Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

# An end-to-end pipeline for detection of endolysin proteins from raw sequencing reads

Bachelor Thesis

2022
Juraj Vašut

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

# An end-to-end pipeline for detection of endolysin proteins from raw sequencing reads

Bachelor Thesis

| | |
|---|---|
| Study Programme: | Bioinformatics |
| Field of Study: | Computer Science and Biology |
| Department: | Department of Applied Informatics |
| Supervisor: | MSc. Andrej Baláž |

Bratislava, 2022
Juraj Vašut


Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

# ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Juraj Vašut
**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:** An end-to-end pipeline for detection of endolysin proteins from raw sequencing reads
*Program na detekciu endolyzínov z nespracovaných sekvenačných čítaní*

**Anotácia:** Bakteriofágy sú vírusy špecializujúce sa na infikovanie baktérií. Ich životný cyklus sa skladá z lyzogénnej a lytickej fázy. Na konci lytickej fázy rozkladajú bunkovú stenu hostiteľa, čím zabíjajú baktérie. Proteín, ktorý používajú na rozkladanie bakteriálnej bunkovej steny, sa nazýva endolyzín.
Endolyzíny majú potenciál v liečbe vážnych bakteriálnych infekcií, ale ich identifikácia vyžaduje veľa experimentov v laboratóriu. Potreba bioinformatického programu, schopného zredukovať priestor možností a tým aj počet experimentov, je signifikantná.
Cieľom práce bude vytvoriť bioinformatický program, ktorý napomôže objaveniu nových endolyzínov. Vstup pre program bude vo forme čítaní zo sekvenačného experimentu. Program poskladá čítania do kontigov, identifikuje vírusové sekvencie, odpredikuje pozíciu génov, identifikuje ich funkciu a vyhodnotí výsledky. Výstupom bude množina génov, ktoré sú s vysokou pravdepodobnosťou endolyzíny.
Úlohou študenta bude naštudovať bioinformatické nástroje pre jednotlivé kroky programu a navrhnúť, ktoré z nich budú integrované do programu. Následne študent vytvorí ľahko inštalovateľný program na detekciu endolyzínov a porovná vytvorený program s inými dostupnými riešeniami.

**Vedúci:** MSc. Andrej Baláž
**Katedra:** FMFI.KAI - Katedra aplikovanej informatiky
**Vedúci katedry:** prof. Ing. Igor Farkaš, Dr.

**Dátum zadania:** 15.10.2021

**Dátum schválenia:** 22.10.2021 doc. Mgr. Bronislava Brejová, PhD.
garant študijného programu

.....
študent

.....
vedúci práce


Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

## THESIS ASSIGNMENT

| | |
|---|---|
| **Name and Surname:** | Juraj Vašut |
| **Study programme:** | Bioinformatics (Joint degree study, bachelor I. deg., full time form) |
| **Field of Study:** | Computer Science<br>Biology |
| **Type of Thesis:** | Bachelor´s thesis |
| **Language of Thesis:** | English |
| **Secondary language:** | Slovak |

**Title:** An end-to-end pipeline for detection of endolysin proteins from raw sequencing reads

**Annotation:** Bacteriophages are viruses specialized in infecting bacteria. Their life-cycle consists of a lysogenic and a lytic phase. At the end of the lytic phase, they disintegrate the host's cell wall, effectively killing bacteria. The protein they use to disintegrate the bacteria is called endolysin.
These endolysins have the potential to be used in the treatment of serious bacterial infections, but their identification requires a lot of experiments in the laboratory. The need for an end-to-end bioinformatics pipeline, capable of reducing the search space and therefore the number of experiments, is significant.
This work aims to develop the bioinformatics pipeline, which can assist the discovery of novel endolysins. The input for the pipeline will be in the form of reads from the sequencing experiment. The pipeline will consist of assembling the reads into contigs, identifying the viral sequences, predicting the location of the genes, identifying their function and evaluating the results. The output will be a set of genes with a high likelihood of being endolysins.
The student's tasks will be to study multiple bioinformatics tools for each step of the pipeline and to propose a selection of tools, which will be integrated into the pipeline. The student will create an automatic end-to-end pipeline that can be easily installed and used to detect endolysins and compare the pipeline with existing solutions.

| | |
|---|---|
| **Supervisor:** | Master of Science Andrej Baláž |
| **Department:** | FMFI.KAI - Department of Applied Informatics |
| **Head of department:** | prof. Ing. Igor Farkaš, Dr. |
| **Assigned:** | 15.10.2021 |
| **Approved:** | 22.10.2021 |

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

.....
Student

.....
Supervisor

**Acknowledgments:** I would like to thank my supervisor MSc. Andrej Baláž for his helpful advice and guidance during the work on this thesis.

# Abstrakt

Nadmerné používanie antibiotík viedlo k vývoju multirezistentných baktérií. Tieto baktérie sú odolné voči veľkému množstvu antibiotík, čo sťažuje liečbu infekcií nimi spôsobenými. Z tohto dôvodu sa vyvíjajú ďalšie metódy liečby bakteriálnych infekcií. Jednou z metód je fágová terapia zahŕňajúca použitie bakteriofágových endolyzínov na zacielenie na konkrétne baktérie. Kvôli nedostatku nástrojov navrhnutých na analýzu vírusových genómov si objavovanie nových endolyzínov vyžaduje použitie zložitých zreťazených spracovávaní s veľkým počtom krokov. Aby sme uľahčili objavenie nových endolyzínov, predstavujeme jednoduchý nástroj Phendol schopný presne predpovedať endolyzíny v nespracovaných párovaných sekvenčných čítaniach.

**Kľúčové slová:** endolyzín, anotácia génov, baktériofág, skladanie kontigov, zreťazené spracovanie

# Abstract

The overuse of antibiotics has led to the evolution of multiresistant bacteria. These bacteria are resistant to a large number of antibiotics, making the infections caused by them more difficult to cure. For this reason, other methods to cure bacterial infections are being developed. One of the methods is phage therapy involving the use of bacteriophage endolysins to target specific bacteria. Due to the lack of tools designed for the analysis of viral genomes, the discovery of new endolysins requires the use of complex pipelines with a large number of steps involved. To facilitate an easier discovery of new endolysins, we introduce a simple end-to-end pipeline Phendol capable of accurate prediction of endolysins in raw paired-end sequencing reads.

**Keywords:** endolysin, gene annotation, bacteriophage, contig assembly, pipeline

# Contents

**Introduction** **1**

**1 Biological background** **3**

1.1 Bacteriophages vs. Antibiotics . . . . . 3

1.2 Life cycle of a phage . . . . . 4

1.3 Endolysins . . . . . 6

1.4 Data preparation . . . . . 7

1.4.1 Propagation and extraction . . . . . 7

1.4.2 Sequencing . . . . . 8

**2 Available tools** **11**

2.1 multiPhATE . . . . . 11

2.2 RASTtk . . . . . 12

**3 Pipeline** **15**

3.1 Seqtk . . . . . 15

3.2 SPAdes . . . . . 15

3.2.1 Terminology . . . . . 16

3.2.2 Algorithm . . . . . 16

3.3 Phanotate . . . . . 19

3.3.1 Algorithm . . . . . 20

3.4 BEDTools . . . . . 22

3.5 Transeq . . . . . 22

3.6 Blast . . . . . 23

3.6.1 Algorithm . . . . . 23

3.7 Blast2out . . . . . 25

3.8 Snakemake . . . . . 26

3.9 Phendol . . . . . 27

3.10 Conda . . . . . 28

3.11 Summary . . . . . 28

**4 Testing** **31**
4.1 Source of data . . . . . 31
4.2 Settings . . . . . 31
4.2.1 Phendol . . . . . 32
4.2.2 multiPhATE . . . . . 32
4.2.3 RASTtk . . . . . 32
4.3 Comparison . . . . . 33

**Conclusion** **35**

**Appendix A** **41**

# List of Figures


1.1 Structure of a bacteriophage. . . . . 3
1.2 Lytic cycle. . . . . 5
1.3 Lysogenic cycle. . . . . 5
1.4 Library preparation (a) and sequencing (b) using Illumina Miseq[15]. . 8

3.1 Potential errors in the graph: bulge (A), tip (B), chimeric read (C), repeat (D) [3]. . . . . 17
3.2 Steps the pipeline takes while annotating. . . . . 28

4.1 Number of predicted endolysins per sample by analysis method. . . . . 34

# Introduction

The overuse of antibiotics leads to the evolution of multiresistant bacteria immune to a larger amount of antibiotics [23]. This is facilitated by the ability of bacteria to quickly adapt to their environment. In presence of antibiotics, the surviving bacteria adapt to unfavourable conditions by gaining resistance, making antibiotics less effective. Since antibiotics are the main remedy in combating bacterial infections, infections caused by multiresistant bacteria are difficult to treat, making them dangerous and possibly deadly [20]. To prevent such bacteria from potentially gaining resistance to every available antibiotic, several types of antibiotics are kept outside of regular use to act as the last resort against highly resistant bacteria.

Since the last resort antibiotics only act as a buffer, making them a temporary solution, and the discovery of new antibiotics is difficult and time-consuming, research is conducted to develop new methods to treat bacterial infections. One of the methods developed is phage therapy [30]. This method is based on the ability of a bacteriophage to infect and kill bacteria as a byproduct of the reproduction of the bacteriophage during its lifecycle [8]. The protein used to disintegrate the cellular membranes of the bacteria is called endolysin. Since the disintegration of cellular membranes is sufficient to kill the bacteria, the application of endolysins to bacteria consequently causes the death of the bacteria.

Despite the need for the discovery of endolysins, the lack of interest in the analysis of viral genomes in the past led to a low number of tools designed for this purpose. The lack of tools designed for virus analysis forced researchers to work using tools not equipped to work with viral genomes properly, generally designed for analysis of bacterial genomes, leading to the creation of complex pipelines. Presently, more tools designed for the analysis of viral genomes are available. However, many pipelines are yet to adopt these tools into their workflow. Out of the existing pipelines, none are designed to directly output predicted endolysins.

To accommodate for this lack of pipeline, the goal of our thesis is to introduce an end-to-end pipeline designed to detect endolysin proteins from raw sequencing reads. The pipeline assembles raw paired-end reads from sequencing equipment into contigs, locates sequences coding viral genes, and predicts, which genes have a high likelihood of being endolysins. In the creation of the pipeline, we prioritize simplicity and ac-

cessibility to allow usage of the tool for researchers not accustomed to working with bioinformatics tools.

In the first chapter of the thesis, we explain the advantages of phage therapy compared to antibiotics. We describe the life cycle of a bacteriophage and the role of endolysins in it. Next, we describe the structure of endolysins and the mechanism behind their behaviour. We also outline common procedures used to acquire samples for analysis.

In the second chapter, we introduce pipelines addressing a similar issue as the one outlined in this work. We explain the differences between them and compare them to our solution.

In the third chapter, we describe our pipeline. We describe every tool we use in detail and clarify the algorithms operating them. In this chapter, we also convey how we use each tool and how we connect them to form a single pipeline.

In the fourth chapter, we evaluate our pipeline. We explain the method used to obtain data for testing, and parameters used in testing and compare results from our pipeline with results from tools described in the second chapter.

# Chapter 1

# Biological background

In this chapter, we explain the advantages of phage therapy compared to antibiotics. We introduce biological processes happening when bacteriophage infects bacteria and the lytic properties of endolysins. We also outline chemical processes used for the extraction and sequencing of bacteriophage DNA.

## 1.1 Bacteriophages vs. Antibiotics

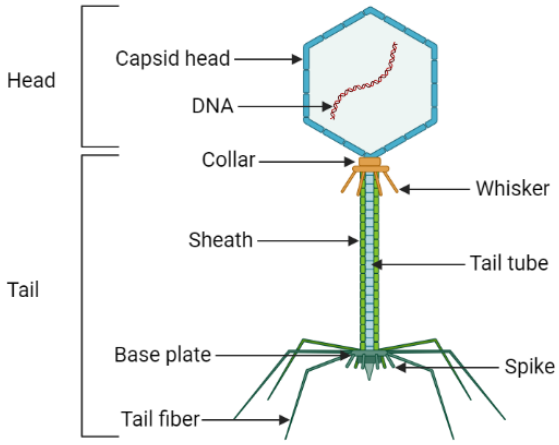


Figure 1.1: Structure of a bacteriophage.

Bacteriophages or phages are a type of virus which evolved specifically to be able to infect bacteria. They are composed of a molecule of nucleic acid encased in a protein structure (Figure 1.1). While there are thousands of varieties of phages, each phage usually infects only one type or a few types of bacteria [11]. This characteristic is utilized in phage therapy, which uses phages as an alternative to treatment using antibiotics.

There are two main advantages of phage therapy [18] when compared to antibiotics treatment. Antibiotics eliminate bacteria regardless of whether the bacteria are harmful, beneficial or do not affect the body. This in turn leads to damage of gut microbiota, which can create a change in bacterial metabolites, disrupt bacterial signalling and antimicrobial peptide secretion or damage regulation of the function of gut immune cells [32]. Unlike antibiotics, phage therapy targets only a specific type of bacteria allowing the body to maintain a healthy microbial environment [18].

Another advantage is that, unlike antibiotics, bacteriophages are alive and as such are subject to evolution. Because bacteria are evolving, they can gain resistance to antibiotics to which they are exposed. When a large variety of bacteria gain resistance to an antibiotic, the antibiotic is rendered ineffective. Similarly, when a bacteria gains resistance to multiple antibiotics, the treatment becomes even more difficult. Presently, many multiple antibiotics resistant bacteria strains already exist, like some strains of Staphylococcus aureus or Mycobacterium tuberculosis [10]. As a result, it becomes necessary to develop a new antibiotic. However, this demands arduous research as well as substantial financial support.

In contrast, when a bacteria develops resistance to a particular strain of bacteriophage, the bacteriophage as a result of its evolution develops another method of infecting that bacteria, bypassing the resistance to the virus. Since this process happens naturally, uncovering new phages capable of infecting bacteria is significantly less demanding. It also implies an inexhaustible supply of treatment for bacterial infections, which is becoming increasingly more important with bacteria gradually developing resistance to an increasing number of antibiotics.

## 1.2 Life cycle of a phage

For the phage to infect a bacteria, its tail fibres bind to specific receptors on the surface of the bacteria. While tail fibres and receptor pairing are highly specific, different types of phages might use the same receptors on membrane [11]. The phage then creates a puncture in the bacterial membrane. Next, the nucleic acid is expelled from the phage through its tail and injected into the bacteria. When in the cytoplasm, the viral genome in some cases becomes circular and resembles a plasmid. After entering the bacteria, nucleic acid enters one of two cycles: lytic or lysogenic.

In the lytic cycle (Figure 1.2), the viral nucleic acid is transcribed into messenger RNA (mRNA). If the viral genome consists of DNA, it is directly transcribed into mRNA. In case when it consists of RNA, it is first transcribed using an enzyme, reverse transcriptase, into DNA and then transcribed into mRNA. This mRNA utilizes cellular mechanisms of the host to destroy the nucleic acid of the host [11]. After the destruction

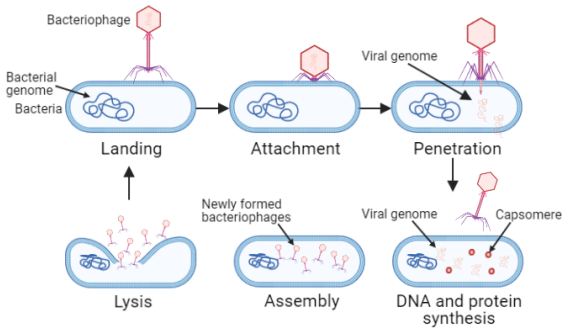


Figure 1.2: Lytic cycle.

of the host nucleic acid, the viral genome is replicated and transcribed to produce proteins required for the assembly of new viruses. After enough proteins and the viral genome is produced, they are assembled to create new bacteriophages. Next, the phage produces the enzyme, endolysin, which causes the lysis of the cellular membrane. By destroying the membrane, newly formed bacteriophages are released, ending the lytic cycle.

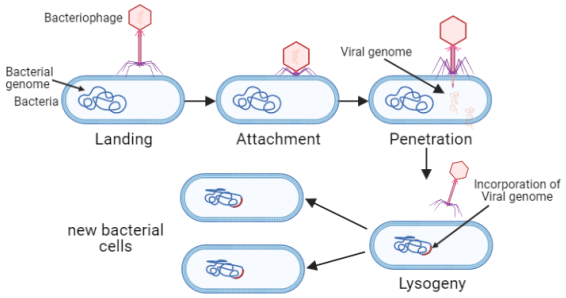


Figure 1.3: Lysogenic cycle.

The lysogenic cycle (Figure 1.3) differs from the lytic cycle by not immediately destroying the nucleic acid of the bacteria. Instead, it integrates its nucleic acid into the host genome, creating prophage. This is accomplished either by site-specific recombination or by random transposition [11]. After integrating into the host genome, the prophage remains in a dormant state. The cellular mechanism of the bacteria remains unaffected by the prophage, so the bacteria continues its regular functions without alteration. During cell division, the prophage replicates with the host chromosomes resulting in the new bacterial cells already being infected by the phage. This process of replication is repeated until the conditions of the environment deteriorate. The deterioration can be induced by physical factors like UV radiation, low nutrient concentration or chemical factors. When the conditions deteriorate, the prophage might switch from lysogenic to the lytic cycle.

The lysogenic cycle of the phage has the advantage of increasing the number of bacteriophages created from a single specimen. Since the phage is replicated during cell division along with the host cell, the number of proteins required to infect the same number of hosts is halved with each division, meaning the phage utilizing the lysogenic cycle can reproduce in worse conditions than a phage only utilizing the lytic cycle.

## 1.3 Endolysins

Because the purpose of phage therapy is the treatment of bacterial infection, the part of the life cycle of a bacteriophage of most interest is the production of endolysin. Endolysins, alternatively termed phage lysins, are peptidoglycan hydrolases used by bacteriophages to enzymatically degrade the cellular membrane of the host bacteria, resulting in osmotic imbalance leading to rapid lysis of the membrane [26]. The structure of lysins is in big part affected by whether the targeted bacteria are Gram-positive or Gram-negative, as the cellular membranes of these groups have different structures.

Endolysins of bacteriophages targeting Gram-positive bacteria have evolved in a way, where catalytic activity and substrate recognition are separated into two distinct varieties of functional domains, enzymatically active domains and cell wall binding domains [26]. Enzymatically active domains impart the catalytic mechanism of lysin, the mechanism for cleaving specific bonds in a cellular membrane of bacteria. The cell wall binding domain is responsible for targeting the lysin to its substrate and keeping it bound to parts of the cell wall after cell lysis, reducing the probability of lysis of surrounding cells not yet infected by the phage.

On the contrary, endolysins of phages targeting Gram-negative bacteria tend to be small single-domain globular proteins without a specific domain for binding to the cell wall [26]. Unlike endolysins from Gram-positive background, damage to surrounding bacteria in the case of Gram-negative bacteria is prevented by its characteristic outer membrane, which protects the cell wall from the outside environment. Lysin is encased in the outer membrane, resulting in the prevention of damage to other bacteria. It is surmised, that these endolysins fulfil their catalytic role more effectively as opposed to endolysins for Gram-positive bacteria [26], which are bound to one site on the membrane and as such reduced effectivity.

To further increase the diversity of possible endolysin architectures, the structure of endolysins from Gram-positive background can have more than two domains. Most prominent structures include two N-terminal enzymatically active domains and one C-terminal cell wall binding domain, central cell wall binding domains separating two terminal enzymatically active domains, among others [26]. Almost all currently described Gram-positive endolysins are encoded by a single gene, simplifying their localization

in the phage genome.

Enzymatically active domains encompass the ability of an endolysin to catalyze a breakdown of the cell membrane. Based on the bond of the cell membrane the endolysin attacks, endolysins can be classified into five different groups: N-acetyl-β-D-muramidases (lysozymes) and lytic transglycosylases that cleave one of the glycosidic bonds of a sugar strand, N-acetyl-β-D-glucosaminidases cutting another glycosidic bond in the sugar strand, N-acetylmuramoyl-L-alanine amidases hydrolysing amide bond between sugar and peptide parts and endopeptidases cleaving the peptides making up interconnecting stem portion of the membrane [26]. Any of these methods lead to destabilization and breakdown of the cell membrane.

Cell wall binding domains allow an endolysin to recognise and bind (not using a covalent bond) to ligands within the cell membrane or other molecules associated with the cell wall. This significantly reduces the range of activity for the enzymatically active domains. The spectrum of the cell wall binding domains can range from encompassing an entire genus of bacteria (lysostaphin domain targeting SH3b-like cell wall common to staphylococcal strains), making it generally broader than the host range of the particular phage, to the specificity of a single strain (endolysins of Listeria phage binding to groups of Listeria containing very specific ligands) [26].

## 1.4 Data preparation

To gain usable information about bacteriophages, it is necessary to extract their DNA and be able to analyse it. As a result of the phage DNA being relatively small, the phage needs to be replicated to provide a viable sample. The DNA then needs to be extracted from the phage. Once the phage DNA is extracted, it can be sequenced to provide data for bioinformatics analysis.

### 1.4.1 Propagation and extraction

The propagation and extraction of phage DNA are dependent on the species of phage in question [14]. In general, however, the process is carried out by following similar steps. Since phages do not have their own replication apparatus, in order to acquire a sufficient amount of DNA for sequencing, it is necessary to utilize the replication apparatus of bacteria. For this purpose, bacterial hosts are first grown independently in an advantageous environment.

Once the hosts are sufficiently grown, a solution containing phages is mixed in with the host culture. The resulting culture is then incubated to facilitate phage replication. In the process, the lysis of the host is achieved, releasing the replicated phages into the environment. After a period of time, the culture is transferred to a centrifuge and

spun. The bacterial pellet settled at the bottom of the substrate is discarded while the remaining supernatant is kept. The resulting substrate is then treated with RNase and DNase. After, a solution of NaCl and PEG (poly(ethylene glycol)) is added to the culture and incubated. As a result, the phage particles then precipitate.

The resulting substrate is then centrifuged, creating a phage pellet from the precipitated particles. The pellet is extracted from the substrate and resuspended in a buffer. To remove phage protein capsid, phenol is added and the substrate is incubated. After the protein dissolves, the aqueous layer is removed. Ethanol is then added to the DNA and the substrate is centrifuged. Finally, extracted DNA is concentrated.

### 1.4.2 Sequencing

After the extraction of phage DNA is completed, the DNA needs to be sequenced in order to be analysed by our tool. There are many ways to sequence DNA. Since our tool is specialized to work with reads from Illumina sequencers, the process of sequencing described is focused on Illumina MiSeq [25].

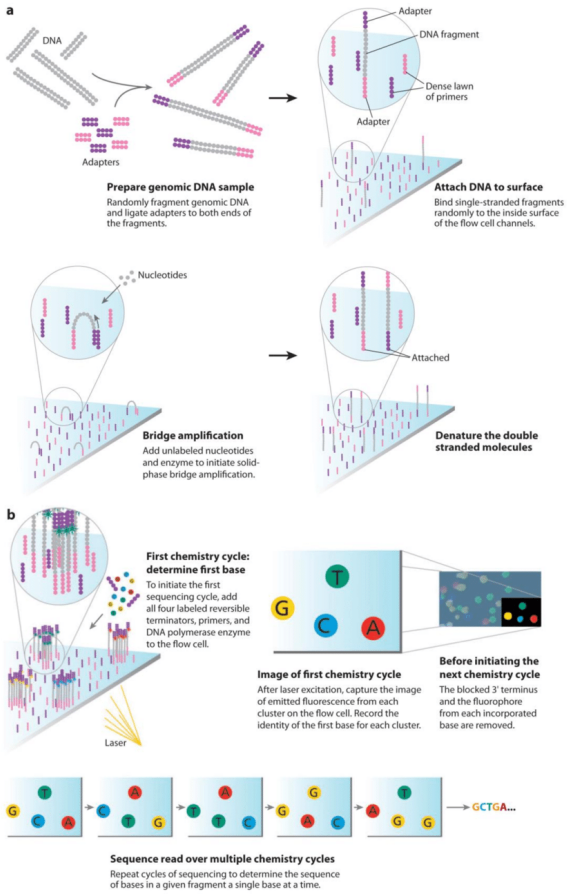


Figure 1.4: Library preparation (a) and sequencing (b) using Illumina Miseq[15].

Before starting the sequencing, the sequencer requires a DNA library (Figure 1.4a).

To create a DNA library, DNA is first fragmented into small parts (500-1000 bp). After the addition of special barcoding sequences to the fragments, short oligonucleotides (adaptors) are bound to the fragments. The adaptors are complementary to primer sequences on a glass disc. The fragment is attached to the glass disc by a primer on one end, while the other is held close to another primer using its adaptor. After the attachment to the glass disc, a new strand of DNA complementary to the attached strand is synthesized. This new strand is attached to the primer the first strand is held close to by its adaptor. After the synthesis, the two strands separate, and the bond between the adaptor and the primer is released. This form of replication is repeated many times, creating thousands of copies of the fragment in close proximity forming a cluster. This happens on the glass disc for every fragment of DNA, creating a DNA library.

With the library created, the sequencing can proceed (Figure 1.4b). During sequencing, a substrate called "mastermix" is used. Mastermix contains primers, DNA-polymerase and 4 different types of marked fluorescent nucleotides with a sequence inhibiting polymerization (terminators) on its 3' -end. The nucleotides are bound to the fragments of DNA based on complementarity. Next, the excess mastermix is removed from the disc. The sequencer then measures the fluorescence of the bound nucleotides to determine their identity. After the measurement, the terminators split from the sequence allowing another nucleotide to connect. This process is cyclically repeated until the entire library is read. The result is a set of sequences of DNA with adaptor sequences. These sequences are then processed by bioinformatic pipelines.

# Chapter 2

# Available tools

The discovery of antibiotics reduced interest in phage research, leading to a lack of tools designed for phage annotation. When the analysis of phage DNA is required, many companies either execute the analysis manually by using individual bioinformatics tools or use a tool not designed for phage annotation. In this chapter, we introduce some tools used not designed for phage annotation as well as some recent tools purpose-built for this task. We also explain the main differences between our approach and the tools mentioned.

## 2.1 multiPhATE

One of the most recent tools for phage annotation is multiPhATE [6]. It is a high-throughput pipeline driver invoking the PhATE annotation pipeline, allowing the annotation of a specified set of phage genomes. PhATE uses up to four gene callers: GeneMarkS, Glimmer, Prodigal and Phanotate with Phanotate being developed most recently and more optimized for use with phage genomes [21]. As input, multiPhATE uses a configuration file with a list of genomes for PhATE and a set of parameters controlling software execution. The user specifies the names of files in fasta format containing phage genome, output directories, and other data required for genome analysis. The user can also specify some optional analyses.

For each genome, PhATE begins by using gene callers previously specified by a user to perform gene calling. In case of multiple gene callers being used, PhATEs output is a table containing side-by-side comparisons of the gene calls as well as numbers and lengths of gene calls for each algorithm. It also includes several common and unique calls to each algorithm.

After the gene calling, PhATE uses blastn (nucleotide databases search), blastp (protein databases search) and jackhmmer to identify similarities to the phage genome and predict its gene and peptide sequences using multiple databases: National Center

of Biological Information (NCBI) virus genomes, Refseq proteins, refseq genes, virus proteins and Non-Redundant protein sequence database, Swissprot, Phage Annotation Tools and Methods, Kyoto Encyclopedia of Genes and Genomes and a fasta sequence dataset derived from the database of phage Virus Orthologous Groups (pVOG).

The output of PhATE includes: output from gene call algorithms, gene and translated peptide files in fasta format, combined-annotation summary files, raw Blast and HMM outputs, fasta files containing predicted peptides and the members of identified pVOG families where the peptide may be assigned. Using multiPhATE results in all genomes being annotated.

Our pipeline in large part emulates this behaviour, however, there are some differences. While multiPhATE uses phage genomes as an input, our tool uses raw reads from the sequencer, making it more specific to the data assemblies are from. Unlike multiPhATE, our tool does not require the entire genome for its operation. We have also prioritized locating possible endolysin sequences as opposed to annotating the entire genome. This means that while results from multiPhATE need to be further analysed to find phage endolysins, our tool outputs endolysins directly. To do this, our tool uses a different database. Our database used in Blast is custom made and only includes known endolysins. Almost all differences between our tool and multiPhATE are caused by our desire to be able to input raw reads from the sequencer and receive a direct output containing endolysins. In simpler terms, we prioritize simplicity of execution to the possibly more versatile tool.

## 2.2 RASTtk

Another tool used for phage annotation is the RAST tool kit (RASTtk) [4]. It is a modular version of the annotation tool RAST designed to allow the creation of custom annotation pipelines. Even though RAST itself is designed to work with bacterial and archaeal genomes, using the tool kit gives users the opportunity to create pipelines capable of annotating phage genomes using scripts featured in it. During annotation, RASTtk uses the specified scripts to produce data, which is then collected to form the whole genome. While the workflow of custom pipelines is highly specific, There exists an abstract layout that the RASTtk pipeline follows.

The pipeline begins by transforming a set of contigs into a file format called Genome Typed Object (GTO). The transformed file is formatted as a human-readable JSON file. To do this, RASTtk uses the script "rast-create-genome." Following this transformation, each step takes an input GTO and enhances it using another script. This creates a new and enhanced GTO file. The set of scripts used to enhance the information in the GTO file is defined by the user.

In the case of phage annotation, one of the useful features is rast-call-features-prophage-phispy, which can be used to find the section of contigs with phage DNA. This is done by using a tool PhiSpy designed to find prophages in bacterial genomes by combining similarity- and composition-based strategies. Other useful scripts are rast-call-features-CDS-genemark/-glimmer3/-prodigal that use gene callers GeneMarkS, Glimmer or Prodigal to find coding sequences in the contigs. The user can also use different methods to annotate proteins.

After every specified script enhanced the input, the resulting GTO file can be exported using rast-export-genome. This script can return the results in different formats, including FASTA, Genbank, and feature table. [4]. The output includes all input sequences with features annotated by selected scripts. For phage annotation, these results need to be filtered afterwards to only include sequences containing phage DNA.

Since RASTtk, similarly to default RAST, is designed to work with bacterial and archaeal genomes, many of its scripts are designed to deal with the annotation of those types of genomes. For example, every gene caller available in RASTtk is designed to look for genes in bacterial genome [4]. These gene callers are still able to find phage genes since phages generally use the same start and stop codons. However, they are largely incapable of discovering overlapping genes, which can appear in a viral genome due to its reduction. For this purpose, Phanotate used by our tool is better, since it is designed to work specifically with the phage genome.

As mentioned with multiPhATE, RASTtk is not designed to work with raw reads from a sequencer, allowing it to receive input from a larger variety of sequencers at a cost of increased complexity. Since the resulting output from RASTtk contains all sequences, not just the requested ones, the output needs to be further modified by removing sequences not fulfilling the search criteria. This again increases the complexity of the annotation process.

Both tools share a common issue in not having a conda package. Instead, both tools need to be manually downloaded and installed. Additionally, every dependency needs to be installed separately. This results in the installation of the tools itself being very complex and tedious. Our pipeline does not possess this problem and can be installed with all of its dependencies by using a single command, making it much simpler.

# Chapter 3

# Pipeline

Our pipeline uses a combination of various tools to modify input from Illumina sequenced paired-end reads. In this chapter, we explain used tools and their integration into the pipeline.

## 3.1 Seqtk

Seqtk[17] is a tool capable of fast processing of sequences in FASTA or FASTQ format. In our pipeline, we use the ability of seqtk to extract a subsample of reads from an input file. By using the same random seed for two paired FASTQ files, we get a fixed set of paired reads with the requested number of sequences. The subsampling is especially useful when working with large files because due to the nature of SPAdes, which requires notably larger available memory than the size of the input files. If the client does not possess the required amount of memory, by using random subsampling, the tool is still able to assemble the sample into contigs.

Lowering the number of reads results in shorter runtime of SPAdes at the cost of accuracy. At this stage our tool focuses on accuracy, so the number is set to the highest value. The value is adjustable for added flexibility allowing our tool to operate with lower memory demands.

## 3.2 SPAdes

SPAdes[3] si a short read assembler designed for the assembly of small single-cell and multi-cell bacterial reads. While assemblies of viral DNA are not its speciality, it has been proven that SPAdes produces consistently accurate results even when compared to virus-oriented assemblers [29]. SPAdes works on reads from Illumina and IonTorrent. It can also create hybrid assemblies using PacBio, Oxford Nanopore and Sanger reads. The reads from Illumina can be either paired-end, mate-pairs or single reads. The

method used for assembly is based on using k-mers, subsequences with a length of k from reads, to create a de Bruijn graph on which further theoretical operations are executed. The assembler also performs an error adjustment, increasing the reliability of produced contigs.

### 3.2.1 Terminology

To understand the algorithm SPAdes uses, several terms need to be explained. Hub is a vertice of a directed graph with the number of edges leading to it is different from one. The number of edges leaving it is different from one as well. When two hubs are connected by a path of non-hub vertices, the path is called a hub-path (h-path). Each edge in the graph belongs to a unique h-path. For every h-path, its first edge is called a hub-edge (h-edge).

### 3.2.2 Algorithm

The algorithm of SPAdes can be simplified into four stages: assembly graph construction, k-bimer adjustment, paired assembly graph construction and contig construction [3].

The first stage begins with the construction of a multisized de Bruijn graph. This graph is created from k-mers by creating vertices with the first labelled by the prefix of the k-mer and the second by its suffix. These vertices are connected by an edge representing the k-mer. Merging vertices with the same labels results in the creation of a de Bruijn graph. To make this graph multisized, a different value of k is used based on the coverage of a region. In regions with lower coverage, the value of k used is lower. Conversely, in high-coverage regions, the value used is higher.

With the graph created, SPAdes locates and corrects errors in the graph caused by errors in reads. To discover which h-paths in the graph are correct, it implements an improved gradual h-path removal strategy. One of the improvements lies in iterating through h-paths and updating the list of h-paths as soon as one is removed. It also at some points runs only bulge corremovals, which are considered safer than other removals, because they maintain information on removed h-paths. Lastly, it restricts the removal of h-paths to only those, which start with a hub with at least two outgoing edges and end with a hub with at least two incoming edges. For tips, this restriction only applies to the hub which has both incoming and outgoing edges. Every type of error corrected using this strategy has a procedure designed to fix it [3].

Miscalled bases and indels in the read tend to create two distinct paths, with at least one being an h-path, connected to the same start and stop hubs, called bulges (Figure 3.1A). To fix these bulges, SPAdes uses a procedure called "bulge correction and removal" (bulge corremoval). In this method, SPAdes iterates through all h-paths

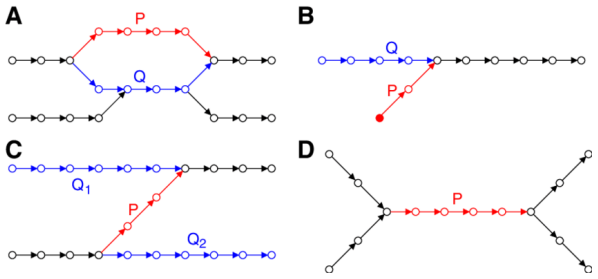


Figure 3.1: Potential errors in the graph: bulge (A), tip (B), chimeric read (C), repeat (D) [3].

in increasing order of coverage. Once it locates a bulge, from the two paths forming the bulge, every edge of the h-path is mapped to an edge in the other path. After the mapping, the mapped path is removed from the graph, and the coverage of the remaining path is increased. The information about the path removed in the corremoval is stored in a map, which makes it possible to backtrack corremovals. SPAdes takes advantage of the backtracking of corremovals later when creating a paired assembly graph.

Errors at the start or end of a read can lead to a sequence of multiple stray edges protruding from the graph called tip (Figure 3.1B). SPAdes determines, whether an h-path is a tip by considering whether there is an alternative h-path, if the length of the h-path is below a specified threshold and if the average coverage of the h-path is below a specified threshold. To perform the removal of tips, SPAdes iterates through all h-paths in order of ascending length up to the length threshold. Each h-path satisfying the conditions of being a tip is removed from the graph. When a path is removed, the parameters of the affected part of the graph are recomputed to reflect the new version of the graph. This makes it possible to remove all tips in a single iteration through the graph.

Chimeric reads and incidental short overlaps between reads sometimes lead to the creation of chimeric h-paths (Figure 3.1C). For an h-path to be considered chimeric, its start hub has to have at least two outgoing edges and its stop hub has to have no more than two incoming edges, and its length and coverage must be below a specified threshold. Additionally, some heuristics are employed to remove chimeric h-paths that do not satisfy the coverage limit because of amplification in the reads. To find h-paths satisfying the conditions, SPAdes iterates through all h-paths in order of ascending coverage.

After these graph simplifications, all isolated h-paths with lengths lower than 200 are removed.

In the second stage, SPAdes uses read-pairs to estimate the genomic distance be-

tween h-paths linked by them. The pair of h-paths is then connected by aggregation of the estimated distances between reads in read-pairs linking the h-paths. To connect the paths, read-pairs undergo a series of transformations [3].

In B-transformation, the read-pairs are transformed into k-bimers. Each k-bimer contains two k-mers from the read-pairs and a distance between the k-mers calculated as:

$$d - i_1 + i_2$$

where $i_1$ and $i_2$ are starting positions of the k-mers on the reads and $d$ is the approximate genomic distance between reads. Since each k-mer is represented in the graph as an edge, the k-bimer is also referred to as a "biedge".

Following B-transformation, H-transformation transforms biedges into h-biedges. Every biedge defined by edges residing on h-paths undergoes this transformation. Since, as mentioned earlier, every edge in the graph belongs to an h-path, H-transformation is performed on all biedges. For biedge with edges $a$ and $b$ and distance between them $d$, a h-biedge $H(a|b,d)$ is constructed as follows:

$$H(a|b,d) = (\text{h-edge}(a)|\text{h-edge}(b), D)$$

The variables h-edge($a$) and h-edge($b$) represent h-edge on the same h-path as the edge in question. The value of D is calculated as:

$$D = d + i_a - i_b$$

with $i_a$ ($i_b$) being an index of which edge from the beginning is $a$ ($b$). By executing this transformation, information about every h-biedge is collected into a histogram. Every histogram is a multiset of h-biedges with the same h-edges. Since the index for edges on the same h-path can be different, the distance estimate specified in h-biedge may vary.

After creating the h-biedge histogram, it undergoes a transformation. The histograms and the paths of the graph are analyzed by a fast Fourier transform algorithm. The analysis derives accurate distance estimates between h-edges. Using an adjustment operation, each histogram is transformed into a small number of adjusted h-biedges with distance estimates.

The h-biedges adjusted with A-transformation are then used by E-transformation to recalculate distances between biedges. For each h-biedge $(\alpha|\beta, D)$, E-transformation creates a set of biedges $(a|b,d)$, where $a$ ($b$) belongs to the same h-path as $\alpha$ ($\beta$). Distance $d$ for each biedge is calulated in the same way as in H-transformation:

$$d = D - i_a + i_b$$

where $i_a$ ($i_b$) is an index of which edge from the beginning is $a$ ($b$). The result of the transformation is a set of biedges with accurate distance estimates.

In stage three, SPAdes constructs a paired assembly graph. To construct a paired assembly graph, SPAdes attempts to find an Eulerian cycle consistent with all biedges in the de Bruijn graph [3]. An Eulerian cycle is considered consistent with a biedge if it contains instances of both edges from biedge at a distance specified in biedge. By creating a de Bruijn graph from a set of biedges where vertices are labelled as the start or stop of a biedge and directed edges labelled as biedges, the h-paths of the resulting graph are shared by the Eulerian cycles consistent with biedges. To reduce the time and memory requirements, SPAdes uses h-biedges to create the graph [3]. It creates a graph in a way, where vertices are labelled as start with the biedge with the lowest offset and stop with the biedge with the highest offset. The edge between the vertices is then labelled with the h-biedge. Since doing this only simplifies the graph, the h-paths in it are shared by Eulerian cycles consistent with biedges as well. The cycles are found in the assembly graph by pairing it with the h-biedge graph.

In the final stage, SPAdes outputs created contigs from the paired assembly graph. The contigs are represented by the h-paths in the paired assembly graph.

Since our tool is designed to work with paired-end reads, we use SPAdes in a configuration using the paired-ends library. Our tool also uses the default values for k-mers of 21, 33 and 55. Even though the input reads contain metagenomic data, we do not use metaSPAdes because we are unsure if technical sequences are present in the data and quality control could increase the runtime of the tool. For the same reason, we also avoid using -careful flag.

## 3.3 Phanotate

Phanotate[22] is a gene caller designed to identify the location of genes in phage genomes. The location of a gene in the DNA is called an open reading frame (ORF). The DNA covered by an ORF is transcribed and further utilized. Phanotate identifies the location of genes by locating ORFs. Since gene callers designed to work specifically with phage genomes are sparse, the prediction of phage genes is more accurate compared to many different gene callers. It accomplishes this by working under two assumptions. Firstly, since the phage genome is limited by physical constraints, the genome needs to be compact [13]. The compactness is partially provided by allowing a minimal amount of non-coding DNA. Secondly, because phage genes tend to be co-transcribed, they are ordinarily situated on the same strand of DNA [1]. Taking this into account, Phanotate handles the genome as a network of paths where ORFs are more favourable while overlapping ORFs and switching of DNA strains are less favourable. Phanotate then utilizes the Bellman-Ford algorithm to find the best path in the resulting weighted graph.

### 3.3.1 Algorithm

First, Phanotate creates a weighted graph representing ORFs. As a start for ORF, it allows ATG, GTG and TTG codons. To end an ORF, codons allowed are TAA, TAG and TGA. By default, the minimal length of an ORF is set to 90 nucleotides. In the graph, the nodes represent only start and stop codons. Edges between them have different meanings depending on what nodes they connect. Edges connecting the start node with the following stop node in the same read frame and on the same strand of DNA represent ORF. Gaps are represented by edges connecting the stop codon and either subsequent start codon in any reading frame on the same strand or subsequent stop codon on an alternate DNA strand. Overlaps are also represented by edges connecting the stop codon. However, they connect to either the preceding start codon in another reading frame on the same strand or the preceding stop codon on an alternate strand. Since phages seldom have gaps between ORFs, only ORFs separated by around 300 bp and less are connected by an edge. If the sequence contains a large section of DNA without an ORF, the ORFs on both sides of the section are connected with an edge with a linear penalty.

The weight of the edges is calculated based on their nature. For the edge representing ORF, Phanotate calculates weight as an adjusted likelihood of not finding a stop codon in an ORF of the length. This is done by first counting the fraction of each base in each ORF. This then determines the probability of encountering a specific end codon, which is used to calculate the probability of encountering any stop codon:

$$P(\text{stop}) = P(\text{TAA}) + P(\text{TAG}) + P(\text{TGA})$$

Calculated probability is then used to calculate the probability of not encountering any stop codon:

$$P(\text{not stop}) = 1 - P(\text{stop})$$

The P(not stop) value is sufficient in genomes having an average content of nucleotides G and C (GC content). To avoid the creation of spurious ORFs with substantial length in genomes with high GC content, Phanotate utilizes minimum and maximum GC frame plots (GCFP). The GCFPs are generated in several steps.

First, three read frames of the genome are read starting from an appropriate base and looking at a codon starting with that base to calculate the percentage of GC bases on a 120 bp window of each frame. Next, by iterating through codons of a set of ORFs starting with the ATG codon, Phanotate determines, which position of codon has the highest GC content and maintains a running total for that position. For a minimal GC frame plot, the process determines the lowest GC content. This results in a set of frequencies of GC bases for each of the three positions in ORFs starting with the ATG codon. The frequencies are then used to estimate the favoured reading frame at

any location. Each of the frequencies is then divided by the highest, resulting in values ranging between zero and one, with one being the maximal or minimal GC frame. These values are then used to exponentiate P(not stop).

The scores of ORFs are further modified by the weighted ribosomal-binding site (RBS) score. The RBS score is determined using the Shine-Dalgarno RBS system [12]. ORF scores are adjusted more by the probability of the first codon being a start. The probability is calculated as a normalized frequency of start codons from genes on 2133 phage genomes contained in GenBank. The weight of edges representing ORF is negated in the final calculation to denote them as favourable in the graph. The final calculation of the weight of the edge representing ORF is:

$$w_{orf} = -\frac{1}{\prod_{c=1}^{\text{codons}} (P(\text{not stop})^{\text{GCFPmax}_{\text{maxGCframe(c)}}\text{GCFPmin}_{\text{minGCframe(c)}}})} \cdot \text{RBS} \cdot \text{START}$$

In cases, where the edge represents a gap or an overlap, the next ORF can be on any strand of the DNA. Since the phage genes tend to be on the same strand, the switch of strands is penalised by adding a multiplicative inverse of the probability of switch to the resulting weight of the edge. The probability is expressed by a variable P(switch) attaining a value of 0 or 0.05 calculated from a set of annotated genes from 2133 phage genomes available on GenBank.

The weight of the gap is calculated using a similar method to the weight of ORF. The differences are that gaps are not corrected by the GC frame plot and that the average probability of not finding a stop codon is genome-wide and exponentiated by the length of the gap. In the case of a switch, a multiplicative inverse of variable P(switch) is added to the multiplicative inverse of the calculated probability resulting in the equation:

$$w_{gap} = \frac{1}{\underline{P}(\text{not stop})^{len}} + \frac{1}{P(\text{switch})}$$

For edges representing overlaps, the weight is calculated as the average of the two weights of the ORFs in the overlap by the length of the overlap. Similarly to the weight of the gap, in the case of a switch, a multiplicative inverse of variable P(switch) is added, resulting in an equation:

$$w_{gap} = \frac{1}{(\frac{P(\text{not stop})_1 + P(\text{not stop})_2}{2})^{len}} + \frac{1}{P(\text{switch})}$$

The calculated weights are afterwards transformed into distances by using the multiplicative inverse. With the distances and nodes, the resulting weighted graph is processed by the Bellman-Ford algorithm. Phanotate then writes ORFs from the shortest path calculated by the algorithm as an output.

Our tool slightly modifies Phanotate. The reason behind this is that since Phanotate is designed to work on whole genomes, it assumes that every input sequence

contains at least one ORF. However, our tool does not produce entire genomes for Phanotate to work on due to redundancy and a possible decrease in processing speed. Therefore, we modified the script so that if a sequence does not contain any ORFs, Phanotate will skip that sequence. We also set the format of the output as tabular, since when Phanotate exports output in fasta format, it does not include the information on which strand was an ORF found. This information is not crucial to our tool, however, it could prove useful in further analysis.

## 3.4 BEDTools

BEDTools[24] is a set of utilities created to efficiently perform common operations on genomic features. It uses a genome-binning algorithm. This algorithm assigns genomic features to 16 kb segments (bins) for the length of the chromosome using a hierarchical indexing scheme. Due to the assignment of bins, the tool only needs to compare features of two sets shared between the same or nearby bins, resulting in an accelerated search for overlapping features.

Our tool employs operation getFasta. This operation extracts parts of FASTA sequences based on a file in bed format. The bed file format is used to contain information on locations of examined features of sequences. The format can have between 3 and 12 columns per feature with 3 required being the name of the sequence and the first and last position of the feature. Additionally, our bed file contains a name for the feature and the strand on which the feature is located. The score is required in the file to maintain the correct file structure. Since we do not need it for other purposes, the score is marked as a dot, meaning it is omitted.

In our pipeline, the output from Phanotate is in tabular format, since in FASTA format Phanotate removes information on which strand the discovered ORF is located. Because this information can be useful in further analysis of the sequences, the output from Phanotate is exported in tabular format. Next, the output is transformed into bed format using UNIX utilities. Afterwards, the bed file is used to extract FASTA sequences of discovered ORFs. While our method is slower than directly exporting the results from Phanotate in FASTA format, the difference in expediency is negligible even with large inputs.

## 3.5 Transeq

EMBOSS transeq[19] is a tool designed to translate nucleotide sequences into their protein equivalent. Translation can be performed in different combinations of reading frames. If necessary, it can be restricted to specific sections of sequences. The transla-

tion is facilitated by a predetermined genetic code with a selection of codes available in the tool.

Transeq is used by our pipeline to translate ORFs discovered by Phanotate into protein sequences. Since the function of a protein is set by its protein structure rather than its nucleotide structure, the translation allows the pipeline to more accurately predict proteins with the desired function. Input for transeq in our tool is already filtered to only include ORFs, making restriction of the tool to specific sections unnecessary. For the same reason, the only translation performed is in the first forward frame. The genetic code of phages is identical to the standard code used by transeq, making it possible to omit in the setup of the tool.

## 3.6 Blast

Basic Local Alignment Search Tool (Blast)[2] is a sequence analysis tool designed to perform a sequence similarity search of DNA or protein sequences on a database of known sequences to infer the function of a sequence from similar sequences in the database. It performs local alignments using a similar method as the Smith-Waterman algorithm [27]. The main advantage of using BLAST over the standard Smith-Waterman algorithm lies in its ability to produce results quickly. To achieve quick results, BLAST uses heuristics. This causes BLAST to be less accurate and permits some similarities to not be detected. The drawback of lower precision is largely offset by BLAST being approximately 50 times faster than the Smith-Waterman algorithm, making BLAST the most widely used tool for the examination of DNA and protein sequences.

### 3.6.1 Algorithm

BLAST workflow is divided into several steps [7]. First, BLAST reads the query search parameters and the database and removes low-complexity regions and sequence repeats from the query. The region is considered low-complexity in cases where its sequence is composed of a small number of elements. These regions are removed to prevent them from confusing the program by having high scores. The removal of these regions is facilitated by programs SEG and DUST, used on protein and DNA sequences respectively. Filtering of tandem sequences is handled by program XNU.

After filtering unwanted regions, BLAST makes a word list of the query sequence. A word in this sense is a subsequence of fixed length. The list is created by passing through the sequence one base at a time and creating a word starting from each base with the fixed length until every base is included in a word. The words are then assigned scores by comparison to all words of the same length. The score of the comparison of

each pair is created according to a scoring matrix. By using a neighbour word score threshold, the number of possible matches is reduced. For a word to remain as a viable matching word, its score is required to be higher than the threshold. The remaining words form a search tree to allow their fast comparison.

The database is then scanned for exact matches with the remaining words. Matches in the database are used as seeds for gap-free alignments. Afterwards, BLAST stretches the alignment from the exact match location in both directions until the total score of the alignment begins to decrease, creating high-scoring segment pairs (HSP). The HSPs with scores lower than a cutoff score (S) are removed, leaving only significant HSPs.

Using Gumbel extreme value distribution, the probability of score S being equal or greater than variable x can be calculated as:

$$p(S \geq x) = 1 - \exp(-e^{-\lambda(x-\mu)})$$

To calculate $\mu$, the equation used is:

$$\mu = \frac{\log(Km'n')}{\lambda}$$

In this equation, values of $\lambda$ and K are estimated by fitting the distribution of gap-free alignment scores, query sequence and shuffled versions of the database, to the Gumbel EVD. Since alignment starts near the end of a sequence is not likely to build optimal alignment, the length of a sequence is shortened to the effective length labelled m' and n' and is calculated as:

$$m' \approx m - \frac{\ln Kmn}{H}$$

$$n' \approx n - \frac{\ln Kmn}{H}$$

The variable H represents the average expected score for each aligned pair of residues in an alignment of two random sequences. Values in the lookup table given by Altschul and Gish are $\lambda = 0.318$, K = 0.13 and H = 0.4. These values can be used instead of calculating custom ones, however, this method is not accurate.

One of the most significant values produced by BLAST is the predicted number of times a random sequence from the database would by chance have the score S higher than x. This value is called the expected score (E), and is calculated for a database containing D sequences with the equation:

$$E \approx 1 - e^{-p(S>x)D}$$

When the probability is less than 0.1, the E value can be approximated using the Poisson distribution:

$$E \approx pD$$

After calculating the E value, BLAST combines HSP regions into longer alignments when possible. The Poisson method is then used to compare the significance of new HSP regions. These HSP regions can contain gaps as well as insertions and deletions. With the HSPs solved, BLAST returns an output with only matches, that have an E value lower than a set threshold.

Our tool uses a custom database of discovered endolysins downloaded from the database Phalp. Phalp database contains sequences for endolysins and tail fibre lysins of bacteriophages [5]. From this database, we extract only sequences of endolysins. We use a version of BLAST called BLASTP, which is used for alignment search on protein sequences. The output we use from BLASTP is the query sequence id, id of the subject sequence, start and end of the alignment on the query sequence, percentual identity between sequences, and the length of an alignment and subject sequence. For the search, we do not specify any other parameters.

## 3.7 Blast2out

Blast2out is our custom python script designed to transform tabular output from Blast to desired output in fasta format containing sequences predicted to encode endolysins. It includes several options for adjustment of the output format. As an input, it requires a file containing ORFs in fasta format, parameter based on which the output is sorted, desired minimal length of endolysins, whether Uniprot accessions of hits from Blast should be included and output from Blast in tabular format. Blast2out requires the ORF file and Blast output to have sequences sorted in the same order. Since the script is only used as an output creator and the files are sorted in the same way by default, it is deemed unnecessary to sort them in the script.

The script starts by reading entries from blast output. For each entry, it checks if the name of the entry is the same as the name of the ORF sequence it holds in its memory. While the names are the same, the script appends selected information about the entries with a length greater than or equal to the minimal length set by the user. Once the names no longer match, Blast2out calculates the coverage of the current ORF. The calculation is done by first sorting information on coordinates of the hits and then counting the number of bases of ORF covered by the entries. The result is divided by the total length of the ORF. The product is the coverage of the ORF. The name of the ORF, its coverage, entries covering it and the sequence are then appended to the list of processed ORFs. After that, the script resumes reading the entries.

Once the entire input is read, the list containing processed ORFs is sorted based on the option set by the user. The options for sorting include sorting by the name of the ORF, length of its sequence, coverage by hits from Blast and amount of Blast hits

on the sequence. After sorting, Blast2out prints the results. If the option is set, the results are printed with Uniprot accessions.

Our pipeline uses Blast2out to transform Blast output, which does not include all the required information, into a file in fasta format. The sequence is crucial for our tool, as it is what we consider a possible endolysin. We use the information about coverage as an indicator of the likelihood that the ORF in question encodes endolysin. We do this based on the assumption that to retain their lytic properties, the unknown endolysins have a similar protein structure to endolysins already discovered. The Uniprot accession is not included in our output, since it can be unnecessarily long and complicate the reading of the results.

## 3.8 Snakemake

Snakemake[16] is a text-based workflow management system designed to facilitate reproducible and scalable data analyses. It uses a similar pattern to GNU Make [28]. Analogously to GNU Make, Snakemake workflows are made up of rules that specify how to create output files from input files. The rules form a directed acyclic graph based on dependencies automatically resolved by the manager. Snakemake workflows are described in a human-readable Python-based language, making them highly accessible. To allow scalability, the scheduling algorithm of Snakemake can be provided with specific information on priorities in the workflow as well as available resources.

Our workflow consists of six rules. There are no forks in the workflow, meaning that all rules are executed in an exact sequence.

The first rule uses as an input two files containing paired-end reads and uses the tool Seqtk to create a sample file with a specified number of sequences.

The second rule takes the output from the first rule, using SPAdes to make contigs from raw reads. It then modifies the contigs file by removing unnecessary information following the names of the contigs. The modified file is saved to the results folder along with a GFA file containing information to create a graph from created contigs. These files are moved to the folder to be accessible in case of further analysis of results and are the output of this rule.

The third rule uses a modified contigs file as an input. It uses Phanotate to find the locations of ORF in a tabular input, which is then modified using the awk command into a bed file.

Using the bed file as an input, the fourth rule extracts ORF sequences from the contigs file.

The fifth rule transforms the nucleotide ORF sequences into protein sequences using Transeq.

The sixth rule works with the file containing protein ORF sequences. By using Blastp, it searches a database of known endolysins prepared beforehand for hits with sequences in the input file. The rule then filters out hits, that have a low identity or cover only a short part of a sequence from the database, with the awk command. Finally, it uses Blast2out to attach hits to ORF sequences and calculate the coverage of ORFs to determine which ORFs have a high probability to be endolysins.

When the workflow finishes, the folder for results contains four files: `contigs.fasta` and `assembly_graph_with_scaffolds.gfa` from the first rule, `ORF_phanotate.fasta` from the fourth rule and `endolysins.fasta` from the sixth rule. The predicted endolysins are in the file `endolysins.fasta` while the remaining files are included mainly for further analysis.

## 3.9 Phendol

Phendol is a bash script responsible for interaction between the user and the pipeline. It receives input file names and optional parameters, processes them and starts Snakemake using them to control it. It processes the parameters as flags, reading them one at a time and saving them to variables. The input files have to be preceded by flags `-r1` and `-r2` for forward paired-end reads and reverse paired-end reads respectively. They also need to be in a fastq.gz format commonly used by Illumina sequencers.

There are currently several available optional flags to modify settings of the tool, usually having short and long versions. With option `-h` (`--help`), the user can browse all available flags as well as see a usage message. The option `-d` (`--dry`) allows the user to do a dry run of the program, which checks whether the program can run with set input.

Apart from input files, Phendol has set default values for all flags. By default, Phendol is set to not sample input data and work with entire files. This can, however, lead to high memory usage. For this reason, it is advisable to use subsampling (`-s`/`--subsample`) in cases where the used equipment has lower technical specifications. Threads used by SPAdes (`-t`/`--threads`) are set to 16, which is the same number as default for SPAdes. For the filtering of Blast search, the minimal percentage of identical matches (`-p`) is set to 75.0% and the minimal percentage of coverage of sequence from the database (`-c`/`--coverage`) is set to 0.5 (range [0.0-1.0]). Another default value is for minimal length of predicted endolysins (`-ml`/`--min_length`), which is set to 50 amino acids. This value was deemed optimal since it removes short proteins with a low probability of being endolysins while keeping the value as low as possible.

When considering output format, the default sorting of the output (`-f`/`--sort`) is by name, as it is easier to work with this form of output in additional analysis. For

similar reason, Uniprot accessions (`-u`/`--uniprot`) are not included by default.

We have decided not to include many different options, as too many algorithmic options can have a discouraging effect on wet lab users. Since this tool aims to be accessible to less technically proficient users, the options are directed at the functions, which are easy to understand.

## 3.10 Conda

Conda[31] is a package and environment management system designed to install, run and update packages. Its main purpose is to simplify the management and accessibility of packages. Our tool is highly dependent on different tools. To install every tool necessary correctly would require a non-trivial effort. By building our tool as a Conda package, it can be installed by using a single command. Conda then installs our tool as well as all dependencies and resolves any possible conflicts. This simplification of installation makes our tool user-friendly.

## 3.11 Summary

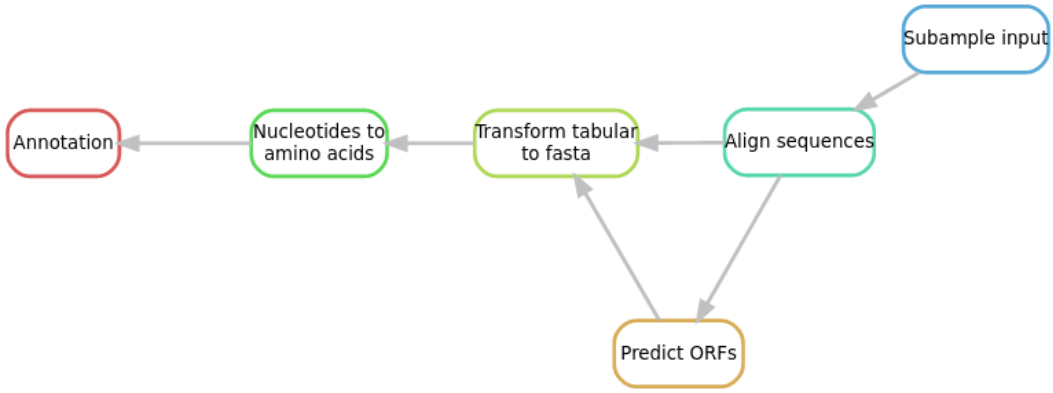


Figure 3.2: Steps the pipeline takes while annotating.

Phendol, our workflow, can be summarized in a few steps (Figure 3.2). The first step is the assembly of contigs. In this step, the tool processes input reads by first creating a subsample from the files using Seqtk and then creating an assembly using SPAdes. The second step is the location of ORFs on the resulting assembly. By using Phanotate, the tool first finds the coordinates of ORFs likely to contain phage genes. Using Bedtools on the coordinates, the Phendol finds DNA sequences of the ORFs. These ORFs are used in the third step. The third step is the comparison with an endolysin database. In this step, Phendol translates DNA into protein sequences using

Transeq. Using Blast, the database of endolysins is searched for matches to translated sequences. The fourth step is the filtering of endolysins. In this final step, sequences are filtered by Blast2out to leave only those our tool assumes to be endolysins. The success of this assumption is evaluated in the next chapter.

# Chapter 4

# Testing

This chapter displays results generated using our tool and compares them with results from similar programs. It also discusses the source of data used for comparison as well as settings used by individual programs.

## 4.1 Source of data

For our tests to have any informative value, we need to know a large amount of information about the samples used. Since we did not possess samples with sufficient information, we decided to simulate our samples. We simulate our data using InSilicoSeq, a tool designed to generate simulated sequenced reads from sequences input by the user. The reads can be generated using either one of the pre-build error models or a custom model generated using a bam file with aligned reads. The pre-build error models include models for Illumina sequencers MiSeq, HiSeq, and NovaSeq.

To generate our samples, we use all three default error models. As the input sequences, we use a variable number of random phage genomes downloaded from NCBI. For our purposes, we used either ten or fifty genomes. We also adjust InSilicoSeq to generate one or five million reads per sample. We have decided that using other numbers of reads or genomes is redundant and thus do not use them. With these settings, we generate a total of twelve paired-end input samples.

## 4.2 Settings

Aside from Phendol, we process the generated samples using multiPhATE and RASTtk. In this section, we describe the settings used for each tool.

### 4.2.1 Phendol

We set Phendol to sample two million reads from the generated samples. This allows us to test Phendol on a full-sized dataset in case of the samples containing one million reads and test it on a sampled dataset when working with samples containing five million reads. We used 32 threads to run SPAdes. Contigs generated by SPAdes were later used in the analysis by other tools as well.

To test different setups of Phendol, we set the minimal percentage of identical matches in blast search to 75% and 90%. For the same reason, we set the minimum percentage of coverage of sequences from the endolysin database to 50% and 75%. We also made an analysis using values of 50% for identity and 30% for coverage.

The minimal length of endolysins was left at the default value of 40 amino acids. Finally, we sorted the output by the percentage of the sequence covered by hits from the database.

To reduce the runtime, we ran SPAdes only once per sample by manually copying its results into the working directories of other analyses. With these settings, we ran the pipeline five times per sample, resulting in a total of sixty analyses.

### 4.2.2 multiPhATE

In the comparison, we use multiPhATE2, which is an enhanced version of multiPhATE. MultiPhATE uses a config file to specify all parameters. In the config file, we enabled the use of gene callers Phanotate, Prodigal, and Glimmer. We also enabled the blastp search. MultiPhATE has two default databases against which it can run the blastp search. These are Prokaryotic Virus Orthologous Groups (pVOGs) [9] and The Phage Annotation Tools and Methods (PhAnToMe) database. In our setup, we use both.

We also set the list of genomes used in the analysis in the config file. In our case, the genomes set were the contigs produced by SPAdes during the run of Phendol. With the configuration complete, we ran multiPhATE to get the analyses for every sample.

### 4.2.3 RASTtk

We attempted to install the command-line version of RASTtk, however, since the command-line version is outdated and not usable, we used the online browser version.

For the `create-genome` script, we set the domain as virus and the genetic code to 11 Archaea, most Bacteria, most Virii, and some Mitochondria). In our custom pipeline used for annotation, we use most of the scripts used in the default pipeline except for `call-strep-suis-repeat` and `call-strep-pneumo-repeat`, since the sequences we are annotating do not contain bacterial DNA. We also switch off the

`resolve-overlapping-features` script because viral genomes tend to contain overlapping features.

We used this pipeline to analyse the contigs created by SPAdes during the run of Phendol.

## 4.3 Comparison

In running RASTtk, we were unable to get results using the sample `50HiSeq1000000`. As a result, we only achieved results in the remaining eleven samples. In these results, we only counted the number of sequences directly designated by RASTtk as encoding endolysins, since the number of genes discovered with a less precise designation (e.g. Hypothetical protein) is impractically large and it is not feasible to verify them in the laboratory.

While running multiPhATE, we encountered several errors due to unknown reasons. This led to only analyses on samples `MiSeq` being completed. As with RASTtk, we only include sequences designated as encoding endolysins by multiPhATE directly.

Since the number of phages discovered by Phendol tended to vary between the number discovered with an identity of 75% and coverage of 50% and the number discovered with an identity of 90% and coverage of 75%, we do not include other combinations of these values in our comparison. We also filter the results of the analysis with identity set to 50% and coverage 30% because the unfiltered result contains a large number of sequences with a low probability of being an endolysin. We consider sequence as having a low probability of being an endolysin when the coverage of the sequence by Blast hits is lower than 50%. In the comparison, we include both the filtered and unfiltered analysis.

When comparing the results, we decided to base our ground truth on the number of genomes included in each sample. In doing so, we assume that every phage genome contains at least one endolysin encoding gene to facilitate its ability to complete its lytic cycle. We also assume that every phage genome contains at most one endolysin encoding gene. In our comparison, we prefer false positives to false negatives. We make this preference because the false positives can be further verified in wet-lab while the false negatives cannot be restored. As a result, our ground truth is ten for analyses done on samples containing ten genomes and fifty for analyses done on samples with fifty genomes.

Compared with RASTtk, Phendol was generally capable of annotating a larger number of endolysins, with the filtered analysis predicting a significantly higher number of endolysins (Figure 4.1).

When compared with multiPhATE, Phendol discovered mostly a lower number of

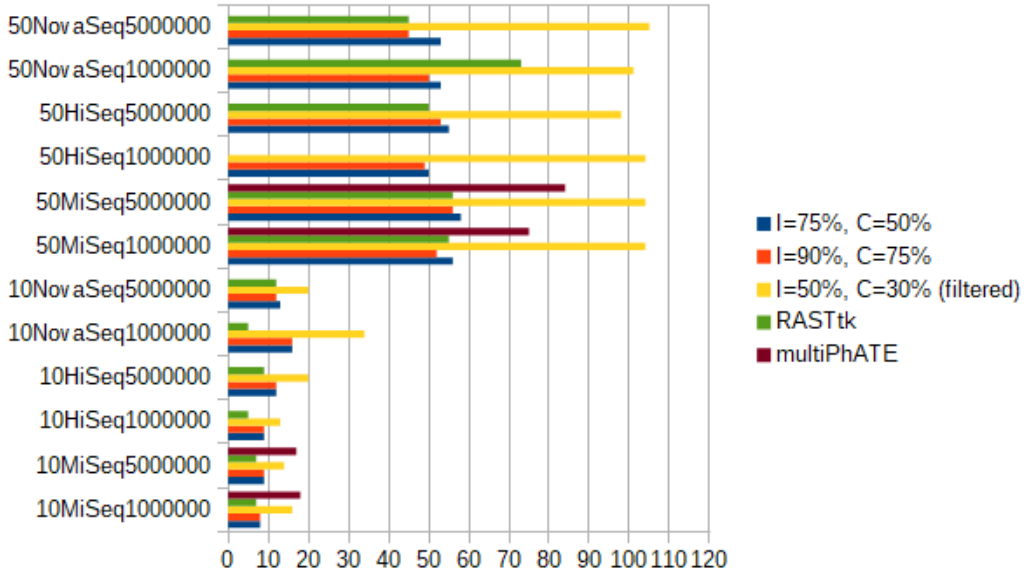


Figure 4.1: Number of predicted endolysins per sample by analysis method.

endolysins while being closer to our ground truth (Figure 4.1). The only exception was the filtered analysis, which predicted a similar or higher number of endolysins.

In comparison with both RASTtk and multiPhATE, the predictions made by Phendol with identity between 75-90% and coverage between 50-75% were closer to our ground truth making Phendol more accurate. By decreasing the identity and coverage Phendol has the option of decreasing the number of false negatives at the cost of an increase in the number of false positives. This exchange is advisable in order to discover endolysins more thoroughly as it facilitates more wet-lab verification.

# Conclusion

Results from our comparison of Phendol with other tools show that Phendol is capable of annotating endolysins with similar or higher accuracy than other annotation pipelines. The biggest contribution of Phendol lies in the simplicity of its usage. While installation of other tools took a significant amount of effort, the installation of Phendol can be accomplished with a single command. For instance, in our effort to compare our tool with RASTtk, we were unable to execute the command line application. We were instead forced to use the online browser version which requires a registration manually approved by the creators of RASTtk.

Execution of Phendol is also considerably easier. Thanks to Conda, Phendol can be initialized from any directory. The parameters Phendol uses can be adjusted using flags and the only required user input is the location of files for analysis. On the contrary, using multiPhATE requires the input files to be in a particular directory, specific setup using a config file, and preparation of databases beforehand, making multiPhATE cumbersome.

Since the results of Phendol are predicted endolysins, the tool is more suited for the analysis than the other tools, which produce results that need to be further filtered to leave only endolysins remaining.

Every feature mentioned constitutes a more user-friendly interface, which is one of the goals of this work. In the future, we hope to extend the functionality of Phendol by including more methods to assemble reads. This would allow Phendol to work with different sequencing reads than the currently used paired-end reads as well as facilitate the use of more precise assemblers. We would also like to add the option of automatically updating the used endolysin database to include new and verified endolysins in order to make Phendol more user friendly.

# Bibliography


[1] Sajia Akhter, Ramy K Aziz, and Robert A Edwards. Phispy: a novel algorithm for finding prophages in bacterial genomes that combines similarity-and composition-based strategies. *Nucleic acids research*, 40(16):e126–e126, 2012.

[2] 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.

[3] Anton Bankevich, Sergey Nurk, Dmitry Antipov, Alexey A Gurevich, Mikhail Dvorkin, Alexander S Kulikov, Valery M Lesin, Sergey I Nikolenko, Son Pham, Andrey D Prjibelski, et al. Spades: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of computational biology*, 19(5):455–477, 2012.

[4] Thomas Brettin, James J Davis, Terry Disz, Robert A Edwards, Svetlana Gerdes, Gary J Olsen, Robert Olson, Ross Overbeek, Bruce Parrello, Gordon D Pusch, et al. Rasttk: a modular and extensible implementation of the rast algorithm for building custom annotation pipelines and annotating batches of genomes. *Scientific reports*, 5(1):1–6, 2015.

[5] Bjorn Criel, Steff Taelman, Wim Van Criekinge, Michiel Stock, and Yves Briers. Phalp: A database for the study of phage lytic proteins and their evolution. *Viruses*, 13(7):1240, 2021.

[6] Carol L Ecale Zhou, Stephanie Malfatti, Jeffrey Kimbrel, Casandra Philipson, Katelyn McNair, Theron Hamilton, Robert Edwards, and Brian Souza. multiPhATE: bioinformatics pipeline for functional annotation of phage isolates. *Bioinformatics*, 35(21):4402–4404, 05 2019.

[7] Martin Gollery. Bioinformatics: sequence and genome analysis. *Clinical Chemistry*, 51(11):2219–2220, 2005.

[8] Fernando L Gordillo Altamirano and Jeremy J Barr. Phage therapy in the postantibiotic era. *Clinical microbiology reviews*, 32(2):e00066–18, 2019.

[9] Ana Laura Grazziotin, Eugene V Koonin, and David M Kristensen. Prokaryotic virus orthologous groups (pvogs): a resource for comparative genomics and protein family annotation. *Nucleic acids research*, page gkw975, 2016.

[10] Patrick Guilfoile and I Edward Alcamo. *Antibiotic-resistant bacteria*. Infobase Publishing, 2007.

[11] Burton Guttman, Raul Raya, and Elizabeth Kutter. Basic phage biology. *Bacteriophages: Biology and applications*, 4, 2005.

[12] D Hyatt et al. Prodigal: prokaryotic gene recognition and translation initiation site identification. bmc bioinformatics 11, 119–119, doi: 10.1186. 2010.

[13] Han Suh Kang, Katelyn McNair, Daniel A Cuevas, Barbara A Bailey, Anca M Segall, and Robert A Edwards. Prophage genomics reveals patterns in phage genome organization and replication. *BioRxiv*, page 114819, 2017.

[14] Manuel Kleiner, Lora V Hooper, and Breck A Duerkop. Evaluation of methods to purify virus-like particles for metagenomic sequencing of intestinal viromes. *BMC genomics*, 16(1):1–15, 2015.

[15] Kamila Knapik. *Genetic analysis of bacteriophages from clinical and environmental samples*. PhD thesis, 07 2013.

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

[17] Heng Li. seqtk toolkit for processing sequences in fasta/q formats. *GitHub*, 767:69, 2012.

[18] Derek M Lin, Britt Koskella, and Henry C Lin. Phage therapy: An alternative to antibiotics in the age of multi-drug resistance. *World journal of gastrointestinal pharmacology and therapeutics*, 8(3):162, 2017.

[19] Fábio Madeira, Matt Pearce, Adrian Tivey, Prasad Basutkar, Joon Lee, Ossama Edbali, Nandana Madhusoodanan, Anton Kolesnikov, and Rodrigo Lopez. Search and sequence analysis tools services from embl-ebi in 2022. *Nucleic Acids Research*, 2022.

[20] Maryn McKenna. The last resort: health officials are watching in horror as bacteria become resistant to powerful carbapenem antibiotics–one of the last drugs on the shelf. *Nature*, 499(7459):394–397, 2013.

[21] Katelyn McNair, Carol Zhou, Elizabeth A Dinsdale, Brian Souza, and Robert A Edwards. PHANOTATE: a novel approach to gene identification in phage genomes. *Bioinformatics*, 35(22):4537–4542, 04 2019.

[22] Katelyn McNair, Carol Zhou, Elizabeth A Dinsdale, Brian Souza, and Robert A Edwards. Phanotate: a novel approach to gene identification in phage genomes. *Bioinformatics*, 35(22):4537–4542, 2019.

[23] Hiroshi Nikaido. Multidrug resistance in bacteria. *Annual review of biochemistry*, 78:119–146, 2009.

[24] Aaron R Quinlan and Ira M Hall. Bedtools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, 26(6):841–842, 2010.

[25] Rupesh Kanchi Ravi, Kendra Walton, and Mahdieh Khosroheidari. Miseq: a next generation sequencing platform for genomic analysis. *Disease gene identification*, pages 223–232, 2018.

[26] Mathias Schmelcher, David M Donovan, and Martin J Loessner. Bacteriophage endolysins as novel antimicrobials. *Future microbiology*, 7(10):1147–1171, 2012.

[27] Temple F Smith, Michael S Waterman, et al. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[28] Richard M Stallman and Roland McGrath. Gnu make-a program for directing recompilation. 1991.

[29] Thomas DS Sutton, Adam G Clooney, Feargal J Ryan, R Paul Ross, and Colin Hill. Choice of assembly software has a critical impact on virome characterisation. *Microbiome*, 7(1):1–15, 2019.

[30] Dieter Vandenheuvel, Rob Lavigne, and Harald Brüssow. Bacteriophage therapy: advances in formulation strategies and human clinical trials. *Annual review of virology*, 2:599–618, 2015.

[31] Yuxing Yan and James Yan. *Hands-on data science with Anaconda: utilize the right mix of tools to create high-performance data science applications*. Packt Publishing Ltd, 2018.

[32] Sheng Zhang and De-Chang Chen. Facing a new challenge: the adverse effects of antibiotics on gut microbiota and host immunity. *Chinese medical journal*, 132(10):1135, 2019.

# Appendix A: Implementation

This thesis includes an electronic attachment containing the source code of our tool as well as files necessary to build a Conda package. The scripts used by our tool are contained in the `scripts` directory.

We recommend installation using the Conda environment. The instructions for installation of the tool using Conda are included in the `README.md` file.