UNIVERSITY OF CALIFORNIA,
IRVINE

Fixing What's Wrong: The Problem of Coding Anomalies in Genomic Data

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computational Science

by

Katelyn McNair

Dissertation Committee:
Professor Anca Segall, Co-Chair
Professor Xiaohui Xie, Co-Chair
Professor Rob A Edwards
Professor Alex Ihler
Associate Professor Katrine Whiteson

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my committee chair, Rob Edwards, who has the attitude and the substance of a genius: he continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Without his guidance and persistent help this dissertation would not have been possible.

I would like to thank my committee members, Professor Xiaohui Xie and Professor Anca Segall, whose work demonstrated to me that concern for global affairs supported by an "engagement" in comparative literature and modern technology should always transcend academia and provide a quest for our times.

In addition, a thank you to Professor Katrine Whiteson and Professor Alex Ihler, who introduced me to Linguistics, and whose enthusiasm for the "underlying structures" had lasting effect.

Chapter 2 of this dissertation is a reprint of the material as it appears in Oxford University Press Bioinformatics 5(22), 2019, 4537–4542, used with permission from The Authors. Chapter 3 of this dissertation is a reprint of the material as it appears in MDPI Microorganisms 2021, 9, 12, used with permission from The Authors.

VITA

Katelyn Alyssa McNair

2001 B.S. Cellular and Molecular Biology, San Diego State University

2008 M.S. Computational Science, San Diego State University

FIELD OF STUDY

Computational Biology

ABSTRACT OF THE DISSERTATION

Fixing What's Wrong: The Problem with Frameshifts in Genomic Data

by

Katelyn McNair

Doctor of Philosophy in Computational Science

University of California, Irvine, 2023

Professor Anca Segall San Diego State University, Chair

Professor Xiaohui Xie University of California, Irvine, Co-Chair

Genomic sequencing is at the forefront of biological research, in part due to the ever-increasing accessibility of sequencing technology. What used to take an entire lab devoted solely to nucleotide sequencing, can now be accomplished by a single person on a tiny USB device plugged into an old laptop. Around 80% of this data is microbial in nature: originating from Bacteria, Archaea, or viruses. Generally, the goal of this sequencing is to identify and analyze the genes that occur within the DNA (or RNA). This is because it is the genes that carry out the functions and are the building blocks of every cell. Frameshifts, where coding sequences switch between frames on the collinear strand, are ubiquitous in this genomic data, as both artificially induced sequencing error and naturally occurring cellular processes. These frameshifts break the genes into different frames, which confound downstream gene prediction analysis, since all current genome-based analyses of microbes uses methods that are remiss when it comes to identifying these frameshifts. In order to improve current genomic research, we developed

software that is able to identify both artificial and natural frameshifts within an input genome using a convolutional neural network to classify fixed-size windows taken from a genome in a scrolling manner. Windows that come from a coding gene are classified as such while windows taken from intergenic regions or out-of-frame with a coding gene are not classified as coding. The individual window predictions are then analyzed by a change-point algorithm to detect when a coding gene begins or ends.

INTRODUCTION

Genomic sequencing is at the forefront of most biological research, partly due to the ever-increasing accessibility of sequencing technology**.** What used to take an entire lab devoted solely to nucleotide sequencing, can now be accomplished by a single person on a tiny USB device plugged into an old laptop. The largest repository for sequence data, NCBI's Sequence Read Archive (SRA) is currently around 36 petabytes; and has seen exponential growth, doubling every year, which means roughly twenty trillion nucleotide bases are added every single day. We have previously shown that around 80% of this data is microbial in nature: originating from Bacteria, Archaea, or viruses [1]. This trend is also seen in NCBI's repository for full genomes, where currently there are 348k prokaryotic genomes, 44k viral genomes, but only 19k eukaryotic genomes. Some of this disparity is caused by the size of respective genomes, since eukaryotes tend to be in the gigabases range, while prokaryotes and viruses tend to be in the megabases and kilobases range. However, other factors also influence these numbers, for example prokaryotes and viruses are haploid, and the genes of eukaryotes are often organized into introns (coding bases) and exons (noncoding bases). The reason so much research is focused on microbes, is that they are involved in almost every facet of human life; from health, to industry, to environment, and yet we know so very little of the full breadth of genomic diversity and community complexity. Generally, the end goal of sequencing an organism is to identify and analyze the genes that occur within the DNA. This is because it is the genes that carry out the functions and are the building blocks of every cell.

In the early years of genomic sequencing technology, the presence and location of a gene was proved through experimental evidence using genetic engineering and proteomics. But as the rate of sequencing increased exponentially, and computing power caught up, gene finding became more predictive in nature. One approach is to use the sequences of experimentally validated genes and to find similar genes within an unknown genome and infer that they likewise code for a protein. In contrast to empirical homology-based gene finding are *ab-initio* methods that use intrinsic properties and clues of protein coding genes in order to predict their occurrence in an unknown genome. To understand the tell-tale signs and basic layout of a protein coding gene we turn to the 'central dogma' of molecular biology, which describes the interplay between the genome sequence and the genes, where information flows from DNA, to RNA, to protein. One strand of the DNA serves as a template, and a portion of the DNA is transcribed into RNA by RNA polymerase. The length of the transcribed region varies, beginning at a promoter site in the DNA which the polymerase binds to and continues until the polymerase reaches a terminator sequence causing disassociation of the polymerase from the DNA [2].

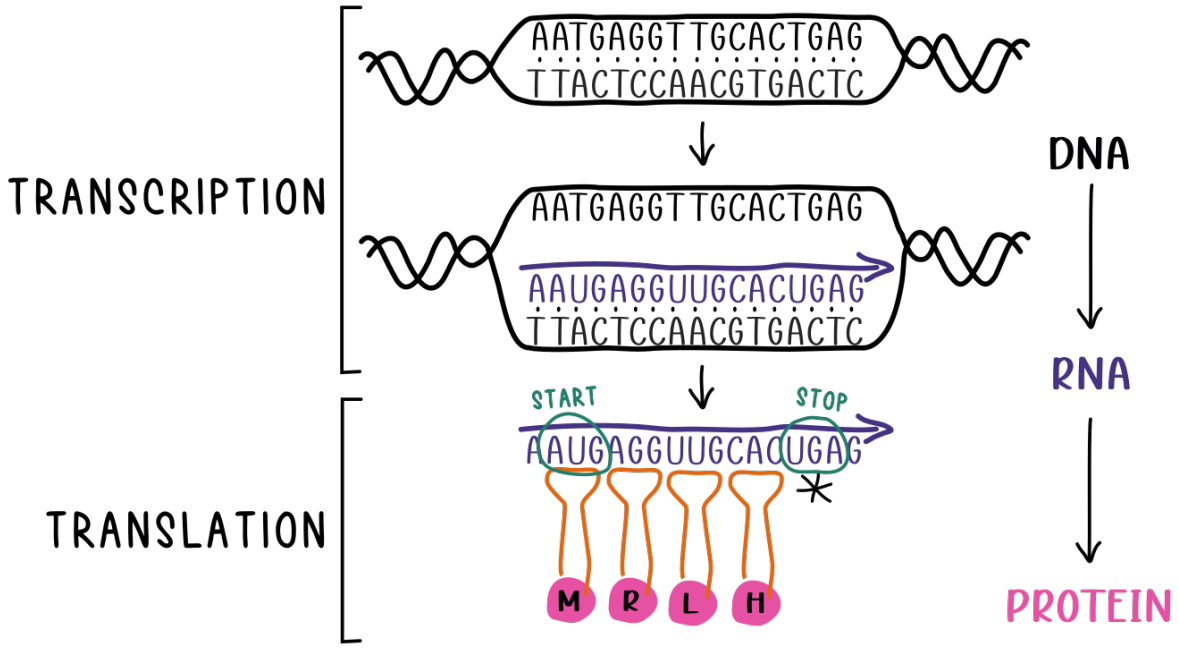


Figure 1 The flow of genetic information in the Central Dogma of molecular biology

Contained within the newly transcribed RNA are one or more genes. Some genes code for RNA molecules that perform cell functions, like transfer RNA (tRNA) and ribosomal RNA (rRNA), but the vast majority are protein coding genes. The basic arrangement of a coding sequence (CDS) is a start codon, followed by a stretch of nucleotides in triplets that are translated into amino-acids, followed by a stop codon (Figure 1). Translation begins when a ribosome binds to the RNA just upstream of a start codon which orients the start codon in the vicinity of its catalytic site. A special initiator methionine tRNA is recruited and bound through complementary base pairing to the start codon. The anti-codon for the Met-tRNA is UAC, which is why the most commonly used start codon is AUG ($\frac{UAC}{AUG}$), followed by the less common near cognate codons GUG and UUG that are possible due to weak but permissible binding of G:U and U:U base pairing ($\frac{UAC}{GUG}$ and $\frac{UAC}{UUG}$). Translation of the gene progresses as aminoacyl-tRNAs are matched to

their cognate nucleotide triplet on the RNA and the amino acids carried by the tRNAs are joined together to form a polypeptide chain of the protein encoded by the gene. Termination of protein synthesis occurs when the ribosome encounters a nucleotide triplet that does not have a matching tRNA, which is known as a stop codon, and the most common are UAG, UGA, and UAA. There are exceptions to these, as well as start codons, and their usage changes based on the evolutionary history and GC content of a genome. Additionally, some bacteria and phages even carry genes for cognate tRNAs that match one of the stop codons, thereby allowing ribosomes to "readthrough" stop codons. This abridged version of the central dogma, in addition to excluding RNA to DNA retroviruses, glosses over the step of RNA processing. Eukaryotes have introns/exons, whereby after RNA is transcribed, it is chopped up and spliced together differently before being translated. This complicates gene detection in eukaryotes, because their genes are fragmented across the DNA sequence. In contrast, prokaryotes and phages do not have introns/exons, for the most part, so the sequence of a gene in an RNA transcript is a 1 to 1 mirror of that sequence in the DNA. This collinearity allows us to forego the RNA step and looking for genes directly in the DNA of bacteria, archaea, and phages.

Not every occurrence of the nucleotide triplets matching start and stop codons actually encode for a gene. Due to the six-frame nature of RNA, *start* and *stop* triplets occur semi randomly in frames that do not happen to encode for proteins. Additionally, the triplets matching *start* codons (AUG, GUG, UUG) regularly occur *in-frame* in the middle of genes where they encode for their respective amino-acid (methionine, valine, leucine). In contrast, *stop* codons cannot occur *in-frame* in the middle of protein coding genes since they would cause translation to stop, and this stretch between a *stop* codon and the next *in-frame stop* codon is called an open-

reading frame (ORF). It is the maximum length of consecutive non *stop* codons, and is therefore where protein coding genes can occur. All that is needed for an ORF to contain a putative *coding* gene is that one of the nucleotide triplets that serve as *start* codons also occurs *in-frame* (with the stop codons).

Gene finding/prediction is the process of identifying the *true* protein coding genes from among those spurious *start*-to-*stop* codon occurrences that exist randomly from chance alone and do not actually encode for a protein. Various methods were initially proposed that used purine to pyrimidine codon bias [3] or the maximum chi-squared of codon frequencies [4]. The first computer program to actually be developed and released used a set of known protein coding genes to calculate an observed codon preference, and then iterated through sliding windows of an input genomic sequence in order to distinguish which windows come from a *coding* frame and which come from *noncoding* frames [5]. In lieu of a set of known genes, a set of the longest ORFs could be used instead. Results were promising; however, the code was only tested on the single genome of bacteriophage phiX174, the only completely sequenced genome at the time that only contains ten genes [6]. The program was never given a name, and was intended as an initial proof of concept for a more complete gene detection method that would also include start codon location, ribosomal binding site prediction, and splice junctions to determine if a region codes for a protein.

When GenMark came out in '93, gene finding really came into its own, both commercially and artistically. The algorithm, later renamed GeneMark, used species-specific inhomogeneous Markov models trained on protein coding genes from known genomes to predict the occurrence of genes in a new unknown genome [7]. Over time, successive improvements were

made to the original algorithm and a multitude of GeneMark variations were released, as well as spawning a whole host of competing tools that also used Markov models to predict coding genes [8]–[13]. Alternate methods for gene finding were implemented, the most notable being Prodigal, which uses multiple coding signatures such as codon GC usage, codon bias, and ribosomal binding sites to predict the occurrence of coding genes [14]. The one thing that all these gene callers share in common is that they rely on the assumption that every protein coding gene begins with a start codon and ends with stop codon. They operate by first looking for one of the three most common start codons (AUG, GUG, and UUG), following it to the next in-frame stop codon to get the set of all open reading frames (ORFS), then using various computational methods to distinguish which ORFs are indeed protein coding, and which ORFs do not code for a protein.

CHAPTER 1

1. Introduction

Phages provide unique challenges for bioinformatics. There is a limit to how much DNA can be packaged in a capsid, and therefore phage genomes are generally short, typically in the range 20–100 kb. By necessity, their genomes are compact: phage genes are shorter than their bacterial homologs are frequently co-transcribed, and adjacent open reading frames (ORFs) often overlap [15]. In a few cases, phage genes are encoded within each other [16], [17]. In contrast, bacterial genes generally are longer, separated by intergenic spacers and frequently switch strands [15]. There are no bioinformatics tools specifically designed to identify genes in phage genomes, so algorithms designed to identify bacterial genes are typically used [18]. For example, from 31 phage genomes published between October 14, 2016 and August 1, 2018, the genes in ten phage genomes were identified by GeneMark software (GeneMark/GeneMarkS/GeneMark.hmm), the genes in 10 phage genomes were identified by RAST, the genes in 7 phage genomes by Glimmer, 3 phage genomes each by Geneious [19], the NCBI ORF Finder [20], PHAST (which uses Glimmer as a gene caller) [21], PROKKA (which uses Prodigal as a default gene caller) [22], 2 phage genomes by Prodigal and 1 phage genome by MetaVir [23]. Each of these algorithms relies on information that is not available and calculations that are not possible with short genomes. For example, there are no conserved genes in phage genomes that can be used to build universal training sets [24], fewer genes means the statistics used to identify start codons are less accurate [25], and because many phage genes or the proteins they encode have no homolog in the databases, similarity searches are unreliable [26]. There are alternate gene calling approaches, such as using positional

nucleotide frequency [27], or the multivariate entropy of amino acid usage used by Glimmer [28], but these are designed for complete bacterial genomes and have not been optimized for use with phage genomes.

Here, we introduce a novel method for gene identification that is specifically designed for phage genomes. We make several presumptions based on studying hundreds of phage genomes. First, we noted that since phages have physical limits on their genome size they contain minimal non-coding DNA. Second, we showed that phage genes are usually on the same strand of the DNA, presumably because they are co-transcribed [15], [29]. Based on these observations, we designed a completely novel approach to phage gene identification, tiling opening reading frames within a genome to minimize non-coding DNA bases and switching between the forward and reverse strands. We treat a phage genome as a network of paths in which ORFs are more favorable, and overlaps and gaps are less favorable. We solved this weighted graph problem using the Bellman-Ford algorithm [30], [31], and by optimizing the parameters for phages genomes we are able to enhance phage gene prediction algorithms. In the absence of supporting data to confirm our new predictions, we turned to high-volume sequence similarity searches to explore the predicted proteins. Regions of the genome that encode proteins are more likely to be conserved at the amino acid level than regions that encode regulatory regions, replication regions, sites of integration and other, DNA-based, information components of the phage genome [32]. These searches showed that the predicted phage genes might encode novel proteins that have been missed by existing gene callers designed to annotate bacterial genomes.

2. Materials and Methods

2.1 The PHANOTATE algorithm

The first step PHANOTATE takes in identifying the genes in a phage genome is creating a weighted graph from the ORFs in that genome. By default, we allow for three start codons ($codons_{start}$ = {*ATG, GTG, TTG*}), and three stop codons ($codons_{stop}$ = {*TAA, TAG, TGA*}), and the default minimum length of an ORF is 90 nt. The directed weighted graph consists of nodes that represent start and stop codons, and edges that represent either an *ORF* if the edge connects a start codon to a subsequent stop codon in the same reading frame; a *gap* if the edge connects a stop codon to a subsequent start codon in any reading frame on the same strand, or if the edge connects a stop codon to a subsequent stop codon on the alternate strand; or an *overlap* if the edge connects a stop codon to a preceding start codon in any other reading frame on the same strand, or to a preceding stop codon on the alternate strand. Since phages rarely have >300 bp of untranslated DNA, and to reduce computational burden, we only connect ORFs within ±300 bp of each other. When there is a very large span without an ORF, we connect ORFs on each side of the region with a linear penalty.

For each edge, we calculate a weight depending on the feature type: ORF, overlap, or gap. To calculate the weight of an ORF (w_{orf}), we use an adjusted likelihood of not finding a stop codon in an ORF of this length. We count the fraction of each base in each ORF, and use that to determine the overall probability encountering a stop codon over the entire ORF:

$$P(stop) = P(TAA) + P(TAG) + P(TGA)$$

We then calculate *P(not)* to obtain the probability of NOT encountering a stop codon:

$$P(not) = 1 - P(stop)$$

Using P(*not*) alone to calculate the path through the genome is sufficient for genomes with an average GC content, however high GC content genomes have extremely long spurious open reading frames caused by their bias of generally having a G or C in the third codon position of their protein-encoding genes, which then forces a C or G in the first position in the opposite strand, limiting the options for including stop codons in the genome. To overcome this, we incorporated two GC frame plot scores into our final calculation. The initial GC frame plot score was inspired by Prodigal, but we have adapted that and we also include both minimum GC frame plot and maximum GC frame plot. We start by reading the three frames of the genome one base at a time, looking at the codon starting at that base, and calculating the %GC content over a 120 bp window for each of the three reading frames. Taking the set of ORFs that start with ATG, we iterate through the codons of those ORFs and determine which position (1st, 2nd or 3rd) has the maximum GC content, and maintain a running total for that position. Similarly, we calculate a GC frame plot minimum score by recording the minimum GC content.

This gives us a count for the GC frequency of the three positions, which we translate into scores by dividing each by the position with the highest count, bringing the preferred maximum GC position to 1, and the others to <1. This yields two sets of three-position scores that range between 0 and 1, with 1 being the maximal or minimal GC frame, and can be used to estimate which of the six frames is potentially *coding*, at any location. For instance, if the input genome had a bias where half of its max GC frame was in the third frame, and the other half split evenly between the first and second frame, once normalized, the GCFPmax scores would be [0.5, 0.5, 1]. The GC frame plot scores are used to exponentiate the *P(not)* score. For example, if a codon's GCFPmax score was 1, which would match the preferred frame, then *P(not)* is

unchanged. However, if a codon's GCFPmax score is less than 1, indicating that the current ORF is in a different frame to the preferred GC frame at that location in the genome, then that codon's *P(not)* value is reduced in the final calculation. Scores for ORFs are modified by a weighted ribosomal-binding site (RBS) score. Since little is currently known about the diversity of RBSs in phages, we employed a similar likelihood-based Shine-Dalgarno RBS system used previously [14]. In addition, we adjust the ORF score by the start codon used, based on the frequency of start codons in GenBank in 2,133 phage genomes, and then finally the weight is negated to denote these edges as favorable in the network:

$$w_{orf} = -\prod_{c=1}^{codons} (Pnot)^{GCFPmax_{maxGCframe(c)}GCFPmin_{minGCframe(c)}} * RBS * START$$

When continuing from a stop codon either in a gap or an overlap, the next ORF maybe on either strand of the DNA sequence. However, phage genes are usually on the same strand, and unlike bacterial genes, they rarely switch strands [18]. If a strand switch occurs, then a strand switch penalty is included in the weight of the gap or overlap, where *P*(switch) is equal to 0.05, otherwise no penalty is added: *P*(switch) = {0, 0.05}. This penalty value was calculated from our set of annotated genes derived from the 2133 phage genomes to occur at a rate of around 5% per protein-encoding gene. This is in contrast to bacterial genomes, whose protein coding genes switch strands at a rate of around 25%.

Gap weights w_{gap} need to be proportionally scaled to the ORF weights in order to balance the network, so we adapted the ORF scoring equation by not correcting by the GC frame plot, using a genome-wide average probability of not finding a stop codon $\overline{P}$(not), exponentiating by the length of the gap combined with the P(switch) value.

$$w_{gap} = \bar{\mathrm{P}}(not)^{len} + P(switch)$$

Overlap weights w_{overlap} also need to be proportionally scaled to the ORF weights, so they are calculated by averaging the two coding weights of the ORFs in the overlap, exponentiating by the length of the overlap, and adding a penalty if a strand switch occurs.

$$w_{overlap} = \left(\frac{P(not)_1 + P(not)_2}{2}\right)^{len} + P(switch)$$

In order to use these weights in the Bellman-Ford algorithm, they must be transformed into 'distances', so for each of the above weights we take the multiplicative inverse of the probabilities to create a weighted graph network. Our Python package *fastpath*, which is a custom implementation of the Bellman-Ford algorithm in ANSI C for computational speed, is used to find the shortest path through the network.

2.2 Comparison with other gene callers

We compared gene identification between PHANOTATE and the three most popular gene callers used to identify genes in phages: GeneMarkS, Glimmer, and Prodigal using a set of 2133 complete phage genomes, which were downloaded from the GenBank FTP server [33]. We did not include nine Mycoplasma and Spiroplasma phages, which use an alternative genetic code. We ran PHANOTATE and each of the three alternative gene callers with default (or 'phage' if available) parameters on each phage genome. In addition, the 'meta' option was used to allow Prodigal to run on genomes smaller than 20 kb. To compare the algorithms, we counted the number of ORFs predicted by each respective algorithm and compared those predictions to the corresponding genes in GenBank.

2.3 Statistical analyses

All analyses were performed in Python using the statsmodels and scipy modules [34], [35]. ANOVA, Tukey's honest significant difference test, Levene's test, Cohen's f2 test and t-tests were performed on ln(x + 1)-normalized length or count data.

2.4 Validation against the sequence read archive

In the absence of direct protein measurements, we used conserved similarity to test whether ORFs are likely to encode proteins. To create a positive control set, we combined the 223 385 ORFs that were predicted to encode proteins by one or more of Glimmer, GeneMarkS or Prodigal. To create a negative control set, we identified the 1,122,336 ORFs over 90nt that were not predicted to encode proteins by any software (Glimmer, GeneMarkS, Prodigal or PHANOTATE). Finally, we also identified the 15,105 ORFs that were unique to PHANOTATE (Figure 2). We previously developed PARTIE [1] to identify the random community genomes (metagenomes) in the NCBI Sequence Read Archive (SRA) [36]. We used LASTAL [37], [38] to compare six-frame translations of a 100,000 read sample of the sequence reads from these metagenomes in the SRA to the predicted protein sequences from the ORFs. Sequences with an expect value $<1 \times 10^{-10}$ were considered significant. The differences in means were compared using a one-way ANOVA followed by a post hoc Tukey's test to identify the variables driving any difference. Normality was tested using Levene's test [39]. Cohen's f2 test was used to determine effect size. These datasets are uneven and large and therefore direct comparisons may lead to small effects being found to be significant. To overcome this, we measure both Cohen's f^2 and d values to measure effect size [39], [40]. In addition, we subsample 1000 proteins with replacement at random from the entire pool of ORFs and use those in the ANOVA. We repeat this calculation 1000 times to determine whether the PHANOTATE

predictions are similar to either the set of positive predicted proteins or the negative control set of ORFs that were not predicted to encode proteins.

The Git repository contains a detailed description of the approach used to compare the SRA reads to the predicted ORFs, contains a link to the alignment data, and contains Jupyter notebooks with the statistical analysis reported below.

https://github.com/deprekate/PHANOTATE

3. Results

PHANOTATE is a novel gene caller designed explicitly to identify phage genes. It treats the genes in a genome like a network and finds the shortest path using the Bellman-Ford algorithm. To test the accuracy of PHANOTATE, we ran the code on 2,133 phage genomes that we downloaded from GenBank, calculated the number of genes predicted, and compared the results to the gene predictions by the three most popular methods for gene calling in phages: GeneMarkS, Glimmer, and Prodigal. In total, we identified 239,072 genes from 2,133 phage genomes (Table 1).

Table 1 Numbers and lengths of the genes predicted by the different gene callers

Gene Caller	Genes called	Mean length (nt)	Standard deviation of gene length (nt)
Phanotate	225,518	603	708
GeneMarkS	213,101	628	719
Glimmer	211,278	631	719
Prodigal	211,886	631	720

There was no statistically significant difference in the mean lengths of the genes predicted by Glimmer or Prodigal, while the mean lengths of the genes predicted by PHANOTATE and GeneMarkS were statistically significantly different to those called by the other algorithms [$F(3, 861\,779) = 440.45$, $P = 0.0$]. However, the effect size of the difference was very small ($d < 0.1$ in every pairwise comparison).

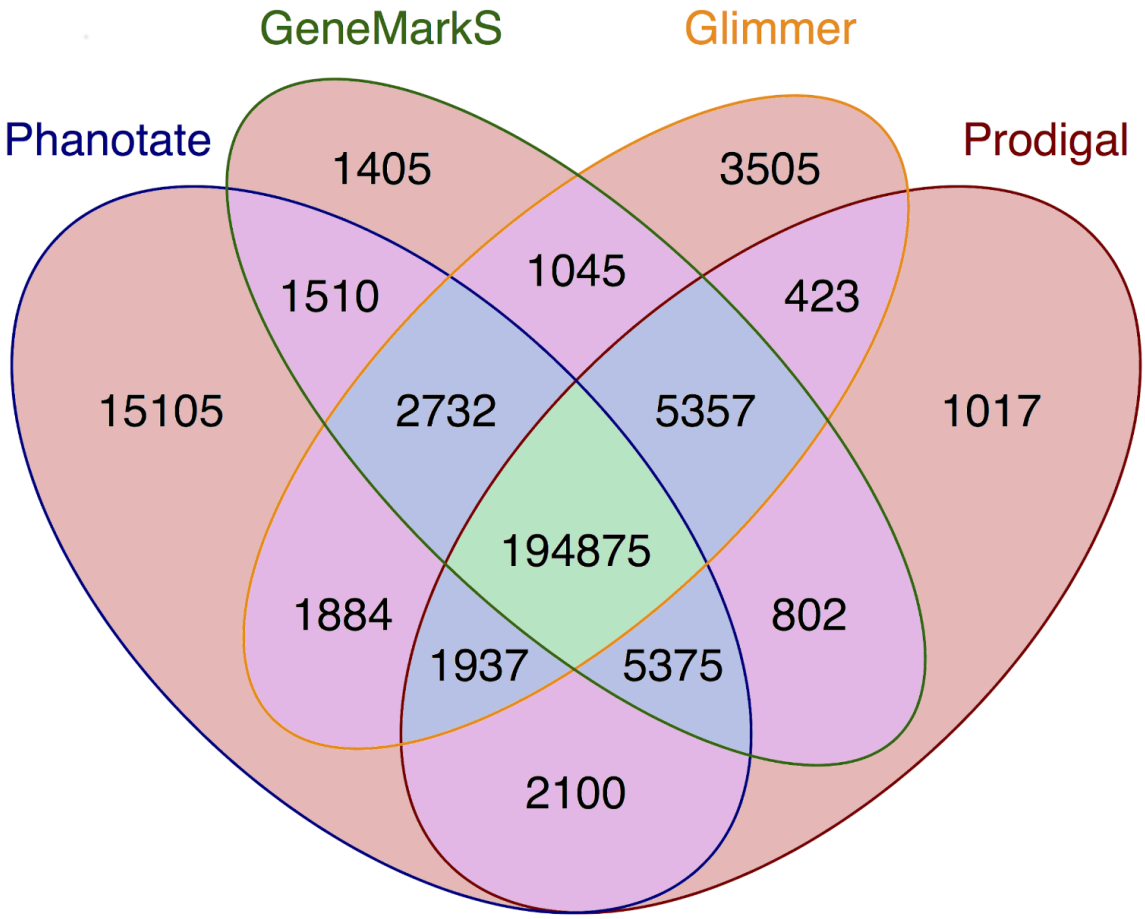


Figure 2 Number of genes predicted by each of four different gene prediction algorithms and the combinations thereof. Orange background: predicted by a single algorithm; green background: predicted by two algorithms; blue background: predicted by three algorithms

Each of the tools identified a set of predicted genes that were not identified by any of the other software (Figure 2). PHANOTATE version 1.0 predicted 15,105 genes (6% of the total number of

genes predicted by all software) that were not predicted by other gene prediction algorithms. An ANOVA comparison between the lengths of the genes identified by 1, 2, 3 or 4 gene callers identified significant variation [$F(1, 861\,781) = 21\,312.85$, $P = 0.0$], but the effect size was very small ($d = 0.02$). A *post hoc* Tukey's test showed that there was no difference between the lengths of genes identified by a single gene caller or two gene callers ($P > 0.05$), but that all other pairwise comparisons were different. When we consider just the unique genes that were identified by each algorithm the ANOVA comparison identified significant variation in the lengths of the genes [$F(3, 20\,856) = 56.6$, $P = 0$], but again the effect size was very small ($d = 0.01$). The *post hoc* Tukey's test showed that there were two groups that were significantly different between groups but not within groups ($P < 0.05$). Glimmer ($M = 217$ nt, SD = 174.35) and Prodigal ($M = 226$ nt, SD = 151.58) had indistinguishable mean lengths of unique genes, while the mean lengths of PHANOTATE ($M = 210$ nt, SD = 245.94) and GeneMarkS ($M = 183$ nt, SD = 109.06) were indistinguishable.

We cannot simply rely on the GenBank annotations to be correct. First, the proteins annotated in GenBank are typically predicted by the gene callers used in this comparison. Second, many of the current phage genome annotations in GenBank are filled with false positives. For example, in the Shiga toxin-converting phages (NC_004913 and NC_004914), every ORF longer than 160 bp has been annotated as a protein-coding gene. There are also abundant examples of false negatives, protein-coding genes present in the genome that were not identified by the annotation software used. The most obvious false negatives are genes shorter than 100 bp, since this is an often-used arbitrary minimum cutoff. Small genes that do not show strong coding signals, such as shared homology to known or hypothetical genes in the databases, or

shared codon usage, are often excluded by other gene annotators in an effort to minimize false positives.

The best experimental approach to determine whether these genes encode proteins would be to identify the proteins via proteomics. However, there are few published phage proteomics studies [41], [42], and in those studies, the raw proteomics data are not provided. Rather the authors only indicate which ORFs were matched, frequently using proprietary software and typically using gene calls made using the algorithms discussed here. This precludes our ability to use proteomics data to validate gene identification in phages.

In the absence of third-party validation datasets and experimental datasets, we turned to evolution to test whether the genes we predict in these phages may encode proteins. We hypothesized that protein-encoding genes are more likely to be evolutionarily conserved than ORFs that are not translated into proteins. Protein-encoding genes are constrained by the function of the protein. A variant of this approach has previously been used to identify genes in bacterial genomes [32]. When we compared the genes that PHANOTATE predicted to the proteins in the GenBank non-redundant (nr) protein database [33], there was significant similarity to 23% of the predicted proteins (expect value < 10^{-10}). This is similar to the 1–30% of phage proteins that typically have similarity to the GenBank nr database, and the remainder is often called the 'phage dark matter' [43]. The mean lengths of the predicted genes that did not match to GenBank (243nt) was significantly shorter than the mean length of those genes that matched GenBank (229nt) [$t(1000) = 3.02$, $P < 0.005$] but the effect size was small ($d = 0.19$). This may suggest that shorter proteins are under-represented in the database because of arbitrary lower limits on gene callers, shorter proteins have less statistical significance in

similarity searches, or PHANOTATE is identifying more, shorter, ORFs and incorrectly suggesting they are proteins. We, therefore, sought an additional assurance of the genes predicted by PHANOTATE.

For a more rigorous analysis of the ability of sequence similarity to discriminate between coding and non-coding genes, we turned to the largest repository of sequence data, the NCBI SRA [36]. Specifically, we extracted 94,652 random community metagenomes we previously identified [1]. We constructed two control datasets: a set of presumed positive predictions comprised of all ORFs predicted by Glimmer, GeneMarkS and/or Prodigal (but not those only predicted by PHANOTATE), and a set of known negative annotations of ORFs that are longer than 90 bp and not predicted to encode proteins by any of the software used here, including PHANOTATE. We mapped the reads from the SRA to the ORFs using the translated search algorithm LASTAL [37], [38]. When we compared the number of reads that mapped for all ORFs that had at least one read map, significantly more reads mapped to the ORFs predicted to be proteins (mean 1871.5 reads mapped; standard deviation 15933.2), than our negative control set (mean 136.0 reads mapped; standard deviation 1316.9) (Fig. 2) [$F(2, 149770) = 37900$, $P = 0.00$]. There was a large effect size for this comparison ($d = 0.9$), as can be seen in Figure 3. This analysis confirms that we are more likely to find reads mapping to ORFs if they encode proteins than if they do not encode proteins, and therefore we can use this approach to determine whether the ORFs predicted by PHANOTATE alone are likely to encode proteins.

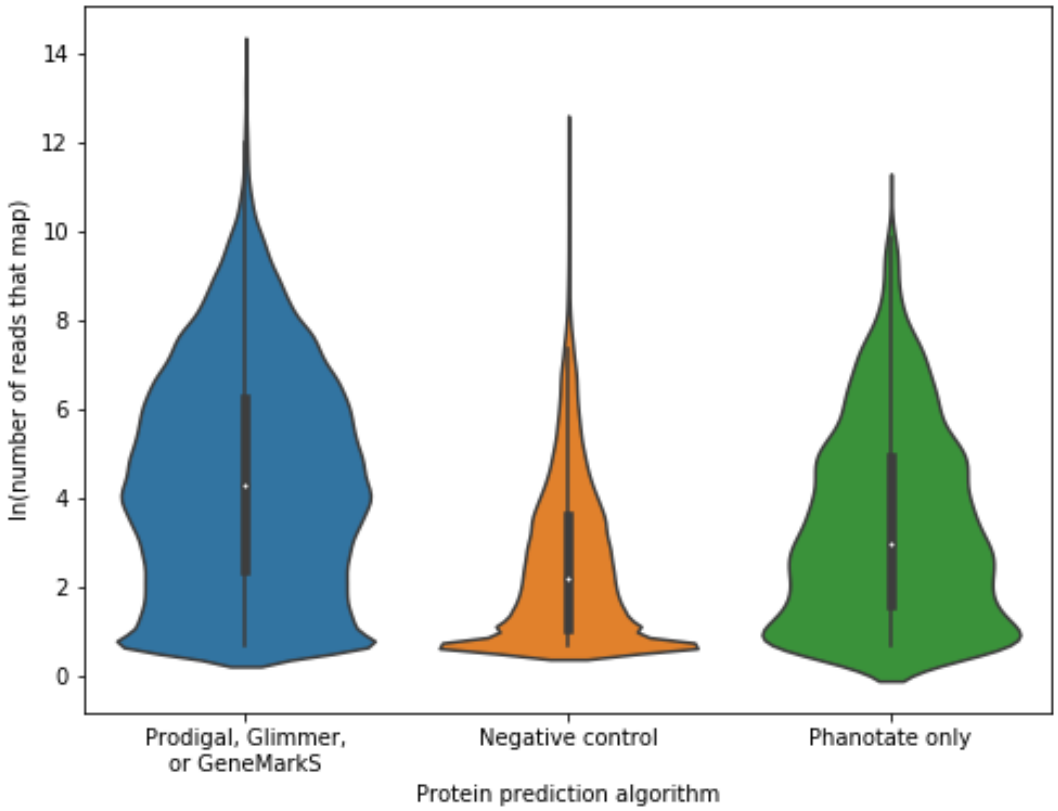


Figure 3 Violin plot of the ln(number or reads that map) to each of the ORFs predicted either by one (or more) of Prodigal, Glimmer or GeneMarkS; by no gene prediction algorithms (negative control); or by PHANOTATE alone

When we compare the ORFs that are only predicted by PHANOTATE and not predicted by the other ORF callers (~6% of all the ORFs identified) with the two control sets, 72% of the time the ORFs predicted by PHANOTATE had mean read abundance that was indistinguishable from the mean abundance of the true proteins, but 79% of the time the mean read abundance was similar to the ORFs that were not predicted to be proteins. Similarly, the medium effect size suggests that similarities to ORFs identified by PHANOTATE lie between those predicted by any gene caller (d = 0.42) and those not predicted by any caller (d = 0.47) as can be seen in Figure 3. The PHANOTATE predictions, therefore, lie between the 'true positives' from the other

software and the ‘true negatives’ of all other ORFs, suggesting, but not confirming that they may encode real proteins.

One of the unique features of PHANOTATE is that it is essentially reference free. Other programs, such as Prodigal, GeneMark and Glimmer, use hidden Markov models that require either a priori knowledge of the composition of protein-encoding genes or the identification of sufficient protein-encoding genes in the genome to build a training set. This is problematic when annotating phage genomes since most potential ORFs do not have homology to any known gene and the small phage genomes do not provide enough candidates to create a robust training set. In addition, many phage genes are horizontally transferred, and thus have different properties and signals from each other. Future versions of PHANOTATE may include the option to use these various gene properties, including hexamer frequency, codon bias and non-Shine-Dalgarno RBS detection, and also provide a mechanism to mask functional noncoding bases, such as those in RNAs, repeats, and attachments sites to further increase the accuracy of the gene calls.

CHAPTER 2

1. Introduction

The first genome ever sequenced was that of Bacteriophage MS2 [44]. Twenty years later the first bacterial genome, *Haemophilus influenza*, was sequenced, and with it the need to computationally predict where protein-coding genes occur in prokaryotic genomes [45]. This gave rise to the first of the gene annotations tools, GeneMark [7], Glimmer [8], and CRITICA [32], a decade later Prodigal [14], and most recently PHANOTATE [18] for viral genomes. One thing each of these tools share in common is the necessity for a training set of *good* genes: genes that are highly like to encode proteins, and that the software can use to learn the features that segregate *coding* open-reading frames (ORFs) from *noncoding*. GeneMark and GLIMMER, and to an extent Prodigal and PHANOTATE, all require precomputed gene models to find similar genes within the input genome, and the better these training models, the better the predictions that each tool makes. Both GeneMark and CRITICA rely on previously annotated genomes to predict genes in the input query genome. GeneMark selects one of its precomputed general heuristic models based on the amino-acid translation table and GC content of input genome. CRITICA uses shared homology between known genes and the input genome, as well as non-comparative information such as contextual hexanucleotide frequency. In contrast, GLIMMER, Prodigal, and PHANOTATE create gene models from only the input query genome, which removes the dependence on reference data. Each uses a different method to select ORFs for inclusion in a training-set, on which a gene model is built. GLIMMER builds its training-set from the longest (and thus most likely to be protein-encoding) ORFs, which are predicted by its LONGORFS

program. Prodigal uses the GC frame plot consensus of the ORFs, and performs the first of its two dynamic programming steps to build a training-set. PHANOTATE creates a training-set by taking all ORFs that begin with the most common start codon ATG.

An additional method, which has subsequently been added as an option to LONGORFS, is the Multivariate Entropy Distance (MED2) algorithm [46], which finds the Entropy Density Profiles (EDP) of ORFs and compares them to precomputed reference EDP profiles for *coding* and *noncoding* ORFs. The EDP of an ORF is a 20-dimensional vector $S=\{s_i\}$ of the entropy of the 20 amino-acid frequencies p_i and is defined by:

$$s_i = \frac{-1}{H} p_i log p_i \quad where \quad H = -\sum_{j=1}^{20} p_j log\, p_j \qquad \textbf{(equation 1)}$$

This approach relies on *coding* ORFs having a conserved amino-acid composition that is different from *noncoding* ORFs. This differential can be seen when comparing the observed amino-acid frequency of known phage protein coding genes to the expected frequencies (Figure 1A), which are based purely on the percent AT|GC content of the genome. Since *coding* ORFs have a bias towards certain amino-acids, and *noncoding* ORFs have frequencies approximately dependent on the nucleotide composition, each will cluster separately in 20-dimensional amino-acid space (Figure 4B).

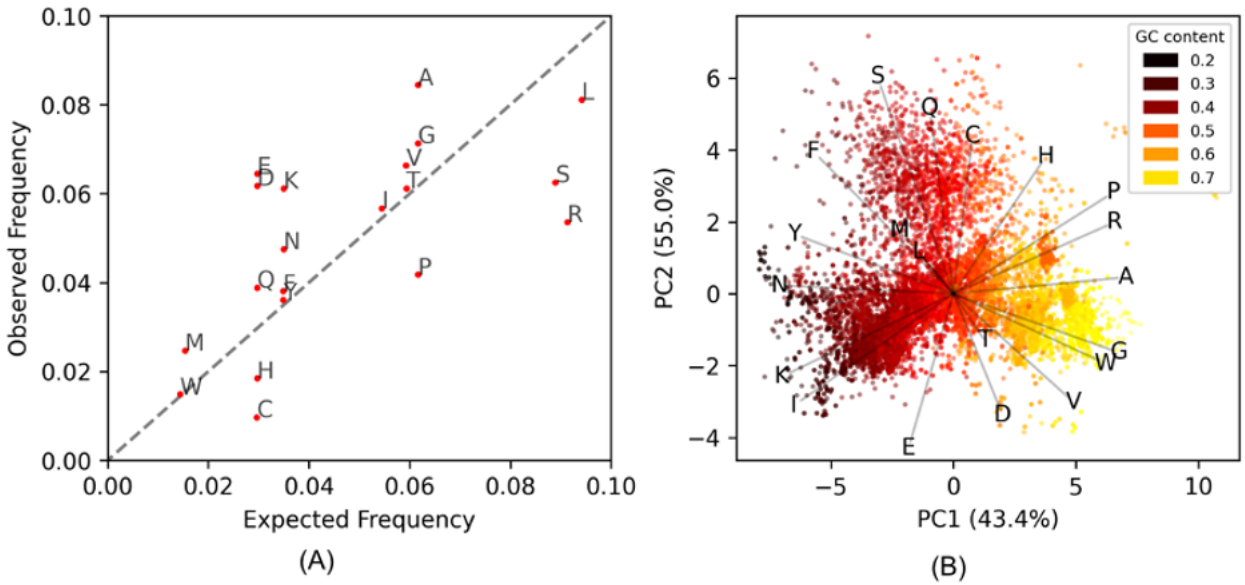


Figure 4 Visualizing the amino acid composition of open reading frames. **(A)** Comparison of average amino acid occurrence across 14,179 phage genomes. Points correspond to the 20 different amino acids and are labeled according to their IUPAC single letter abbreviations. Observed frequencies come from the annotated genome consensus gene calls, while the expected frequencies come from the overall codon probabilities calculated from the GC content. Amino acids above the diagonal identity line occur more frequently than expected in coding ORFs, and those below it occur less frequently than expected, which alludes to a coding bias signal. **(B)** The averaged amino acid frequencies of coding ORFs change based on the GC content. The previous consensus calls were averaged for each genome and then plotted using Principle Component Analysis (PCA), and colored based on the GC content of the genome. The (red) lower GC content genomes tend to favor the amino acids [FYNKI] with AT rich codons, while the (yellow) high GC content genomes tend to favor the amino acids [PRAGW] with GC rich codons.

MED2 uses these observed amino-acid frequency EDPs from known *coding*|*noncoding* genes, in the form of average "centers", to classify potential ORFs in other genomes. One problem that arises is that observed amino acid frequencies, and the EDPs derived from them, change based on the AT|GC content of a genome (Figure 4B). This is because the codon triplets for amino acids do not change even when the probability of those codons occurring changes due to AT|GC content. The way MED2 overcomes this is by using two different sets of *coding*|*noncoding* references, one for low GC content genomes and one for normal and high GC content genomes. Aside from the complication of forcing a discrete scale onto continuous data, there is also the issue that amino-acid composition can change independent of GC content. A good example of this is the *Propionibacterium* phages, which are the darker orange points at (4,1)

in Figure 4B that cluster with the yellow high GC content genomes despite having a normal GC content. Another problem with reference-based gene prediction, and the pitfall of all supervised learning, is that only genes similar to those already known are predicted. As such we sought to implement a reference free method for identifying protein-coding genes within a stretch of DNA, herein titled GOODORFS. Our approach takes the EDP metric, and expands on it by adding the three stop codons and the ORF length. However, rather than using reference EDP profiles, we use unsupervised learning (namely KMeans) to cluster the ORFs, and then denote the cluster with the lowest variation as the coding ORFs.

2. Materials and Methods

http://github.com/deprekate/goodorfs

The GOODORFS program, along with the data, can be obtained from the GitHub repository, and is available as Python3 code with the dependencies NUMPY [47] and scikit-learn [48]. PCA transformations were performed using the decomposition package from scikit-learn, and plots were created using Matplotlib [49]. Points were plotted in an interlaced pattern based on either stop codon location or genome ID, rather than by categorical legend order. Alpha transparency was adjusted to suit individual plots.

2.1. Data

A list of 14,254 phage genome ids was downloaded from the October 2019 snapshot of the Millard Lab bacteriophage database (millardlab.org, accessed on 19 October 2019) and subsequently the corresponding FASTA files were retrieved from GenBank. The genome sizes range from 1,417bp to 497,513bp, with a mean length of 58,497bp ± 53,767bp. From this dataset, we removed 55 genomes that came back empty, and 20 that belong to Mollicutes

phages (9 Spiroplasma, 8 Mycoplasma, 3 Acholeplasma) that use an alternative codon translation table, leaving us with a total of 14,179 phage genomes. Since genome annotations in GenBank are quite often incomplete or incorrect, we reannotated these genomes using the four available annotation programs (GeneMarkS, Glimmer3, Prodigal, and PHANOTATE), and to ensure reliable predictions, we only included predicted genes that were called by two or more programs. GeneMarkS (version 4.32) was run with default parameters and the *--phage* option. Glimmer was run using the supplied *g3-from-scratch* script, which creates a gene model from the input genome sequences using the LONGORFS program. For Glimmer (version 3.02), we changed the default minimum gene length from 90 bp to 87 bp, to match the other three programs gene length default settings. This is because Glimmer does not include the three nucleotides of the stop codon in the calculation of gene length, unlike the three other tools. Prodigal (version 2.6.3) was run with default parameters, with a change to the source code lowering the MIN_SINGLE_GENOME value from 20,000 to 1, to allow it to run on genomes smaller than 20k bases. PHANOTATE (version 1.5.0) was used with all default settings.

2.2. Algorithm

The GOODORFS algorithm is comprised of four steps: finding the ORFS, calculating the ORF EDPs, clustering, and choosing the cluster that contains the coding ORFs (Figure 5).

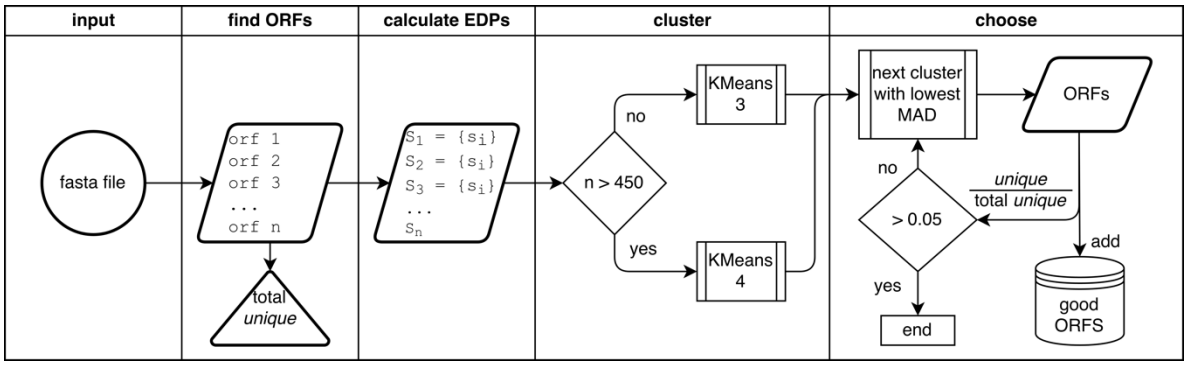


Figure 5 Flowchart of the GOODORFS workflow

2.2.1 Finding ORFS

For each genome, all potential ORFs were found in both frames by finding any start codon (ATG, GTG, TTG) and then following it to a stop codon (TGA, TAA, TAG), and only taking ORFs with lengths equal to or greater than 90 bp. We also include ORFs that run off either end without requiring either a start or stop codon, setting the start|stop position to the first|last available codon. This usage of multiple start codons per single stop codon leads to essentially having the "*same"* ORF, just alternate truncated versions, in our data. This was done because the correct start codon was not known, but the assumption was made that during clustering, the ORFs with correct start positions will cluster separately from those with incorrect start positions. We do however limit the number of these "redundant" ORFs in the data, by only including the first half (going from the outermost|furthest start codon in towards the stop codon) of the alternate start position ORFs. This was because some ORFs would have hundreds of alternate start positions: whether due to extremely long genes (as in the case of tape-measure proteins or RNA polymerase, which can be thousands of amino acids long), or shorter genes that have a repeated motif that includes start codons. Additionally, sequencing or assembly errors, where the same codon is repeated hundreds of times, can create an ORF that is composed entirely of start codons. Each of these alternate start positions all belong to the *same* stop codon, therefore, the term

"*unique"* ORF can be thought of as one specified by a single *unique* stop codon, and thus, all alternate start positions are essentially the *same* ORF.

2.2.2 Calculating the EDPs

Each ORF was translated using the Standard Code (GenBank's translation table 1) into the 20 amino acids (ARNDCEQGHILKMFPSTWYV) and the 3 stop codons: amber (+), ochre (#), umber (*). The 23 characters were counted and then divided by the total to get the frequencies, and the EDP was calculated according to Equation 1. To the EDP vector we also added a 24^{th} element, the length of the ORF. Before clustering we normalized the features of our data using the StandardScaler function from scikit-learn. Clustering was performed using the KMeans function (default parameters) from scikit-learn with three clusters for genomes with less than 450 *unique* ORFs, and four clusters for those with more. In order to control for the stochasticity of KMeans, which is not a deterministic algorithm, one thousand models were created and the model with the lowest inertia was kept.

2.2.3 Clustering and Prediction

To predict which cluster contained the coding ORFs, we calculated the Mean Absolute Deviation (MAD) of the points, excluding the ORF length values, summing the remaining 23 elements, and selected the one with the lowest value. The ORF length feature is excluded because even though it helps to resolve clusters, the coding ORFs tend to have a larger variance in their lengths, which biases the MAD sums. Even by limiting the number of alternate start sites per ORF included into the data, quite often clustering would assign all of these *same* ORFs to their own cluster. To overcome this, if a cluster was composed of less than 5% of the total *unique* ORFs in the dataset, that cluster and the next lowest MAD cluster were merged and selected as the

coding cluster. From the ORFs of the predicted coding cluster, we took the longest alternate truncation for each *unique* ORF and added it to the output set of *good* ORFs.

2.4. Performance Analysis

The MED2 program was run with all default values except with the change in the source code of minimum ORF length from 90 bp to 87 bp, because like Glimmer, it does not include the stop codon triplet into the ORF length calculation. We only ran the MED2 algorithm, and not the Translation Initiation Sites (TISModel) program, which uses Ribosomal binding site motifs to further refine the MED2 ORF predictions. The LONGORFS program was run during the Glimmer consensus annotation step, using the parameters above. Both Prodigal and PHANOTATE do not have functionality to log the set of training genes created, so a single line was added to the source code of each repo to print out the stop codon (which is a unique identifier) of each gene in the training set. The *diff* patch files to make this change are available in the GOODORFS repo, and can be applied via the command *git apply*.

3. Results and Discussion

We began our work by taking the previously published EDP metric but expanded it to also include the three stop codons, amber (+), ochre (#), and umber (*). We also appended the length of the ORF to the vector. The purpose of this is two-fold: First, the EDP metric loses informational content when converting from amino acid counts to frequencies. Certainly, a short ORF with a given frequency coding bias is less significant than a much larger ORF with the same frequency bias, since the latter maintained that bias over many more codons. Second, the ORF length bolsters the clustering step since coding ORFs are generally longer than noncoding (for our dataset, mean lengths were 595bp and 345bp respectively).

To demonstrate our approach, we took the representative genome, *Caulobacter* phage CcrBL9 [50]; chosen since it is the largest genome in our dataset that has a high GC content (> 60%), so it will have significantly more noncoding ORFs than coding, which allows for visualizing all of the categories in Figure 6. We then found all the potential ORFs in the genome, calculated their 24-dimensional EDPs, and used PCA analysis to plot them in two dimensions (Figure 6A). The *coding* ORFs (blue) and the *noncoding* ORFs (red) cluster separately, and this same pattern is observed across all other genomes (data not shown).

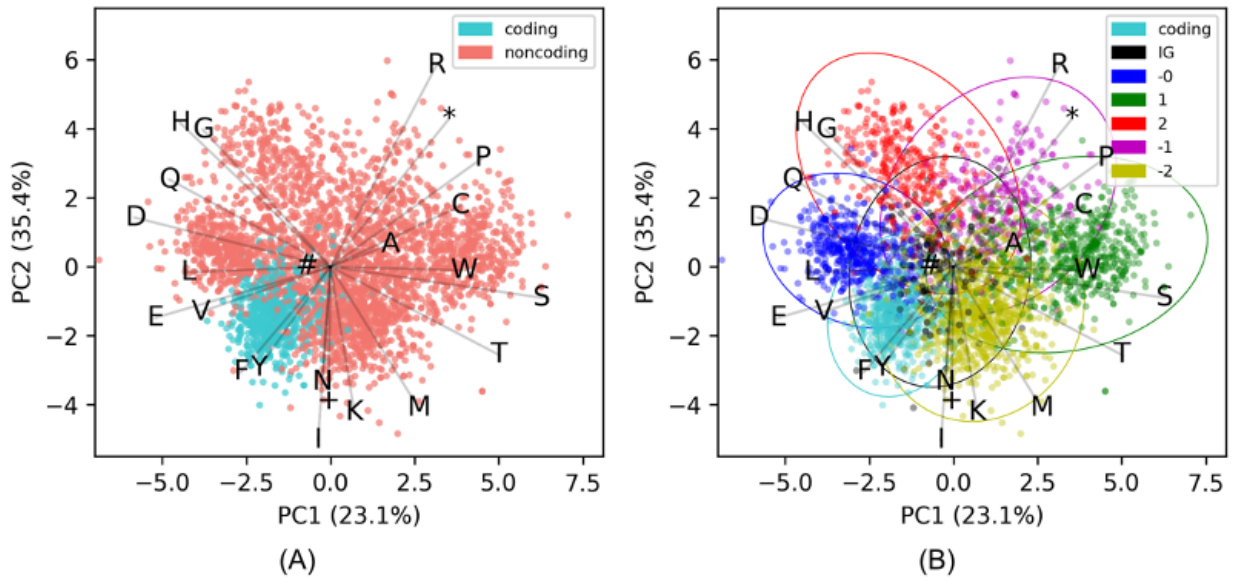


Figure 6 **(A)** The amino acid frequency EDPs of coding and noncoding ORFs for the representative genome *Caulobacter* phage cluster separately. All potential ORFs were found, taking only the longest (i.e. the first outermost available start codon) truncation, finding the amino acid frequencies, coloring according to whether they are in the consensus annotations (coding), and then plotted in a PCA. **(B)** The same potential ORFs from the previous figure, except with the noncoding ORFs colored according to their offset in relation to the coding frame (0-, 1, 2, 1-, 2-), or intergenic (IG) if they do not overlap with a coding ORF.

Initially we began using KMeans with two clusters, coding and noncoding. This worked well for genomes with average and low GC content, but did not work well on high GC content genomes (mean F1-scores were 0.72 and 0.49 respectively). This is because the probability of encountering a stop codon is lower in high GC content, and so they have about twice as many noncoding ORFs

as coding. This can be seen in Figure 6A where the red *noncoding* ORFs far outnumber the blue *coding* ORFs. Because the KMeans algorithm tends to distribute points into equally sized clusters, this would lead to far more *noncoding* ORFs to be incorrectly clustered with the *coding* ORFs. To overcome this, we initially changed to KMeans of three clusters, so that the *coding* ORFs would get their own cluster and the *noncoding* ORFs would be split between the two others clusters. Upon inspection of the data it is apparent that the *noncoding* ORFs fall into six different clusters, one for each reading frame *offset* (Figure 6B), and that not all have completely random non-conserved amino-acid frequencies. The first cluster of *noncoding* ORFs are those in intergenic (IG) regions, and for the most part, have amino-acid frequencies that are completely random and follow the expected frequency based on GC content alone. The other five clusters are those that overlap with a *coding* ORF, and are denoted by an *offset*, which can be thought of as the number of nucleotide-base shifts it takes to get to the *coding* frame. In contrast to the IG, the *offset* clusters overlap with a *coding* ORF, and have a slightly conserved amino-acid frequency. This is because even though they themselves do not encode proteins, they are not independent of the *coding* ORFs that they overlap with; they share the same nucleotides, just *offset* and in different frames. These non-intergenic *noncoding* ORFs fall into five categories (1+, 2+, 0-, 1-, 2-), and are always in relation to the *coding* frame, denoted as 0+ (which is synonymous with *coding*). The same-strand categories (1+ and 2+) correspond to when the *coding* ORF is in frame *n* and so the two *noncoding* ORFs are n+1 and n+2. Likewise, the three opposite-strand categories (0-, 1-, and 2-) correspond to a *coding* ORF at frame n and *noncoding* ORFs at n+0, n+1, and n+2, except in the reverse direction. Thus, we would expect there to be one conserved amino acid cluster for the *coding* frame, five spurious semi-conserved clusters, and one cluster for intergenic regions

(Figure 6B). Since we are working with phages, which can range down to only 18 *unique* ORFs for a genome, it is not possible to cluster ORFs into these seven clusters, so we settle on clustering into three for small genomes (less than 450 *unique* ORFs) and four clusters for larger genomes (over 450 *unique* ORFs). This cutoff was chosen because when using 4 clusters, genomes with less than 400 *unique* ORFs started to fail. Likewise, when using only 3 clusters, genomes with more than 500 *unique* ORFs began failing. We could have continued this pattern of setting multiple staggered cutoffs depending on genome size, until reaching seven clusters, but did not observe significant improvements when using more clusters, even for the largest genomes. We still need to pick which cluster contains the *coding* ORFs, since unsupervised clustering does not assign labels. Because *coding* ORFs have a conserved amino-acid frequency, they will have a higher "density" cluster when compared to *noncoding* ORFs - in 24-dimensional ordination the points will be highly clustered. This can be observed in Figure 6B, where the blue *coding* ORFs are a dense cluster, while the *noncoding* ORFs are a sparse cloud. We used the Mean-Absolute-Density (MAD) of each cluster to quantify the "density" in the 23-dimensional EDP space, since the MAD better accounts for outliers in the data. The cluster with the lowest sum of MADs (excluding the ORF length feature) was selected as the coding cluster.

To test the efficacy of our method, we took a set of 14,179 annotated phage genomes, and for each genome we identified all potential ORFs, calculated the amino-acid EDPs, clustered them, labeled the densest cluster as *coding*, and then calculated the F1-score to measure the performance. The average of the F1-scores was 0.85 ± 0.13, with only 32 genomes failing (an F1 score of less than 0.1). Unsurprising, all the failed genomes were very small in size, with ten or less annotated protein-coding genes (and 60 or less *unique* ORFs). Additionally, all the genomes

that failed belonged to the taxa *Microviridae*. Whether this is due to correlation (about 62% of the genomes with less than 60 *unique* ORFs were *Microviridae*) or causation remains undetermined. As shown in the plot of amino acid versus GC content (Figure 4B), there is a large cloud of points that do not follow the horizontal trend, clustering instead at the top of the figure separately along the vectors that represent the FSQC amino acids, and most of these genomes belong to *Microviridae*. This suggests that the *Microviridae* do not use the Standard Code, but rather one that potentially substitutes one or more of the over-observed amino acid codons [FSQC] for the codons that are under observed [VDE]. This hypothesis is supported by the dozen or so phages from other taxa that group with the *Microviridae* away from their expected locations in Figure 4B. Examples of two non Microviridae phages with unusual amino acid composition are shown in Figure 7, where no discernable separation of *coding* from *noncoding* is observed. Both of these genomes cluster with the Microviridae at the top of Figure 4B away from the expected frequency, and their larger genome sizes reinforces the possibility that they are using a different codon translation table.

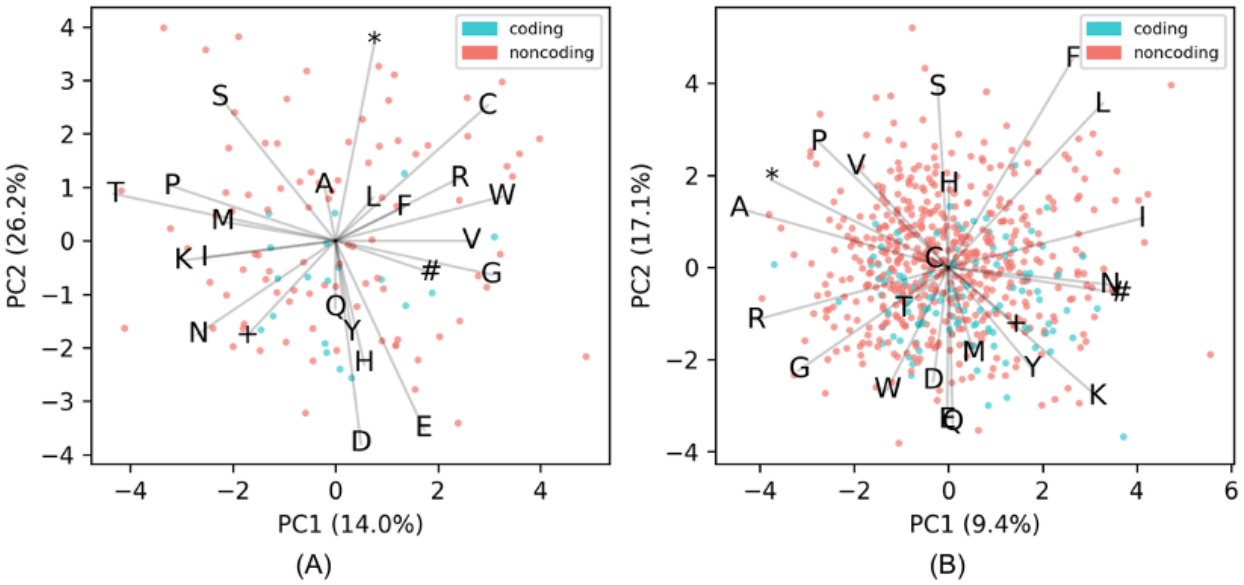


Figure 7 Two examples of phage genomes where the amino acid frequencies of coding and noncoding ORFs do not follow the general trend of clustering separately. Shown here are unique ORFs for **(A)** the filamentous phage Ralstonia RSM1 and **(B)** Escherichia phage fp01. Other filamentous phages show the same lack of observable coding bias, which could be due to the small genome size, however it is clear that the Escherichia phage does not use the Standard genetic code.

We compared the F1 scores of our analysis to that from four other similar programs that create a training set of genes from an input genome: LONGORFS (Glimmer), MED2, PHANOTATE, and Prodigal (Figure 8). For LONGORFS, 49 genomes had an F1-score less than 0.1, and the overall mean F1-score was 0.53. For MED2, there were six genomes that caused the program to crash, of the remainder there were 1,143 genomes with an F1-score less than 0.1, and the mean F1-score was 0.55. No genomes had an F1-score less than 0.1 with PHANOTATE, since its method sacrifices precision for recall, and subsequently PHANOTATE had the lowest mean F1-score of 0.43. Prodigal performs well, with an F1-score of 0.75 and no genomes failing (an F1-score less than 0.1).

Table 2 Comparison of various performance metrics for GOODORFS and four other similar methods. The listed values are means across all 14,179 genomes; except for the number of genomes failed, which is the number of genomes that had F1-scores lower than 0.1. All values are rounded up to the nearest significant digit.

	GOODORFS	**LONGORFS**	**MED2**	***Phanotate**	***Prodigal**
precision	0.79	0.92	0.60	0.28	0.65
recall	0.90	0.39	0.56	0.96	0.92
accuracy	0.94	0.89	0.87	0.58	0.65
F1-score	0.83	0.53	0.55	0.43	0.75
genomes failed	32	49	1139	0	0
runtime (sec)	488	1	1	11	1

* For PHANOTATE and Prodigal only the initial training set creation step was run, and not the entire gene finding algorithm.

Comparing the mean F1-scores of the four alternative methods to the 0.83 obtained from GOODORFS shows just how much better our method is at recovering *coding* ORFs compared to other methods (Table 2). Of the four alternative methods, Prodigal performs the best, but at this point in the complex algorithm, Prodigal has already performed one of its two rounds of dynamic programming to identify *coding* ORFs. Plotting the individual genome F1-scores from GOODORFS versus the four other methods shows the distribution, where points above the diagonal identity line, of which there are many, signify that GOODORFS is outperforming the competing method. Many of the points that have low F1-scores (< 0.5) with GOODORFs, also happen to belong to the class of genomes that have unusual amino-acid frequencies. The two representative genomes discussed above, *Ralstonia* phage RSM1 and *Escherichia* phage fp01, had GOODORFS F1-scores of 0.38 and 0.23 respectively. These two genomes did not fare much better with the other programs with the highest F1-scores of 0.56 and 0.41 coming from Prodigal, which is not beholden to codon translation tables, except in the form of start and stop codons.

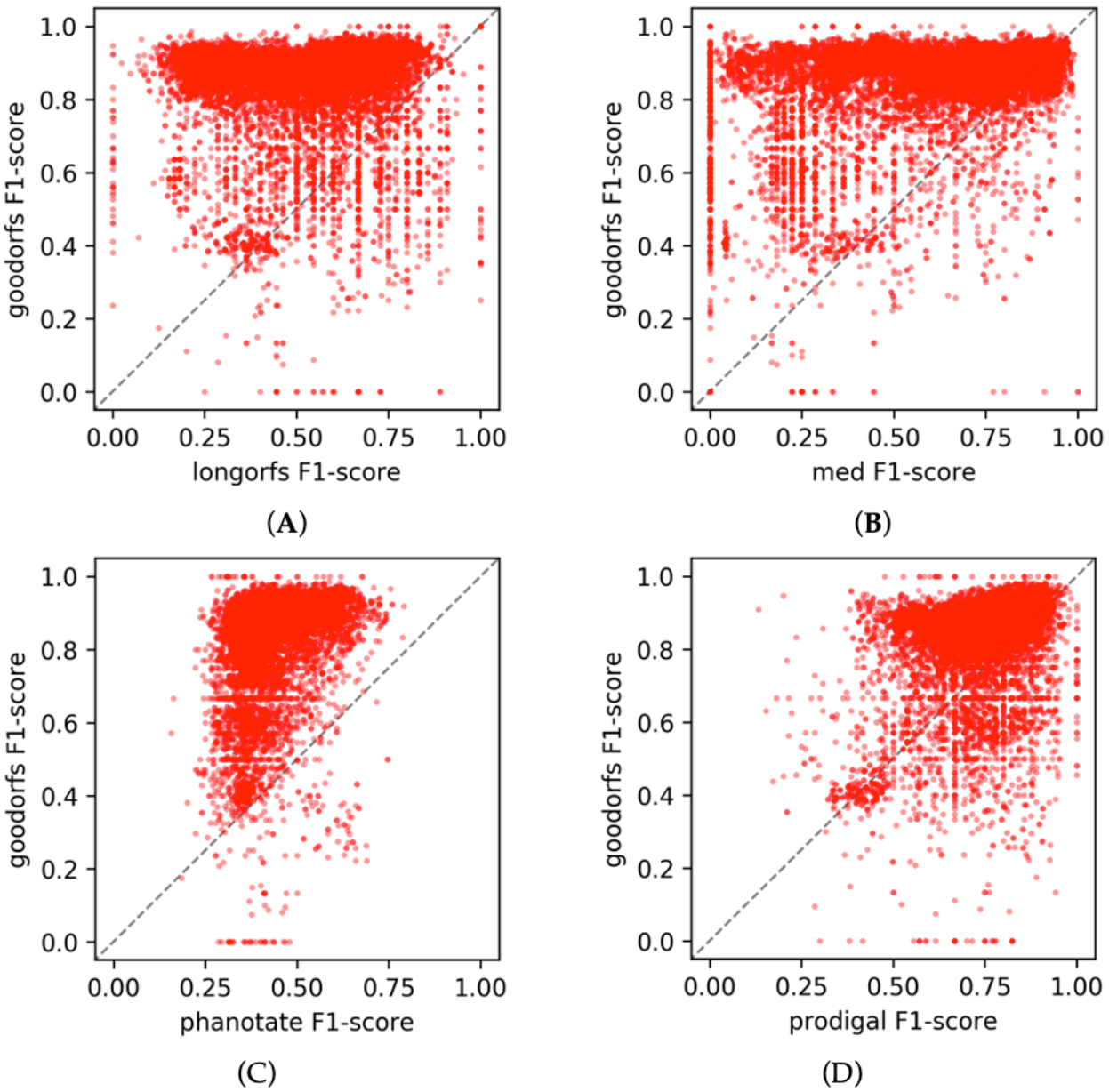


Figure 8 Comparison of the F1 scores for all 14,179 genomes between GOODORFS and **(A)** LONGORFS **(B)** MED2 **(C)** PHANOTATE's training set creation step **(D)** Prodigal's training set creation step. In each panel the dotted line represents x=y, and so points above and to the left of the line represent more accurate protein encoding gene identification by GOODORFs while points to the lower/right of the line indicate less accurate protein encoding gene identification. Points on the line indicate agreement between the algorithms.

All the previous performance results are based on simple ORF counts. If we were to instead normalize by ORF length, the F1-scores for methods that favor recall over precision (GOODORFS, Phanotate, Prodigal) would all increase because, as we have shown, the true-positives (*coding* ORFs) are longer than the false-positives (*noncoding* ORFs). This is relevant because training on a

gene set usually involves iterating over the length of the genes, and so longer ORFs have more emphasis than shorter ORFs. An example of this would be in calculating synonymous codon usage bias in a set of genes, where a long ORF with many codons would contribute more than a short with few codons. Another aspect of training on a gene model, is that a high F1-scoring method might not be the best choice for a given gene prediction tool. Sometimes it is better to have a very precise training set with very few false-positives. For instance, this might be the case with Glimmer, which uses the very low F1-scoring method LONGORFS to create a training set. Perhaps having only a few true-positive genes is worth not having any false-positives contaminating the training gene model. Whether to adopt our GOODORFs method into phage gene finding tools or pipelines is left up those researchers to determine on a case-by-case basis.

An area where our GOODORFS method might excel is gene prediction in metagenomes. All of the most readily used methods for gene prediction in metagenomic reads (MetaGene [51], Orphelia [52], MetaGeneMark [53], metaProdigal [54], FragGeneScan [13]) rely on reference databases of known genes, which as previously discussed, has its disadvantages. In contrast, GOODORFS is reference free; the only prior information needed is the amino-acid translation table. Depending on the sequencing technology used, generally a read only contains a fragment of a protein-coding gene, with the beginning or the end (or sometimes both) of the gene extending beyond the length of the read. Because GOODORFS allows for ORF fragments (that lack a start or stop codon) at the edges of the input sequence, it has the ability to work on metagenomic reads. All that is needed is to bin reads according to their GC content and then run the bins through GOODROFS in batches in order to predict gene fragments within the reads.

4. Conclusions

We have demonstrated the benefits of our GOODORFS method over similar alternative methods. The more accurate the set of genes used for training a gene model, the more accurate the final gene predictions will be, and will lead to significant improvements in current gene finding programs. We plan on incorporating GOODORFS into the PHANOTATE code to replace its current naïve method for creating a training set of all ORFs that start with the codon ATG. Glimmer, which uses LONGORFS, which had the second lowest F1-score (after PHANOTATE), could potentially benefit from switching to using GOODORFS. Since Glimmer is quite modular in its workflow, it is quite easy to change it to use GOODORFS. To facilitate this change, we even tailored our code to mirror the command line arguments and output format of LONGORFS, and have included the code to implement this patch in the GOODORFS github repo. The one lingering shortcoming of GOODORFS is that the current version is coded in Python, which is a scripting language and is significantly slower than compiled code, as evidenced in our runtime calculations. In order to make GOODORFS more competitive over competing methods, we have begun work on a faster compiled C version.

CHAPTER 3

3.1 Introduction

For the last half-century, the *Central Dogma* has shaped our understanding of molecular biology. In the idealized version of genetic information flow: DNA is transcribed into RNA, which is then translated into protein. During transcription, a polymerase reads along a strand of the DNA, one base at a time, catalyzing the production of a complementary strand of RNA. During translation, a ribosome reads along the RNA, one codon at a time, forming a polypeptide chain of amino acids until it reaches a *stop* codon. Since a codon is composed of a nucleotide triplet, the ribosome shifts three nucleotides at a time in order to maintain the fidelity of the amino-acid encodings. The first exception to this +3 indexing was hypothesized in 1975 to explain the occurrence of peptides slightly larger than expected during *in vitro* translation of the four known genes in the phage *MS2* genome. A small fraction (5%) of the synthetase gene, whose product is 62kDa, migrated during electrophoresis as if it were 66 kilodaltons, approximately 40 amino acids longer than expected [55]. Stop codon readthrough was ruled out, and several possibilities were hypothesized: protein splicing, base insertion or deletions during transcription, post-translational modifications, and even that the ribosome might "retranslate" portions of the gene. However, it was the very last possibility surmised *"that 7 % of the ribosomes translating the synthetase shift out of phase and bypass the normal termination signals"* proved to be correct. In subsequent work, it was shown that the rate at which the larger protein was translated could be affected by adjusting the concentrations of the different

tRNAs, and also revealed that one of the other four genes, a coat protein, also had variable protein sizes [56]. Although the significance of the frameshifted synthetase product for phage replication has yet to be determined, the function of the frameshift in the coat gene was to terminate coat protein synthesis early and present the frameshifted ribosome at the start of the overlapping lysis gene [44]. Presumably, once enough copies of the coat protein were translated, sufficient complete virions have been assembled, and sufficient levels of lysis protein have increased, it is then time to rupture and escape from the doomed host cell.

It was not until genetic sequencing became more widely available and the region around the known frameshift in the *gag*/*pol* gene of the mouse mammary tumor retrovirus *Rous sarcoma* was sequenced that a more detailed model of the necessary "signals" involved in backward ribosomal frameshifting was proposed. In this model, there is a slippery sequence upon which the ribosome shifts several bases, paired with downstream stem-loop or pseudoknot secondary structures, "*that may act by stalling translating ribosomes, thereby promoting the tRNA slippage*" of the bound codon:anticodon pairs in the E and P sites of the ribosome [57].

Over the years, other signals contributing to the backward model were identified (Figure 9), most notably ribosomal binding site (RBS) interference. Although first shown to play a role in forward frameshifts [58], it was later found that RBS interference also plays a role in backward frameshifting [59]. Typically, bacterial translation initiation requires a Shine-Dalgarno (SD)-like sequence a few bases upstream of the start codon. This sequence promotes recruitment of the ribosome onto the mRNA because the small subunit of the 16S rRNA molecule has a complementary *anti* Shine-Dalgarno sequence found near the entrance channel of the ribosome that recognizes and binds to the RBS. Therefore, if during translation the ribosome

encounters a slippery sequence that has an RBS-like motif properly positioned on the mRNA, it interacts with the *anti* Shine-Dalgarno sequence on the ribosome and facilitates frameshifting.

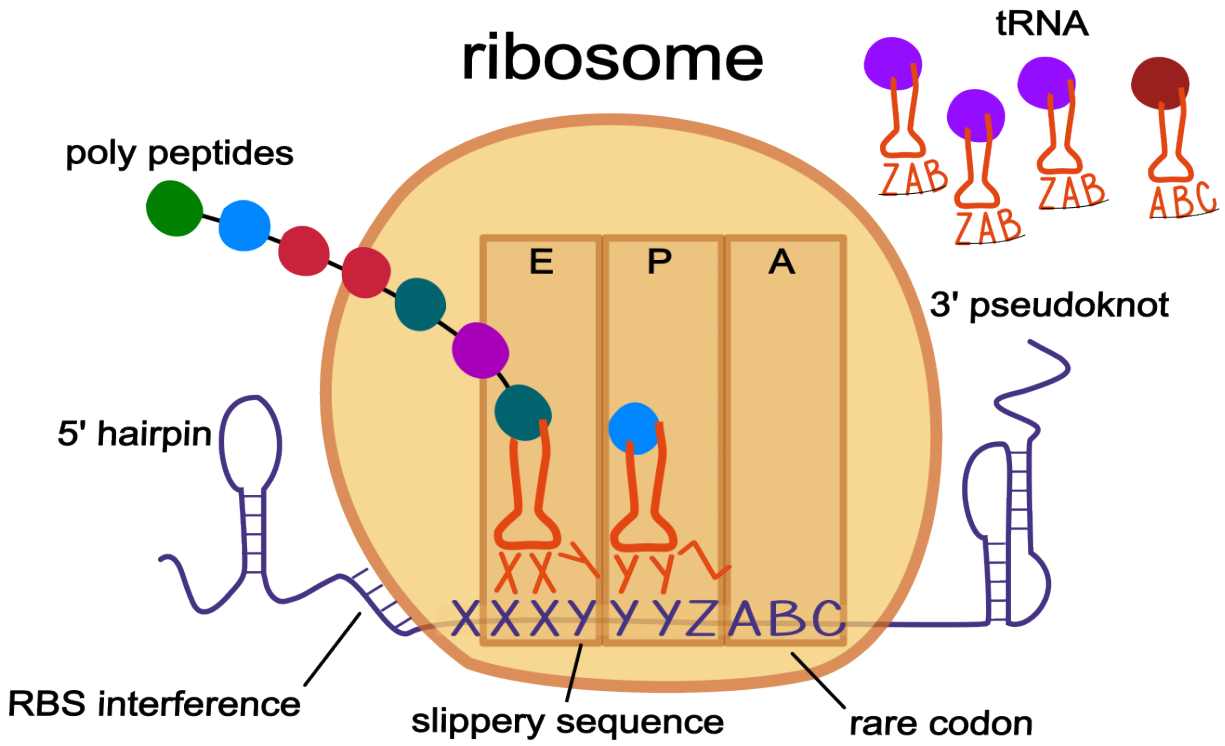


Figure 9) Some known cellular properties that are thought to contribute to programmed ribosomal frameshifts. The bi-directional model is shown with backwards -1 frameshifts, although the properties may also be associated with forward +1 frameshifts. The general model is based around a slippery sequence, in this case the canonical XXXYYYZ motif that was first identified in the gag/pol gene of a murine retrovirus. During translation as the ribosome is reading along in the 5' to 3' direction, it encounters the slippery site in its E and P site, which are both filled with their matching cognate tRNA. The presence of 3' secondary structure along with a 5' ribosomal binding like sequence work to pause the ribosome. The presence of a rare codon (ABC) is generally thought to induce forward frameshifts, however the ratio of the waiting A-site codon (A0) versus the ratio of the codon shifted into (A1) may also be involved in backwards -1 frameshifts since moving the A-site into frame with a rare codon would be unfavorable. While paused at the slippery sequence, the absence of 5' secondary structure allows the ribosome to slip backwards 1nt putting it into the -1 frame. The motif of the slippery site allows the anti-codons of the tRNAs (XXY and YYZ) to satisfactorily re-bind with the -1 codons (XXX and YYY) due to the permissible nature of the third—wobble—position. The new A-site codon (ZAB) is then filled with its cognate tRNA, the bases pairing of the 3' secondary structure momentarily disassociates, allowing the ribosome to proceed with translation in the -1 frame.

Surprisingly, the only slippery sequence motif identified to date is the original heptamer motif XXXYYYZ, first posited for the *gag*/*pol* gene backward frameshift [60]. This mechanism involves

the simultaneous (backward) slippage of two tRNAs along the mRNA within the decoding center of the ribosome (Figure 9). During translation, when the ribosome reaches the slippery site and the two tRNAs XXY and YYZ are bound to their respective codons on the mRNA, cellular signals combine to shift the two tRNAs backwards one base, so they are paired with XXX and YYY codons (Figure 9). Even though the new pairing is not exact, since each codon/anti-codon pairing has one mismatched nucleotide, the wobble nature of the third position allows for transitory stability until the ribosome recruits a tRNA matching the new -1 A-site codon, and translation continues in the -1 frame. For forward frameshifts, no such general motif has been identified yet. Instead, a ribosome encountering an unfulfilled rare codon during a gene translation can pause long enough to cause a forward shift into the +1 frame. It is generally thought that 3' secondary structures do not play a role in +1 forward frameshifts; however, a 3' pseudoknot plays a role in the +1 programmed ribosomal frameshift of mammalian ornithine decarboxylase antizyme [61]. The presence of such knots downstream of +1 forward frameshift sites might instead be the result of bidirectional ribosomal frameshifting, which has been shown to occur in both the human *prfB* gene [62] and the ORF1a polyprotein of the SARS-Cov-2 coronavirus [63]. In the bidirectional model, secondary structures on both sides of a slippery site help to regulate ribosomal frameshifting; secondary structure on one side acting to nudge the ribosome backwards or forwards, while an attenuator structure on the other side works to block the shift. The structure downstream of the slippery sequence can be either simple hairpins or more complex pseudoknots, while upstream structures are limited to hairpins since they are quicker to form once the ribosome has passed. The full extent of bidirectional control at slippery sites has yet to be determined; perhaps all frameshifts have some limited

bidirectional control, and we have only identified the more obvious antipodal pseudoknots, while ignoring the lesser hairpins on the attenuation (upstream) side.

The lack of experimental evidence for slippery site motifs other than the original XXXYYY sequence as well as limited knowledge of the full extent of cellular properties that contribute to inducing ribosomal frameshifts, makes creating software to predict their occurrence in genomic data rather difficult. This might account for why there is not a single universal tool available for predicting programmed ribosomal frameshifts.

The few available tools that investigate programmed ribosomal frameshifts do not leverage machine learning methods since they only look for a single specific slippery sequence motif and show all possible locations of that motif [64], [65], or were designed to only predict frameshifts for a specific gene of a specific taxon [66], [67]. Here we present PRFect, a new computational tool that is intended to predict ribosomal frameshifting of all types of coding genes in complete genomes from all domains of life, that is both accurate and also very easy to use.

3.2 Methods

github.com/deprekate/prfect

All the code and data presented here are available in the GitHub repository. All of the code exclusive to the PRFect package was written in Python3 in order to be user-friendly and easily updateable for future improvements. PRFect is also available on the Python Package Index PyPI (pypi.org) as an easily installable command-line program that downloads and installs with a single command: *pip install prfect*. The PRFect package does require the third-party dependencies: *scikit-learn* [48]and *numpy* [47], as well as the additional packages, *genbank*,

score_rbs, *linearfold* [68], and *hotknots* [69]. The last two were adapted from their original C code libraries into Python packages that auto-install along with the other packages when the previous single previous command is used to install PRFect.

Obtaining Data

To obtain ribosomal frameshift data, we downloaded 3,679 phage genomes in GenBank format from the Actinobacteriophage Database phagesdb.org [70]. Genes exist as *CDS* features within the GenBank format [71]. No explicit designation indicates if or where ribosomal frameshifting occurs within a gene. However, when a *CDS* feature has discontinuous locations in the GenBank file, they are denoted by using the *join* keyword in the coordinates. Figure 10 shows a small example GenBank file with two genes. The first example gene occurs from nucleotide 1 to nucleotide 100, while the second example gene exists in two locations denoted through by the *join* keyword.

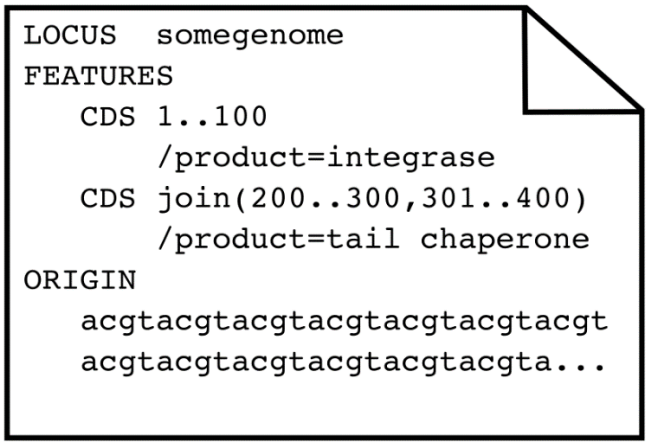


Figure 10) A graphical representation of an exemplar GenBank file. A GenBank file has various keywords in a specific order. Only the three most pertinent sections are shown: LOCUS, FEATURES, and ORIGIN. The first line of the file always contains the keyword LOCUS followed by the name of the genome. Then comes the keyword FEATURES followed by lines that contain the type of feature, in this case, coding

sequences (CDS) and the location within the genome of that feature. The features can have descriptor tags that describe the various properties of that feature. Each feature tag begins with a forward slash, and in this case the two features are tagged with functional annotations through the use of the /product keyword. The last important section is the ORIGIN, which contains the DNA or RNA backbone of the particular viral genome. More details about the GenBank flat file format can are on the NCBI website (https://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html).

Since frameshifted genes have multiple locations, they can be found by looking for CDS *Features* that use the *join* keyword. However, in addition to ribosomal frameshifting, other reasons can cause a gene to exist in multiple locations, such as splicing, mobile elements that insert into genes, genes spanning break points in circular genomes, and even sequencing errors. To distinguish ribosomal frameshifted genes from other causes, we required only two sets of coordinates within 10bp of each other, which was chosen through manual inspection of the genomes. Ideally, the two pairs of ribosomal frameshifted genes would be separated by either 0 (backward) or 1 (forward) nucleotides. However, the *joined* ribosomally frameshifted genes of the SEA-PHAGES data varied from 0 to 7 nucleotides apart, while the genes that were *joined* due to other causes (discussed below) varied from 50 to 818 nucleotides apart. Of the 3,679 genomes, 2,489 phages had one or more *CDS* features with the *join* keyword, giving a total of 2,557 *joined* CDS *Features*. Of these, 61 were at opposite ends of the genome, indicating genes split due to the circularization of the genome. There were 20 *joined* features with coordinates separated by more than 10 bp that were excluded from the training data: six in genes coding for major capsids, lysins, and DNA methylases (caused by inteins from homing endonucleases); twelve in genes coding for minor tail proteins (due to group 1 introns); one encoding a "structural" protein; and one encoding a tail assembly chaperone. The two pieces of the tail assembly chaperone were separated by 72bp, which we predict was an annotation error, along with all the other joined genes with coordinates further than 10bp apart. There were 2,476

frameshifted genes (one per genome) and 360,977 genes without frameshifts. The frameshifted genes were split into their two constituent fragments, giving sets of two consecutive "pseudo" gene pairs for positive cases. For example, the frameshifted chaperone gene in Figure 10 would be split into two CDS *features* with locations (200..300) and (301..400), and the *join* keyword removed. All of the genes were evaluated pairwise to determine if an overlap could occur between consecutive genes, allowing for the possibility that the two genes are a single frameshifted gene. Figure 11 shows an example of such an overlap in a pair of consecutive genes (denoted in green).

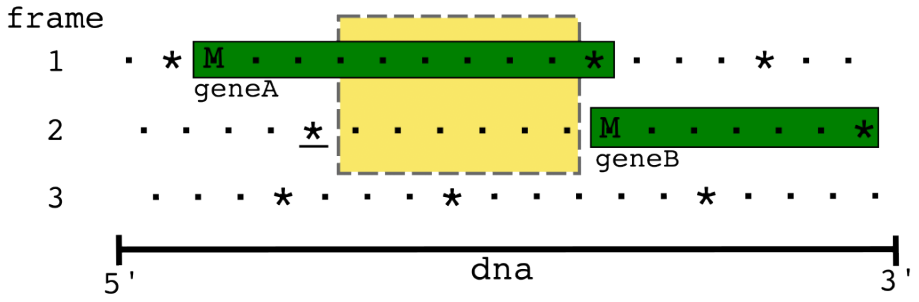


Figure 11) An example genome with two overlapping genes. Dots represent codons, asterisks represent stop codons, and M represents start codons. Only the forward three frames are shown. There is a gene in the first frame whose stop codon overlaps with the start codon of the gene in the second frame, however since the closest (underlined) stop codon in the 5' left direction from the start codon of *geneB* is six codons away, there is the possibility of a frameshift occurring in this overlap (yellow dashed box) region.

In the example, *geneA* (frame 1) is 10 codons long and overlaps with 1 codon of *geneB*; since the first stop codon upstream (in the 5' direction) from *geneB* (frame 2) is six codons to the left, there is the possibility that a frameshift could occur within this (yellow) overlap region of Figure 11. If there were a frameshift, *geneB* would not be a complete gene, but rather part of an alternate frameshifted translation of the fusion *geneAB.* Most consecutive genes were either on

opposite strands (thus, ribosomal frameshifting is untenable) or did not have the possibility of overlap, leaving only 67,664 gene pairs to search for negative case slippery sites.

Slippery Site Motifs

The only slippery sequence to appear in the literature is the previously mentioned XXXYYY motif (Jack 1988). However, this "*threethree* motif" (the number denotes the same nucleotide repeated three times and then another nucleotide repeated three times) is not present in most frameshift overlaps, so we explored recurring patterns that could similarly serve as motifs. Originally, we tried using various automated tools to detect motifs, such as the MEME Suite [72], but this proved troublesome due to the highly repetitive nature of our training data, so we were forced to manually inspect the overlap regions of the annotated frameshifts. Focusing only on those annotated frameshifts that lacked the *threethree* motif, we looked for novel nucleotide patterns that could function similarly to the *threethree* wobble base pairing dynamic. For backwards frameshifts, we found the eight different slippery sequence motifs: *six*, *threethree*, *fivetwo*, *twofive*, *twofour*, *threetwotwo*, *five*, and *twoonefour* (Figure 12). For forward frameshifts we only looked for the two motifs *four* and *three;* however, we also required that the codon of the +1 frame A-site relative abundance (A1) is greater than the codon of the +0 frame (A0) A-site. Requiring the A1 codon relative abundance to be more favorable limits the number of candidate forward slippery sites found and speeds up runtimes because three (and four) bases in a row occur very often in a genome. Since some motifs are subsets of other motifs (i.e. *twofive* is also *twofour*, *six* is also *threethree,* and *four* is also *three*), the motif with a lower probability of occurring randomly takes precedence.

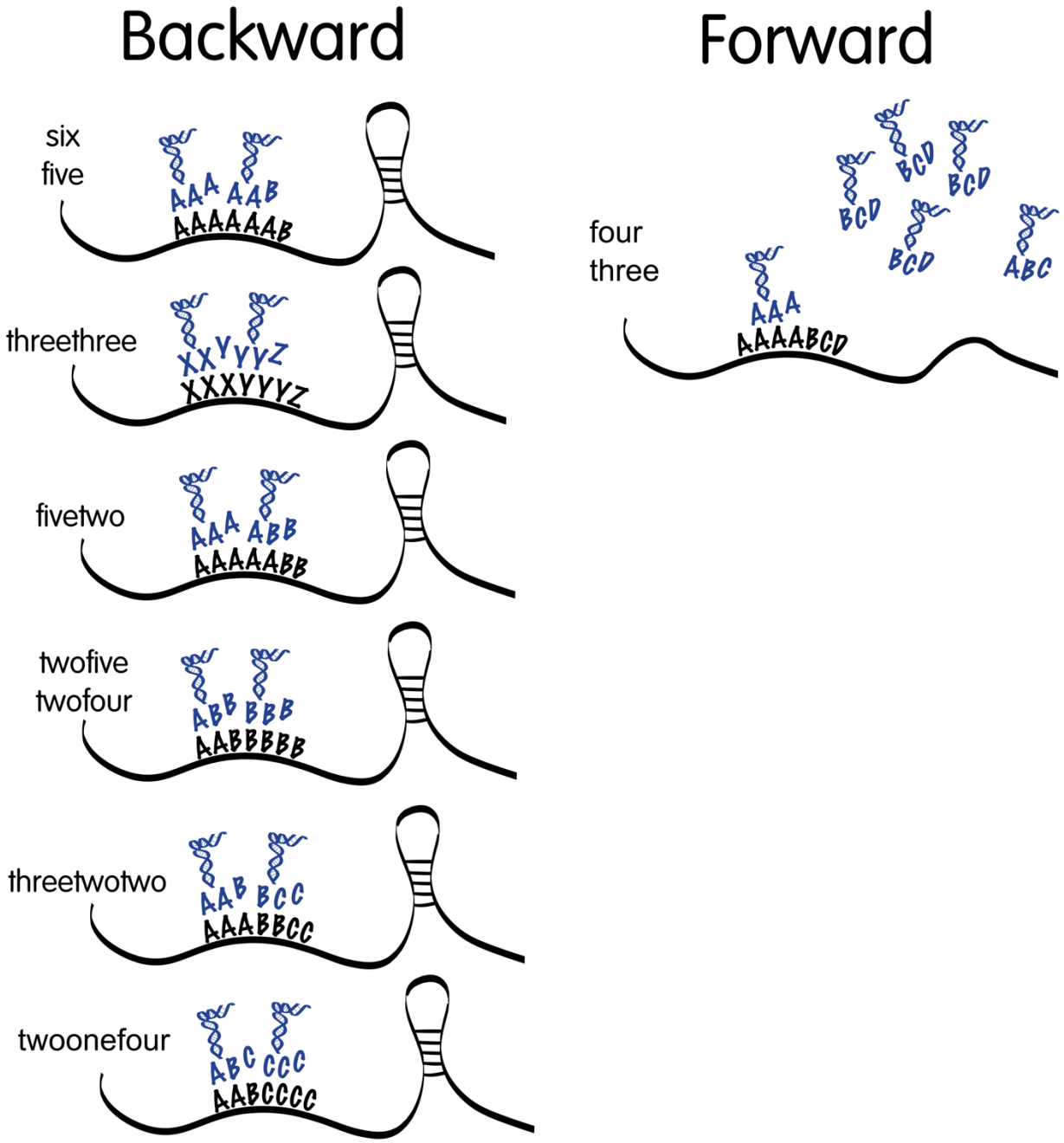


Figure 12 The ten different motifs that PRFect uses as slippery sites to then predict which of those sites find are involved in programmed ribosomal frameshifting. Shown is the originally proposed XXXYYYZ motif for backward frameshifting, along with 9 other proposed motifs that might also allow for the wobble base pairing mismatch initialy discovered with the XXXYYYZ motif. The motifs are named according to the repeated bases (i.e. six is 6 bases in a row), are shown in descending order based on the probabilty of seeing that motif by chance alone, and some are grouped together for brevity. The black letter correspond to the bases (codons) of the mRNA transcript while the blue base correspond to the bases (anti-codons) of the two bound tRNA. The tRNA are shown in their frameshifted position (backwards or forwards) and show how the third weak wobble position of the tRNA might not match the new frameshifted pairing. For example in the original threethree motif the tRNAs XXY and YYZ are bound to the mRNA and then shift back one base so that only their 1st an d 2nd bases are paired correctly while the 3rd wobble bases are mismatched. The probabilty of

seeing three (and four) bases in a row in given genome is quite common, so for the forward frameshifts we also require that the codon of the waiting A-site (ABC) is rarer than the codon of the +1 A-site (BCD). The codon rarities are calcualted at run time on the input genome by iterating through all of its coding genes and counting the occurrence of each of the 64 different codon posibilties. Requiring the +1 A-site to be more common than the waiting +0 A-site codon in forward frameshifts helps to cut down on false-positive preditions but mainly serves to speed up runtimes; since three bases in a row is quite common and calculting the MFE of a window is computational time intensive.

Once all possible motifs were identified, we were left with a set of 106,692 different sites as potential slippery sequences, of which 3,711 were from (*true*) frameshifted genes, and 102,981 were from not thought to be frameshifted gene pairs (*false*). Since the frameshifts are not experimentally tested, we do not know the actual location of the slippery site, only the approximate location that the researcher guessed it to be. Therefore, we cannot ascertain exactly which of the motifs in a *true* frameshift overlap region is the actual slippery sequence. To mitigate error induced by including incorrect, randomly occurring slippery sites into the dataset as *true* cases, any such motifs that occurred further than 10bp away from where the shift was annotated to occur were denoted as *false* cases. This left 2,718 *positive* cases (2,368 backward and 350 forward) and 103,989 *negative* cases.

Properties contributing to translation efficiency (and potential pausing)

For every slippery sequence motif, cellular properties relevant to the translation process were aggregated based on the motif occupying the E-site and P-site of the ribosome, with the A-site being empty and waiting for tRNA recruitment to occur. The first property was the direction of the frameshift, forward or backward. The next two properties were the relative frequency of the waiting +0 A-site codon, and the relative frequency of the -1 or +1 frameshifted A-site codon. The frequencies are found during the genome file reading step by iterating through all the annotated coding genes and counting the relative occurrence of the 64 different codons.

This accounts for the idea that if the codon waiting in the +0 A-site has few matching cognate tRNAs available in the cytoplasm while the tRNA for the ±1 A-site codon is quite abundant, the A1 codon is slightly more favorable than the A0 codon, and so the occurrence of a frameshift is more favorable. The next two properties added were different methods for scoring the presence of a ribosomal binding site (RBS) upstream of the P-site. The first method is a reimplementation of the 28 variable bins utilized by Prodigal for gene calling, where each bin is an integer from 0 to 27 and corresponds to a given RBS motif and nucleotide spacer sequence [14]. The other is a reimplementation of the method employed by the RAST website, which uses the observed frequencies of 191 different RBS motifs with 10 different nucleotide spacer sequence sizes [73]. To estimate the 3' secondary structure, the minimum free energy (MFE) of the 50 bp and 100 bp windows were added using two different secondary structure prediction tools: LinearFold, which predicts simple hairpins, and HotKnots, which can predict pseudoknots. LinearFold produces MFE prediction scores identical to the widely used (but not available as an easily installable Python package) RNAFold program from the Vienna software package [74]. For more complex secondary structure predictions that include pseudoknots, the HotKnots tool was run with the most recent DP09 parameter set [69]. A parameter sweep of the data using pairs of window sizes from 30bp to 120bp, in 10bp increments, and with offsets of 0 to 6, in 3 bp increments was attempted. The results were ambiguous, and no apparent accuracy peaks were observed, so we used the conventional 50bp and 100bp windows taken just after the three A-site bases of the slippery sequence (an offset of 3). The MFE scores were scaled by dividing by the window length, and since the MFE score is biased by the GC content of the window in question [75], we further normalized the MFE/bp by also dividing by the GC content.

The last property added to the model was the number of bases between the slippery sequence and the +0 *in-frame* stop codon. This property helps distinguish between more probable motifs near the *in-frame* stop codon from those that occur randomly much further upstream of the *in-frame* stop codon. For instance, in the example in Figure 11, slippery sites that occur towards the right side of the yellow box would be slightly more probable, or at least differentiable, than those occurring towards the left side of the box.

Table 3 The various cellular properties used to classify slippery sequences

property	Description	range
DIR	The direction of the frameshift	-1, +1
RBS1	The Prodigal ribosomal binding site score	0 - 27
RBS2	The RAST ribosomal binding site score	0.0 - 6.3
MOTIF	The slippery sequence motif	0 - 9
A0	The frequency of the A-site codon usage in all genes	0.0 - 1.0
A1	The frequency of the +1 A-site codon usage in all genes	0.0 - 1.0
LF50	The normalized LinearFold minimum free energy calculation of the 50bp window	0.0 - 1.0
LF100	The normalized LinearFold minimum free energy calculation of the 100bp window	0.0 - 1.0
HK50	The normalized HotKnots minimum free energy calculation of the 50bp window	0.0 - 1.0
HK100	The normalized HotKnots minimum free energy calculation of the 100bp window	0.0 - 1.0
N	The distance of the slippery sequence from the *in-frame* (+0) stop codon	0 - ∞

The eleven (DIR, RBS1, RBS2, MOTIF, A0, A1, LF50, LF100, HK50, HK100, N) properties were then used to train a histogram-based gradient boosting classification tree to predict the direction of the frameshift: 0 for no frameshift, -1 for backwards, and +1 for forward. Since only -1 and +1 ribosomal frameshifts appear in the SEA-PHAGES data, other frameshift size categories (such as -2 and +2) were omitted as predictive categories. The HistGradBoost Classifier module from the Python Scikit-Learn package was used with default parameters

except for the L2 regularization parameter, which was set to 1.0 to help prevent overtraining on the data. In addition, the *early_stopping* was turned off so that the training/validation/testing results would be deterministic rather than stochastic. Since multiple potential slippery sites can occur within the *overlap* region of a single pair of consecutive genes, only the highest-scoring slippery site (if any) is returned as the predicted PRF site by PRFect.

Training and Validation Data

The SEA-PHAGES genomes are highly repetitive; many of the phages have near identical, closely related, taxa in the database, and some phages are exact duplications of other genomes. The presence of multiple copies of the same genome makes splitting the data into training and validation more complex; therefore, four different *leave-one-out* levels were used: CLUSTER, SUBCLUSTER, MASH95, and GENOME. CLUSTER and SUBCLUSTER are the two taxonomic levels that phages are assigned to during the SEA-PHAGES workflow [76]. For example, in a CLUSTER split, all of the phage genomes of one CLUSTER are removed, a HistGradBoost model is trained on the remaining genomes, and then predictions are made for the genomes of the omitted CLUSTER. As there are 89 different CLUSTERS, 89 different validation models were built, and the resulting predictions were merged. Likewise, there are 102 SUBCLUSTERS, hence 102 models were built, and the predictions at that level were merged. Since there are so few CLUSTERS and SUBCLUSTERS represented in the data, and the lowest taxonomic level GENOME is dubious due to the training set contamination discussed above, an intermediary taxonomic level, MASH95, was calculated for all of the genomes. This taxonomic level was assigned by using the genome distance estimation of MASH [77] to cluster the genomes at 95% identity, which is analogous to

a MinHash distance of 0.05. The parameters used were the same as recommended in the publication: a sketch *size* of 400 and *k* of 16.

Testing Data

To assess the performance of *PRFect* on unrelated genomes and non-chaperone genes, a general PRF model was first trained on all of the SEA-PHAGES genomes that contained a *joined* tail assembly chaperone (*TAC*) gene, and then five different sources of frameshift data were tested. The first is the RECODE database [78]; because it is currently unavailable, an archive.org snapshot of the 2010 downloadable files was used. The data includes many types of coding anomalies, spans all domains of life, and covers all gene functions. Each entry in the database corresponds to a single gene, for which the nucleotide sequence and site of the frameshift is given. From the SQL file, 725 entries labeled as "ribosomal_frameshift" were extracted and converted to GenBank files, each of which contained only a single gene. The second source of frameshift data was the set of 28 phage genomes listed with a potential frameshift site in the region analogous to the tail assembly chaperone gene of phage lambda [79]. We downloaded from GenBank the accessions that were listed in the supplementary documents, and for those genomes that were missing the specified frameshift (as a joined CDS feature), we manually added the gene to the file at the specified slippery sequence location. The third source of frameshift data was the FSDB (Frameshift Database) which contains 253 frameshifts in a graphical website [80]. The website was parsed to get the GenBank accession number of each frameshift, the slippery sequence, and the location of the slippery sequence, which was then used to retrieve the files from GenBank. In contrast to the other test datasets, due to the size and number of genomes in the FSDB an automated python script was used to add a *joined*

feature to the GenBank file at the location specified in the frameshift data. Additionally, any pre-existing *joined* features were left in place. The fourth source of frameshift data was 106 virus genomes with known or predicted occurrences of ribosomal frameshifting in genes of different functions [81]. The provided accession numbers were used to download the genome files from GenBank, and as before, frameshifted genes were added to those files lacking the specified feature. The fifth and last source of frameshift data was the quite topical single genome for the coronavirus Covid19. The GenBank file for Covid19 (accession NC_045512) contains 12 genes, one of which is frameshifted.

Results

Validation Sets

To estimate the accuracy of *PRFect*, a leave-one-out training validation approach was used at each of the four different taxonomic levels. At each level, the different groups of that level were iterated, leaving out all genomes of the iterated group, training a model on the remaining groups, predicting frameshifts on the left-out group, and then merging the predictions of all groups. Each potential slippery site either comes from a *joined* gene (i.e. a tail chaperone) or from two non-joined adjacent overlapping genes that happen to have a spuriously occurring slippery sequence motif. The PRFect algorithm was used to predict whether the potential slippery site promotes PRF or not. At the highest taxonomic validation split (leaving out all genomes of the same CLUSTER from training), out of 2,476 *joined* known PRF genes, 1,486 were correctly predicted as having PRF, which is a 61% recall (Figure 13).

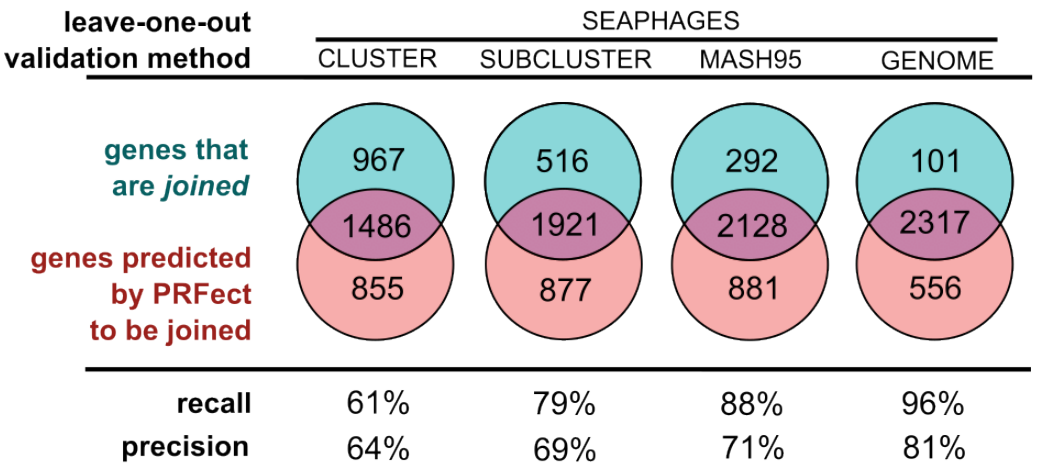


Figure 13) Training and leave-one-out validation on the SEAPHAGES data. While there were 3,679 genomes in the SEAPHAGES database, only 2,476 of them had a tail chaperone gene that was annotated as being frameshifted based on the gene having two locations in the genome linked by the use of the *join* keyword. A simple leave-one-out validation at the GENOME level could not reasonably be used alone to estimate the accuracy of PRFect, since the same genome—or a very close relative—might be present multiple times in the database. Likewise, the CLUSTER|SUBCLUSTER level validation was too broad, so an additional taxonomic level was created using the MINHASH algorithm, and genomes were clustered at 95% genome identity.

As expected, when the models are trained on more data, that includes similar genomes, the recall greatly increases, where at the lowest taxonomic validation split (leaving out only the single respective GENOME and then training on all other GENOMES) there were 2,317 out of the 2,476 *joined* known PRF genes predicted as frameshifting (True Positives), which is a 96% recall rate. As discussed in Methods (Training Data), since nearly the same genome may be found in the SEA-PHAGES data more than once, the GENOME level validation may be biased. Hence, the true real-world recall rate of PRFect falls somewhere between the CLUSTER level and GENOME level, depending on how similar a newly sequenced input genome is to previously known genomes used for training. Not shown in Figure 13 are all of the non-*joined* genes that were correctly predicted as not frameshifting (True Negatives), and only around 1,000 out of

the 360,000 total non-*joined* genes in the SEAPHAGES genomes were incorrectly predicted as having a frameshift, giving the models an *accuracy* of 99%.

Testing Sets

Five alternate data sources were assembled to evaluate the performance of the pre-trained models on data not seen during training and different from the Actinobacteria phage genomes, which are known to have high %GC sequences. These five sources ranged from manually-curated online databases of frameshift data from all domains of life to lists of known frameshifts from publications to the single genome of the Coronavirus SARS-Cov2.

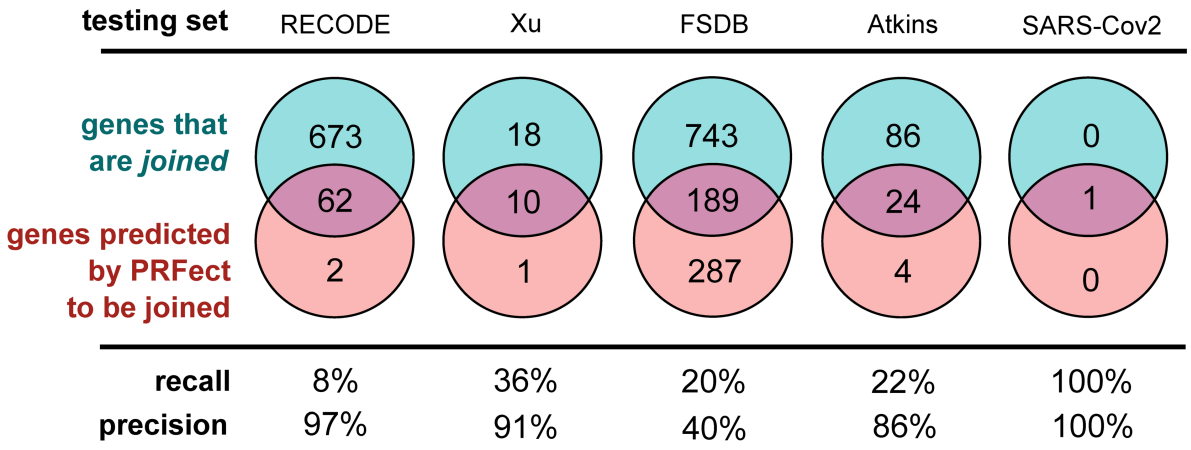


Figure 14) The various performance scores were calculated the same way as the previous validation accuracy, where the predicted slippery site had to belong to a *joined* gene and be within 10 bases of the frameshift annotation to be labeled a True Positive prediction. Predicted slippery sites further than 10 bp, or from genes that were not *joined*, were considered False Positives. True and False Negatives were labeled using a similar scheme.

The RECODE dataset

The RECODE data contains 725 ribosomal frameshift sites, each of which is composed of only the single *joined* gene in question. The database contains 244 backward frameshifts and 481

forward frameshifts. As previously mentioned, the sequence files of the RECODE dataset are composed of only a single gene and not the entire genome. PRFect cannot adequately calculate codon usage to identify the rarity of each codon, which is why all of the 56 correct, true positive predictions for the RECODE data belong to backward frameshifts, while none of the 481 forward frameshifts was predicted correctly.

The Xu phages

As mentioned, the RECODE data is not representative of real-world genomic data since it is only the single frameshifted gene, so we looked for other sources of ribosomal frameshift data to test the accuracy of PRFect. Xu et. al. examined the conservation of the translational frameshift in bacteriophage tail assembly chaperone genes and found 28 phage genomes with two genes that share homology to the two parts of the tail assembly chaperone gene of phage lambda. Out of the 28 *joined* TAC genes in the Xu phage genomes, ten were identified correctly as utilizing ribosomal frameshifting, and only one gene (a hypothetical one) was incorrectly predicted to contain a frameshift.

The Frame Shift Database

The third source of frameshift data was the FSDB (Frameshift Database), which was a comprehensive compilation of experimentally known or computationally predicted data about programmed ribosomal frameshifting. The database contains 253 frameshifts from all domains of life and functions; unfortunately, it has not been updated since its inception fifteen years ago.

Covid19

The last data source for our testing sets was the single genome of SARS-CoV-2, the virus that causes Covid 19. The genome contains 12 genes, one of which is a polyprotein that contains a ribosomal frameshift to translate a much longer version of the polyprotein. PRfect exactly predicts the single frameshift in the polyprotein gene, and without any other False Positive predictions of the other genes.

Discussion

SEAPHAGES

Phage and prophage genomes are being increasingly studied for their roles in almost every facet of human life; from health, to industry, to environment. The SEA-PHAGE initiative is one of many numerous sequencing efforts underway to help better our understanding of the full breadth of genomic diversity and molecular complexities of the bacteriophage world. The function of PRFect is to detect those translationally abnormal genes in phage (and bacterial) genomes that are potentially subject to ribosomal programmed frameshifting. As a side benefit, PRFect also uncovered previously unknown "shifty" sequence motifs that are likely to induce ribosomal slippage and promote frameshifting. One of the issues when working with the SEA-PHAGES genome database, like most sequence databases in general, is that it has a lot of misannotated data. There are no estimates on how prevalent tail assembly chaperone frameshifting is; while it is somewhat conserved among double-stranded tailed phages [79], it is certainly not present in every single SEA-PHAGE genome. Out of the 3,679 SEA-PHAGE genomes we downloaded, only 2,476 contained a *joined* PRF gene (2,397 tail assembly chaperones and 79 hypotheticals). Of the remaining 1,203 SEA-PHAGE genomes without an annotated PRF

gene, a fuzzy word search revealed that 347 had a single *tail assembly chaperone* (TAC) gene, while 342 had two TAC genes. It would be reasonable to presume that a genome with two TAC genes should accordingly have a *joined* PRF gene, so we took the "False Positive "genes (those predicted by PRFect to be joined but that were not joined in the file) at each of the five validation split levels, and grouped them into very general functional categories and a spillover OTHER category (Figure 15). As usual, the genes of *hypothetical* function eclipsed all other categories. The second largest functional category was *tail assembly chaperones*, suggesting that many SEA-PHAGE genomes lacking a *joined* PRF gene were due to errors caused by incomplete annotations. Around 80% of these "false positive" TACs were from genes that were not *joined* in the genome file, while the remaining 20% were from TAC genes that were *joined* in the sequence file, but were considered as *false* due to the location of the predicted frameshift being more than 10bp away from where it was annotated as occurring in the sequence file. It is unknown whether the PRFect prediction is wrong or whether the genome annotation location is wrong. Of the 161 TAC genes where PRFect predicted a different location for the frameshift, there were 136 that did not have any motif at the original location, suggesting that perhaps the genome annotation is wrong. We suspect that many of the genes annotated as encoding *hypothetical* proteins are also TAC genes, since the evidence for adding a *joined* PRF gene to the genome annotation is the presence of two adjacent TAC genes. If the genome is so divergent that one or both of the TAC genes lack sufficient sequence similarity to known TAC genes, they will be reported as *hypothetical*, and subsequently will not be *joined* during the manual annotation of the genome.

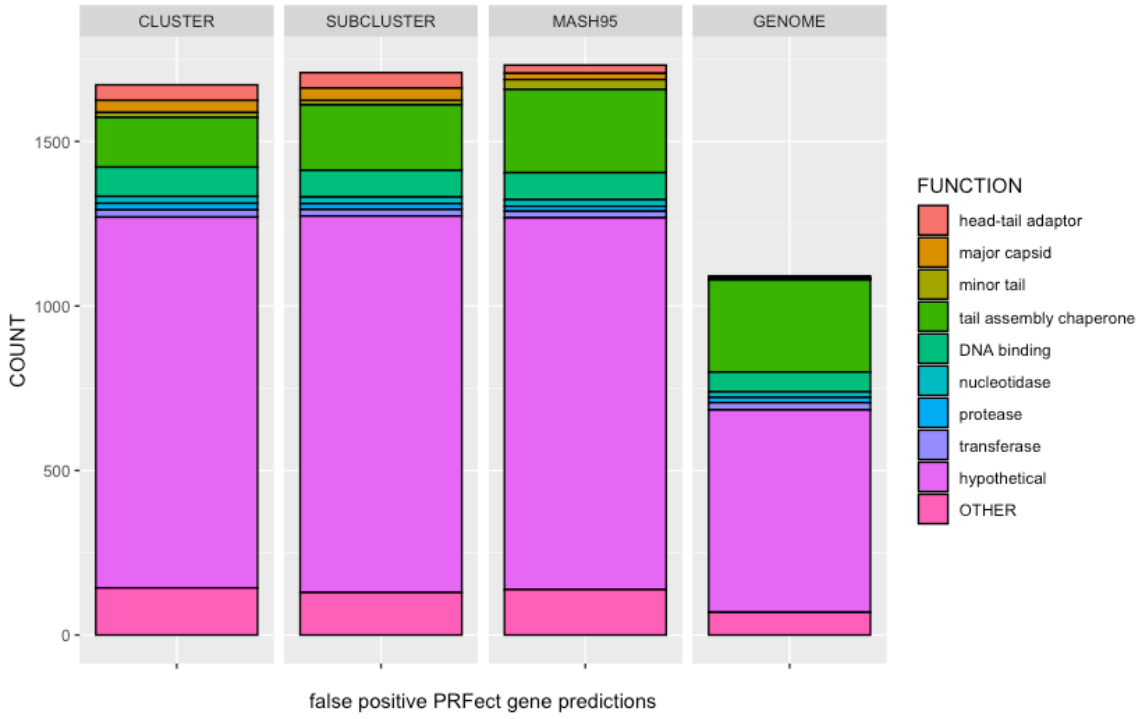


Figure 15) The various gene functions of the false positive PRF predictions during the leave-one-out validation levels. False Positive genes were two adjacent genes predicted by PRFect to contain a frameshift, but that were not joined accordingly. Gene functions were grouped into 12 broad categories based on fuzzy word matching and manual group estimation to simplify the data. Some categories were groupings of similar cellular functions, like the various enzymes, while some were just standardizing the existing function names. For example, there were 25 different spellings of hypothetical protein and 29 different versions of tail assembly chaperone. Gene functions that could not be reasonably assigned to one of the 12 categories were grouped into the OTHER category. The counts are twice the numbers shown in the previous figure because if a gene is predicted by PRFect to be joined when it is not actually a joined gene, it will thus be two genes of potentially different functions

Specific protein functional annotations allowed us to perform a literature search for evidence supporting the true- and false-positive classifications. For example, the gene encoding *Cro* is known to contain a frameshift when expressed in *E. coli* [82]. Other genes in the "False Positives" that are known to contain a frameshift were: *lysins*, *methyltransferases*, *DNA/RNA polymerases*, *IS3 family transposases*, *RusA*/*RuvC*, and *major/minor capsids* [83]–[89]; *minor tail* and *tape measure* genes which could be incorrectly labeled *TAC* genes; as well as *VIP2 ADP-ribosyltransferases* and *MuF-like* which have been found in genomes as a single fusion gene

[90]. The relevance of fusion genes is that some T4-like phages do not utilize ribosomal frameshifting to get the two forms (short and long) of the TAC gene but instead carry one copy of the short gene and one copy of the short gene fused with the second longer part [91]. Therefore, if two genes occur in some genomes as a single fusion gene, there is the possibility that when those genes occur as separate genes in different frames, they may utilize PRF to get the fusion protein during translation.

One last source of concern with the SEA-PHAGES data is that there were 103 genomes that did not have a slippery site motif within 10bp of the annotated putative frameshift site. This could be due to the location being wrong, sequencing error, or the slippery site is not one of the 10 motifs that PRFect looks for. Of those 103 genomes, there were 22 that did not have a slippery site motif anywhere within the overlap region, which suggests that either the slippery sequence is of a motif that PRFect does not look for or that the sequence contains sequencing error.

The RECODE data

One of the more critical cellular properties that characterize forward frameshifts is the ribosome encountering a rare codon, which PRFect cannot adequately determine for the RECODE data because it comprises partial genomes. An alternate mechanism that can also induce forward frameshifts is that instead of a rare codon, the ribosome pauses at a *stop* codon. PRFect treats all of the 64 possible codons equally to calculate the frequency of each codon in the genome, and since stop codons are one per gene, they can appear as rare codons. PRFect still requires that a *three* or *four* motif be present before the *stop* codon, and only 33 of the forward frameshift slippery sites in the RECODE data have such a motif. Of the remaining

448 forward RECODE frameshifts that use a different motif, 295 have the nucleotides CUU, and 139 have the nucleotides UCC just before the *stop* codon. In the first case of CUU_U (the fourth base must be U, since all three *stop* codons start with U), is perfectly base paired with its cognate tRNA GAA, and the ability of G to weakly bind U, allows for the GAA to pair with UU_U in the +1 frame [92]. All of the +1 RECODE frameshifts with the CUU* motif (the asterisk denotes a stop codon) belong to bacterial release factor 2 (*prfB*) genes, which suggests that they comprise a negative feedback loop to finely tune the level of PrfB protein in the cell. When cellular levels of PrfB are high, it binds to the ribosome complex and terminates translation at the CUU* stop codon, resulting in translation of the shorter nonfunctional PrfB protein variant. When cellular levels of PrfB are low, the ribosome encounters the slippery site and pauses much longer since there is no PrfB to terminate translation. Eventually, a forward base slip occurs that shifts the ribosome into the +1 frame, forming the longer functional PrfB protein. When randomizing the codon before the stop, it was shown that the next two most slippery motifs are CCC and UUU [92]; which supports our hypothesis that both *three* and *four* are valid slippery site motifs for +1 forward frameshifts. In the second case of UCC_U, no explanation for the codon:anticodon re-pairing is provided in the literature, and the AGG tRNA has only two base positions that pair (the second and *wobble* third via G:U pairing) with the +1 codon CCU. Interestingly, the gene that is translated by this frameshift is ornithine decarboxylase antizyme (OAZ1), which inhibits polyamine synthesis and import. The +1 frameshift is more frequent at high cellular polyamine levels, leading to more OAZ1 protein, which reduces polyamine levels. One way this is accomplished is that during the translation of OAZ1 in the absence of polyamines, the nascent peptide interacts with the ribosome and prevents its own synthesis,

leading to increased polyamine levels [93]. Polyamines are also known to bind to rRNA (ribosomes), mRNA, and tRNA; therefore, polyamines may alter the translation machinery and make a transient motif out of the UCC nucleotide pattern. Another possibility is that they are not +1 frameshifts at all; it was shown *in vitro* that the mammalian OAZ1 frameshift is ostensibly a +1 shift, though strangely when the exact same sequence was expressed in *S. cerevisiae*, proteomics revealed that the frameshift is reached through a -2 shift rather than a +1 shift [61], [94]. The OAZ1 frameshift also has a pseudoknot 3' of the slippery site, which is usually utilized in backward frameshifts since the downstream secondary structure impedes the forward progress of the ribosome. It is possible that the pseudoknot is only one part of bidirectional PRF control and that polyamines influence the stability of the pseudoknot as another aspect of the negative feedback loop control. If we were to add CUU* and UCC* to the motifs that are searched for, as well as use the entire genome, presumably PRFect would detect the *prfB* and OAZ1 frameshifts correctly (which make up 58% of the RECODE data), and the recall would go up dramatically for the RECODE dataset.

The Xu phages

We had expected that PRFect would perform quite well on the Xu dataset since it was comprised of the same TAC gene used for training. However, PRFect correctly identified 11 of the 28 frameshifted TAC genes from the phage genomes. The cause of three of the missed frameshifts was that the slippery sequence motif at the frameshift sites was not one of those searched for by PRFect. The remaining incorrect predictions seem to be caused by a combination of the GC content being much lower or the distance of the slippery site (N) from the in-frame stop codon being much higher (> 30nt) than the SEA-PHAGES data. The GC content

has been shown to affect the minimum free energy of the downstream secondary structure. The average GC content of the SEA-PHAGES used for the training was 64%, compared to only 47% in the Xu phages. The slippery site's average distance (N) from the in-frame stop codon is 26nt in the training set and 36nt in the Xu dataset.

The FSDB data

Due to the size of the FSDB entries, many of which are full prokaryotic genomes or entire Eukaryotic chromosomes full of thousands of genes, we were unable to manually curate the data to ensure that both the slippery site indicated in the FSDB was present and add it if it was not, or to remove those *joined* genes that were not the indicted frameshift. Although there were only 253 frameshifts listed on the FSDB website, there were 897 total *joined* coding sequences across all of the genomic files. The FSDB data was included as a test set for completeness and illustrative purposes, with the precision and recall performance of PRFect being marginal at best but the result does show the performance accuracy of PRFect. The accuracy is based on the *false-positive* to *true-negative* ratio, and considering that there were 150,000 non-*joined* genes in the dataset and that almost all were accurately predicted as true-negatives, PRFect had an accuracy of >99%.

Atkins

The Atkins database consists of eukaryotic viruses that tend to organize their genes into a single gene spanning the entire genome as one uninterrupted open reading frame [81]. The gene is translated as one large polyprotein which is then later cleaved by proteases into the constituent proteins. Many of the frameshifts were shifts into a frame that contained an early stop codon,

thus acting as a method to create a much shorter version of the polyprotein. Consequently, the length of the slippery site from the *in-frame stop* codon (N) was the entire length of the second frameshifted portion of the gene. The average N for the SEAPHAGES data that was trained upon was 26nt, while the average N for the Atkins genomes was 169nt. The length N is one of the features that was added to help discriminate the True Positive slippery sites near to the *in-frame stop* codon from the *true-negative* slippery sites occurring much farther in the 5' direction of the overlap. Of the 43 genomes with N greater than 51nt, only two were successfully predicted as containing a frameshift by PRFect, and oddly enough, they are -2 frameshifts that present as +1 frameshifts with the slippery site motif of *three* nucleotides in a row (GUU_UUU).

One of the concerns that we have deferred until now that applies to all ribosomal frameshift analysis is that there are more than just -1 (backward) and +1 (forward) frameshifts. Various translational coding anomalies can cause the ribosome to skip around on the mRNA more than just a single base, including -2 shifts, +2 slips, +5 steps, +6 hops, and even a colossal +50bp jump[58], [95]. Despite the range of nucleotides that may be jumped, all frameshifts are still present in the data as either -1 or +1 offset. Because there are only three coding frames per strand direction, so if there is a shift from the 0 frame, regardless of direction, you are either in the -1 or +1 frame. A +2 shift presents the same as a -1 shift, a -2 shift presents the same as a +1 shift, a +4 shift presents the same as a +1 shift, and so on. There is also the possibility of a shift that would put the ribosome back into the 0 frame, i.e. via a -3 or +3 shift. The functional purpose of this would be to skip one or more codons, however there are tRNA reassignments

that allow *st*op codon readthrough (essentially a +3 frameshift), while there are no documented cases of ribosomal programmed frameshifts of multiples of 3 nucleotides.

SARS-CoV-2

The last source of data of the SARS-CoV-2 genome with 12 genes, one of which is a polyprotein that contains a ribosomal frameshift. Unlike many of the Atkins viral genomes with PRFs that cause a shorter version of the polyprotein to be translated, the PRF in SARS-Cov-2 frameshift causes a longer version of the polyprotein to be translated. Thus, the distance of the slippery site from the in-frame stop codon is only 15nt. This short distance is just one of the contributing properties that enables PRFect to perfectly predict the frameshift in the polyprotein gene without any other False Positive predictions on the other genes. This single example shows that despite PRFect being trained on one specific coding gene of prokaryotic phage genomes, it uses universal cellular properties rather than sequence homology. The generalized model allows it to identify frameshifts in a broad range of coding genes and diverse taxonomical clades.

Comparing Performance

PRFect is the first and only tool that predicts programmed ribosomal frameshifts. The only other tools available, FSFinder and KnotInFrame, do not predict PRFs but show all possible locations of a given slippery site motif within a genome. FSFinder finds four motifs: *threethree* for the backward PRFs; and *threeStop*, *UCCstop*, and *CCUstop* for the forward frameshifts. All these motifs appear relatively frequently in a genome by chance, so a second version of the tool (FSFinder2) added the requirement that the motif is located within a gene overlap region (Figure 11) to help reduce the *false-positive* rate, but even this more restrictive version still has

very low *precision* when run on our datasets. FSFinder had the best *precision* (41%) on the RECODE data composed of single genes and the worst *precision* (1%) on the FSDB composed of huge eukaryotic genomes. The recall was not much better: it did find the slippery site of the single frameshifted gene in SARS-Cov-2 but only found around 25%-56% of the slippery site in the frameshifted genes of the other datasets. KnotInFrame looks for only the *threethree* motif with downstream secondary structure, then scores and sorts the results showing however many the user chooses, with a default of 11 (per strand). The performance was also unacceptable: KnotinFrame did find the slippery site in the single frameshifted gene of SARS-Cov-2, but it also reported 21 other matches for the *threethree* motif in the genome. The real design flaw of KnotInFrame was evident in the SEA-PHAGES data, which had 3,479 genomes with 2,476 annotated *joined* genes, where KnotInFrame found 523 of the true-positive slippery sites and more than 48,000 extra (*false-positive*) slippery sites locations (Supplementary Figure 1 and 2).

Future Improvements

We did not consider taking into account the specific nucleotides that can occur in the base positions of a motif during the development of PRFect. The original XXXYYYZ heptamer motif has a consensus that only specific nucleotides can occupy each position: where X can be any nucleotide, Y can be A|U, and Z can be A|U|C [96]. For the SEAPHAGES data, we observed no such positional base limitations, and each of the four nucleotides was found in each of the three XYZ positions of the true-positive joined genes with *threethree* motifs Other motifs had more narrow nucleotide limitations, such as *fivetwo* which had the nucleotides GGGGGAA across all of the thousand motifs of the SEAPHAGES data. The problem with implementing

positional nucleotide constraints is that our SEAPHAGES training data is biased toward high GC% content and highly repetitive, which would cause machine learning models trained on the data to follow the bias. One aspect that could prove to be an improvement to the motifs used by PRFect is considering G:U base pairing of the slippery site codons and the bound tRNAs in the E and P sites. In the previously mentioned *fivetwo* observed motif of GGGGGAA, the second tRNA CUU rebinds with the codon GGA. Only the first position (and the wobble third) are complementary using regular A:U and C:G pairing. However, if G:U base pairing is considered, then all positions are complementary. It may be that some of the motifs that we hypothesize to exist are instead limited to only with specific nucleotides in certain motif positions that allow for G:U base pairing. An example is that the most common bases observed for the hypothetical *threetwotwo* motif (CCCGGAA) have G:U pairing that mimics a *threethree* motif. Though supporting our postulate of new alternative motifs is the fact that the majority of the *twoonefour* motifs cannot be explained through G:U pairing alone.

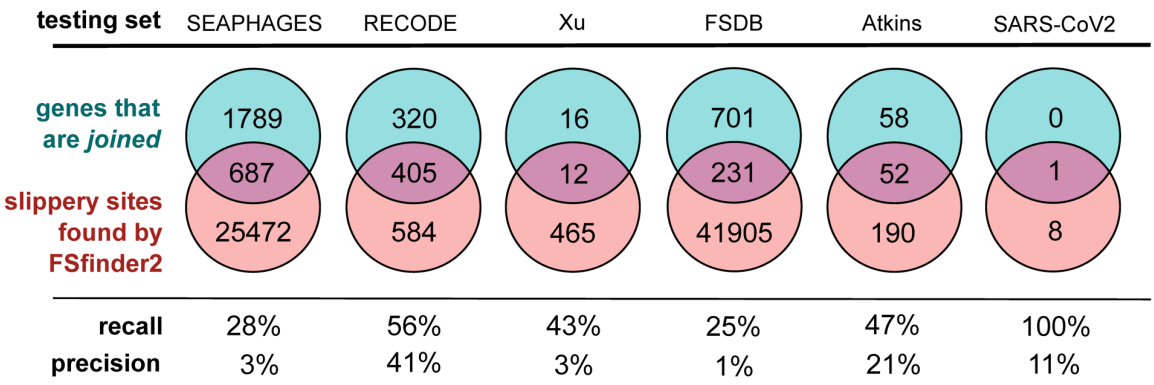
Another possible improvement would be to adjust the codon rarity based on identical codons upstream of the potential frameshift site codon to consider the temporal nature of the tRNA pool. If a codon is repeated once or more in short succession within a coding gene, that tRNA can be consumed quicker than tRNA is recharged. Thus, two or more moderately *infrequent* codons repeated back-to-back would deplete their cognate tRNA from the pool and ultimately act as a rare "hungry" codon. It has been shown that tandem repeats have been shown to induce ribosomal frameshifting in-situ in *E. coli* [97]–[99], are responsible for the frameshifting associated with many human diseases [100], [101], as well translational pausing involved in the synthesis of amino acids and polyamines [102], [103]. Dividing the waiting A-site codon

frequency by the number of times it occurs within some upstream window of a given length would give a more nuanced measure of how "hungry" a rare codon is. Additionally, adjusting the frequency of a given codon by its near-cognates could improve the model even further.

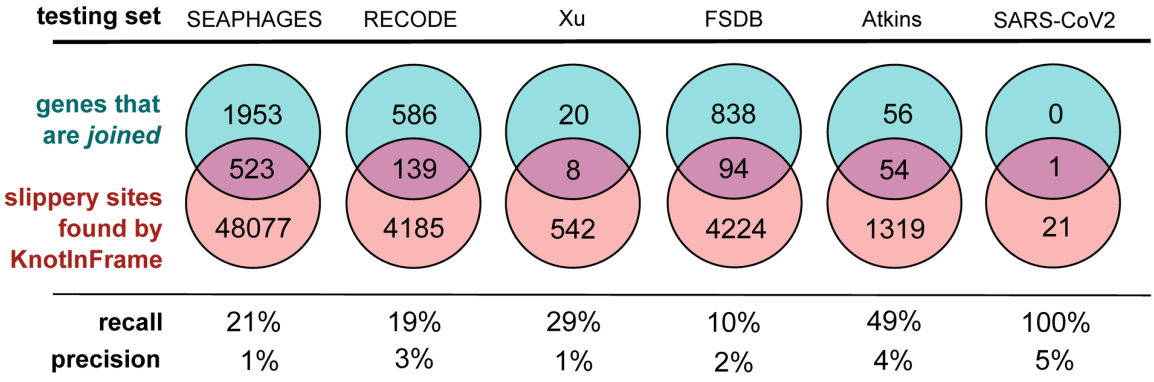
Conclusion

PRFect is the only tool currently available for predicting programmed ribosomal frameshifts in a given genome with a very high degree of accuracy. It is easily installable with a single command, and since the code is open source, we have established a path towards improving PRFect by including additional cellular properties, such as new motifs. We expect PRFect to change and adapt to the field as more is discovered about programmed ribosomal frameshifting and as ever-increasing genomic data is added to the model. For PRFect to predict that two adjacent genes within a genome annotation are one frameshifted gene with its two (or more) constituent parts split into different frames, those parts need to be predicted as distinct genes by gene calling software. The four most popular gene-finding tools, GeneMark, Glimmer, Prodigal, and Phanotate, all function by looking for *start* codon to *stop* codon pairs within the same contiguous frame of the genome [8], [14], [18], [104]. The downstream second part of a ribosomally frameshifted gene does not necessarily have a start codon, so traditional gene prediction algorithms might not find it. However, the codons that can serve as *start* codons (AUG, GUG, and UUG) also occur quite frequently within a gene, where they code for standard amino acids, which generally allows for gene finders to call the downstream fragment as a gene. When the second part of a frameshifted gene does not have any codon matching a valid start codon, we have a new gene finding tool, Genotate, that detects coding regions within a genome without relying on *start* codon to *stop* codon pairs (in preparation). So rather than

relying on other third-party gene finding tools that may not detect all the fragments of a ribosomally frameshifted gene, we will have in place a general workflow where Genotate processes the nucleotide genome to get the many coding regions that essentially mirror the coding genes and fragments, which are then supplied to PRFect in order to predict programmed ribosomal frameshifts between adjacent coding regions.



Supplementary Figure 1 The performance of FSFinder2 on the various datasets.



Supplementary Figure 2 The performance of KnotInFrame on the various datasets.

CHAPTER 4

Introduction

All previous gene finding methods to date for prokaryotes and phages (GeneMark, Glimmer, Prodigal, Phanotate) rely on the assumption that every protein coding gene begins with a *start* codon and ends with *stop* codon. They operate by looking for one of the three most common *start* codons (AUG, GUG, and UUG) and following it to the next in-frame *stop* codon to get all of the potential open reading frames (ORFS), then using various computational methods to distinguish which ORFs are indeed protein coding, and which ORFs do not code for a protein and whose 'starts' and 'stops' occur through change alone (Figure 16).

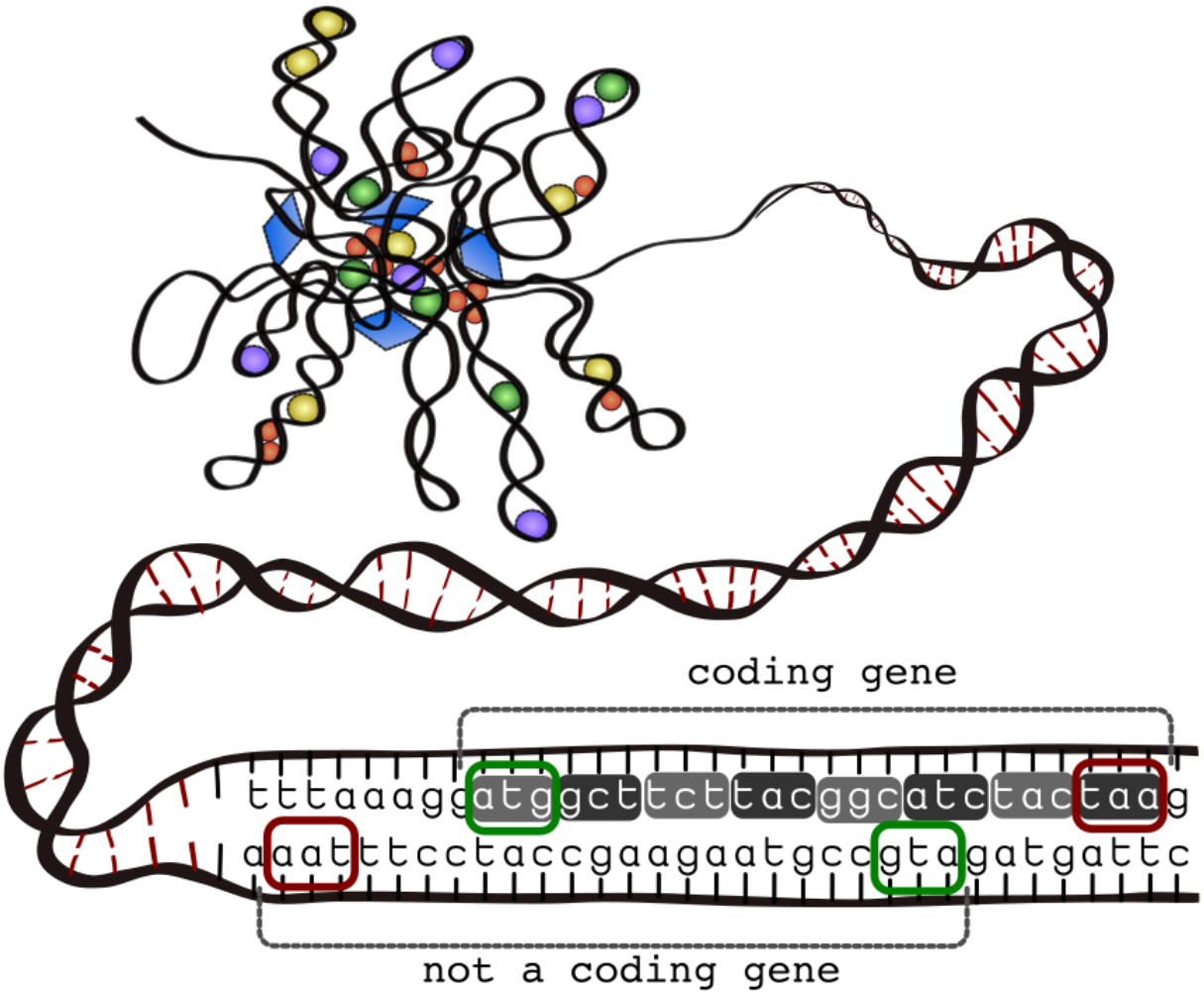


Figure 16) Prokaryotic and phage genomes are co-linear, so they allow for looking in the DNA of the genome for protein coding genes. All current gene finders look for a start codon, in this case ATG (green), and then follow it to the next stop codon, in this case TAA (red). Once all putative coding ORFs are found, various computational methods are used to predict which do indeed code for a protein, and which are only spuriously occurring start and stop codon triplets. The example shows how spurious start and stop codons that are not part of a coding gene, can exist in frames relative to a true coding gene in the forward strand (grey).

The preceding chapter focused on programmed ribosomal frameshifting, presenting an instance where a protein-coding region initiates without a start codon. Other reasons for a coding region to lack a start codon is include, but are not limited to, alternate start codons, inteins, and ever-present sequencing errors. Gene callers employ a process of identifying which open reading frames (ORF) code for a protein by creating a consensus coding signal for the

input genome through a range of biostatistical methods, then designating ORFs that match the consensus signal as coding genes while excluding those ORFs that deviate from it. Commonly considered coding signatures encompass properties such as amino-acid frequencies, positional GC content, ribosomal binding sites, codon and di-codon usage, among others [9], [14], [18], [51] . However, a limitation of this approach lies in its potential to overlook coding genes that deviate from the consensus, as various genome-specific factors can contribute to such deviations. It is well known that the ratios of G and C bases to A and T bases vary across the genome, leading to fluctuations that result in distinct coding signatures for genes situated in regions with varying GC% content [105], [106]. Furthermore, complicating matters is the potential introduction of genes via horizontal gene transfer (HGT), which is notably common in prokaryotes and viruses [107], [108]. Although genes are typically transmitted vertically from one generation to the next, instances of horizontal transfer occur, where genetic material is shared horizontally between related taxa. Prokaryotes and viruses, due to their single-celled nature, can exchange genetic material through three mechanisms of horizontal gene transfer: transformation, conjugation, and transduction.

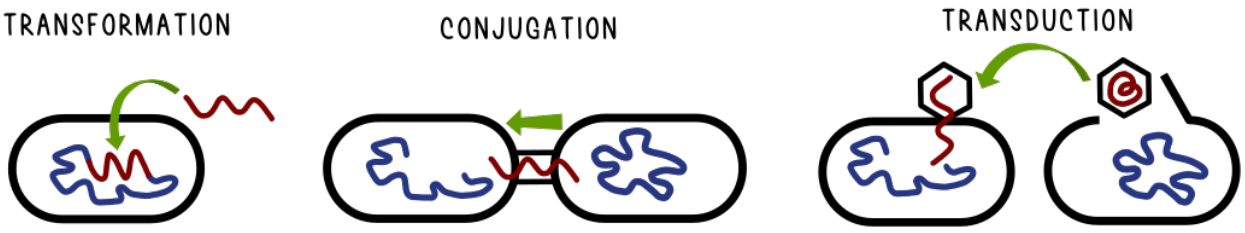


Figure 17 Horizontal gene transfer is the process of an organism incorporating foreign genetic material (red) into its own (blue) genome.

Transformation occurs when exogenous free DNA from the environment and its integration into the genome; conjugation occurs when the foreign genetic material is transferred from another cell via mating; and transduction occurs when genetic material is transferred between cells by a virus (Figure 17). Naturally, a newly acquired gene would not bear the same coding signature biases as preexisting genes, as it has not undergone the same evolutionary pressures.

As a result, using a single consensus signal for coding genes within a bacterial or phage genome would result in a substantial number of false negative gene predictions. Furthermore, gene callers that depend on an ORF finding phase that relies on a fixed set of start and stop codons would lead to the oversight of atypical coding genes, including those with rare start codons, stop codon read-through, and ribosomal frameshifts. Prompted by these limitations, we developed a novel gene caller named Genotate that operates independently of start and stop codons and consensus signals, achieving a higher level of accuracy compared to all other available gene finders.

Genotate operates by utilizing a trained convolutional neural network to classify fixed-size windows, taken from a genome in a scrolling manner. As a consequence, coding genes and intergenic sequence are split into multiple windows. Windows that come from a coding gene

are classified as such while windows taken from intergenic regions or out-of-frame with a coding gene are not classified as coding. The individual window predictions are then analyzed by a change-point algorithm to detect when a coding gene begins or ends. Piecing the coding genes into multiple windows eliminates the necessity for an ORF finding step, enabling the detection of typical coding genes as well as anomalous coding genes.

METHODS

github.com/deprekate/genotate

All of the code and data presented here is available in the above *github* repository. All of the code exclusive to the Genotate package was written in *Python3* in order to be user friendly and easily updateable for future improvements. Genotate is also available on the Python Package Index PyPI (pypi.org) as an easily installable command-line program that downloads and installs with a single command: *pip install genotate*. The Genotate package does require the third-party Python dependencies: *TensorFlow* [109] and *Ruptures* [110] as well as the first party package *genbank,* all of which auto-install when the previous single previous is used to install Genotate. The hyper-parameters of the TensorFlow neural network were determined with KerasTuner (github.com/keras-team/keras-tuner) and the phage genomes by taking the odd accession numbers as training and the even accession numbers as validation. During the final training, the data was instead split into five leave-one-out validation sets described below.

Genotate Algorithm

The workflow of Genotate has three main phases: window labeling, change-point detection, and gene refinement. In the first phase, overlapping 87bp windows are taken incrementally from each of the three forward and three reverse frames of the genome (Figure 18 left). Each window within the

genome is assigned a type determined solely by the center codon. These types are categorized as *intergenic* (I), *coding* (C), and *noncoding* (N). Windows are labeled as *coding* when they align in-frame with a coding gene, while *noncoding* windows are not in-frame and/or located on the strand opposite a coding gene. *Intergenic* windows are precisely defined as the sequence found on both strands between coding genes. The reason for disambiguating *intergenic* and *noncoding*, is that while *intergenic* windows will have random nucleotide frequencies and patterns, *noncoding* windows will have anti-*coding* frequencies and patterns since they overlap with *coding* genes.

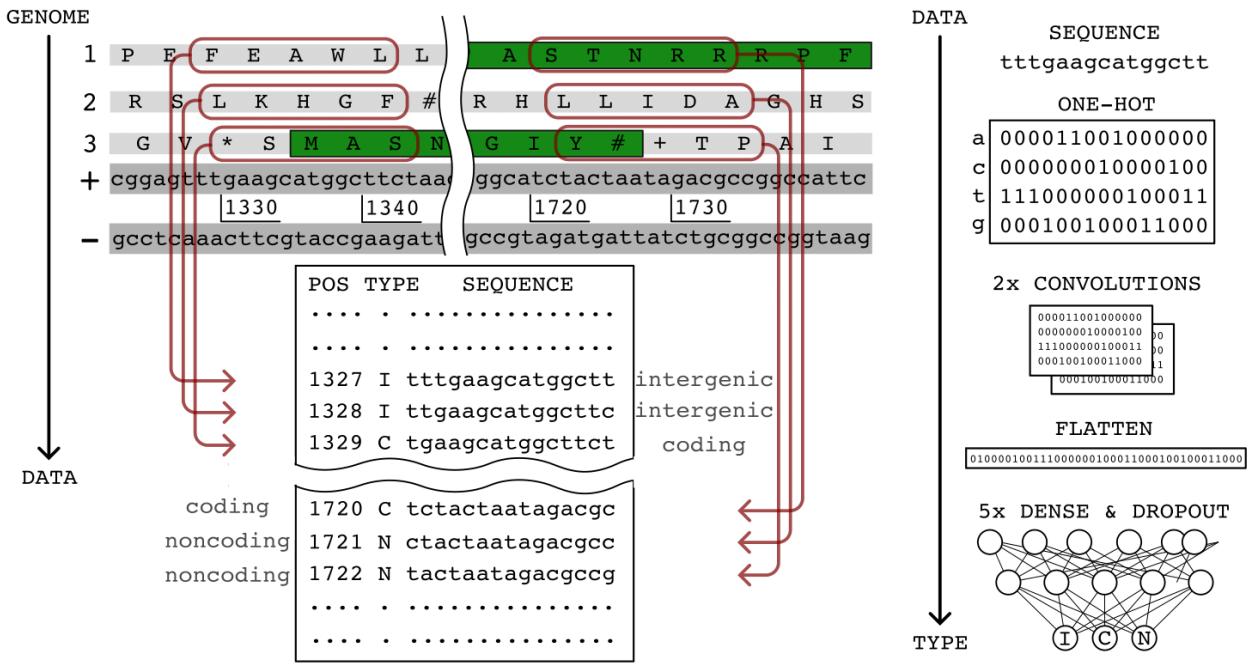


Figure 18) The first phase of Genotate is taking overlapping windows from the genome and then using a neural network to predict the type of window: *coding*, *noncoding*, or *intergenic*. The left panel shows how a genome annotation with coding genes is parsed into overlapping windows of the three different types. The right panel shows how a single window is one-hot encoded and passed through a convolutional neural network in order to predict the type of the window. The model was trained in a reverse fashion, on windows with a known type, using a leave-one-out 5-fold cross validation. The example windows are sized 15bp and are depicted in the three-frame amino acid translations for simplicity (the neural network sees the windows as DNA sequences). The rule that dictates how the type of a window is defined, is based on the type of the center codon, which also loosely correlates to a majority rule. The window at 1329 begins with the stop codon TGA|* yet it is still labeled as *coding* since it's center codon (ATG|M) belong to a *coding* gene. *Noncoding* windows have their center codon, as well as a majority of their codons, out-of-frame with protein coding genes. *Intergenic* windows have their center codon, as well as a majority of their codons, outside of protein coding genes. A GenBank annotation file of genome length *n* nucleotides will yield *2n* rows of windows: *n* windows for the forward frames and *n* windows for the reverse complement frames. To classify the type of each window a convolutional neural network is used. The sequence of each window is *one-hot* encoded then two convolutional layers are performed, followed by a flatten layer, then

five dense layers with dropout between them ending with one layer of three softmax neurons that each give the probability of the window *type* being intergenic, coding, and noncoding.

Each nucleotide window is one-hot encoded, passed through two convolutional layers, five dense layers with dropout, and a softmax layer consisting of three neurons. This last layer outputs the type probabilities (I, C, N) corresponding to that particular window (Figure 18 right).

In the second phase of Genotate, the probability curves are analyzed by the Ruptures change-point detection package to determine the transitions between *intergenic*, *coding*, and *noncoding* frames for each of the six reading frames. Shown in the top of Figure 19 is an example of the three forward frame probability curves for the bacteriophage phiX174 genome, which is a single-stranded DNA virus that has ten protein-coding genes all coded on the same plus [+] strand. For simplicity, only the probability of coding (C) curves are shown; probabilities for *intergenic* (I) and *noncoding* (N) are not displayed; though their curves can be inferred since the base location (bp) of each frame's (I,C,N) curves of sum to one. At the bottom of the figure, the locations of the ten genes in the genome are indicated and color-coded according to their respective frames.

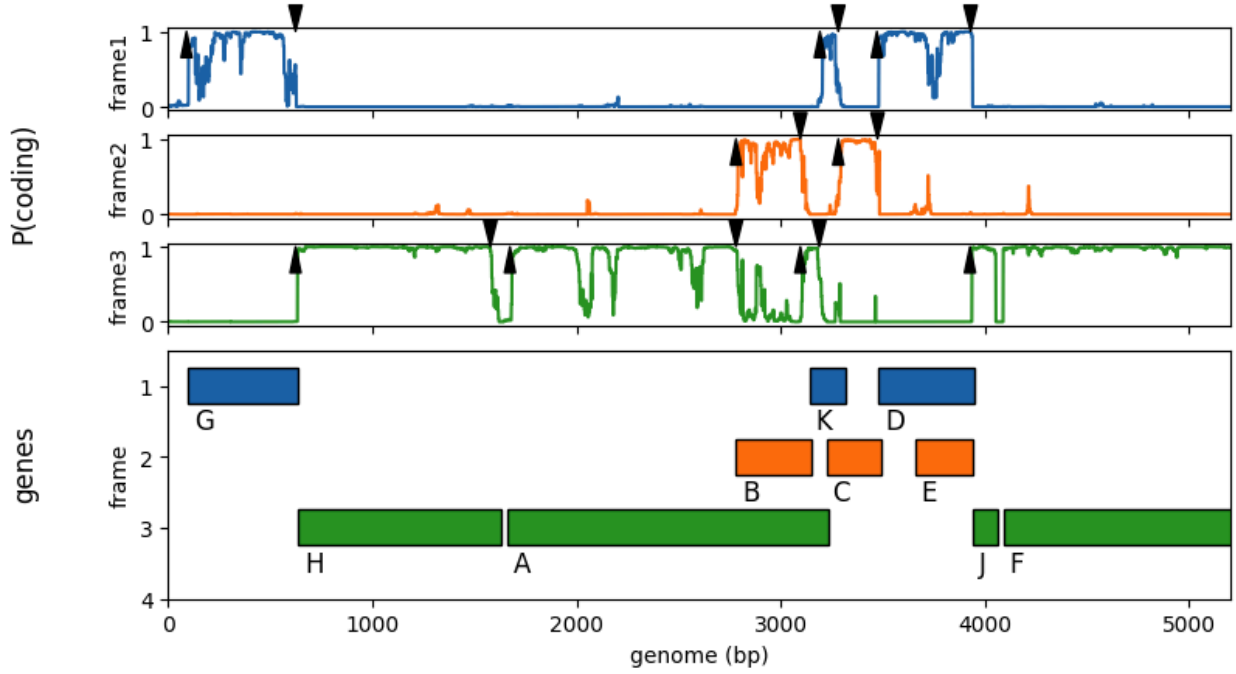


Figure 19) The coding (C) probability curves output by the neural-network, for the forward three frames of the bacteriophage phix174.

The first gene (G) of phiX174 occurs in the frame 1 (blue) from 100-627bp, and a corresponding peak in the Pcoding curve for frame1 is seen over that region. The next two genes (H and A) of phiX174 occur in frame 3 (green), and likewise the Pcoding curve for frame3 peaks from 600 to almost 3,000bp. The black arrowheads are the locations where Ruptures has predicted change-points to occur in that frame; for simplicity, only change-points that involve going into or out of *coding* (C) are shown. The region between two adjacent change-points is labeled as *coding*, *noncoding*, or *intergenic*, by averaging the probabilities of each (I,C,N) curve for that region, and choosing the curve with the highest average. For instance, the peak in the Pcoding of frame1 from 100 to 627bp would cause the average of the *coding* (C) for that region to be somewhere around 1, while the *intergenic* (I) and *noncoding* (N) averages would be around 0. The next region in frame1 between change-points 600-3000bp would end up having an average *coding* (C) probability around 0, with the *noncoding* average around 1 and the *intergenic* also

around 0. In this manner, the coding gene regions can be estimated separately and independently in all six frames of the genome by applying Ruptures to the set of (I,C,N) curves for each frame.

The genome of phiX174 contains several *nested* genes, which is when two genes occur in different frames at the same location in a genome: gene*B* is completely *nested* within gene*A* and gene*E* is completely *nested* within gene*D*. Genotate is unique among gene finders in that it allows completely *nested* genes to be detected, since all other tools enforce mutually exclusive gene calls that prevent the detection of completely *nested* genes. However, Genotate limits gene calls to the three frames of a single mutually exclusive strand for the purpose of improving runtimes and to reduce false-positive gene calls. This does prevent Genotate from detecting completely *nested* genes that occur on opposite strands, but currently there are only a few experimentally validated cases where two genes occur at the same location within a genome completely *nested* but in opposite directions. An example of opposite-strand nested genes can be visualized in Figure 16, where a coding gene exists in the forward direction and a noncoding open reading frame exists in the reverse direction. If that open reading frame in the reverse direction were to actually code for a gene, then the two genes would be opposite-strand nested genes. The existence of an overlapping gene on the antisense strand was first discovered in the envelope (*env*) protein of the human immunodeficiency virus, which encodes a second transmembrane protein (*asp*) in one of the reverse frames [111]; and then in the baseplate gene of bacteriophage T4, which encodes two replication proteins (*repEA* and *repEB*) in the reverse complement frames [112]. Similar instances have been identified in prokaryotic genomes, including the genes *Pic/shET1* of *S. flexneri 2a*, the genes *Pfl_0939/cosA* of *P. fluorescens Pf0-1*, and the iron regulator gene (*dmdR1*) of *S. coelicolor A3(2)* that encodes the secondary metabolism gene *adm* in the reverse frames [113]–[115]. When *nested* genes occur on the opposite strand in eukaryotes, they are located within the introns of the other gene. As a result, they do not truly overlap since the codons of the two genes do not share complementary nucleotide bases.

During transcription the sense strand of the DNA is copied into mRNA from the antisense template strand (Chapter 1 Figure 1). Protein coding genes exist exclusively in the three frames of the single-stranded mRNA transcript, and since opposite strand *nested* genes are extremely rare, only the three frames of the sense strand need to be analyzed. To determine when the sense-strand is oriented in the forward frames of the plus strand and when it is oriented in the reverse complement frames of the minus strand, a preliminary pass is performed. This pass involves summing the six coding (C) probability curves into the *forward* and *reverse* probability curves and then using Ruptures to identify transitions between intergenic, forward, and reverse, within the DNA of the genome. These preliminary pass sense-strand predictions are analogous to mRNA operon transcripts, which means that only those three frames need to be searched for coding genes. In the final pass, each "operon" sub-region's three frames are individually analyzed using the Ruptures package to identify transitions between coding, noncoding, and intergenic regions within each frame. By focusing solely on the three frames of the sense strand, false-positive gene calls in the reverse complement frames are avoided, since an intriguing characteristic of the coding patterns and signals within a genome is that when a coding gene exists in a particular frame, the corresponding direct reverse-complement frame on the opposite strand often exhibits strong coding signals as well. These spurious coding signatures in the 'shadow' of a true-positive gene often result in false positive gene calls in gene detection software.

The Ruptures change-point detection method requires the two parameters: size and penalty. The *size* parameter corresponds to the minimum size allowed between changepoints. In order to precisely resolve coding genes at the single codon level, we set the *size* parameter to one during the final pass. The two discretionary parameters for the *penalty* and the *size* of the preliminary pass were estimated by running each of the 5,572 phage genomes through the Genotate workflow and varying the penalty and size parameters. The penalty parameter was varied from 1 to 20 in increments of 1 and the size parameter was varied from 10 to 100 in increments of 10, and then calculated which parameter

combination had the maximum F3-score on training phage genomes. We chose the F3-score over the F1-score since we care more about false negatives and also suspect that many of the Genotate false positive gene calls actual coding genes that are missing from the genome annotation files. The parameter combination that had the highest F3-score was a size of 60 and a penalty of 1. Incidentally, the parameter combination with the maximum F1-score was a size of 30 and a penalty of 16.

In the third and final phase of Genotate, the predicted coding-regions are analyzed and transformed into putative coding-genes positioned within the genome. The distinction between a coding-region and a coding-gene lies in the fact that gene calls typically encompass a start codon at one end and a stop codon at the other, while coding-region predictions are agnostic to stop and start codons and are not specifically delineated at the ends, potentially resulting in a lack of a valid start or stop codon. In addition, the coding regions may contain internal stop codons, particularly when neighboring genes occur consecutively in the same frame. An example might be if Genotate fails to detect the brief transition to intergenic in the phix174 genome between *geneH* and *geneA* or between *geneJ* and *geneF* (Figure 19). As a result, it may merge one or more coding genes into a single coding region. To properly format the gene calls, we are compelled to incorporate the concept of stop codons into the Genotate workflow. If we were to adjust the coding regions using the conventional approach of naively assuming the genome follows the standard genetic code (codon translation table) and trimming the coding regions to end with one of the three stop codons (UAA, UAG, UGA), we would inevitably encounter the same pitfall faced by all other gene finders. This pitfall involves relying on an initial ORF finding phase that hinges on a predetermined set of start and stop codons. For innovation and to improve the accuracy of Genotate, we leverage the coding-region calls to reveal the genetic code of the input genome, enabling us to predict whether any of the three stop codons are bypassed and read-through with a cognate tRNA [116]. To determine if stop codon read-through occurs, Genotate counts the occurrence of each of the three stop codons (UAA, UAG, UGA) within the last 10 codons of each of the

predicted coding regions, and then compares to the occurrence of stop codons in the other remaining upstream bases. In a genome that utilizes all three stop codons, we would expect to observe them predominantly in the downstream codons, with the exception of where two or more consecutive genes in the same frame were merged into one coding region. In contrast, for a genome that deviates from the standard genetic code and exhibits stop codon read-through, it is anticipated that the frequency of one of the three stop codons will be higher in the middle and upstream bases than in the last few downstream bases of the coding regions. Once the frequencies of the three stop codons in the middle and end regions are counted, if a specific stop codon is found to occur in the middle of the coding-regions a third or more times, that stop codon is considered to be read through in that genome. The stop-codon usage is then used to split the coding-regions at the necessary stop codons, and then the coding-gene calls that do not have a stop-codon in their terminus, the location of their 3' end is adjusted to the next valid downstream stop codon.

Due to the lack of comprehensive data on experimentally validated start codon locations, Genotate does not currently adjust the 5' end of the gene calls to a start codon. However, a similar approach to the stop codon prediction could be employed in the future.

Obtaining Data

To obtain genomic data we downloaded all prokaryotic and viral genomes from the GenBank *Assembly* database that were at the *chromosome* or *complete* assembly level, and excluded genomes marked as *partial* or *anomalous*. Each GenBank genome file contains the genomic (DNA or RNA) sequence and the locations of the genes, as well as other genomic *Features* and associated metadata. Many of the genomes did not have any genes annotated so we calculated the coverage of coding sequence (CDS) genes for each genome and then removed those that had coverages below 90%. We also removed the eukaryotic viruses, leaving only bacterial and archaeal phages. Because the genomes are highly

repetitive; many of them have near identical taxa in the database. In fact, some genomes are exact duplications of other genomes. In order to fairly calculate error metrics during the training/validation/testing steps, the genomes were dereplicated using MASH with default parameters to cluster the genomes at a distance of 0.05, which is analogous to 95% genome identity [77]. For each cluster a representative genome was kept based on the genome with the highest coding gene coverage or as one of the 18 genomes to be used later as a test set, leaving 8,116 prokaryotic genomes and 5,584 phage genomes. The six prokaryotic genomes set aside in order to be used as a test set were from a recent publication that uses the model organisms: *B. subtilis, C. crescentus, E. coli, M. genitalium, P. fluorescens, and S. aureus* to evaluate various gene finding tools [117]. The twelve phage genomes that were set aside as a test set were six phages that infect the previously mentioned bacterial species, respectively: Bacillus phage SPP1, Caulobacter phage CcrColossusus, Escherichia phage T4, Mycoplasma phage P1, Enterobacteria phage PRD1, and Staphylococcus phage PVL. An additional five model genomes were added to the phage test set (Escherichia phiX174, Escherichia phage Mu, Escherichia phage Lambda, Salmonella phage P22, and Mycobacterium phage L5) and CrAssphage cr109_1. The last phage was added as a test genome since it belongs to the clade of highly abundant crAssphages and has been shown to perform stop codon read-through. Interestingly, unlike the *M. genitalium* and Mycoplasma phage P1, which utilize the non-standard Translation Table4, which reads through codon UGA, many of the crAss phage lineages read through the UAG codon [118]. The training data was made by removing the test set genomes along with any other genomes that clustered with them at the 95% identity level.

Training the Models

The genomic data files from the previous section were used to train the convolutional neural network in order to predict the I,C,N type of a given window. A leave-one-out k-fold cross-validation was

performed on the phage genomes and then the bacterial genomes. Each set of training genomes was split into 5 subsets by assigning each genome to a subset based on taking the modulo 5 of the last digit of the GenBank accession number, giving 5 equally sized subsets. For each fold, one subset is set aside and a neural network is trained on the genomes of the remaining four subsets, and the left-out subset is used to calculate the validation error and determine when to stop training. Initially only the phage genomes were used to train the 5 fold models, since the phage dataset was much smaller (0.2GB) than the prokaryotic dataset (21GB). The five models were trained in parallel on 8x Tesla A100 GPUs, which took about four days and 150 epochs. The final epoch weights from the phage training were used to initialize the weights prior to training on the prokaryotic genomes, which took about a year and 50 epochs.

Testing the Models

To evaluate the performance of the Genotate workflow, the nucleotide sequences of the six bacterial and twelve phage genomes that were not included in the training set, were shown to their respective bacterial or phage models. Since five models were trained for both the bacterial and phage genomes, Genotate uses all five models during the window prediction phase and then averages the probabilities into a single set of intergenic, coding, and noncoding (I,C,N) curves. After analyzing the averaged probability curves using Ruptures, Genotate proceeds to generate a GenBank file for each genome, which includes the gene calls obtained through the aforementioned process.

Results and Discussion

The nucleotide FASTA files for the six bacterial genomes were annotated by Genotate and were compared to the predictions from eleven other gene finding tools previously benchmarked with the ORForise tool [117]. We excluded those tools that used taxa-specific models (Human, Staph, E. coli) and instead of using their tool ORForise that uses convoluted logic to match gene calls from a subject to a

reference, we instead relied on the simpler approach of matching putative gene calls to reference gene calls based solely on the identification of the stop codon: if two gene calls both terminate at the same stop codon, then for all intents and purposes they are the same gene. For example, if the reference genome has a gene that starts at X and ends at Y, and the subject genome has a predicted gene call that starts at Z and ends at Y, then that gene is a positive match to the reference.

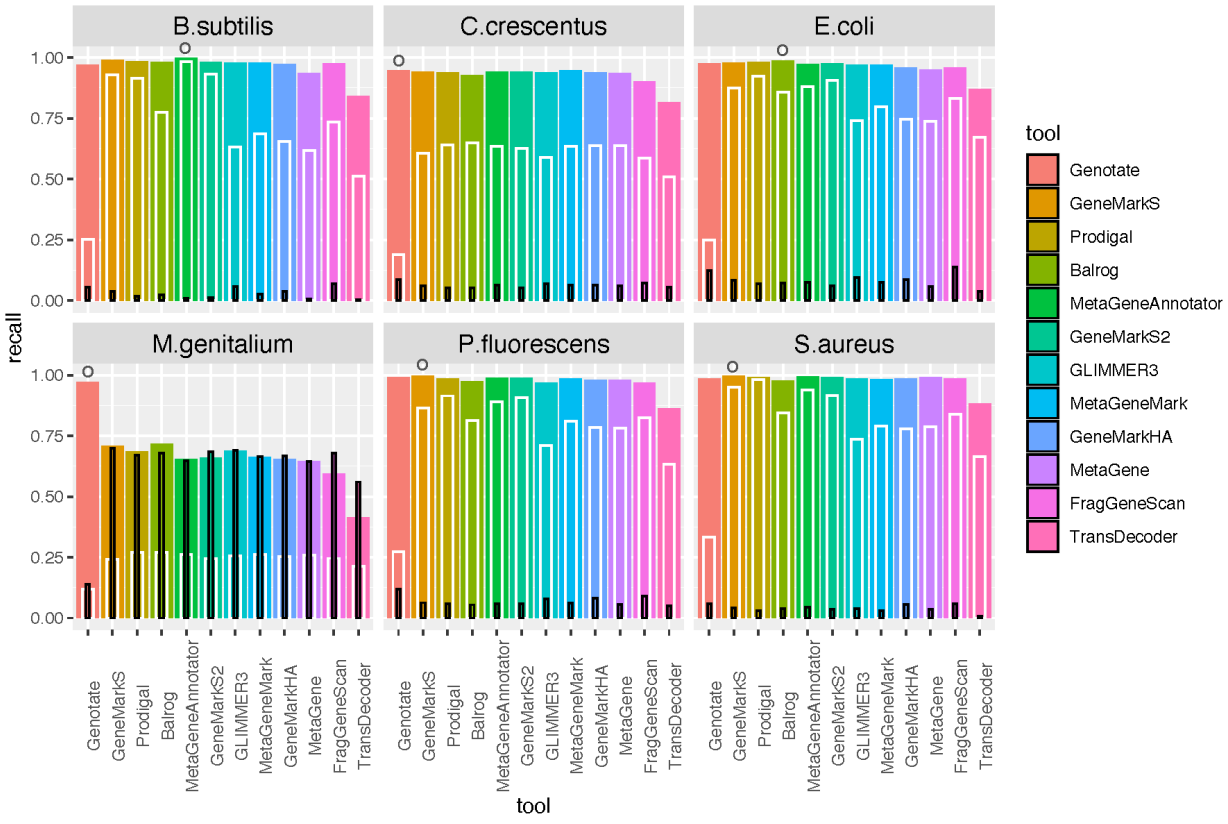


Figure 20) Performance of Genotate versus the eleven other gene finding tools on six different bacterial taxa. The solid bars correspond to the recall based on *partial* matches of only the stop codon, while the white inset bars correspond to the recall for *perfect* matches where a predicted gene call matches to the stop codon and the exact start codon listed in the reference genome. The black bars correspond to the false discovery rate, which is the percentage of the putative gene calls that are false positive. A circle above a column indicates which tool had the highest recall rate for that taxa.

Figure 20 shows the results from the different tools for gene finding on the six model bacterial genomes. The solid bars correspond to the recall, the percentage of genes present in the reference genome that are correctly predicted to occur when the nucleotide sequence of that genome is analyzed with a

respective tool, and are *partial* matches that only take into account whether the predicted gene call and the reference gene have the same stop codon. The white inset bars correspond to the recall for *perfect* matches where a predicted gene call matches to the stop codon and the start codon for a gene listed in the reference genome. The black bars correspond to the false discovery rate, which is the percent of predicted gene calls that are false positives. The tools are sorted in decreasing order of performance, with the best tool on the left followed by the lower performing tools on the right (or top down for the legend) based on the average recall rate for that tool across all six model organisms (Figure 20). The results were fairly comparable for the five bacterial species that utilize the normal translational table, with different tools performing the best on different taxa and mostly fluctuating between 93±1% for *C. crescentus* and 98±1% for the other four taxa. The real standout was for the *Mycoplasma genitalium* genome that performs stop codon read-through and so utilizes a different translational table. This caused every tool except Genotate to have recall values below 75% and false discovery rates around 100%. In contrast, Genotate recognized that the *M. genitalium* genome utilized a non-standard translation table, which allowed Genotate to correctly predict more than 99% of the genes that are currently in the reference genome. The high false discovery rates of the other tools on the *M. genitalium* genome are due to those tools assuming that UGA serves as a stop codon and essentially splitting single genes into multiple fragments at the UGA stop codon. Since the stop codon of those fragments do not exist in the reference genome, those fragments are counted as false positives. Genotate was also the top performing tool on the *C. crescentus* genome, while GeneMarkS was the top performing tool on *P. fluorescens* and *S. aureus*, MetaGeneAnnotator was the top performing tool on *B. subtilis*, and Balrog was the top performing tool on the E. coli genome. One area where Genotate performed worse than the other tools was in the low number of *perfect* gene calls. This is to be expected since unlike all the other tools, its gene calls are not explicitly trimmed to begin at a *start* codon. As mentioned in the Introduction to this chapter, all of the other gene calling tools use a

discrete methodology for calling genes, by first looking for all *start* to *stop* codon pairs within a nucleotide genome, getting the set of all open-reading frames, then predicting which ORFs are protein coding genes and which do not code for a protein. When other tools only get a *partial* match, it is because they have chosen one of the other codons that can serve as start codons (AUG, GUG, UUG) as the location where translation of that gene begins. When not serving as a start codon, the AUG, GUG, or UUG codons only occur within a fraction of the codons in the middle of a gene, around 5% without taking into account any coding biases. In contrast, any codon can be predicted by Genotate as the location of the start of translation, since its method does not rely on first looking for *start* to *stop* codon pairs to get the set of all open-reading frames. If the change-point prediction is off by even a single codon, then that will cause the gene call from Genotate to be labeled as *partial*. The fact that Genotate gets even 25% of the genes calls *perfect* is quite an achievement. Future version of Genotate can implement methods to accurately determine where a gene begins by looking for a start codon and other indicators of translational initiation, such as ribosomal binding sites and/or the lack of secondary structure [119], [120]. Genotate also had higher false discovery rates than the other tools, but this is probably due to the fact the genes in model genomes were annotated by using one or more of the other tools. Any genes present in the genomes that were not correctly predicted by whichever tool was used for gene finding, those genes would be missing from the reference annotation; therefore, if those genes are called by Genotate, they would be labeled as "false positives". This is one of the paradoxical complications when developing and testing new gene finding methods: no new tool can do better than whichever tool was used to annotate the genome, and worse, any genes missed by that tool but called by the new tool are counted as wrong. It is possible that the genes present in the reference genome, but not called by Genotate are not really genes; conversely, some of the genes called by Genotate but absent from the reference genome are, in fact, real genes. If this were the case, then the recall rate of Genotate is underestimated while the false discovery rate is overestimated.

The results on the twelve model phage genomes was more impressive, with Genotate being the top performing tool on every genome (Figure 21). The six phage genomes in the top left corner correspond to the previous six bacterial genomes, which are known hosts for that respective phage. The reason that every tool had a recall rate below 75% for bacteriophage SPP1 is because 25 of the 97 genes present in the current annotation are opposite-strand *nested* genes. Genotate limits *nested* gene calls to a mutually exclusive strand, while the other four tools do not allow any completely *nested* gene calls. As previously discussed, opposite strand *nested* genes are extremely rare, with only a few cases that have been experimentally validated. The probability that 25% of the genes of bacteriophage SPP1 are opposite strand *nested* genes is extremely unlikely. The more obvious explanation is that those gene calls are false positives, and that during annotation open reading frames that do not actually encode for a protein were mistakenly added to the genome. This genome demonstrates just how pervasive annotation error is in genomic data, the same data that was used to train Genotate. Not only do the reference genomes contain an abundance of erroneously *called* genes, but also equal numbers of genes that were not *called*, and so are erroneously lacking from the annotations.

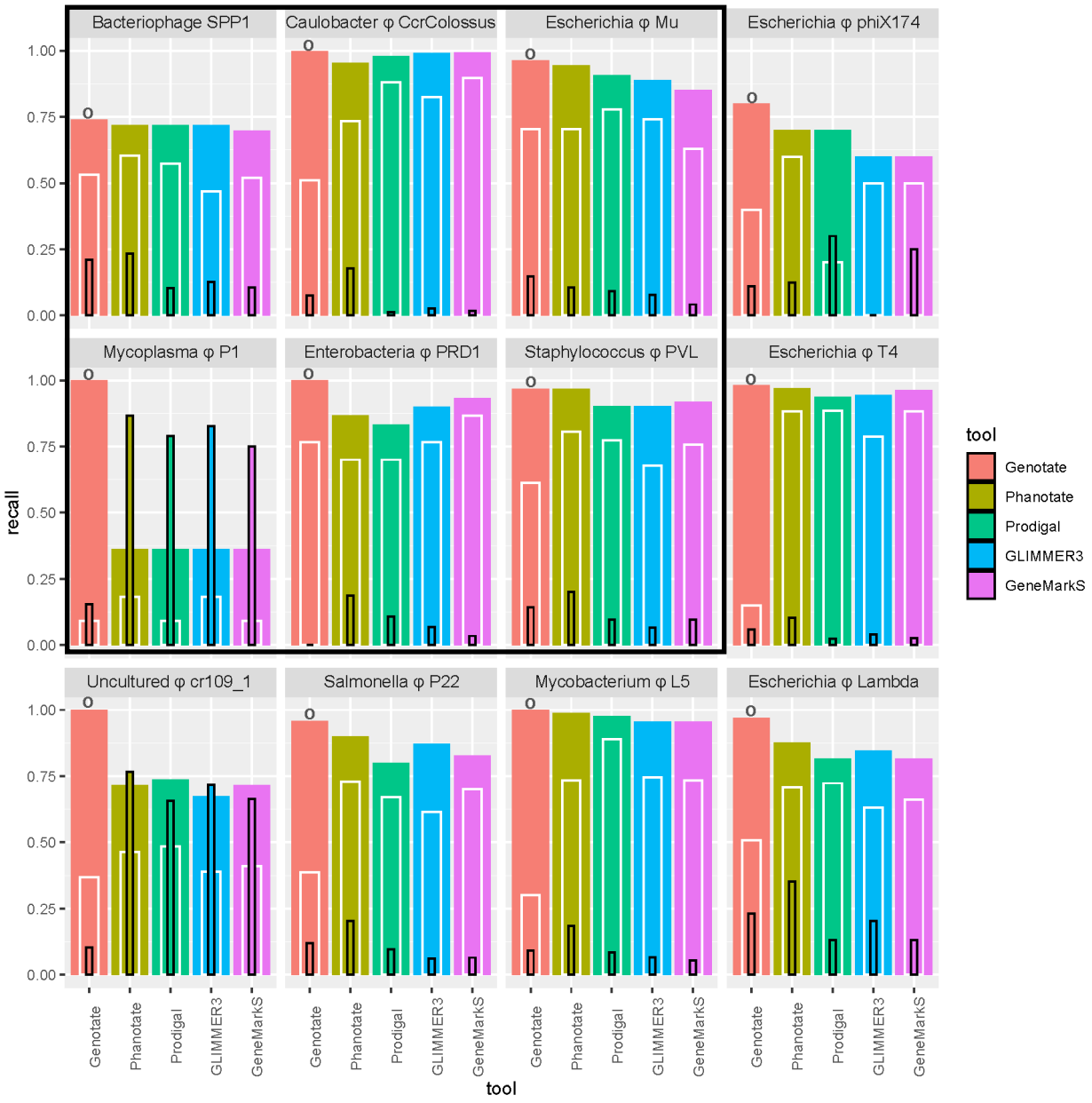


Figure 21) Performance of Genotate versus four other gene finding tools on twelve different bacterial taxa. The solid bars correspond to the recall based on partial matches of only the stop codon, while the white inset bars correspond to the recall for perfect matches where a predicted gene call matches to the stop codon and the exact start codon listed in the reference genome. The black bars correspond to the false discovery rate, which is the percentage of the putative gene calls that are false positive. A dot above a column indicates which tool had the highest recall rate for that taxa.

For the *Caulobacter* phage, Genotate only missied a single gene, one that is missed by all of the tools;

the gene in question is 25 codons (75nt) long, which is below the 90nt length cutoff of the other tools.

The minimum length cutoff of Genotate is much lower, primarily because it does not yet incorporate a start codon refinement phase. For instance, let's consider a hypothetical scenario where there's a 90nt reference gene correctly predicted by the other tools, but predicted by Genotate to only be 87nt and placing an invalid start codon as the first codon of the gene. In this case, if the cutoff were set at 90nt like the other tools, Genotate would be prevented from calling the gene. However, with the implementation of a start codon refinement phase, the beginning of the gene could potentially be adjusted to the correct start codon, thereby restoring the gene's length to 90nt and allowing it to meet the minimum cutoff length. The lower minimum gene length does not give Genotate an unfair advantage since among the reference genomes, there are only nine genes shorter than 90nt and below the minimum cutoff length of the other tools. Out of these nine genes, only five are called by Genotate. If these five gene calls were excluded from the analysis, Genotate would still exhibit the highest recall on all of the model phage genomes, though Genotate and Phanotate would be tied for the highest recall on the *Staphylococcus* phage PVL. Not only was Genotate the tool with the highest recall on the PRD1 genome, correctly calling 100% of the coding genes present in the reference genome, but it also had zero false positive gene calls, giving a perfect annotation. The *Mycoplasma* phage P1, like its host, performs stop codon read-through, and impressively, Genotate identifies the usage of an alternate translation table and accurately predicts an astonishing 100% of the coding genes present in the reference genome annotation. The other four tools all assume that the Mycoplasma phage uses the standard translation table, and correctly predict only around 30% of the coding genes present in the reference genome. The genes of the *Mycoplasma* phage are split into fragments at every internal UGA codon by the other tools, causing a high number of false positive gene calls. The phix174 genome is extremely compact with many of its genes overlapping (Figure 19), three of which are completely *nested*, which is why the other four tools top out at only 7 correctly called genes. Genotate correctly calls 8 of the 10 currently annotated genes. However, when the genome from Figure 19 is analyzed

using the same workflow (which requires only the stop codon), Genotate correctly calls 9 of the 10 coding genes present. The discrepancy arises from the fact that the phiX174 genome is circular, and the genome used for Figure 19 was created by re-circularizing the test set genome at the origin of replication, ensuring that none of the genes were split into two pieces. In contrast, the test set genome for phiX174 has its ends in the middle of *geneA* and *geneB*, causing those two genes to be split into two pieces that are joined in the GenBank file using the 'join' keyword (refer to Chapter 3 Figure 10). Genotate accurately calls both halves of the split *geneB*, which are 135bp and 1404bp. However, it only correctly calls one piece of *geneB* that is 303bp. The other piece of *geneB*, containing the stop codon, is only 51 bases long, which is below the minimum length cutoff, causing it to be missed by Genotate, leading to a missed gene call. When the re-circularized genome of Figure 19 is analyzed using the same workflow, Genotate correctly recalls 9 of the 10 coding genes present in the reference. Genotate only misses *geneE*, presumably because there are genomes in the training set that biologically contain genes similar to both *geneD* and *geneE*, but only *geneD* is present in the GenBank annotation files, while *geneE* is missing due to it being a completely nested gene, typically overlooked by other gene prediction tools. Genotate misses five coding genes from the *Escherichia* phage T4 genome: two small hypotheticals (gp47.1 and ndd.3), one same strand nested gene (gp30.3), and two opposite strand nested genes (repEB and repEA). The T4 genome also contains several other coding anomalies: one programmed ribosomal frameshift and three inteins caused by homing endonucleases. None of these anomalies are delineated at their ends by both start and stop codons. As such, Genotate more accurately predicts the locations of the multi-frame gene fragments, often within ten codons, while the other tools truncate the fragments to a shorter interior start-like codon. Future versions of Genotate that implement a start of translation method to trim the gene regions to a start codon could take the absence of a start codon where a gene is predicted to begin to alert the presence of a coding anomaly, such as rare start codons, inteins, ribosomal frameshifts, and sequencing errors. *Escherichia* phage Lambda is one of the most

studied and well annotated genomes, and Genotate only misses two small l hypothetical genes (Nin57 and orf64). Like phage T4, lambda phage also contains known coding anomalies: a ribosomal frameshift (*geneG* and *geneGT*) and the experimental validated nested genes *Rz* and *Rz'* prime. Genotate perfectly predicts that the *T* fragment of *geneGT* begins at the non-start codon AAG which happens to be part of the known GGGA-AAG slippery site sequence. Surprisingly, Rz' prime is absent from the GenBank file (GCA_000840245.1) downloaded from the 'Assembly' database. This lends credence to the idea that the training set may contain genomes with Rz/Rz' homologs, but only one of the genes is correctly annotated while the other is missing. The Salmonella phage P22 also contains a Rz prime homolog that is missed by Genotate, as well as two other small <90bp hypothetical genes (ninA and orf25) that are missed by Genotate. The last phage genome is not quite a model organism, in fact quite the opposite, it belongs to the uncultured crAssphage isolate cr109_1. The genome was chosen because, like the Mycoplasma phages, it performs stop codon readthrough; though unlike the Mycoplasma phages that read through the UGA codon, some crAssphages read through the UAG codon. Genotate detects that the crAssphage genome performs stop codon read through, though of a different codon than in the *Mycoplasma* phage genome, and correctly calls 100% of the genes present in the reference annotation file.

References

[1] P. J. Torres, R. A. Edwards, and K. A. McNair, “PARTIE: a partition engine to separate metagenomic and amplicon projects in the Sequence Read Archive,” *Bioinformatics*, vol. 33, no. 15, pp. 2389–2391, Aug. 2017, doi: 10.1093/bioinformatics/btx184.

[2] S. Adhya, P. Sarkar, D. Valenzuela, and U. Maitra, “Termination of transcription by Escherichia coli RNA polymerase: influence of secondary structure of RNA transcripts on rho-independent and rho-dependent termination,” *Proc Natl Acad Sci U S A*, vol. 76, no. 4, pp. 1613–1617, Apr. 1979, doi: 10.1073/pnas.76.4.1613.

[3] J. C. Shepherd, “Method to determine the reading frame of a protein from the purine/pyrimidine genome sequence and its possible evolutionary justification.,” *Proceedings of the National Academy of Sciences*, vol. 78, no. 3, pp. 1596–1600, Mar. 1981, doi: 10.1073/pnas.78.3.1596.

[4] M. J. Shulman, C. M. Steinberg, and N. Westmoreland, “The coding function of nucleotide sequences can be discerned by statistical analysis,” *J Theor Biol*, vol. 88, no. 3, pp. 409–420, Feb. 1981, doi: 10.1016/0022-5193(81)90274-5.

[5] R. Staden and A. D. McLachian, “Codon preference and its use in identifying protein coding regions in long DNA sequences,” *Nucleic Acids Research*, vol. 10, no. 1, pp. 141–156, Jan. 1982, doi: 10.1093/nar/10.1.141.

[6] F. Sanger, A. R. Coulson, G. F. Hong, D. F. Hill, and G. B. Petersen, “Nucleotide sequence of bacteriophage λ DNA,” *Journal of Molecular Biology*, vol. 162, no. 4, pp. 729–773, Dec. 1982, doi: 10.1016/0022-2836(82)90546-0.

[7] M. Borodovsky and J. McIninch, “GENMARK: Parallel gene recognition for both DNA strands,” *Computers & Chemistry*, vol. 17, no. 2, pp. 123–133, Jun. 1993, doi: 10.1016/0097-8485(93)85004-V.

[8] A. L. Delcher, D. Harmon, S. Kasif, O. White, and S. L. Salzberg, “Improved microbial gene identification with GLIMMER,” *Nucleic Acids Research*, vol. 27, no. 23, pp. 4636–4641, Dec. 1999, doi: 10.1093/nar/27.23.4636.

[9] A. A. Salamov and V. V. Solovyev, “Ab initio gene finding in Drosophila genomic DNA,” *Genome Res*, vol. 10, no. 4, pp. 516–522, Apr. 2000, doi: 10.1101/gr.10.4.516.

[10] S. Foissac, P. Bardou, A. Moisan, M.-J. Cros, and T. Schiex, “EUGÈNE’HOM: a generic similarity-based gene finder using multiple homologous sequences,” *Nucleic Acids Research*, vol. 31, no. 13, pp. 3742–3745, Jul. 2003, doi: 10.1093/nar/gkg586.

[11] T. S. Larsen and A. Krogh, “EasyGene – a prokaryotic gene finder that ranks ORFs by statistical significance,” *BMC Bioinformatics*, vol. 4, no. 1, p. 21, Jun. 2003, doi: 10.1186/1471-2105-4-21.

[12] M. Stanke, R. Steinkamp, S. Waack, and B. Morgenstern, “AUGUSTUS: a web server for gene finding in eukaryotes,” *Nucleic Acids Res*, vol. 32, no. Web Server issue, pp. W309-312, Jul. 2004, doi: 10.1093/nar/gkh379.

[13] M. Rho, H. Tang, and Y. Ye, “FragGeneScan: predicting genes in short and error-prone reads,” *Nucleic Acids Res*, vol. 38, no. 20, p. e191, Nov. 2010, doi: 10.1093/nar/gkq747.

[14] D. Hyatt, G.-L. Chen, P. F. LoCascio, M. L. Land, F. W. Larimer, and L. J. Hauser, “Prodigal: prokaryotic gene recognition and translation initiation site identification,” *BMC Bioinformatics*, vol. 11, no. 1, p. 119, Mar. 2010, doi: 10.1186/1471-2105-11-119.

[15] H. S. Kang, K. McNair, D. A. Cuevas, B. A. Bailey, A. M. Segall, and R. A. Edwards, “Prophage genomics reveals patterns in phage genome organization and replication.” bioRxiv, p. 114819, Mar. 07, 2017. doi: 10.1101/114819.

[16] J. Cahill *et al.*, “Genetic Analysis of the Lambda Spanins Rz and Rz1: Identification of Functional Domains,” *G3 (Bethesda)*, vol. 7, no. 2, pp. 741–753, Dec. 2016, doi: 10.1534/g3.116.037192.

[17] E. J. Summer, J. Berry, T. A. T. Tran, L. Niu, D. K. Struck, and R. Young, “Rz/Rz1 lysis gene equivalents in phages of Gram-negative hosts,” *J Mol Biol*, vol. 373, no. 5, pp. 1098–1112, Nov. 2007, doi: 10.1016/j.jmb.2007.08.045.

[18] K. McNair, C. Zhou, E. A. Dinsdale, B. Souza, and R. A. Edwards, “PHANOTATE: a novel approach to gene identification in phage genomes,” *Bioinformatics*, vol. 35, no. 22, pp. 4537–4542, Nov. 2019, doi: 10.1093/bioinformatics/btz265.

[19] M. Kearse *et al.*, “Geneious Basic: An integrated and extendable desktop software platform for the organization and analysis of sequence data,” *Bioinformatics*, vol. 28, no. 12, pp. 1647–1649, Jun. 2012, doi: 10.1093/bioinformatics/bts199.

[20] I. T. Rombel, K. F. Sykes, S. Rayner, and S. A. Johnston, “ORF-FINDER: a vector for high-throughput gene identification,” *Gene*, vol. 282, no. 1–2, pp. 33–41, Jan. 2002, doi: 10.1016/s0378-1119(01)00819-8.

[21] D. Arndt *et al.*, “PHASTER: a better, faster version of the PHAST phage search tool,” *Nucleic Acids Res*, vol. 44, no. W1, pp. W16-21, Jul. 2016, doi: 10.1093/nar/gkw387.

[22] “Prokka: rapid prokaryotic genome annotation | Bioinformatics | Oxford Academic.” https://academic.oup.com/bioinformatics/article/30/14/2068/2390517 (accessed Sep. 08, 2023).

[23] S. Roux *et al.*, “Metavir: a web server dedicated to virome analysis,” *Bioinformatics*, vol. 27, no. 21, pp. 3074–3075, Nov. 2011, doi: 10.1093/bioinformatics/btr519.

[24] F. Rohwer and R. Edwards, “The Phage Proteomic Tree: a genome-based taxonomy for phage,” *J Bacteriol*, vol. 184, no. 16, pp. 4529–4535, Aug. 2002, doi: 10.1128/JB.184.16.4529-4535.2002.

[25] C. H. Wu, H. Huang, L.-S. L. Yeh, and W. C. Barker, “Protein family classification and functional annotation,” *Computational Biology and Chemistry*, vol. 27, no. 1, pp. 37–47, Feb. 2003, doi: 10.1016/S1476-9271(02)00098-1.

[26] S. Roux, S. J. Hallam, T. Woyke, and M. B. Sullivan, “Viral dark matter and virus-host interactions resolved from publicly available microbial genomes,” *Elife*, vol. 4, p. e08490, Jul. 2015, doi: 10.7554/eLife.08490.

[27] J. Besemer and M. Borodovsky, “Heuristic approach to deriving models for gene finding,” *Nucleic Acids Res*, vol. 27, no. 19, pp. 3911–3920, Oct. 1999, doi: 10.1093/nar/27.19.3911.

[28] Z. Ouyang, H. Zhu, J. Wang, and Z.-S. She, “Multivariate entropy distance method for prokaryotic gene identification,” *J Bioinform Comput Biol*, vol. 2, no. 2, pp. 353–373, Jun. 2004, doi: 10.1142/s0219720004000624.

[29] S. Akhter, R. K. Aziz, and R. A. Edwards, “PhiSpy: a novel algorithm for finding prophages in bacterial genomes that combines similarity- and composition-based strategies,” *Nucleic Acids Res*, vol. 40, no. 16, p. e126, Sep. 2012, doi: 10.1093/nar/gks406.

[30] R. Bellman, “On a routing problem,” *Quart. Appl. Math.*, vol. 16, no. 1, pp. 87–90, 1958, doi: 10.1090/qam/102435.

[31] L. R. Ford and D. R. Fulkerson, “Maximal Flow Through a Network,” *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, Jan. 1956, doi: 10.4153/CJM-1956-045-5.

[32] J. H. Badger and G. J. Olsen, “CRITICA: coding region identification tool invoking comparative analysis,” *Mol Biol Evol*, vol. 16, no. 4, pp. 512–524, Apr. 1999, doi: 10.1093/oxfordjournals.molbev.a026133.

[33] D. A. Benson *et al.*, “GenBank,” *Nucleic Acids Res*, vol. 45, no. D1, pp. D37–D42, Jan. 2017, doi: 10.1093/nar/gkw1070.

[34] “SciPy: Open Source Scientific Tools for Python | BibSonomy.” https://www.bibsonomy.org/bibtex/24b71448b262807648d60582c036b8e02/neurokernel (accessed Sep. 08, 2023).

[35] S. Seabold and J. Perktold, “Statsmodels: Econometric and Statistical Modeling with Python,” *Proceedings of the 9th Python in Science Conference*, pp. 92–96, 2010, doi: 10.25080/Majora-92bf1922-011.

[36] NCBI Resource Coordinators, “Database resources of the National Center for Biotechnology Information,” *Nucleic Acids Res*, vol. 44, no. D1, pp. D7-19, Jan. 2016, doi: 10.1093/nar/gkv1290.

[37] S. M. Kiełbasa, R. Wan, K. Sato, P. Horton, and M. C. Frith, “Adaptive seeds tame genomic sequence comparison,” *Genome Res*, vol. 21, no. 3, pp. 487–493, Mar. 2011, doi: 10.1101/gr.113985.110.

[38] S. L. Sheetlin, Y. Park, M. C. Frith, and J. L. Spouge, “Frameshift alignment: statistics and post-genomic applications,” *Bioinformatics*, vol. 30, no. 24, pp. 3575–3582, Dec. 2014, doi: 10.1093/bioinformatics/btu576.

[39] J. Fowler, L. Cohen, and P. Jarvis, *Practical Statistics for Field Biology*. John Wiley & Sons, 2013.

[40] S. Nakagawa and I. C. Cuthill, “Effect size, confidence interval and statistical significance: a practical guide for biologists,” *Biol Rev Camb Philos Soc*, vol. 82, no. 4, pp. 591–605, Nov. 2007, doi: 10.1111/j.1469-185X.2007.00027.x.

[41] C. K. Fagerquist *et al.*, “Top-down proteomic identification of Shiga toxin 2 subtypes from Shiga toxin-producing Escherichia coli by matrix-assisted laser desorption ionization-tandem time of flight

mass spectrometry," *Appl Environ Microbiol*, vol. 80, no. 9, pp. 2928–2940, May 2014, doi: 10.1128/AEM.04058-13.

[42] W. H. Pope *et al.*, "Genomics and proteomics of mycobacteriophage patience, an accidental tourist in the Mycobacterium neighborhood," *mBio*, vol. 5, no. 6, p. e02145, Dec. 2014, doi: 10.1128/mBio.02145-14.

[43] J. L. Mokili, F. Rohwer, and B. E. Dutilh, "Metagenomics and future perspectives in virus discovery," *Curr Opin Virol*, vol. 2, no. 1, pp. 63–77, Feb. 2012, doi: 10.1016/j.coviro.2011.12.004.

[44] R. A. Kastelein, E. Remaut, W. Fiers, and J. van Duin, "Lysis gene expression of RNA phage MS2 depends on a frameshift during translation of the overlapping coat protein gene," *Nature*, vol. 295, no. 5844, pp. 35–41, Jan. 1982, doi: 10.1038/295035a0.

[45] R. D. Fleischmann *et al.*, "Whole-genome random sequencing and assembly of Haemophilus influenzae Rd," *Science*, vol. 269, no. 5223, pp. 496–512, Jul. 1995, doi: 10.1126/science.7542800.

[46] H. Zhu, G.-Q. Hu, Y.-F. Yang, J. Wang, and Z.-S. She, "MED: a new non-supervised gene prediction algorithm for bacterial and archaeal genomes," *BMC Bioinformatics*, vol. 8, p. 97, Mar. 2007, doi: 10.1186/1471-2105-8-97.

[47] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, Art. no. 7825, Sep. 2020, doi: 10.1038/s41586-020-2649-2.

[48] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. null, pp. 2825–2830, Nov. 2011.

[49] "Hunter: Matplotlib: A 2D graphics environment - Google Scholar." https://scholar.google.com/scholar_lookup?title=Matplotlib:+A+2D+Graphics+Environment&author=Hunter,+J.D.&publication_year=2007&journal=Comput.+Sci.+Eng.&volume=9&pages=90%E2%80%9395&doi=10.1109/MCSE.2007.55 (accessed Sep. 08, 2023).

[50] K. Wilson and B. Ely, "Analyses of four new Caulobacter Phicbkviruses indicate independent lineages," *J Gen Virol*, vol. 100, no. 2, pp. 321–331, Feb. 2019, doi: 10.1099/jgv.0.001218.

[51] H. Noguchi, J. Park, and T. Takagi, "MetaGene: prokaryotic gene finding from environmental genome shotgun sequences," *Nucleic Acids Res*, vol. 34, no. 19, pp. 5623–5630, Nov. 2006, doi: 10.1093/nar/gkl723.

[52] K. J. Hoff, T. Lingner, P. Meinicke, and M. Tech, "Orphelia: predicting genes in metagenomic sequencing reads," *Nucleic Acids Res*, vol. 37, no. Web Server issue, pp. W101-105, Jul. 2009, doi: 10.1093/nar/gkp327.

[53] K. Gemayel, A. Lomsadze, and M. Borodovsky, "MetaGeneMark-2: Improved Gene Prediction in Metagenomes." bioRxiv, p. 2022.07.25.500264, Jul. 27, 2022. doi: 10.1101/2022.07.25.500264.

[54] D. Hyatt, P. F. LoCascio, L. J. Hauser, and E. C. Uberbacher, "Gene and translation initiation site prediction in metagenomic sequences," *Bioinformatics*, vol. 28, no. 17, pp. 2223–2230, Sep. 2012, doi: 10.1093/bioinformatics/bts429.

[55] J. F. Atkins and R. F. Gesteland, “The synthetase gene of the RNA phages R17, MS2 and f2 has a single UAG terminator codon,” *Molec. Gen. Genet.*, vol. 139, no. 1, pp. 19–31, Mar. 1975, doi: 10.1007/BF00267992.

[56] J. F. Atkins, R. F. Gesteland, B. R. Reid, and C. W. Anderson, “Normal tRNAs promote ribosomal frameshifting,” *Cell*, vol. 18, no. 4, pp. 1119–1131, Dec. 1979, doi: 10.1016/0092-8674(79)90225-3.

[57] T. Jacks, K. Townsley, H. E. Varmus, and J. Majors, “Two efficient ribosomal frameshifting events are required for synthesis of mouse mammary tumor virus gag-related polyproteins,” *Proc Natl Acad Sci U S A*, vol. 84, no. 12, pp. 4298–4302, Jun. 1987, doi: 10.1073/pnas.84.12.4298.

[58] R. B. Weiss, D. M. Dunn, J. F. Atkins, and R. F. Gesteland, “Slippery runs, shifty stops, backward steps, and forward hops: -2, -1, +1, +2, +5, and +6 ribosomal frameshifting,” *Cold Spring Harb Symp Quant Biol*, vol. 52, pp. 687–693, 1987, doi: 10.1101/sqb.1987.052.01.078.

[59] B. Larsen, N. M. Wills, R. F. Gesteland, and J. F. Atkins, “rRNA-mRNA base pairing stimulates a programmed -1 ribosomal frameshift,” *J Bacteriol*, vol. 176, no. 22, pp. 6842–6851, Nov. 1994, doi: 10.1128/jb.176.22.6842-6851.1994.

[60] T. Jacks, H. D. Madhani, F. R. Masiarz, and H. E. Varmus, “Signals for ribosomal frameshifting in the rous sarcoma virus gag-pol region,” *Cell*, vol. 55, no. 3, pp. 447–458, Nov. 1988, doi: 10.1016/0092-8674(88)90031-1.

[61] S. Matsufuji *et al.*, “Autoregulatory frameshifting in decoding mammalian ornithine decarboxylase antizyme,” *Cell*, vol. 80, no. 1, pp. 51–60, Jan. 1995, doi: 10.1016/0092-8674(95)90450-6.

[62] W.-P. Huang, C.-P. Cho, and K.-Y. Chang, “mRNA-Mediated Duplexes Play Dual Roles in the Regulation of Bidirectional Ribosomal Frameshifting,” *International Journal of Molecular Sciences*, vol. 19, no. 12, Art. no. 12, Dec. 2018, doi: 10.3390/ijms19123867.

[63] C. Roman, A. Lewicka, D. Koirala, N.-S. Li, and J. A. Piccirilli, “The SARS-CoV-2 Programmed −1 Ribosomal Frameshifting Element Crystal Structure Solved to 2.09 Å Using Chaperone-Assisted RNA Crystallography,” *ACS Chem. Biol.*, vol. 16, no. 8, pp. 1469–1481, Aug. 2021, doi: 10.1021/acschembio.1c00324.

[64] C. Theis, J. Reeder, and R. Giegerich, “KnotInFrame: prediction of −1 ribosomal frameshift events,” *Nucleic Acids Res*, vol. 36, no. 18, pp. 6013–6020, Oct. 2008, doi: 10.1093/nar/gkn578.

[65] Y. Byun, S. Moon, and K. Han, “A general computational model for predicting ribosomal frameshifts in genome sequences,” *Comput Biol Med*, vol. 37, no. 12, pp. 1796–1801, Dec. 2007, doi: 10.1016/j.compbiomed.2007.06.001.

[66] P.-Y. Liao, Y. S. Choi, and K. H. Lee, “FSscan: a mechanism-based program to identify +1 ribosomal frameshift hotspots,” *Nucleic Acids Research*, vol. 37, no. 21, pp. 7302–7311, Nov. 2009, doi: 10.1093/nar/gkp796.

[67] M. Mikl, Y. Pilpel, and E. Segal, “High-throughput interrogation of programmed ribosomal frameshifting in human cells,” *Nat Commun*, vol. 11, no. 1, Art. no. 1, Jun. 2020, doi: 10.1038/s41467-020-16961-8.

[68] L. Huang *et al.*, “LinearFold: linear-time approximate RNA folding by 5’-to-3’ dynamic programming and beam search,” *Bioinformatics*, vol. 35, no. 14, pp. i295–i304, Jul. 2019, doi: 10.1093/bioinformatics/btz375.

[69] J. REN, B. RASTEGARI, A. CONDON, and H. H. HOOS, “HotKnots: Heuristic prediction of RNA secondary structures including pseudoknots,” *RNA*, vol. 11, no. 10, pp. 1494–1504, Oct. 2005, doi: 10.1261/rna.7284905.

[70] W. H. Pope *et al.*, “Expanding the Diversity of Mycobacteriophages: Insights into Genome Architecture and Evolution,” *PLOS ONE*, vol. 6, no. 1, p. e16329, Jan. 2011, doi: 10.1371/journal.pone.0016329.

[71] E. W. Sayers *et al.*, “Database resources of the national center for biotechnology information,” *Nucleic Acids Res*, vol. 50, no. D1, pp. D20–D26, Jan. 2022, doi: 10.1093/nar/gkab1112.

[72] T. L. Bailey, J. Johnson, C. E. Grant, and W. S. Noble, “The MEME Suite,” *Nucleic Acids Res*, vol. 43, no. W1, pp. W39-49, Jul. 2015, doi: 10.1093/nar/gkv416.

[73] R. Overbeek *et al.*, “The SEED and the Rapid Annotation of microbial genomes using Subsystems Technology (RAST),” *Nucleic Acids Res*, vol. 42, no. Database issue, pp. D206-214, Jan. 2014, doi: 10.1093/nar/gkt1226.

[74] R. Lorenz *et al.*, “ViennaRNA Package 2.0,” *Algorithms Mol Biol*, vol. 6, p. 26, Nov. 2011, doi: 10.1186/1748-7188-6-26.

[75] E. Trotta, “On the Normalization of the Minimum Free Energy of RNAs by Sequence Length,” *PLoS ONE*, vol. 9, no. 11, p. e113380, Nov. 2014, doi: 10.1371/journal.pone.0113380.

[76] G. F. Hatfull, “Mycobacteriophages: genes and genomes,” *Annu Rev Microbiol*, vol. 64, pp. 331–356, 2010, doi: 10.1146/annurev.micro.112408.134233.

[77] B. D. Ondov *et al.*, “Mash: fast genome and metagenome distance estimation using MinHash,” *Genome Biology*, vol. 17, no. 1, p. 132, Jun. 2016, doi: 10.1186/s13059-016-0997-x.

[78] P. V. Baranov *et al.*, “RECODE: a database of frameshifting, bypassing and codon redefinition utilized for gene expression,” *Nucleic Acids Res*, vol. 29, no. 1, pp. 264–267, Jan. 2001.

[79] J. Xu, R. W. Hendrix, and R. L. Duda, “Conserved Translational Frameshift in dsDNA Bacteriophage Tail Assembly Genes,” *Molecular Cell*, vol. 16, no. 1, pp. 11–21, Oct. 2004, doi: 10.1016/j.molcel.2004.09.006.

[80] S. Moon, Y. Byun, and K. Han, “FSDB: A frameshift signal database,” *Comput Biol Chem*, vol. 31, no. 4, pp. 298–302, Aug. 2007, doi: 10.1016/j.compbiolchem.2007.05.004.

[81] J. F. Atkins, G. Loughran, P. R. Bhatt, A. E. Firth, and P. V. Baranov, “Ribosomal frameshifting and transcriptional slippage: From genetic steganography and cryptography to adventitious use,” *Nucleic Acids Res*, vol. 44, no. 15, pp. 7007–7078, Sep. 2016, doi: 10.1093/nar/gkw530.

[82] S. Escobedo, I. Rodríguez, P. García, J. E. Suárez, and B. Carrasco, “Differential expression of cro, the lysogenic cycle repressor determinant of bacteriophage A2, in Lactobacillus casei and Escherichia coli,” *Virus Res*, vol. 183, pp. 63–66, Apr. 2014, doi: 10.1016/j.virusres.2014.01.010.

[83] C. A. Shearman, K. L. Jury, and M. J. Gasson, “Controlled expression and structural organization of a Lactococcus lactis bacteriophage lysin encoded by two overlapping genes,” *Appl Environ Microbiol*, vol. 60, no. 9, pp. 3063–3073, Sep. 1994, doi: 10.1128/aem.60.9.3063-3073.1994.

[84] I. Brierley, “Macrolide-Induced Ribosomal Frameshifting: A New Route to Antibiotic Resistance,” *Molecular Cell*, vol. 52, no. 5, pp. 613–615, Dec. 2013, doi: 10.1016/j.molcel.2013.11.017.

[85] A. L. Blinkowa and J. R. Walker, “Programmed ribosomal frameshifting generates the Escherichia coli DNA polymerase III gamma subunit from within the tau subunit reading frame.,” *Nucleic Acids Res*, vol. 18, no. 7, pp. 1725–1729, Apr. 1990.

[86] N. Mejlhede *et al.*, “−1 Frameshifting at a CGA AAG Hexanucleotide Site Is Required for Transposition of Insertion Sequence IS1222,” *J Bacteriol*, vol. 186, no. 10, pp. 3274–3277, May 2004, doi: 10.1128/JB.186.10.3274-3277.2004.

[87] G. J. Sharples and R. G. Lloyd, “Resolution of Holliday junctions in Escherichia coli: identification of the ruvC gene product as a 19-kilodalton protein,” *J Bacteriol*, vol. 173, no. 23, pp. 7711–7715, Dec. 1991, doi: 10.1128/jb.173.23.7711-7715.1991.

[88] P. García, I. Rodríguez, and J. E. Suárez, “A −1 Ribosomal Frameshift in the Transcript That Encodes the Major Head Protein of Bacteriophage A2 Mediates Biosynthesis of a Second Essential Component of the Capsid,” *J Bacteriol*, vol. 186, no. 6, pp. 1714–1719, Mar. 2004, doi: 10.1128/JB.186.6.1714-1719.2004.

[89] H. Jiang *et al.*, “Orsay virus utilizes ribosomal frameshifting to express a novel protein that is incorporated into virions,” *Virology*, vol. 450, pp. 213–221, Feb. 2014, doi: 10.1016/j.virol.2013.12.016.

[90] D. Jacobs-Sera *et al.*, “Genomic diversity of bacteriophages infecting Microbacterium spp,” *PLoS One*, vol. 15, no. 6, p. e0234636, 2020, doi: 10.1371/journal.pone.0234636.

[91] M. Vladimirov, V. Gautam, and A. R. Davidson, “Identification of the tail assembly chaperone genes of T4-Like phages suggests a mechanism other than translational frameshifting for biogenesis of their encoded proteins,” *Virology*, vol. 566, pp. 9–15, Jan. 2022, doi: 10.1016/j.virol.2021.11.003.

[92] J. F. Curran, “Analysis of effects of tRNA:message stability on frameshift frequency at the *Escherichia coli* RF2 programmed frameshift site,” *Nucl Acids Res*, vol. 21, no. 8, pp. 1837–1843, 1993, doi: 10.1093/nar/21.8.1837.

[93] L. Kurian, R. Palanimurugan, D. Gödderz, and R. J. Dohmen, “Polyamine sensing by nascent ornithine decarboxylase antizyme stimulates decoding of its mRNA,” *Nature*, vol. 477, no. 7365, Art. no. 7365, Sep. 2011, doi: 10.1038/nature10393.

[94] S. Matsufuji, T. Matsufuji, N. M. Wills, R. F. Gesteland, and J. F. Atkins, “Reading two bases twice: mammalian antizyme frameshifting in yeast.,” *EMBO J*, vol. 15, no. 6, pp. 1360–1370, Mar. 1996.

[95] W. M. Huang *et al.*, “A persistent untranslated sequence within bacteriophage T4 DNA topoisomerase gene 60,” *Science*, vol. 239, no. 4843, pp. 1005–1012, Feb. 1988, doi: 10.1126/science.2830666.

[96] R. Ketteler, “On programmed ribosomal frameshifting: the alternative proteomes,” *Front Genet*, vol. 3, p. 242, Nov. 2012, doi: 10.3389/fgene.2012.00242.

[97] R. A. Spanjaard and J. van Duin, “Translation of the sequence AGG-AGG yields 50% ribosomal frameshift,” *Proc Natl Acad Sci U S A*, vol. 85, no. 21, pp. 7967–7971, Nov. 1988, doi: 10.1073/pnas.85.21.7967.

[98] D. Mcnulty, B. Claffee, M. Huddleston, M. Porter, K. Cavnar, and J. Kane, “Mistranslational errors associated with the rare arginine codon CGG in Escherichia coli,” *Protein expression and purification*, vol. 27, pp. 365–74, Mar. 2003, doi: 10.1016/S1046-5928(02)00610-1.

[99] O. L. Gurvich, P. V. Baranov, R. F. Gesteland, and J. F. Atkins, “Expression Levels Influence Ribosomal Frameshifting at the Tandem Rare Arginine Codons AGG_AGG and AGA_AGA in Escherichia coli,” *J Bacteriol*, vol. 187, no. 12, pp. 4023–4032, Jun. 2005, doi: 10.1128/JB.187.12.4023-4032.2005.

[100] K. Usdin, “The biological effects of simple tandem repeats: Lessons from the repeat expansion diseases,” *Genome Res.*, vol. 18, no. 7, pp. 1011–1019, Jul. 2008, doi: 10.1101/gr.070409.107.

[101] S. E. Wright *et al.*, “CGG repeats trigger translational frameshifts that generate aggregation-prone chimeric proteins,” *Nucleic Acids Res*, vol. 50, no. 15, pp. 8674–8689, Aug. 2022, doi: 10.1093/nar/gkac626.

[102] T. Ben-Zvi, A. Pushkarev, H. Seri, M. Elgrably-Weiss, K. Papenfort, and S. Altuvia, “mRNA dynamics and alternative conformations adopted under low and high arginine concentrations control polyamine biosynthesis in Salmonella,” *PLoS Genet*, vol. 15, no. 2, p. e1007646, Feb. 2019, doi: 10.1371/journal.pgen.1007646.

[103] H. Suzuki, T. Kunisawa, and J. Otsuka, “Theoretical evaluation of transcriptional pausing effect on the attenuation in trp leader sequence,” *Biophys J*, vol. 49, no. 2, pp. 425–435, Feb. 1986, doi: 10.1016/S0006-3495(86)83652-9.

[104] A. V. Lukashin and M. Borodovsky, “GeneMark.hmm: new solutions for gene finding,” *Nucleic Acids Res*, vol. 26, no. 4, pp. 1107–1115, Feb. 1998, doi: 10.1093/nar/26.4.1107.

[105] A. Grigoriev, “Analyzing genomes with cumulative skew diagrams,” *Nucleic Acids Res*, vol. 26, no. 10, pp. 2286–2290, May 1998, doi: 10.1093/nar/26.10.2286.

[106] S. Karlin, A. M. Campbell, and J. Mrázek, “Comparative DNA analysis across diverse genomes,” *Annu Rev Genet*, vol. 32, pp. 185–225, 1998, doi: 10.1146/annurev.genet.32.1.185.

[107] G. J. Stewart, C. A. Carlson, and J. L. Ingraham, “Evidence for an active role of donor cells in natural transformation of Pseudomonas stutzeri,” *J Bacteriol*, vol. 156, no. 1, pp. 30–35, Oct. 1983, doi: 10.1128/jb.156.1.30-35.1983.

[108] M. Syvanen, “Cross-species gene transfer; implications for a new theory of evolution,” *J Theor Biol*, vol. 112, no. 2, pp. 333–343, Jan. 1985, doi: 10.1016/s0022-5193(85)80291-5.

[109] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.” 2015. Accessed: Sep. 08, 2023. [Online]. Available: http://download.tensorflow.org/paper/whitepaper2015.pdf

[110] C. Truong, L. Oudre, and N. Vayatis, “Selective review of offline change point detection methods,” *Signal Processing*, vol. 167, p. 107299, Feb. 2020, doi: 10.1016/j.sigpro.2019.107299.

[111] C. Vanhée-Brossollet, H. Thoreau, N. Serpente, L. D’Auriol, J.-P. Lévy, and C. Vaquero, “A natural antisense RNA derived from the HIV-1 env gene encodes a protein which is recognized by circulating antibodies of HIV+ individuals,” *Virology*, vol. 206, no. 1, pp. 196–202, Jan. 1995, doi: 10.1016/S0042-6822(95)80034-4.

[112] R. Vaiskunaite, A. Miller, L. Davenport, and G. Mosig, “Two New Early Bacteriophage T4 Genes, repEA and repEB, That Are Important for DNA Replication Initiated from Origin E,” *J Bacteriol*, vol. 181, no. 22, pp. 7115–7125, Nov. 1999.

[113] M. Behrens, J. Sheikh, and J. P. Nataro, “Regulation of the Overlapping pic/set Locus in Shigella flexneri and Enteroaggregative Escherichia coli,” *Infection and Immunity*, vol. 70, no. 6, pp. 2915–2925, Jun. 2002, doi: 10.1128/iai.70.6.2915-2925.2002.

[114] M. W. Silby and S. B. Levy, “Overlapping protein-encoding genes in Pseudomonas fluorescens Pf0-1,” *PLoS Genet*, vol. 4, no. 6, p. e1000094, Jun. 2008, doi: 10.1371/journal.pgen.1000094.

[115] S. Tunca, C. Barreiro, J.-J. R. Coque, and J. F. Martín, “Two overlapping antiparallel genes encoding the iron regulator DmdR1 and the Adm proteins control siderophore [correction of sedephore] and antibiotic biosynthesis in Streptomyces coelicolor A3(2),” *FEBS J*, vol. 276, no. 17, pp. 4814–4827, Sep. 2009, doi: 10.1111/j.1742-4658.2009.07182.x.

[116] M. Ryoji, K. Hsia, and A. Kaji, “Read-through translation,” *Trends in Biochemical Sciences*, vol. 8, no. 3, pp. 88–90, Mar. 1983, doi: 10.1016/0968-0004(83)90256-6.

[117] N. J. Dimonaco, W. Aubrey, K. Kenobi, A. Clare, and C. J. Creevey, “No one tool to rule them all: prokaryotic gene prediction tool annotations are highly dependent on the organism of study,” *Bioinformatics*, vol. 38, no. 5, pp. 1198–1207, Mar. 2022, doi: 10.1093/bioinformatics/btab827.

[118] N. Yutin *et al.*, “Analysis of metagenome-assembled viral genomes from the human gut reveals diverse putative CrAss-like phages with unique genomic features,” *Nat Commun*, vol. 12, no. 1, p. 1044, Feb. 2021, doi: 10.1038/s41467-021-21350-w.

[119] H. M. Salis, “The ribosome binding site calculator,” *Methods Enzymol*, vol. 498, pp. 19–42, 2011, doi: 10.1016/B978-0-12-385120-8.00002-4.

[120] T. Schurr, E. Nadir, and H. Margalit, “Identification and characterization of E.coli ribosomal binding sites by free energy computation,” *Nucleic Acids Research*, vol. 21, no. 17, pp. 4019–4023, Aug. 1993, doi: 10.1093/nar/21.17.4019.