Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Sequence based classification of bacteriophages

Bachelor Thesis

2018
Andrej Baláž

Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics

Sequence based classification of bacteriophages

Bachelor Thesis

Study programme:	Bioinformatics
Study field:	Informatics and Biology
Department:	Department of Computer Science
Supervisor:	Mgr. Jaroslav Budiš

Bratislava, 2018
Andrej Baláž

40313610

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Andrej Baláž
Study programme:	Bioinformatics (Joint degree study, bachelor I. deg., full time form)
Field of Study:	Computer Science, Informatics Biology
Type of Thesis:	Bachelor´s thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Sequence based classification of bacteriophages

Annotation: Bacteriophages are viral organisms that infect and replicate inside bacterial organisms. Since they are highly specific to a particular strains of bacteria, their presence in biological samples may identify the exact form of bacterial infection and thus lead to a more precise diagnosis for a targeted treatment. The main complication in rutine diagnosis is unclear understanding of underlaying mechanisms behind their function.

Aim: The goal of the thesis would be to discover relationship between functional components of bacteriophage sequence and its potential host. Student would prepare a database of known bacteriophages and their hosts. He would identify their functional elements and cluster them according to genomic sequence similarities. He would propose a classification method based on components that are highly enriched in bacteriophages of specific hosts.

Supervisor:	Mgr. Jaroslav Budiš
Department:	FMFI.KI - Department of Computer Science
Head of department:	prof. RNDr. Martin Škoviera, PhD.
Assigned:	24.10.2017
Approved:	25.10.2017

doc. Mgr. Bronislava Brejová, PhD.
Guarantor of Study Programme

.... Student

.... Supervisor

40313610

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Andrej Baláž
Študijný program:	bioinformatika (Medziodborové štúdium, bakalársky I. st., denná forma)
Študijné odbory:	informatika biológia
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	anglický
Sekundárny jazyk:	slovenský

Názov: Sequence based classification of bacteriophages
Klasifikácia bakteriofágov na základe ich genomickej sekvencie

Anotácia: Bakteriofágy sú vírusové organizmy infikujúce bakteriálnych hostiteľov, v ktorých sa aj replikujú. Vďaka ich vysokej špecifickosti ku konkrétnemu kmeňu baktérií, ich prítomnosť v biologických vzorkách môže identifikovať presnú formu bakteriálnej infekcie a viesť k precíznej diagnóze a následnej cielenej liečbe. Hlavná komplikácia v rutinnej diagnóze je nejasné pochopenie ich funkčných mechanizmov.

Cieľ: Cieľom práce bude objaviť vzťahy medzi funkčnými komponentami sekvencií bakteriofágov a ich potenciálnymi hostiteľmi. Študent pripraví databázu známych bakteriofágov a ich hostiteľov. Lokalizuje ich funkčné elementy a nájde skupiny na základe podobnosti genomických sekvencií. Navrhne metódy ich klasifikácie na základe komponentov, ktoré sú vysoko zastúpené v bakteriofágoch konkrétnych hostiteľov.

Vedúci:	Mgr. Jaroslav Budiš
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	prof. RNDr. Martin Škoviera, PhD.
Dátum zadania:	24.10.2017

Dátum schválenia: 25.10.2017 doc. Mgr. Bronislava Brejová, PhD.
garant študijného programu

....
študent vedúci práce

Acknowledgement: I would like to thank my supervisor Mgr. Jaroslav Budiš for his guidance during the work on this thesis and also during my career in the field of bioinformatics. His patience and his challenging tasks created great environment for my learning and development. I would also like to thank my colleagues for their inspiring conversations and advices. I am also thankful to doc. Mgr. Bronislava Brejová, PhD. and doc. Mgr. Tomáš Vinař, PhD. for their comments on my work, which helped to improve its quality. Special thanks for my family and friends for their support during my studies.

Abstract

Bacteriophages are viral particles that infect and replicate inside bacterial organisms. Since they are specific to a particular strain of bacteria, advances in their research could lead to novel means of targeted treatment without adverse effects on natural microbiome in patient's body. Moreover, this treatment could be effective against bacterial strains with antibiotic resistance. We consider as one of the main bottlenecks of bacteriophage research the inability to cultivate some of the phages due to missing information about their hosts and insufficient description of their genes. We aim to address this issue with our bioinformatics pipeline which can predict bacteriophage hosts from genomic sequence and which can give us additional information about gene importance for their function.

Keywords: bacteriophage, bioinformatics, host prediction

Abstrakt

Bakteriofágy sú vírusové častice, ktoré infikujú baktérie a replikujú sa v nich. Pretože sú vysoko špecifické na konkrétny kmeň baktérií, pokrok v ich výskume by mohol viesť k novým prostriedkom liečby bakteriálnych infekcií bez vedľajších účinkov na prirodzený mikrobióm pacienta. Taktiež by táto liečba mohla byť efektívna proti multirezistentným baktériám. Pokladáme za jedno z hlavných úskalí vo vývoji týchto metód neschopnosť kultivácie niektorých bakteriofágov s chýbajúcimi informáciami o ich hostiteľoch a nedostatočný popis ich génov. Tento problém sa snažíme riešiť v našej práci bioinformatickou metódou určenou na predikciu hostiteľov z genomickej sekvencie bakteriofágov. Tento postup nám tiež umožňuje získať informácie o dôležitosti jednotlivých génov pre funkciu bakteriofágov.

Kľúčové slová: bakteriofágy, bioinformatika, predikcia hostiteľa

Contents

List of Figures

List of Tables

Introduction

Discovery of penicillin was one of the most important discoveries of the last century. Many similar antibiotic substances were discovered after penicillin proved itself as an efficient weapon against bacteria. In the year 1940, antibiotics were first prescribed as medication to serious bacterial infections. Since then, antibiotics helped mankind to fight bacteria and saved millions of lives.

However, easy access and their excessive use led to overuse, resulting in bacterial strain immune to antibiotic treatment. In less than 10 years from introduction of penicillin to the market, penicillin resistant bacterial strain became a significant problem. The drugs discovered after penicillin followed a similar trend.

In the year 1972, vancomycin was introduced to the market. Resistance to this drug was so problematic to induce in laboratory conditions that it was believed it will not develop resistance in real conditions. Unfortunately, resistance to this drug was reported in 1979 and 1983.

Nowadays, resistance to nearly all developed antibiotics was observed. The pace at which mankind is able to discover new medicaments for treating bacterial infections is slowing down, which, together with the relatively fast pace of resistance growth, could cause a severe problem.

Bacteriophages, natural predators of bacteria, could introduce novel methods in the battle against bacterial infections. Although, they are well known to mankind for over hundred years, research in this field was slowed down after introducing antibiotics to the public. Therefore, we still do not know a lot of details about them. Sometimes even basic information about their hosts cannot be found in databases.

In our work, we will propose a method for predicting a host of a bacteriophage based on its genomic sequence. We will also show, how our method could provide a new insight on data about bacteriophages available online.

In the first chapter, we will describe basic terms of the molecular biology as they will be important for the work. We will put certain terms into context, such as nucleic acids and amino acids. We will also picture how we can work with data representing these terms in the computer. Bacteriophages, central topic of this work, will be introduced in this chapter as well. We will provide details about their role in the environment, their structure, taxonomical classification, life cycle and their potential as antibacterial agents. At the end of this chapter, we will summarize approaches to predicting hosts of bacteriophages taken by other researchers.

The second chapter will be devoted to bioinformatics algorithms used in the work. Comprehensive description of the alignment problem will be provided. We will define the problem and we will illustrate algorithms used for solving it. Moreover, data clustering will be described in this chapter.

In the third chapter we will provide detailed description of the created software for prediction of bacteriophages hosts. Because this software will use a lot of third party tools, we will describe them in detail. Furthermore, the most common biological databases will be mentioned. We will depict each step of our program and we will provide reasons for decisions made during programming.

In the fourth chapter we will show methods used for data analysis and how we interpreted the results. We will illustrate principal component analysis and the decision tree classifier. Results from these analyses will be provided. We will also show how we evaluated these results to support method correctness. At the end we will discuss limitations of this work and we will outline ideas for further work.

Chapter 1

Biological background

In this chapter we will present basic biological terms, which we will use later in this work. Firstly, we will describe molecules performing functions in living organisms and their representation in bioinformatics and computational biology. Then, we will explain relationships between these molecules. We will also provide detailed description of bacteriophages, which will be the central topic of our work. Description of what they are and where we can find them will be provided. Furthermore, we will explain their life cycle and we will point out some of their use. We will also show currently valid taxonomical classification and the structure of a typical representative of this group of organisms. At the end of this chapter we will point out what was already done in the field of their hosts prediction and what we would like to achieve.

1.1 Biological molecules

Molecules are basic components of every living organism on Earth. Vast majority of biological function are mediated by them. Next, we will present the central types of molecules in molecular biology, which we will use in our analysis.

1.1.1 Deoxyribonucleic acid

Deoxyribonucleic acid (DNA) is a long chain of nucleotides (bases) of four types - adenine, cytosine, guanine and thymine. These bases are connected through a phosphodiester bond, connection between the 3' carbon atom of one deoxyribose and the 5' carbon atom of the second deoxyribose. The structure of DNA consists of two complementary strands coiled around each other in the form of a double helix. In this double helix adenine is paired with thymine and cytosine with guanine through hydrogen bonds [1].

In bioinformatics and computational biology, we usually represent a base by its first letter - A for adenine, C for cytosine, G for guanine and T for thymine. Due to linear organisation of nucleotides, a DNA molecule can be represented as a word from alphabet {A, C, G, T}, called DNA sequence. We usually store biological sequences in simple plain text files. Probably the most commonly used format is the FASTA format. It can consist of one or multiple records, each representing a single DNA sequence. Each record starts with the identifier of the sequence, which is followed by lines of the DNA sequence written in direction from 5' to 3'.

1.1.2 Ribonucleic acid

Ribonucleic acid (RNA) is, similar to DNA, it is a polymeric molecule consisting of four types of nucleotides. There are three main differences between DNA and RNA molecules. Firstly, the sugar-phosphate backbone of RNA contains ribose instead of deoxyribose. This change in structure makes RNA less stable as it is more prone to hydrolysis. Another distinctive feature of RNA is that it contains uracil instead of thymine. Additionally, RNA appears in nature mostly as a single-stranded molecule, whereas DNA is mostly double-stranded [1]. This characteristic allows RNA to form more complex structures and show enzymatic activity. In bioinformatics, sequences of RNA are usually written, similarly as with DNA, from 5' to 3', with the difference of U for uracil instead of T for thymine.

1.1.3 Protein

Proteins are polymers consisting of amino acids connected by a peptide bond. There are 20 basic proteinogenic amino acids encoded in DNA sequences and 2 proteinogenic amino acids incorporated into proteins by a unique mechanism. Protein performs overwhelming majority of functions in living organisms. Thanks to this characteristic, they will be of high importance in our analysis. In bioinformatics, protein sequences are represented as a string of 1-letter abbreviations of their amino acids. All proteinogenic amino acids with their 1-letter codes can be found in the Table (1.1)

1.2 Flow of genetic information

Genome is a complete genetic material of an organism. It usually consists of DNA molecules, but some organisms can have genome in the form of RNA molecules. In genome, there is written entire information about the composition of particular organism with functional elements included. These functional elements, also called *genes*, are encoded in the sequence of DNA or RNA. Genes can be located in genomes using bioinformatics tools, which usually search for coding DNA sequences (CDSs). CDS is a region of genome that starts with the start codon and end with the stop codon. These

amino acid	1-letter code	amino acid	1-letter code
alanine	A	arginine	R
asparagine	N	aspartic acid	D
cysteine	C	glutamine	Q
glutamic acid	E	glycine	G
histidine	H	isoleucine	I
leucine	L	lysine	K
methionine	M	phenylalanine	F
proline	P	serine	S
threonine	T	tryptophan	W
tyrosine	Y	valine	V
selenocysteine	U	pyrrolysine	O

Table 1.1: Amino acids

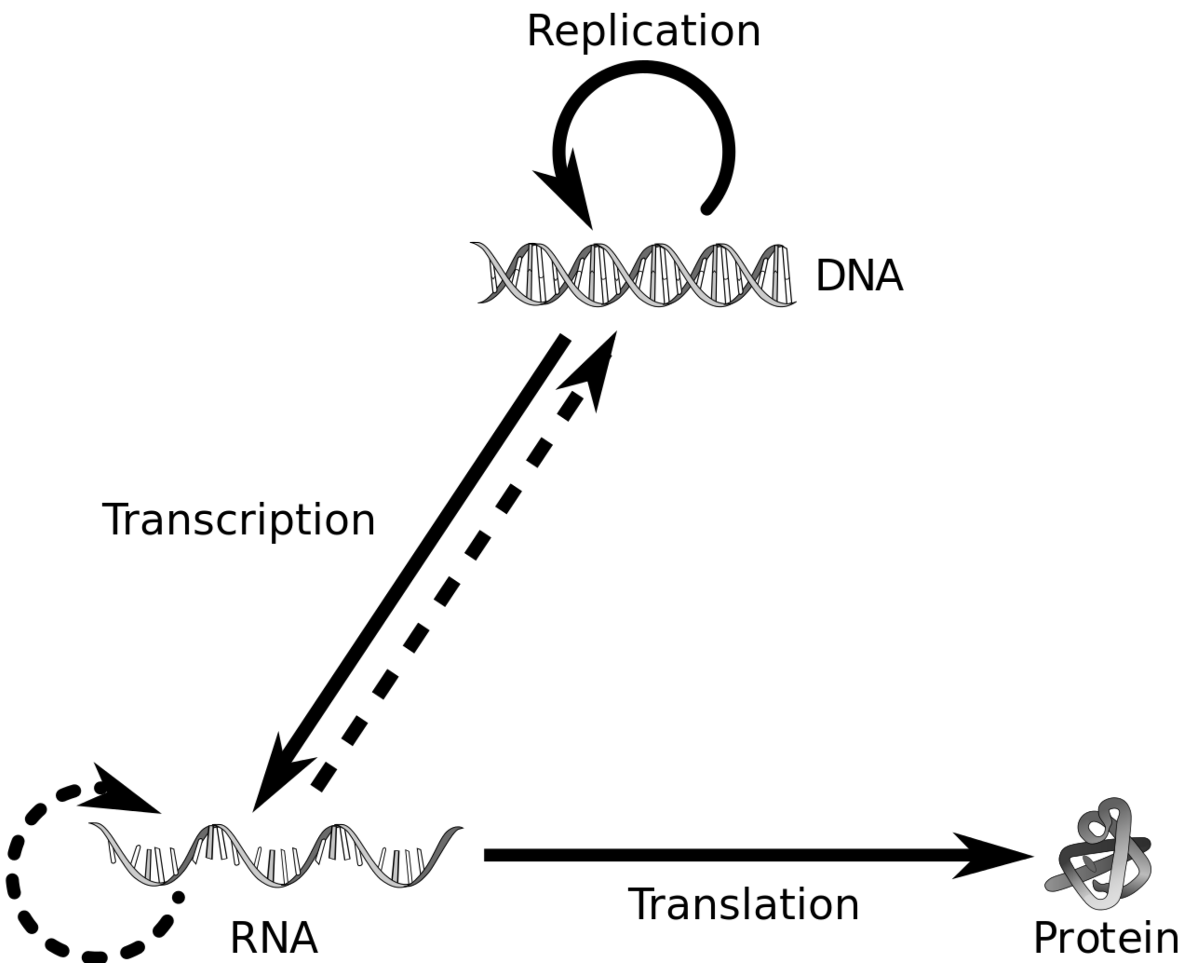
regions can be directly translated into amino acid chains using *standard codon table*. In this table, every combination of three nucleotides corresponds to amino acid, with exception of the combinations TAA, TAG and TGA, which correspond to stop codons. The relationships between biological molecules and the flow of genetic information in most living organisms are described in *central dogma of molecular biology*. Most organisms store their genetic information in the nucleus in the form of DNA. Some parts of that DNA are transcribed into the RNA molecules. Afterwards, ribosome translates the RNA sequence into a protein that is based on the codon table. Proteins are then folded into their natural 3D structure and prepared to realize their corresponding functions.

Despite of central dogma of molecular biology needs advanced compartments as ribosome for ensuring the flow of genetic material, not all organisms do have these compartments. Viruses tends to exploit hosts mechanisms for expression of viral genes and replication of themselves.

We based our work on the central dogma of molecular biology and on the assumption that genes will be useful indicators when predicting the ability to infect certain hosts.

1.3 Bacteriophages

Bacteriophages (phages) belong to the group of viruses, which have the capability to infect and replicate within bacterial hosts. It is estimated that they are the most abundant group of entities on Earth with their estimated count around 10^{31}[3]. In comparison, it is estimated that the count of all bacteria on planet Earth is around 10^{30}. They are one of the most highly diverse group in the biosphere, with their



[2]

Figure 1.1: Central dogma of molecular biology, arrows show the flow of information between biological molecules, full lines symbolizing usual flow in living organisms and dashed line symbolizing flow of information used by some primitive organisms (viruses)

genomes spanning from few genes to as many as hundreds of genes. We can find them in every place on earth, where bacteria are able to live, even inside our bodies. It is believed that one of the most saturated location of their occurrence is sea, with $9 \cdot 10^8$ virions per milliliter of seawater at the surface and 70% of bacteria infected [4]. Thanks to the high abundance of bacteriophages, their impact on shaping of the environment is outstandingly significant, reducing considerable amount of bacteria.

1.3.1 Taxonomical classification

Classification of bacteriophages is difficult mainly due to their high diversity and genome mosaicism. [5] Consequently, there exists no universal marker similar to universal markers in bacteria, according to which we will be able to classify bacteriophages based on their genetic information. This is because no genes are strongly conserved within all bacteriophages. Despite of these facts, there is taxonomical classification of phages. This classification was created by International Committee on Taxonomy of Viruses (ICTV) and it categorizes each phage according to its morphology and nucleic acid. This taxonomy recognizes nineteen different families of phages. Although this taxonomy is currently in use, many biologists feel it is outdated and in need of revision. [5]

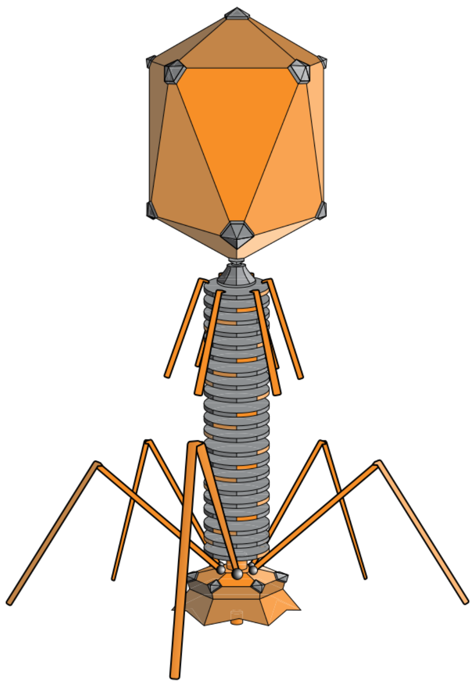
1.3.2 Structure of a typical bacteriophage

Given the high variety of bacteriophages, they come in a lot of different sizes and shapes. Each bacteriophage consists of genetic information in form of DNA or RNA and capsid, a protein coat usually composited from higher number of identical protein units [6]. We will describe the most typical form of tailed bacteriophages, which is abundantly found in nature.

As we can see in the Figure 1.2, this bacteriophages body consists of a head and a tail. These parts are created from proteins and in addition, the head contains DNA or RNA of the bacteriophage genome. The tail is used to attach and to inject the genetic code into bacteria. At the end of the tail, there are proteins, which are able to bind to specific receptors on the surface of a bacteria. Thanks to this mechanism, bacteriophages tend to have a high specificity in selection of their prey.

1.3.3 Life cycle of bacteriophages

Generally, there are two strategies that bacteriophages use to secure their survival and replication; the *lysogenic cycle* and the *lytic cycle* [8]. Viruses tend to use these strategies in different proportions, but usually prefer to choose one of them. The lysogenic cycle results in incorporations of phage genetic information into host DNA



[7]

Figure 1.2: Structure of a typical bacteriophage, it is composited by head, in which the genetic material is stored and tail, through which the genetic material is passed during the infection

or creation of a circular replicon in the bacterial cytoplasm. By this approach, the genetic material of a phage inside the host, called prophage, is duplicated together with the host genome and after cell division, both daughter cells contain DNA of the bacteriophage. In dependence on the following events, viral DNA can be released from hosts genome and start proliferation of new phages via the lytic cycle. The lytic cycle is characterized by the lysis of bacterial cells membrane and their subsequent death. It starts by injecting bacteriophage genome into bacteria. After this step virus is not incorporated, but it compromises bacterial translation apparatus to produce more viruses. Once enough virions have been produced, special viral proteins dissolve the bacterial cell and virions are released into surrounding space.

1.3.4 Potential usage of bacteriophages

Their ability to kill or alter behavior of highly specific strains of bacteria makes bacteriophages a valuable target for research. Humankind tried to harness the power of phages since their discovery in the year 1917. This discovery is attributed to a French-Canadian microbiologist Félix d'Hérelle, who experimented extensively with phages and introduced the concept of *phage therapy*[9]. Unfortunately, after the discovery of antibiotics, research in this field suffered from insufficient funding and the development

was significantly slowed down.
Nowadays, when humanity face the threat of multiresistant bacteria and phages could bring a new methods into the battle against pathogenic bacteria, the research in this field begins to flourish. We can see the potential of their use in the field of food industry as the substances that are prolonging life and improve the quality of our food [10]. In the field of pharmacy, they could be used as medicines for bacterial infections [11]. Their potential of use is significant where we encounter the need to control the lives of microbial communities. Moreover, due to their high specificity to a particular strain of bacteria, these new practices will likely be without adverse effects on natural microbiome.

1.3.5 Host identification

In our work we focus on prediction of phage hosts from genomic sequence. We based our method on the assumption that bacteriophages with similar set of genes will infect the same bacterial host. The decision to create host predicting models was made based on its straightforward use in searching for phages suitable for phage therapy.

Attempts to classify bacteriophages from a genomic sequence were already made. In the study "The Phage Proteomic Tree: a Genome-Based Taxonomy for Phage"[3] from year 2002, researchers performed analysis based on genomes resulting in phylogenetic tree compatible with ICTV system. In their work, they proved phages indeed do not contain any universal genetic marker, which could be used as sequence for classification. They also showed that classification based on whole proteome of phage is a reasonable approach. In 2016 a similar tool, called HostPhinder, for predicting hosts from genomic sequences appeared [12]. Their approach to classification was through rate of similar k-mers, short sequences of nucleotides of length k. With this approach they achieved the results of 81% of correctly predicted genera. In our work we tried a different approach, where we looked for similar genes instead of rate of similar k-mers.

Chapter 2

Bioinformatics methods

In this chapter we would like to thoroughly explain methods and algorithms used in bioinformatics. Firstly, we will explain details about sequence alignment. We used tools implementing ideas of sequence alignment in the practical part of this work, therefore it is important to clarify what it is and how it is realized. Secondly, we will explain data clustering. We will introduce the concept of it, how we can use it and we will describe algorithm we used in the work.

2.1 Sequence alignment

In this section we would like to thoroughly explain what sequence alignment is, since it is one of the most essential tasks in bioinformatics. We will describe two main types of sequence alignment, with emphasizing the differences. We will also explain the basic algorithms used for solving these problems and we will also describe the heuristic method used in case the input is too big to be computable by standard approaches that guarantee optimal solution.

2.1.1 Global alignment

Usage

Global alignment is commonly used for comparison between two sequences with roughly the same length. This comparison can help us to discover mutations in a sequence that causes a certain phenotype, or we can use the comparison to determine highly conserved regions with a potential to carry a gene. The relationships between conserved regions and mutations in certain sequences often serve as a basic assumption for construction of phylogenetic trees. Therefore, global alignment can tell us a lot about the nature of evolution. Global alignment is also capable of calculating a similarity score for two sequences.

`A-TTGATGG`	`-ATTG-ATGG`
`AATTCAAC-`	`AATTCAAC--`

Table 2.1: Global alignment examples

Problem statement

Let the input for the problem be a set of two sequences consisting of nucleotides, for example, $X = ATTGATGG$ and $Y = AATTCAAC$. Then the output is represented as a matrix, where each row is one sequence with possible gaps between nucleotides, representing insertion or deletion of the sequence. We can see potential solutions in the Table (2.1)

As we can see, every alignment could have more than one valid solution. Despite of multiple possible solutions, not every solution is of equal value to us. We want to find the best possible solution to this problem and for this purpose, we implement a scoring scheme to determine what is the best solution. *Scoring scheme* consists of rules, which add numerical value to each column of pairwise alignment. For example, we can evaluate match in column with score $+1$, mismatch with score -1 and alignment to gap with score -1. With this scoring scheme, we can evaluate quality of a particular alignment. For alignment on the left side of the Table (2.1), resulting score is $+1 - 1 + 1 + 1 - 1 + 1 - 1 - 1 - 1 = -1$ and for alignment on the right side of the Table (2.1) $-1 + 1 + 1 + 1 - 1 - 1 + 1 - 1 - 1 - 1 = -2$. From this example we can see that according to our scoring scheme, alignment on the left is better.

Scoring schemes

In practice, we could use a more complex scoring scheme that better reflects reality. For example, substitution between purines (adenine, guanine) or substitution between pyrimidines (cytosine, thymine) occur more often, because they do not require change in the number of rings in the chemical structure of these nucleotides [13]. When we try to align two proteins, a more complex scoring scheme is inevitable. Amino acids differ in many parameters such as: polarity, size and structure of their side chains. This influences the probability of a substitution occurring between two amino acids. Therefore, there is a higher probability that Leucine will be substituted for Isoleucine, rather than Aspartic Acid. To solve the complexity of substitutions between amino acids, matrix BLOSUM62 (2.1) was created. Another layer of complexity, that does a better job of reflecting reality is the *affine gap penalty function.* This reflects the fact that insertions and deletions do not usually occur only on one nucleotide, but often a longer region of DNA is deleted or inserted. Affine gap penalty solves this issue by having a higher negative score for opening a new gap in alignment and a lower negative gap for extension of an already created gap.

Ala	4																			
Arg	−1	5																		
Asn	−2	0	6																	
Asp	−2	−2	1	6																
Cys	0	−3	−3	−3	9															
Gln	−1	1	0	0	−3	5														
Glu	−1	0	0	2	−4	2	5													
Gly	0	−2	0	−1	−3	−2	−2	6												
His	−2	0	1	−1	−3	0	0	−2	8											
Ile	−1	−3	−3	−3	−1	−3	−3	−4	−3	4										
Leu	−1	−2	−3	−4	−1	−2	−3	−4	−3	2	4									
Lys	−1	2	0	−1	−3	1	1	−2	−1	−3	−2	5								
Met	−1	−1	−2	−3	−1	0	−2	−3	−2	1	2	−1	5							
Phe	−2	−3	−3	−3	−2	−3	−3	−3	−1	0	0	−3	0	6						
Pro	−1	−2	−2	−1	−3	−1	−1	−2	−2	−3	−3	−1	−2	−4	7					
Ser	1	−1	1	0	−1	0	0	0	−1	−2	−2	0	−1	−2	−1	4				
Thr	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	1	5			
Trp	−3	−3	−4	−4	−2	−2	−3	−2	−2	−3	−2	−3	−1	1	−4	−3	−2	11		
Tyr	−2	−2	−2	−3	−2	−1	−2	−3	2	−1	−1	−2	−1	3	−3	−2	−2	2	7	
Val	0	−3	−3	−3	−1	−2	−2	−3	−3	3	1	−2	1	−1	−2	−2	0	−3	−1	4
	Ala	**Arg**	**Asn**	**Asp**	**Cys**	**Gln**	**Glu**	**Gly**	**His**	**Ile**	**Leu**	**Lys**	**Met**	**Phe**	**Pro**	**Ser**	**Thr**	**Trp**	**Tyr**	**Val**

[14]

Figure 2.1: BLOSUM62, table providing score of substitution between all pairs of amino acids, commonly used in the protein alignment

Needleman-Wunsch algorithm

For searching optimal global alignment, we usually use the Needleman-Wunsch algorithm. It is an algorithm from a group of *dynamic programming* algorithms. This means that the main problem is divided into smaller problems, which are computable more easily and solutions to them are stored in memory. At each occurrence of a small problem, we can look into stored solutions, where we can find it. Then the main problem is reconstructed from already computed subproblems. Dynamic programming algorithms offer saving time on computation at the expense of a higher memory usage.

Needleman-Wunsch algorithm produces a table as shown in the Figure (2.2). It starts by putting the first sequence we want to align to the first row and the second sequence to the first column. Before each sequence there is one gap to cover the case if we would not want to align first letter of a particular sequence right from the beginning. The table is then initialized with a series starting from 0 and decreasing by 1 each step on the second row and with the same series on the second column. After initialization, the table starts to be filled from top left corner following this rule:
Into each cell $A_{i,j}$ write maximum of:

- $A_{i-1,j-1} + s(X_i, Y_j)$,
- $A_{i-1,j} + g()$,

```
--TAATAACTCTCTGAATAA
          ||||||
CGGCGGCGGTCTCTGCC---
```

Table 2.2: Local alignment example

- $A_{i,j-1} + g()$,

where $s(X_i, Y_j)$ returns score of match/mismatch from scoring scheme and $g()$ returns a score of the gap penalty (possibly affine gap). The final score of the alignment can be found in the bottom right corner of the filled table. Specific alignments can be found by tracking all possible paths to this value. The time and space complexity of Needleman-Wunsch algorithm is $\mathcal{O}(nm)$, where n is the length of the first sequence and m is the length of the second sequence.

2.1.2 Local alignment

Usage

In comparison with global alignment, *local alignment* search for regions inside sequences with high similarities and does not provide alignment from beginning to end. It is generally useful when searching for a small subsequence inside a vast sequence. For example, searching for a gene inside the whole bacterial genome. It can be also used when comparing two different sequences and we want to find out if they contain any highly similar sequence.

Problem statement

Similarly to what we have done with the global alignment, in this problem we search for optimal local alignment according to the defined scoring system. The difference is that we do not know where the alignment in both sequences starts and where it ends. For example, $X = TAATAACTCTCTGAATAA$ and $Y = CGGCGGCGGTCTCTGCC$ can be aligned as in the Figure (2.2) and the score is calculated just from the first aligned base to the last aligned base.

Scoring

Scoring of local alignment is similar to global alignment and we are allowed to use the same methods we use in the global alignment. Since the score is calculated just from the first aligned base to the last aligned base, the score of our local alignment would be $+1 + 1 + 1 + 1 + 1 + 1 = 6$, because there are 6 matches $(+1)$ and no gaps (-1) or mismatches (-1).

Smith-Waterman algorithm

Local alignment can be found with dynamic programming algorithm similar to Needleman-Wunsch algorithm. This is called Smith-Waterman algorithm and there are only two changes compared to Needleman-Wunsch. First change is that matrix is not initialized with decreasing series, but second row and second column are filled with zeroes. Second difference is in the rule as follows:
Into each cell $A_{i,j}$ write maximum of:

- 0,
- $A_{i-1,j-1} + s()$,
- $A_{i-1,j} + g()$,
- $A_{i,j-1} + g()$.

After completing the table, we need to find the highest number in it. This number is a resulting score of our local alignment. Following the path similarly as in global alignment we can reconstruct the alignment. The space and time complexity of this algorithm is also $\mathcal{O}(nm)$.

2.1.3 Word methods

Needleman-Wunsch and Smith-Waterman algorithms are sufficient for simple comparisons of sequences, but their time complexity is not good enough when we want to search through very long sequences that are common in genomics. It is often the case that we want to find the most similar sequence to ours in enormous bioinformatics database containing genomes of large amount of organisms. This is particularly useful if we want to find the potential source of our sequence or comparing it to all known proteins to get some indicators about its potential function. For this purpose, various *heuristic algorithms* were developed. These algorithms do not guarantee finding the most optimal solution but are orders of magnitude faster and therefore usable also for searching in vast databases.

BLAST

Basic Local Alignment Search Tool (BLAST) [16] is probably the most widely used tool in bioinformatics. The algorithm distinguishes between target sequences and query sequences. Target sequences are sequences from which we create a database of all included k-mers. For every k-mer in every sequence we save its position. For example, for sequence GATCGATAG and given $k = 3$ we create database as follows:

```
GAT = 1, 5
ATC = 2
TCG = 3
CGA = 4
ATA = 6
TAG = 7
```

Next, we find every k-mer from query sequence in the database. The matching k-mers are called cores of alignment and serves as starting points for extending of alignment. The extending is performed in a similar manner to that of Smith-Waterman algorithm. The difference is that we do not need to search through the whole matrix, but we search for good alignments only in the immediate surrounding of cores. Firstly, we extend cores without the possibility to insert any gaps into alignment. Secondly, we can join these extended cores together using the gaps, if the resulting score will be bigger than before joining.

E-value

E-value is used to assess the relevancy of the resulting alignment. It can be interpreted as how many alignments in database could reach the same score or better purely by chance. This means, the closer the E-value is to zero, the higher level of significance can be assumed. E-value is automatically produced by BLAST.

2.2 Data clustering

In this section we will explain what is data clustering. We will describe its use in data analysis and bioinformatics and how we used it in our program. We will briefly define the problem and we will explain the algorithm used for solving it.

2.2.1 Usage

Data clustering is used to group data according to similar characteristics. In computational biology it can be used to cluster organisms to examine relationships between different groups or to study structure of population. By clustering genes, we can investigate their relationships and then extend functional annotations of particular genes to their closely related genes. Clustering can also be used to distinguish different types of tissue or group drugs with similar effects according to their mechanism of action.

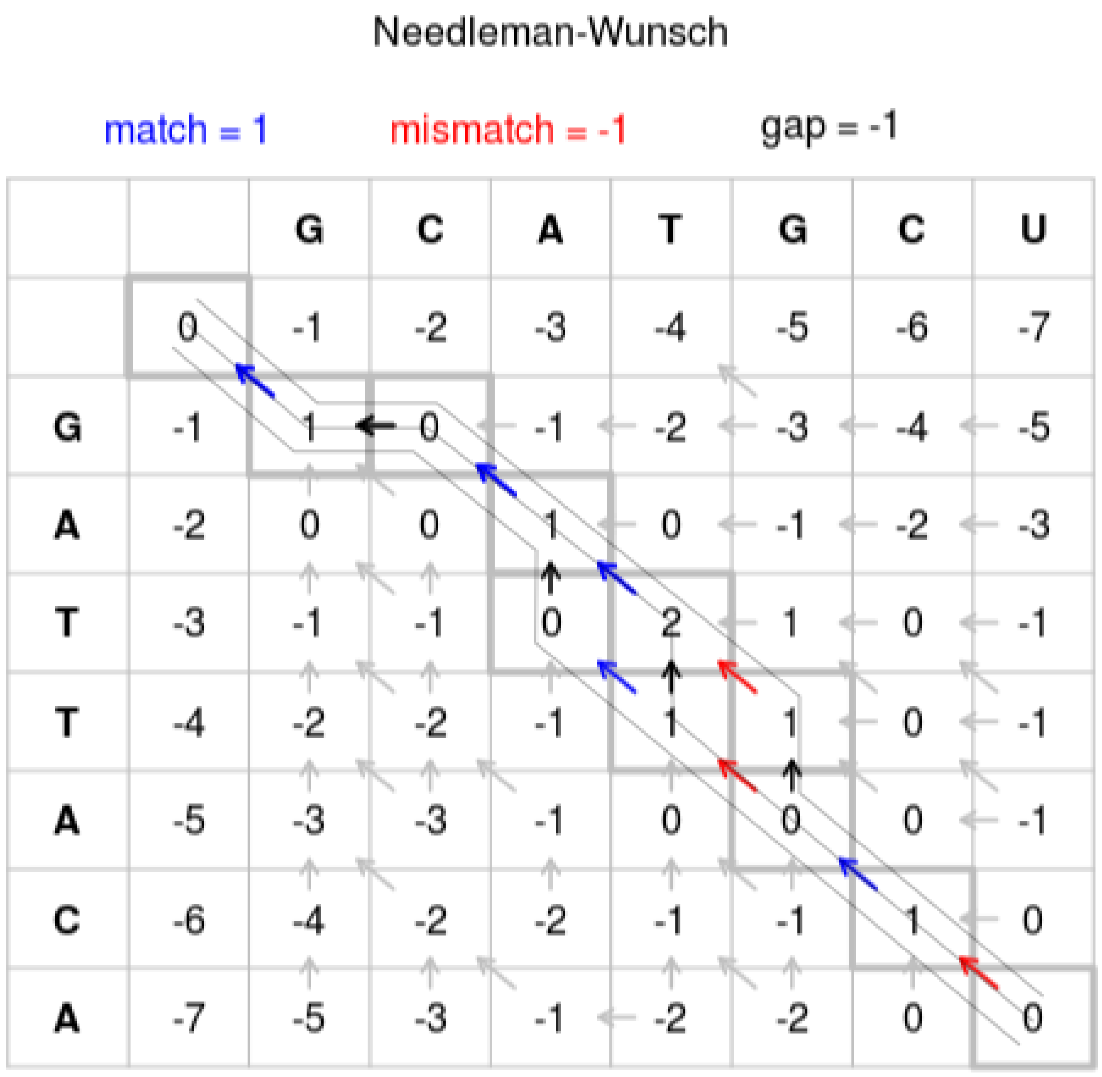
In our work, we clustered genes with sequential similarities. We expected, that this approach will create clusters of genes with closely related functions.

2.2.2 Definition of problem

Clustering is a problem where we want to determine closely related entities and put them in distinct groups, also called clusters. Cluster can be characterized by presence of many connections between cluster members. Members from different clusters have lesser probability to be connected and the distance between them is usually longer. Although, to intuitively understand clustering is not complicated, there exists a lot of clustering models with slightly different approaches, which makes exact general definition difficult. In our case, a graph representing our dataset was created. Vertices of this graph were individual genes and edges were alignments of corresponding sequences. Weights of these edges were alignment scores. Proper clustering of this graph was determined using Markov Cluster Algorithm.

2.2.3 Markov Cluster Algorithm

Markov Cluster Algorithm [17] is searching for clusters by simulating random walks on the graph. It assumes that these random walks will infrequently go from one cluster to different cluster and mostly will stay in the starting cluster. The algorithm simulates random walks through the graph deterministically with two operations, expansion and inflation. *Expansion* corresponds with squaring of stochastic matrix, non-negative matrix where each column sums to 1 and entry a_{ij} corresponds to probability of going from node j to node i. *Inflation* is defined by entry-wise powering of stochastic matrix to the power of r, followed by scaling step to get stochastic matrix again. It is observed that for $r > 1$ inflation favors more probable walks over less probable walks. Expansion can be interpreted as random walk with many steps. Inflation then enlarges the effect of random walks within a cluster and reduces random walks across a different cluster. By repeating expansion and inflation operations, the equilibrium is reached which produces the clusters. These clusters do not contain any paths between each other and the only paths in graph are those within clusters. This algorithm does not need any prior knowledge of clusters and resulting clusters emerge from the primary structure of the graph. Furthermore, size of clusters can be regulated by changing parameter r of inflation operation. Bigger r will cause algorithm to converge to equilibrium faster and will produce smaller resulting clusters. In practice, r from 1.2 to 5.0 is used.



		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

[15]

Figure 2.2: Needleman-Wunsch, visualization of table used in dynamic programming algorithm, arrows signalizing, from which cell the maximum was calculated, optimal solution is highlighted with coloured arrows

Chapter 3

Modelling bacteriophage genomes

In this chapter we will present parts of our pipeline used for data retrieval and manipulation. Since we used a considerable amount of third party libraries and tools, we will explain them in detail and we will clarify our reasons for using them. We will also provide some basic stats of the data as number of downloaded genomes and number of obtained genes to later support statistical significance of our results.

3.1 Pipeline overview

Our pipeline consists of python scripts and publicly available bioinformatics software. When writing the code, we have taken care of its readability, sustainability and extensibility.

We implemented our pipeline in workflow management system *Snakemake* [18] that was primarily designed for writing reproducible bioinformatics pipelines. Snakemake is inspired by GNU make, but it uses python-like syntax with elements similar to pseudo code. Furthermore, it is fully portable, depending only on Python executables and libraries. Snakefile consists of rules, where each rule is defined by its input files, output files and commands.

When the Snakemake is executed, it runs first rule in specified Snakefile. If the rule is missing input files, it scans through the whole Snakefile and looks for rules that are capable of creating required files. This process is repeated until there is a rule which can be completed or until there is a rule whose input is not possible to create by any other rule. In the former case the execution starts running, in the latter case an error message is displayed. By this approach it is ensured that we do not execute any unnecessary rules nor any rules that have been already completed. This is an important feature for our program as some rules can take several hours to complete, even on powerful computational cluster. Another useful characteristic of the Snakemake engine is the ability to produce graphical visualization of particular Snakefile in format of directed acyclic graph. Simplified graphical representation of our pipeline is in the Figure 3.1.

The pipeline starts with downloading of publicly available data. After downloading, we merge all records and eliminate duplicated records. Next, we extract genes of the bacteriophages. Consequently, phage genomes represented as sets of genes are split into a training and a testing set. Similarity between the genes from the training set are calculated and based on those, clusters of similar genes are produced. From these clusters, the binary matrix is created. This matrix is later used in classification.

3.2 Downloading phage genomes

The first step in our pipeline is downloading of data from three publicly available databases. Although they cover the majority of currently sequenced and published phages, we made this step easily extensible for adding new sources of information in the future. New sources can be added by writing a new download script, naming it `{script_dir}\download_from_{db}.py` and appending this name into a variable `DATABASES` in Snakefile.

3.2.1 GenBank database

National Center for Biotechnology Information (NCBI) provides an access to *GenBank*[19] database. This database is a comprehensive source of genomic data with more than 200 million genomic sequences of all life's domains. NCBI administers the GenBank database free of charge and give researchers the possibility to access data through various interfaces as web-based retrieval services, FTP and Entrez[20]. Despite of these facts, there are shortcomings of using GenBank. With recent breakthrough of high-throughput medical technologies the amount of data flowing into GenBank database every day is enormous. Therefore, it is unreasonable to check all the data. Sequences are primarily submitted by individuals from all around the globe and are not thouroughly reviewed. This causes redundancy of sequences and sometimes it even creates contradictions between information in system.

We obtained data using python library Biopython [21], which implements python wrapper NCBI Entrez. Besides that, we used Biopython to facilitate processing of standard file formats used in bioinformatics. When downloading sequences, we also created unique identifiers for each record. Those were used later in the pipeline. Reasons behind the decision to use custom identifiers was the ability to remove duplicated sequences and the possibility to find out multiple sources of each sequence in our dataset. Downloading from GenBank was our largest source of data with 6704 downloaded records.

3.2.2 ViralZone database

ViralZone provides highly reliable data about viruses, including bacteriophages. Information about the structure of a capsid, a genome, life cycle, replication mechanisms, taxonomy, geographical location and host are included. This website does not store sequences internally, rather it delivers links to *RefSeq*[22] database. Compared to GenBank, RefSeq database contains fewer sequences, but all of these sequences are curated and manually reviewed.

Our custom script was used to download records from RefSeq database. Although, large portion of sequences downloaded was identical with GenBank records, some sequences were unique. Another advantage in performing this action was that it enabled us to pair genomic sequences from RefSeq with more comprehensive information from ViralZone portal. By performing this process, we obtained 2107 records.

3.2.3 PhagesDB

PhagesDB is a database specialized in bacteriophages infecting bacteria from phylum Actinobacteria. This phylum is of great importance, because of its contribution to the soil system in the form of decomposing of organic matter. This phylum also contains the genus Mycobacterium, which includes pathogens causing tuberculosis and leprosy in humans [23]. The database was designed to avoid the time between sequencing and data availability. Authors declare, at the time of their publication, there was more than 600 records of bacteriophages that were not yet in GenBank. Furthermore, PhagesDB stores more biologically relevant data , such as discovery details, sequencing details, characterization details, sequence file and a plaque picture. We downloaded 2491 phage records with our automatized script using the publicly available Application Programming Interface (API).

3.2.4 Merging and removing of duplicated records

Downloaded records were highly redundant, mainly because many of those records were present in more databases at once. To solve this issue, we merged datasets together and removed duplicated records. For merging we used the standard unix command `cat`. For removing duplicated sequences, we created a custom Python script. This script made use of custom identifiers, which were assigned to every sequence that was downloaded. In case more identical sequences were found, their custom identifiers were rewritten with the identifier of the first sequence. This approach preserved the relationships between one particular sequence and all data related to it. Thus, we were able to track phages, based on their identifiers, to their source databases and also connect them with all data that was already downloaded. After removing of duplicated sequences our dataset contained 6277 phage records. This suggests high duplication ratio between the databases and even in the GenBank alone.

3.3 Extraction and annotation of genes

Next step in the pipeline was to identify genes on genomic sequences and annotate them with their biological function. Although gene annotations of particular genomes are part of genomic records in used databases, we decided to annotate sequences from scratch. This way we ensured consistency of annotations accross our data . Another advantage of annotating from scratch is that annotations will be up-to-date with current human knowledge.

3.3.1 Prokka

We used publicly available pipeline called *Prokka* [24] to identify and annotate genes. First, coordinates of coding DNA sequences (CDS) were found with Prodigal tool[25]. After the locations of genes are predicted, Prokka can start to annotate functions of all CDSs. This is usually done through comparing of a sequence to a database of sequences with an experimentally determined function. The function of protein with the best match is then assigned to the new CDS. Prokka, by using this approach searches through multiple databases. Starting with the most reliable source, which is usually the smallest, it scans all the databases, continuing with the less accurate one. The databases used with their corresponding order are as follows: An optional user-defined database, UniProt[26], RefSeq[22], Pfam[27] and TIGRFAM[28]. If no match is found across databases, protein is labelled as `hypothetical protein`. After annotation, resulting genes were selected and saved to files in format suitable for further use in the pipeline.

UniProt

The UniProt database is acollection of more than 60 million protein sequences and their corresponding detailed information. Prokka uses just a small fraction of proteins backed up by experimental evidences. This typically provides information about approximately 50% of queried proteins.

RefSeq

RefSeq, provides information about genomic and protein sequences. It contains more than 2.5 million protein records. Multiple sources are integrated in annotation of genes. Furthermore, all records are curated by the NCBI staff members.

Pfam

Pfam is a database consisting of protein families and domain records. Protein families are groups of protein sharing evolutionary history. This shared evolutionary history is often expressed by closely related functions and sequence similarity. Protein domains are parts of a protein sequence responsible for a particular interaction. Members of the same protein domain usually share a high sequence similarity. Protein families and domains are mostly characterized by Profile Hidden Markov Models. Pfam incorporates more than 6100 of those models.

TIGRFAM

TIGRFAM, similarly as Pfam, contains protein families characterized by Profile Hidden Markov Models. It contains more than 4200 models with comprehensive description of family structure and function.

3.4 Datasets used

At the beginning of this step, our dataset consisted of: phage genomic records, their corresponding genes, information about phage hosts and functional annotation of genes. As our work used techniques of supervised machine learning, we needed to split the dataset to a *training set* and a *testing set*. Furthermore, we created set for other records, which we decided not to use due to missing information about host or due to host outside of our group of interest. We decided to group phages according to the genus of their hosts. After calculating the number of phages in each group, we selected first eight genera with the highest count of records as groups of our interest. These were Mycobacterium, Streptococcus, Escherichia, Gordonia, Arthrobacter, Pseudomonas, Lactococcus and Staphylococcus. Number of phages within each group can be found in the Table (3.1). All other genera were excluded from the dataset due to the insufficient number of samples. Phages without information about their hosts were also excluded. Subsequently, we divided remaining data into a training set and a testing set at a ratio of 4:1. The resulting training set included 2787 records of bacteriophages and resulting testing set included 699 records.

3.5 Alignment of genes

In our pipeline, we used alignment to find similarity scores between genes in the training set. We needed these scores later in the pipeline at clustering step. Software CrocoBLAST [29] was used for this purpose. CrocoBLAST is a wrapper around BLAST algorithm which makes better use of parallelization than standard BLAST maintained

genera	count	genera	count
Mycobacterium	1619	Streptococcus	354
Escherichia	323	Gordonia	293
Arthrobacter	240	Pseudomonas	236
Lactococcus	219	Staphylococcus	184

Table 3.1: Counts of records within genera

by NCBI. With this software we were able to reduce the time needed for the alignment step from around four days to one day. Resulting file was in tab separated format, where first column was gene identifier of query sequence, second column was gene identifier of the target sequence and third column was e-value of alignment.

3.6 Clustering

3.6.1 MCL

Firstly, we used Markov Cluster Algorithm implemented in package MCL [17]. This software is popular in the bioinformatics community for its capabilities to work with big data. As input data we used E-values from CrocoBLAST results. These E-values were automatically transformed into a similarity score to enable creation of the adjacency matrix. We used the value 1.2 as the inflation parameter. This was the smallest value recommended by the developers. The reason for using the smallest possible inflation value was that we wanted to create clusters that are as large as possible This approach also reduced the number of different clusters. In our work, we needed as few reasonable clusters as possible, mainly due to the number of phage records in our dataset. If we had too many clusters, we would have too many features for classifier and we would risk overfitting of our final models to the training dataset. We created 15017 gene clusters.

3.6.2 MCL with global alignment

To get more accurate clustering, we tried different approach, where we determined similarity score from the global alignment. All the matches from CrocoBLAST search were aligned with the needleman-wunsch algorithm [?] and the resulting score of alignment was used instead of the E-value. Implementation of the Markov Cluster Algorithm was executed with an inflation parameter value of 1.2, without any transformation of the similarity score. By this approach we created 9176 final clusters.

3.6.3 Spectral clustering

In this work, we also experimented with a different clustering algorithm, Spectral clustering. SCPS implementation[30] was tested in the pipeline. Although authors of this software declare quality of clusters quantified by a measure that combines sensitivity and specificity to be better by 28% in comparison to MCL algorithm, our memory was not sufficient for the program to run with all the input data.

3.7 Annotation of gene clusters

Clusters of genes were annotated to determine their function. We expected proteins with similar biological function to be included in same cluster. For functional annotation of particular proteins we used software *InterProScan* [31]. The reason to use InterProScan annotations instead of Prokka annotations was because InterProScan annotation had better standardized descriptions of functions. This tool scans given protein sequences against the protein signatures in databases PROSITE [32], PRINTS [33], Pfam [27], ProDom [34] and SMART [35]. After acquiring of annotations for all proteins in a particular cluster, we calculated number of occurrences of each distinct biological function. These statistics represented our annotation of a certain cluster. We manually reviewed the annotations of the biggest clusters to evaluate relevancy of created clusters. Although a lot of proteins remained without any assigned function, our expectation of clusters containing proteins with similar function was met in most cases. Therefore, we assumed that reasonable clustering was achieved.

3.8 Reducing phage genomes

One of the most crucial part of our analysis was *binary matrix* created in this step. Rows in this matrix represented particular phage records and columns represented particular protein clusters. The entry $a_{i,j}$ in matrix was filled with 1 if phage i contained gene from cluster j and 0 otherwise. Custom python script and files produced in previous steps were used for this task. Resulting matrix contained 2787 rows and 15017 columns and served as a main input file for the machine learning algorithms.

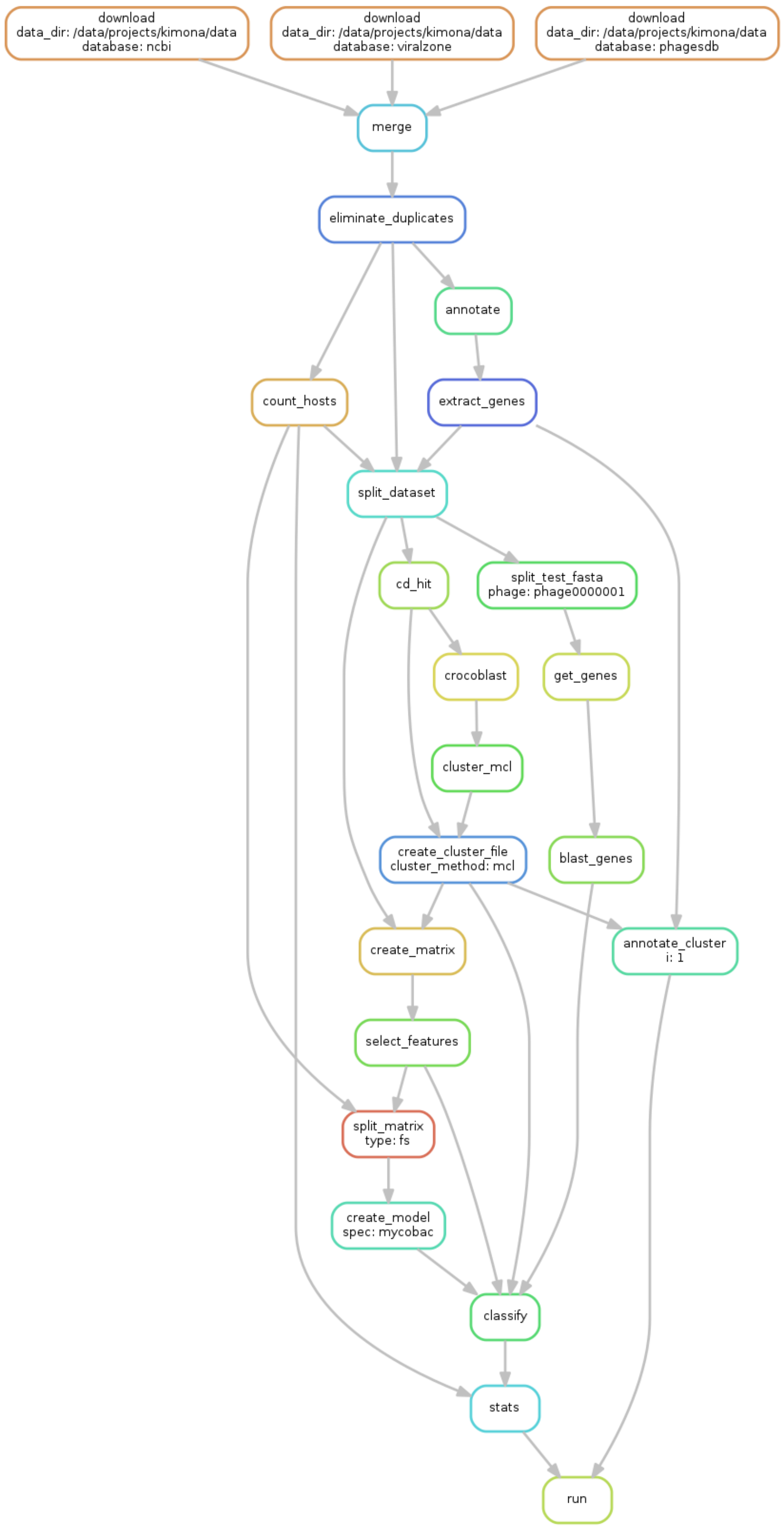


Figure 3.1: Workflow visualization, cells symbolizing rules in workflow, arrows create acyclic graph, where rules at the top need to be completed before executing rules at the bottom

Chapter 4

Data analysis and results

In this chapter we will describe approaches we used for analysis of binary matrix created in previous chapter. Firstly, we will visualize data to get a basic idea about how closely related these data are and how much variance they represent. Next, we will describe the method used for selecting important characteristics. We will also explain how we generated our predictive models and we will reveal how we evaluated them.

4.1 Principal component analysis

Principal component analysis (PCA) is a method mostly used for visualization of high dimensional data. For matrix with r rows and c columns it creates $min(r-1, c)$ distinct principal components. Principal components are linearly uncorrelated variables created in such way that first principal component preserves most variance from the original dataset, with each subsequent principal component preserving less than previous one [36].

For principal component analysis we used python library scikit-learn [37]. Reduced representation of phages in form of a binary matrix was used as an input. First few principal components were used to create plots in python library matplotlib. In the Figure (4.1) we can see data visualized with the principal component one on the x-axis and the principal component two on the y-axis. Each data point represents one phage record and the color of the particular data point corresponds to genus of that phage. As we can see, most phage records are located around the center, with some distinct groups of Mycobacterium and Staphylococcus phages outside the center. Although, this could suggest difficulties with distinguishing different genera, it was not the case as first two principal components retained less than 21% of dataset variability. Therefore, we assumed binary representation of phages is reasonable and proceeded with a different method of analysis.

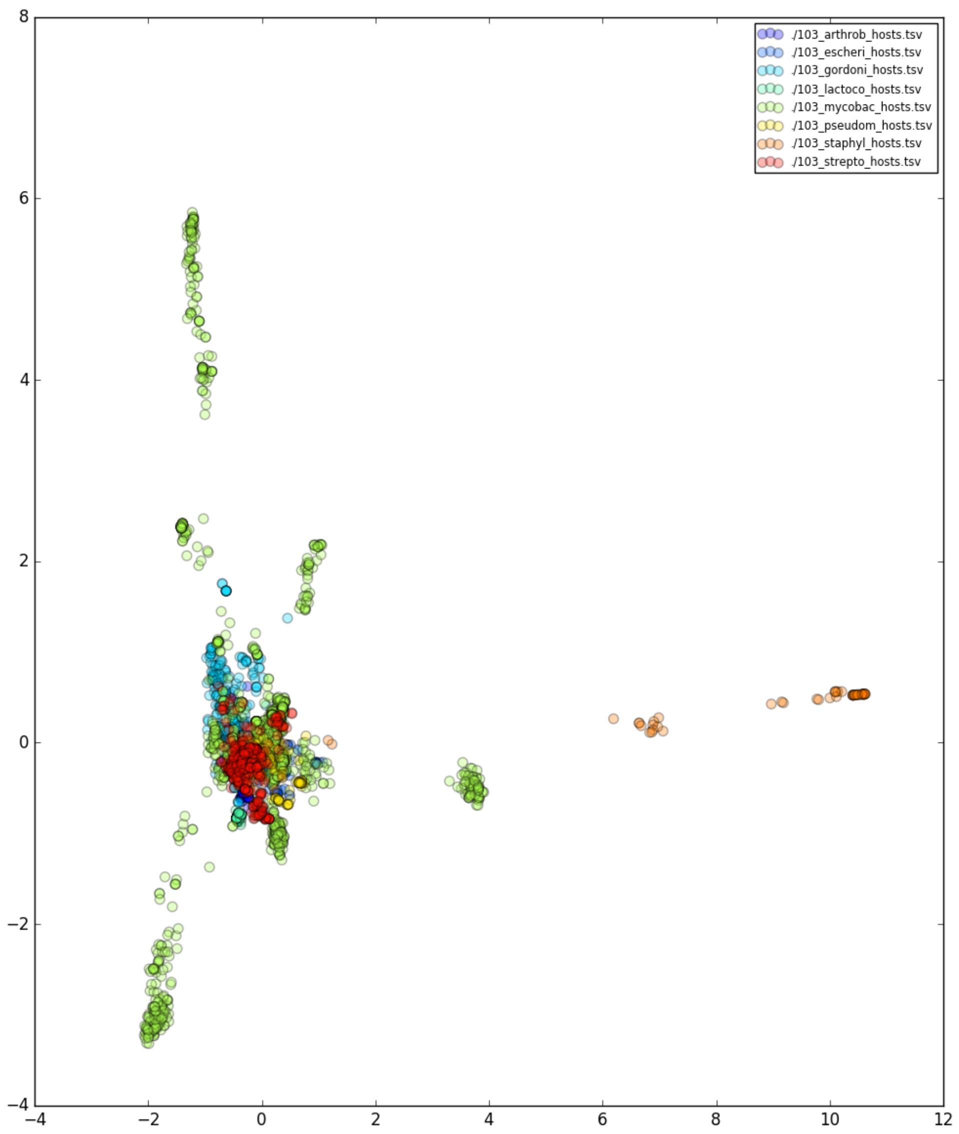


Figure 4.1: Principal component analysis - first two principal components, PC1 (11.57% of variability) on the x-axis, PC2 (9.19% of variability) on the y-axis

4.2 Decision tree

Decision tree is a model, which we can imagine as a binary tree. In this tree, every node is either decision node or the end node. Decision node contains condition and has two child nodes. First child node represents all cases, where the condition was not met and the second child node represents all cases, where condition was met. These child nodes can be decision nodes or end nodes. End nodes do not contain any conditions and they represent the final decision made by model.

Decision tree is the simplest predictive model and contains conditions exclusively. Usually, we need two sets of data to create the model, commonly called features and labels. Features represent data which are known before prediction and labels represent expected results of prediction. For example, the data representing features could be size, weight, length of nose and color and labels could symbolize if animal described by those features is a dog or it it is a cat.

Decision tree model has advantages, which led us to a conclusion that it will suit our needs. Namely, these are the options to easily visualize the model, interpret the results, availability in many programming libraries and its good documentation. The fact that decision tree model is a white box model allowed us to extract further information from our models, for example importance of particular clusters for predictions.

Of course, there are also disadvantages of this model. One of the biggest disadvantage is that the decision tree model is prone to overfitting of data. It means that sometimes the resulting model does not generalize the data well enough and the accuracy of real predictions can be lowered. Another disadvantage is, the models are often biased when predicting multiple classes with an unbalanced dataset. These issues was addressed with methods mentioned further.

4.3 Feature selection

Up to now, our training dataset consisted of a matrix with 2787 rows, representing phages and 15017 columns, representing gene clusters. This high dimensionality of our data could lead to the increased probability of overfitting of models on data. To address this concern, we decided to perform feature selection. Feature selection is a process of removing dimensions with low importance from the dataset. The reason to prefer feature selection over feature extraction methods as PCA presented in previous section was that we wanted our tree models to be representable in terms of important clusters rather than in terms of principal components. Because we expected a lot of clusters with small number of genes, our choice for feature selection method was the *Variance Threshold.* The Variance Threshold method is a simple method that removes columns with variance under certain threshold. With this technique we removed all columns in the matrix with ones in more than 99% of cases or with zeros in more than 99% of cases. The reduced matrix had 2787 rows and 1818 columns.

4.4 Building decision tree models

In our work we used a Decision Tree Classifier from python library scikit-learn. For each group of phages with hosts from eight selected genera we created one model. Each of those models was trained to answer a question, whether one particular phage was able to infect bacteria from a particular genus. For example, for a created model Mycobacterium we could ask if the given phage sequence is able to infect bacteria from Mycobacterium genus. The model will return one or zero, where one represents that model assumes that the given sequence belongs to a Mycobacterium phage and zero represents that model assumes it does not belong to Mycobacterium phage. Models were trained with reduced matrix used as features. The ability to infect particular genus was used as labels. To prevent overfitting of our trees, we also used the parameter `min_impurity_split=0.03`. This enabled a threshold for splitting leaves and therefore only nodes with an impurity index greater than 0.03 were divided. The threshold 0.03 was determined empirically. Lower values created a tree with many nodes, where the risk of overfitting was high and greater values did not have enough nodes to maintain model's accuracy. With this approach we created a model for each of our eight selected host genera and visualized it with a python library graphviz. We can see example of this visualization in the Figure 4.2.

We can see that cluster number 172 provides the best division of phages infecting the Arthrobacter genus. Most Arthrobacter phages have some gene from this cluster. This suggests genes from this cluster are of high importance for the Arthrobacter phages and could participate on the important mechanism for infecting bacteria from genus Arthrobacter. Of course, these claims need to be supported by additional evidence and biological experiments, but it can provide valuable insight into the data. We can analyze other graph nodes similarly. Finally, the models were stored using python library pickle for later use.

4.5 Classification of phage sequences

For classification we expected to have complete sequence of bacteriophage. This sequence was annotated using Prokka and genes were extracted using a custom script. Extracted genes were aligned using BLAST with a database of genes from the training set. We assigned a cluster number to all newly obtained genes based on the cluster number of the most similar gene from the BLAST database. Thereafter, vector of ones

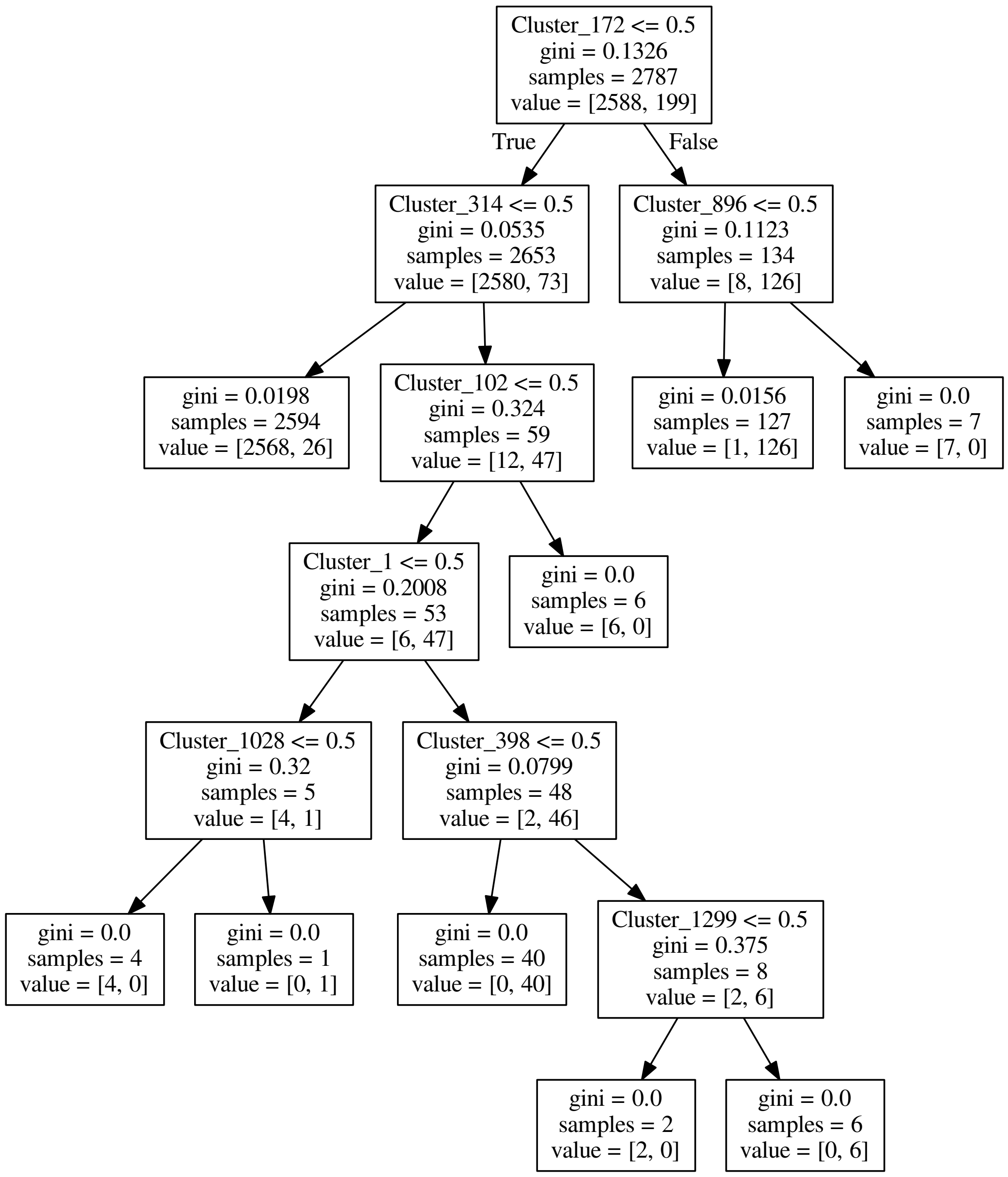


Figure 4.2: Arthrobacter model

model	correct	FP	FN	accuracy
Arthrobacter	683	11 (1.57%)	5 (0.71%)	97.71%
Escherichia	679	17 (2.43%)	3 (0.42%)	97.13%
Gordonia	647	39 (5.57%)	13 (1.85%)	92.56%
Lactococcus	680	3 (0.42%)	16 (2.28%)	97.28%
Mycobacterium	686	11 (1.57%)	2 0.28%	98.14%
Pseudomonas	686	6 (0.85%)	7 (1.00%)	98.14%
Staphylococcus	685	0 (0.00%)	14 (2.00%)	97.99%
Streptococcus	686	2 (0.28%)	11 (1.57%)	98.14%

Table 4.1: Evaluation of models

and zeros was created for each phage representing if the phage contained a gene from a particular cluster - similar to a new row in a matrix. This vector was passed to the decision tree model and resulting prediction was saved.

4.6 Evaluation of models

To examine accuracy of our models, we classified all bacteriophages from our test dataset. Test dataset contained 699 phage records. Resulting predictions were aggregated and number of correctly predicted $(TP + TN)$, false positive (FP) and false negative (FN) was recorded. The table was constructed from this data. *Accuracy* was calculated as $TP + TN/(TP + TN + FP + FN)$, *false positive* percentage as $FP/(TP+TN+FP+FN)$ and *false negative* percentage as $FN/(TP+TN+FP+FN)$. We can see summary of model statistics in the Table 4.1.

4.7 Limitations and future work

Regardless of the high accuracy of our predictions, we are aware of the potential improvement that might be possible in the future. Sequences used were mostly annotated only with one host. Despite high phage specificity, some phages can have multiple hosts. One of the goals in the future will be to collect more accurate dataset with experimentally confirmed non-hosts. Also, relatively small number of phages are known to mankind, despite of approximately 200 newly discovered phages appears in databases every month. With a greater and a more accurate dataset, we would be able to create models to the taxonomical levels of species or even strains. In the future, we would also like to try this method for predicting from incomplete sequences, as this will mostly be the case in practice. This could lead to novel methods for indirect diagnosis of a microbiome.

Conclusion

The importance of the bacteriophages as a research subject is rising mainly due to the decreasing effectiveness of antibiotic treatments. Bacteriophages could be used as novel weapons in the fight against bacteria. The goal of this work was to examine relationships between bacteriophage genomes and its bacterial host.

At first, we collected publicly available data, annotated phage genomes for genes and clustered these genes according to their similarity. Then, we proposed reduced representation of each bacteriophage based on genes, which were observed in its genome. We demonstrated feasibility of this representation, using principal component analysis. In representations that were transferred into lower dimensions we were able to roughly distinguish between clusters of bacteriophages with a distinct bacterial host. Finally, we used a decision tree classifier to refine our prediction. This machine learning method allowed us to create predictive models enabling us to assign a bacterial host with high accuracy using only information about the genome of the bacteriophage. Models were also interpretable in the terms of gene clusters. This allowed us to highlight clusters with high potential of being important for phage specificity.

The method demonstrated in this work could also be used for other biological traits, which depend on the genes included in phage genome. This could bring a new insight on phages and their functions.

Bibliography

[1] Bruce Alberts. *Molecular biology of the cell.* Garland science, 2017.

[2] Philippe Hupé. Central dogma of molecular biology, 2012.

[3] Forest Rohwer and Rob Edwards. The phage proteomic tree: a genome-based taxonomy for phage. *Journal of bacteriology*, 184(16):4529–4535, 2002.

[4] K Eric Wommack and Rita R Colwell. Virioplankton: viruses in aquatic ecosystems. *Microbiology and molecular biology reviews*, 64(1):69–114, 2000.

[5] Daniel Nelson. Phage taxonomy: we agree to disagree. *Journal of bacteriology*, 186(21):7029–7031, 2004.

[6] Ranjan V Mannige and Charles L Brooks III. Periodic table of virus capsids: implications for natural selection and design. *PLoS One*, 5(3):e9423, 2010.

[7] Adenosine. A typical t-even bacteriophage, 2009.

[8] G Bertani. Lysogenic versus lytic cycle of phage multiplication. In *Cold Spring Harbor symposia on quantitative biology*, volume 18, pages 65–70. Cold Spring Harbor Laboratory Press, 1953.

[9] Martha RJ Clokie, Andrew D Millard, Andrey V Letarov, and Shaun Heaphy. Phages in nature. *Bacteriophage*, 1(1):31–45, 2011.

[10] Britta Leverentz, William S Conway, Wojciech Janisiewicz, and Mary J Camp. Optimizing concentration and timing of a phage spray application to reduce listeria monocytogenes on honeydew melon tissue. *Journal of food protection*, 67(8):1682–1686, 2004.

[11] Mzia Kutateladze and Revaz Adamia. Bacteriophages as potential new therapeutics to replace or supplement antibiotics. *Trends in biotechnology*, 28(12):591–595, 2010.

[12] Julia Villarroel, Kortine Annina Kleinheinz, Vanessa Isabell Jurtz, Henrike Zschach, Ole Lund, Morten Nielsen, and Mette Voldby Larsen. Hostphinder: A phage host prediction tool. *Viruses*, 8(5), 2016.

[13] Koichiro Tamura and Masatoshi Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial dna in humans and chimpanzees. *Molecular biology and evolution*, 10(3):512–526, 1993.

[14] Hannes Röst. Blosum, 2008.

[15] Slowkow. Needleman-wunsch, 2008.

[16] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[17] A. J. Enright, S. Van Dongen, and C. A. Ouzounis. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, 2002.

[18] Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012.

[19] Dennis A Benson, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and David L Wheeler. Genbank. *Nucleic acids research*, 36(Database issue):D25, 2008.

[20] Donna Maglott, Jim Ostell, Kim D Pruitt, and Tatiana Tatusova. Entrez gene: gene-centered information at ncbi. *Nucleic acids research*, 39(suppl_1):D52–D57, 2010.

[21] Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

[22] Kim D Pruitt, Tatiana Tatusova, and Donna R Maglott. Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic acids research*, 35(suppl_1):D61–D65, 2006.

[23] Christine L Cosma, David R Sherman, and Lalita Ramakrishnan. The secret lives of the pathogenic mycobacteria. *Annual Reviews in Microbiology*, 57(1):641–676, 2003.

[24] Torsten Seemann. Prokka: rapid prokaryotic genome annotation. *Bioinformatics*, 30(14):2068–2069, 2014.

[25] Doug Hyatt, Gwo-Liang Chen, Philip F LoCascio, Miriam L Land, Frank W Larimer, and Loren J Hauser. Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC bioinformatics*, 11(1):119, 2010.

[26] UniProt Consortium. Uniprot: a hub for protein information. *Nucleic acids research*, 43(D1):D204–D212, 2014.

[27] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik LL Sonnhammer, et al. The pfam protein families database. *Nucleic acids research*, 32(suppl_1):D138–D141, 2004.

[28] Daniel H Haft, Brendan J Loftus, Delwood L Richardson, Fan Yang, Jonathan A Eisen, Ian T Paulsen, and Owen White. Tigrfams: a protein family resource for the functional identification of proteins. *Nucleic acids research*, 29(1):41–43, 2001.

[29] Ravi José Tristão Ramos, Allan Cézar de Azevedo Martins, Gabrielle da Silva Delgado, Crina-Maria Ionescu, Turán Peter Ürményi, Rosane Silva, and Jaroslav Koča. Crocoblast: Running blast efficiently in the age of next-generation sequencing. *Bioinformatics*, 33(22):3648–3651, 2017.

[30] Tamás Nepusz, Rajkumar Sasidharan, and Alberto Paccanaro. Scps: a fast implementation of a spectral method for detecting protein families on a genome-wide scale. *BMC bioinformatics*, 11(1):120, 2010.

[31] Evgeni M Zdobnov and Rolf Apweiler. Interproscan–an integration platform for the signature-recognition methods in interpro. *Bioinformatics*, 17(9):847–848, 2001.

[32] Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Edouard De Castro, Petra S Langendijk-Genevaux, Marco Pagni, and Christian JA Sigrist. The prosite database. *Nucleic acids research*, 34(suppl_1):D227–D230, 2006.

[33] Terri K Attwood. The prints database: a resource for identification of protein families. *Briefings in bioinformatics*, 3(3):252–263, 2002.

[34] Florence Corpet, Jérôme Gouzy, and Daniel Kahn. The prodom database of protein domain families. *Nucleic Acids Research*, 26(1):323–326, 1998.

[35] Jörg Schultz, Frank Milpetz, Peer Bork, and Chris P Ponting. Smart, a simple modular architecture research tool: identification of signaling domains. *Proceedings of the National Academy of Sciences*, 95(11):5857–5864, 1998.

[36] Alvin C Rencher. Principal component analysis. *Methods of Multivariate Analysis, Second Edition*, pages 380–407, 2002.

[37] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[38] Bruce Alberts. *Molecular biology of the cell.* Garland science, 2017.

[39] TF Smith and WA BEYER. M. s. waterman. 1976.

[40] Marketa J. Zvelebil and Jeremy O. Baum. *Understanding bioinformatics.* Garland Science, 2008.

[41] EMBOSS Group et al. The european molecular biology open software suite.

[42] NCBI Resource Coordinators. Database resources of the national center for biotechnology information. *Nucleic acids research*, 44(Database issue):D7, 2016.

[43] Chantal Hulo, Edouard De Castro, Patrick Masson, Lydie Bougueleret, Amos Bairoch, Ioannis Xenarios, and Philippe Le Mercier. Viralzone: a knowledge resource to understand virus diversity. *Nucleic acids research*, 39(suppl_1):D576–D582, 2010.

[44] Daniel A. Russell and Graham F. Hatfull. Phagesdb: the actinobacteriophage database. *Bioinformatics*, 33(5):784–786, 2017.