into the technology involving large databases, finishing with a glimpse of the current state of the art of web development.

2.1 Satellite DNA

The term "satellite DNA" was first proposed by Pech et al. [6] in 1979, but it was not until Singer's publication [7] on 1982 that the term was made popular. The use of the term is rather difficult to nail down, as it refers to a wide variety and highly variable part of the eukaryotic genome, besides, nowadays, satDNA is not only reduced to eukaryotic genomes, in fact, the study of the satDNA in the prokaryote world is one of the main focus of the ALGGEN group here at the UPC [8].

There are some key components that have to be taken into account when it comes to defining satDNA. For each satellite, the location, repeat length and copy number are some of the features that can be analyzed in order to better classify the satellite families, in fact, the high variability of these features imply some differentiation in the definition of satDNA in literature. In this context, apart from normal satellites, microsatellites and minisatellites can also be found. Microsatellites are simply tandem repeats of less than 10 bp in length in arrays less than 1 Kb distributed in loci throughout the genome. Minisatellites are similar but have longer repeats (> 10 bp) [1].

Highlighted in red, in the Figure 2.1, an example of a satellite is presented. It will serve as a model to explain in more detail all the parameters that can be used in order to classify and short satDNA when conforming a large database.

```
----------TTCGGATTTACGTTCTATTTTTTATTAGAATCACGTGAGTTCGACGAGT
CAGAACTCATGATTCTGAAAAAAGTTGAAATCTGAACTTTATAGATCGATTCTTTAAAT
GTGGGAACTACCACAAATCACAATTTTACAGTTTAATCTTGAATAC
GTGGGAACTACTACAAATCACAATTTTACAGTTCAATCTTGAATAC
GTGGGAACTACCACAAATCACAATTTTACAGTTCAATCTTGAATAC
GTGGGAACTACCACAAATCACAATTTTACAGTTCAATCTTGAATAT
GTGGGAACTACCACAAATCACAATTTTACAGTTCAATCTTGAATAC
GTGGGAACTACCACAAATCACAATTTTACAGTTCAATCTTGAATAC
GTGGGAACTACCACAAATCACAATTTTACAGTTCAATCTTGAATAT
GTGGGAACTACTACTTTACTAAACGCCGAACCTATCTTACTTAAAAAACGAATCCATTC
TAAAAAAAGGTGATTTCTATTTATAAATTTATTTTCACCCAATCATTTACTTTTAAAAA
TCGGACCGTCTTTTAAGTTCAAAAT---------
```

Figure 2.1: Example of satDNA.

In this case, the sequence underlined,"GTGGG" is called the seed, which is the part of the satellite that marks the beginning of the repetition sequence. The sequence above is an example

of a satellite that has eight repeats with a total length of 333 bases. Notice that in the last base of the repeat, there is an error that causes some bases to be C and others to be T, this is called a dislocation mutagenesis [9]. These type of errors occur with frequency in DNA sequencing, moreover, regions of DNA containing many copies of small repeated sequences (such as satDNA) are particularly prone to this type of error [10] and, of course, they are taken into account when computing the satDNA sequences.

2.1.1 satDNA computation

The first methods for determining satDNA sequences were introduced in 1961 by Kit et al. [11] and by Seouka et al. [12] and they consisted on a method based on the density-gradient ultracentrifugation. This method, along with the Cot analysis developed by Britten and Waring [13], were the two predominant methods for the characterization of repeated DNA sequences during the 60s and the 70s [1]. Altough these and other methods like the one proposed by Singer in 1982's publication, *Highly repeated sequences in mammalian genomes* [7], were crucial for the evolution of satDNA research, they were pure experimental processes that had their disadvantages. Firstly, they were very difficult to perform and required a lot of time to do so, besides, there were many monomers that escaped the so mentioned procedures when satDNA was poorly expressed in the genome or when dealing with species with large genome size [1]. The explanation of biotechnological processes for satDNA identification is key in the evolution of repeat genome families research but is not under the scope of this thesis, as in this context, it is more interesting to focus on the computer based computation of satDNA. There are two main types of computation processes, one that use reference genomes for the computation and one that does not need assembled genome data to do so.

Graph-based computation

The sequencing of genomes using Next-Generation Sequencing (NGS) was a milestone in satDNA sequencing and in genomics in general. It led to numerous advances in the study of repetitive DNA with the development of several computational tools like RepeatExplorer [14][15]. RepeatExplorer uses graph-based genomics approach for similarity-based partitioning of whole genome sequences with the help of NGS [14]. Another tool developed with a different perspective by the same group is the Tandem Repeat Analyzer (TAREAN) [16], which appeared from a different pipeline by using also graph-based analysis but with the implementation of an unsupervised algorithm that uses k-mers obtained by decomposing read sequences from corresponding clusters [16]. The above mentioned programs are particularly useful when there is no previously assembled genome as a reference. This is also the case of satMiner, which is a bioinformatics tool developed by Ruiz-Ruano et al. that is particularly useful for finding satellites that are extremely rare in the genome [17] by the iteration of the RepeatExplorer program and the elimination of the satellites that are already known in order to keep only the satellite families that are not known yet. All the above tools are graph-based genomics approaches. Graph-based algorithms do not need an assembled genome sequence in order to operate.

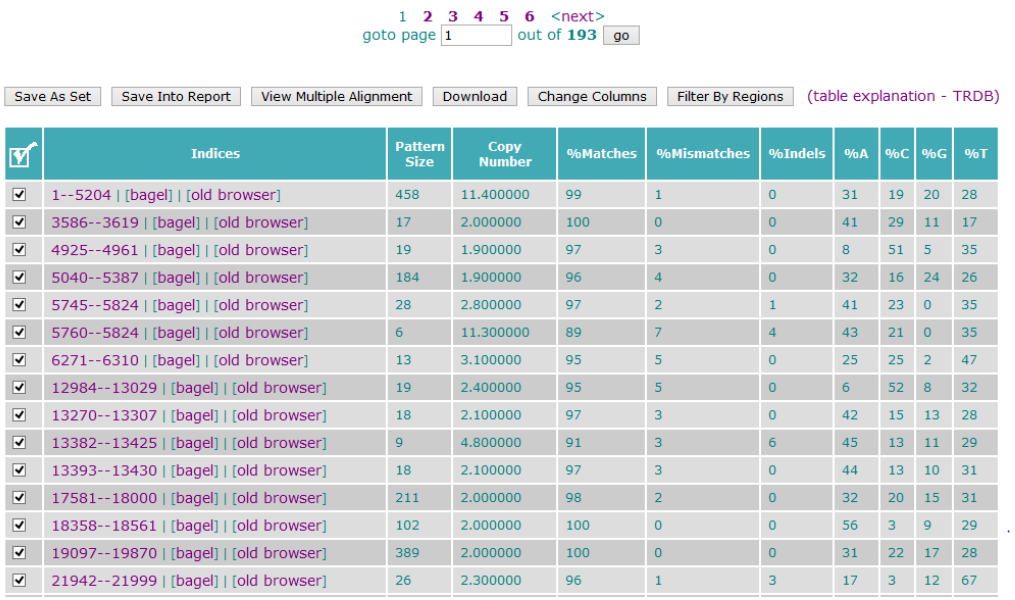
Assembled genome computation

The other type of computing tools that can be found are the ones that need an assembled genome sequence as a reference in order to compute the satDNA. Under this particular category falls SATFIND [8] and Tandem Repeat Finder (TRF) [18]. As stated in the previous chapter, SATFIND is the program developed by the ALGGEN group and it is the one that has been used in this master's thesis.

2.2 Satellite databases

There are two relevant satDNA databases currently available online. These are the Tandem Repeats Database (TRDB) and PlantSat. The TRDB was developed by Gary Benson at the Boston University and it contains a variety of tools for repeat analysis, including a tandem repeat finder

with query and filtering capabilities [18]. In the Figure 2.2 a screenshot of the TRDB is presented. The data of the satDNA is ordered by the following columns. The *Indices* column gives a range of numbers that represent where in the genome sequence is the particular satellite located. *Pattern Size* indicates the full length of the repeat sequence. The *Copy Number* is the actual number of repetitions in tandem for the particular satellite. The next columns, *Matches, Mismatches and indels* indicate the errors that the satellite has, this gives an idea of the computation parameters that were set prior to computation. The last four columns indicate the percentage of each nucleotide present in the satDNA sequence.



Indices	Pattern Size	Copy Number	%Matches	%Mismatches	%Indels	%A	%C	%G	%T
1--5204 | [bagel] | [old browser]	458	11.400000	99	1	0	31	19	20	28
3586--3619 | [bagel] | [old browser]	17	2.000000	100	0	0	41	29	11	17
4925--4961 | [bagel] | [old browser]	19	1.900000	97	3	0	8	51	5	35
5040--5387 | [bagel] | [old browser]	184	1.900000	96	4	0	32	16	24	26
5745--5824 | [bagel] | [old browser]	28	2.800000	97	2	1	41	23	0	35
5760--5824 | [bagel] | [old browser]	6	11.300000	89	7	4	43	21	0	35
6271--6310 | [bagel] | [old browser]	13	3.100000	95	5	0	25	25	2	47
12984--13029 | [bagel] | [old browser]	19	2.400000	95	5	0	6	52	8	32
13270--13307 | [bagel] | [old browser]	18	2.100000	97	3	0	42	15	13	28
13382--13425 | [bagel] | [old browser]	9	4.800000	91	3	6	45	13	11	29
13393--13430 | [bagel] | [old browser]	18	2.100000	97	3	0	44	13	10	31
17581--18000 | [bagel] | [old browser]	211	2.000000	98	2	0	32	20	15	31
18358--18561 | [bagel] | [old browser]	102	2.000000	100	0	0	56	3	9	29
19097--19870 | [bagel] | [old browser]	389	2.000000	100	0	0	31	22	17	28
21942--21999 | [bagel] | [old browser]	26	2.300000	96	1	3	17	3	12	67

Figure 2.2: Screenshot of the TRDB

The TRDB is a very advanced tool that has a lot to offer. Not only all the tables are interactive and full filterable, but the data can also be downloaded to later be analysed at will by the researchers. It also includes a system in which anyone can log in and save their particular set of satDNA and share it with the community.

The PlantSat database, developed by Jirí Macas at the Institue of Plant Molecular Biology of the Czech Republic, offers a database that integrates sequence data available from various resources with supplementary information including repeat consensus sequences, abundances and chromosomal localizations for sequences of plant genomes [19]. A screenshot of the PlantSat database is provided in Figure 2.3. The structure of this database differs from the one proposed by the team working at the TRDB, in this particular web page, the information about satellites is not presented in a table format, in contrast, an index with the species is provided with the information of the satellite presented once clicked. Information regarding the repeat length (in bp), location of the repeated sequence and the actual sequence itself is provided in a very clear format. PlantSat does not allow users to download the data but provides links to GenBank data.

2.3 Databases

We live in an era in which 2.5 quintillion bytes of data are created every day, and the number is growing every day. The storage and management of that amount of data is not trivial at all, in fact, major tech companies and universities have to be very creative in order to handle in an efficient manner all that flow-stream of data. The most common way to storage data is with the use of databases. Databases are organized collections of data that are accessed electronically and that are stored either in computers or in servers. They allow the storage and access of a large quantity of data which information and metadata can be related with other databases or other sources of information. The use of databases from a computer science perspective began in the

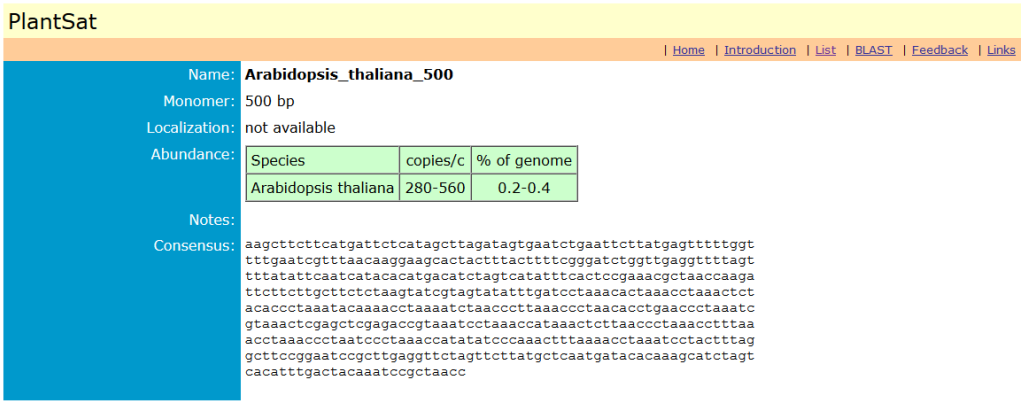


Figure 2.3: Screenshot of the PlantSat database

60s, with the first implementation of the database management system (DBMS). The most used language today for database utilities is the structured query language or SQL, developed by Donal D. Chamberlin in 1974 at IBM. Most database engines nowadays are based on SQL, and it has become pretty much the standard when it comes to database creation, management and query. The list of the most used database engines nowadays includes MySQL, mongoDB, MariaDB, DB2 and PostgreSQL, which is the one that has been used for this project.

2.4 Web development

Web development is complex process in which there are a lot of different elements and disciplines that need to be applied in order to make a functional and efficient web application. To better illustrate the development process of a web application I will introduce the different parts of a web application and give some examples of what technologies and approaches can be done for each element of the development process.

2.4.1 Front-end

In web-development, the front-end section refers to the part of web development that converts data into graphical information, i.e. the part that the user sees. It conforms the portion of the application relative to the user interface and the exchange of information between the user and the application itself. For this purpose, the typical programming languages used are HTML, CSS and JavaScript.

2.4.2 Back-end

In every web application there is a part of the process that is located on the server and makes the application run, this will be all the portion of the code that is hidden to the user. Back-end development focuses on building all the logic and functionality of the application, besides, the architecture of the system itself heavily relies on the back-end in order to function efficiently. Relevant programming languages used in back-end development include PHP, Python, Ruby and Node.js.

Chapter 3

Analysis and Design

This chapter is dedicated to the analysis, definition and design of the characteristics and requirements of the proposed application.

3.1 Definition of the project

The first stage of the project is to classify it and specify the main objective and characteristics in order to better understand the required tools that will be needed in the development of it.

The main objective of the project is stated on the title and it is the **design and development of a web-accessible database for satDNA**. To conquer this objective, a web application is needed as well as a large database that can handle huge quantity of data, all of that deployed on a server located at the UPC. It is intended for the satellite data to be computed with the SATFIND program.

Data should be provided in the form of an interactive and intuitive table which needs to be filterable and orderable by any of its columns. Columns should provide all the information relevant to the satellite data, including the name of the species, its unique GenBank identifier, the length of the satellite, the number of repetitions, the length of the sequence, the seed and the actual sequence itself. The application should also allow the user to download the available data in both *.csv* and *.xls* formats the same way that the TRDB does.

The purpose of this database is to store satDNA data for the majority of species possible. In order to filter and reduce the number of input genomes that are to be computed and introduced in the database, only complete genome assemblies available in GenBank are going to be selected. For the first stages of the project, in which this master's thesis is included, only genomes regarding arqueas and bacteria genomes need to be included in order to continue with the research of satellites in the prokaryote world realized by the ALGGEN group [8] [5].

One of the main drawbacks of the current workflow for using the SATFIND program is the inability of the tool to compute more than one genome sequence at a time, and it is for this very reason that a method to batch compute multiple series of sequences without the need to do it manually is a must for this project.

3.2 Requirements

When developing web applications, there is a strong need to focus on security aspects, as internet connected databases are specially vulnerable to cyberattacks. In this context, it is necessary to protect the application and the data integrity against two of the most well-known server exploits. These are the *SQL injection* and *Cross-site request forgery* (CSRF). SQL injection is a vulnerability of internet accessible databases that allow attackers to perform unwanted querys on the database that could potentially alter and compromise data stored in the database. CSRF is a malicious exploit that allows a trusted user to send non-authorized messages to the server's application, potentially causing harm both to the server and the application itself.

One of the key aspects of this application is its simplicity. When deployed, there is no need or intention to require user login authentication, file and image upload or anything of that kind. The application should only show an interactive, static table with data regarding tandemly repeated DNA, this will reduce the complexity and potential harms of the application as well as allow it to be more robust and efficient.

3.3 Design

Once the requirements are stated, the design process begins. In this phase, the most suitable tools regarding the application's requirements are to be selected.

- For the development of the actual application, it has been decided to use **Django** [20]. Django is a high-level **Python** [21] developing framework for web applications. The use of Django offers a lot of advantages. It is easy to learn and use, taking advantage of the flexibility a programming language like Python offers, besides, all the cybersecurity aspects mentioned before, like the protection of the site against SQL injection and CSRF attacks is well supported and automated on this framework. It has also been used instead of other tools like PHP and JSON because the integration of the framework with databases is very optimized and straight-forward. There are several tools and libraries that have been used along with Django, these will be mentioned later in the implementation chapter.
- For small projects, the selection of a database engine is almost trivial, as most of them work pretty well and very similar to each other, however, taking into account the potential size of the database when working with huge amount of data, it is necessary to pick a database engine that is most suitable for the desired purpose. In particular, the database engine that has been used is **PostgreSQL** for its ability to work with large scalable quantities of data [22].
- The design of the Front-end has to be very simple as stated in the requirements section. The front-end should only provide a table with the satellite data. For the design of the actual database, the columns have been selected to match the output data shown by the SATFIND program. The name of the columns are presented in the Table 3.1. These names represent, in order, the id of the element in the database, the unique identifier of the GenBank, the name of the genome, the position of the satellite inside the genome, the satellite length, the number of repetitions, the length of the repetition, the seed of the satellite and the full sequence of the satDNA sequence.

Table 3.1: Columns of the database

id	Genbank id	Name	Indice	Satlength	Numrep	Replength	Seed	Seq

- It has also been stated that there is a strong need for the batch computation of satDNA with the help of the SATFIND program. For this requirement, **Bash** has been used. This programming language, born as a shell for UNIX systems, is a very powerful tool that allows the automation of process and commands in a few lines of code.
- Taking into account that the files needed both to compute the satellites and to populate the database have to be treated in some way, the use of **AWK** is the perfect solution, as it is a very fast and powerful programming language suitable for file and data treatment whose scripts can also be automated with the help of Bash. A more detailed description of what has been done with both Bash and AWK will be presented in the next chapter.

Chapter 4

Implementation

The purpose of this chapter is to detail everything that has to do with the implementation of the project. First of all, a detailed description of the development environment is going to be made. Following, the main programming platforms will be introduced with an explanation of their function inside the project. Lastly, the process of data and file treatment that had to be done in order to compute the satDNA data and populate the final database is going to be explained as well.

4.1 Development equipment

This section of the software includes the hardware and software characteristics of the employed development equipment

4.1.1 Hardware

The project has been realized on its entirety on a **msi ge60-2oe** laptop with the following characteristics:

- **Intel i5-4200M** CPU with 4 cores at **2.5GHz**.
- **8GB RAM** DDR3 memory.
- **256 GB** SSD.
- **Nvidia GTX765M** graphics card.

4.1.2 Software

Operative system

- **Windows 8.1 Pro 64bits** as a main operative system.
- **VMware Workstation 15 Player** virtual machine running **Ubuntu 18.04 LTS 64bits** with 4 dedicated GB of RAM.

Programming software

- **Atom** to make all the coding [23].
- **Git** for version control.
- **pgAdmin III** and **postgreSQL** for database management and administration.

Documentation software

- **LaTeX** for the creation of this document.

4.2 Platforms

In this section the programming tools and languages used for the development of this project are going to be explained, along with the libraries that have been used for specific requirements of the web application.

4.2.1 Django

As mentioned in the previous section, Django is a web development framework based on Python. Django projects have a very clear, modular structure that allows the developer to use and recycle parts of other projects. Django is also very suitable for working with libraries and third party applications that work with Python. To be more specific, the libraries that have been used for the development of **DataSat** are the following:

- **Django Tables 2** [24]. It is the library used for the data visualization of the application. It basically does make the creation of interactive tables very quick and simple.
- **Django Filter** [25] is a very powerful Django library that will make the table created filterable with custom filters. Thanks to this library, the user can query the database extensively as there is a form filter for each of the columns.
- **Psycopg2** [26]. This library is called a database adapter and it is necessary for the implementation of postgreSQL type databases with Django.

4.2.2 pgAdmin III

It is the framework tool used to manage and administrate the PostgreSQL database. This type of program allows the administrator of the database to create new tables and schemes, add or delete users, populate databases and everything related with the design and development of a database in a very simple manner and with the help of a GUI.

4.2.3 Ubuntu 18.04 LTS

In theory, adding an operating system as a programming platform itself is quite strange, as it is so much more, however, the usage of a UNIX type operating system like this one, offers a lot of advantages and opportunities that have been heavily used in the development of DataSat. The use of Ubuntu's Bash terminal is something that cannot be ignored, as all the processes regarding file and data treatment run on it, and it also offers full integration with the standalone version of the SATFIND program, that is written on C and runs flawlessly on the Ubuntu's terminal.

4.3 Satellite computation

As stated in previous sections, the satellite computation has been done with SATFIND. SATFIND is a quite complex program on the inside but it is very simple to use. It basically has a series of input and output parameters that customize the way that the satellites of the input genome (in FASTA format) are computed.

4.3.1 Input parameters

Number of the first contig

Contigs are overlapping regions of DNA that together form a canonical region of DNA. By default, the contig selected is the first one.

First Selected contig

This parameter indicates if we want to select a different number of contigs. If the number is 0, it means all the contigs are going to be selected. If the number is different the selected contig will be the one indicated by the number.

Las selected contig

This parameter works the same way the previous one does. With the combination of the first and the last selected contig the user may chose the range of contigs that they want to compute the satellites of. If this number is 0, this parameter is ignored.

Minimum repetitions of the cluster

This is one of the most important input parameters as it indicates the minimum number of repetitions that a sequence must have in order for it to be considered an accepted satellite.

Cluster length

The cluster length indicates the length of the selected cluster. The higher this value is the more potential satellites that the program will find.

Length of the pattern

If this value is 0, SATFIND will search for a pattern of a specific length indicated on the next parameter. Any other value will make the program to search for patterns of every length.

Pattern size

This value indicates the specific size of the target pattern in case the previous parameter has a value of 0.

Target pattern

If on the two previous parameters the user has chosen to look for a pattern of a specific length, with this parameter the user can select a specific pattern to look for.

Range

This value indicates the error margin of the length of the satellites found. This is one of the parameters that is included taking into account the proper nature of the satellites that can include errors as explained in the state of the art section.

Percentage

There are some situations in which the program finds some satellites that are not particularly interesting for the amount of errors they have. With the help of this value the user can define the tolerance of the satDNA found by SATFIND in order to reduce the dimension of the number of output sequences.

Minimum number of repeats

This value works with the same principle as the previous one in the sense that it can reduce the amount of satellites accepted in order to eliminate those satellites that for any particular reason are not useful. In particular, this parameter refers to the minimum number of repeats that a sequence must have in order to be considered accepted. Sequences found that do not match the criteria selected in this parameters will be rejected.

Minimum length of the repeats

By adjusting this number the user indicates the minimum length of the repetition sequence that will be considered candidate for a satellite. If the value is 0 no length constraints will be applied.

Maximum length of the repeats

This value works like the previous one but selecting the maximum length of the repeat sequence. Inputting a 0 in this parameter will have the same effect as the previous parameter.

4.3.2 Output parameters

The output parameters have nothing to do with the actual computation of the satDNA sequences but rather with the way SATFIND shows the data and what information it shows.

Accepted clusters

This parameter has five options:

- 0: Don't print anything
- 1: Print the list of the accepted clusters
- 2: Print the list of the accepted clusters with the corespondent representation
- 3: Print the list with every repeat of each cluster
- 4: Print the list of only the selected repeats with the selected range.

Rejected clusters

This parameters offers 3 options:

- 0: Don't print anything
- 1: Print information about rejected clusters
- 2: Print all the repeats of the rejected clusters.

Number of errors

This parameter refers to the number of errors in the pattern to find when finding a specific cluster.

Align clusters

There are five options for this parameter:

- 0: Don't align the cluster sequences
- 1: Print each of the canonical sequence of every repeat of the cluster in FASTA format.
- 2: Print the canonical sequences of the repeats of the selected clusters in FASTA format.
- 3: Print the alignment of every repeat of each cluster
- 4: Print the alignment of the selected repeats of each cluster

Prefix

Indicates the prefix of the name of the satellites found.

Name of the file

Selects the name of the output file that is going to be created with the chosen information.

Optional parameters

This parameters are optional and are chosen just if the alignment option has been selected

- Number of columns for the alignment
- Name of the file of the weight matrix
- Value of the gap opening (-3 by default)
- Value of the gap extension (-1 by default)
- Value of the gap extrem (-0.5 by default)

To better illustrate the computation process of the satellites with the help of the SATFIND program, on the annex A, there are a couple of screenshots showing the aspects of the input and output files involved in the process.

4.4 File and data treatment

When populating a database this large, with tens of thousands of files, the process needs to be automated, as there is no healthy way for any human being to do this process manually. For this project, starting with the download of the genome files from GenBank, until the final population of the database, there are a couple of processes that need to be automated.

4.4.1 File renaming

When batch downloading a series of genomes assemblies from GenBank, the data is given in a *.rar* file with all the genome assemblies for the selected query. The main problem is that the names of these files do not correspond with the names of the species, in fact, the name of the given files is a series of random number and letters that have to do with the query itself, and nothing to do with the actual genome name. This is a major problem, not only because it is impossible to know what genome we're referring to until we open the file and see the actual name and unique GenBank identifier of the genome in the header, but also because the SATFIND gathers the name of the species by the name of the file for its computation, so there will be no way of telling which species have been computed other than going to the list of the generic name given by the GenBank query and changing it manually. For this purpose, an AWK script has been developed. The script basically gathers the name of the header file (that is located inside the file and not in the name as previously stated) and renames the file with the name of the specie and its unique identifier. This process is then automated for all the files in a given directory with the help of a Bash script.

4.4.2 Batch computing using SATFIND

As stated in the introduction, one of the main drawbacks of the current state of SATFIND is its inability to work with multiple input files. This problem is quickly addressed with the help of Bash, as making a script that computes all the genome files on a given directory is an easy process. In order to compute all the files of the genomes, the parameters used for the task have been the same as the ones used by Subirana and Messeguer in their last publication, *Satellites in the prokaryote world* [8]. These parameters are:

- All contigs selected.
- 4 minimum repetitions for the satellites to be considered as accepted.
- 800 bp for the cluster length.
- 10 bp for the length of the pattern.
- 60 for the percentage of the accepted clusters.
- No minimum or maximum length for the repeats of the clusters

4.4.3 Formatting of the output files

As it can be observed in the Figure A.2, the output files of the SATFIND program has a human-readable way of printing the data that represents a problem if the data wants to be introduced on a database. The easiest way to populate a database is with the help of *.csv* files, so there is a strong need to find a way to extract the desired information of the output files of SATFIND and put it in a *.csv* file. This is made possible with the help of AWK as well. A script has been developed in order to convert the output files in the desired manner and, again, the process has been automated with the help of Bash in order to batch format all the files for a given directory.

On the Figure 4.1 there is a schematic picture of the file and data treatment process since the download of the files, to the final *.csv* files that can be later be used to populate the database tables.

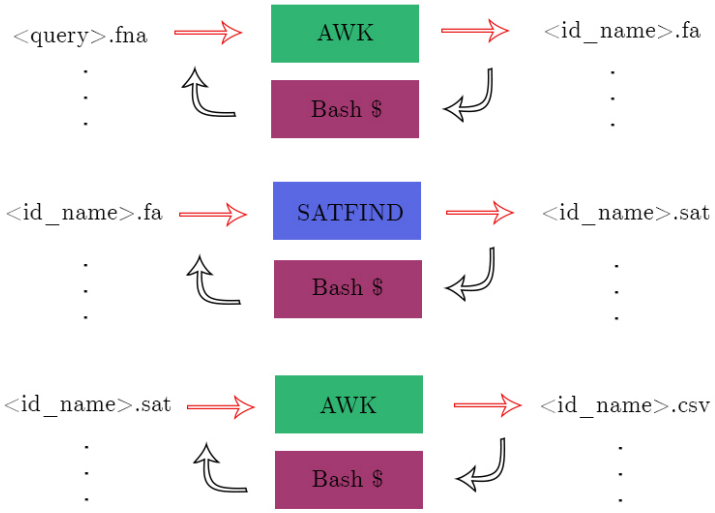


Figure 4.1: File treatment process

4.5 Project Structure

This section will gather all the information regarding the Django project that defines the web application. Assuming all the satellites are computed and the database tables are populated with the satDNA data, the process of creating the application is made very simple with the help of Django. As explained in previous sections, Django is a very powerful framework that allows the user to reuse and recycle parts of different projects very easily. The parts that conform a project are called apps. The DataSat project has three different apps, these are called *database*, *datasatv2* and *django_tables2*. A screenshot of the application (running on a local server on the virtual machine) is shown in the Figure 4.2. There, all the characteristics and requirements previously stated in the design chapter are shown. The simplicity of the interactive tables, the filters that can be applied and all the columns as designed in the Table 3.1.

4.5.1 Database

This app contains all the relevant code that gathers the information from the PostgreSQL tables and builds the models and classes necessary to print the information in the web application. It is made up by the following *.py* files (Python code files):

database/models.py

This file contains the code that builds the model class that will be printed as a table in the application with information of all the columns of the table created in the PostgreSQL database. This columns are the same that were shown in the Table 3.1.

database/tables.py

This file's code is the one that creates the table (from a template) that will later be printed on the web application's screen.

database/filters.py

Here, all the filters regarding the table query can be found.

database/views.py

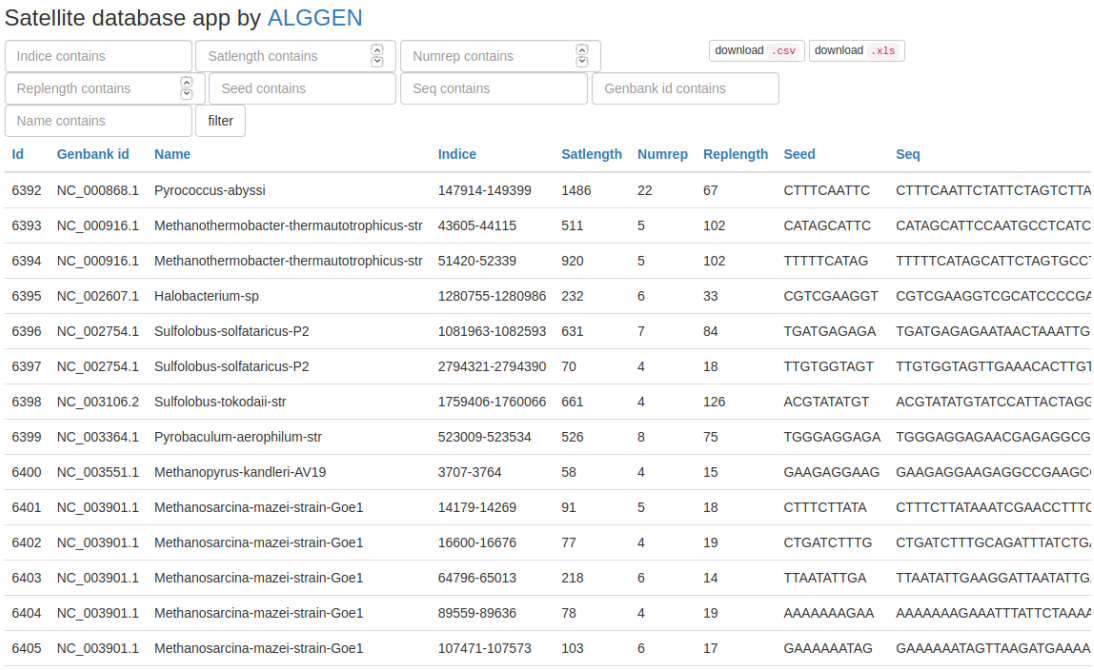
The code inside this file is the one that prints the information on the screen. It gathers the information from the *tables.py* and the *filters.py* to use them dynamically and interactivelly in the actual web application.

4.5.2 django_tables2

This app remains unchanged from the source code found on GitHub and it is the one that creates the templates and manages the filters to later print the information on the screen. In other words, it is the engine that creates the interactive tables that will be used to show the information in a simple a dynamic way.

4.5.3 Databasev2

It is the main app of the project. It communicates all the different apps of the projects and manages all the information regarding dependencies and url information as well as all the settings that can be tuned on Django to configure the application and the security.



Id	Genbank id	Name	Indice	Satlength	Numrep	Replength	Seed	Seq
6392	NC_000868.1	Pyrococcus-abyssi	147914-149399	1486	22	67	CTTTCAATTC	CTTTCAATTCTATTCTAGTCTTA
6393	NC_000916.1	Methanothermobacter-thermautotrophicus-str	43605-44115	511	5	102	CATAGCATTC	CATAGCATTCCAATGCCTCATC
6394	NC_000916.1	Methanothermobacter-thermautotrophicus-str	51420-52339	920	5	102	TTTTTCATAG	TTTTTCATAGCATTCTAGTGCC
6395	NC_002607.1	Halobacterium-sp	1280755-1280986	232	6	33	CGTCGAAGGT	CGTCGAAGGTCGCATCCCCGA
6396	NC_002754.1	Sulfolobus-solfataricus-P2	1081963-1082593	631	7	84	TGATGAGAGA	TGATGAGAGAATAACTAAATTG
6397	NC_002754.1	Sulfolobus-solfataricus-P2	2794321-2794390	70	4	18	TTGTGGTAGT	TTGTGGTAGTTGAAACACTTGT
6398	NC_003106.2	Sulfolobus-tokodaii-str	1759406-1760066	661	4	126	ACGTATATGT	ACGTATATGTATCCATTACTAGC
6399	NC_003364.1	Pyrobaculum-aerophilum-str	523009-523534	526	8	75	TGGGAGGAGA	TGGGAGGAGAACGAGAGGCG
6400	NC_003551.1	Methanopyrus-kandleri-AV19	3707-3764	58	4	15	GAAGAGGAAG	GAAGAGGAAGAGGCCGAAGC
6401	NC_003901.1	Methanosarcina-mazei-strain-Goe1	14179-14269	91	5	18	CTTTCTTATA	CTTTCTTATAAATCGAACCTTTC
6402	NC_003901.1	Methanosarcina-mazei-strain-Goe1	16600-16676	77	4	19	CTGATCTTTG	CTGATCTTTGCAGATTTATCTG
6403	NC_003901.1	Methanosarcina-mazei-strain-Goe1	64796-65013	218	6	14	TTAATATTGA	TTAATATTGAAGGATTAATATTG
6404	NC_003901.1	Methanosarcina-mazei-strain-Goe1	89559-89636	78	4	19	AAAAAAAGAA	AAAAAAAGAAATTTATTCTAAA
6405	NC_003901.1	Methanosarcina-mazei-strain-Goe1	107471-107573	103	6	17	GAAAAAATAG	GAAAAAATAGTTAAGATGAAAA

Figure 4.2: Screenshot of DataSat running on a local server

Chapter 5

Conclusions and future work

5.1 Conclusions

The aim of this project was to develop a web accessible, large database for satellite DNA data. The application should provide the data in a shortable and filterable way, so the information is easy to gather for later use in further research. As it can be seen on the screenshot of Figure 4.2, all the requirements previously stated were fulfilled. In the state of the art chapter, two additional databases were presented [18] [19]. On the one hand, comparing DataSat with TRDB, it is safe to say that the TRDB has a lot of functionality, but lacks the amount of species and simplicity that DataSat offers, in fact, once it is deployed, DataSat will be the largest database available on the internet with eukaryotic and prokaryotic species, including arqueas and bacteria. On the other hand, PlantSat has a different way to show the data and has the great inconvenience that only shows satDNA families of plants.

The solution developed will be very useful for the research that is going to be done in the ALGGEN group during the following months, and for any other research groups or individuals that will need a database with satellite sequences for a large quantity of species.

5.2 Future work

One of the main advantages of the modularity of the application and the scripts developed for the file and data treatment is that the current state of the application can be updated with ease in the future, and it will, as there are a lot of species that still need to be included in the database. Another thing to keep in mind is that the application is not still deployed, and that work is currently being done at the time this lines are being written.

When it comes to scalability, there are some adjustment that have been suggested by the system administrators at the RDlab at the UPC in order to boost the performance of the application. All of this work will be realized in the upcoming months continuing my job as a research assistant in the ALGGEN group.

Annex A

SATFIND computation files

This annex shows a couple of screenshots to better illustrate the computation process of the satellites with the help of the SATFIND program. On the Figure A.1, a screenshot of a portion of an input genome file is shown (in FASTA format). The Figure A.2 shows a screenshot of the aspect of the output files of the SATFIND program with the satellites already computed as well as the sequences generated.

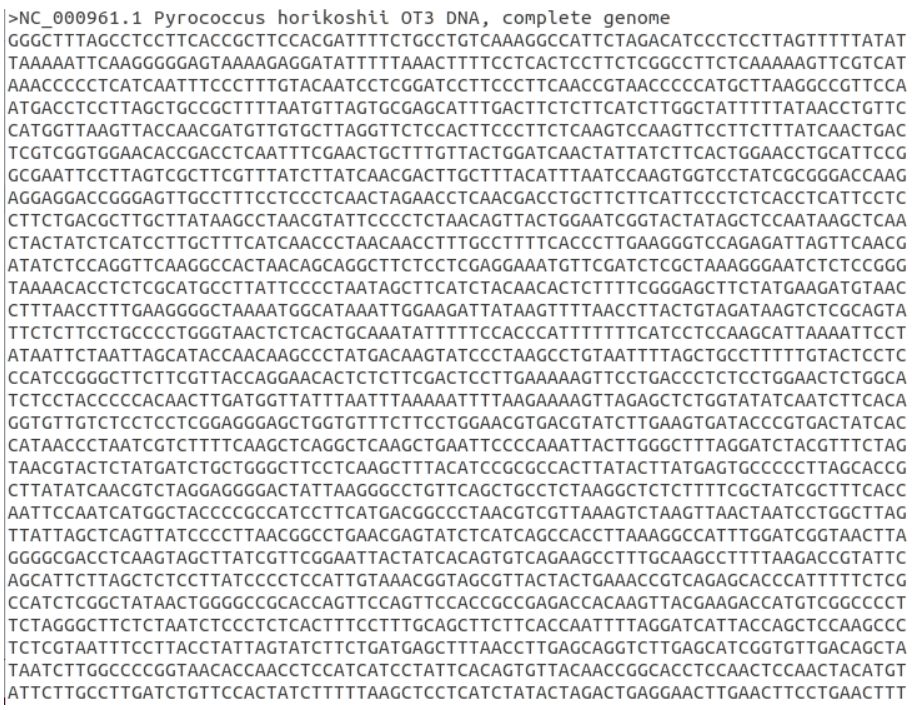


Figure A.1: Genome file downloaded from GenBank in FASTA format for the computation of the satellites using SATFIND

```
 >gen185_1:4704-4817 Satlength=114 Nr of Repeats=6 RepeatLength=17 seed=TAATATTCTG
TAATATTCTGGTTCCTC
TAATATTCTGGTTATTC
TAATATTCTGATTTTTC
TAATATTCTGTTTTCTG
TAATACTCTGATCCCTC
>gen185_1:61916-61972 Satlength=57 Nr of Repeats=4 RepeatLength=14 seed=TAGAATGTGT
TAGAATGTGTATGA
TAGAATGTGTATGA
TAGAATGTGTATGA
TAGAATGTGTATGG
>gen185_1:97462-97592 Satlength=131 Nr of Repeats=10 RepeatLength=13 seed=TCTTTTATAA
TCTTTTATAAGTG
TCTTTTATAAGTG
TCTTTTATAAGTG
TCTTTTATAAGTG
TCTTTTATAAGTG
TCTTTTATAAGTG
TCTTTTATAAGTG
TCTTTTATAAGTG
TCTTTTATAAGTG
TCTTTTATAAGTG
>gen185_1:123533-123677 Satlength=145 Nr of Repeats=12 RepeatLength=12 seed=TATTTGTATT
TATTTGTATTTG
TATTTGTATTTG
TATTTGTATTTG
TATTTGTATTTG
TATTTGTATTTG
```

Figure A.2: Output file of the SATFIND program.

Bibliography

[1] M. A. Garrido-Ramos, "Satellite dna: An evolving topic," *Genes*, vol. 8, no. 9, p. 230, 2017 (cit. on pp. 1, 3, 4).

[2] X. Zhuang, C. Yang, K. R. Murphy, and C.-H. C. Cheng, "Molecular mechanism and history of non-sense to sense evolution of antifreeze glycoprotein gene in northern gadids," *Proceedings of the National Academy of Sciences*, vol. 116, no. 10, pp. 4400–4405, 2019 (cit. on p. 1).

[3] H. T. Baalsrud, O. K. Tørresen, M. H. Solbakken, W. Salzburger, R. Hanel, K. S. Jakobsen, and S. Jentoft, "De novo gene evolution of antifreeze glycoproteins in codfishes revealed by whole genome sequence data," *Molecular biology and evolution*, vol. 35, no. 3, pp. 593–606, 2017 (cit. on p. 1).

[4] T. R. Gregory, "Genome size evolution in animals," in *The evolution of the genome*, Elsevier, 2005, pp. 3–87 (cit. on p. 1).

[5] J. A. Subirana and X. Messeguer, "A satellite explosion in the genome of holocentric nematodes," *PLoS One*, vol. 8, no. 4, e62221, 2013 (cit. on pp. 1, 7).

[6] M. Pech, T. Igo-Kemenes, and H. G. Zachau, "Nucleotide sequence of a highly repetitive component of rat dna," *Nucleic acids research*, vol. 7, no. 2, pp. 417–432, 1979 (cit. on p. 3).

[7] M. F. Singer, "Highly repeated sequences in mammalian genomes," in *International review of cytology*, vol. 76, Elsevier, 1982, pp. 67–112 (cit. on pp. 3, 4).

[8] J. A. Subirana and X. Messeguer, "Satellites in the prokaryote world," *BMC evolutionary biology*, vol. 19, no. 1, p. 181, 2019 (cit. on pp. 3, 4, 7, 13).

[9] S. T. Lovett, "Encoded errors: Mutations and rearrangements mediated by misalignment at repetitive dna sequences," *Molecular microbiology*, vol. 52, no. 5, pp. 1243–1253, 2004 (cit. on p. 4).

[10] L. A. Pray. (2008). DNA Replication and Causes of Mutation errors in dna replication, [Online]. Available: https://www.nature.com/scitable/topicpage/dna-replication-and-causes-of-mutation-409/ (visited on 12/26/2019) (cit. on p. 4).

[11] S. Kit, "Equilibrium sedimentation in density gradients of dna preparations from animal tissues," *Journal of molecular biology*, vol. 3, no. 6, 711–IN2, 1961 (cit. on p. 4).

[12] N. Sueoka, "Variation and heterogeneity of base composition of deoxyribonucleic acids: A compilation of old and new data," *Journal of Molecular Biology*, vol. 3, no. 1, 31–IN15, 1961 (cit. on p. 4).

[13] M. Waring and R. J. Britten, "Nucleotide sequence repetition: A rapidly reassociating fraction of mouse dna," *Science*, vol. 154, no. 3750, pp. 791–794, 1966 (cit. on p. 4).

[14] P. Novák, P. Neumann, and J. Macas, "Graph-based clustering and characterization of repetitive sequences in next-generation sequencing data," *BMC bioinformatics*, vol. 11, no. 1, p. 378, 2010 (cit. on p. 4).

[15] P. Novák, P. Neumann, J. Pech, J. Steinhaisl, and J. Macas, "Repeatexplorer: A galaxy-based web server for genome-wide characterization of eukaryotic repetitive elements from next-generation sequence reads," *Bioinformatics*, vol. 29, no. 6, pp. 792–793, 2013 (cit. on p. 4).

[16] P. Novák, L. Ávila Robledillo, A. Koblížková, I. Vrbová, P. Neumann, and J. Macas, "Tarean: A computational tool for identification and characterization of satellite dna from unassembled short reads," *Nucleic acids research*, vol. 45, no. 12, e111–e111, 2017 (cit. on p. 4).

[17] M. d. A. Dulio, R. Utsunomia, F. J. Ruiz-Ruano, S. N. Daniel, F. Porto-Foresti, D. T. Hashimoto, C. Oliveira, J. P. M. Camacho, and F. Foresti, "High-throughput analysis unveils a highly shared satellite dna library among three species of fish genus astyanax," *Scientific reports*, vol. 7, no. 1, p. 12 726, 2017 (cit. on p. 4).

[18] G. Benson, "Tandem repeats finder: A program to analyze dna sequences," *Nucleic acids research*, vol. 27, no. 2, pp. 573–580, 1999 (cit. on pp. 4, 5, 17).

[19] J. Macas, T. Meszaros, and M. Nouzova, "Plantsat: A specialized database for plant satellite repeats," *Bioinformatics*, vol. 18, no. 1, pp. 28–35, 2002 (cit. on pp. 5, 17).

[20] Django Software Foundation and individual contributors, *Django*, version 2.1. [Online]. Available: https://www.djangoproject.com/ (cit. on p. 8).

[21] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009, ISBN: 1441412697 (cit. on p. 8).

[22] M. Stonebraker and G. Kemnitz, "The postgres next generation database management system," *Communications of the ACM*, vol. 34, no. 10, pp. 78–92, 1991 (cit. on p. 8).

[23] GitHub, *Atom*, version 1.42.1, Dec. 16, 2019. [Online]. Available: https://atom.io/ (cit. on p. 9).

[24] J. P. Waagmeester, *Django tables 2*, https://github.com/jieter/django-tables2, 2019 (cit. on p. 10).

[25] C. Gibson, *Django filter*, https://github.com/carltongibson/django-filter, 2016 (cit. on p. 10).

[26] F. D. Gregorio, *Psycopg2*, https://github.com/psycopg/psycopg2, 2019 (cit. on p. 10).