PhageOne: Inferring the Grammar of Bacteriophage Genomes

Skylar Lysbeth Martin

Abstract

The world faces a global health crisis with the rising cases of antibiotic-resistant bacteria caused by an over-reliance on antibiotics. Bacteriophages are viruses that infect bacteria, and therefore could be used as an alternative to antibiotics. Phages replicate and evolve with their host making them a robust treatment for bacteria. The difficult part about phage therapy is that a specific phage only infects specific bacteria. Engineering phage genomes to make them more adept for phage therapy could solve this search problem. Just like a writer edits a sentence using grammatical rules, we propose a framework for inferring the grammar of bacteriophage genomes. This inference can be applied to a singular genome to allow for informed genome edits and therefore more effective phages for therapy.

A thesis presented for the degree of
Bachelors of Science

Computer Science Department
University of Colorado Boulder
04/14/2021

Contents

1 Introduction

We face a global health crisis with the rise of antibiotic-resistant bacteria (Sulakvelidze, 2001). Looking toward alternative treatments, such as bacteriophage therapy, could solve this problem. (Monteiro, 2019). Bacteriophages are double-stranded nucleic acid viruses that infect bacteria. Past work in phages is mainly motivated by two aspects: medical application, such as phage therapy (Sulakvelidze, 2001; Monteiro, 2019; Dedrick and Spencer, 2019), and expanding the understanding of virus evolution (Hatfull, 2010a,b; Hendrix, 2003).

Phage therapy uses a well-designed cocktail of phages to treat patients with bacterial infections. For example, in 2019 Dedrick and Spencer (2019) were working on finding a phage cocktail to treat a 15-year-old cystic fibrosis patient infected with recurrent M. abscessus. Unfortunately, during the time it took to find phages that infected the patient's bacterial strain the 15-year-old encountered complications requiring a lung transplant. This left her with an estimated survival of less than 1%. After three months, phages were found and a cocktail of the phages, *ZoeyJ*, *Muddy*, and *BP*, was created and administered to the patient saving her life (Dedrick and Spencer, 2019).

Phage cocktails require a mix of phages to better handle variation in the bacterial strains. Unlike antibiotics, phages evolve with their bacterial host. The cocktail's genetic diversity provides a larger pool of variation to be selected for to combat future mutants of the bacteria. Finding phages that infect a specific bacteria is a complicated problem because bacteriophages have a specific set of hosts they can bind to, known as the host range. The search problem's time complexity comes is due to the 10^{31} species of phages estimated in existence, of which 3,700 of which have been characterized(Hatfull, 2010a).

About half of the 3,700 phages are temperate which are not used in phage therapy because they integrate into the host's genome making the bacteria immune to subsequent phage infections (Hatfull, 2010a). Phages that do not integrate into their hosts are known as lytic phages. Additionally, temperate phages selective pressures for their ability to keep their host alive while they are a passenger so they often evolve to possess antibiotic-resistant and antitoxin genes (Monteiro, 2019). With the advancements of genome engineering we edit phages to lose their temperate machinery and therefore become lytic (Dedrick and Spencer, 2019; Pires, 2016). This is the case with ZoeyJ, a temperate phage, which was found to infect M. abscessus and was subsequently edited to become lytic by deleting a temperate gene (Dedrick and Spencer, 2019). The ability to edit any phage to become lytic doubles the number of phages we can utilize for therapy. Recent work has been done to alter the host range by swapping binding sites on phages (Pires, 2016). The goal of this project is create guides on how to edit a genome without causing loss of function. This knowledge could be applied to the discovery stage of phage therapy. Instead of finding multiple phages to infect a bacteria, we could find a singular phage and then permute its genome to create a set of mutant phages to be used in the cocktail.

In natural language, structure and tense follow grammatical rules in order to convey meaning, or semantics. Similarly, phage genomes require structure and co-functionality between genes to be viral. Our work exploits the 3,700 sequenced phages to find structural patterns and co-functional genes. Modeling structure and co-functionality reveals global patterns, ie the genomic grammar. This knowledge can be mapped onto a singular phage to give insight on which areas to edit without disturbing shared regulation or gene dependencies. Ultimately, these editing guides can be utilized to permute an existing phage to create artificial diversity for a phage cocktail.

There are three main methods to gain insight across multiple phage genomes: dot plots, genome maps, and phylogeny networks (Smith 2013, Pope 2015). We have generated a plot for each method across the same four phages, Arlo, Perseus, D29, and Mulan to compare each visualizations utility (1). Dot Plots are a traditional way to compare the nucleotide similarity of multiple phages. Dot plots visualize a binary matrix where i,j is True if genome 1's nucleotide at the ith position is the same as genome 2's nucleotide at jth position (figure 1a)(Krumsiek, 2007). For bacteriophages genome maps are most commonly viewed using the tool Phamerator, which displays genomes like tracks and has nucleotide similarity comparison between adjacent tracks (figure 1b) (Cresawn and Hatfull, 2011). Phylogenies abstract the entire genomes as nodes and represent their ancestral similarity to other genomes in a tree. The closer in time

the common ancestor the more similar the two genomes (Meier-Kolthoff, 2017). We want to expand on gene based visualisation, like Phamerator, and make it able to scale to thousands of genomes to identify genomic grammar.

To be able to fully take advantage of this growing repository of phages we need to create methods and visualizations that reveal patterns across thousands of genomes. Our research bridges the gap between these comparative tools to allow for cross genome structural and functional modeling. Using these models we aim to answer three questions: **what**, **where**, and **how** a phage genome should be edited.

A vital question in genome editing is **what** genes we can insert, delete, or replace, while still maintaining a functional phage. In order to maintain a functional genome we must be aware of co-functional genes, ie gene dependencies. Some genes require a partner gene to complete a common function, we consider these genes to be co-functional and co-dependent. The deletion of one co-functional genes would inhibit the other from completing their shared function. However, functionality does not necessarily need to be bidirectional. To model dependencies we first create a network of co-occurrence, where the nodes in the network are genes and the edges represent the co-occurrence of two genes in at least one genome. Then every edge can be converted to a probability, given the presence of gene i what is the probability of gene j also being present in the genome. While this method of determining dependencies may find false positives due to evolution, it is a first approach in computational prediction of dependencies.

Modeling structure helps answer the question of **where** to do an edit. The mechanism behind expression regulation, i.e. how much/when a gene should be translated into a protein, requires that geospatial organization of the genome. Genes sharing regulatory regions allows for phages to have less space used for regulation and more room to store functional genes. However, co-expression is only possible if those regions are nearby each other in the genome. Hence, there are selective pressures for a meticulously organized genome. This is why modeling the ordering of genes, is vital when considering where to edit a genome. Modeling genome structure can be done by viewing the genome ordering as a *hidden markov model* (HMM). Linguists began to model sentences using HMM but their states are words as opposed to genes. Through this

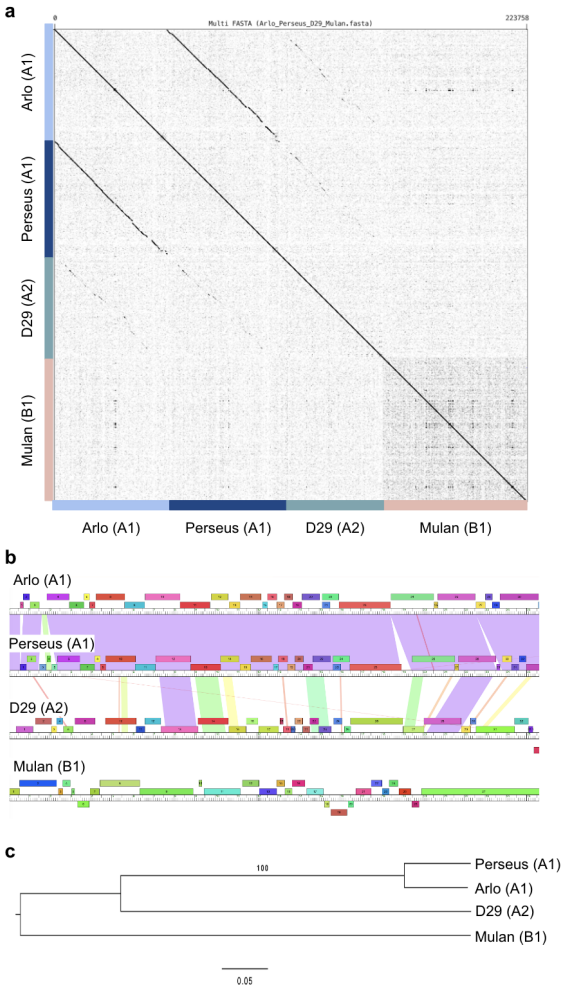

Figure 1: Genome Comparison Methods. All three methods are ran on the same genome set: Arlo (A1), Perseus (A2), D29 (A2), and Mulan (B1). To the right of each name is the sub-cluster which they belong, this grouping is by nucleotide similarity cut off of 75%. **A** shows the dot plot generated by VICTOR (Meier-Kolthoff, 2017). We can see along diagonal similarity as expected, a genome compared to itself if 100% similar. Then varying diagonal lines between phages, least of all with the three A cluster phages compared to Mulan (B1). **B** is the genome map created by Phamerator showing up to the 25,00 bases of the four genomes (Cresawn and Hatfull, 2011). Nucleotide similarity is shown through the shaded regions between the tracks. We see almost exact similarity between Persues (A1) and Arlo (A1), then some shared regions between D29 (A2) and Perseus (A1), and zero similarity between Mulan (B1) and D29 (A2). **C** shows the phylogeny created with Gepard (Krumsiek, 2007). As we would expect Arlo (A1) and Perseus (A1) are most closely related, followed by D29 (A2), and least related Mulan (B1).

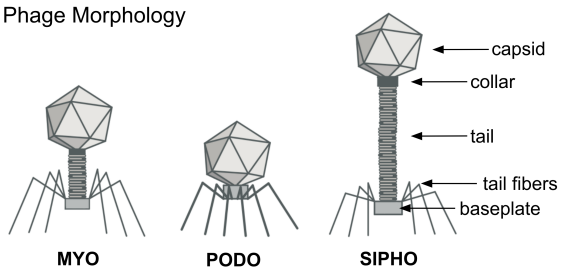

Figure 2: Bacteriophage Physical Morphology.

modeling we can discover where to take advantage of expression or where not disrupt it.

Now that we have the what and where, we must address the **how** to edit phage genomes. The final aspect of our tool is to generate oligonucleotide substrates needed for a specific genome edit. BRED stands for Bacteriophage Recombineering Electroporated DNA (Marinelli and Hatfull, 2008). BRED utilizes homologous recombination to edit phage genomes, this is the same mechanism that causes horizontal gene transfer between phages. Phage recombination can be used to alter a phage genome in three ways by either adding, deleting, or replacing a single region. There are five oligonucleotides needed for a specific edit to complete the BRED protocol, the editing substrate and four primers. The editing substrate contains 100 base pair flanking homologous arms and between them is the edit. The substrate is deterministic based on the edit. Primers must have unique binding sites, not bind to each other(hetrodimer), and not bind to itself (homodimer). Additionally, all primers must have the same melting temperatures. The complexity of material generation lies in the PCR primer design, which our algorithm addresses.

Our tool, PhageOne, thus provides a three fold purpose: to empower researchers to discover organisational/structural patterns in the genome, to utilize those findings to edit phage genomes, and based on their desired edit generate the necessary materials for the BRED protocol.

1.1 Bacteriophage Biology

Bacteriophages are double-stranded DNA viruses that infect bacteria. Phages have three **physical morphotypes**: myoviridae, podoviridae and, most common, siphoviridae (figure 2). Siphovirdae phages have a long tail while myoviridae have a short tail and podoviridae lacks a tail entirely. The **host range** of a phage is the set of bacterial species a phage can infect. A phage's virulence to a specific host is determined by its ability to bind to the bacterial cell membrane. This lock and key binding is facilitated by the phage's tail fibers binding to a receptor on the bacterial cell membrane. To infect a bacteria cell the phage first binds to the cell membrane receptor via the tail fibers. Next the phage's baseplate is lowered to bind to the receptor. Then DNA moves from the capsid down the tail and is injected into the host cell.

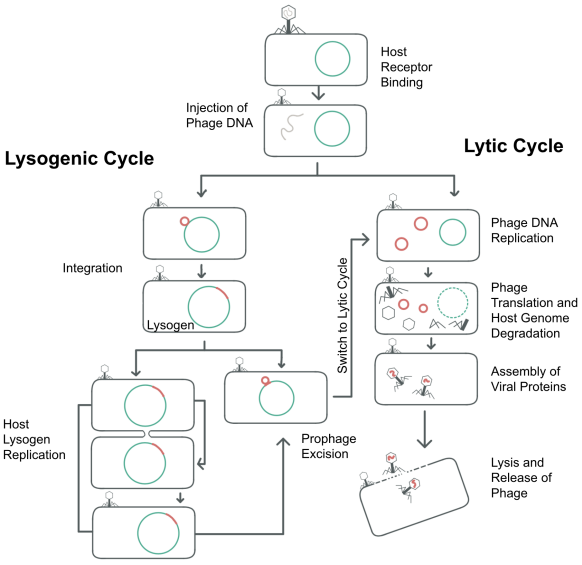

Figure 3: Bacteriophage Life Cycle Diagram.

After the initial binding and DNA injection, there are two life cycles a phage can undergo: lysogenic or lytic. The **lytic life cycle** focuses on reproduction of phage while the **lysogenic life cycle** integrates the phage genome into the bacteria (3). Lytic phages can only undergo the lytic life cycle, while temperate phages can undergo both lytic or lysogenic life cycles. Once DNA is injected into the host the temperate phage must make a lysogeny decision: will it enter the lytic or lysogenic life cycle? The lysogenic life cycle starts with the phage integrating DNA into the host's genome, via the **integrase** protein, and then acting as a passenger to the host cell's replication. The integrated phage genome is referred to as a **prophage** and its host a **lysogen**. In order to prevent unintentional degradation of the host membrane the phage expresses the **repressor** gene to repress lytic genes. An environmental trigger can cause this temperate phage to excise, cut-out, its genome from the host and move into the lytic life cycle. The lytic life cycle begins with the degradation of the host genome and the hijacking of host machinery to replicate the phage's DNA. Next phage proteins will begin to be

made and self assemble into phages. Once the new phages are assembled within the bacteria the phage will express the **lysin** genes to lyse (cleave open) the cell membrane. These newly created phages then go on to infect bacterial hosts and start their own life cycles (Hatfull, 2010a). Needless to say, the complexity of the phage life cycle requires the precise regulation of the genes in each stage.

The different nature of these two life cycle types create different selective pressures on the phage's genome (Monteiro, 2019). Temperate phages are able to replicate with the bacteria which could be an effective form of survival in sparsely populated environments. Bacteria with an integrated phage genome is known as a **lysogen** and the integrated phage genome is known as a **prophage**. Temperate phages also have evolved to have genes that increase host fitness, like antibiotic resistance and anti toxin genes (Monteiro, 2019). The repressor the prophage expresses not only acts on itself but also represses incoming phages with similar repressor operator sites, which is known as **lysogen immunity**. This immunity is what makes temperate phages a worse candidate for phage therapy than lytic phages.

The complexity of both life cycles requires coordinated gene regulation. Elements involved in regulation are enhancers, repressors, and promoters. Enhancers or repressors attract or block transcriptional machinery to/from the promoter of a gene. They can also affect expression of an entire region of the genome, ie multiple promoters. If one regulatory regions can regulate a set of genes then it is more space efficient for the genome to be organized based on life cycles stage. Additionally, it is shown in bacterial genomes that multiple proteins can share one promoter (Nijveen, 2012). A promoter is the region directly upstream of the gene which attracts machinery to transcribe the gene into RNA. Sharing promoters or translation start sites requires that genes be directly adjacent to each other. When transcribing a shared promoter the transcriptional machinery can bind to a gene then once it gets to the end it can hop to the next gene as opposed to detaching from the DNA and having to re bind to a promoter. The mechanisms behind all three of these regulatory elements make it advantageous for the genome to be organized by life cycle stage. We can exploit this principle to model the grammar of the genome.

Phage genomes are relatively short, typically

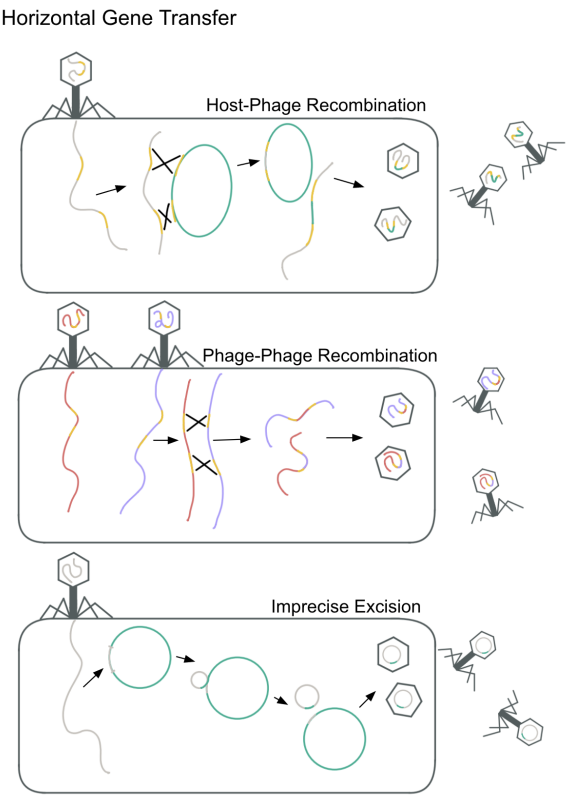

Figure 4: Mechanisms of Horizontal Gene Transfer.

less than 100,000 bp long, making them a good model genome to expand the scientific community's knowledge of horizontal evolution (Hatfull, 2010b). A mechanism of their evolution is via *horizontal gene transfer* (HGT), between phages to phages and phages to hosts (Hatfull, 2010a). HGT occurs via imprecise excision and **homologous recombination** (figure 4). When a prophage switches to the lytic life cycle they excise from the host genome. This excision can be faulty leaving part of the prophage in the host and part of the host genome in the phage. Another mechanism for HGT is homologous recombination. Recombination occurs between phage-phage and host-phage. For this to occur there must be two homologous regions in both the phage and the other genome. The homologous regions bind then the differing regions between the homology are swapped. This swapping between genomes via HGT causes phage genomes to be highly mosaic in their contents.

1.2 Related Work

HHMI's Science Education Alliance-Phage Hunters Advancing Genomics and Evolutionary Science (**SEA-PHAGES**) program started in 2008 with 12 universities partnering to lead semester

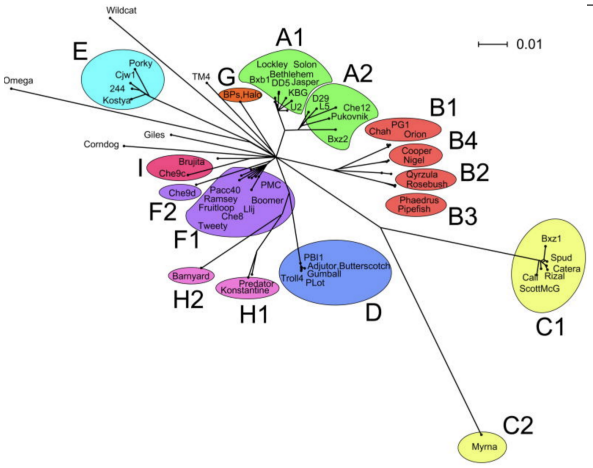

Figure 5: Clustering of 60 Phage Genomes Hatfull (2010b)

long phage discovery and phage genomics labs. This program has lead to the mass discovery of phages by undergraduate students across the country (Hatfull, 2010b). Characterized phage **lysates** (phages suspended in buffer) were sent to the Hatfull lab in Pittsburgh and starting in 2015, data on these phages were stored on **PhagesDB** (Russell and Hatfull, 2016) The SEA-PHAGES program along with their public database have allowed for large computational studies of phages genomes.

One of the first studies utilizing this data was by Hatfull (2010b) who studied 60 Mycobacteriophages and clustered them based on their nucleotide similarity. To minimize single phage clusters and maximize similarity, cluster and subcluster cut off was determined to be over 50% and 75% respectively. Nucleotide similarity gives equal importance to each base regardless if that base ends up coding for a protein. A more recent comparative study in 2015 by Pope WH (2015) clustered over 600 phages using the same method, but also investigated intra-cluster shared phams. A phams is a gene cluster based on protein sequence calculated using the Phamerator tool (Cresawn and Hatfull, 2011). Pham clusters were computationally found by maximizing protein similarity and minimizing **orphams**, single gene phams. Comparing shared phams allows us the better understand two phages functional similarities and moves away from giving value equally across the entire genome.

Recent computational work for phages focuses on host prediction(Arndt, 2016; Edwards et al., 2015; Cheng et al., 2018; Li et al., 2020; Villarroel, 2016) and automatic functional annotation(Arndt,

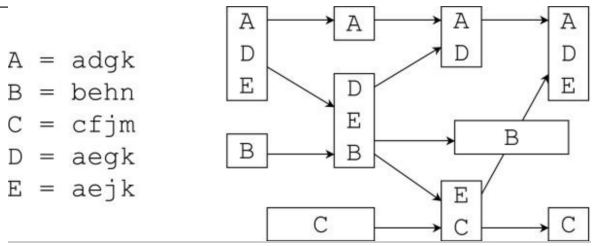

Figure 6: Mosaic graphs Belcaid and Poisson (2010)

2016; McNair K., 2018; Ecale Zhou et al., 2019; Ramsey et al., 2020). For our approach of modeling both co-functionality and co-regulation are primarily studied from wet lab work. Structural modeling focuses on **synteny**, which is the co-location or ordering of genes in a genome. Synteny modeling in the literature is motivated by modeling mosaicism caused by horizontal gene transfer, new metric of similarity between phage genomes, and to infer gene function. There are three pieces of work we will discuss *Mosaic Graphs and Comparative Genomics in Phage Communities* by Belcaid and Poisson (2010), *Phagonaute: A web-based interface for phage synteny browsing and protein function prediction* by Delattre (2016), and *Gene Network Visualization and Quantitative Synteny Analysis of More than 300 Marine T4-Like Phage Scaffolds from the GOS Metagenome* by André M Comeau (2010). These focus on different approaches to visualizing and constructing synteny networks.

Belcaid and Poisson (2010) creates mosaic graphs to better understand recombination modules, ie regions swapped between phages via HGT. Before constructing their graph they identify their modules. These modules are nucleotide sequences at least 400 base pair that are present across multiple genomes (meaning exactly or at a high nucleotide similarity). This work is unlike the other two papers, because they do not limit their modules to genes, and instead focus on regions. This is more biologically accurate because the mechanism of homologous recombination is oblivious to where genes are located. Thus, the module approach models homologous recombination more accurately. Once they have determined their modules they construct a graph using them where the directed edges represent adjacency of modules within genomes. A toy model of this network is shown in figure 6. In the paper they use these mosaic graphs to create recombination networks between phages. These are constructed using each genome as a node and shared modules between two genomes as edges

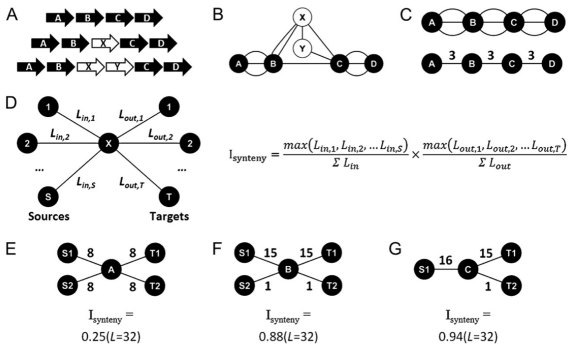

Figure 7: Synteny Graph Method André M Comeau (2010)

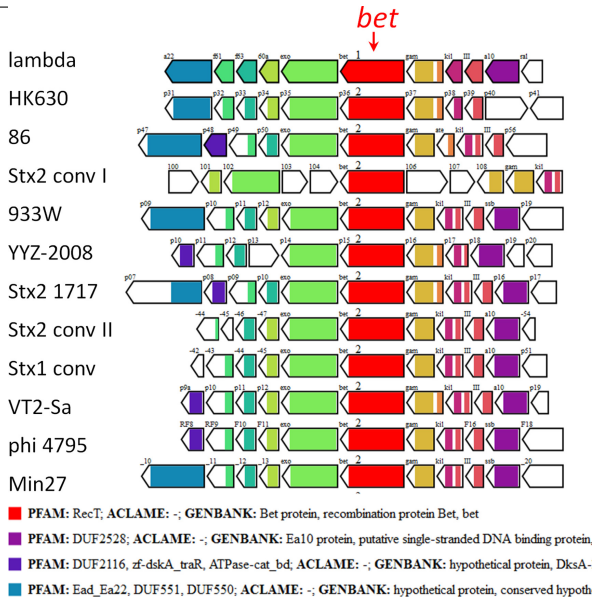

Figure 8: Phagonaute's Synteny Genome Maps Delattre (2016)

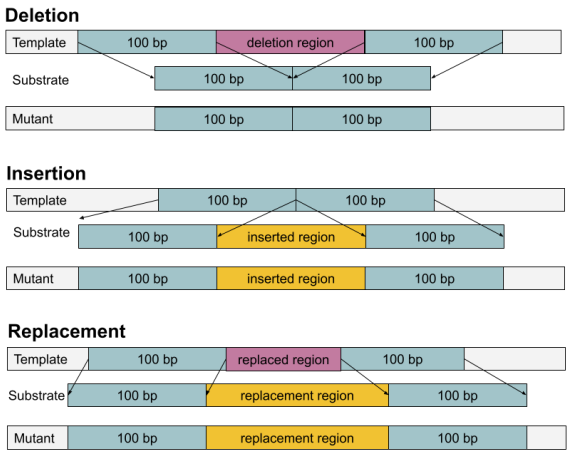

Figure 9: BRED Editing Substrates in Relation to the Type of Edit.

(Belcaid and Poisson, 2010). Our work with synteny focuses on utilizing synteny models to understand gene regulation, therefore, for our purposes we use genes as modules.

André M Comeau (2010) created a framework for building gene-level synteny networks and made a method to quantify synteny of a particular gene in the network. They created these networks using the genomes of over 300 Marine T-4 like phages. To focus on modeling important regions on the genome, ie the core genome, they remove any genes that are ORFans (genes that appear in only one genome). They construct a network where edge i to j represents gene i being upstream of gene j and the weight of the edge is how many genomes that adjacency occurs in (figure 7). Their Index of Synteny is the max weight of an in-edges divided by the sum of all weights of in edges multiplied by the max weight of an out edge divided by the sum of all weights of out edges. This methodology allows for aggregation of genomes to find common structural patterns. Their metric does not take into account the number of genomes and gene exists in, for example if a gene exists in one genome then automatically its synteny score is 1. While a gene present in hundreds of genome that always has the same upstream and downstream gene also has a score of 1. We approach this inequality by visualizing both the frequency of the gene and edge weights.

In Delattre (2016) they created an application, Phagonaute that combines homology search and synteny to visualize the context of a gene. The input for their application is the phage name, gene product number, number of neighbors *n*, and nucleotide similarity probability cut off. Then the region, previous n neighbors, the gene of interest, and the next n neighbors, is searched against the phages genomes in their database. The result is displayed much like Phamerator's genome map with a genome on each row and homology between regions indicated in matching colors. The main conclusion for this work is how the context of a gene could give insight to a unknown gene's function and their relationship to other genes. We want to expand on this notion by extending our co-functionality search to the entire genome opposed to just a discrete number of neighbors. This allows the user to be exploratory with synteny instead of having to know which specific gene might be of interest.

To take full advantage of our editing guides we

also want to assist the user in the editing process by generating the required oligonucleotides. Recombination is utilized to edit phage genomes, this method is known as Bacteriophage Recombineering of Electroporated DNA (BRED) developed by Marinelli and Hatfull (2008). BRED requires an electrocompetent host with recombination machinery, phage DNA and an editing substrate. BRED occurs most similarly to phage-phage recombination with two sets of DNA injected into the host.

To prerequisite material for the BRED protocol is creating a electrocompetent host with recombination machinery. To make this host they introduce a plasmid to the bacteria to add recombination genes. This plasmid, pjV53, also contains kanamaycin antibiotic resistant gene which allows them to select for bacteria that had successful uptake of the plasmid da Silva (2013). After preparing materials, the first step in the BRED protocol is transporting both sequences into the host cell. This transportation is done by electrophoration which is where we shock the cell which increases the membrane permeability allowing for DNA to pass through. Then if the editing substrate was correctly designed it should bind to the phage DNA and recombine. Examples of the editing substrate per edit type is shown in figure 9. The experiments done by Marinelli and Hatfull (2008) expect less than 10% of phage genomes to take up the edit. Due to this small percentage the time intensive aspect of this protocol is isolating the recombinant strain.

2 Data

The data for this project is made available by SEA-PHAGE's Actinobacteriophage Database, *PhagesDB* (Russell and Hatfull, 2016). PhagesDB hosts data collected by the SEA-PHAGES program. PhagesDB currently has 18,258 discovered phages, of which 3,731 are sequenced. There are three main data sets for this project: phage metadata, phage gene data, and fasta files. Phage metadata includes their cluster, sub-cluster, most common cluster life cycle, physical morphotype, genome length, annotation status. Morphotype and cluster are not uniformly distributed among sequenced phages, as shown in figure 10. The phage gene data contains the gene product number, protein sequence, functional annotation (if genome is annotated and gene has known function), gene pham, translational start/stop location, translation direction. The fasta files store the phages' DNA sequences.

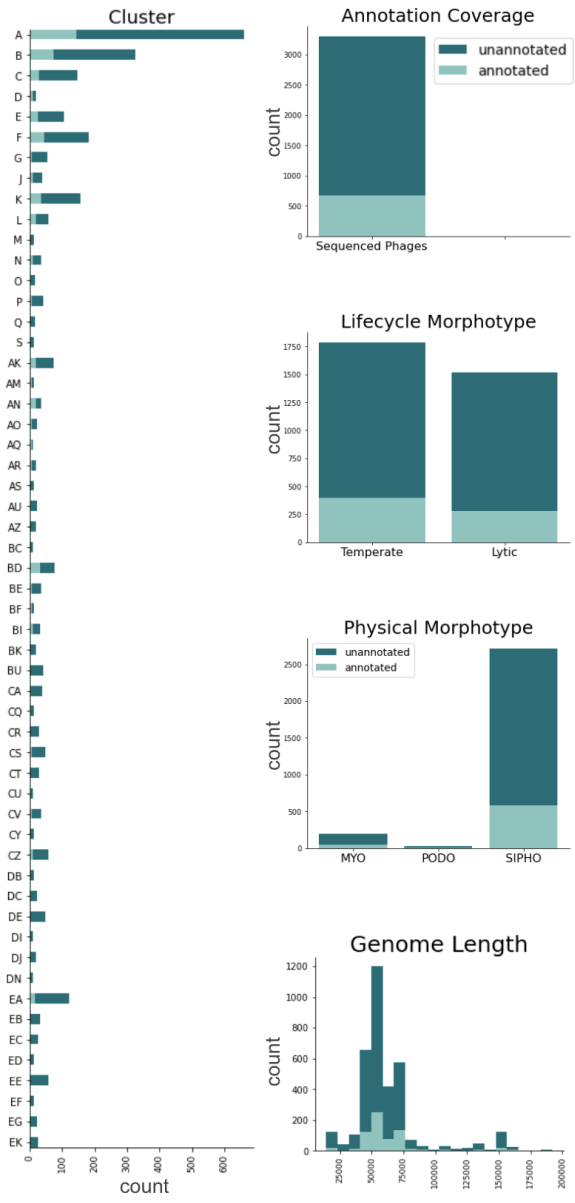

Figure 10: Sequenced Phages Distribution

3 Approach

3.1 Representations of a Gene

The first step in natural language processing is to clean the text and convert it into smaller bits, commonly referred to a tokens and the process is called tokenization (Loper and Bird, 2002). Words are often tokenized based on their semantic meaning by removing modifiers. For example, *ran*, *run*, *running* are all tokenized to *run*. The method of tokenization is a trade off between capturing overall meaning and specificity. If we have the same number of words in the text as tokens, then we are unable to extract meaning from the text. Just like words in a sentence we have ordered lists of genes in a genome. Similarly we must decide how to represent a the gene.

A gene's semantic meaning is their function. While their syntax is their nucleotide sequence. Multiple nucleotide sequences can code for one amino acid (protein) sequence since most amino acids are determined by the first two nucleotide. Which makes proteins fairly robust to mutations in the nucleotide sequence. Proteins carry out functions, so we can start with tokenizing to the protein sequence. In our data set there are $348,734$ total genes and $155,704$ unique protein sequences. Two higher level descriptors that are more general than protein sequence are a gene's pham and function. There are $23,224$ unique phams, and $1,668$ unique functions. Function is a manually annotated description of what the protein does. Only annotated phage genomes have functional annotations and the majority of genomes are not annotated (figure 10). Additionally the majority of genes in phage genomes have *no known function*s (NKF). Also protein function does not encode compatibility. For instance, the major capsid and collar protein are common among most phage, however, functionally a capsid from an A cluster phage might be incompatible with a collar from an E cluster phage. For these reasons we believe that function is disregards compatibility and exclusive of well researched genes. Pham is a clustering a genes based on protein sequence, which allows for genes to be clustered based on functional and molecular makeup. Hence, for our purposes we will use phams as gene tokens. Function will be used as a explanatory label and for visualizations, but because of their bias in annotation, not for modeling purposes.

Figure 11: Function Cleaner

3.2 Data Cleaning

The SEA-PHAGE program maintains a list of 229 approved functions. The unique number of functions in phagesDB's gene data is $1,668$. Functional annotations are manually decided by students and researchers and are therefore prone to human error. A common problem with functional annotations were misspellings or synonyms, for instance, the approved function *lysin A* was represented in the data set as *lys A* and *lsyin A*. To resolve these errors we manually corrected all non-approved function names shared by more than one gene. We created a script that assisted this correction process. The script extracted a list of phams that were assigned to a gene with the ambiguous function name. For each pham, the functions assigned to genes in that pham that are in the approved function list are added to an array of possible functions. Then this list was mapped to a dictionary where the key is the approved function name and the value is the number of times it appeared on the possible functions list. Then the script displays this dictionary as a series of rows sorted by value, and the user can pick which function most accurately disambiguates the given function name. An example of the function cleaner script output is shown 11 figure.

3.3 Modeling Co-Regulation

In the Spanish language, we expect the adjective after the noun, while in English it is expected before the noun it modifies. Just like ordering is important to meaning in language, it is vital to genomes. Promoters can be shared in genes and therefore it

requires those to be directly adjacent. Genes close in proximity can share regulatory regions, like enhancers or silencers. Disruption in these regions can create incorrect expression which inhibits the ability of the phage to progress through its life cycle.

To model co-regulation we focus on the ordering of genes, this is known as synteny. Synteny network's nodes represent a gene and a directed edge between gene i to gene j is weighted by the probability of gene i being directly upstream of gene j. To construct this graph we use a matrix to store the edges for a quick lookup time of both in-degree and out-degree distribution. For *n* unique gene identifiers we make a lookup table that maps them [0,n-1] indices. Each genome is made into an ordered list where sorted by gene product number but with the value of the gene token. Then we traverse gene by gene down each genome while keeping track of the previous and current gene to add to matrix A, A[prev,curr]. After all genomes have been traversed, the matrix A[i,j] is the number of genomes that the $i \rightarrow j$ adjacency exists in. To alter the elements into probabilities each row of A is divided by its row sum. The row sum for each row is stored as the gene count array. This is the framework for building a synteny network for the entire database of genomes. We find that synteny, the probabilities of adjacencies, are highest at the beginning of the genome. Past work has referred to the beginning half as the core genome while the latter is known as hyperplastic (due to HGT) Hatfull (2010a). This aligns with our analysis.

3.4 Modeling Co-Functionality

Similar to how an adverb can depend on a verb or a pro-noun depends on a noun, genes depend on each other to complete a function. Modeling co-functionality requires us to extrapolate which genes have dependencies on each other to complete their function. For the lysogenic life cycle, we could imagine that the excise is dependent on the integrase because how could a region be excised from a genome if it was unable to be integrated. To model this we assume that all genes in our model need to function for a viral genome to exist. First we cant represent each genome as a bag of genes, ie order doesn't matter. Then we can make a network that represents the co-occurrence of genes. Nodes in this network represent genes and are weighted by the total number of that gene across all genomes. An edge i to j is undirected and represents gene i and gene j co-occur in at least one genome. Each edge is weighted by the number of genomes in which that co-occurrence is present.

We assume 2 types of dependency: bidirectional and unidirectional. Bidirectional dependencies are shown when in the co-occurrence graph when two vertices, i and j, $i_{weight} = (i,j)_{weight} = j_{weight}$, ie the genes are always present together. Unidirectional dependence of i on j is when $i_{weight} = (i,j)_{weight} \neq j_{weight}$, ie i always appears with j but j does appear without i. A dependency network can be made with these filtering rules to the co-occurrence network. To make this switch the co-occurrence matrix is viewed to be directed, so each edge i-j becomes $i \rightarrow j$ and $i \leftarrow j$. If a node and one of its out-edges share the same weight then that edge is a dependency and we add it to the dependency network. This can be done by dividing each row by the row sum and only keeping values of 1. This step should sound familiar from the construction of the synteny network, we are finding the probability that given gene i we see gene j. Then we are left with a directed weighted network where each edge represents dependency.

3.5 Editing Guides

Editing a specific sentence requires a broader knowledge of grammar, Grammar in NLP can be learned utilizing a training set of correct sentences. Both the synteny and dependency networks give us this context for phage genomes. In order to utilize this information we can map both networks onto a singular genome.

For the dependency network this mapping is done by taking the union of genomes in our target genome with the genes in the dependency graph. With the subset of nodes all edges connected to that subset are extracted. Visualizing the dependency network is more clear in matrix form using the heat map visualization format. The element (i,j) represents a that gene j depends on gene i and is colored by the number of genomes containing this dependency. We also visualize two bars, parallel to the row and column. These are the a 1 dimensional heat map summing the row or column they lie on. This can show whether is gene is more often dependent on by others or dependent on others. This can visually tell us which genes are important and therefore we should not delete. Likewise it shows which regions are not depended on, which could be

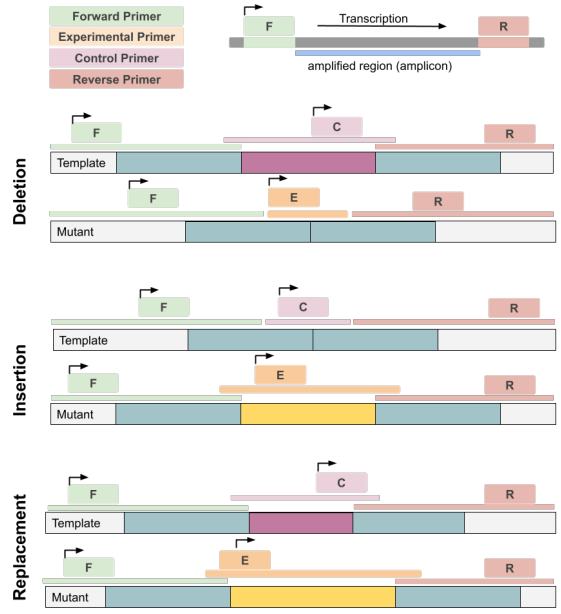

Figure 12: Designing Primers for BRED Protocol. This figure shows the search space for each primer for the four primer types: forward, control, experimental, and reverse.

swapped out to permute the phage genome.

To obtain the synteny mapping onto a genome we take the subset of nodes that exist in our target genome. This is visualized by mapping a genome in a line and changing the thickness of edges based on the probability of that adjacency. The node size is based on the number of genomes that gene exists in. This allows us to visually evaluate the importance of the probability relative to the total number of genomes. For instance if the upstream gene i is only present in one genome then is edge will be conserved. That edge is less significant than a upstream gene present in 200 genomes that has conserved adjacency.

3.6 BRED Material Generation

The BRED protocol requires a substrate along with control and experimental primer sets. The substrate contains the edit template nucleotide sequence flanked by 100bp homologous arms. For example, if we want to delete the region 200:300bp from genome G (a string of nucleotides) then our substrate would $G[100:200] + G[300:400]$ with an empty editing template. The replacement of the region with sequence S requires the substrate $G[100:200] + S + G[300:400]$. If we want to insert S in the location 300 base pair, then our substrate would be $G[200:300] + S + G[300:400]$. As shown in these examples, based for a given edit there is only one possible editing substrate sequence, making this part of the problem quite easy.

This problem becomes more complex for the primer generation because primers do not have exact location, they just need to be located within a region. In BRED a set of two primers are used for PCR to amplify regions of DNA then each amplicon can be run on a gel. An amplicon is the region between the two primers (figure 12) and the amplicon's length can be experimentally determined by running a gel. Running a gel of the PCR product for the experimental primer set allows us to check if our edit was taken up in the phage genomes. This is important because we can validate if we have isolated our mutant phage by running both experimental and control primer sets. If there is only amplification from the experimental set but no amplification from control the set, then we know we only have mutant phages in our isolate.

In BRED we need 4 primers, of which makeup 3 primer sets to be used in PCR. The forward primer (F) needs to bind with both unedited/edited DNA, the control (C) that should only bind the unedited DNA, and the experimental (E) should only bind to the edited DNA. The reverse (R) should bind to both the unedited/edited DNA and will be used as a downstream primer of the other three primers (figure 12). The region that the primer can bind, ie the search space for our primer decision, is shown in figure 12. If a search region is 100 bases long and our primer needs to be 20 bases then we have 80 unique primers for that region. If we have a 100 base region for each of the four 20 base primers then we have a total of (80+80+80)*80 primer sets to chose from. The primer set is chosen by optimizing characteristics of the set. Most importantly all four primers need to have the same melting temperature so they are compatible for PCR temperature cycles. We want to minmize the number of primers, we can do this by reusing the reverse for all three upstream primers. However, this is only possible if all four primers in a set have the same melting temperature. Then we need to access whether the primers will self bind (homodimerize), and if the F,E,C will bind with the R primer (heterodimerize). We find these attributes using Primer3. First we minimize temperature range to the first decimal point, then homodimer and hererodimer scores by the thousandth place. The top ranking primer

set is provided to the user through the PhageOne application.

3.7 PhageOne Application

The BRED material generator and editing guides are both made available on the web application, PhageOne. The user can clone the code from github.com/sky123martin/PhageOne and step through README.md directions to run the application. This github also hosts all the jupyter research notebooks created for the rest of the analysis, stored under the folder 'research_notebooks'. The application uses the WSGI web app micro framework, Flask, and was developed using test driven development. The front end uses bootstrap for the CSS template and D3 for visualization. The application setup starts by downloading all the phage DNA sequences, phage metadata, and gene information files. In total this sums to about 1GB of data, so both these tasks are multi-threaded to speed up the download. Then dependencies and synteny networks across all the genomes with the pham tokenizer are constructed and stored in the GML format.

The methods stated in the previous subsection are used to generate BRED materials given a specific edit. The user can input: phage name, edit type (replacement, insertion, deletion), edit location (start/stop bp or gene product number), DNA template (input sequence or eGFP), and translation direction of inserted/replaced template DNA (F, R, match replacement gene). Then the application calculates and serves the results for the primers and substrate. Additionally, the user can download a .csv of all possible primer sets if their ranking for these attributes differ from ours.

The application can also be used to create editing guides. These guides are displayed with interactive visualizations that are generated in browser using JavaScript's D3. In browser construction of visuals reduces server calls and allows for interactivity that a pre-made figure would lose. To search editing guides the user only needs to input the phage's name.

The full genomes synteny and dependency graphs made during setup are not viewable by our application because of the sheer size of the networks, over 20,000 nodes and 300,000 edges. This amount of data is difficult to visualize and inhuman to process, so we didn't surface them on PhageOne.

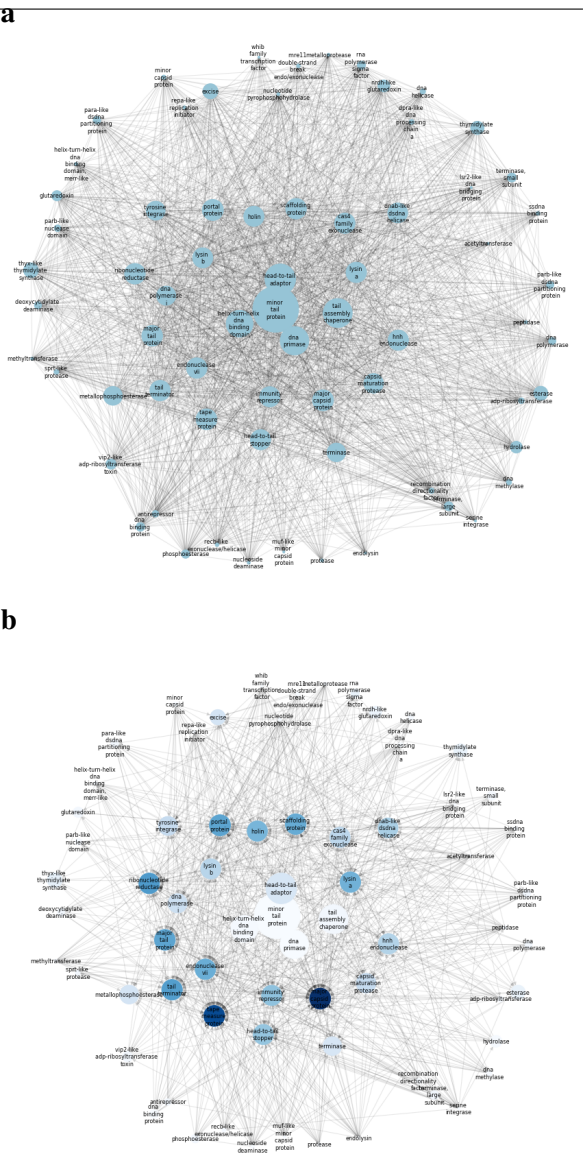

Figure 13: Co-Occurrence and Dependency Networks of A2 sub-cluster phages tokenizing by Function.

4 Results

First, we will go over general statistics for our underlying dependency and synteny networks. Then we will show the output on PhageOne of these networks mapped onto a singular genome, we refer to these as editing guides. Finally we discuss the BRED material generator the second component to the PhageOne application.

4.1 Dependency and Synteny Networks

The co-occurrence network was created across all 3,731 genomes using pham to tokenize the genes. The undirected and weighted network has a total of 23,011 nodes (genes) and 2,603,892 edges

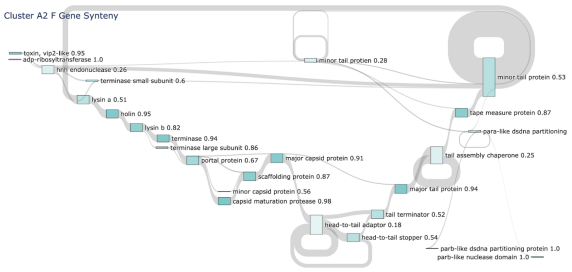

Figure 14: Synteny Network

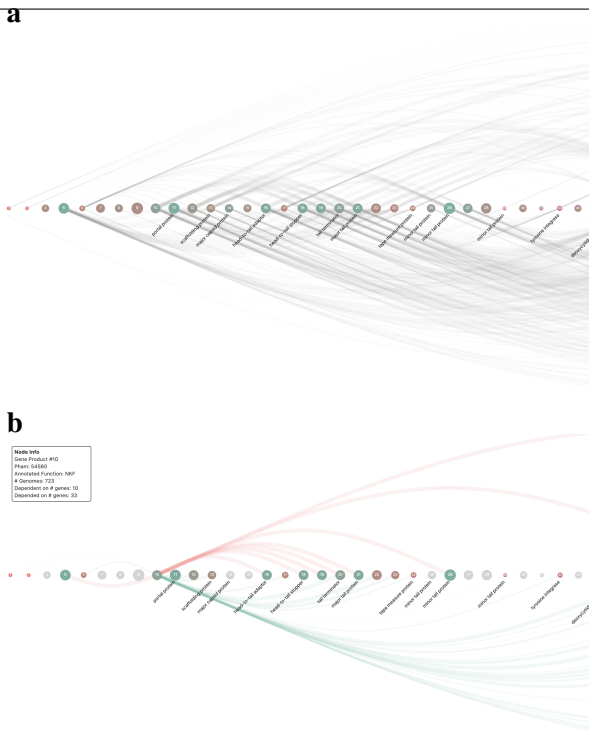

Figure 15: PhageOne Application: Dependency Network Displayed on D29 (A2) Phage in Flow Diagram Visualization. (a) shows flow diagram without any user hover. (b) shows when user hovers over a specific gene. The radius of each circle represents how many genomes that gene is present in.

(co-occurrences). The co-occurrence network was then converted to the directed dependency network. The dependency network 23,011 genes and 2,104,313 edges. The number of edges from the co-occurrence network only slightly decreased. This is likely due to how a co-occurrence edge that represents a bidirectional dependency converts into two edges in the dependency network. Reciprocity in the dependency network is 0.317, meaning that 31.7% of the dependencies are bidirectional. The average number of dependencies connected to a node, i.e. the average number of both in-edges and out-edges, is 183. About 5% of genes solely depend on others (0 in-edges), 3% are solely depended on (0 out-edges), and 92% are both depended on and dependent on others.

To show the structure of the co-occurrence and dependency networks we rebuilt them on the A2 sub-cluster with function as the tokenizer, as shown in figure 13. We use function tokenizer so we have a small enough amount of nodes to make a readable network. The size of the nodes represent the number of genomes they appear in. In the dependency network, the shade of the nodes represent the in-degree centrality, i.e. the number of other genes dependent on that gene. As we expect there is a central core to both the networks. But it is also shown that the most central nodes in the co-occurrence network when seen from the dependency perspective are not depended on by other genes. For instance, a central node in the co-occurrence network is the minor tail protein. However, when converted to the dependency network we see that it has few dependencies, while the smaller node major capsid protein has a far greater number of dependent genes. This shows the importance of conversion from the co-occurrence to the dependency network.

To model co-regulation we constructed a synteny network across all genomes using the pham token. The total number of nodes in our network is 23,011 nodes and 44,950 edges. Average each gene is adjacent to 3.9 other unique gene tokens. To visualize this network we rebuilt it with the A2 sub-cluster using function as the gene tokenizer as shown in figure 14. In this visualization we are able to see the variability of conserved ordering down the genome. This aligns with literature's belief of a core region followed by a hyperplastic one.

4.2 PhageOne Application

The PhageOne application hosts the BRED material generation and the editing guides. Over this section we will be discuss the visualizations and utility of the web application.

The editing guides hosted on PhageOne are constructed from dependency and synteny networks. These larger reference networks are constructed using all genomes and pham as the tokenizer. The editing guides take the dependency/synteny networks and map a singular phage genome. The user inputs the name of the phage through a form and the application returns editing maps for that phage.

The dependency editing guide is visualized in

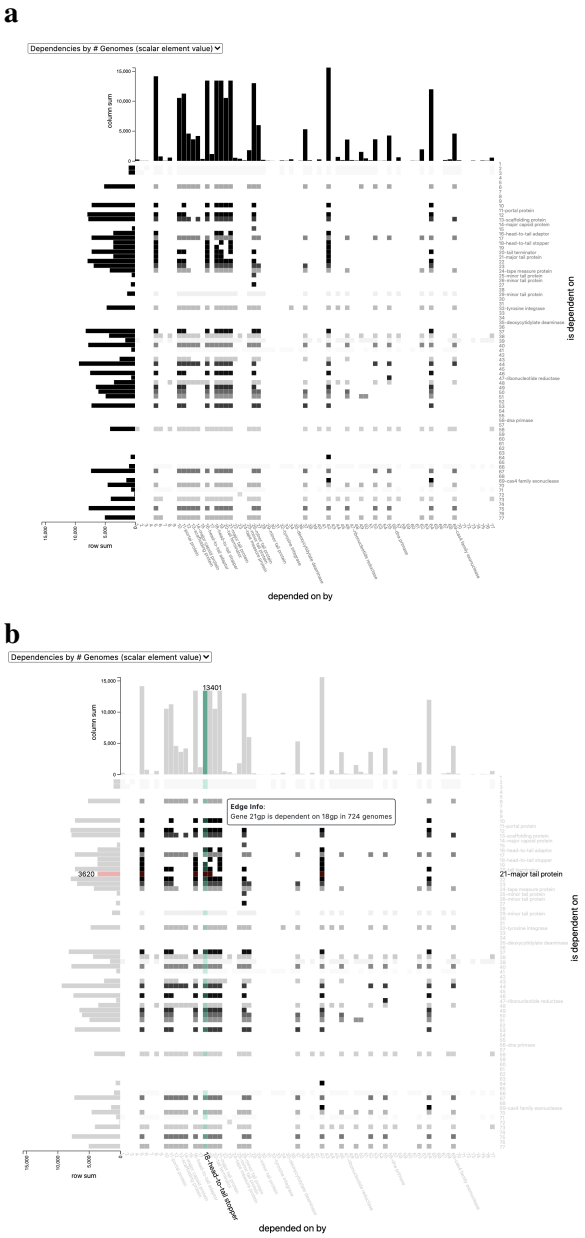

Figure 16: PhageOne Application: Dependency Network Displayed on D29 (A2) Phage in Matrix Visualization. Subfigure (a) shows the matrix without user interaction. (b) shows interactivity when user hovers over edge. Element i,j represents the dependency from gene i to gene j and is weighted by the number of genomes containing this dependency. The bar plots on the top and the left side of the matrix show the column and row sum respectively.

two ways, a flow diagram (figure 15) and a matrix (figure 16). The flow diagram shows each gene as a node in the order they appear in the target phage's genome. This diagram is helpful to understand where dependencies are flowing to and out of. We see a lot of edges under the nodes meaning that dependencies seem to be located at the beginning of the genome.

Looking at this diagram we can decide which genes will have least impact to delete, ie the least dependence upon genes. For instance, we see that the first two genes are present in only a few genomes and have no genes that are dependent on them. This makes gp 1 and gp 2 good candidates to delete or replace to mitigate the edits affect on phage functionality. Another question we might want to ask is how important is a gene, this can give us insight to which genes should be functionally researched. This is a difficult task using the flow diagram, so instead we can visualize the network as a heat map (figure 16). If we look at the column sum at the top of the matrix we can see how many instances a certain gene is depended on by others. In D29, gene product 5 is solely depended on by others and does not depend on other genes. Despite its importance in the gp 5 doesn't have an annotated function. Therefore, we might want to spend resources discovering what its function is because it seems to be important to so many other genes. The dependency network can not only assist in editing but also in prioritization of genes for functional research.

The synteny editing guide is visualized as a flow diagram. This network allows us to ask questions about where co-regulation occurs. In D29 we see a small amount of thick edges between nodes in the first half of the genome (figure 17a). While farther down in the genome we observe more complex structure shown by the increase of edges of less weight (figure 17b). In our visualization we can hover over this gene and reveal its upstream and downstream adjacency's (figure 17c). This holds with the hypothesis of hyper-plasticity occurring later in the genome. A question we can ask of this network is where we should make an edit. Using synteny we can identify conserved regions, either to take advantage of their regulation or not disrupt it with out edit. In D29, we could do a edit later in the genome where it is more plastic as to not disrupt regulation. In D29, if we want our gene to be expressed with structural proteins we can add

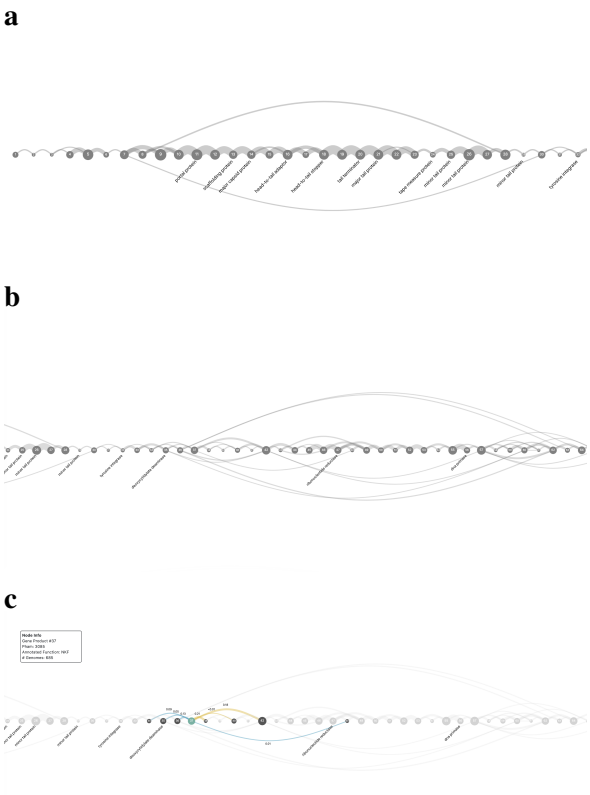

Figure 17: PhageOne Application: Synteny Network Displayed on D29 (A2) Phage in Flow Diagram Visualization. This diagram is laid-out similar to the dependency flow diagram but with different edges. The node size represents the number of genomes that gene exists in. The edge i to j represents the probability of gene i being upstream of gene j. When the user hovers on a node the connected edges and associated nodes are highlighted. Blue edges represent an incoming edge and yellow edges represent an out going edge. When highlighted the probability of each adjacency is annotated above the edge.

it to the end of the conserved region after gp 28 (figure17).

The other portion of the application assists the user in generating oligonucleotides for the BRED protocol. The user inputs their desired edit into the form. Based on the edit the form takes different inputs. After the selection insertion edit type the form will display inputs for the base location, orientation, and inserted DNA. For replacement the form requires the user to input the location (either in gp # or start/stop bp), orientation, and replacement DNA. For deletion the user has to input the location in either a gene product number of start/stop. If the user's inputs are validated by the application then they are redirected to the results page. The results page visualizes all the materials of the edit in spacial relation to each other (figure 18). Additionally all substrates and primers are displayed with their attributes.

5 Discussion

As previous work has emphasized, we need to start viewing genomes in the context of others. With the SEA-PHAGES program and their expanding data set we are able to make inference across thousands of genomes. With the constant influx of new genes into phages via HGT it is important to differentiate parts of the genome that are vital to the function of the phage, whether that be regulatory regions or highly depended upon genes. In doing this we are able to make more informed edits. The complexity of primers design can defer people from undergoing certain protocols. The BRED material generation allows the biologist spend less time on computation and refocus on their experiments.

The dependency network is a novel method on this data and can lay the framework for more complex models of dependency. For example, our approach to the construction this network does not take into account if a gene has a functional synonym. Despite this we can still learn about which genes have high importance to other genes. Which genes are dependent on it can give us clues to its own function. It can also provide scientists a way to rank which genes would have the most impact to our knowledge of phages. We can follow the dependencies in the network to construct functional mutant phages that can be used together in a phage cocktail.

The scientific community is still trying to learn more about identifying regulatory regions, specif-

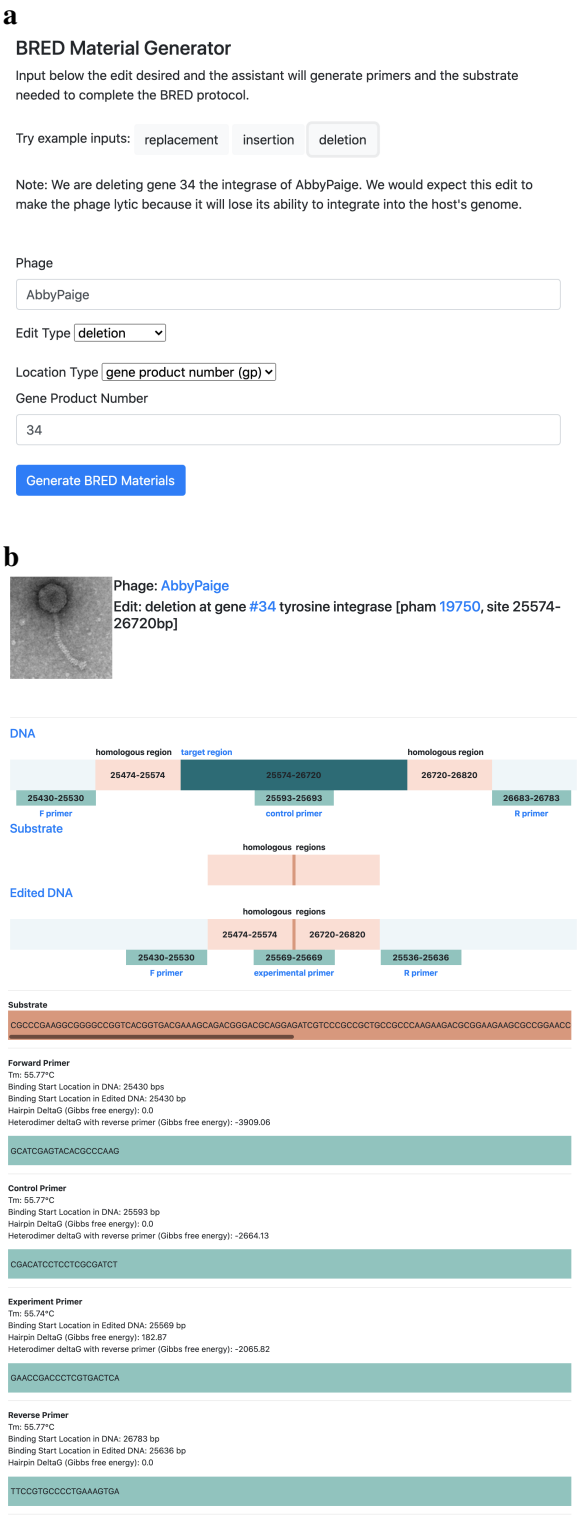

Figure 18: PhageOne Application: BRED Material Generation Form and Results.

ically promoters. Our synteny networks could be used as an additional attribute for the prediction of a promoter. Because we don't know exactly how to identify a promoter designing one is even less understood. Using the synteny network we can hijack shared promoters and let the phages regulation do the work.

With our current set of data we are unable to learn which dependencies lead to temperate or lytic phages. This is because life cycle is only recorded at a cluster level. This work could be extended to analyze the difference between dependency networks of temperate vs lytic phages. This could help us predict key players in the temperate life cycle. With this information we can make more extensive edits to temperate phages to insure they a robustly temperate. With HGT if a mutant phage is in the presence of its ancestral phage then it could have the deleted gene swapped back in. Deleting multiple genes allows us to reduce the probability of a switch back to temperate.

6 Future Directions

This project can extend in two aspects: building out the application and validating computational findings in the wetlab. The application only includes phage subsets of the synteny and dependency networks. Filtering by sub-cluster could allow us to have a small enough network subset to render in browser. We are moving forward to run wet lab experiments using BRED. These tests will validate both the BRED material generator and the validity of synteny/dependency networks.

We plan to execute three experiments in collaboration with the Living Matter Lab. First, we will replace the repressor gene with a florescent gene (eGFP) from temperate phage Perseus. Deletion of the repressor was done in ZoeyJ and we know that this edit will result in a change from temperate to lytic (Dedrick and Spencer, 2019). This is so we can check our protocol is working and the material generator creates the proper substrate. Second, we will insert a fluorescent gene (eGFP) without a promoter at the end of a conserved region. We expect our inserted gene to take advantage of shared promoter mechanics. If our hypothesis is correct and our gene is being expressed, we can expect to see fluorescent spots in our recombinant phages plate. Thirdly, we plan to delete a gene with no dependencies. We expect this edited phage to continue being functional. If this is true, we should be able

to isolate it and validate the edit using PCR with the experimental primer set.

7 Acknowledgments

My fascination with phages and their complex genomes started my junior spring semester in the Phage Genomics lab under the guidance of Dr. Nancy Guild and Megan Greening. This thesis was done in collaboration with the Living Matter Lab directed by Dr. Mirela Alistar. In the LM Lab, I have been able to design wetlab protocols and do bench work under the mentorship of Drecey Albin. Over the past couple of years my advisor, Dr. Aaron Clauset, has taught me how to think about and model complex systems. I am so grateful for all of my mentors' time and resources they have provided for me to undertake this thesis.

References

H M Krisch André M Comeau, Christine Arbiol. 2010. Gene network visualization and quantitative synteny analysis of more than 300 marine t4-like phage scaffolds from the gos metagenome. *Molecular biology and evolution*, 27(8).

Marcu Sajed Pon Liang Wishart Arndt, Grant. 2016. Phaster: a better, faster version of the phast phage search tool. *Nucleic acids research*, 44(W1).

Bergeron Belcaid and Poisson. 2010. Mosaic graphs and comparative genomics in phage communities. *Journal of computational biology : a journal of computational molecular cell biology*, 17(9).

Jing-Hui Cheng, Hui Yang, Meng-Lu Liu, Wei Su, Peng-Mian Feng, Hui Ding, Wei Chen, and Hao Lin. 2018. Prediction of bacteriophage proteins located in the host cell using hybrid features. *Chemometrics and Intelligent Laboratory Systems*, 180:64–69.

Day Jacobs-Sera Hendrix Cresawn, Bogel and Hatfull. 2011. Phamerator: a bioinformatic tool for comparative bacteriophage genomics. *BMC Genomics*, 12(395).

Garlena-Russell Ford Harris Gilmour Soothill Jacobs-Sera Schooley Hatfull Dedrick, Guerrero-Bustamante and Spencer. 2019. Engineered bacteriophages for treatment of a patient with a disseminated drug-resistant mycobacterium abscessus. *Nat Med*, 25.

Fagoonee Guerois-Petit Delattre, Souiai. 2016. Phagonaute: A web-based interface for phage synteny browsing and protein function prediction. *Virology*, 496:42–50.

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

Robert A. Edwards, Katelyn McNair, Karoline Faust, Jeroen Raes, and Bas E. Dutilh. 2015. Computational approaches to predict bacteriophage–host relationships. *FEMS Microbiology Reviews*, 40(2):258–272.

Hatfull. 2010a. Mycobacteriophages: Genes and genomes. *Annual Review of Microbiology*, 64(1):331–356. PMID: 20528690.

Lawrence-Pope Russell Ko Weber Patel Germane-Edgar Hoyte Bowman Tantoco Paladin Myers Smith Grace Pham O'Brien Vogelsberger Hryckowian Wynalek Donis-Keller Bogel Peebles Cresawn Hendrix Hatfull, Jacobs-Sera. 2010b. Comparative genomic analysis of 60 mycobacteriophage genomes: Genome clustering, gene acquisition, and gene size. *Journal of Molecular Biology*, 397(1):119–143.

Smith Hendrix, Hatfull. 2003. Bacteriophages with tails: chasing their origins and evolution. *Research in Microbiology*, 154(4):253–257.

Rattei Krumsiek, Arnold. 2007. Gepard: a rapid and sensitive tool for creating dotplots on genome scale. *Bioinformatics (Oxford, England)*, 23(8).

M. Li, Y. Wang, F. Li, Y. Zhao, M. Liu, S. Zhang, Y. Bin, A. I. Smith, G. Webb, J. Li, J. Song, and J. Xia. 2020. A deep learning-based method for identification of bacteriophage-host interaction. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, page 63–70, USA. Association for Computational Linguistics.

Swigoňová Balachandran-Oldfield an Kessel Marinelli, Piuri and Hatfull. 2008. Bred: a simple and powerful tool for constructing mutant and recombinant bacteriophage genomes. *PloS one*, 3(12).

Pusch G.D. Overbeek R. Dutilh B.E.-Edwards R. McNair K., Aziz R.K. 2018. Phage Genome Annotation Using the RAST Pipeline. *Methods in Molecular Biology*, 1681.

Göker Meier-Kolthoff. 2017. VICTOR: genome-based phylogeny and classification of prokaryotic viruses. *Bioinformatics*, 33(21):3396–3404.

Costa Azeredo Monteiro, Pires. 2019. Phage therapy: Going temperate? *Trends in Microbiology*, 27(4):368–378. Special Issue: Antimicrobial Resistance and Novel Therapeutics.

Passel Nijveen, Matus-Garcia. 2012. Promoter reuse in prokaryotes. *Mobile Genetic Elements*, 2(6):279–281. PMID: 23481313.

Sillankorva Azeredo Lu Pires, Cleto. 2016. Genetically engineered phages: a review of advances over the last decade. *Microbiol Mol Biol Rev.*, 80(3).

Russell DA Jacobs-Sera D Asai DJ Cresawn SG Jacobs WR Hendrix RW Lawrence JG Hatfull GF Pope WH, Bowman CA. 2015. Whole genome comparison of a large collection of mycobacteriophages reveals a continuum of phage genetic diversity. *Elife*.

Jolene Ramsey, Helena Rasche, Cory Maughmer, Anthony Criscione, Eleni Mijalis, Mei Liu, James C. Hu, Ry Young, and Jason J. Gill. 2020. Galaxy and apollo as a biologist-friendly interface for high-quality cooperative phage genome annotation. *PLOS Computational Biology*, 16(11):1–19.

Daniel A Russell and Graham F Hatfull. 2016. PhagesDB: the actinobacteriophage database. *Bioinformatics*, 33(5):784–786.

Broussard Marinelli Bastos Hirata-Hatfull Hirata da Silva, Piuri. 2013. Application of BRED technology to construct recombinant D29 reporter phage expressing EGFP. *FEMS Microbiology Letters*, 344(2):166–172.

Morris Sulakvelidze, Alavidze. 2001. Bacteriophage therapy. *Antimicrobial Agents and Chemotherapy*, 45(3):649–659.

Jurtz Zschach Lund Nielsen Larsen Villarroel, Kleinheinz. 2016. Hostphinder: A phage host prediction tool. *Viruses*, 8(5).