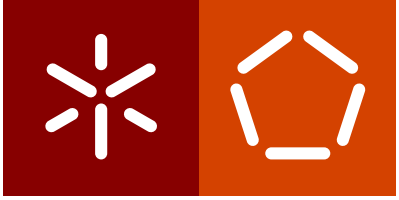
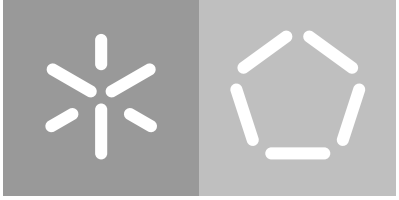


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Pedro Henrique Matela Aidos Manso de Araújo

Bacteriophage-host determinants: identification of bacteriophage receptors through machine learning techniques

February 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Pedro Henrique Matela Aidos Manso de Araújo

Bacteriophage-host determinants: identification of bacteriophage receptors through machine learning techniques

Master dissertation
Master Degree in Bioinformatics

Dissertation supervised by
Oscar Manuel Lima Dias
Hugo Alexandre Mendes de Oliveira

February 2021

Acknowledgments

If you're going through hell, keep going

Winston Churchill

Queria começar por agradecer aos meus orientadores pela confiança depositada, por me terem aceite para desenvolver este trabalho. Ao professor Hugo Oliveira, pela ajuda constante e enorme vontade de aprender e transmitir conhecimento. Ao professor Oscar Dias, o meu profundo obrigado, pela ajuda dentro e fora da tese e pela disponibilidade.

Agradeço também aos colegas do laboratório, Diogo e Fernando, pelas ajudas nos servidores, e um agradecimento especial à Marta, pela disponibilidade em ajudar e pelas reuniões, quando a tese parecia que não queria avançar. Aos colegas da sala 09, sem os quais a segunda metade da tese não teria sido a mesma.

Um agradecimento às pessoas sem as quais não teria conseguido fazer a tese. Ao Dias, pelas horas a bater código, pela partilha de scripts e conhecimento. Carolina, pelas caminhadas e disponibilidade nos piores momentos. Joana por todas as gargalhadas e pela companhia constante, independentemente de tudo. Débora, mestre do LaTeX, sempre disponível para conversas e para ajudar. Obrigado pela preocupação e motivação que me deram.

Finalmente agradeço à minha família, que esteve sempre lá para mim. Em especial à minha mãe, por me conseguir aturar, apesar de ser muito difícil. Ao meu irmão, programador mais a sério que eu, por me ajudar com tecnicalidades, e sem o qual não seria a pessoa que sou hoje. Ao meu irmão mais novo, por ser uma fonte constante de energia positiva.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Abstract

Bacterial resistance to antibiotics is nowadays becoming a major concern. Several reports indicate that bacteria are developing resistance mechanisms to various antibiotics. Moreover, the processes involved in the development of new antibiotics are lengthy and expensive. Therefore, an alternative to antibiotics is needed. One promising alternative are bacteriophages, viruses that specifically infect bacteria, causing their lysis. Hence, it would be interesting to discover which bacteria a specific phage recognizes. The bacterial receptors determine phage specificity, using tail spikes/fibres as receptor binding proteins to detect carbohydrates or proteins, in bacterial surface. Studying interactions between phage tail spikes/-fibres and bacterial receptors can allow the identification of interaction pairs. Machine learning algorithms can be used to find patterns in these interactions and build models to make predictions.

In this work, PhageHost, a tool that predicts hosts at a strain level, for three species, *E. coli*, *K. pneumoniae* and *A. baumannii* was developed. Several data was extracted from GenBank, retrieving general, protein and coding information, for both phages and bacteria. The protein data was used to build an important phage protein function database, that allowed the classification of protein functions, namely, phage tail spikes/fibres. In the end, several machine learning models with relevant protein features were created to predict phage-host strain interactions. Compared with previously performed works, these models show better predictive power and the ability to perform strain-level predictions. For the best model, a Matthews correlation coefficient (MCC) of 96.6% and an F-score of 98.3% were obtained. These best predictive models were implemented online, in a server under the name PhageHost (https://galaxy.bio.di.uminho.pt).

Keywords: bacteriophages, phages, host prediction, bacterial strain, machine learning.

Resumo

Resistência bacteriana a antibióticos está a tornar-se uma preocupação hoje em dia. Várias bactérias foram descritas desenvolvendo mecanismos de resistência a diversos antibióticos. Aliado a isto, estão os longos e dispendiosos processos envolvidos no desenvolvimento de antibióticos. Por isso, há a necessidade de procurar uma alternativa aos antibióticos. Uma alternativa promissora são os bacteriófagos, vírus que infetam especificamente bactérias e levam à sua lise. Posto isto, seria interessante descobrir qual a bactéria que um certo fago reconhece. A especificidade de fagos é dada pelos recetores da superfícies das bactérias que conseguem reconhecer. Eles usam proteínas das *spikes*/fibras para reconhecer recetires proteicos ou hidratos de carbono nas bactérias. Estudar as interações entre *spikes*/fibras das caudas de fagos e recetores bacterianos pode permitir a identificação de pares de interação. Algoritmos de aprendizagem máquina podem ser utilizados para descobrir padrões nestas interações e construir modelos para realizar previsões.

Neste trabalho, a ferramenta PhageHost foi desenvolvida. Permite a previsão de hospedeiros ao nível da estirpe, para três espécies, *E. coli*, *K. pneumoniae* e *A. baumannii*. Vários dados foram extraídos do GenBank, nomeadamente informações gerais, de proteína e codificante, para fagos e bactérias. Com todos os dados proteicos, uma base de dados importante foi construída, que permitiu a classificação de funções proteicas, nomeadamente, *spikes*/fibras das caudas dos fagos. Finalmente, vários modelos de aprendizagem máquina, com características proteicas relevantes, capazes de prever interações fago-hospedeiro, a nível da estirpe. Em comparação com outros trabalhos semelhantes, estes modelos demonstraram melhor poder preditivo, assim como capacidade de prever interações a nível da estirpe. Para o melhor modelo foram obtidos um coeficiente de correlação de Matthews de 96.6% e um F-score de 98.3%. Os melhores modelos foram implementados online, num servidor com o nome PhageHost (https://galaxy.bio.di.uminho.pt).

Palavras-Chave: bacteriófago, fago, previsão de hospedeiro, estirpe bacteriana, aprendizagem máquina.

Contents

Abbreviations

aa Amino acid. 53

ANN Artificial Neural Networks. 20, 33, 40, 54–56, 58, 59

BLAST Basic Local Alignment Search Tool. 16, 17, 24, 34, 36, 37, 42, 44, 45, 48

C Cytosine. 17, 21

Cas CRISPR associated proteins. 17

CRISPR Clustered Regularly Interspaced Short Palindromic Repeats. 17–19

CSV Comma-Separated Values. 34

CT Conjoint Triad. 25, 39

DNA Deoxyribonucleic acid. 1, 4–6, 13, 14, 17, 20, 21

dsDNA double-stranded DNA. 4, 5

FN False Negatives. 29

FP False Positives. 29

G Guanine. 17, 21

HMM Hidden Markov Models. 24

ID Identifier. 34, 43–45, 47, 48, 62, 65

ID3 Iterative Dichotomiser 3. 32

IDE Integrated Development Environment. 43

KNN K-Nearest Neighbours. 20, 32, 40, 46, 54–56, 58, 59

LPS Lipopolysaccharide. 9–11

LR Logistic Regression. 33, 40, 54–56, 58, 59, 82

MAD Mean of Absolute Deviation. 29

MCC Matthews correlation coefficient. v, 28, 41, 47, 59, 66

ML Machine Learning. 15, 18–21, 23–27, 30, 31, 33, 35, 38–42, 46–48, 51, 52, 54–59, 63, 66

NAG *N*-acetylglucosamine. 7, 8

NAM *N*-acetylmuramic acid. 7, 8

NCBI National Center for Biotechnology Information. 15, 34, 35, 43, 44, 47, 49, 62, 65

PDB Protein Data Bank. x, 22, 23

PECC Percentage of Examples Correctly Classified. 28, 29

PSI Position-Specific Iterative. 24

PSSM Position-Specific Scoring Matrix. 22

RBF Radial Basis Function. 31

RF Random Forests. 20, 40, 41, 47, 54–59, 63, 66

RMSE Root Mean Squared Error. 29

RNA Ribonucleic acid. 4, 5, 17

ROC Receiver Operating Characteristic. 29

RSA Relative Accessible Surface Area. 38

ssDNA single-stranded DNA. 5

SSE Sum of Square Errors. 29

SVM Support Vector Machines. x, 20, 31, 32, 40, 47, 54, 55, 58, 59, 63, 66

TN True Negatives. 29

TP True Positives. 29

TSV Tab-Separated values. 42, 63, 64

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Bacteriophages (phages), are viruses that infect and lyse bacteria. Generally, phages recognize their hosts by adsorption of their tail fibres/spikes proteins to the bacterial receptors, that can either be of proteins or polysaccharides. After adsorption, they enter the cell, replicate and lyse the bacterial host, releasing the progeny[1].

Development of resistance by bacteria is a major healthcare issue in current time. Antimicrobial resistance has spread to the general public. Bacteria have and are developing resistance mechanisms to various antibiotics[2;3]. Various mechanisms of resistance exist, from the destruction or modification of the antibiotic, efflux from the cell, modification of receptors, to mutations and acquisition of foreign DNA (plasmids)[2].

There is, therefore, a need for an alternative to antibiotics. A promising alternative are phages. Recently, there has been a renewed interest in phages applied to humans, as an alternative to antibiotics. Therefore, several phages have been sequenced and characterized, which gives us more knowledge and insight into phage-bacterial interaction. The large amount of data generated is also essential for building algorithms that identify them.

Identifying interactions between phages and their hosts is crucial in order to know which virus to apply to a given bacterial infection. Tail spikes/fibres are proteins used by phages to recognize bacterial hosts, defining their host range[4]. To successfully know, beforehand, which phage is used against a given bacteria, we propose machine learning based approaches, using these tail proteins as features for classification and using phage tail-host receptor interactions for classification, at strain level

Work performed in this area was only applied to predict interactions at bacterial species level, at most. Other more general works of molecular interactions were mostly based on tridimensional structures, which are scarcely available for phages.

There are three types of machine learning algorithms: unsupervised, supervised and reinforcement learning[5]. We will use supervised learning, in which, a dataset is represented by a set of features with

known outputs. For the training, each set of features has a correct answer. With the model built, a given set of inputs will predict a certain output. This methodology has the potential to predict almost every phage host, since it involves prediction of protein and polysaccharide-based bacterial receptors.

1.2 Objectives

The main objective of this work is to develop supervised machine learning models for the analysis of phage-host specificity. The models will be generated using supervised methods to predict phage receptors specific to a given set of hosts. More specifically, the work will address the following scientific/technological objectives:

- Review the relevant literature for machine learning methods and their applications in related scenarios;
- Studying the available data sources of phage and bacterial datasets, retrieving their protein and coding sequences;
- Developing pipelines for tail fibre/spike classification, using the available data and evaluating the different alternatives based on the defined criteria;
- Developing machine learning supervised pipelines for phage-host interactions, using the available data and evaluating the different alternatives based on the defined criteria;
- Publishing the developed tools on a web-platform, to make it easy to use for scientists;
- Writing the thesis and, possibly, scientific publications with the main results of this work.

2 State of the art

2.1 Historical context on bacteriophages

Phages were first discovered by Frederick Twort in 1915, after observing antibacterial activity. It was only two years later that Felix d'Herelle proposed the existence of phages, as virus that predate bacteria, after his observations of the formation of clear areas in bacterial cultures, when inoculated with a mixture containing phages[6;7]. He noted that the application of phages in patients with hemorrhagic dysentery correlated with disease clearance[8].

Phage therapy was pursued in the following years, being used, for example, by d'Herelle, to treat dysentery, cholera, or bubonic plague. Phages even started to be commercially produced in France and in the United States. However, due to the ambiguity in the effectiveness of these preparations and because antibiotics started to be commercially produced, phage therapy was abandoned in western countries. Only in some eastern european countries phage thereby was continued[6–8].

In Soviet Russia and Poland, several papers regarding phage therapy were written. In one paper, more than 30 000 children with bacterial dysentery disease were tested with *Shigella* phages. Almost half of the group did not receive treatment (control), while the rest were administered with phages. During the study period, the disease was about 4 times more prevalent in the control group[6]. Besides the human testing, clinical trials can be performed on animals[9]. One of the more prevalent studies reports the administration of mice with a lethal dose of *E. coli*. It was observed that injection of phages into the mice was more effective than the use of antibiotics[10]. Many phages specific to bacterial species have been reported in animal testing[11].

Even though these studies look promising, attention has only been given recently to phage therapy as an alternative to antibiotics. This is because, in the past, most studies failed to do basic methodologies, such as use of a control sample of the population. Phage therapy effect was not clear, since formulations were not properly prepared. This meant replication of results was hard to achieve and an appropriate phage for a target bacteria was not being carefully selected[6;9;12]. The recent attention is derived from the emergence of bacteria resistance to antibiotics. As this problem gets worse and attracts more mediatic attention, phages come up as an interesting and promising alternative.

In comparison to antibiotics, phages are specific to their target. So, unlike antibiotics, phages should not affect the normal bacterial flora of our organism[6]. In fact, there have been no reports of side effects from the administration of phages[6;11]. Phages replicate upon reaching the site of infection, increasing in number, as opposed to antibiotics that are often metabolized and eliminated. While development of resistance exists against phages, the effect is not as severe as in the case of antibiotic resistances. Contrary to antibiotics where the developing of new drugs can take many years and is costly[6], phage therapy could be used as cocktails, for example, selecting phages that are easily isolated from nature and that target the same bacterium using different receptors. This thesis proposes a solution to finding suitable phages for therapy.

2.2 What are bacteriophages?

Bacteriophages, (phages), are viruses that solely target and kill bacteria. They have a genome encapsulated in a protein shell, called the capsid. Although occurring very rarely, phages can be further surrounded by a lipidic layer, called the envelope[1]. The majority of phages have tails, structures that are essential in phage adsorption to bacteria. In terms of the genome, phages can have DNA or RNA, in their capsid, and it can be double or single stranded and even circular or linear[1;13]. The genome size varies from 3.5 to 500 kb[8]. Most phages contain enzymes needed to penetrate and inject the genetic material into the bacteria[1].

One way to classify phages is to divide them into tailed, polyhedral, filamentous or pleomorphic, with the latter three not possessing tails[8;13]. Tailed phages can be further classified several families, with the most important three being *Myoviridae* (long contracted tail), *Siphoviridae* (long non-contracted tail) and *Podoviridae* (shot non-contracted tail)[7;8;14]. This *Caudovirales* order is the most abundant, representing about 96% of all phages[13;14].

All tailed phages possess linear dsDNA with an icosahedral or elongated head. At the end of the tail, that is not connected to the head, there can be a baseplate structure to which tail fibres/spikes connect to. These structures can be nonexistent, but the fibres/spikes still exist, as they are required for phage-host interaction. The tail structures are always proteins and can have enzymatic activity[15].

Cubic or polyhedral phages, as said before, lack tail structures. In terms of their genome, they either have DNA or RNA, double or single stranded and circular or linear. They can have a lipidic capsid or even an envelope[13]. Filamentous phages are long filaments that can be enveloped. They possess either a circular ssDNA or a linear dsDNA. Pleomorphic is the viral group that contains the fewest number of known phages, all possessing linear or circular dsDNA[13].

2.3 Bacteriophage life cycles

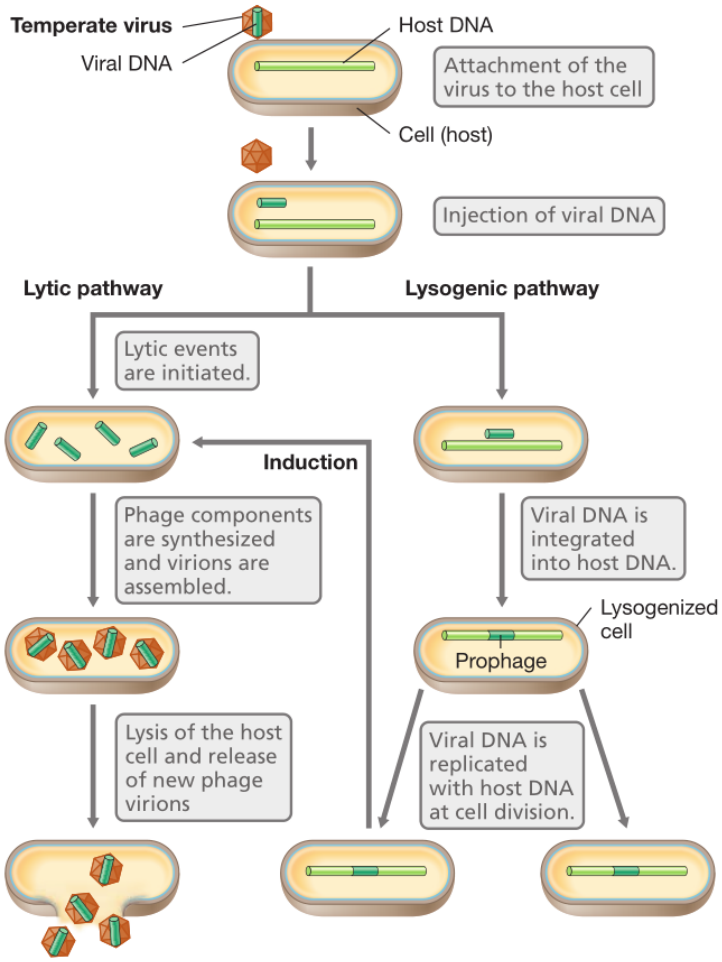


Figure 1.: Schematic representing both life cycles of phages. Image extracted from Madigan et al.[1]

In terms of life cycle, the phage can have a lytic or lysogenic life cycle (figure 1)[1;8]. In both cases, the phage requires a host cell to produce its viral components to successfully replicate. The general steps of infection are the same for both types of phages, namely adsorption, DNA injection, viral replication and cells lysis.

The adsorption, in most cases, involves the interaction of the phage tails with bacterial receptors, that can include surface proteins, polysaccharides, lipopolysaccharides and even pili or filaments[1;7;16]. Therefore, the adsorption step is crucial in defining the specificity of each phage.

The next step is injection of the genetic material into the host. This can be accomplished by enzymes present either in the tails components or in the capsid, that normally degrade the cell wall[1;7], exposing the membrane. Nobrega et al.[17] exemplifies enzymatic activities found in phages. During injection, only the genetic material enters the cytoplasm of the host. The capsid and remaining components stay outside the cell.

There is an important distinction between lytic and lysogenic phages. In the lytic (or virulent) phages, the bacterial metabolism is directed to synthesis of new viral particles, to be assembled into complete phages and released during lysis of the host. The lysogenic (or temperate) phages, however, remain in a resting state, called lysogeny, in which the viral genome is integrated into the bacterial DNA or remains in the cytoplasm, as a plasmid. In either case the viral genome is replicated in synchrony with the host cell, being then transmitted to all of the bacteria's progeny during cell division[1;7].

The next step is directed towards the synthesis of the nucleic acids and proteins required to assemble the virus, the assembly and packaging of the phage. In the case of temperate phages, this process only occurs when the cell is in a stress condition, in which the virus exits the lysogeny state and proceeds with the lytic pathway. For a given phage, several viral particles can be produced in a single cycle. The efficiency of this process is phage, host and environmental-dependent[1].

The lysis of the cells is the final step of the phage cycle. This event in timely synchronized by the expression of mostly two important proteins, the holin and the endolysin. After the new phage particles are properly assembled, the holin permeabilizes the bacterial cytoplasmic membrane and the endolysin degrades the peptidoglycan, thereby causing the burst of the cells and the release of the phage offspring[18].

2.4 Phage-host recognition

As mentioned, tailed phages use their tails to target bacteria, through the host receptors. This tail-host receptor interaction is what defines the host range and specificity of phages to their bacterial targets[4;17]. Successfully identifying these interactions can allow us to determine in advance, which phage to apply to a given bacterial infection/contamination.

2.4.1 Bacterial receptors

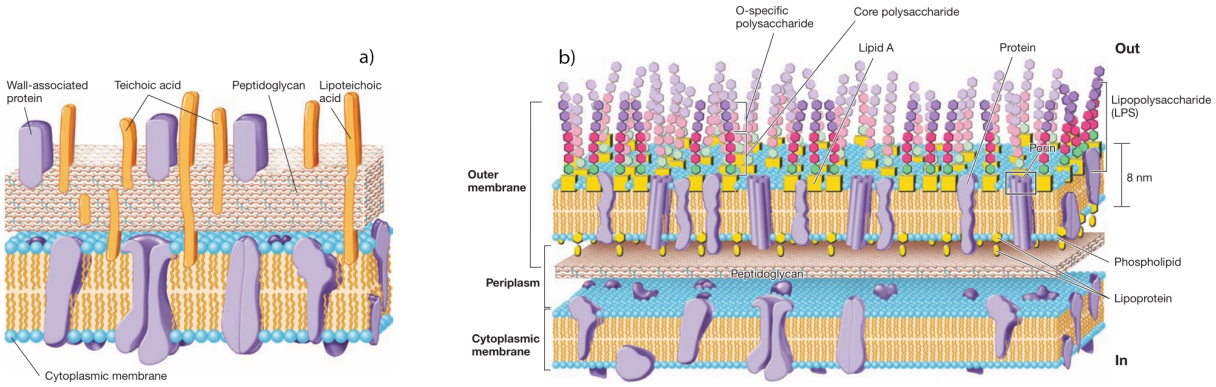


Figure 2.: Structure of gram-positive (a) and gram-negative(b) cell wall, membrane and its components. Image adapted from Madigan et al.[1]

The type of receptors vary considerably whether if phage is targeting a gram-positive or gram-negative bacterial host. In figure 2 the structures of cell wall, membrane and different receptors, for both bacterial types is represented. In gram-positive bacteria, the majority of receptors are found in the cell wall, for being the outermost component in these bacteria. The peptidoglycan itself can be targeted as a phage receptor[19]. Monteville et al.[20] described seven phages that adsorb to cell wall sugars, like glucose and rhamnose, and to the *N*-acetylglucosamine (NAG) of the cell wall itself. Gaidelyte et al.[21] described phage Bam35, able to adsorb to peptidoglycan *N*-acetylmuramic acid (NAM) of *Bacillus thuringiensis*. Several more examples are provided in table 1.

One of the most common receptors in gram-positive bacteria are the teichoic acids that can be attached directly to the peptidoglycan of the cell wall, or to constituents of the membrane[4;22;23]. It has been reported that bacteria having similar teichoic acid structure can be targeted by the same phages[4].

Teichoic acids are composed of a disaccharide connected to peptidoglycan via a phosphodiester bond (in the case of wall teichoic acids) and to a polyol repeat unit, also by a phosphodiester bond[22].

Several phages are known to adsorb to wall teichoic acids[19;24]. Xia et al.[25], for example, studied several phages that interacted with NAG bound to wall teichoic acids and one phage that interacted with the backbone of the teichoic acid itself. There have been reports of phages interacting with lipoteichoic acids, such as phage ϕSLT that targets *Staphylococcus aureus*[26]. Since the structure is similar to wall teichoic acids, phage proteins also bind to the same basic units. In the example given, phages interact with the repeating unit[26]. More examples are given in table 1.

There are a few phage receptors in gram-positive bacteria described, namely, protein YueB is used by phage SPP1 to recognize *Bacillus subtilis*[27;28] and, GamR, a LPXTG protein located in the surface of *Bacillus anthracis*, that is targeted by phage γ[29]. Phages present in the first line of table 1 have the ability to bind to a membrane protein, named PIP (phage infection protein)[20]. A list of all known bacterial protein receptors are listed in table 1.

Table 1.: List of known phage receptors found in gram-positive bacteria.

Phage	Host	Receptor	Reference
c2, ml3, 1, TP901-1, ϕLC3	*Lactococcus lactis*	Cell wall polysaccharides	Monteville et al.[20]; Ainsworth et al.[30]
A118	*Listeria monocytogenes*	Peptidoglycan and wall teichoic acids	Habann et al.[31]
A511	*L. monocytogenes*	Peptidoglycan	Wendlinger et al.[32]
Bam35	*B. thuringiensis*	NAM of peptidoglycan	Gaidelyte et al.[21]
ϕ29, ϕ25, SP01	*B. subtilis*	Glycosylated wall teichoic acid	Sao-Jose et al.[27]; Yasbin et al.[33]; Xiang et al.[34]
ϕ11, ϕ47, ϕSa2mw	*S. aureus*	NAG of wall teichoic acid	Xia et al.[25]
ϕ11	*S. aureus*	O-acetyl of cell wall NAM	Li et al.[35]
ϕ812	*S. aureus*	Wall teichoic acid backbone	Xia et al.[25]

Continued on next page

Table 1: continued from previous page

Phage	Host	Receptor	Reference
ϕ29	*B. subtilis*	Wall teichoic acids	Xiang et al.[34]
ϕSLT	*S. aureus*	Lipoteichoic acids repeat unit	Kaneko et al.[26]
LL-H	*Lactobacillus delbrueckii*	Glucose and glycerol of lipoteichoic acids	Munsch-Alatossava and Alatossava[36]
SPP1	*B. subtilis*	Protein YueB	Sao-Jose et al.[27]; Baptista et al.[28]
γ	*B. anthracis*	Protein GamR (LPXTG)	Davison et al.[29]
c2, ml3, kh, l, 1	*L. lactis*	PIP	Monteville et al.[20]

In gram-negative bacteria, the structure and composition of the cell wall and membrane is different from their gram-positive counterparts. These have an external membrane mainly composed of lipopolysaccharide (LPS) and proteins, followed by a thin peptidoglycan cell wall and an inner membrane[19;37]. These bacteria are highly permeable, possessing a high number of transmembrane proteins that form channels. These proteins can be used as receptors for phages. The different LPS sites can be used in phage interaction with host[4;23]. Unlike gram-positive bacteria, several gram-negative's phage receptors have been identified, being either proteins or saccharides[19].

As the name indicates, LPS are composed of saccharides and fatty acids. They have a core composed of lipid A, that is a disaccharide of glucosamines with phosphate groups attached, connected to the core polysaccharide. This core can bind to O-antigen, with both of these containing several carbohydrate units[19;38]. Phages are highly specific to LPS receptors containing the O-chain (also called smooth LPS), displaying a narrow host range. LPS containing only the core and not the O-chain (called rough LPS) can be recognized by phages, but, in this case, the host range is more broad, since the core is more conserved[4;19]. Phages T7 and SSU5, for example, bind with their tail fibres to rough LPS of *Escherichia coli* and bacterial species of the genus *Shigella* and *Salmonella*[39;40]. More examples of phages binding to LPS are present in table 2.

The proteins located in the outer membrane can be transport proteins, channels, like porins, or structural proteins that interact with the cell wall. All of these are exposed to the exterior and can be used

in phage interaction, as several examples have been reported[4]. Phage T4, for example, has shown the ability to adsorb to OmpC (outer membrane protein C) of *E. coli* (as well as smooth LPS)[41]. OmpC from *S. enterica* are targeted by phage S16[42]. Bacteriophage Yep-ϕ has shown the ability to bind to outer membrane proteins Ail and OmpF (and to rough LPS) of *Y. pestis*[43]. Other examples of phages recognizing protein receptors include: phage SPC35, which interacts with BtuB (cobalamin outer membrane transporter) from *S. enterica*[44]; phage λ use protein LamB (a maltoporin) of *E. coli*[45]; phage H8 initially adsorbs to FepA (an outer membrane receptor) and then to TonB (a transport protein) of both *E. coli* and *S. enterica*[46]. A more complete list of phages and bacterial proteins they adsorb to is present in table 2.

Besides the most obvious receptors found in either gram-positive or gram-negative bacteria, phages can adsorb to receptors found in capsules, pili, flagella and other filaments[17]. In these cases, however, the phage does not release the genetic material upon receptor recognition. Rather, it moves until the bacterial inner surface is reached[4]. There have been reports of phages that attach themselves to flagella with the head, leaving the tails free. This way, upon reaching the surface, the tails are used for adsorption[47]. There are reports on phages that bind via their tails to the flagella[17].

Phages ϕCb13 and ϕCbK interact with proteins in the flagellum of *Caulobacter crescentus*[48]. Bae and Cho[49] reported phages MPK7, LKA1, ϕKMV, able to bind to TFP (type IV pili) of *Pseudomonas aeruginosa*. Several *Acinetobacter* and *Klebsiella* phages are reported to recognize their hosts through their capsular polysaccharides[50;51]. Table 2 reports examples of phages interactions to gram-negative receptors.

Table 2.: List of known phage receptors found in gram-negative bacteria.

Phage	Host	Receptor	Reference
T7, SSU5	*E. coli*, *Shigella* spp, *Salmonella* spp	Rough LPS	González-García et al.[39]; Kim et al.[40]
M1	*E. coli*	OmpA	Hashemolhosseine et al.[52]
T4	*E. coli*	OmpC, smooth LPS	Washizaki et al.[41]
S16	*S. enterica*	OmpC	Marti et al.[42]
Yep-ϕ	*Yersinia pestis*	Ail, OmpF, rough LPS	Zhao et al.[43]
SPC35	*S. enterica*	BtuB	Kim and Ryu[44]

Continued on next page

Table 2: continued from previous page

Phage	Host	Receptor	Reference
BF23	*E. coli*	BtuB	Bradbeer et al.[53]
λ, K10	*E. coli*	LamB	Gehring et al.[45]; Roa[54]
H8	*E. coli*, *S. enterica*	FepA, TonB	Rabsch et al.[46]
ST27, ST29, ST35	*S. enterica*	TolC	Ricci and Piddock[55]
T5, ES18	*E. coli*, *Salmonella*	FhuA	Breyton et al.[56]
9NA, P22, Det7	*S. enterica*	Smooth LPS	Andres et al.[57]; Perez et al.[58]; Walter et al.[59]
Mu, P1, D108	*Erwinia* spp, *E. coli*	Smooth LPS	Sandulache et al.[60]
Sf6	*S. exneri*	Smooth LPS	Parent et al.[61]
T6	*E. coli*	Tsx	Manning and Reeves[62]
T3	*E. coli*	Smooth LPS	Prehm et al.[63]
N4	*E. coli*	NfrA	McPartland and Rothman-Denes[64]
JG004	*P. aeruginosa*	Smooth LPS	Le et al.[65]
ϕCR30	*C. crescentus*	Paracrystalline surface layer protein	Edwards and Smit[66]
JG004, ϕCTX	*P. aeruginosa*	Core of LPS	Garbe et al.[67]; Yokota et al.[68]
ES18	*S. enterica*	FhuA	Killmann et al.[69]
L-413C, P2 *vir1*, ϕJA1, T7	*Y. pestis*	LPS core	Filippov et al.[70]
ω8	*S. enterica*	FhuA	Killmann et al.[69]
ϕCb13, ϕCbK	*C. crescentus*	Flagellin	Guerrero-Ferreira et al.[48]
iEPS5	*S. enterica*	Flagellin	Choi et al.[71]
PBP1	*B. pumilus*	Flagellin	Lovett[72]
ϕAcS2, ϕAcM4	*Asticcacaulis biprosthecum*	Flagellin	Pate et al.[47]
7-7-1	*Rhizobium lupini*	Flagellin	Lotz et al.[73]

Continued on next page

Table 2: continued from previous page

Phage	Host	Receptor	Reference
ϕAcM2	*A. biprosthecum*	Pili	Pate et al.[47]
MPK7, LKA1, ϕKMV	*P. aeruginosa*	TFP (type IV pili)	Bae and Cho[49]
K2	*K. aerogenes*	Capsular polysaccharides	Yurewicz et al.[74]
ViO1, o507-KN2-1	*S. enterica*, *K. pneumoniae*	Capsular Vi antigen	Pickard et al.[75]; Hsu et al.[76]
ϕK1-5	*E. coli*	Capsular polysaccharides	Scholl et al.[77]

2.4.2 Bacteriophage tails

As this thesis focuses on tailed phages, only mechanisms of adsorption from tails of the most known phage groups will be mentioned. *Myoviridae* are the only phages with contractile tails. *Siphoviridae* have long non-contractile tails, while *Podoviridae* have short tails (figure 3)[13;17]. These protein tails are essential in recognizing the bacterial host receptors described in the previous section.

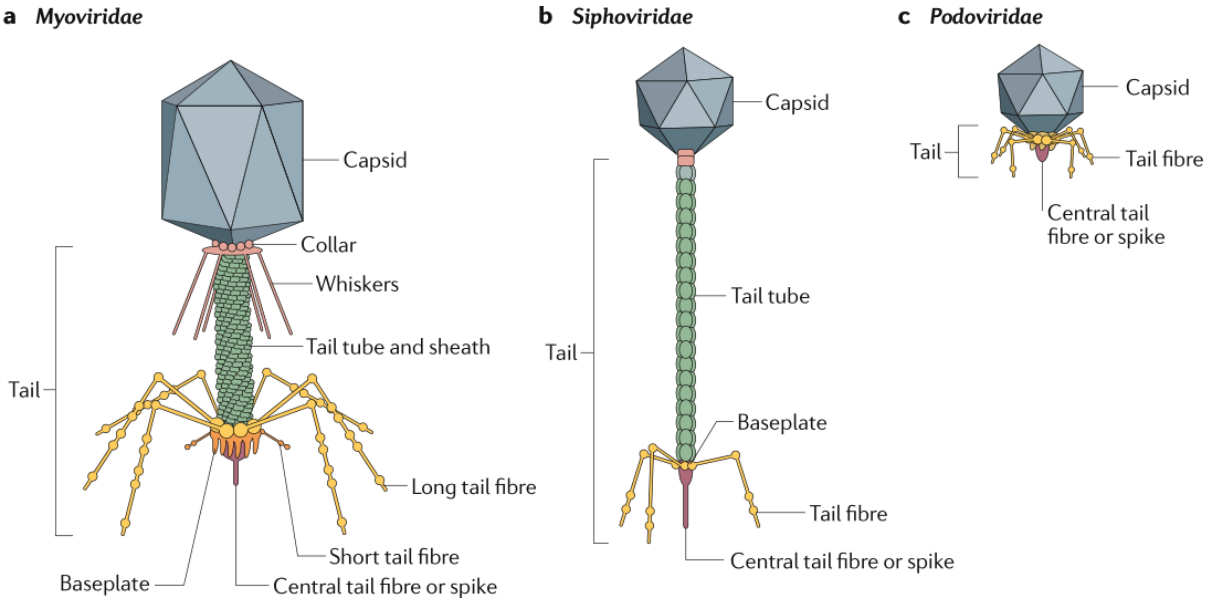


Figure 3.: Representation of the three families of tailed phages. Image extracted from Nobrega et al.[17]

Since *Myoviridae* have contractile tails, this is the most complex family of the three. In their structure, myophages possess a tail tube, attached to the capsid and to the baseplate complex, which is the structure to which tail fibres and tails attach to. This tube is enveloped in a sheath that allows contraction to happen, allowing viral DNA to be expelled into the bacterial host. When bacterial receptors interact with tail fibres and spike, the virus contracts and infects the host[17]. Depending on the bacterial receptor type, phages of this family can have different baseplate structures. If the receptor is a protein, phages have conical tail tips, but if the receptors are polysaccharides, the tail tip will generally be a complex baseplate[15].

A well studied myophage, bacteriophage T4 (phage that infects the gram-negative *E. coli*), containing long and short tail fibres in the baseplate, begins by interacting with the bacterial hosts via the longer tail fibres. Each of these binds itself to the host receptor, reversibly, ultimately finding an optimal position. When two or three of these are bound to bacterial receptors simultaneously, the baseplate changes its conformation in order for the short tail fibres to attach irreversibly to the host. After this, the baseplate suffers further conformational changes and the sheath contracts. Consequently, the tail tube penetrates the bacterial membrane and injects the viral DNA into the bacteria[17].

Siphoviridae, on the other hand have long non-contractile tails and therefore, lack the sheath. Siphophages have been reported to have a simpler baseplate structure (when compared to the myophages), or can even lack this structure. Siphophages that target gram-positive bacteria have the baseplate, while phages targeting gram-negative have a simpler structure or even lack it. Like *Myoviridae*, depending on the type of host receptor, the baseplate can be more complex, or have a conical shape. Further than this, however, siphophages, infecting gram-positive bacteria, which, interact with protein receptors have a tail fibre or spike. Phages that interact with polysaccharide receptors of gram-positive bacteria have side fibres in their baseplate. Siphophages that infect gram-negative bacteria have a more general structure and can contain tail spike and tail fibres[15;17].

Phage T5, for example targets protein receptors of gram-negative bacteria. Therefore, it has three side tail fibres, which increase adsorption to the host. This phage possesses a central tail spike that contains one receptor binding protein specific for a transmembrane transporter[17]. The siphophage p2 infects gram-positive bacteria and targets polysaccharides from the host. As a result, it has six side tail fibres, each containing three receptors specific for the host's polysaccharide. This phage initially connects just one receptor in each of the tail fibres. This causes a conformational change that alters the structure of the virus, enabling all the receptors to bind to the host in a irreversible way[15;17].

The last group of the tailed phages family is *Podoviridae*. Podophages are composed of a short non-contractile tail, lacking a sheath. The general structure is similar across all podophages. The lower part of the tail can have six or twelve tail fibres or spikes, but it lacks a baseplate, that was observed in most cases of the previous two groups. Each phage belonging to this group has a similar mechanism of adsorption and infection. Their tail spikes or fibres interact with the host's receptors to cause either a conformational change that allows DNA to be released or have enzymes that allow penetration of the host and release of the genomic information [15;17]

2.5 Bacterial resistance and phage survival

Like it happens with antibiotics, bacteria have the ability to develop resistance in order to survive/avoid phage infection. One of the advantages of phage therapy to antibiotics is that phages can mutate to overcome these resistances. In this section, only resistance mechanisms at receptor level will be mentioned, since they are the only ones relevant for this work. Labrie et al. [78] described in more detail mechanisms associated with blocking DNA entry, DNA degradation, which will not be mentioned here.

The first level of bacterial defense against phages is at the receptor level. Bacteria can build physical or chemical barriers that block the adsorption of phages. One of the mechanisms observed is the direct blocking of phage receptors. This can be achieved by changes to the structure of the receptors, such as binding of molecules, or changes in the tri-dimensional conformation of the receptor. One example is the production of protein A by *S. aureus*, that binds to the bacterial receptors and blocks phage access [79]. Nonetheless, phages are able to mutate their tail receptors, which allows them to recognize new receptors. From these mutations, phages might even be able to interact with more than one bacterial receptor, which is advantageous, for example, when bacteria repress the production of one receptor, phages will still be able to infect the host via the other receptor [16;78].

Another mechanism at the receptor level is the production of an extracellular matrix by bacteria. This matrix is a layer of polymers that physically block the phage from accessing the receptors. Phages can develop to recognize this matrix and can even produce enzymes to degrade it. *Pseudomonas* species produce a matrix called alginate. The species producing it was found to be phage resistant. However, a

phage was found to have developed alginate lyase that dispersed the matrix and allowed phage-bacteria interaction[80].

2.6 Computational approaches for prediction of phage-bacteria interactions

Discovery of interactions between two cellular structures has typically been studied in laboratories. Usually these processes take a long time and numerous resources. Using computational approaches to predict these interactions, has the advantage of being faster, more automatized and requiring less resources. Obviously, experimental data is required to build these tools. There is a large amount of information available, enabling its development, with satisfactory results.

Relevant databases are available containing information on interactions between phages and its hosts and can be used to develop these predictive tools. The Virus-Host Database[81] has 4014 sequenced phages, with sequenced host species defined; however, only seldom does this database specifies the strain. Zhang et al.[82] manually verified published articles for phage and their binding receptors. PhageReceptor provides a high-quality database for experimentally proven receptors phages adsorb to, further dividing them into receptor type, while, not specifying host strain. PhagesDB[83] contains information on mycobacteriophages, including sequence and host strain information. The National Center for Biotechnology Information (NCBI) Virus database[84] is probably the most complete and thorough. It provides a collection of phages, available in the GenBank[85], with genomic information available and other relevant features.

Many tools have been developed and made available to predict interaction between phages and hosts. These can be based on genomic sequence similarity, for example, or interactions between phage proteins and bacterial proteins. These approaches may use machine learning (ML) algorithms for prediction. Regarding interactions between phage proteins and bacterial carbohydrates, it is more complicated to develop an algorithm as these interactions are more dynamic since they are weak[86]. Computational approaches for these interactions have been described, although most of them are too specific and not applicable in the context of the present thesis. The most relevant algorithms will be mentioned in this section.

2.6.1 Genome sequence-based approaches

A work by Edwards et al.[87] reviewed and evaluated several computational based approaches to predict phage-host interactions. Most of these were based on the similarity of sequences between the interacting pair. One of the approaches is based on abundance profiles. As viruses depend on their bacterial hosts to replicate, it is expected that the number of phages increase with a higher number of their hosts. Although previous works already reported this condition[88;89], only Edwards et al.[87] tried to properly use this premise to predict and evaluate a phage's host. However, only 12% of correct predictions were obtained, when classifying interactions at the species level. This method does not seem ideal, because it appears to have a really low predictive power and it requires knowledge of the whole metagenome. This means it is not possible to identify a host for any given phage as they must coexist and have been described in a metagenome, for example.

Other methodologies evaluated by Edwards et al.[87] are based on genomic sequences. A brief summary is shown in table 3. The first one is genetic homology. It is self-explanatory, as only homology between whole genomic sequences of phage and bacteria is sought. It is expected that interacting pairs have similarity regions, as phages may insert their genes into the bacterial genome[1], while phages can also incorporate bacterial genes into their genome, providing competitive advantage or even increasing host range[90;91]. Edwards et al.[87] carried out BLASTn (nucleotide-nucleotide) and BLASTx (translated nucleotide-protein), to test this methodology. Basic Local Alignment Search Tool (BLAST) is a tool that does sequence comparisons, returning a score for each pair[92]. Using these tools, better results for classification at bacterial species level were observed, with BLASTn, at about 30% of correct hosts predicted. For BLASTx similar results were reported. However, the number of false positives was much higher[87].

Table 3.: Computational genome based approaches to predict phage-host interactions[87]. Code and datasets used are present in http://edwards.sdsu.edu/PhageHosts/

Name	Description	Advantages	Disadvantages
Genetic homology	Looks for homology between genomic sequences of phage and bacteria	–	Not precise, not applicable to every phage

Continued on next page

Table 3: continued from previous page

Name	Description	Advantages	Disadvantages
CRISPR based	The CRISPR system incorporates viral DNA, allowing for homology search	Specific for each pair	Applicable to a low number of cases
Exact matches	Based on CRISPR systems, conserved regions and integrated prophages	Also specific, while being applicable to a larger number of pairs	—
Oligonucleotide profiles	κ-mer profiles are created and compared	Applicable to every pair	Assumptions can be verified by chance, lowest percentage of correctly predicted pairs

CRISPR (Clustered Regularly Interspaced Short Palindromic Repeats) and Cas (CRISPR associated proteins) system based approaches were used for prediction. These are used as a mechanism of immunity for bacteria against phages. Basically, the CRISPR system incorporates viral DNA and upon a recurring infection, the recognition of this sequence leads to a production of Cas proteins, that conjugate with the CRISPR transcribed RNA sequence. This complex binds to the phage genome and cleaves it[8;16]. Therefore, in this sequence there should be an identifiable high similarity with the phage genome. On this basis, Edwards et al.[87] aligned the CRISPR spacer sequences to phage genomes with BLASTn, adapting it to enable the use of much shorter sequences. The results look encouraging, as they are specific. However, as pointed out by the authors, there are few significant hits. Furthermore, not all bacteria encode the CRISPR-Cas system (around 50%), making them unavailable for phage prediction[93].

Another methodology was based on exact matches between phage and bacterial genome. These include the CRISPR systems, conserved regions and integrated prophages (temperate phages). This approach is sensitive, specially for longer exact matches. Despite, only 40% of correct classification of bacterial species was obtained. This was the best performance of all tested methodologies[87].

The last approach was formulated on oligonucleotide profiles, in which all the bacterial genome is fragmented into k-mers of a defined length. A profile is created as a vector containing the relative frequency of all k-mers. Two cases were created, in which length was 1 and 3, corresponding, respectively, to GC

content and codon usage. In every case, Euclidean distance between the phage's and bacteria's profile was used for classification. For the best case, the classification was correct on only 17% of cases[87].

These methods described by Edwards et al.[87] have the disadvantage that most limit the number of organisms that can be analysed. CRISPR classification, for instance, is limited to bacteria that enclose this system. On the other hand, oligonucleotide profile classification is limited to lysogenic phages. Besides this, classification was performed on species level, at most, with results that leave much to be desired.

HostPhinder[94] is a tool that performs prediction of host bacteria against a phage, at species level. It is based on genomic similarity of a query phage against similar already described phages. Basically, the assumption is that similar phages infect the same bacterial species. This was performed by calculating k-mers on the query, and comparing them with the reference phages, calculating a similarity score. In the end, a 74% of correct predictions was obtained for classification at species level, which is a vast improvement comparing with the previously mentioned work.

Another work by Gałan et al.[95] used ML to distinguish between eukaryotic viruses from phages. This tool based its calculations on mono, di and trinucleotide frequencies in the viral genome, classifying the virus as eukaryotic or bacteriophage. The best results obtained were of 95% correct predictions. However, this classification is quite simple, for the problem proposed here.

VirHostMatcher-Net[96] is a tool that predicts virus and host interactions, for metagenomes. It is based on genome sequences, by calculating features such as CRISPR sequences, homology and alignment scores. The best predictive score was of 85%, observed at genus level (above species), at most. Once again, although the results seem promising, it is not as specific as this work intends to be.

PHISDetector[97] uses genetic features, as well as protein-protein interaction information. For the purpose of consistency, the whole of this tool will be mentioned in this section. A correct classification of 85% of interactions, at bacterial species level, was achieved. CRISPR features were constructed by predicting CRISPR spacers in bacterial genomes. These spacers are aligned against phage genomes, to find hits, retrieving features related to these. A prophage analysis was also performed, in which, a prophage was identified in a bacterial genome, genomic and protein sequences were retrieved, ran against the phage genome and proteome, for homology scores. Simple genomic sequence similarity scores were calculated through k-mer based distance and similarity metrics. The protein analysis was performed for homologous proteins between the pair, in which interaction scores were retrieved. For this work the results appear

satisfactory, however, basing the predictions solely on similarity may limit the search space. As will be discussed in the next section, protein-protein interaction may not be the most optimal way to approach this problem. Furthermore, the classification is performed on species level, less specific than the proposed work.

These methods have the disadvantage that most limit the number of organisms that can be analysed. CRISPR classification, for instance, is limited to bacteria that enclose this system. Therefore, in the next sections, alternative computational approaches for predicting interactions between phage and host receptors will be discussed.

2.6.2 Protein-protein interactions

The ubiquitous phage-bacteria proteins' interactions are a relevant advantage when developing a computational approach. Unlike the genome based approaches in which there were differences between species that hindered the classification, every phage possesses an adsorption step. Furthermore, phages are composed of proteins that interact with the host. Nevertheless, only two works are known to have used protein-protein interactions for classification [98;99], besides the work mentioned in the previous section [97].

In the work of Leite et al. [98] ML algorithms to predict phage-bacteria pairs were developed. Firstly, 1064 phage sequences from the GenBank and PhageDB databases were extracted. In this process, every necessary information was checked to ascertain if it was complete, namely, phage gene information, host information and protein sequences. If either was missing or could not to be determined, the phage would be excluded from the dataset. Regarding the respective bacterial host, these needed complete gene and protein information, otherwise the host and respective phage would not be included. In the end, the dataset was composed of the mentioned 1064 phage-bacteria pairs. However, 915 of them corresponded to the same bacterial host, which is not ideal when running ML algorithms. To attenuate this effect, the remaining pairs were oversampled by replicating the interactions around 300 times. This methodology might not be ideal for ML, because it can lead to overfitting, where the model learns the examples far too well [100]. A negative interaction dataset was created, composed of phage-bacteria sets that theoretically do not interact with each other. However, the creation of this dataset might be hard to achieve. The number of negative interactions would be far higher than the number of positive ones, since the number of bacteria

a phage does not interact with is higher than the number of interacting hosts. Given this fact, this negative dataset would have a higher dimension. The same number of positive and negative interactions were maintained.

Having extracted all the information, the features to be used in the algorithms were defined. To study protein-protein interactions, datasets based on interactions between domains and another based on protein primary structure from the interacting pairs were created. From the domain interaction dataset, for each, protein pair, all the domain-domain interaction scores were obtained. The sum of all these scores, would correspond to the protein-protein interaction score. Therefore, each phage-host pair would be described by a vector of protein-protein interactions (Protein-Protein Interaction). The data was normalized to be applied in ML. For the remaining dataset, protein primary structure information was used, including amino acid composition, chemical elements composition and molecular weight. For each protein-protein interaction, mean and standard deviation were calculated for all features[98].

Every dataset created, was tested with four ML algorithms, K-Nearest Neighbours (KNN), Random Forests (RF), Support Vector Machines (SVM) and Artificial Neural Networks (ANN), with cross-validation. It was noticed that KNN and RF lead to overfitting, reflected on high prediction values. For the other two algorithms, overfitting is not mentioned, but it is still a possibility, as the overall the prediction values seem quite good.

Even though the process described by Leite et al.[98] might not be robust, it is indeed promising. It could be improved by, for instance, creating a dataset containing more bacterial species. For this, there is a need to explore a higher number of databases to find more information. Nevertheless, the approaches for obtaining protein-protein interaction scores seem to work well.

In the work of Boeckaerts[99], ML methods were used to predict phage-host interactions, at a species level. To build the dataset, phage tail spikes and fibres protein sequences were extracted, from UniProt KB[101]. A total of 425 phage proteins were obtained from UniProt[99]. For each protein, the respective bacterial host information was added, if available. Each protein was characterized by phage and bacterial host information and protein and DNA sequences information. The dataset was further filtered to 330 instances, thus ML algorithms would have to classify only three different outputs (only three bacterial species were considered)[99].

The next step was feature construction, from the dataset. As reported by Edwards et al.[87], features were built based on DNA sequences. These included frequency of each nucleotide, GC-content and codon frequency and usage bias. Besides DNA based, protein features were also considered, such as amino acid frequency, molecular weight, aromaticity, among others, and protein secondary structure features (a more complete list is present in the thesis[99]).

Four ML algorithms were used, logistic regression, linear discriminant analysis, random forests and gradient boosting. A four-fold cross-validation and hyperparameter optimization were used to build the models. For all the models, a score above 88% was observed, with the best performing model being linear discriminant analysis[99].

The protein interaction pairs for classification of phage-bacteria pairs appear to be promising. Methods based on these pairs are used to classify phages for a given bacteria well. However, it would be interesting to include interactions between phage proteins and bacterial carbohydrates, as interactions may involve these two biomolecules. No work was found, to specifically predict these interactions, based on phage-bacteria interactions. In the next section, approaches using protein-carbohydrate interactions are mentioned. Although these mostly refer to specific carbohydrates or to enzymes that do not have interest to this thesis, the basis of each approach can be applied in a new methodology to classify phage-host pairs.

2.6.3 Protein-carbohydrate interactions

Through the same logic as in protein-protein interactions, in the adsorption step, phage proteins also interact with external carbohydrates from bacteria. These types of interactions are specific and predictions should not rely on many assumptions, as is the case for genome-based predictions. However, to the best of our knowledge, no work specific for phage-bacteria interactions is available. In the next section, more general works predicting interactions between proteins and proteins or carbohydrates will be discussed.

Table 4.: Computational methods for prediction of protein-protein and protein-carbohydrate interactions.

	Name/Reference	Description
3D structure	Taroni et al.[103]	Prediction of glycan binding sites
	Shionyu-Mitsuyama et al.[104]	Spatial distribution of protein atoms around carb-binding sites
	COTRAN[105]	Looks for galactose-binding sites. Uses geometrical and structural features
	SPOT-CBP[105]	Based on existing 3D structures of proteins that interact with carbohydrates
	ISMBLab[106]	Based on the distribution of proteins in the surface of carbohydrates
	InCa-SiteFinder[107]	Uses van der Waals forces, carbohydrates and amino acid propensities
	MaSIF[108]	Based on a representation of the molecular surface. For every point, geometric and chemical features are calculated
Sequence	PROCARB[109]	Finds carb-binding residues from single sequences or evolutionary profiles
	PreMieR[110]	Uses binary and PSSM, composition profiles
	MOWGLI[111]	Predicts mannose-binding proteins based on PSSM profiles
	SPRINT-CBH[112]	Based on sequences in the interaction pockets

2.6.4 Protein interaction methods not specific to phages

The methods described in this section are not used for phage-bacteria interactions. However, these could provide insights into how to perform predictions for this work. In table 4, a list and short description of these tools is present. A separation can be made for these tools into methods that rely on tri-dimensional structures and sequences for prediction. Malik et al.[102] did a review that covers these methodologies, for protein-carbohydrate interactions.

Methods based on the tri-dimensional structure, limit the possibility of applicable cases, because there is a limited number of receptor binding protein structures from *Caudovirales* available. Searching the Protein Data Bank PDB[113] (a database of biomolecular structures) for phage tail proteins, 11 entries were retrieved, even though not all are likely to be relevant (figure 4 (A)). A query on the UniProt KB[101] for phage tails, retrieved 48 entries with structures available, with possible overlap with the 11 from PDB (figure 4 (B)). However, these can provide good insight and alternative to how proteins bind to bacterial proteins or carbohydrates, if available.

Works, based on structures, highlighted that features such as charged residues that contact with the polysaccharide, hydrophobicity, or interacting forces between protein and carbohydrate can be important in

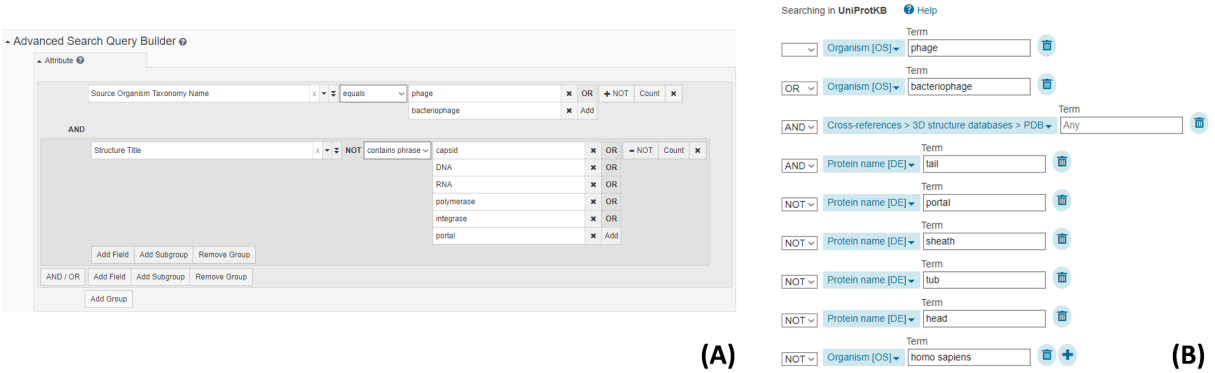


Figure 4.: Queries made on PDB (A) and UniProt (B), for phage tails. A search was performed for organisms containing bacteriophages mentions, while removing entries that mentioned unwanted functions.

the ML algorithms tested[114;115]. These two methods, even though the performance seemed satisfactory, were built on small datasets.

Another example of using tri-dimensional structures for prediction was the work developed by Taroni et al.[103]. Here, three parameters (relative accessible surface area, protrusion index and residue propensity) were found to best predict glycan binding sites, with and accuracy of 65%. However, this work is quite dated, with mixed results.

Other methods based on the spatial arrangement of the proteins were developed[104;105;116]. These all use the protein structure to predict if binding to polysaccharides occurs. The first one[104] is probably the most interesting for this study, since it evaluates different parameters for prediction. The second one[105] is not as helpful or applicable to many cases, since it is based on detecting proteins interacting only with galactose. The last[116] predicts the structure of a protein, based on already known arrangements of proteins that are known to interact with carbohydrates. It is available online with the name SPOT-CBP. One issue with this method is that it requires the knowledge of the tri-dimensional structure, coupled with knowledge of the way a carbohydrate interacts with a protein. In the case of phage-host interactions this method would not be feasible, as there is no known related work.

ISMBLab[106;117] predicts the binding of a variety of ligands to protein structures. A ML model was constructed for 30 protein atoms, based on the physicochemical characteristics of superficial atoms. It predicts the probability of each atom being involved in binding to the specified ligand. Besides only predicting probabilities for single atoms, instead of the whole protein, it is a very lengthy process that performs predictions for just one protein at a time. Using this for multiple structures, would not be feasible, since it requires lag times and manual uploading of each structure.

Another similar method is InCa-SiteFinder[107], that uses van der Waals forces, carbohydrates and amino acid propensities to identify carbohydrate binding sites. The authors reported good specificity and sensitivity scores. However, this tool is not available (as of 17/02/21).

MaSIF[108] is based on a representation of the molecular surface of proteins. For every point of this surface, geometric and chemical features are calculated and applied to a deep learning algorithm. For protein-protein interactions it outputs a score for each vertex that corresponds to the probability of being involved in an interaction. The authors have reported a supposedly better score than other available tools.

Overall, it does not seem feasible to apply tri-dimensional structures for prediction of interactions, due to the low number of available phage structures. A very high number of structures would have to be constructed through homology modelling of the already limited number available. Besides, most of these methods focus on predicting general interaction sites for each protein, when the preferred goal would be to predict interactions with certain proteins or carbohydrates.

Malik et al.[102] also described methods based on sequence information. Even though there are methods described for only one type of carbohydrate, the way the datasets and algorithms were set up can give an idea on how to apply ML to a larger variety of proteins interacting with carbohydrates. Works, based on structures, highlighted that features such as charged residues that contact with the polysaccharide, hydrophobicity, or interacting forces between protein and carbohydrate can be important in the ML algorithms tested[114;115]. These two methods, even though the performance seemed satisfactory, were built on small datasets.

Other ML methods based on protein sequences were developed. These are mostly based on sequence conservation, on the basis that binding sites can not suffer several mutations, or they will risk losing function. On this note, ML algorithms have used Hidden Markov Models (HMM)[118] or Position-Specific Iterative BLAST (PSI-BLAST)[119] to detect homologies. A database containing known and predicted protein binding carbohydrates, was developed and is available online with the name PROCARB[109]. Besides this database, the authors developed a prediction tool for protein binding carbohydrates, CBS-Pred. Unfortunately, this tool does not seem to be available, at the moment of writing. Only the database seems to be accessible. Malik et al.[102] describe methods mostly for application to specific carbohydrates. Tools are also available in the form of web-servers, such as PreMieR[110] and MOWGLI[111]. There is a method that

uses both sequence and structure information for prediction[112] and it is available and functional online, with the name SPRINT-CBH.

A method to describe protein-protein interactions, named Conjoint Triad (CT), was described by Wang et al.[120]. It is simple, in the sense that it only requires protein sequences and is reported to have very good predictive power. Briefly, each amino acid is assigned into one of seven groups, based on dipoles and volumes of side chains. A sequence is then represented by seven different characters. With this, different features are calculated, such as, k-mers of size two, local descriptor, in which ten subsequences are created and analysed for composition, transition and distribution. Even though a high number of features are created, this method is based on a simple and easily applicable premise, with desirable results.

Overall, sequence based methods seem to be the best solution to apply, as of right now. However, should more information be made available relating to protein structure, it would certainly be advantageous to use this data for more accurate prediction and characterization of protein interactions.

2.7 Machine learning (ML)

Machine learning (ML) algorithms solve problems by finding relationships in the data. These algorithms adapt and improve with experience, that is, with new data inputted, they make/improve on relationships[121;122]. ML algorithms are to be applied to situations where a large amount of data has unknown patters that need to be discovered, when a problem is not well known or when there is a problem that needs to adapt to ever-changing conditions. The following concepts are essential better to understand these algorithms[121;123].

- **Dataset** is a collection of data in the form of a table. Usually, rows represent the different samples/instances and the columns represent its attributes. One of these attributes can be used as the output for ML.
- **Instances** correspond to observations of the data and are characterized by a set of features.
- **Attributes** or **features** are what describe the samples, and can be continuous, categorical or binary.

- A **model** describes/summarizes the data learnt and is used for prediction based on that data. It finds patterns in data and tries to predict an output for another data set based on those patterns.
- **Classifier** is the function that assigns a test data to an output.
- **Algorithms** are mathematical/logical instructions that create the best model with the training data. There are a multitude of algorithms described later on in the text.
- **Sample** is a collection of instances, with the respective features.

2.7.1 Development of an ML algorithm

Although there are different types of algorithms, all have the same general development steps. The first step is collecting data and preparation for application on an ML algorithm, as instances. Data collection is not straightforward, as good knowledge of the problem to solve is required, to select the data useful for solving the problem. Selecting a large amount of data, will most likely bring disadvantages, since the algorithm will base its learning on useless data for the problem in question[121].

Care is needed when aggregating data as each attribute might need to be standardized according to factors, meaning specific values should not be considered from absolute values, but relatively to another factor[123]. Other initial processes involve cleaning the data to remove/replace missing data and can require data to be normalized, discretized or categorized. Defining the format of the dataset is important, for it to be easily accessible.

After the dataset is processed, ML can be applied by dividing the data into train, test datasets. The train dataset will likely contain the largest portion of the original dataset and is used to build the model through ML algorithms. Precautions should be taken when constructing this dataset; for instance, checking wether data is balanced, preventing the over-representation of one type of attribute. The division in train and test datasets can be performed by sampling techniques, like cross-validation, or leave-one-out, for datasets with a small number of instances. These methods help in improving the accuracy. The remaining test dataset is used to evaluate if the model created has a good performance. From these test results, the algorithm or dataset is then tweaked to find the best predictive model that will finally be applied to predict other cases[121].

2.7.2 Types of ML algorithms

ML algorithms can be supervised, unsupervised, semi-supervised, or reinforcement learning. In supervised learning, the dataset contains the attributes and labels for each example, in which labels are a feature subject to the predicting algorithms. The model is learns how this labelled training data determines the relations between the vector of features (input) and the output feature. With this model, an external vector of features with unknown output can validate this instance. Overall, these algorithms aim at minimizing errors; thus, overfitting can become a relevant problem (explained in a later section)[121].

Unlike the previous algorithm, in unsupervised learning the data does not contain labels. It is used mainly to find relations in data; hence, there are no wrong answers. The model is created with a set of instances as input and no output attribute. Practical examples include clustering, in which the goal is to simply find relations between examples of the dataset, or dimensionality reduction, in which the goal is to reduce the number of features of the dataset[124].

Semi-supervised is a combination of the previous two algorithms. It uses a few labelled instances and a larger of unlabelled. It is useful because there are large amounts of unlabeled data, since it is easier to obtain data, than to classify it. It has the same objective as supervised learning, to classify unknown attributes of a given instance[121].

In reinforcement learning, the algorithm has the objective of maximizing the reward. As described in Awad and Khanna[121], this algorithm works like a trial-error, where certain actions are rewarded or penalized. The algorithm automatically determines the ideal behavior, obtaining the best reward.

Given this work's nature, supervised learning algorithms will be used, as it allows for classification and output prediction, while providing better predicting power. Furthermore, there is a fair amount of data available for use in instances and attribute classification.

2.7.3 Evaluation of model performance

After the model has been built based on the training dataset, the model performance can be calculated with the test subset. As the test set is composed of instances that have not been considered for the model, they can assess how the model predicts new instances. Depending on the type of output feature,

the model evaluation metrics will be different. For instance, regarding classification, concepts need to be defined,namely wether the prediction is either correct or incorrect. An NxN confusion matrix can be built, in which N is the number of possible ways to classify the instance (number of possible outputs). A simple confusion matrix is represented in table 5. Usually, the real values are represented in the rows, while the predicted values are in the columns. The diagonal cells of a confusion matrix correspond to the correctly predicted values[121].

Table 5.: A 2x2 confusion matrix, for a classification problem of an attribute with two possible values.

Real \ Predicted	Positive	Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative

These confusion matrices, allow calculating important metrics, such as precision, defined as the ratio of correct positive predictions by the total number of positive predictions (equation 1) and recall/sensitivity, which is the ratio of correct positive predictions and the total number of real positive examples in the set (equation 2). Precision is the percentage of relevant results and recall is the percentage of correctly classified positive cases. There are cases in which improving precision is preferred (a higher number of true positives, while having low false positives), instead of improving sensitivity (high number of positive values, reducing false negatives). Usually, increasing one leads to a decrease in the other, and a compromise must be reached. Alternatively, an in-between metric named accuracy/PECC (Percentage of Examples Correctly Classified), defined as the ratio of correctly classified instances by the total number of instances (equation 3) is used[124]. The last metrics to mention for classification are the F1 score, which is considered an average of precision and recall and defined as two times the ratio of precision times recall by the sum of precision and recall (equation 4)[123;125], and Matthews correlation coefficient (MCC), considered the more balanced score, taking into account true, false positives and negatives[126].

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{2}$$

$$PECC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

$$F1 = 2 \times \frac{Precision \times Sensitivity}{Precision + Sensitivity} \tag{4}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{5}$$

A good way to visualize these metrics is through ROC (Receiver Operating Characteristic) curves. These graphs are built with recall (also called True Positive Rate for these kinds of graphics) in the y-axis and False Positive Rate, defined as $FPR = \frac{FP}{FP+TN}$, in the x-axis. The graph provides a summary of the performance of the model and the higher the area under the ROC curve, the better the model.

Regression problems, in which the output variable is numerical and continuous, are more straightforward for evaluating performance. The models are evaluated based on the distance between the predicted values and the real values. Several metrics can be calculated, such as SSE (Sum of Square Errors), RMSE (Root Mean Squared Error), which is the square root of the ratio of SSE and the number of instances; and MAD (Mean of Absolute Deviation) (equations 6-8) to perform this assessment. For a model to have a good performance, the closer to zero these metrics are, in theory, the better the model[124;127].

$$SSE = \sum_{N}^{i=1} (Predicted_i - Actual_i)^2 \tag{6}$$

$$RMSE = \sqrt{\frac{SSE}{N}} \tag{7}$$

$$MAD = \frac{1}{N} \sum_{N}^{i=1} |Predicted_i - Actual_i| \tag{8}$$

These performances should be handled with care, since underfitting and overfitting can impair models. Underfitting occurs when the model does not learn the training dataset well and cannot predict the output correctly, because the training dataset is small or the selected features are not relevant to the

problem at hand. On the other hand, overfitting happens when the model learns the examples too well, because there are too many features, or the algorithm is too complex. Methods such as feature selection, reshaping of the training dataset can help prevent overfitting[124].

Cross-validation, leave-one-out or bootstrap are more robust ways to assess performances. The first two methods are useful when there are few instances in the training dataset. Cross-validation starts by dividing the data into several defined parts. The ML algorithm is trained with all subsets except for one, which will be used in training. This process is repeated for all created subsets, and the performance is provided by the average of the defined subsets[123;124].

Leave-one-out is similar to cross-validation in every aspect except one. The only difference is that each training/testing subset is created by assigning only one instance to the test, while the rest of the data is used in training. Although having a higher computational burden it is useful for smaller datasets. The performance is also determined by the average of all the subset's performance[123].

Lastly, bootstrap is more useful for larger datasets. It has the same principle as cross-validation, but instead of creating subsets without replacement (in cross-validation instances can not be repeated in different subsets), it can create subsets with repeated instances. It has the advantage of allowing to have instances that are used in the training algorithms and can be used for testing. The performance is provided by the average of all the sub-models created[123].

2.7.4 Model optimization

After creating several models, these have to be evaluate, using methods such as the hyperparameter optimization. All ML algorithms have various parameters that control the model performance. The objective of hyperparameter optimization algorithms is to provide a model with parameters that obtain the best performance for a dataset[123].

An example of a hyperparameter optimization algorithm is the grid-search. Although simple, this algorithm can be computationally demanding, as it will test every possible combination of the different parameters and its inputted values. For example, testing an algorithm with two parameters, each with four possible values will lead to 16 models to analyse[124].

More efficient methods are available, such as random and bayesian search. Whereas the former's parameterization is based on a statistical distribution which defines the number of combinations, the latter determines parameters based on past evaluations, in which the parameters' previous assessments are considered[124].

2.7.5 Model selection

Ensemble learning algorithms are model selection methods that combine several models with low performance called weak learners, instead of trying to build the best model. Combining multiple learning algorithms and the results, leads to more robust and better performing models and can reduce overfitting, rather than just using one algorithm[123].

An example of these ensemble learning algorithms is Bagging. Several weak learners are combined, receiving equal weight; thus, the final predictions will average of all the model's predictions. One application of bagging are Random Forests, that use decision trees as weak learners[123;124]. Another ensemble learning algorithm is boosting, which considers previously built models, by assigning higher weights to models that perform better, like AdaBoost[123].

2.7.6 Machine learning algorithms

Although several algorithms are available, only the ones considered relevant for this work will be considered.

SVMs[128] are ML algorithms used for classification or regression problems. SVMs try to find the hyperplane that better separates two classes. Each new observation is assigned to a class; thus, usually being used for binary classification[121]. A hyperplane will not separate all examples correctly. SVMs account for a soft margin (possible errors) that go through the hyperplane boundary, which prevent outliers from building a poor model[129]. Another parameter is called kernel function, which is useful for cases in which the hyperplane is not linear. The most used kernel is Radial Basis Function (RBF) and the most important parameters are C and gamma that define the hyperplane margin and the points considered for separation, respectively. A large gamma value can lead to overfitting as all points will be separated

from the rest. Conversely, small values of gamma might lead to underfitting[124;130], as shown in figure 5.

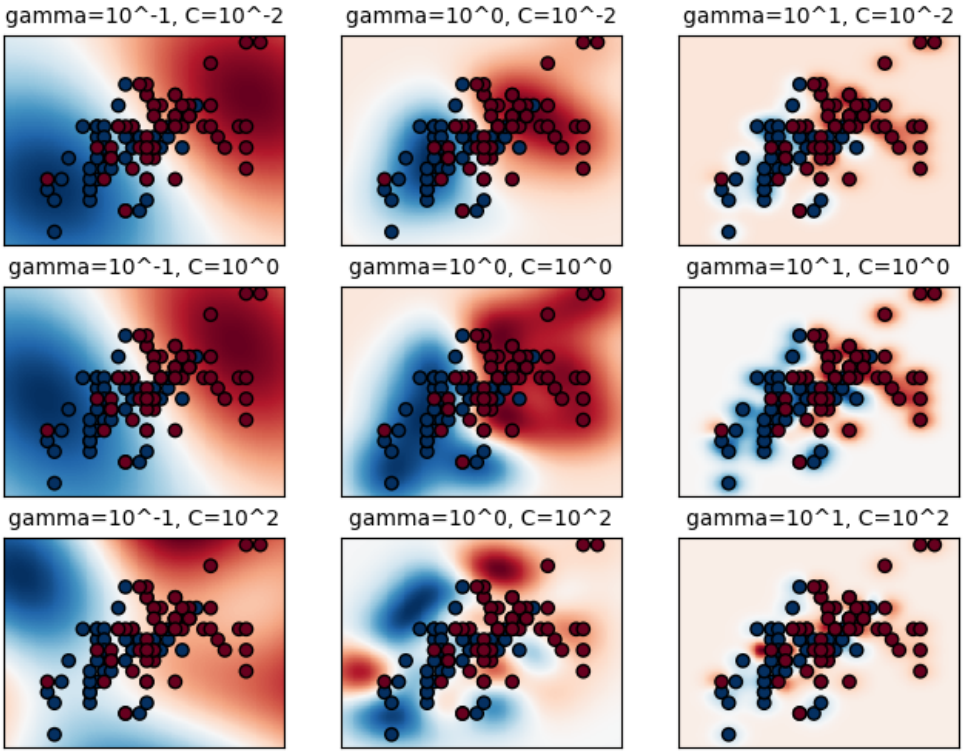


Figure 5.: Changes in classification of two variables, represented in red and blue, through changes in the values of C and gamma, parameters for the SVM algorithm. From left to righ, gamma's value increases; from top to bottom, the value of C increases. Image extracted from Scikit-learn[130]

KNN[131] is a classification algorithm that bases classification of new examples on their closeness to classes of the training subset, by calculating distances and assigning the example to the closest class. Hence, labelled examples for the training dataset, the distance metric and the number of nearest neighbours (k), are required for this algorithm. The higher the value of k, the more classes with single points will be created. Lower values can lead to misclassification, because of underfitting. There is a need to balance this k value[121].

Trees[132] use a decision tree structure, in which the nodes have rules, which indicates an example of what branch it should move. After going through several nodes and branches, the algorithm reaches the leaves that contain the classification output for that example[123]. The most used algorithm is ID3 (Iterative Dichotomiser 3)[132], in which the split of a node does not depend on attributes. Hence, an optimal solution cannot be reached. The algorithm stops when there are no more attributes to go through,

the maximum tree depth has been reached, or when all examples have been classified. The larger the depth of the tree, the higher the likelihood of overfitting[124].

Logistic regression (LR)[133] despite what the name might indicate, is a classification algorithm applicable to binary problems and can be used in multiple classifications. It uses the sigmoid function for classification. If one test instance's output is closer to 0 than 1, it will be classified as 0. The output is a probability value, ranging from 0 to 1. This calculation is based on linear regression variables, calculated from the training dataset[123;124].

Artificial Neural Networks (ANN)[134] are inspired by brain neurons. This algorithm is used for classification or regression problems. An input is passed through multiple connections, each associated with a weight. Then, a value is calculated with the input values and the weights. This value is applied to an activation function, in which the output is determined. This activation function can be binary (if the input value is below a specific number, the function will classify it as 0 or 1), logistic (as explained before), hyperbolic, among others[121].

ML pipelines were developed in this work to determine existing interactions between phage proteins and the bacterial receptors. Given the possible application of phages to eliminate bacteria in therapy or industrial processes, developing a tool to quickly identify phages that target a particular host can be advantageous. Although such an approach is not new[98;99], it is possible to improve these works by increasing the number of bacterial hosts and improving datasets. The novelty of this work was to find relationships and patterns between phage and bacterial strains.

3 Methods

3.1 PhageHost dataset

The target for this study, PhageHost, was defined with phages that infect *E. coli*, *Klebsiella pneumoniae* and *Acinetobacter baumannii*, gram-negative bacterial species. A table containing all available sequenced bacteriophages, for the species defined, was obtained from National Center for Biotechnology Information (NCBI) Virus database[84]. A comma-separated values (CSV) file[135] with three columns, phage accession identifier, phage name and bacterial host was retrieved from the database.

Different approaches were followed, to find bacterial hosts for each phage. Initially, the respective GenBank entry was obtained from the phage identifier (ID)[85]. Using Biopython's Entrez package[136], the infecting bacteria name was obtained for each phage when available. There was either a "host" or "lab_host" feature in the phage's GenBank entry, that specified this information. The bacteria name would only be added if the strain was specified. The next step was to search the NCBI nucleotide database for the bacterial strains associated with each phage (**name search**), though only for the three bacteria defined before, excluding contigs, gene sequences, or shotgun sequencing, thus filtering for complete genomes. Furthermore, the IDs would be automatically added if these contained the strings "NC_", "NZ_", "AC_", "CP", "AE", "CY", "AP" at the beginning of the ID (based on NCBI handbooks[137;138]). However, this process was fallible, and a manual check was necessary for dubious cases.

The other method of finding bacteria that possibly interact with phages was to query, for each phage, all the associated PubMed publications (**PubMed search**). Another query was performed for each publication against the nucleotide database, to find related bacteria. With the same rules as before, only appropriate IDs were selected. From these PubMed articles, a text search was also performed in the abstracts (**abstract search**). This search aimed at finding references to strings related to the bacterial species' strain specifications.

A search for prophages was performed in bacterial genomes (**prophage search**). Prophages are phages that integrate into the bacterial genome[139]. Therefore, if a phage sequence is found in a bacterial genome, almost certainly the phage will infect that bacteria. Based on this, a BLASTn search was performed with phage genomes as input against a specific bacterial species[92]. Finally, a check on all

the added bacteria was performed by accessing the nucleotide NCBI accession and a manual filter: if any reference to virus, phage, or unwanted bacterial species was present in the accession title, it would be removed from the PhageHost dataset.

A method to count the number of bacteria added for the four search methods was implemented. However, many accesses to the NCBI Entrez services were required, rendering the processes time consuming and, if the internet connection was unstable, the counting would be halted. Hence, a way of resuming queries was implemented, though, resuming the counting of bacterial strains was not accounted for. Furthermore, this counting method also accounted for bacteria that were later removed from the PhageHost dataset.

After the maximum amount of information was extracted, all protein sequences and functions were extracted and saved in dictionaries, for each phage and bacterium present in the dataset. If protein sequence information was missing from an organism, the entry would be removed from the PhageHost dataset.

3.2 Data filtering

With every phage and bacterium associated with its proteome, there was a need to filter more relevant proteins, to reduce the size and dimensions of the data, and find more relevant ML features. For the bacteria, the process was simply performed by filtering proteins, considering only external proteins. In theory, only these could be used for possible phage binding. Finding the type of carbohydrates or capsules present outside each bacteria was also analysed though not retrieving relevant results. External proteins were to be filtered using PSORTb[140]. However, PSORTb was too slow, and thus all bacterial proteins were used.

Even though there are several carbohydrate databases available online, no relevant information for exterior sugars extracted. A capsular database, the *E. coli* K antigen 3-dimensional structure database (EK3D) was analysed[141]. Databases of bacterial carbohydrates were also explored (Bacterial Carbohydrate Structure DataBase[142]) and other more general (GlyTouCan[143], PolysacDB[144], Glycosciences[145]). Capsular prediction tools, such as Kaptive[146], for *K. pneumoniae* and *A. baumannii*, and SerotypeFinder[147], for *E. coli*, were also explored, but not implemented in the current work.

For phages, only phages' proteins that possibly interact with a host were considered. First, a local BLASTp was performed, using proteins with annotated functions to create a local database. Hits were considered significant for e-values lower than one and bit scores of at least 30. These thresholds were defined based on a manual curation of the query proteins' attributed functions. In the cases of multiple hits, the lowest e-value and maximum bit scores were considered. The e-value was set to higher than what is common for BLAST searches, as an homology search was performed against a small database. Thus, e-value scoring was not as relevant as the bit score, which provides a normalized result independent of sequence length[148]. From the manual curation, values higher than 30.0 usually indicated proteins with significant homology.

After this, CD-HITs were performed for all available proteins[149]. CD-HIT is used for clustering biological sequences. The program ran, setting the identity threshold as 90%, which guaranteed that highly similar proteins are clustered. After the clusters were defined, the annotations would be propagated to all proteins. CD-HIT was implemented together with BLAST, as it is a different algorithm. Since BLAST works by comparing a sequence to every other in the database and finds the most similar, CD-HIT instead creates clusters of similarity. CD-HIT might not group the most similar sequences in the same cluster, but guarantees that the sequences are significantly similar. Hence, BLAST complements this gap, by finding the most similar hit for each protein[92;150].

Finally, if unknown proteins were still present, a domain/family search was performed with InterProScan[151], which retrieves domains from various sources, namely Pfam[152], CDD[153], PROSITE[154], among others. The e-value threshold was set to 1.0, because the InterPro search retrieves significant matches and hits from irrelevant databases were removed. A specific protein could be associated with multiple domains/families. Therefore, manual validation and verification was performed and only relevant functions were used to attribute functions.

This process is represented in figure 6 and was repeated three times, to ensure a minimal amount of unknown proteins.

A filtering on known protein functions and domains was performed, to obtain phage tails and proteins that interact with the host's receptors. The process was a mix of automatic filtering according to keywords and manual verification. Words associated with nucleic acid binding, gene regulation, capsid proteins, lysozyme, amidase were used to filter unwanted proteins. Words associated with tail fibres, tail spikes,

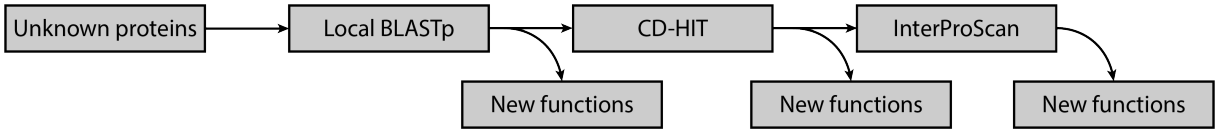


Figure 6.: Schematic representing the process of finding functions of unknown phage proteins. Starting with unknown proteins, a local BLASTp was performed and new functions found. Then, all proteins were clustered with CD-HIT to find additional functions. The remaining unknown proteins were submitted to an InterProScan search to find functions. This process can be repeated using the remaining unknown proteins.

and host binding proteins were used to filter proteins that potentially bind to host receptors. A complete list is present in chapter 4.

3.3 Feature construction

After the bacteria and phage data were properly filtered, the best approach for constructing features relied on sequence information. The sequence information does not limit the dataset, as countless organisms have the protein sequences described. As explained before, other types of information, namely structural and carbohydrates, are lacking and would significantly reduce an already small dataset. Furthermore, it is possible to explore protein sequences to obtain satisfactory predictive models (described in more detail in section 2.6.4).

Before features were calculated, organization is crucial to obtain significant and relevant results. The objective of PhageHost is to predict phage-host interactions at the strain level. The algorithm's output as a strain designation for each phage would be unfeasible, due to the high number of outputs and few training cases for each strain.

Moreover, this approach is complex in terms of input. Ideally, only phage proteins should be used as input to predict the bacterial strain. However, such an input does not seem sufficient to classify interactions with such high specificity, as it would require a much broader training dataset. Boeckaerts [99] resorted to phage data, however their classification was at species level. In contrast, Leite et al. [98] resorted to both bacteria and phage data for species-level classification.

Using both phage and bacteria data seems to be the best way to give accurate prediction. Methods to make a trinary output prediction of 'Yes', 'No' and 'Unknown' were analysed. The positive cases would be considered the ones identified and present in PhageHost's dataset (as described in section 3.1), the

negative cases would be assumed to be pairing phages of one species to bacteria of different species, assuming that phages do not infect bacteria outside their species range. Usually, phages have narrow host ranges, rarely infecting hosts outside the defined species, or closely related organisms[155]. While not always correct, it should be a fair assumption to take, since the cases where a phage infect more than one species are rare. The problem of the trinary output arises for the 'Unknown' cases. It was suggested that phages of a given species would be classified as unknown when paired with a host of the same species, in which a positive interaction was not found. The problem that arises is that phages not found to have positive prediction, that in reality have, could be classified as unknown, which is not correct, creating an incorrect bias in predictions. Besides this, if two similar phages grouped with two similar bacteria, one with demonstrated positive interaction and the other marked as unknown, it would lead to a similar set of features having two possible outputs, which, in turn would lead to dubious and probably wrong outputs.

On this note, a binary output, 'Yes' and 'No', was selected for the dataset applied to ML. Positive interactions are found in the PhageHost dataset and negative are as described in the previous paragraph. For unknown cases (same species interactions with no proven output), no classification was assigned. Each PhageHost dataset entry would correspond to an interacting or non-interacting pair of a phage and a bacterium strain.

The next question is how to balance the data, and not have an overrepresentation of either case, which would lead to a bias towards that response. If this were to happen, the ML model's fitting could lead to incorrect predictions. Two datasets were created, one with the approximate same number of positive and negative outputs and one with three times more positive cases than negative. It was intended to also create a dataset with three times more negative cases than positive; however, feature calculation time was extremely lengthy. For the balanced dataset, negative cases were built by randomly assigning 12 bacteria from the two host species, different from the species it infects. The negative cases were built by pairing each phage randomly with four different bacteria species.

After building the datasets, features described by Leite et al.[98] were explored. DOMINE was explored to extract interaction scores for proteins[156], without success. Instead, sequence information features were explored, such as amino acid frequency, molecular weight, aromaticity, flexibility, all calculated with Biopython functions[136]. Another interesting set of features was calculated with NetSurfP[157], which allows predicting relative accessible surface areaRSA, with deep learning. If this value was above a certain

threshold, an amino acid could be considered superficial and possibly involved in binding sites. These superficial amino acids were characterized and used as ML features. These features were discarded, as initial results were not adequate.

Different features were then explored. The features described by Wang et al.[120] and in section 3.3 were calculated. According to their dipoles and volumes of side chains, each amino acid is placed in one of seven groups, first used in the Conjoint Triad (CT) method[158]. Sequences are represented as a set of seven numbers (table 6), and features constructed based on them. A list of all the features is present in the supplementary table 19.

Table 6.: New classification of amino acids into seven groups based on dipoles and side chains' volumes.

Group 0	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
Ala, Gly, Val	Cys	Ile, Leu, Phe, Pro	Tyr, Met, Thr, Ser	His, Asn, Gln, Tpr	Arg, Lys	Asp, Glu

One of the features, called local descriptor[120], involved dividing the sequence into ten groups, as shown in figure 7. This method (labelled **grouping**) was given the name of 'phage_group_X_n' or 'bacteria_group_X_n', where X is the letter corresponding to the groups (A-J), and n the amino acid group (0-6). Relative frequencies of amino acids were obtained for each group and these values were normalized by the total number of proteins considered for each calculation. For example, for group A, seven features are calculated, corresponding to the normalized frequency of each amino acid.

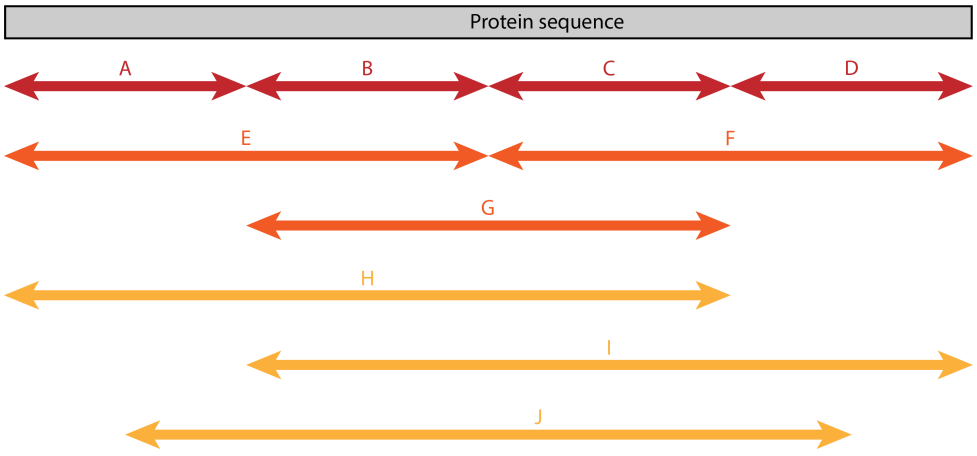


Figure 7.: Representation of the local descriptor method, in which a sequence is divided into ten groups. Four of them (A, B, C, D) represent 25% of the sequence, three of the groups (E, F, G) represent 50% and the other three (H, I, J) 75% of the sequence.

Another feature was the composition of the seven amino acid, that involved calculating each amino acid's relative frequency. The features for this method (labelled **composition**) were named 'phage_comp_n' or 'bacteria_comp_n', where n is the number of the amino acid (0-6). The values were normalized by a min-max scaling, before the same normalization performed in grouping. The min-max scaling is presented in equation 1, in which for a set of frequencies x, each value y is normalized to the minimum and maximum of the set. Thus all frequencies are within the same range and are easily comparable.

The final set of features added were **k-mers**, of size two, named 'phage_kmer_n' where n is a given k-mer of size two. Counting k-mers involves calculating the frequency of subsequences, in this case, of size 2, within a sequence. As before, min-max scaling and normalization against the number of proteins of an organism were performed.

$$x_{norm} = \frac{x_y - min(x)}{max(x)} \tag{1}$$

Two datasets were created to be tested with ML algorithms. Both had the same number of positive cases, the difference being in the number of negatives. One of them had approximately the same number of negative features as positives and the other had one third of the amount of positives as negatives.

3.4 Machine learning

The train and test datasets were created with 80% of the original dataset, whereas the remaining 20% was available for validation. The test dataset was created with 30% of the available dataset while the remaining 70% were used for training. These partitions were performed so the output resulted in a balanced proportion of 'Yes' and 'No'.

ML algorithms were applied to predict phage-bacteria strain interactions. First, different standardization were tested for one dataset, against five ML algorithms, KNN, random forests (RF), SVM, artificial neural networks (ANN) and logistic regression (LR). F-scores (equation 4 of section 2.7) were calculated for the cases of no standardization, standard feature scaling, min-max scaling, maximum absolute value and normalization. For no standardization unprocessed values were used. For standard scaling, min-max

scaling, maximum absolute value, equations 2, 3 and 4 were used to calculate new values, where x represents a feature, x_i a single value. For normalization, new values were calculated allowing the features to follow a normal distribution.

$$x_{scaled} = \frac{x_i - mean(x)}{std(x)} \tag{2}$$

$$x_{scaled} = \left(x_i - \frac{min(x)}{max(x)} - min(x)\right) \times (max(x) - min(x)) + min(x) \tag{3}$$

$$x_{scaled} = x_i \div max(x) \tag{4}$$

The same five ML algorithms were used to evaluate the created , balanced (same number of positive and negative cases) and unbalanced (3 times more positives than negatives) datasets, mentioned in section 3.3. F-scores were calculated and confusion matrices built to understand the performance of the two datasets better.

Dataset reduction was performed with recursive feature elimination and a five-fold cross-validation, using RF. Removed features were checked, and the performance of the five ML algorithms assessed, with a five-fold cross-validation, returning accuracy scores and running time, in seconds. If the running time was lower, and the score was not significantly different, the reduced dataset was selected for the ML algorithm.

Hyperparameter tuning was performed for the algorithms, with a grid search five-fold cross-validation, against a manually defined space of parameters for each algorithm. Thus, the best parameters were retrieved, and the F-score, MCC (equation 5), precision and recall metrics calculated. Feature importance was calculated with the parameters for each algorithm, to avoid overfitting. Permutation importance was calculated, and the features sorted according to their bias towards the model performance.

A method for predicting interactions between a pair of phage and bacteria was implemented, and the dataset fitted to the ML algorithm. When the phage or bacteria were not available in the dataset, the proteins were retrieved and filtered, allowing to implement the same features. After the pair contained all features and was properly standardized, the interaction prediction was performed, with the output being a simple 'Yes' or 'No'.

3.5 Galaxy implementation

A method to apply the studied ML algorithms for prediction was implemented. If either phages or bacteria were absent from the created dataset, the proteins were retrieved, or imported as a FASTA file. The filtering was performed with the local known phage protein database by running a BLAST search. An optional InterProScan was also implemented to complement the BLAST. The same set of features were calculated and standardized. After selecting the algorithm for prediction, a binary output of 'Yes' or 'No' was returned and saved in a tabular format file, TSV.

PhageHost was implemented in the Galaxy platform[159], with the Planemo software[160], to provide an user-friendly interface, and, is available online at https://galaxy.bio.di.uminho.pt.

4 Development

This work was developed in Python 3.8.5[161], using the Python IDE (integrated development environment) PyCharm, divided into 6 core scripts, and an extra one for running in Galaxy, as depicted in figure 8. All the code is available at https://gitlab.bio.di.uminho.pt/PedroAraujo/phage_host.

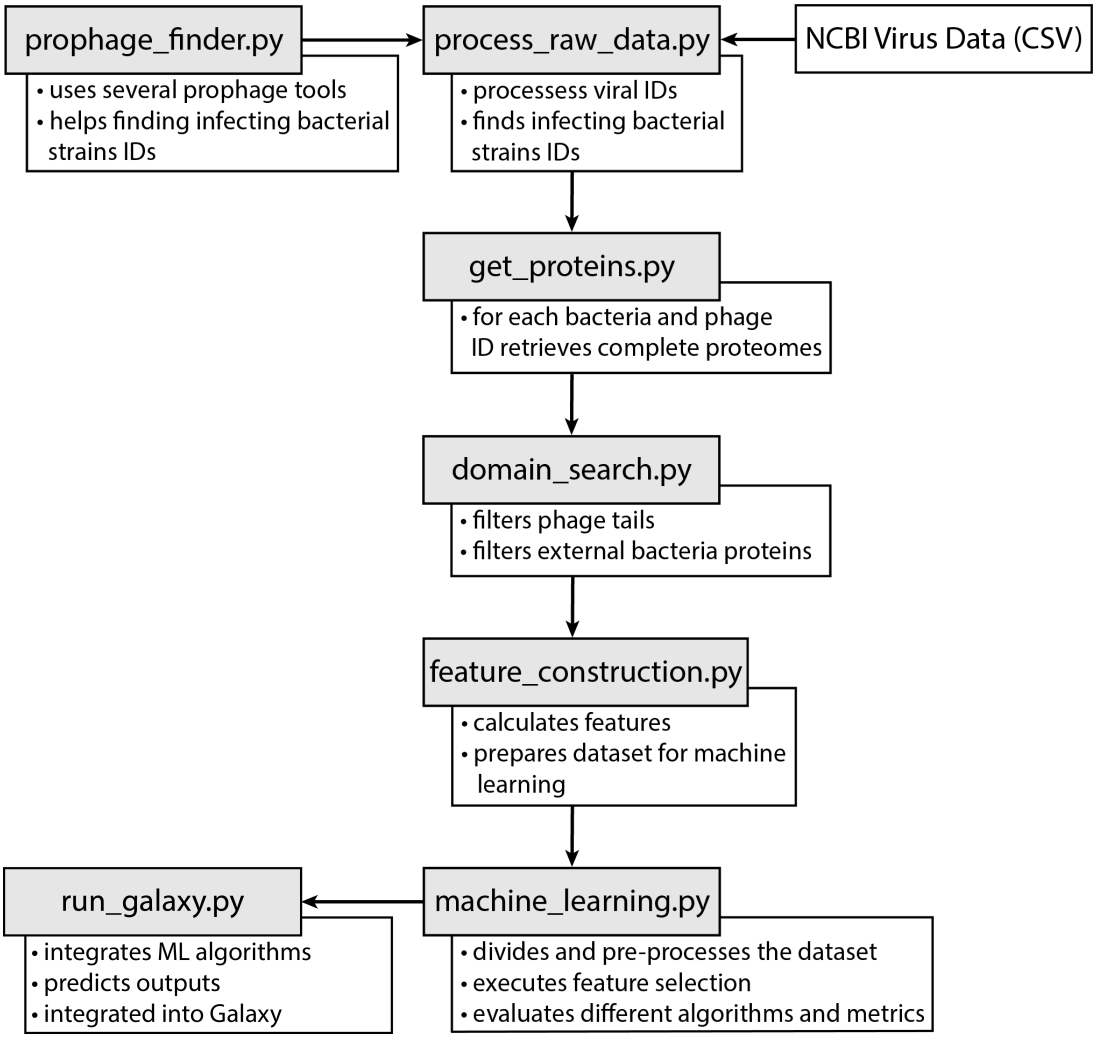


Figure 8.: Schematic representing the workflow and structure of this work.

process_raw_data.py

A file containing phage IDs and respective names was downloaded from NCBI Virus[84] and processed with the "process_raw_data.py" script. This script updated the file, by adding GenBank[162] IDs from related bacteria. The original file had three columns, phage ID, phage name, and host bacteria name, labelled as 'Accession', 'Species', 'Host', respectively.

The phage's GenBank information was accessed, with Entrez functions from Biopython, to complete the host's names[136]. These allow accessing Entrez Programming Utilities (E-utilities) which facilitate searching and querying the NCBI[163]. If a reference to a 'host' or 'lab_host' was present, and the name present in the file did not have strain information specified, it would be added. A column 'Host_ID' was created, to add infecting bacteria IDs that include a list of IDs for each phage.

The script includes three methods. The first uses the "Entrez.elink" function, to find PubMed publications[164] associated with the phage ID and, from these publications, another "elink" search is performed to find GenBank IDs of bacteria. The second method resorted to the 'Host' names present, searching for bacterial IDs through "Entrez.esearch". This search was performed specifically for the described organisms, including keywords, such as "complete sequence" and excluding keywords such as "shotgun", "phage", "cds", "gene".

The final method is similar to the first, as it finds PubMed articles. However, it seeks the abstracts for bacterial strains. The "esearch" was set with the same keyword filtering as before.

All three methods automatically added IDs when strings "NC_", "NZ_", "AC_", "CP", "AE", "CY", "AP" (based on NCBI handbooks[137;138]) were present. However, a manual check was required for the remaining cases. When a phage entry did not have bacterial strains associated, it would be removed, providing a CSV file containing 'Accession', 'Species', 'Host' and 'Host_ID' information.

Finally, a method to check every bacteria ID was implemented, in which an "Entrez.efetch" was performed with the ID, returning the respective GenBank entry. Only the description was verified, and if a reference to the three bacterial species defined ('pneumoniae', 'coli', 'baumannii') was not available, or the words virus or phage were not present, the bacteria was removed from the dataset.

prophage_finder.py

"prophage_finder.py" was implemented further to complete the CSV with bacteria IDs and positive cases. It used the Biopython's BLAST[92] to find potential prophages integrated into bacterial genomes, "Blast.NCBIWWW", for querying and "Blast.NCBIXML", for processing the output. Phage genomes were used as input, against a specific bacterial species, and the expected value threshold was set to 10e-5. Using the JSON package[165], the results were saved as a dictionary, where a bacteria was associated with a list of phage IDs.

get_proteins.py

The "get_proteins.py" script was used to extract phage and bacterial proteomes from the complete CSV data. Using the "Entrez.efetch" function, GenBank features were extracted from IDs and a dictionary created. An organism (phages and bacteria) ID was associated with another dictionary, in which a protein ID was associated to a list of protein function and sequence. Dictionaries for both types of organisms were saved with the JSON package, as "phagesProteins.json", for phages, and for bacteria the files were named after their ID. Simultaneously, a FASTA file was created for all phages ("phagesProteins.fasta"). The FASTA headers contained the organism ID and the protein ID separated with a hyphen, to more easily identify.

domain_search.py

The script "domain_search.py" was used to reduce the dimensions of the files and complexity of the information, by extracting only external proteins of bacteria and tail proteins of phages. The external proteins were obtained with the dockerized version of PSORTb[140]. First, by extracting the pre-computed results[166] and then performing a local search for the remaining bacteria. The compilation of these outputs was saved as "results_psort.faa.out".

Proteins with unknown function were annotated by three methods, to find phage tails. A local BLAST database was created with the command 'makeblastdb' using the proteins with known function. Unknown proteins were run against this database, with the command 'blastp' and new functions were attributed when the e-value was lower than 1.0, and the bit-score was higher than 30.0. CD-HIT clustering[149] was also implemented, with the command 'cd-hit' and the threshold set as 90%. If known proteins were clustered with unknown, the function would be propagated, when relevant. For the remaining, an InterProScan search[151] was performed for domains and families, with the command 'interproscan' and the outputs manually processed to filter relevant results.

feature_construction.py

The actual filtering of proteins was implemented in "feature_construction.py". The script's class constructor filtered bacteria, by reading the "results_psort.faa.out" and finding proteins classified as 'OuterMembrane_Score' or 'Extracellular_Score'. The matching proteins were saved as "externalProts.json", for easier read in future runs. For phages, a word search was performed for protein functions. Any reference to 'tail fibre', 'lyase', 'sialidase', or others (full list available in table 7), would be considered a

Table 7.: List of keywords to filter phage tail proteins. The positive and negative tags mean that if a protein function contains at least one of the positive words, but does not contain any negative words, it will be considered a phage tail.

Positive	fiber, fibre, spike, hydrolase, bind, depolymerase, peptidase, lyase, sialidase, dextranase, lipase, adhesin, baseplate, protein h, recognizing, protein j, protein g, gpe, duf4035, host specifity, cor protein, specificity, baseplate component, gp38, gp12 tail, receptor, recognition, tail
Negative	nucle, dna, rna, ligase, transferase, inhibitor, assembly, connect, nudix, atp, nad, transpos, ntp, molybdenum, hns, gtp, riib, inhibitor, replicat, codon, pyruvate, catalyst, hinge, sheath completion, head, capsid, tape, tip, strand, matur, portal, terminase, nucl, promot, block, olfact, wedge, lysozyme, mur, sheat

host binding protein, as proposed by Latka et al.[167] and verified by manual verification of domains and functions.

After the filtering was performed, the ML datasets were built in the constructor as a pandas dataframe[168]. A line was built for each interacting phage bacteria pair, with the 'Yes'. A phage was paired with twelve random bacteria from the two different species for the negative cases, assembling a dataset with a 1:1 ratio of positive:negative cases. A dataset with a 3:1 ratio was also assembled. Based on the features described in section 3.3, this script implemented them with the scikit-bio package's, that facilitated processing sequences and calculating frequencies[169]. It was used for grouping and composition, to count the frequency of amino acids' types with the function "Sequence.count". The function "Sequence.kmer_frequencies" was used to retrieve k-mer frequencies. The grouping features' columns were named 'phage_group_X_n' or 'bacteria_group_X_n', the composition columns were named 'phage_comp_n' or 'bacteria_comp_n' and the k-mers columns 'phage_kmer_n' or 'bacteria_kmer_n'. The values were calculated for all features, and included in the dataset, in the column identifying the feature. The pandas dataframes were saved with the pickle package, and named "FeatureDataset" and "FeatureDataset_3_1".

machine_learning.py

The "machine_learning.py" script initiates by importing the "FeatureDataset" file and, with scikit-learn[170], it executes every process required to create and evaluate ML models. All scoring metrics undermentioned were calculated with scikit-learn's "metrics" functions, such as "accuracy_score", "f1_score", "precision_score", "recall_score", "matthews_corrcoef" and "confusion_matrix". The five ML algorithms were used for all methods described below, with scikit-learn's "neighbors.KNeighborsClassifier", for KNN,

"svm.LinearSVC" for SVM, "ensemble.RandomForestClassifier" for random forests, "neural_network.MLPClassifier" for artificial neural networks and "linear_model.Logistic Regression" for logistic regression.

First, the data was standardized with "preprocessing" functions, like "StandardScaler", "MinMaxScaler", "MaxAbsScaler" and "Normalizer". The dataset was divided into validation, training and test, with the "model_selection.train_test_split" method. The scoring metrics were calculated with training and testing datasets to determine the most appropriate standardization techniques for subsequent analysis.

Performance of the two datasets created, "FeatureDataset" and "FeatureDataset_3_1" was calculated. Using the same five ML algorithms, and the training and testing datasets, F-score and confusion matrices were calculated. Once again the best performing dataframe was selected for analysis.

A feature reduction was performed with "feature_selection.RFECV", with five fold cross-validation and RF. A reduced dataset, "dataset_reduced", was created and saved as a pandas dataframe. The reduced and the full dataset, were used to perform five fold cross-validation for all ML algorithms, with "model_selection. StratifiedKFold". The execution time of the cross-validation was measured, with the time package, as well as the accuracy. For each ML algorithm, the selected dataset had the best compromise between execution time and accuracy.

Every ML algorithm hyperparameters were tunned with "model_selection.GridSearchCV", using a manually defined space of parameters, against the validation dataset, returning optimal parameters. F-score, MCC, precision and recall were calculated, to evaluate performance and a "inspection.permutation_importance" function was used to avoid overfitting. Ideally, no feature would have greater importance than others.

If either the phage or bacteria were not present, their proteomes would be retrieved and filtered through processes similarly described for "domain_search.py" and "feature_construction.py" to predict interactions between phage-bacteria pairs. The first step was calculating the input pair's features, as described before, for "feature_construction.py". The scaler used in the constructor was used to standardize the features, with "preprocessing.StandardScaler", was saved and used here. Then, a simple "predict" function was applied to the constructed models and the predictions were outputted.

run_galaxy.py

This script implements methods of protein retrieval, filtering and prediction of interaction, based on previously described methods. Regarding input, the algorithm supports either a list of NCBI IDs or a FASTA

file, containing a single organism's proteome. If the IDs are present in the locally created dataset, these are imported with the JSON package; otherwise, Biopython's "Bio.Entrez" package is used to retrieve protein sequences from GenBank entries.

Initially, the phage's proteome is filtered by creating a local database with previously predicted and known protein functions, using the comand 'makeblastdb'. The 'blastp' command is run for the phage proteomes, against this protein database, assigning new functions. Although optional, an InterProScan search, 'interproscan' can also be run to complement the local BLAST.

The phage tails are filtered using the strings present in table 7. Subsequently, the prediction is implemented by importing methods from "feature_construction.py". The entries are processed and standardized with NumPy and scikit-learn's "preprocessing.StandardScaler". Finally, the appropriate ML algorithm is imported, the dataset created from previous scripts fitted, and the entries' output predicted, with scikit-learn.

This script is called when running the tool through Galaxy, assisted by an XML (Extensible Markup Language) file, "run_galaxy.xml", which allows the user to easily define inputs and retrieving outputs.

5 Results and Discussion

5.1 Finding phage-bacteria interacting pairs

The table obtained from NCBI Virus (15/12/2020) included 2953 phages infecting the three defined species (2350 of *E. coli*, 210 of *A. baumannii* and 393 of *K. pneumoniae*). After running the methods described in section 3.1, 960 phages were associated with at least one bacterial strain. The number of phages removed, 1993, indicates that there is still a considerable number of viruses that need to be studied and characterized. PhageHost's dataset contained 857 phages of *E. coli*, 26 of *A. baumannii* and 77 of *K. pneumoniae*. This disparity could lead to a serious unbalancing in the machine learning models. However, this will be discussed in a further section.

A total of 1990 different bacteria were found for the 960 phages. The much higher number of bacteria than phages is expected, as phages interact with multiple bacterial strains. The bacterial strains include 1350 *E. coli*, 195 *A. baumannii* and 449 *K. pneumoniae* bacteria. Looking at these numbers and comparing with the number of phages, the two underrepresented phages have a relatively higher number of bacteria. While *E. coli* have, on average 1.6 bacteria per phage, *A. baumannii* and *K. pneumoniae* have an average of 7.5 and 5.8 bacteria per phage, respectively. Figure 9 shows that, *E. coli* has a higher frequency of phages that infect a low number of bacterial strains, whereas, for the other two species, a higher frequency of phages infect multiple bacterial strains. This distribution can slightly help dealing with the unbalancing of the data.

Counting the number of bacterial strains added with the methods described in section 3.1, was only possible for the **name search** method, since this was the fastest process, requiring less queries to the servers. The number of added bacteria strains was 2266 and, since the PhageHost dataset accounted for 2150 bacterial strains, it confirms that the current implementation of counting added strains will not produce meaningful results. A simultaneous search and removal method must be constructed to know objectively how many bacterial strains are added for each method.

These methods have issues that need to be addressed and considered before moving forward. The **name search** method retrieves bacterial strains that theoretically are infected by a given phage, while in reality this could be false. For the **abstract search**, it is not guaranteed that a strain mentioned in the

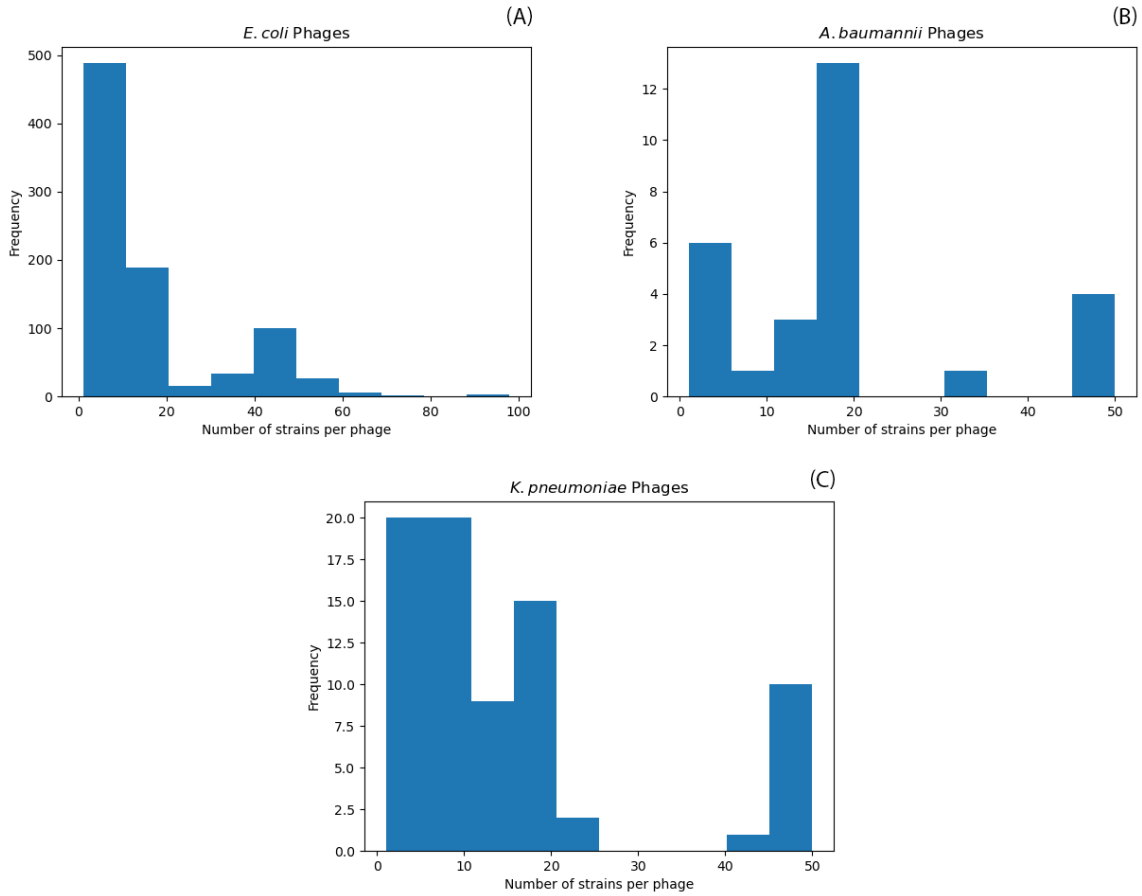


Figure 9.: Frequencies of strains for each phage, (A) *E. coli*, (B) *A. baumannii* and (C) *K. pneumoniae*. In each graphic, in the x axis, the number of strains for each phage is represented, and in the y axis, the frequency of the occurence of each number.

article is positively related to a phage. It is possible that a strain is mentioned for cases of non-interaction. **Prophage search** could find integrated prophages, but because of mutations, currently the phage is not able to infect the bacteria strain.

Nonetheless, a novel unique dataset, **PhageHost**, containing sequenced phages and interacting sequenced bacterial strains was created. This dataset is available at https://nextcloud.bio.di.uminho.pt/s/xE9zTD8dW4WZFZx, with the name "NCBI_Phage_Bacteria_Data.csv".

Comparing PhageHost with other databases, the Virus-Host Database[81], despite having 4014 sequenced phages, only seldom does this database specifies the strain. It has the disadvantage of mostly conveying bacterial species information. Furthermore, it only contains 344 *Escherichia* phages, 53 *Acinetobacter* phages and 108 *Klebsiella*. PhageReceptor[82] has the advantage of providing a high-quality database for experimentally proven phage receptors; however, it does not specify host strain. It contains

information for 107 *E. coli* phages, three *A. baumannii* and one *K. pneumoniae*. This database's applicability is that, for each phage, it would be possible to identify the receptor and, more importantly, hosts holding that receptor. This is feasible for proteins, as bacterial strains can have fully sequence genomes and protein information available. However, for carbohydrates such information is harder to obtain, as discussed in the next section. The PhagesDB although containing a plethora of information, it does not provide information on the target species of PhageHost's dataset.

More information could be extracted from online catalogues of bacteriophages, in which, bacteriophages are available, with known hosts described. A couple of examples include the DSMZ collection [171] and ATCC collection [172]. It would be possible to extract this information and determine if the phages and the hosts are sequenced. However, this was not performed because organism sequence information was not readily available and a query for the species' strains was required in most cases.

5.2 Constructing and filtering bacterial data

As explained in section 3.2, external carbohydrates and capsules for each bacteria were sought. Several carbohydrate databases were explored, since the GenBank accessions and other tools did not provide this information. The EK3D [141] contains several structures and chemical formulas of *E. coli* capsules, as well as proteins involved in their formation. However, it is not possible to retrieve any strain information, and perhaps a manual check might be required. Similar proteins could be sought in PhageHost's dataset; however, the only information provided is related to the capsular group rather than to specific capsule types.

For the Bacterial Carbohydrate Structure DataBase [142], although providing more information on organisms and strains, a manual verification on the information extracted from scientific articles was required. Even though PolysacDB [144] is well organized and has relevant information, it contains few entries. GlyTouCan [143] and Glycosciences.de [145] only contain carbohydrate structures. Furthermore, even if these structures could be retrieved and associated with the correct strain, it would add the complexity of trying to find a way of integrating these into ML models into this work. Several works described interactions between proteins and carbohydrates using tridimensional protein structures to feed the ML models, as described in section 2.6.4.

Knowledge of the carbohydrates that phages target could provide better insight into phage-host range. If it is experimentally proven that a phage interacts with a certain capsular type, this phage can likely interact with every bacteria possessing it. PhageHost's dataset, from the previous section, could be expanded and completed with this information. Furthermore, even though protein-carbohydrate interactions are the most common way of phage adsorption, it does not represent every case.

Another way for phages to interact with bacteria is through their external proteins. Hence, a simple method to predict external bacterial proteins for every bacteria was implemented with PSORTb[140]. The purpose was to input all the proteins into a FASTA file and process the output. Due to the web service's file size limitations, a local Dockerized version of PSORTb was used. Pre-computed results and division of the files into smaller parts were used in order to reduce the data's dimensions; yet no output was obtained. Therefore, all the bacterial proteins are being used to calculate the features to be used in ML, making the feature calculation process more lengthy and resource consuming. The protein filtering would reduce run costs, and it would theoretically improve the features, as processing all proteins may introduce a bias due to non-binding proteins. Filtering for external proteins would attenuate this bias. Although not all are involved in binding, most unwanted proteins would have been removed.

5.3 Constructing and filtering phage data

The methods for finding phage proteins functions, described in section 3.2, reduced the number of unknown proteins. Instead of the 960 phages present in PhageHost's dataset, 974 phages were used in the protein function database. Hence, a larger database of phage proteins functions was built, giving it more examples and more confidence. It would also be possible to increase the size of this database, by adding phages not present in the PhageHost dataset.

Finally, phage tails were filtered. From the initial protein database, of 974 phages, 6 phages with no tail or spike proteins were identified. Of these, five were non-tailed phages and the only tailed phage had no identified tail/spike proteins. The former, belonged to filamentous or capsular families of phages, *Inoviridae* and *Leviviridae* and did not possess these proteins[13;14]. This result indicates that the process of finding protein functions was successful, as organisms that were not supposed to have tails or spikes did not contain them. However, it is noteworthy that non-tailed phages were included in this dataset, as

references to tail spikes were found and considered for host binding. Over 300 phages belonging to the *Microviridae* family are present in the dataset, because, although not tailed, all have spikes. Furthermore, a file containing the functions of tail proteins was created to evaluate tail filtering. Overall the algorithm seemed to only filter proteins relevant as tails or receptor binding proteins, from a more manual curation. Moreover, tails were identified for tailed phages, while for non-*Caudovirales* phages, only spikes were identified, indicating that this method and the created phage tail database have promising results, since they can identify each type of protein correctly.

The phage protein database with known function is available at https://nextcloud.bio.di.uminho.pt/s/xE9zTD8dW4WZFZx, with the name "phagesProteins.json"

5.4 Feature exploration and construction

Regarding data balancing, not to have an overrepresentation of either positive or negative case, which would lead to a bias towards that response, two datasets were constructed, balanced and unbalanced. The total number of positive interactions obtained was 12673, which, must remain unchanged regardless of the chosen dataset size. The balanced dataset had the same approximate number of positive and negative outputs and the unbalanced had three times more positive cases than negative. It was intended to also create a dataset with three times more negative cases than positive; however, feature calculation time was extremely lengthy. The balanced dataset had 11314 negative interactions and the unbalanced had 3954.

The features described by Wang et al.[120] and in section 3.3 were used. In total, for each pair, 252 features were created, plus one for the output, 140 for grouping (10 groups×7 aa×2 organisms), 14 for composition (7 aa×2 organisms) and 98 for k-mers (49 k-mers×2 organisms). Although providing good results (discussed in the next section), all these calculations were very time-consuming. For the balanced dataset, all the calculations took more than two entire days to finish executing (Windows Subsystem for Linux, Windows 10 Pro 64-bits, AMD Ryzen™5 1400, 8,00GB RAM, NVIDIA GeForce®GTX 1650). Increasing the dataset would aggravate the situation; thus a dataset with more negative cases was not constructed.

5.5 Machine learning performance

After the datasets were constructed and completed with the features, these were tested in ML algorithms. Five models were used, based on the significance to the problem at hand. KNN, RF, SVM, ANN and LR. These were the algorithms used for other similar works and are default when it comes to ML. As this problem is binary, SVM and LR were used, as these were originally designed for these problems. As for KNN and ANN, these provide different ways of looking at data and finding patterns, besides being commonly used for ML problems. Finally, RF was used because it is an ensemble method, which should theoretically perform well and avoid overfitting.

5.5.1 Dataset standardization

After the dataset was properly divided into training, testing and validation sets, the first analysis performed was on the type of standardization and its influence on the performance of ML models. Data standardization is important for algorithms that predict outputs, based on distances between points or trends in datasets. KNN, for instance, relies on distance between points, to attribute these to different clusters. SVM tries to separate the points with a hyperplane, thus depending on distances. In ANN, generally, standardization should be performed, to prevent the algorithm learning from a single feature with a high range of values, merely because it was not standardized.

Therefore, four data standardization approaches were explored, standard, MinMax, maximum absolute value scaling and normalization. Besides being compared to one another, these were compared model performance-wise for the unstandardized dataset. The results for the five ML algorithms are present in table 8, as F-scores. Standard scaling of the dataset led to better predictive power, with maximum absolute value scaling close behind, more significant for ANN and LR. As expected, no standardization led to worse results, as well as normalization, for SVM, ANN and LR. This might happen because with normalization, features with high variability will still display it. These features are not standardized to be in accordance with the remaining.

Between the different ML algorithms effects of different standardization techniques are also evident. The algorithms dependent on standardization for model performance were highly influenced: SVM and

Table 8.: Performance of standardization techniques with five ML algorithms. The scores are given as F-scores.

Model \ Standard	None (F-score)	StandardScale (F-score)	MinMaxScale (F-score)	MaxAbsScale (F-score)	Normalize (F-score)
KNN	0.9551	0.9615	0.9529	0.9549	0.9606
RF	0.9848	0.9870	0.9848	0.9857	0.9791
SVM	0.8266	0.8963	0.8912	0.8978	0.7660
ANN	0.9356	0.9801	0.9288	0.9412	0.9181
LR	0.7327	0.8972	0.8607	0.8775	0.5924

Table 9.: Performance of the two datasets created, balanced and unbalanced, with five ML algorithms. The scores are given as F-scores.

Dataset	KNN (F-score)	RF (F-score)	SVM (F-score)	ANN (F-score)	LR (F-score)
Balanced (1:1)	0.9620	0.9867	0.8816	0.9748	0.8813
Unbalanced (3:1)	0.8515	0.9376	0.6220	0.9064	0.6179

ANN perform significantly worse when no standardization is applied, as well as LR. The method that was less affected was RF, as it is not as dependent on standardization os data. In the end, using standard scaling led to better results for the ML models. Therefore, it was selected for all analyses.

5.5.2 Assessing the number of negative cases

The two datasets created, balanced and unbalanced, were tested, to verify the best ratio of positive:negative cases, for the five algorithms. The goal was to evaluate the two datasets and not to compare the algorithms. Theoretically, however, a model will perform worse in an unbalanced dataset, as these are unable to learn the data properly. For instance, underfitting, in which the algorithm generalizes the data and leads to poor models. Prediction results for both datasets are presented in table 9. For a more thorough look at dataset's performance, confusion matrices were built, providing further insight into the predictions, as shown in tables 10 and 11 for the unbalanced dataset, and tables 12 and 13 for the balanced dataset.

From the results in table 9, it is clear that the unbalanced dataset had a worse performance for every model produced. The only model that obtains a somewhat acceptable performance is the RF. Likewise, ANN, has interesting results as it relies on feedback to improve predictions. The remaining methods

Table 10.: Confusion matrices for the **unbalanced** dataset, for two ML algorithms, KNN and RF.

	KNN				RF		
Real \ Pred	Positive	Negative	Total	Real \ Pred	Positive	Negative	Total
Positive	3051	64	3115	Positive	3079	36	3115
Negative	192	734	926	Negative	77	849	926
Total	3243	798		Total	3156	885	

Table 11.: Confusion matrices for the **unbalanced** dataset, for two ML algorithms, ANN and LR.

	ANN				LR		
Real \ Pred	Positive	Negative	Total	Real \ Pred	Positive	Negative	Total
Positive	3077	38	3115	Positive	3012	103	3115
Negative	127	799	926	Negative	466	460	926
Total	3204	837		Total	3478	563	

Table 12.: Confusion matrices for the **balanced** dataset, for two ML algorithms, KNN and RF.

	KNN				RF		
Real \ Pred	Positive	Negative	Total	Real \ Pred	Positive	Negative	Total
Positive	2931	80	3011	Positive	2963	48	3011
Negative	148	2598	2746	Negative	61	2685	2746
Total	3079	2678		Total	3024	2733	

Table 13.: Confusion matrices for the **balanced** dataset, for two ML algorithms, ANN and LR.

	ANN				LR		
Real \ Pred	Positive	Negative	Total	Real \ Pred	Positive	Negative	Total
Positive	2917	94	3011	Positive	2700	311	3011
Negative	72	2674	2746	Negative	345	2401	2746
Total	2989	2768		Total	3045	2712	

perform rather poorly, since they can not correctly learn from the data and, thus, are unable to separate the two classes correctly.

When performing an overall analysis of the confusion matrices for the unbalanced dataset, a high number of negative values are incorrectly classified as positive (false positives). Perhaps the higher weight these have during the learning phase, leads to a bias in the learning towards the positive cases, which could be considered underfitting, as the models are too simple and unable to discern the data.

As for the confusion matrices for the balanced datasets, it is perceptible that the ML models perform better than for the unbalanced datasets. The number of false positives or false negatives is significantly lower, and the total number of positives and negatives predicted is close to the real values, indicating that it was able to learn the data. Hence, this dataset is more appropriate to use than the unbalanced. It would be interesting to explore another dataset, containing more negative cases, than positives, and determine if the data would lead to better models.

5.5.3 Feature selection

Feature selection for the balanced dataset was performed with RF, the best performing algorithm from previous analyses. A total of 66 features were removed, leaving the reduced dataset with 186 features. A list of the removed features is present in supplementary table 20, in which, 32 features were related to phages and the remaining 34 were related to bacterial features, indicating that, overall, the dataset is well balanced. Regarding the three different types of features (**grouping**, **composition** and **k-mers**), for **grouping**, 20 out of 140 features were removed (ratio of 0.143). For **composition**, 2 out of 14 were removed (ratio of 0.143). Finally, for **k-mers**, 44 of 98 were removed (ratio of 0.449).

It seems that **k-mers** are not as relevant for the ML. Looking at **k-mers** for the different organisms, 19 of the removed **k-mers** were for phages and 25 for bacteria. Features for an organism do not seem less relevant than the other. It would be interesting to confirm wether changing the size of k-mers from two to three would lead to more relevant features. However, as mentioned in section 5.4, the time to calculate all the features was more than two days, with 252 features. Using k-mers of size three, k-mer features would increase from 98 to 686, which would exponentially increase run time for calculations. Therefore, this was not performed.

The reduced dataset was compared with the complete, using five-fold cross-validation, for all five ML models, measuring classification accuracy, and time to run, reaching a compromise between model performance and running time. The results are present in table 14. Regarding the best performing models, KNN, RF and ANN, feature selection led to somewhat better performances, while the time to run was significantly reduced, especially for KNN and RF. Whereas for the remaining two models, SVM and LR, classification performance was worse with the reduced dataset, though time differences were negligible. Therefore, for KNN, RF and ANN the reduced dataset will be used for further analyses, while SVM and LR models, will use the complete dataset.

Table 14.: Effect of feature selection on the performance of the dataset. The scores are provided as classification accuracies and time, in seconds.

Dataset	Metric	KNN	RF	SVM	ANN	LR
Complete	Accuracy	0.9677	0.9863	0.8911	0.9734	0.8920
	Time (s)	81.73	235.53	7.67	175.75	5.91
Reduced	Accuracy	0.9696	0.9873	0.8733	0.9745	0.8764
	Time (s)	63.61	205.09	5.91	166.82	5.03

5.5.4 Hyperparameter tuning

Hyperparameter tuning was performed with grid search against defined space of possibilities. The parameters that provided the best ML models were obtained, and different performance metrics, that help evaluate performance with greater detail. All these informations are present in table 15. Furthermore, feature importance was evaluated, with permutation importance, to determine wether the good model fitness is due to overfitting the data.

Overall, comparing the F-scores in this table with table 14, shows that the worse performing models, SVM and LR obtained better results. For the remaining models, no significant differences in performance were found. The results are not exactly the same as before, as the training, testing and validation datasets are built randomly, which confirms the results and quality of the predictive models.

The better performing models are RF and ANN, both having similar metrics overall. The model with worse metrics is LR.

Feature importance was explored, to prevent overfitting. If a set of features is given a high importance, then the model is dependent on these features, learning them too well. For RF and KNN, no features highly

Table 15.: Hyperparameter tuning on the five ML models. A list of the best parameters for each model are present, as well as detailed scoring metrics.

Model	Hyperparameters	F-score	MCC	Precision	Recall
KNN	'algorithm': 'auto', 'leaf_size': 5, 'n_neighbors': 2, 'p': 1, 'weights': 'distance'	0.9657	0.9281	0.9562	0.9753
RF	bootstrap': False, 'criterion': 'gini', 'min_samples_leaf': 2, 'min_samples_split': 4, 'n_estimators': 200, 'oob_score': False	0.9834	0.9656	0.9812	0.9858
SVM	'C': 10, 'degree': 2, 'gamma': 'auto', 'kernel': 'rbf'	0.9572	0.9092	0.9477	0.9669
ANN	'activation': 'tanh', 'alpha': 0,0001, 'hidden_layer_sizes': 200, 'learning_rate': 'adaptive', 'solver': 'adam'	0.9828	0.9634	0.9780	0.9877
LR	'C': 10, 'penalty': 'l2', 'solver': 'liblinear'	0.9111	0.8138	0.9143	0.9079

influenced the model's decision making, while on SVM and ANN some features were more determining than others, although not significantly. LR here also performed the worse out of every model, with eleven features that influenced the output, present in supplementary table 21.

5.5.5 Overall model performance

The best performing model is RF, because of the performance after hyperparameter optimization and since it has no bias in the features. These results can be compared with the works, of Leite et al.[98] and Boeckaerts[99]. Since the positive cases are built from the same species interactions and negatives are built from different species, this classification is similar to these works. Although the models developed in PhageHost can predict species interactions, they can predict strain level interactions as well. Hence, the scoring allows determining wether host species are correctly classified and compare these with the published works.

PhageHost's best performing model vastly outperforms the best model from Boeckaerts[99]. However, PhageHost's model requires two inputs, a phage and a host, while Boeckaerts's model only requires a phage input. Furthermore, the PhageHost model only requires protein information, being thus simpler.

Leite et al.[98] also use both phage and host data, to predict interactions, though only at species level. The immediate advantage of PhageHost is that it can to predict interactions at strain level, since it was

trained with that specificity. Furthermore, the way Leite et al.[98] built the dataset, leads to unbalancing and classifications may seem better than they actually are. The features are quite simple and do not rely on filtering, which might make sense in species classification scope. Their data was also trained with a small number of bacteria, leading to more bias. Their best model performance is similar to PhageHost's best model, although the mentioned problems may be falsely improving classification accuracy.

PhageHost's models were tested by predicting infection of phages against bacteria. This test ran only for examples present in the dataset with no output predicted, to decrease running time. For example, an *E. coli* phage tested against *E. coli* hosts should not classify every entry as a positive interaction. Furthermore, interactions with the other two host's species should lead to negative outputs, for most cases. Nonetheless, should positive interactions be predicted, it would be interesting to have a more detailed look into that pair.

Three tables were built, for interaction of three phages from different species, against seven bacterial strains, for each species, to evaluate the model's performance applied to specific cases. Tables 16, 17 and 18 detail interactions between phages and *E. coli*, *A. baumannii* and *K. pneumoniae* bacteria, respectively.

For *E. coli* bacteria, table 16, predictions seem reasonable for the *E. coli* phage, as not all are classified as having positive interactions. Hence, the model seems to differentiate between strains. However, when looking at the other two phages, some unexpected positive interactions were found, likely associated with the random assembly of the negative cases in the PhageHost dataset. Perhaps if more negative cases were considered that included *A. baumannii* and *K. pneumoniae* phages, the model could have better predictions. Instead, since few positive and negative interactions are present for the two species, predictive power is not as great as for *E. coli* phages, which is the most represented in the dataset.

As for *A. baumannii* bacteria, table 17, the same unexpected predictions occur for the *E. coli* phage. Since *A. baumannii* is the least represented in the dataset, it could explain these results, as the model may not be able to learn from such few examples. In the case of *K. pneumoniae*, the predictions seem correct, indicating that negative cases were well balanced between *k. pneumoniae* bacteria and *A. baumannii* phages. The *A. baumannii* shows every single interaction as positive, which may be due to the lack of variety in the dataset, as well as the fact that almost every *A. baumannii* phage was found to infect several bacteria.

Table 16.: Example of three phages from each species and predicted interactions with *E. coli* host strains.

	NC_007414 (*E. coli* O157:H7 EDL933)	MK033499 (*E. coli* C600)	CP032679 (*E. coli* K-12 MG1655)	NC_013008 (*E. coli* O157:H7 TW14359)	NC_023323 (*E. coli* ACN001)
NC_050154 (Escherichia phage)	Yes	Yes	Yes	Yes	No
NC_047817 (Klebsiella phage)	No	Yes	No	No	Yes
NC_041915 (Acinetobacter phage)	Yes	No	Yes	Yes	No

Table 17.: Example of three phages from each species and predicted interactions with *A. baumannii* host strains.

	NC_010606 (*A. baumannii* ACICU)	NZ_LN865143 (*A. baumannii* CIP70.10)	CP053100 (*A. baumannii* ATCC 17978)	CP050911 (*A. baumannii* DT-Ab020)	NZ_KP890934 (*A. baumannii* BM2686)	NC_006877 (*A. baumannii* pMAC)
NC_050154 (Escherichia phage)	Yes	Yes	Yes	Yes	Yes	Yes
NC_047817 (Klebsiella phage)	No	No	No	No	No	No
NC_041915 (Acinetobacter phage)	Yes	Yes	Yes	Yes	Yes	Yes

Finally, *K. pneumoniae* bacteria, table 18 seem to have very good predictive results. This example shows no infection of phages from different species, while predictions for the *K. pneumoniae* phage seem to differentiate between strains. An explanation for this, could be that the negative cases were well balanced for *K. pneumoniae* phages, while the number of positive cases are significant enough to make predictions.

Overall, further study is required on the balance of the negative interactions. Due to a lapse, negative phages' interactions were built only with *K. pneumoniae* bacteria, for *E. coli* and *A. baumannii* phages, while *K. pneumoniae* phages are only paired with *E. coli* bacteria. This should explain these results, which are visibly worse for *A. baumannii* phages.

Ideally, phage-bacteria should be balanced. Each one should be paired with bacteria from the other two species. However, the proportion still requires further studying. The question that arises is whether

Table 18.: Example of three phages from each species and predicted interactions with *K. pneumoniae* host strains.

	JN420336 (*K. pneumoniae* pNDM-MAR)	NZ_KY271406 (*K. pneumoniae* H150820806)	MG288678 (*K. pneumoniae* F160070)	MF510496 (*K. pneumoniae* SCKP-LL83)	CP034200 (*K. pneumoniae* KpvST383)	HG918041 (*K. pneumoniae* Kp15)
NC_050154 (Escherichia phage)	No	No	No	No	No	No
NC_047817 (Klebsiella phage)	Yes	Yes	No	Yes	Yes	Yes
NC_041915 (Acinetobacter phage)	No	No	No	No	No	No

A. baumannii phages would require more negative interactions, or if the same proportion should be maintained, to avoid bias towards negative prediction.

Furthermore, for *E. coli* phages, another pertinent question is how many bacteria from the other two species should be used for negative interactions. PhageHost does not seem able to classify all *K. pneumoniae* bacteria as negative, indicating that a higher proportion is necessary. Such bias is perhaps associated with the greater number of *E. coli* examples in the positive dataset.

5.6 Galaxy implementation

Finally, PhageHost was implemented in Galaxy (https://galaxy.bio.di.uminho.pt), an intuitive software that allows users to define the inputs for the prediction easily. PhageHost is depicted in figure 10, accepts NCBI IDs, or FASTA files as inputs, for both phage and bacteria. If the inputs are FASTA files, it only accepts one organism, with the FASTA having protein information for that organism. Otherwise, IDs are recommended as input, as these allow extracting more information, from the GenBank accession, and allow multiple inputs for one organism type, for example, it accepts one phage input and several bacteria, and vice-versa, though it is not possible to input more than one ID for phages and bacteria at the same time, also represented in figure 10.

The tool has two advanced options. Performing an InterProScan search is optional, as it is quite demanding and time-consuming. If left off, tail functions will be sought against the locally created phage

tail database (mentioned in section 5.3). Performing the InterProScan search should complete missing protein functions and is recommended when FASTA input is used for the phage.

The default ML model to perform the prediction is the RF model, that obtained the best predictive power. An option to perform prediction with the SVM model is also available, to theoretically reduce run times. However, it is highly recommended to use the first option.

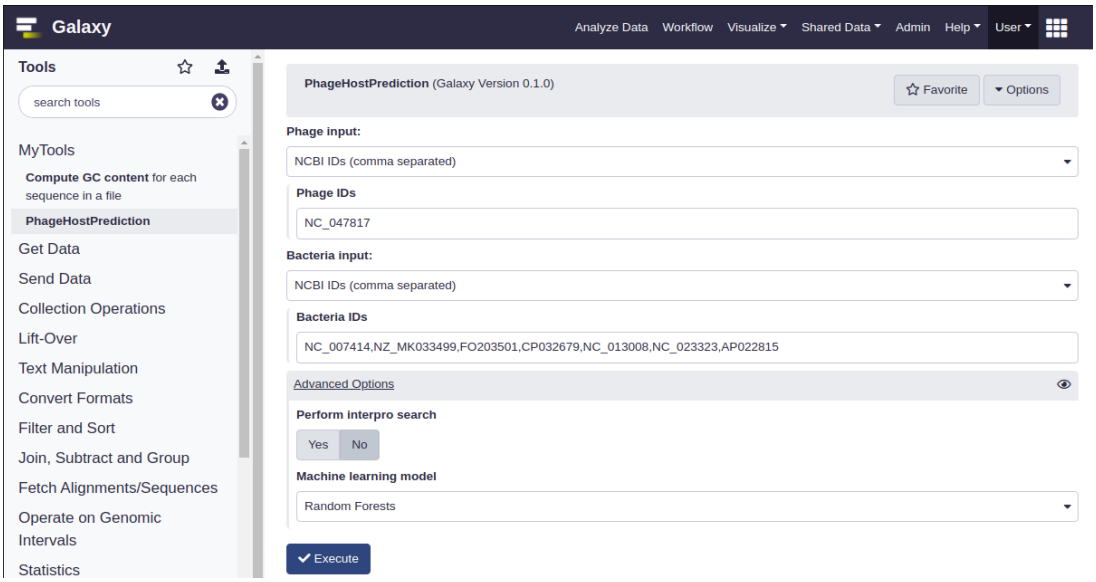


Figure 10.: Galaxy interface of the PhageHostPrediction tool. Example of inputs are provided, along their organization.

The output, is returned in a tabular format (TSV) in which the first column contains the phage-bacteria pair, separated by '-', and the second column contains the prediction result, a simple 'Yes', for positive predictions and 'No' for the negative. An example of an output from the tool is present in figure 11. According to several tests, eight interactions take between 2 to 3 minutes to calculate and generate the output. The output can also be downloaded as a TSV file. A small help section is available for easier understanding of the *modus operandi*.

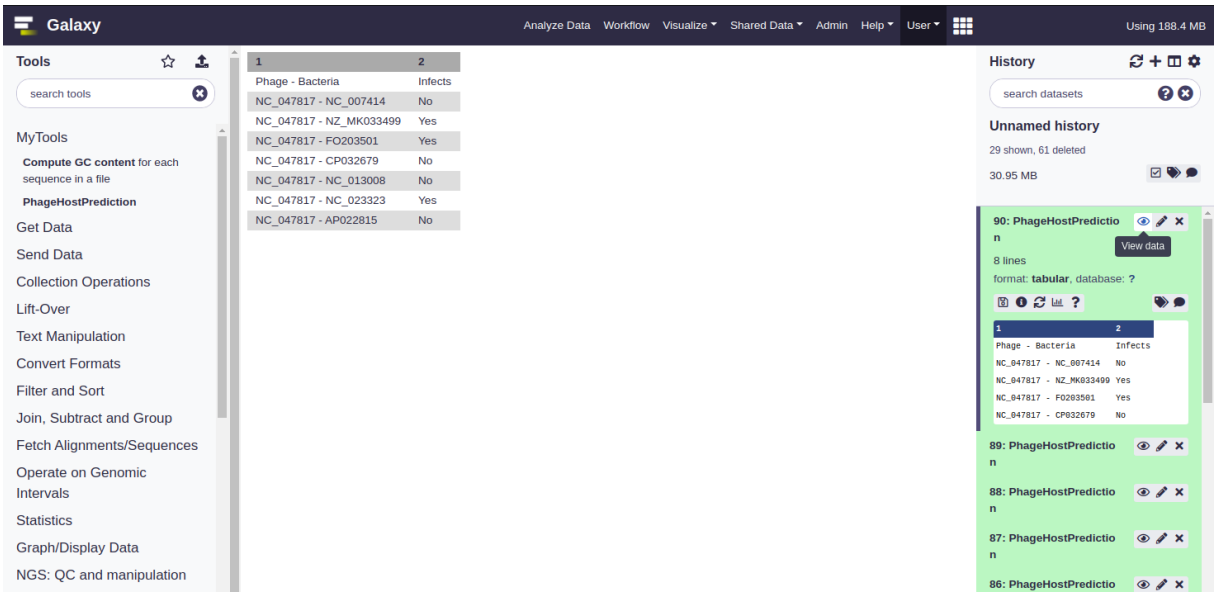


Figure 11.: Output results of an example of a PhageHostPrediction instance. The output is provided in a tabular, TSV format.

6 Conclusions and future work

Overall, a pipeline implementing protein data filtering and extraction and a phage-host prediction method, PhageHost, was developed in this work. Several metrics were tested and better predictive power was obtained, compared to similar works, while also having the ability to predict strain-level interactions.

The first major result obtained from this work was a dataset of predicted positive interactions between phages and bacteria, at strain level, which has never been performed before with this level of detail. Previous databases only described interactions at species level, or described only phage binding receptors. This work managed to develop a dataset of 960 phages with bacterial hosts fully described, for the three species proposed, *E. coli*, *K. pneumoniae* and *A. baumannii*, and could be expanded to other species, through the methods and scripts developed.

Although innovative, this dataset is not perfect. It is not guaranteed that searching the article's abstract will always find positive interaction cases. There could be situations in which a bacteria is mentioned for a phage, but as a negative infection case. For example, improvements could be made with @Note, which would allow for a more rigorous verification of even the whole article, if available, and implementing queries that automatically update the PhageHost dataset. Other methods/databases that identify phage receptors could be used to find bacteria and increase the interaction dataset, as described in sections 2.6 and 3.2. The dataset should also be updated, often, by checking the NCBI Virus website for updates, or using other phage information sources. Inconsistencies are also present in this database, for instance, entries (phage or bacteria) with different IDs, that correspond to the same genome. A manual verification for this is not feasible, due to the of the data's high dimensionality, though it could be verified with programmatic access.

Another important result from this work is constructing a phage protein database, containing more than 960 examples of phages with tails identified. Besides providing information on phage tails, it also contains several other proteins with different functions. It can be used as a basis to find protein functions for related organisms. The main purpose of this database was to find phage tails through a similarity search. Possible improvements to this database, could be to include structural information. However is not feasible as of now, due to the low number of structures available. This purpose could be achieved by

predicting proteins' structural features and finding fingerprints for tails, adding another level of complexity however, due to the high number of predictions involved.

Two ML datasets were obtained, each with 252 features. The best predictive ML model had a MCC score of 96.6% and F-score of 98%. These results are better than previous works, either sequence-based, such as the works from Leite et al.[98], Boeckaerts[99] or genome-based, such as Edwards et al.[87], Villarroel et al.[94], Gałan et al.[95], Wang et al.[96] and Zhang et al.[97] described, while having the capability of making predictions at bacterial strain level.

Pre-processing the dataset, found that using a standard scaler on the dataset leads to better predictive power for all models. Analyzing the number of negative cases, showed that the dataset with more negative cases produced significantly better results. More negative cases should be tested in the future, to improve the data balance. Feature selection reduced the number of features in the dataset to 186, a reduction of 66 features. Finally, hyperparameter tuning was performed and the best predictive models were found with the optimal parameters. Two ML models were implemented into a Galaxy instance, named PhageHost, available online. The more accurate and slightly slower RF model, and the less accurate but slightly faster SVM model, were implemented.

These models need improvements, especially by balancing the dataset's negative cases, to balance species such as *A. baumannii* representation. Nonetheless, innovative predictive results were obtained, when finding host strains for *E. coli* and *K. pneumoniae* phages. A different set of features could also be explored, for instance, by using k-mers of size three, which might represent the protein sequences better. Using NetSurfP could also lead to interesting features, since it would enable the characterization of superficial amino acids, possibly involved in binding.

The main objectives proposed in this thesis were achieved. An interaction database for phage-host, was obtained from different sources. A method to predict and classify phage tails was provided, with a protein database created for 674 phages. Machine learning models were developed and implemented online, for prediction of phage-host interactions at strain level.

Bibliography

[1] Madigan, Michael T. et al. Brock Biology of Microorganisms. In Limited, Pearson Education, editor, *Brock Biology of Microorganisms*, chapter 8, pages 246–261. 14 edition, 2015. ISBN 978-0-321-89739-8.

[2] Alanis, Alfonso J. Resistance to Antibiotics: Are We in the Post-Antibiotic Era? *Archives of Medical Research*, 36(6):697–705, nov 2005. ISSN 01884409. doi: 10.1016/j.arcmed.2005.06.009.

[3] Shankar, PRavi. Book review: Tackling drug-resistant infections globally. *Archives of Pharmacy Practice*, 7(3):110, 2016. ISSN 2045-080X. doi: 10.4103/2045-080X.186181.

[4] Rakhuba, D V et al. Bacteriophage receptors, mechanisms of phage adsorption and penetration into host cell. *Polish journal of microbiology*, 59(3):145–55, 2010. ISSN 1733-1331.

[5] Kotsiantis, S. B. Supervised machine learning: A review of classification techniques. *Informatica (Ljubljana)*, 31(3):249–268, 2007. ISSN 03505596.

[6] Sulakvelidze, Alexander and Alavidze, Zemphira. Bacteriophage Therapy: A Mini Review. *Antimicrobial Agents And Chemotherapy*, 45(3):649–659, 2001. doi: 10.1128/AAC.45.3.649–659.2001.

[7] Weinbauer, Markus G. Ecology of prokaryotic viruses. *FEMS Microbiology Reviews*, 28(2):127–181, may 2004. ISSN 1574-6976. doi: 10.1016/j.femsre.2003.08.001.

[8] Salmond, George P. C. and Fineran, Peter C. A century of the phage: past, present and future. *Nature Reviews Microbiology*, 13(12):777–786, dec 2015. ISSN 1740-1526. doi: 10.1038/nrmicro3564.

[9] Carlton, Richard M. Phage Therapy: Past History and Future Prospects R. M. Carlton: Phage Therapy in the Past and Future. Technical report, 1999.

[10] Smith, H. W. and Huggins, M. B. Successful Treatment of Experimental Escherichia coli Infections in Mice Using Phage: its General Superiority over Antibiotics. *Microbiology*, 128(2):307–318, feb 1982. ISSN 1350-0872. doi: 10.1099/00221287-128-2-307.

[11] Housby, John N and Mann, Nicholas H. Phage therapy. *Drug Discovery Today*, 14(11-12):536–540, jun 2009. ISSN 13596446. doi: 10.1016/j.drudis.2009.03.006.

[12] Furfaro, Lucy L., Payne, Matthew S. and Chang, Barbara J. Bacteriophage Therapy: Clinical Trials and Regulatory Hurdles. *Frontiers in Cellular and Infection Microbiology*, 8(October):376, oct 2018. ISSN 2235-2988. doi: 10.3389/fcimb.2018.00376.

[13] Ackermann, Hans-W. Phage Classification and Characterization. In *Bacteriophages. Methods and Protocols. Volume 1: Isolation, Characterization, and Interactions. 3rd ed.*, volume 1, pages 127–140. 2009. ISBN 978-1-58829-682-5. doi: 10.1007/978-1-60327-164-6_13.

[14] Walker, Peter J. et al. Changes to virus taxonomy and the Statutes ratified by the International Committee on Taxonomy of Viruses (2020). *Archives of Virology*, 165(11):2737–2748, nov 2020. ISSN 0304-8608. doi: 10.1007/s00705-020-04752-x.

[15] Fokine, Andrei and Rossmann, Michael G. Molecular architecture of tailed double-stranded DNA phages. *Bacteriophage*, 4(2):e28281, 2014. ISSN 2159-7081. doi: 10.4161/bact.28281.

[16] Samson, Julie E. et al. Revenge of the phages: defeating bacterial defences. *Nature Reviews Microbiology*, 11(10):675–687, oct 2013. ISSN 1740-1526. doi: 10.1038/nrmicro3096.

[17] Nobrega, Franklin L. et al. Targeting mechanisms of tailed bacteriophages. *Nature Reviews Microbiology*, 16(12):760–773, aug 2018. ISSN 1740-1526. doi: 10.1038/s41579-018-0070-8.

[18] Oliveira, H. et al. Molecular Aspects and Comparative Genomics of Bacteriophage Endolysins. *Journal of Virology*, 87(8):4558–4570, apr 2013. ISSN 0022-538X. doi: 10.1128/JVI.03277-12.

[19] Bertozzi Silva, Juliano, Storms, Zachary and Sauvageau, Dominic. Host receptors for bacteriophage adsorption. *FEMS Microbiology Letters*, 363(4):fnw002, feb 2016. ISSN 1574-6968. doi: 10.1093/femsle/fnw002.

[20] Monteville, M. R., Ardestani, B. and Geller, B. L. Lactococcal bacteriophages require a host cell wall carbohydrate and a plasma membrane protein for adsorption and ejection of DNA. *Applied and Environmental Microbiology*, 60(9):3204–3211, 1994. ISSN 00992240.

[21] Gaidelyte, Aušra et al. The Entry Mechanism of Membrane-Containing Phage Bam35 Infecting Bacillus thuringiensis. *Journal of Bacteriology*, 188(16):5925–5934, aug 2006. ISSN 0021-9193. doi: 10.1128/JB.00107-06.

[22] Brown, Stephanie, Santa Maria, John P. and Walker, Suzanne. Wall Teichoic Acids of Gram-Positive Bacteria. *Annual Review of Microbiology*, 67(1):313–336, sep 2013. ISSN 0066-4227. doi: 10.1146/annurev-micro-092412-155620.

[23] Lindberg, A A. Bacteriophage Receptors. *Annual Review of Microbiology*, 27(1):205–241, oct 1973. ISSN 0066-4227. doi: 10.1146/annurev.mi.27.100173.001225.

[24] Ge, Haojie et al. The "fighting wisdom and bravery" of tailed phage and host in the process of adsorption. *Microbiological Research*, 230(September 2019):126344, jan 2020. ISSN 09445013. doi: 10.1016/j.micres.2019.126344.

[25] Xia, Guoqing et al. Wall Teichoic Acid-Dependent Adsorption of Staphylococcal Siphovirus and Myovirus. *Journal of Bacteriology*, 193(15):4006–4009, aug 2011. ISSN 0021-9193. doi: 10.1128/JB.01412-10.

[26] Kaneko, Jun et al. Identification of ORF636 in Phage SLT Carrying Panton-Valentine Leukocidin Genes, Acting as an Adhesion Protein for a Poly(Glycerophosphate) Chain of Lipoteichoic Acid on the Cell Surface of Staphylococcus aureus. *Journal of Bacteriology*, 191(14):4674–4680, jul 2009. ISSN 0021-9193. doi: 10.1128/JB.01793-08.

[27] Sao-Jose, C., Baptista, Catarina and Santos, Mário A. Bacillus subtilis Operon Encoding a Membrane Receptor for Bacteriophage SPP1. *Journal of Bacteriology*, 186(24):8337–8346, dec 2004. ISSN 0021-9193. doi: 10.1128/JB.186.24.8337-8346.2004.

[28] Baptista, Catarina, Santos, Mário A. and Sao-Jose, C. Phage SPP1 Reversible Adsorption to Bacillus subtilis Cell Wall Teichoic Acids Accelerates Virus Recognition of Membrane Receptor YueB. *Journal of Bacteriology*, 190(14):4989–4996, jul 2008. ISSN 0021-9193. doi: 10.1128/JB.00349-08.

[29] Davison, Sophie et al. Identification of the Bacillus anthracis Phage Receptor. *Journal of Bacteriology*, 187(19):6742–6749, oct 2005. ISSN 0021-9193. doi: 10.1128/JB.187.19.6742-6749.2005.

[30] Ainsworth, Stuart et al. Differences in Lactococcal Cell Wall Polysaccharide Structure Are Major Determining Factors in Bacteriophage Sensitivity. *mBio*, 5(3):e00880–14, may 2014. ISSN 2150-7511. doi: 10.1128/mBio.00880-14.

[31] Habann, Matthias et al. Listeria phage A511, a model for the contractile tail machineries of SPO1-related bacteriophages. *Molecular Microbiology*, 92(1):84–99, apr 2014. ISSN 0950382X. doi: 10.1111/mmi.12539.

[32] Wendlinger, Günther, Loessner, Martin J. and Scherer, Siegfried. Bacteriophage receptors on Listeria monocytogenes cells are the N-acetylglucosamine and rhamnose substituents of teichoic acids or the peptidoglycan itself. *Microbiology*, 142(4):985–992, apr 1996. ISSN 1350-0872. doi: 10.1099/00221287-142-4-985.

[33] Yasbin, Ronald E, Maino, Vernon C and Young, Frank E. Bacteriophage Resistance in Bacillus subtilis 168, W23, and Interstrain Transformants. Technical Report 3, 1976.

[34] Xiang, Ye et al. Crystallographic Insights into the Autocatalytic Assembly Mechanism of a Bacteriophage Tail Spike. *Molecular Cell*, 34(3):375–386, may 2009. ISSN 10972765. doi: 10.1016/j.molcel.2009.04.009.

[35] Li, Xuehua et al. An essential role for the baseplate protein Gp45 in phage adsorption to Staphylococcus aureus. *Scientific reports*, 6(1):26455, sep 2016. ISSN 2045-2322. doi: 10.1038/srep26455.

[36] Munsch-Alatossava, Patricia and Alatossava, Tapani. The extracellular phage-host interactions involved in the bacteriophage LL-H infection of Lactobacillus delbrueckii ssp. lactis ATCC 15808. *Frontiers in Microbiology*, 4(408), 2013. ISSN 1664-302X. doi: 10.3389/fmicb.2013.00408.

[37] Madigan, Michael T. et al. Brock Biology of Microorganisms. In Limited, Pearson Education, editor, *Brock Biology of Microorganisms*, chapter 8, pages 32–63. 14 edition, 2015. ISBN 978-0-321-89739-8.

[38] Wilkinson, Stephen G. Bacterial lipopolysaccharides—themes and variations. *Progress in Lipid Research*, 35(3):283 – 343, 1996. ISSN 0163-7827. doi: https://doi.org/10.1016/S0163-7827(96)00004-5.

[39] González-García, Verónica A. et al. Conformational Changes Leading to T7 DNA Delivery upon Interaction with the Bacterial Receptor. *Journal of Biological Chemistry*, 290(16):10038–10044, apr 2015. ISSN 0021-9258. doi: 10.1074/jbc.M114.614222.

[40] Kim, Minsik et al. Core Lipopolysaccharide-Specific Phage SSU5 as an Auxiliary Component of a Phage Cocktail for Salmonella Biocontrol. *Applied and Environmental Microbiology*, 80(3):1026–1034, feb 2014. ISSN 0099-2240. doi: 10.1128/AEM.03494-13.

[41] Washizaki, Ayaka, Yonesaki, Tetsuro and Otsuka, Yuichi. Characterization of the interactions between Escherichia coli receptors, LPS and OmpC, and bacteriophage T4 long tail fibers. *MicrobiologyOpen*, 5(6):1003–1015, dec 2016. ISSN 20458827. doi: 10.1002/mbo3.384.

[42] Marti, Roger et al. Long tail fibres of the novel broad-host-range T-even bacteriophage S16 specifically recognize Salmonella OmpC. *Molecular Microbiology*, 87(4):818–834, feb 2013. ISSN 0950382X. doi: 10.1111/mmi.12134.

[43] Zhao, Xiangna et al. Outer Membrane Proteins Ail and OmpF of Yersinia pestis Are Involved in the Adsorption of T7-Related Bacteriophage Yep-phi. *Journal of Virology*, 87(22):12260–12269, nov 2013. ISSN 0022-538X. doi: 10.1128/JVI.01948-13.

[44] Kim, Minsik and Ryu, Sangryeol. Spontaneous and transient defence against bacteriophage by phase-variable glucosylation of O-antigen in Salmonella enterica serovar Typhimurium. *Molecular Microbiology*, 86(2):411–425, oct 2012. ISSN 0950382X. doi: 10.1111/j.1365-2958.2012.08202.x.

[45] Gehring, K. et al. Bacteriophage lambda receptor site on the Escherichia coli K-12 LamB protein. *Journal of Bacteriology*, 169(5):2103–2106, 1987. ISSN 0021-9193. doi: 10.1128/JB.169.5.2103-2106.1987.

[46] Rabsch, Wolfgang et al. FepA- and TonB-Dependent Bacteriophage H8: Receptor Binding and Genomic Sequence. *Journal of Bacteriology*, 189(15):5658–5674, aug 2007. ISSN 0021-9193. doi: 10.1128/JB.00437-07.

[47] Pate, Jack L., Petzold, Sara J. and Umbreit, Thomas H. Two flagellotropic phages and one pilus-specific phage active against asticcacaulis biprosthecum. *Virology*, 94(1):24 – 37, 1979. ISSN 0042-6822. doi: https://doi.org/10.1016/0042-6822(79)90435-5.

[48] Guerrero-Ferreira, Ricardo C. et al. Alternative mechanism for bacteriophage adsorption to the motile bacterium Caulobacter crescentus. *Proceedings of the National Academy of Sciences*, 108(24):9963–9968, jun 2011. ISSN 0027-8424. doi: 10.1073/pnas.1012388108.

[49] Bae, H.-W. and Cho, Y.-H. Complete Genome Sequence of Pseudomonas aeruginosa Podophage MPK7, Which Requires Type IV Pili for Infection. *Genome Announcements*, 1(5), oct 2013. ISSN 2169-8287. doi: 10.1128/genomeA.00744-13.

[50] Oliveira, Hugo et al. Ability of phages to infect Acinetobacter calcoaceticus-Acinetobacter baumannii complex species through acquisition of different pectate lyase depolymerase domains. *Environmental Microbiology*, 19(12):5060–5077, dec 2017. ISSN 14622912. doi: 10.1111/1462-2920.13970.

[51] Pan, Yi□Jiun et al. Identification of three podoviruses infecting Klebsiella encoding capsule depolymerases that digest specific capsular types. *Microbial Biotechnology*, 12(3):472–486, may 2019. ISSN 1751-7915. doi: 10.1111/1751-7915.13370.

[52] Hashemolhosseine, Said et al. Alterations of Receptor Specificities of Coliphages of the T2 Family. *Journal of Molecular Biology*, 240(2):105–110, jul 1994. ISSN 00222836. doi: 10.1006/jmbi.1994.1424.

[53] Bradbeer, C., Woodrow, M. L. and Khalifah, L. I. Transport of vitamin B12 in Escherichia coli: common receptor system for vitamin B12and bacteriophage BF23 on the outer membrane of the cell envelope. *Journal of Bacteriology*, 125(3):1032–1039, 1976. ISSN 00219193.

[54] Roa, M. Interaction of bacteriophage K10 with its receptor, the lamB protein of Escherichia coli. *Journal of bacteriology*, 140(2):680–6, nov 1979. ISSN 0021-9193.

[55] Ricci, Vito and Piddock, L. J. V. Exploiting the Role of TolC in Pathogenicity: Identification of a Bacteriophage for Eradication of Salmonella Serovars from Poultry. *Applied and Environmental Microbiology*, 76(5):1704–1706, mar 2010. ISSN 0099-2240. doi: 10.1128/AEM.02681-09.

[56] Breyton, Cécile et al. Assessing the Conformational Changes of pb5, the Receptor-binding Protein of Phage T5, upon Binding to Its Escherichia coli Receptor FhuA. *Journal of Biological Chemistry*, 288(42):30763–30772, oct 2013. ISSN 0021-9258. doi: 10.1074/jbc.M113.501536.

[57] Andres, Dorothee et al. Tail morphology controls DNA release in two Salmonella phages with one lipopolysaccharide receptor recognition system. *Molecular Microbiology*, 83(6):1244–1253, mar 2012. ISSN 0950382X. doi: 10.1111/j.1365-2958.2012.08006.x.

[58] Perez, Gerardo L. et al. Transport of Phage P22 DNA across the Cytoplasmic Membrane. *Journal of Bacteriology*, 191(1):135–140, jan 2009. ISSN 0021-9193. doi: 10.1128/JB.00778-08.

[59] Walter, M. et al. Structure of the Receptor-Binding Protein of Bacteriophage Det7: a Podoviral Tail Spike in a Myovirus. *Journal of Virology*, 82(5):2265–2273, mar 2008. ISSN 0022-538X. doi: 10.1128/JVI.01641-07.

[60] Sandulache, Rodica et al. The cell wall receptor for bacteriophage Mu G(-) in Erwinia and Escherichia coli C. *FEMS Microbiology Letters*, 28(3):307–310, jul 1985. ISSN 03781097. doi: 10.1111/j.1574-6968.1985.tb00811.x.

[61] Parent, Kristin N. et al. OmpA and OmpC are critical host factors for bacteriophage Sf6 entry in S higella. *Molecular Microbiology*, 92(1):47–60, apr 2014. ISSN 0950382X. doi: 10.1111/mmi.12536.

[62] Manning, Paul A. and Reeves, Peter. Outer membrane proteins of Escherichia coli K-12: Isolation of a common receptor protein for bacteriophage T6 and colicin K. *Molecular and General Genetics MGG*, 158(3):279–286, jan 1978. ISSN 0026-8925. doi: 10.1007/BF00267199.

[63] Prehm, P. et al. On a bacteriophage T3 and T4 receptor region within the cell wall lipopolysaccharide of escherichia coli B. *Journal of Molecular Biology*, 101(2):277–281, feb 1976. ISSN 00222836. doi: 10.1016/0022-2836(76)90377-6.

[64] McPartland, Jennifer and Rothman-Denes, Lucia B. The Tail Sheath of Bacteriophage N4 Interacts with the Escherichia coli Receptor. *Journal of Bacteriology*, 191(2):525–532, jan 2009. ISSN 0021-9193. doi: 10.1128/JB.01423-08.

[65] Le, Shuai et al. Mapping the Tail Fiber as the Receptor Binding Protein Responsible for Differential Host Specificity of Pseudomonas aeruginosa Bacteriophages PaP1 and JG004. *PLoS ONE*, 8(7):e68562, jul 2013. ISSN 1932-6203. doi: 10.1371/journal.pone.0068562.

[66] Edwards, P. and Smit, J. A transducing bacteriophage for Caulobacter crescentus uses the paracrystalline surface layer protein as a receptor. *Journal of Bacteriology*, 173(17):5568–5572, 1991. ISSN 0021-9193. doi: 10.1128/JB.173.17.5568-5572.1991.

[67] Garbe, Julia et al. Sequencing and Characterization of Pseudomonas aeruginosa phage JG004. *BMC Microbiology*, 11(1):102, 2011. ISSN 1471-2180. doi: 10.1186/1471-2180-11-102.

[68] Yokota, S, Hayashi, T and Matsumoto, H. Identification of the lipopolysaccharide core region as the receptor site for a cytotoxin-converting phage, phi ctx, of pseudomonas aeruginosa. *Journal of*

Bacteriology, 176(17):5262–5269, 1994. ISSN 0021-9193. doi: 10.1128/jb.176.17.5262-5269.1994.

[69] Killmann, Helmut et al. FhuA Barrel-Cork Hybrids Are Active Transporters and Receptors. *Journal of Bacteriology*, 183(11):3476–3487, jun 2001. ISSN 0021-9193. doi: 10.1128/JB.183.11.3476-3487.2001.

[70] Filippov, Andrey A. et al. Bacteriophage-Resistant Mutants in Yersinia pestis: Identification of Phage Receptors and Attenuation for Mice. *PLoS ONE*, 6(9):e25486, sep 2011. ISSN 1932-6203. doi: 10.1371/journal.pone.0025486.

[71] Choi, Younho et al. Identification and characterization of a novel flagellum-dependent salmonella-infecting bacteriophage, ieps5. *Applied and Environmental Microbiology*, 79(16):4829–4837, 2013. ISSN 0099-2240. doi: 10.1128/AEM.00706-13.

[72] Lovett, Paul S. PBP1: A flagella specific bacteriophage mediating transduction in Bacillus pumilus. *Virology*, 47(3):743–752, mar 1972. ISSN 00426822. doi: 10.1016/0042-6822(72)90564-8.

[73] Lotz, W., Acker, G. and Schmitt, R. Bacteriophage 7-7-1 Adsorbs to the Complex Flagella of Rhizobium lupini H13-3. *Journal of General Virology*, 34(1):9–17, jan 1977. ISSN 0022-1317. doi: 10.1099/0022-1317-34-1-9.

[74] Yurewicz, Edward C et al. Catalytic and molecular properties of a phage-induced capsular polysaccharide depolymerase. *Journal of Biological Chemistry*, 246(18):5607–5616, 1971.

[75] Pickard, Derek et al. A conserved acetyl esterase domain targets diverse bacteriophages to the vi capsular receptor of salmonella enterica serovar typhi. *Journal of Bacteriology*, 192(21):5746–5754, 2010. ISSN 0021-9193. doi: 10.1128/JB.00659-10.

[76] Hsu, Chun-Ru et al. Isolation of a bacteriophage specific for a new capsular type of klebsiella pneumoniae and characterization of its polysaccharide depolymerase. *PLOS ONE*, 8(8):1–9, 08 2013. doi: 10.1371/journal.pone.0070092.

[77] Scholl, D. et al. Bacteriophage K1-5 Encodes Two Different Tail Fiber Proteins, Allowing It To Infect and Replicate on both K1 and K5 Strains of Escherichia coli. *Journal of Virology*, 75(6):2509–2515, mar 2001. ISSN 0022-538X. doi: 10.1128/JVI.75.6.2509-2515.2001.

[78] Labrie, Simon J., Samson, Julie E. and Moineau, Sylvain. Bacteriophage resistance mechanisms. *Nature Reviews Microbiology*, 8(5):317–327, may 2010. ISSN 1740-1526. doi: 10.1038/nrmicro2315.

[79] Nordström, Kristina and Forsgren, Arne. Effect of protein a on adsorption of bacteriophages to staphylococcus aureus. *Journal of Virology*, 14(2):198–202, 1974. ISSN 0022-538X.

[80] Hanlon, Geoffrey W. et al. Reduction in exopolysaccharide viscosity as an aid to bacteriophage penetration through pseudomonas aeruginosa biofilms. *Applied and Environmental Microbiology*, 67(6):2746–2753, 2001. ISSN 0099-2240. doi: 10.1128/AEM.67.6.2746-2753.2001.

[81] Mihara, Tomoko et al. Linking Virus Genomes with Host Taxonomy. *Viruses*, 8(3):66, mar 2016. ISSN 1999-4915. doi: 10.3390/v8030066.

[82] Zhang, Zheng et al. Phage protein receptors have multiple interaction partners and high expressions. *Bioinformatics*, 36(10):2975–2979, may 2020. ISSN 1367-4803. doi: 10.1093/bioinformatics/btaa123.

[83] Russell, Daniel A and Hatfull, Graham F. PhagesDB: the actinobacteriophage database. *Bioinformatics*, 33(5):784–786, mar 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btw711.

[84] Brister, J. Rodney et al. NCBI Viral Genomes Resource. *Nucleic Acids Research*, 43(D1):D571–D577, jan 2015. ISSN 1362-4962. doi: 10.1093/nar/gku1207.

[85] Benson, Dennis A. et al. GenBank. *Nucleic Acids Research*, 41(D1):D36–D42, nov 2012. ISSN 0305-1048. doi: 10.1093/nar/gks1195.

[86] Fadda, Elisa and Woods, Robert J. Molecular simulations of carbohydrates and protein–carbohydrate interactions: motivation, issues and prospects. *Drug Discovery Today*, 15(15-16): 596–609, aug 2010. ISSN 13596446. doi: 10.1016/j.drudis.2010.06.001.

[87] Edwards, Robert A. et al. Computational approaches to predict bacteriophage-host relationships. *FEMS Microbiology Reviews*, 40(2):258–272, 2016. ISSN 15746976. doi: 10.1093/femsre/fuv048.

[88] Stern, Adi et al. CRISPR targeting reveals a reservoir of common phages associated with the human gut microbiome. *Genome Research*, 22(10):1985–1994, oct 2012. ISSN 1088-9051. doi: 10.1101/gr.138297.112.

[89] Nielsen, H Bjørn et al. Identification and assembly of genomes and genetic elements in complex metagenomic samples without using reference genomes. *Nature Biotechnology*, 32(8):822–828, aug 2014. ISSN 1087-0156. doi: 10.1038/nbt.2939.

[90] Lindell, Debbie et al. Photosynthesis genes in marine viruses yield proteins during host infection. *Nature*, 438(7064):86–89, nov 2005. ISSN 0028-0836. doi: 10.1038/nature04111.

[91] Rohwer, Forest and Thurber, Rebecca Vega. Viruses manipulate the marine environment. *Nature*, 459(7244):207–212, may 2009. ISSN 0028-0836. doi: 10.1038/nature08060.

[92] Altschul, Stephen F. et al. Basic local alignment search tool. *Journal of Molecular Biology*, 215 (3):403 – 410, 1990. ISSN 0022-2836. doi: https://doi.org/10.1016/S0022-2836(05)80360-2.

[93] Westra, Edze R et al. The ecology and evolution of microbial CRISPR-Cas adaptive immune systems. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 374(1772):20190101, may 2019. ISSN 0962-8436. doi: 10.1098/rstb.2019.0101.

[94] Villarroel, Julia et al. HostPhinder: A Phage Host Prediction Tool. *Viruses*, 8(5):116, may 2016. ISSN 1999-4915. doi: 10.3390/v8050116.

[95] Gałan, Wojciech, Bak, Maciej and Jakubowska, Małgorzata. Host Taxon Predictor - A Tool for Predicting Taxon of the Host of a Newly Discovered Virus. *Scientific Reports*, 9(1):3436, dec 2019. ISSN 2045-2322. doi: 10.1038/s41598-019-39847-2.

[96] Wang, Weili et al. A network-based integrated framework for predicting virus-host interactions, dec 2018.

[97] Zhang, Fan et al. PHISDetector: a web tool to detect diverse in silico phage-host interaction signals. *bioRvix*, 2020. doi: 10.1101/661074.

[98] Leite, Diogo Manuel Carvalho et al. Computational prediction of inter-species relationships through omics data analysis and machine learning. *BMC Bioinformatics*, 19(S14):420, nov 2018. ISSN 1471-2105. doi: 10.1186/s12859-018-2388-7.

[99] Boeckaerts, Dimitri. *A computational study of bacteria-phage interactions to reveal determinants of phage-host specificity*. Phd dissertation, Universiteit Gent, 2018.

[100] Burnham, Kenneth P. and Anderson, David R. Model Selection and Inference. In *The Journal of Wildlife Management*, volume 65, pages 29–37. Springer New York, New York, NY, 2nd edition, jul 1998. ISBN 9781475729177. doi: 10.1007/978-1-4757-2917-7.

[101] The UniProt Consortium, . UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515, jan 2019. ISSN 0305-1048. doi: 10.1093/nar/gky1049.

[102] Malik, Adeel, Baig, Mohammad H. and Manavalan, Balachandran. Protein-Carbohydrate Interactions. In *Encyclopedia of Bioinformatics and Computational Biology*, pages 666–677. Elsevier, 2019. ISBN 9780128096338. doi: 10.1016/B978-0-12-809633-8.20661-4.

[103] Taroni, Chiara, Jones, Susan and Thornton, Janet M. Analysis and prediction of carbohydrate binding sites. *Protein Engineering, Design and Selection*, 13(2):89–98, feb 2000. ISSN 1741-0134. doi: 10.1093/protein/13.2.89.

[104] Shionyu-Mitsuyama, Clara et al. An empirical approach for structure-based prediction of carbohydrate-binding sites on proteins. *Protein Engineering Design and Selection*, 16(7):467–478, jul 2003. ISSN 1741-0126. doi: 10.1093/protein/gzg065.

[105] Sujatha, M.S. and Balaji, Petety V. Identification of common structural features of binding sites in galactose-specific proteins. *Proteins: Structure, Function, and Bioinformatics*, 55(1):44–65, 2004. doi: 10.1002/prot.10612.

[106] Jian, Jhih-Wei et al. Predicting Ligand Binding Sites on Protein Surfaces by 3-Dimensional Probability Density Distributions of Interacting Atoms. *PLOS ONE*, 11(8):e0160315, aug 2016. ISSN 1932-6203. doi: 10.1371/journal.pone.0160315.

[107] Kulharia, Mahesh et al. InCa-SiteFinder: A method for structure-based prediction of inositol and carbohydrate binding sites on proteins. *Journal of Molecular Graphics and Modelling*, 28(3): 297–303, oct 2009. ISSN 10933263. doi: 10.1016/j.jmgm.2009.08.009.

[108] Gainza, P. et al. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, feb 2020. ISSN 1548-7091. doi: 10.1038/s41592-019-0666-6.

[109] Malik, Adeel et al. PROCARB: A Database of Known and Modelled Carbohydrate-Binding Protein Structures with Sequence-Based Prediction Tools. *Advances in Bioinformatics*, 2010:1–9, 2010. ISSN 1687-8027. doi: 10.1155/2010/436036.

[110] Agarwal, Sandhya et al. Identification of Mannose Interacting Residues Using Local Composition. *PLoS ONE*, 6(9):e24039, sep 2011. ISSN 1932-6203. doi: 10.1371/journal.pone.0024039.

[111] Pai, Priyadarshini P. and Mondal, Sukanta. MOWGLI: prediction of protein–MannOse interacting residues With ensemble classifiers usinG evoLutionary Information. *Journal of Biomolecular Structure and Dynamics*, 34(10):2069–2083, nov 2015. ISSN 0739-1102. doi: 10.1080/07391102.2015.1106978.

[112] Taherzadeh, Ghazaleh et al. Sequence-Based Prediction of Protein–Carbohydrate Binding Sites Using Support Vector Machines. *Journal of Chemical Information and Modeling*, 56(10):2115–2122, oct 2016. ISSN 1549-9596. doi: 10.1021/acs.jcim.6b00320.

[113] Berman, Helen M. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, jan 2000. ISSN 13624962. doi: 10.1093/nar/28.1.235.

[114] Nassif, Houssam et al. Prediction of protein-glucose binding sites using support vector machines. *Proteins: Structure, Function, and Bioinformatics*, 77(1):121–132, 2009. doi: 10.1002/prot.22424.

[115] Khare, Harshvardan et al. Prediction of protein-mannose binding sites using random forest. *Bioinformation*, 8(24):1202–1205, dec 2012. ISSN 09738894. doi: 10.6026/97320630081202.

[116] Zhao, Huiying et al. Carbohydrate-binding protein identification by coupling structural similarity searching with binding affinity prediction. *Journal of Computational Chemistry*, 35(30):2177–2183, nov 2014. ISSN 01928651. doi: 10.1002/jcc.23730.

[117] Tsai, Keng-Chang et al. Prediction of Carbohydrate Binding Sites on Protein Surfaces with 3-Dimensional Probability Density Distributions of Interacting Atoms. *PLoS ONE*, 7(7):e40846, jul 2012. ISSN 1932-6203. doi: 10.1371/journal.pone.0040846.

[118] Eddy, Sean R. What is a hidden Markov model? *Nature Biotechnology*, 22(10):1315–1316, oct 2004. ISSN 1087-0156. doi: 10.1038/nbt1004-1315.

[119] Schäffer, Alejandro A. et al. Improving the accuracy of PSI-BLAST protein database searches with composition-based statistics and other refinements. *Nucleic Acids Research*, 29(14):2994–3005, jul 2001. ISSN 13624962. doi: 10.1093/nar/29.14.2994.

[120] Wang, Jun et al. Protein-Protein Interactions Prediction Using a Novel Local Conjoint Triad Descriptor of Amino Acid Sequences. *International Journal of Molecular Sciences*, 18(11):2373, nov 2017. ISSN 1422-0067. doi: 10.3390/ijms18112373.

[121] Awad, Mariette and Khanna, Rahul. *Efficient Learning Machines*. Number April. Apress, Berkeley, CA, 2015. ISBN 978-1-4302-5989-3. doi: 10.1007/978-1-4302-5990-9.

[122] Jordan, M. I. and Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, jul 2015. ISSN 0036-8075. doi: 10.1126/science.aaa8415.

[123] Witten, Ian H. et al. *Data Mining*. Elsevier, 4th edition, 2017. ISBN 9780128042915. doi: 10.1016/C2015-0-02071-8.

[124] Burkov, Andriy. *The Hundred-Page Machine Learning Book*. 2019.

[125] Scikit-learn, . Precision-Recall. https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html, 2019 (accessed January 23, 2020).

[126] Scikit-learn, . Metrics and scoring: quantifying the quality of predictions. https://scikit-learn.org/stable/modules/model_evaluation.html#matthews-corrcoef, 2020 (accessed January 31, 2021).

[127] Guanga, Alex. Understand Regression Performance Metrics. https://becominghuman.ai/understand-regression-performance-metrics-bdb0e7fcc1b3, 2019 (accessed January 24, 2020).

[128] Boser, Bernhard E., Guyon, Isabelle M. and Vapnik, Vladimir N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, pages 144–152, New York, New York, USA, 1992. ACM Press. ISBN 089791497X. doi: 10.1145/130385.130401.

[129] Noble, William S. What is a support vector machine? *Nature Biotechnology*, 24(12):1565–1567, dec 2006. ISSN 1087-0156. doi: 10.1038/nbt1206-1565.

[130] Scikit-learn, . RBF SVM parameters. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html#sphx-glr-auto-examples-svm-plot-rbf-parameters-py, 2019 (accessed January 25, 2020).

[131] Aha, David W., Kibler, Dennis and Albert, Marc K. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, jan 1991. ISSN 0885-6125. doi: 10.1007/BF00153759.

[132] Quinlan, J. R. Induction of decision trees. *Machine Learning*, 1(1):81–106, mar 1986. ISSN 0885-6125. doi: 10.1007/BF00116251.

[133] Cucchiara, Andrew. Applied logistic regression. *Technometrics*, 34:358–359, 03 2012. doi: 10.1080/00401706.1992.10485291.

[134] Rumelhart, David E., Hinton, Geoffrey E. and Williams, Ronald J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, oct 1986. ISSN 0028-0836. doi: 10.1038/323533a0.

[135] Shafranovich, Y. Common Format and MIME Type for Comma-Separated Values (CSV) Files. Technical report, oct 2005.

[136] Cock, P. J. A. et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, jun 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp163.

[137] Biotechnology Information (US), Bethesda (MD): National Center. The NCBI Handbook [Internet]. Chapter 18, The Reference Sequence (RefSeq) Database. https://www.ncbi.nlm.nih.gov/books/NBK21091/table/ch18.T.refseq_accession_numbers_and_mole/?report=objectonly, 2019 (accessed November 20, 2020).

[138] Biotechnology Information (US), Bethesda (MD): National Center. Accession Number prefixes: Where are the sequences from? https://www.ncbi.nlm.nih.gov/Sequin/acc.html, 2020 (accessed November 20, 2020).

[139] Saussereau, Emilie and Debarbieux, Laurent. Bacteriophages in the Experimental Treatment of Pseudomonas aeruginosa Infections in Mice. In *Advances in Virus Research*, volume 83, pages 123–141. Academic Press Inc., jan 2012. doi: 10.1016/B978-0-12-394438-2.00004-9.

[140] Yu, Nancy Y. et al. PSORTb 3.0: improved protein subcellular localization prediction with refined localization subcategories and predictive capabilities for all prokaryotes. *Bioinformatics*, 26(13): 1608–1615, jul 2010. ISSN 1460-2059. doi: 10.1093/bioinformatics/btq249.

[141] Kunduru, Bharathi Reddy, Nair, Sanjana Anilkumar and Rathinavelan, Thenmalarchelvi. EK3D: an E . coli K antigen 3-dimensional structure database. *Nucleic Acids Research*, 44(D1):D675–D681, jan 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv1313.

[142] Toukach, Philip V. and Egorova, Ksenia S. Carbohydrate structure database merged from bacterial, archaeal, plant and fungal parts. *Nucleic Acids Research*, 44(D1):D1229–D1236, jan 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv840.

[143] Tiemeyer, Michael et al. GlyTouCan: an accessible glycan structure repository. *Glycobiology*, 27 (10):915–919, oct 2017. ISSN 0959-6658. doi: 10.1093/glycob/cwx066.

[144] Aithal, Abhijit et al. PolysacDB: A Database of Microbial Polysaccharide Antigens and Their Antibodies. *PLoS ONE*, 7(4):e34613, apr 2012. ISSN 1932-6203. doi: 10.1371/journal.pone.0034613.

[145] Lütteke, Thomas et al. GLYCOSCIENCES.de: an Internet portal to support glycomics and glycobiology research. *Glycobiology*, 16(5):71R–81R, may 2006. ISSN 1460-2423. doi: 10.1093/glycob/cwj049.

[146] Wyres, Kelly L. et al. Identification of Klebsiella capsule synthesis loci from whole genome data. *Microbial Genomics*, 2(12):e000102, dec 2016. ISSN 2057-5858. doi: 10.1099/mgen.0.000102.

[147] Joensen, Katrine G. et al. Rapid and Easy In Silico Serotyping of Escherichia coli Isolates by Use of Whole-Genome Sequencing Data. *Journal of Clinical Microbiology*, 53(8):2410–2426, aug 2015. ISSN 0095-1137. doi: 10.1128/JCM.00008-15.

[148] Biotechnology Information (US), Bethesda (MD): National Center. BLAST Glossary [Internet]. https://www.ncbi.nlm.nih.gov/books/NBK62051/, 2011 (accessed December 22, 2020).

[149] Li, Weizhong and Godzik, Adam. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, jul 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl158.

[150] Li, W., Jaroszewski, L. and Godzik, A. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, 17(3):282–283, mar 2001. ISSN 1367-4803. doi: 10.1093/bioinformatics/17.3.282.

[151] Jones, P. et al. InterProScan 5: genome-scale protein function classification. *Bioinformatics*, 30 (9):1236–1240, may 2014. ISSN 1367-4803. doi: 10.1093/bioinformatics/btu031.

[152] El-Gebali, Sara et al. The Pfam protein families database in 2019. *Nucleic Acids Research*, 47 (D1):D427–D432, jan 2019. ISSN 0305-1048. doi: 10.1093/nar/gky995.

[153] Lu, Shennan et al. CDD/SPARCLE: the conserved domain database in 2020. *Nucleic Acids Research*, 48(D1):D265–D268, jan 2020. ISSN 0305-1048. doi: 10.1093/nar/gkz991.

[154] Sigrist, Christian J. A. et al. New and continuing developments at PROSITE. *Nucleic Acids Research*, 41(D1):D344–D347, nov 2012. ISSN 0305-1048. doi: 10.1093/nar/gks1067.

[155] Ross, Alexa, Ward, Samantha and Hyman, Paul. More Is Better: Selecting for Broad Host Range Bacteriophages. *Frontiers in Microbiology*, 7(SEP):1352, sep 2016. ISSN 1664-302X. doi: 10.3389/fmicb.2016.01352.

[156] Yellaboina, Sailu et al. DOMINE: A comprehensive collection of known and predicted domain-domain interactions. *Nucleic Acids Research*, 39(SUPPL. 1):730–735, 2011. ISSN 03051048. doi: 10.1093/nar/gkq1229.

[157] Klausen, Michael Schantz et al. NetSurfP□2.0: Improved prediction of protein structural features by integrated deep learning. *Proteins: Structure, Function, and Bioinformatics*, 87(6):520–527, jun 2019. ISSN 0887-3585. doi: 10.1002/prot.25674.

[158] Shen, Juwen et al. Predicting protein-protein interactions based only on sequences information. *Proceedings of the National Academy of Sciences*, 104(11):4337–4341, mar 2007. ISSN 0027-8424. doi: 10.1073/pnas.0607879104.

[159] Afgan, Enis et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2018 update. *Nucleic Acids Research*, 46(W1):W537–W544, 05 2018. ISSN 0305-1048. doi: 10.1093/nar/gky379.

[160] nsoranzo, . Planemo. https://github.com/galaxyproject/planemo, 2021 (accessed January 20, 2021).

[161] Foundation, Python Software. Python language reference, version 3.8.5. https://www.python.org, 2020 (accessed Novembert 20, 2020).

[162] Clark, Karen et al. GenBank. *Nucleic Acids Research*, 44(D1):D67–D72, jan 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv1276.

[163] Biotechnology Information (US), Bethesda (MD): National Center. Entrez Programming Utilities Help [Internet]. https://www.ncbi.nlm.nih.gov/books/NBK25501/, 2019 (accessed November 20, 2020).

[164] Biotechnology Information (US), Bethesda (MD): National Center. Pubmed [internet]. https://pubmed.ncbi.nlm.nih.gov, 2020 (accessed December 13, 2020).

[165] Foundation, Python Software. json — json encoder and decoder. https://docs.python.org/3/library/json.html, 2020 (accessed December 13, 2020).

[166] Peabody, Michael A. et al. PSORTdb: expanding the bacteria and archaea protein subcellular localization database to better reflect diversity in cell envelope structures. *Nucleic Acids Research*, 44(D1):D663–D668, jan 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv1271.

[167] Latka, Agnieszka et al. Modeling the Architecture of Depolymerase-Containing Receptor Binding Proteins in Klebsiella Phages. *Frontiers in Microbiology*, 10, nov 2019. ISSN 1664-302X. doi: 10.3389/fmicb.2019.02649.

[168] Reback, Jeff et al. pandas-dev/pandas: Pandas 1.2.0rc0, February 2020.

[169] scikit-bio. http://scikit-bio.org, 2020 (accessed December 7, 2020).

[170] Pedregosa, Fabian et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[171] Institute, Leibniz. DSMZ-German Collection of Microorganisms and Cell Cultures GmbH. https://www.dsmz.de, 2021 (accessed January 18, 2021).

[172] Standards, LGC. ATCC - LGC Standards. https://www.lgcstandards-atcc.org, 2020 (accessed January 18, 2021).

A Supporting material

The following tables contain features present in the PhageHost dataset, described in section 3.3. The last numbers in each represent the amino acid groups (from table 6), whereas the first and second strings indicate the organism and method to calculate the features. This way, "bact_groupA_0", for instance, is the frequency of the amino acid "0" in group "A", of bacterial proteins, whereas "phage_kmer_12" represents the frequency of the k-mer "12" for phage proteins.

Table 19.: Complete list of features calculated as described in section 3.3.

bact_groupA_0	phage_groupA_0	bact_groupB_0	phage_groupB_0	bact_groupC_0
phage_groupC_0	bact_groupD_0	phage_groupD_0	bact_groupE_0	phage_groupE_0
bact_groupF_0	phage_groupF_0	bact_groupG_0	phage_groupG_0	bact_groupH_0
phage_groupH_0	bact_groupI_0	phage_groupI_0	bact_groupJ_0	phage_groupJ_0
bact_groupA_1	phage_groupA_1	bact_groupB_1	phage_groupB_1	bact_groupC_1
phage_groupC_1	bact_groupD_1	phage_groupD_1	bact_groupE_1	phage_groupE_1
bact_groupF_1	phage_groupF_1	bact_groupG_1	phage_groupG_1	bact_groupH_1
phage_groupH_1	bact_groupI_1	phage_groupI_1	bact_groupJ_1	phage_groupJ_1
bact_groupA_2	phage_groupA_2	bact_groupB_2	phage_groupB_2	bact_groupC_2
phage_groupC_2	bact_groupD_2	phage_groupD_2	bact_groupE_2	phage_groupE_2
bact_groupF_2	phage_groupF_2	bact_groupG_2	phage_groupG_2	bact_groupH_2
phage_groupH_2	bact_groupI_2	phage_groupI_2	bact_groupJ_2	phage_groupJ_2
bact_groupA_3	phage_groupA_3	bact_groupB_3	phage_groupB_3	bact_groupC_3
phage_groupC_3	bact_groupD_3	phage_groupD_3	bact_groupE_3	phage_groupE_3
bact_groupF_3	phage_groupF_3	bact_groupG_3	phage_groupG_3	bact_groupH_3
phage_groupH_3	bact_groupI_3	phage_groupI_3	bact_groupJ_3	phage_groupJ_3
bact_groupA_4	phage_groupA_4	bact_groupB_4	phage_groupB_4	bact_groupC_4
phage_groupC_4	bact_groupD_4	phage_groupD_4	bact_groupE_4	phage_groupE_4
bact_groupF_4	phage_groupF_4	bact_groupG_4	phage_groupG_4	bact_groupH_4
phage_groupH_4	bact_groupI_4	phage_groupI_4	bact_groupJ_4	phage_groupJ_4
bact_groupA_5	phage_groupA_5	bact_groupB_5	phage_groupB_5	bact_groupC_5
phage_groupC_5	bact_groupD_5	phage_groupD_5	bact_groupE_5	phage_groupE_5
bact_groupF_5	phage_groupF_5	bact_groupG_5	phage_groupG_5	bact_groupH_5
phage_groupH_5	bact_groupI_5	phage_groupI_5	bact_groupJ_5	phage_groupJ_5
bact_groupA_6	phage_groupA_6	bact_groupB_6	phage_groupB_6	bact_groupC_6
phage_groupC_6	bact_groupD_6	phage_groupD_6	bact_groupE_6	phage_groupE_6
bact_groupF_6	phage_groupF_6	bact_groupG_6	phage_groupG_6	bact_groupH_6
phage_groupH_6	bact_groupI_6	phage_groupI_6	bact_groupJ_6	phage_groupJ_6
bact_comp_0	phage_comp_0	bact_comp_1	phage_comp_1	bact_comp_2
phage_comp_2	bact_comp_3	phage_comp_3	bact_comp_4	phage_comp_4
bact_comp_5	phage_comp_5	bact_comp_6	phage_comp_6	phage_kmer_00
bact_kmer_00	phage_kmer_01	bact_kmer_01	phage_kmer_02	bact_kmer_02
phage_kmer_03	bact_kmer_03	phage_kmer_04	bact_kmer_04	phage_kmer_05

Continued on next page

Table 19: continued from previous page

bact_kmer_05	phage_kmer_06	bact_kmer_06	phage_kmer_10	bact_kmer_10
phage_kmer_11	bact_kmer_11	phage_kmer_12	bact_kmer_12	phage_kmer_13
bact_kmer_13	phage_kmer_14	bact_kmer_14	phage_kmer_15	bact_kmer_15
phage_kmer_16	bact_kmer_16	phage_kmer_20	bact_kmer_20	phage_kmer_21
bact_kmer_21	phage_kmer_22	bact_kmer_22	phage_kmer_23	bact_kmer_23
phage_kmer_24	bact_kmer_24	phage_kmer_25	bact_kmer_25	phage_kmer_26
bact_kmer_26	phage_kmer_30	bact_kmer_30	phage_kmer_31	bact_kmer_31
phage_kmer_32	bact_kmer_32	phage_kmer_33	bact_kmer_33	phage_kmer_34
bact_kmer_34	phage_kmer_35	bact_kmer_35	phage_kmer_36	bact_kmer_36
phage_kmer_40	bact_kmer_40	phage_kmer_41	bact_kmer_41	phage_kmer_42
bact_kmer_42	phage_kmer_43	bact_kmer_43	phage_kmer_44	bact_kmer_44
phage_kmer_45	bact_kmer_45	phage_kmer_46	bact_kmer_46	phage_kmer_50
bact_kmer_50	phage_kmer_51	bact_kmer_51	phage_kmer_52	bact_kmer_52
phage_kmer_53	bact_kmer_53	phage_kmer_54	bact_kmer_54	phage_kmer_55
bact_kmer_55	phage_kmer_56	bact_kmer_56	phage_kmer_60	bact_kmer_60
phage_kmer_61	bact_kmer_61	phage_kmer_62	bact_kmer_62	phage_kmer_63
bact_kmer_63	phage_kmer_64	bact_kmer_64	phage_kmer_65	bact_kmer_65
phage_kmer_66	bact_kmer_66			

Table 20.: List of removed features, with recursive feature elimination. To better understand the meaning of each feature, their calculation is described in section 3.3.

phage_groupB_0	phage_groupD_0	phage_groupE_0	phage_groupF_0	bact_groupC_1
bact_groupE_1	bact_groupG_1	bact_groupH_1	bact_groupJ_1	bact_groupB_2
phage_groupC_2	bact_groupG_2	phage_groupG_2	phage_groupA_3	phage_groupC_3
phage_groupG_3	phage_groupH_3	phage_groupJ_3	bact_groupC_5	bact_groupB_6
phage_comp_0	phage_comp_5	phage_kmer_00	bact_kmer_00	bact_kmer_01
bact_kmer_02	phage_kmer_03	bact_kmer_03	bact_kmer_04	bact_kmer_05
bact_kmer_10	phage_kmer_11	bact_kmer_12	phage_kmer_14	phage_kmer_15
phage_kmer_16	phage_kmer_20	bact_kmer_20	bact_kmer_21	phage_kmer_22
bact_kmer_22	phage_kmer_24	bact_kmer_25	phage_kmer_26	phage_kmer_30
bact_kmer_30	bact_kmer_32	phage_kmer_33	bact_kmer_33	phage_kmer_35
bact_kmer_35	bact_kmer_36	phage_kmer_40	phage_kmer_41	phage_kmer_43
bact_kmer_43	bact_kmer_45	bact_kmer_46	bact_kmer_50	phage_kmer_51
phage_kmer_53	bact_kmer_54	bact_kmer_55	phage_kmer_56	bact_kmer_63
bact_kmer_66				

Table 21.: List of features that highly influence outputs for the LR model. They are given in order of importance, from the most to the least important. Only significative features are present (above 0.1 importance).

bact_kmer_25	bact_kmer_32	bact_kmer_50	bact_kmer_22	bact_kmer_65	bact_kmer_60
bact_kmer_02	bact_groupJ_1	bact_kmer_42	bact_kmer_24	bact_kmer_03	