ABSTRACT

Title of dissertation:	COMPUTATIONAL METHODS TO IMPROVE GENOME ASSEMBLY AND GENE PREDICTION
	David Kelley, Doctor of Philosophy, 2011
Dissertation directed by:	Professor Steven Salzberg Department of Computer Science

DNA sequencing is used to read the nucleotides composing the genetic material that forms individual organisms. As 2^{nd} generation sequencing technologies offering high throughput at a feasible cost have matured, sequencing has permeated nearly all areas of biological research. By a combination of large-scale projects led by consortiums and smaller endeavors led by individual labs, the flood of sequencing data will continue, which should provide major insights into how genomes produce physical characteristics, including disease, and evolve. To realize this potential, computer science is required to develop the bioinformatics pipelines to efficiently and accurately process and analyze the data from large and noisy datasets. Here, I focus on two crucial bioinformatics applications: the assembly of a genome from sequencing reads and protein-coding gene prediction.

In genome assembly, we form large contiguous genomic sequences from the short sequence fragments generated by current machines. Starting from the raw sequences, we developed software called Quake that corrects sequencing errors more accurately than previous programs by using coverage of k-mers and probabilistic modeling of sequencing errors. My experiments show correcting errors with Quake improves genome assembly and leads to the detection of more polymorphisms in re-sequencing studies. For post-assembly analysis, we designed a method to detect a particular type of mis-assembly where the two copies of each chromosome in diploid genomes diverge. We found thousands of examples in each of the chimpanzee, cow, and chicken public genome assemblies that created false segmental duplications.

Shotgun sequencing of environmental DNA (often called metagenomics) has shown tremendous potential to both discover unknown microbes and explore complex environments. We developed software called Scimm that clusters metagenomic sequences based on composition in an unsupervised fashion more accurately than previous approaches. Finally, we extended an approach for predicting protein-coding genes on whole genomes to metagenomic sequences by adding new discriminative features and augmenting the task with taxonomic classification and clustering of the sequences. The program, called Glimmer-MG, predicts genes more accurately than all previous methods. By adding a model for sequencing errors that also allows the program to predict insertions and deletions, accuracy significantly improves on error-prone sequences.

COMPUTATIONAL METHODS TO IMPROVE GENOME ASSEMBLY AND GENE PREDICTION

by

David Kelley

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Acknowledgments

First and foremost, I acknowledge my thesis advisor Steven Salzberg. His patience while I learned how to do research in computational biology and support as I developed and pursued my own research ideas have had a significant positive impact on my career.

Carl Kingsford, Mihai Pop, and Art Delcher also influenced my development as a computational biologist. I am particularly grateful to Dr. Kingsford for encouraging me to join his group meetings focused on networks in computational biology, which has expanded my knowledge of the field.

My officemates for the majority of my graduate career — Michael Schatz, Adam Phillippy, Cole Trapnell, Saket Navlakha, James White, and Ben Langmead — helped make graduate school incredibly enjoyable. Furthermore, they guided me by their experiences and transformed me from a book smart computer scientist to a more practically skilled one. Many other friends and colleagues at CBCB deserve recognition as well.

While pursuing my undergraduate degree at Syracuse University, I reached out for a senior thesis collaboration and stumbled into a wonderful experience working with biology Professors Scott Pitnick and Al Uly. Drs. Pitnick and Uly guided me through my first true research project and convinced me to continue on to graduate school to study biology.

But the most long-standing and important influence on my life and career has come from my family. My parents Robert Kelley and Janice Collins pushed me to be my best and imparted the value of a technical education. They have been loving, encouraging, supportive, and invaluable at the major decision points in my life. My two brothers and extended family have also been a great influence.

And finally, I am grateful to my loving girlfriend Amy Sawin who supports my hard work and brings much needed balance to my life.

Table of Contents

List of Tables vi

List of Figures vii

1 Background 1
- 1.1 Genome assembly 3
- 1.2 Gene prediction 7
- 1.3 2^{nd}-generation sequencing 9
- 1.4 Metagenomics 12
- 1.5 Summary 12

2 Quake: quality-aware detection and correction of sequencing errors 15
- 2.1 Rationale 15
- 2.2 Results and Discussion 20
  - 2.2.1 Accuracy 20
  - 2.2.2 Genome assembly 24
  - 2.2.3 SNP detection 30
  - 2.2.4 Data quality 33
- 2.3 Conclusions 35
- 2.4 Methods 39
  - 2.4.1 Counting *k*-mers 40
  - 2.4.2 Coverage cutoff 42
  - 2.4.3 Localizing errors 44
  - 2.4.4 Sequencing error probability model 46
  - 2.4.5 Correction search 48
- 2.5 List of abbreviations 52
- 2.6 Acknowledgements 52

3 Detection and correction of false segmental duplications caused by genome mis-assembly 53
- 3.1 Background 54
- 3.2 Results and Discussion 57
  - 3.2.1 Genomes 57
  - 3.2.2 Use of the human genome to check duplications 61
  - 3.2.3 Coverage depth 62
  - 3.2.4 Genes affected by erroneous duplications 64
  - 3.2.5 Unplaced contigs 66
  - 3.2.6 SNPs and indels 67
- 3.3 Conclusions 67
- 3.4 Methods 71
  - 3.4.1 Detection of potential haplotype mis-assemblies 72
  - 3.4.2 Analysis of mate pairs 73
  - 3.4.3 Unplaced contigs 78

3.4.4 Haplotype polymorphisms 79
3.5 List of abbreviations 83
3.6 Acknowledgements 84

4 Clustering metagenomic sequences with interpolated Markov models 85
4.1 Background 86
4.2 Methods 89
4.2.1 Interpolated Markov models 90
4.2.2 K-means clustering framework 92
4.2.3 Scimm 95
4.2.4 Initial partitioning 96
4.2.5 Supervised initial partitioning 98
4.3 Results and Discussion 99
4.3.1 Simulated reads 99
4.3.2 FAMeS 105
4.3.3 In vitro-simulated metagenome 109
4.4 Conclusions 111
4.5 Acknowledgements 114

5 Gene prediction with Glimmer for metagenomic sequences augmented by classification and clustering 115
5.1 Introduction 116
5.2 Methods 120
5.2.1 Glimmer 120
5.2.2 Additional features 120
5.2.3 Truncated length model 123
5.2.4 Classification 127
5.2.5 Clustering 129
5.2.6 Sequencing errors 131
5.2.7 Whole genomes 135
5.2.8 Simulated metagenomes 136
5.2.9 Accuracy 138
5.3 Results 139
5.3.1 Whole genomes 139
5.3.2 Simulated metagenomes - perfect reads 140
5.3.3 Simulated metagenomes - error reads 144
5.3.4 Real metagenomes 147
5.4 Conclusion 148
5.5 Acknowledgements 149

6 Conclusion 150

Bibliography 152

List of Tables

2.1 *E. coli* 36 bp 22
2.2 *E. coli* 124 bp 23
2.3 Velvet *E. coli* assembly 27
2.4 SOAPdenovo bee assembly 28
2.5 *E. coli* SNPs 32

3.1 Erroneous duplications 59
3.2 Unplaced haplotype variants 66
3.3 SNP prior comparison 81

4.1 Varying read length 102
4.2 Varying error rate 103

5.1 Accuracy on whole genomes 140
5.2 Accuracy on simulated metagenomes with perfect reads 142
5.3 Accuracy on simulated metagenomes with error read 144

List of Figures

1.1 Shotgun sequencing 2
1.2 Overlap graph 4
1.3 Assembly graph with repeats 6
1.4 Sequencing cost 11

2.1 Alignment difficulty 25
2.2 Adenine error rate 34
2.3 K-mer coverage 43
2.4 Localize errors 45
2.5 Pseudocode 49
2.6 Correction search 50

3.1 Mis-assembled DCC and DOC 58
3.2 Erroneous duplication lengths 60
3.3 Chimpanzee Contig412.192 61
3.4 SCPEP1 65
3.5 Fragment distributions 75

4.1 Markov models 90
4.2 Scimm pipeline 94
4.3 Unsupervised accuracy 101
4.4 PhyScimm accuracy 104
4.5 FAMeS accuracy 107

5.1 Length distributions 122
5.2 Truncated gene scenarios 125
5.3 Glimmer-MG pipeline 132
5.4 Indel errors 133

Chapter 1

Background

Every individual organism has a genome consisting of deoxyribonucleic acid (DNA) in structures called chromosomes that serve as the instructions for forming that individual. DNA provides an elegant template on which natural selection can act and has allowed the evolution of an incredible diversity of organisms. Elucidating the mechanisms by which DNA performs this function is the major goal of genome biology research.

Over the last 20 years, technologies have emerged to read the sequence of nucleotides composing a strand of DNA. The first large-scale application of sequencing used a method developed by Frederick Sanger that is based on replicating the DNA in the presence of altered nucleotides that halt the elongation of the growing strand [1]. After sorting the halted fragments by length, one can read the original sequence from the final altered nucleotides. In a process called *whole-genome shotgun* sequencing, the entire genome is randomly fragmented into smaller pieces, which are then size-selected for sequencing. To obtain longer range information about the genome, one can perform *paired-end* sequencing where a *read* is sequenced from both ends of a DNA fragment. The pair of reads are then referred to as *mates*.

In 1995, the first full genome of a free-living organism was published for the bacterium Haemophilus influenzae Rd [2]. As sequencing technology has evolved, a

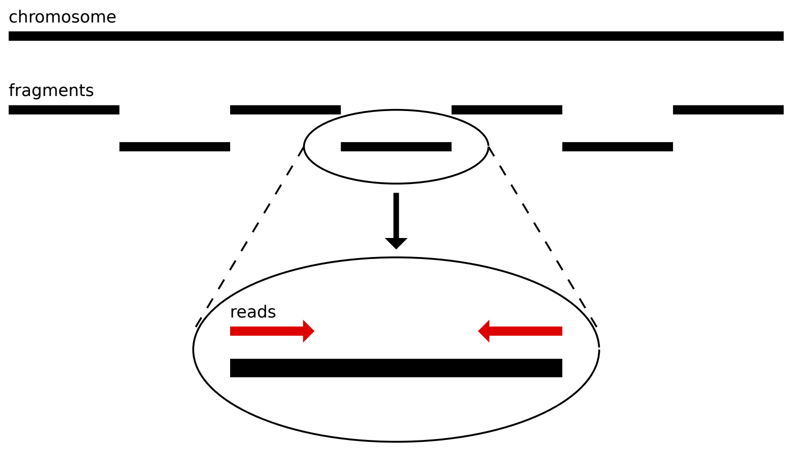


Figure 1.1: In whole-genome shotgun DNA sequencing, the chromosomes are fragmented, e.g. by sonication, and fragments of the desired size are extracted. The ends of these fragments are read by the sequencing machine. Often fragments much larger than the length of a read are chosen so that sequencing both ends of the fragment establishes a distance constraint between the two reads.

large number of full genomes have been sequenced, the most publicized being that of the human genome [3]. These *genome projects* typically consist of the following: After sequencing, one must reconstruct the full chromosomes from the short fragment reads computationally in a process called genome assembly. Next, one would annotate features of the genomic sequence such as the protein-coding genes. Finally, one might compare the organism's genome to previously published genetic sequence to further understanding of how that genome and others have evolved.

The computational requirements of these different aspects of genome analysis spawned the field of bioinformatics. Genome biology datasets, such as whole-genome shotgun sequencing reads, are often large and full of experimental noise. Computer science is needed to effectively manage and analyze the data in order to put it in a tractable form for biologists. For example, as mentioned above, the first step after sequencing a new organism is to assemble the reads to reconstruct the chromosomes. Genome assembly benefits from the application of advances from a subfield of computer science focused on string algorithms. Further, predicting the locations of genes in the reconstructed sequences can be tackled using machine learning algorithms. Bioinformatics methods related to these two instrumental problems — genome assembly and gene prediction — are the focus of this thesis.

1.1 Genome assembly

When sequencing a new organism, our goal is to recreate the entire genome as a string of nucleotides. By repeating the sequencing experiment many times, we make it likely that every part of the genome is covered by multiple reads [4]. The genome assembly problem is to reconstruct large contiguous segments of the chromosomes from the sequencing reads, while avoiding mistakes that create false sequence. A popular and successful framework for solving the assembly problem breaks it up into three steps called overlap, layout, and consensus.

First, to understand the relationships between the reads, we determine where they overlap. Because the sequencing process is inexact, the reads will contain errors where a false nucleotide was substituted for the true one, a false nucleotide was inserted, or a true nucleotide was deleted. Thus, we must allow for mismatches and gaps in the overlap alignments. A naive approach that computed an alignment using dynamic programming [5] for every pair of N reads with length L would require

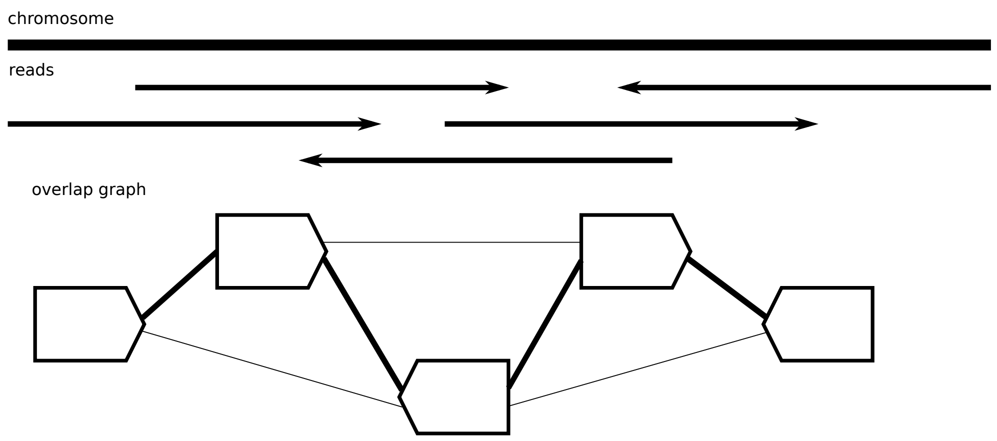


Figure 1.2: Overlaps between sequencing reads can be used to construct a graph where each read is a vertex and each overlap is an edge. Because we must compute overlaps on both ends of the reads, these edges must be bidirectional. The proper assembly is a path that visits all of the vertexes while obeying the bidirectionality constraints.

$O(N^2L^2)$ computation time. A common heuristic first builds a hash table on k-mers (substrings of length k) mapping each individual k-mer to a list of its appearances in the reads. The k-mer size should be chosen so that reads that overlap share a k-mer. Then we can filter the number of alignments computed by focusing only on pairs of reads that share a k-mer.

In the layout stage, the overlaps are used to form a graph where each read is represented by a vertex and overlaps between reads become edges (see Figure 1.2). Note that in a perfect overlap graph, the true assembly is a Hamiltonian path, visiting each read vertex exactly once. In practice, this has little use both because the Hamiltonian path problem is NP-complete [6] and also because overlapping algorithms will struggle to generate a perfect graph because of sequencing errors. Unfortunately, many other combinatorial algorithms one might like to perform on the overlap graph to assemble the reads are also NP-Hard [7]. Instead we must rely on heuristics to assemble most datasets, where the goal is to form contiguous stretches of sequence called *contigs* that are as large as possible without introducing errors. For example, an early approach called Phrap uses a greedy algorithm to merge overlapping reads [8]. Later approaches work to simplify the overlap graph by removing transitive edges that are implied by closer overlaps and merging unambiguously connected vertexes [9, 10]. After forming contigs from the overlap graph, the consensus step uses the reads to make a final base call at each position in the contig.

De Bruijn graphs offer an alternative algorithmic formulation to the overlap and layout stages of the assembly problem that has certain advantages [11]. To build a de Bruijn graph of the reads, one must choose a k-mer size, form a vertex for every k-mer, and draw an edge between two k-mer vertexes when those k-mers are adjacent in one of the reads. Thus, the graph can be built in linear time in the number and lengths of the reads, avoiding the potential quadratic time of an overlap computation step. In theory, the genome can then be assembled by finding a Eulerian path, i.e. one that visits every edge in the graph once, which can be found in linear time in the number of edges. In practice, repeats and errors sufficiently muddle the graph so that rarely is the true genome represented by a unique Eulerian path. Nevertheless, de Bruijn graph-based assemblers that perform substantial additional work to eliminate erroneous paths in the graph have proven successful [12, 13].

When paired-end sequencing has been performed, the assembled contigs can be further arranged into *scaffolds* by incorporating the distance constraints between

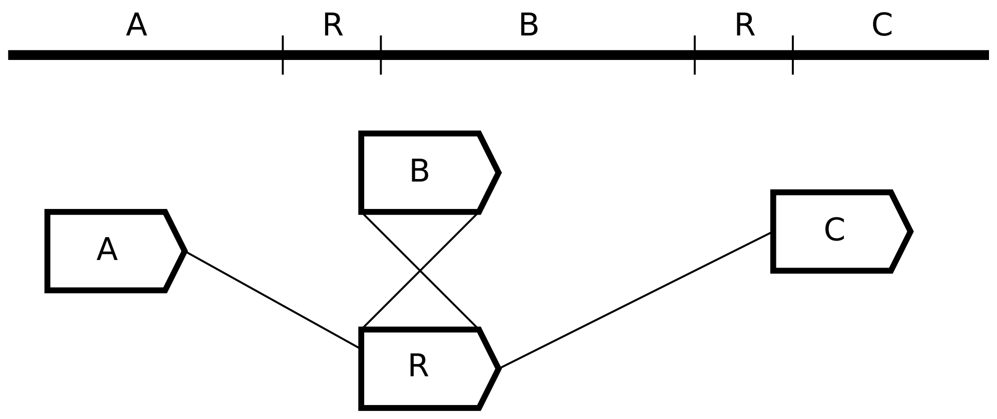


Figure 1.3: The prevalence of repetitive segments of in the genome complicate the assembly problem. Here a repeat R is interleaved with the unique segments A, B, and C. In a scaffolding stage, paired-end read connections between the unique segments should be sufficient to properly layout the genome. In this simple case, the unique Eulerian tour of the graph also describes the true assembly, but the situation is rarely so clear with real genomes and data.

.

reads implied by the original fragment sizes. The relevant computational problems for handling these constraints in the case of errors are NP-hard [14], so again heuristics must be used. Most strategies first try to identify repeat contigs based on their greater read coverage and conflicting mate pair information so that unique contigs can be arranged first [15]. One approach that works well in practice scores each pair of unique contigs as a candidate to be linked based on the closeness and coverage of their mated reads and uses a greedy algorithm to gradually form scaffolds [16].

A completed assembly consists of reads merged into contigs and contigs linked into scaffolds. The larger these are, the more useful the assembly will be for genomics analysis. Larger contigs and scaffolds will generally collect more sequence and more genomic features, such as protein-coding genes. They also allow the genomic context of these features to be analyzed, such as cis-regulatory elements [17]. Finally, multiple genomes can be aligned and compared more accurately with larger sequences. However, the goal of constructing large contigs and scaffolds trades off against the goal of avoiding assembly errors that create false sequence. Because of imperfect data and the use of heuristic-based assembly algorithms, downstream analysis should always consider mis-assemblies [18].

1.2 Gene prediction

Genome assembly describes the genomic sequence, but we also want to understand how it encodes functions. Protein-coding genes are DNA segments that are transcribed to mRNA and translated into proteins. They perform many important functions in the cell, such as giving the cell structure, catalyzing biochemical reactions, and transferring information. The gene prediction problem is to identify the positions of all protein-coding genes, and thus their amino acid sequences, in an unlabeled sequence. Here we focus on prokaryotic gene prediction. Eukaryotic gene prediction has additional complexities related to the splicing out of introns from mRNA transcripts.

Certain properties of genes make this task possible. Every gene begins with one of three start codons and ends with one of three stop codons. Because the gene's DNA sequence ultimately translates to amino acids via a triplet code, coding sequence has compositional constraints that differentiate it from noncoding sequence. Markov chain models capture this composition well by modeling the distribution of a nucleotide in the sequence conditionally on a previous window (e.g. the previous nucleotides in a codon). To begin translation, the ribosome binds upstream of the gene's start codon. Signatures of this ribosomal binding site (RBS), such as the Shine-Dalgarno sequence [19], are also useful for detecting genes.

By modeling these gene features in supervised machine learning algorithms, we can discriminate between coding and noncoding sequence [20]. One popular approach is based on open reading frames (ORFs), or stretches of sequence without a stop codon. First, we use a training set of known genes to learn models for each gene feature. Given a new sequence, we extract every ORF pair of start and stop codons and score their coding potential using the ratio of the likelihood that the sequence came from our coding versus noncoding models. Finally, we find the set of genes with the greatest score satisyfing a maximum overlap constraint (e.g. 50 bp) [21]. Alternatively, we can segment the genome using structured prediction algorithms such as hidden Markov models containing coding and noncoding states [22]. Current methods are very accurate, predicting genes with 99% sensitivity [23].

An accurate set of gene predictions is imperative to understanding the organism and can be used in the following ways. Comparing the gene sequences to a database of known proteins can place those genes in an evolutionary context. It can also assign functional annotations if orthologues have been functionally characterized [24]. Similarly, comparing the genes to each other to identify paralogues may suggest how the genome evolved. Finally, examining the organization of the genes, e.g. as operons [25], helps us understand how the genome encodes the organism's expression patterns.

1.3 2^{nd}-generation sequencing

Recently, a class of 2^{nd}-generation sequencing technologies have emerged that offer far greater throughput at a cost per nucleotide that is orders of magnitude less than Sanger sequencing [26]. 454 Life Sciences [27] and Illumina [28] offer two of the most popular technologies and share a number of features. Both grow many strands of DNA in parallel and detect the incorporation of altered nucleotides by the emission of light [29, 30]. Both technologies deliver shorter reads than Sanger sequencing, currently ~500 bp for 454 and ~150 bp for Illumina, making some applications such as genome assembly more difficult. On the other hand, one can obtain much greater coverage of the genome, which is generally very useful.

For whole-genome shotgun, 2^{nd}-generation sequencing has been fully embraced as the decreased cost per nucleotide has lowered the bar for sequencing a new organism. However, the characteristics of each technology have needed to be carefully handled. For example, mistakes in 454 reads tend to be mis-calls of the number of nucleotides in a long homopolymer run, thus introducing insertions or deletions into the read [30]. Alternatively, Illumina reads rarely have insertions or deletions, but will frequently have substitution errors, particularly towards the end of the read [29]. Bioinformatics methods need to be adjusted to best make use of this new type of data. For example, de Bruijn graph-based assemblers have proven useful because they avoid computing overlaps in the larger sets of short reads.

Nevertheless, the decreased cost of sequencing (see Figure 1.4) has truly put it in the hands of the masses, which is revolutionizing the way biological research is performed. Large scale consortiums, like that for the human genome, are generally no longer needed to sequence individual organisms, but have instead focused on sequencing lots of genomes. For example, recently launched projects include the 1000 Genomes Project to discover all common variation in the human genome [31] and the Cancer Genome Atlas to sequence the tumors and germlines of cancer patients to uncover the genetic basis of the disease [32]. The Genome 10K Project plans to sequence a huge catalogue of vertebrates [33]. Furthermore, as a wider variety of sequencing machines become available, sequencing will become a feasible tool for smaller individual labs as well. 2011 is expected to bring the release of a "compact and economical instrument" called the MiSeq from Illumina [34] and the "simpler, faster, more cost effective and scalable" PGM Sequencer from Ion Torrent [35].

In addition to whole-genome shotgun, high-throughput sequencing has been co-opted for other genome research applications. *RNA-Seq* attempts to capture and sequence an individual's mRNA in order to describe the transcriptome and measure the expression of individual transcripts [37]. *ChIP-Seq* aims to discover DNA sequences in the genome bound by a certain protein by crosslinking the protein to the DNA, pulling down the protein with an antibody, and sequencing the DNA [38]. These experiments and many others are quickly becoming standard tools for interrogating genomes in different ways. And each has introduced its own set of computational challenges in order to efficiently extract maximal biological information from the experiment.

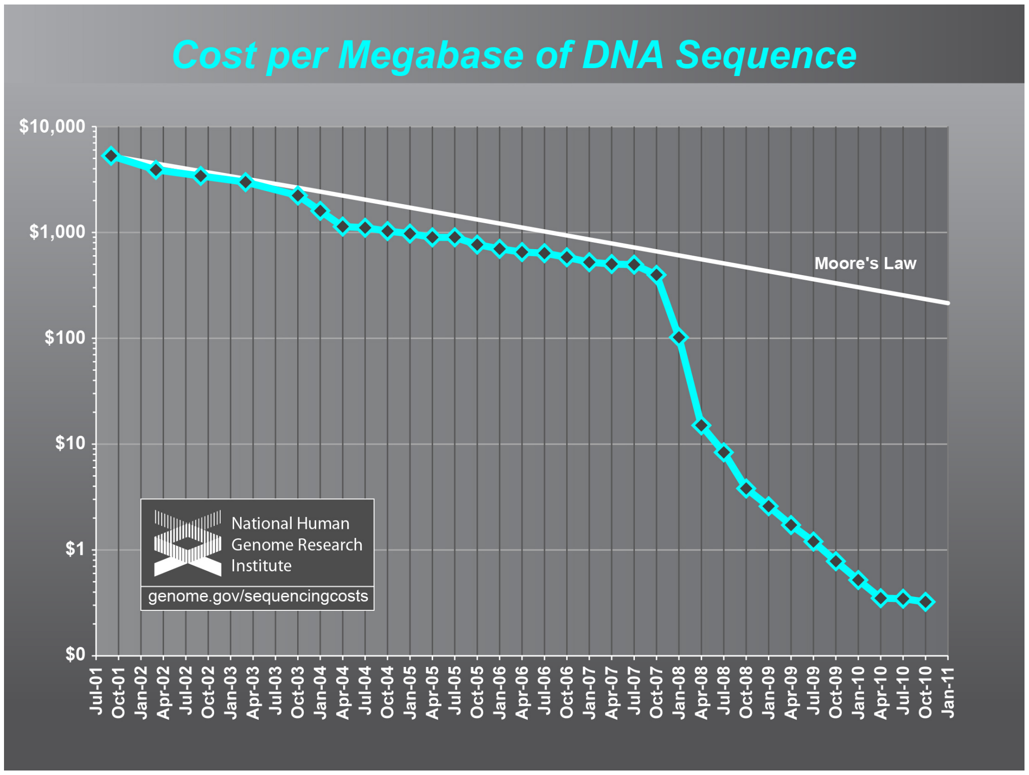


Figure 1.4: Though the cost of sequencing scaled with Moore's Law between 2001 and 2007, roughly decreasing by a factor of two every two years, it has since accelerated with the advent of 2^{nd}-generation technologies. To effectively process and analyze the upcoming flood of sequencing data, bioinformatics innovation will be imperative. [Figure produced by NHGRI [36]]

1.4 Metagenomics

Another novel application of sequencing is *metagenomics*, in which genetic material is sampled from free-living microbial communities for analysis [39]. Individual experiments generally seek to discover what organisms exist in the sample and in what proportions. Then sampling across space and time can be used to compare communities and study their dynamics.

Early experiments focused on the targeted sequencing of well-studied marker genes (such as 16s rRNA) that could then be easily compared to previously sequenced versions of those genes [40]. As the cost has decreased, shotgun sequencing of environmental DNA has become a more attractive option for many purposes. Initial applications highlighted the fantastic potential, such as the ability to assemble substantial portions of unculturable organisms [41] and discover an enormous number of new genes [42, 43]. However, they also made clear how difficult computational analysis of environmental shotgun sequencing reads can be. Frequently, an assembly of the reads will be highly fragmented [44], and the origin of most reads will be unidentifiable via BLAST search [43]. Thus, innovative bioinformatics will be crucial to make the most of metagenomics.

1.5 Summary

The development of sequencing and other high-throughput experiments have poised the field of biology for an incredibly exciting period of research. The data generated over the next few years will give researchers the ability to answer major questions regarding how genomes produce physical characteristics and how they have evolved. Instrumental to these analyses will be computer science research to produce the software and algorithms needed to accurately and efficiently process the experimental data. In this thesis, I introduce a series of bioinformatics methods towards this goal.

When trying to assemble 2^{nd}-generation sequencing reads, processing the raw data to ensure that the algorithms' assumptions are satisfied has become arguably as important as the quality of the algorithms used. For example, preprocessing of the reads to correct sequencing errors has become standard for de Bruijn graph-based assemblers [13, 45, 46]. Chapter 2 describes a method to detect and correct sequencing errors in high-throughput datasets based on coverage of k-mers that improves accuracy by probabilistic modeling of the errors. After the assembly has finished, work still remains to validate the result and confirm that the input data fit the assumptions made and the assembler's heuristics did not cause obvious problems. For example, in a heterozygous region of the genome where the two chromosomes have many differences, the assembler may construct two separate contigs covering the region and then place them nearby, creating the illusion of a segmental duplication. Chapter 3 describes a method to detect this mis-assembly and its application to the chimpanzee, chicken, cow, and dog genomes where thousands of mis-assemblies were found and analyzed.

As sequencing is applied to new areas, bioinformatics methods must be redeveloped or designed from scratch to support the characteristics of the data and the questions biologists are interested in. Metagenomic sequencing has fantastic potential as a tool for studying complex microbial environments, but needs computational support to realize that potential. A natural first question that a researcher would want to ask about their dataset is what organisms are in the mixture and in what abundances. Chapter 4 describes an unsupervised sequence clustering method able to effectively bring together sequences from related organisms and separate those that differ. Similarly to whole-genome projects, researchers are also interested in finding the protein-coding genes on metagenomic sequences. Chapter 5 describes the challenges associated with metagenomics gene prediction and a method that addresses each of them to produce more accurate predictions than all other programs.

Chapter 2

Quake: quality-aware detection and correction of sequencing errors

All sequencing technologies generate imperfect data such that the reads obtained will inevitably contain sequencing errors. These errors complicate downstream bioinformatics tasks such as genome assembly and mapping reads to a reference genome. However, given sufficient coverage of the genome, the errors can be detected and often even corrected back to the true sequence.

This chapter describes work done with Michael Schatz and Steven Salzberg to develop an improved method for error correction in high coverage datasets generated by the Illumina sequencing technology. Our method, called Quake, detects errors by looking for k-mers that appear very few times in the reads. Quake then corrects these errors using probabilistic modeling of sequencing errors and an efficient search through the space of correction sets in order of decreasing likelihood. Quake achieves greater correction accuracy than previous methods on simulated reads and improves genome assembly and variant detection on real data. The following manuscript was published in November 2010 in Genome Biology [47].

2.1 Rationale

Massively parallel DNA sequencing has become a prominent tool in biological research [26, 48]. The high-throughput and low cost of 2^{nd}-generation sequencing technologies has allowed researchers to address an ever-larger set of biological and biomedical problems. For example, the 1000 Genomes Project is using sequencing to discover all common variations in the human genome [31]. The Genome 10K Project plans to sequence and assemble the genomes of 10,000 vertebrate species [33]. Sequencing is now being applied to a wide variety of tumor samples in an effort to identify mutations associated with cancer [32, 49]. Common to all of these projects is the paramount need to accurately sequence the sample DNA.

DNA sequence reads from Illumina sequencers, one of the most successful of the 2^{nd}-generation technologies, range from 35-125 bp in length. Although sequence fidelity is high, the primary errors are substitution errors, at rates of 0.5-2.5% (as we show in our experiments), with errors rising in frequency at the 3' ends of reads. Sequencing errors complicate analysis, which normally requires that reads be aligned to each other (for genome assembly) or to a reference genome (for detection of mutations). Mistakes during the overlap computation in genome assembly are costly: missed overlaps may leave gaps in the assembly, while false overlaps may create ambiguous paths or improperly connect remote regions of the genome [50]. In genome re-sequencing projects, reads are aligned to a reference genome, usually allowing for a fixed number of mismatches due to either SNPs or sequencing errors [51]. In most cases, the reference genome and the genome being newly sequenced will differ, sometimes substantially. Variable regions are more difficult to align because mismatches from both polymorphisms and sequencing errors occur, but if errors can be eliminated, more reads will align and the sensitivity for variant detection will improve.

Fortunately, the low cost of 2^{nd}-generation sequencing makes it possible to obtain highly redundant coverage of a genome, which can be used to correct sequencing errors in the reads before assembly or alignment. Various methods have been proposed to use this redundancy for error correction; for example, the EULER assembler [11] counts the number of appearances of each oligonucleotide of size k (hereafter referred to as k-mers) in the reads. For sufficiently large k, almost all single-base errors alter k-mers overlapping the error to versions that do not exist in the genome. Therefore, k-mers with low coverage, particularly those occurring just once or twice, usually represent sequencing errors. For the purpose of our discussion, we will refer to high coverage k-mers as *trusted*, because they are highly likely to occur in the genome, and low coverage k-mers as *untrusted*. Based on this principle, we can identify reads containing untrusted k-mers and either correct them so that all k-mers are trusted or simply discard them. The latest instance of EULER determines a coverage cutoff to separate low and high coverage k-mers using a mixture model of Poisson (low) and Gaussian (high) distributions, and corrects reads with low coverage k-mers by making nucleotide edits to the read that reduce the number of low coverage k-mers until all k-mers in the read have high coverage [45]. A number of related methods have been proposed to perform this error correction step, all guided by the goal of finding the minimum number of single base edits (edit distance) to the read that make all k-mers trusted [13, 46, 52, 53].

In addition, a few alternative approaches to error correction should be mentioned. Past methods intended for Sanger sequencing involve multiple sequence alignments of reads rendering them infeasible for short read datasets [16, 54, 55].

More recently, a generalized suffix tree of the reads was shown to be an effective data structure for detecting and correcting errors in short reads [56, 57]. De Bruijn graph-based short read assemblers [12, 13, 45, 46, 58] perform substantial error correction of reads in the de Bruijn graph. For example, short dead end paths are indicative of a sequencing error at the end of a read and can be removed, and "bubbles" where a low coverage path briefly diverges from and then reconnects to high coverage nodes are indicative of sequencing errors at the middle of a read and can be merged. Finally, a number of methods have been proposed to cluster reads and implicitly correct sequencing errors in data where the targets vary in abundance such as sequencing of small RNAs or 16s rRNA [59–62].

Although methods that search for the correct read based on minimizing edit distance will mostly make the proper corrections, edit distance is an incomplete measure of relatedness. First, each position in a sequencing read is assigned a quality value, which defines the probability that the basecall represents the true base. Though questions have been raised about the degree to which quality values exactly define the probability of error [63], newer methods for assigning them to base calls demonstrate substantial improvements [64–68], and for our purpose of error correction, the quality values can be useful even if they only rank one base as more likely to be an error as another. We should prefer to edit a read at these lower quality bases where errors are more likely, but edit distance treats all bases the same regardless of quality. Furthermore, specifics of the Illumina technology cause certain miscalls to be more likely than others. For example, bases are called by analysis of flourescent output from base-incorporating chemical reactions, and

A and C share a red detection laser while G and T share a green detection laser. Thus, A and C are more likely to be mistaken for each other than for G or T [63]. Edit distance treats all error substitutions as equally likely.

In this chapter, we introduce a new algorithm called Quake to correct substitution errors in sets of DNA sequencing reads produced as part of >15x coverage sequencing projects, which has become commonplace thanks to the efficiency of 2^{nd}-generation sequencing technologies. Quake uses the k-mer coverage framework, but incorporates quality values and rates of specific miscalls computed from each sequencing project. In addition, Quake incorporates a new method to choose an appropriate coverage cutoff between trusted k-mers (those that are truly part of the genome) and erroneous k-mers based on weighting k-mer counts in the reads using the quality values assigned to each base. On simulated data using quality values from real reads, Quake is more accurate than previous methods, especially with relatively long Illumina reads. Correcting reads guided by edit distance alone, without the use of quality values, results in many more improperly corrected reads. These reads are then chimeric, containing sequence from two distinct areas of the genome, which can be a major problem for assembly software.

Finally, we explore the impact of error correction with Quake on two important bioinformatics applications- *de novo* assembly and detection of variations with respect to a reference genome. Even a sophisticated assembler such as Velvet [12], which performs its own error correction using the assembly graph, benefits from pre-processing the reads with Quake. SOAPdenovo [46], a parallel assembler capable of assembling mammalian-size datasets, also produces better assemblies after error correction. For variant detection, correcting errors before mapping reads to a reference genome results in more reads aligned to SNP locations and more SNPs discovered. Note that Quake and other correction methods that rely on coverage of k-mers are inappropriate for applications where low coverage does not necessary implicate a sequencing error such as metagenomics, RNA-Seq, and ChIP-Seq.

Quake is freely available as open source software from our website [69] under the Perl Artistic License [70].

2.2 Results and Discussion

2.2.1 Accuracy

The two goals of error correction are to cleanly separate reads with errors from reads without errors and to properly correct the reads with errors. To assess Quake's ability to accurately complete these tasks, we simulated sequencing reads with errors from finished genomes (using an approach comparable to the "Maq simulate" program [71]) and compared Quake's corrections to the true reference. For each dataset, we categorized reads and their corrections into four outcomes. As positive outcomes, we counted the number of reads that were properly corrected to their original state or trimmed such that no errors remained. As negative outcomes, we counted the number of reads mis-corrected producing a false sequence or left uncorrected even though they contained errors. Reads were simulated by chooosing a position in the reference genome, using the quality values from an actual Illumina sequencing read, and changing the nucleotides according to the probabilities defined by those quality values. Dohm et al. measured the bias in Illumina specific nucleotide to nucleotide miscall rates by sequencing reads from *Helicobacter acinonychis* and *Beta vulgaris*, aligning them to high quality reference genomes, and counting the number of each type of mismatch in the alignments [63]. At simulated errors, we changed the nucleotide according to these frequencies.

To compare Quake's accuracy to that of previous error correction programs, we corrected the reads using EULER [45], Shrec [56], and SOAPdenovo [46] on a 4 core 2.4 GHz AMD Opteron machine. Quake and the other k-mer based correction tools used $k = 15$. SOAPdenovo's error correction module does not contain a method to choose the cutoff between trusted and untrusted k-mers, so we tried a few appropriate values and report the best results. We similarly tried multiple values for Shrec's strictness parameter that is used to help differentiate true and error reads via coverage. These are very sensitive parameters, and leaving them to the user is a critical limitation of these programs. Alternatively, EULER and Quake determine their parameters automatically using the data.

Table 2.1 displays the average of the accuracy statistics after 5 iterations of simulated 36 bp reads to 40x coverage (5.5M reads) from *E. coli 536* [GenBank:NC_008253]. Quality value templates were taken from the sequencing of *E. coli* K-12 substrain MG1655 [SRA:SRX000429]. The datasets contained an average of 1.17M reads with errors. Of the reads that Quake tried to correct, 99.83% were corrected accurately to the true sequence. Quake properly corrected 88.3% (90.5% including trims) of error reads, which was 6.9% more reads than the second best program SOAPdenovo, made 2.3x fewer mis-corrections than SOAPdenovo, and al-

	Corrections	Trim corrections	Mis-corrections	Errors kept	Time (min)
Quake	1035709.4	26337.0	1744.0	5537.0	14.2
SOAPdenovo	969666.4	120529.0	3912.8	9288.4	12.4
Shrec	964431.8	0.0	165422.0	41733.6	87.6

Table 2.1: Simulated *E. coli* 36 bp reads at 40x coverage averaged over five runs. For each method, we counted the number of reads that were properly corrected to their original state (Corrections), trimmed such that no errors remained (Trim corrections), mis-corrected to false sequence (Mis-corrections), and contained errors but were kept in the set (Errors kept). Quake corrects more reads while mis-correcting fewer reads and keeping fewer reads with errors than all programs.

lowed 1.8x fewer reads with errors. The 5265.4 error reads that Quake keeps have errors that only affect a few k-mers (at the end of the read), and these k-mers happen to exist elsewhere in the genome. We could not successfully run EULER on these short reads.

We performed the same test using 5 iterations on 40x coverage (1.6M reads) of 124 bp reads from *E. coli 536*. Most of these reads had very low quality suffixes expected to contain many errors. Quake handled these reads seamlessly, but the other programs produced very poor results. Thus, we first trimmed every read r to the length

$$l = \arg\max_x \sum_{i=x}^{|r|} (t - q_i) \tag{2.1}$$

By setting $t = 3$, we mainly trim nucleotides with quality value 2 off the ends of the reads, but will trim past a higher quality base call if there are a sufficient number of nucleotides with quality $\leq$2 preceding it. On this data (where full results are displayed in Table 2.2), Quake is 99.9% accurate on reads that it tries to correct.

	Corrections	Trim corrections	Mis-corrections	Errors kept	Time (min)
Quake	283769.4	6581.2	243.0	393.6	11.8
SOAPdenovo	276770.4	2942.6	7019.4	5490.2	16.9
Shrec	165942.7	0.0	33140.3	96626.7	97.1
EULER	228316.4	16577.4	3763.0	414.8	6.9

Table 2.2: Simulated *E. coli* 124 bp reads at 40x coverage averaged over five runs. Column descriptions are the same as Table 1. Quake corrects more reads while mis-correcting far fewer reads and keeping fewer reads with errors than all programs.

Of the 297K error reads, Quake corrected 95.6% (97.9% including trims), 2.5% more than SOAPdenovo, the second most effective program. However, SOAPdenovo makes many more mistakes on the longer reads by mis-correcting 28.9x more reads and keeping 11.9x more reads with errors in the set. Shrec and EULER correct far fewer reads and mis-correct more reads than Quake.

To demonstrate Quake's ability to scale to larger genomes, we simulated 325 million 124 bp reads from the 249 Mbp human chromosome 1 (version hg19), which provided 34x coverage after trimming. Due to the larger size of the sequencing target, we counted and corrected 18-mers in the reads. Of the 15.23M reads containing errors, Quake corrected 12.83M (84.2%) and trimmed to a correct prefix another 0.82M (5.4%). Because we could not successfully run SOAPdenovo using 18-mers, we corrected using 17-mers, a reasonable choice given that the authors of that software chose to correct reads using 17-mers for the entire human genome [46]. Quake corrected 11% more reads than SOAPdenovo, reduced mis-corrections by 64%, and kept 15% fewer error reads. EULER produced very poor correction results, e.g. correcting less than half as many reads as Quake with more mis-corrections and error reads kept. On a dataset this large, Shrec required more memory than our largest computer (256 GB).

Relative to the 124 bp simulated reads from *E. coli*, Quake's attempted corrections were accurate at a lower rate (99.02%) and Quake kept more error reads in the dataset (1.11M, 7.27%). This is caused by the fact that the human genome contains far more repetitive elements than *E. coli*, such as the LINE and SINE retrotransposon families [72]. The more repetitive the genome is, the greater the chance is that a sequencing error will merely change one trusted k-mer to another trusted k-mer, hiding the error. To quantify this property of the two genomes, we computed the percentage of all possible single base mutations to k-mers in each genome which create k-mers that also exist in the genome. In *E. coli* 536, this is true for 2.25% of 15-mer mutations, and in chromosome 1 of the human genome, it is true for 13.8% of 18-mer mutations. Increasing the k-mer size does little to alleviate the problem as still 11.1% of 19-mer mutations are problematic. Nevertheless, allowing a small percentage of error reads may not be terribly problematic for most applications. For example, genome assemblers will notice the lower coverage on the paths created by these reads and clean them out of the assembly graph.

2.2.2 Genome assembly

In *de novo* genome assembly, the goal is to build contiguous and unambiguous sequences called contigs from overlapping reads. The traditional formulation of the

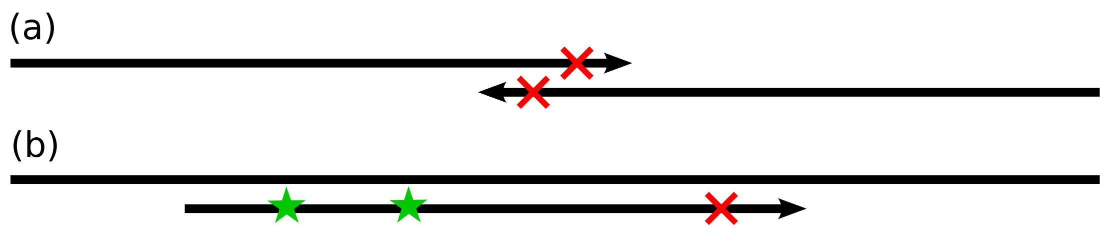


Figure 2.1: Detecting alignments of short reads is more difficult in the presence of sequencing errors (represented as X's). (a) In the case of genome assembly, we may miss short overlaps between reads containing sequencing errors, particularly because the errors tend to occur at the ends of the reads. (b) To find variations between the sequenced genome and a reference genome, we typically first map the reads to the reference. However, reads containing variants (represented as stars) and sequencing errors will have too many mismatches and not align to their true genomic location.

assembly problem involves first finding all overlaps between reads [9], taking care to find all true overlaps between reads sequenced from the same genome location and avoid false overlaps between reads sequenced from remote regions [50]. Because of sequencing errors, we must allow mismatches in the overlap alignments to find all true overlaps, but we cannot allow too many or false overlaps will be found and fragment the assembly. With short reads, we must allow a short minimum overlap length, but in the presence of sequencing errors, particularly when these errors tend to occur at the ends of the reads, we may frequently overlook true overlaps (see Figure 2.1). A de Bruijn graph formulation of the assembly problem has become very popular for short reads [12, 13, 45, 46], but is very sensitive to sequencing errors. A substantial portion of the work performed by these programs goes towards recognizing and correcting errors in the graph.

Having established the accuracy of Quake for error correction on simulated data, we measured the impact of Quake on genome assembly by assembling the reads before and after error correction. One assembly is better than another if it is more connected and more accurately represents the sequenced genome. To measure connectedness, we counted the number of contigs and scaffolds in the assembly larger than 50 bp as well as the N50 and N90 for each, which is the contig/scaffold size for which 50% (90%) of the genome is contained in contigs/scaffolds of equal or larger size. Fewer contigs/scaffolds and larger N50 and N90 values signify that the reads have been more effectively merged into large genomic sequences. In addition, we counted the number of reads included in the assembly because greater coverage generally leads to better accuracy in consensus calling. When a reference genome was available, we used it to validate the correctness of the assembly. We aligned all scaffolds to the reference using MUMmer [73] and considered scaffolds that did not align for their entire length (ignoring 35 bp on each end) at >95% identity to be mis-assembled. We also counted the number of single base differences between the reference and otherwise properly assembled scaffolds. Finally, we computed the percentage of reference nucleotides covered by some aligning scaffold.

Velvet is a widely used de Bruijn graph-based assembler that performs error correction by identifying graph motifs that signify sequencing errors [12], but does not use a stand-alone error correction module like EULER [45] or SOAPdenovo [46]. Thus, we hypothesized that Quake would help Velvet produce better assemblies. To test this hypothesis, we corrected and assembled 152x (20.8M reads) coverage of 36 bp reads from *E. coli* K12 substrain MG1655 [SRA:SRX000429]. We used Velvet's option for automatic computation of expected coverage and chose the de Bruijn

	Contigs	N50	N90	Scaffolds	N50	N90	Breaks	Miscalls	Cov
Uncorrected	398	94,827	17,503	380	95,365	23,869	5	456	0.9990
Corrected	345	94,831	25,757	332	95,369	26,561	4	315	0.9992

Table 2.3: Velvet assemblies of *E. coli* 36 bp paired end reads at 152x coverage. After correcting the reads, more reads are included in the assembly into fewer contigs and scaffolds. N50 and N90 values were computed using the genome size 4,639,675 bp. The N50 value was similar for both assemblies, but N90 grew signficantly with corrected reads. Correcting the reads also improved the correctness of the assembly producing fewer mis-assembled scaffolds (Breaks) and miscalled bases (Miscalls) and covering a greater percentage of the reference genome (Cov).

graph k-mer size that resulted in the best assembly based on the connectedness and correctness statistics discussed above.

Table 2.3 displays the assembly statistics for *E. coli* with Velvet. Quake corrected 2.44M (11.7%) and removed 0.57M (2.8%) reads from the dataset. After correction, 0.75M (3.8%) more reads were included in the assembly, which contained 13% fewer contigs and 13% fewer scaffolds. Though this significant increase in connectedness of the assembly does not manifest in the N50 values, which are similar for both assemblies, the contig N90 increases by 47% and the scaffold N90 increases by 11%. With respect to correctness, the corrected read assembly contained one fewer mis-assembled scaffold and 31% fewer mis-called bases, and still covered slightly more of the reference genome. This improvement was consistent in experiments holding out reads for lesser coverage of the genome (data not shown). As the coverage decreases, the distributions of error and true k-mers blend together and the choice of cutoff must carefully balance making corrections and removing useful reads from low coverage regions. On this dataset, the minimum coverage at which the assembly improved after correction using Quake was 16x.

We also measured Quake's impact on a larger assembly with longer reads by assembling 353.7M Illumina reads, all of them 124 bp in length, from the alfalfa leafcutting bee *Megachile rotundata*, with an estimated genome size of 300 Mbp. (Contact the corresponding author for details on data access.) Assembly was performed with SOAPdenovo [46] using a de Brujin graph k-mer size of 31 and the "-R" option to resolve small repeats. Assembly of the raw uncorrected reads was quite poor because of the very low quality suffixes of many of the 124 bp reads. Thus, we compare assembly of quality trimmed reads (performed as described above), reads corrected using Quake, and trimmed reads corrected with SOAPdenovo's own error correction module. Quake and SOAPdenovo corrected using 18-mers and a coverage cutoff of 1.0.

Correcting errors in the reads had a significant affect on the quality of the assembly as seen in Table 2.4. In the Quake assembly, >123K fewer contigs were returned as contig N50 grew by 71% and contig N90 more than doubled compared

Table 2.4 *(following page)*: SOAPdenovo assemblies of *Megachile rotundata* 124 bp paired end reads. We trimmed the reads before correcting with SOAPdenovo, which greatly improved its performance on our experiments with simulated data. The "Trimmed only" column includes reads trimmed before and during SOAPdenovo correction. Quake trims reads automatically during correction. Correcting the reads reduces the number of contigs and scaffolds, increases the contig sizes, and allows the assembler to include more reads. Quake corrects more reads than SOAPdenovo which results in a slightly better assembly.

Assembly	Trimmed only	Corrected	Removed	Contigs	N50	N90	Scaffolds	N50	N90	Reads
Uncorrected	146.0M	-	12.9M	312,414	2,383	198	90,201	37,138	9,960	167.3M
Corrected SOAPdenovo	134.4M	15.7M	15.6M	188,480	4,051	515	36,525	36,525	9,162	164.8M
Corrected Quake	146.9M	16.5M	13.0M	189,621	4,076	514	37,279	37,014	9,255	167.3M

to the standard approach of only trimming the reads before assembly. Similarly to the simulated reads, Quake is able to correct more reads than SOAPdenovo, which leads to 1.5% more reads included in the assembly than SOAPdenovo and slightly more than the assembly of uncorrected reads. Improvements to the connectedness statistics compared to SOAPdenovo were modest. Surprisingly, although nearly 2.5x fewer scaffolds were returned after error correction with Quake, scaffold N50 remained virtually the same and N90 slightly decreased. We investigated a few possible explanations for this with inconclusive results; e.g. scaffold sizes did not improve substantially after adding back mate pairs excluded due to uncorrectable errors. Because N50 and N90 can be somewhat volatile and the scaffolds in the *E. coli* assembly above did improve after error correction, this is potentially an artifact of this particular dataset, i.e. the library sizes used with respect to the repeat structure of the genome.

2.2.3 SNP detection

A second application of short reads that benefits from error correction is detection of variations, such as single nucleotide polymorphisms (SNPs). In such experiments, the genome from which the reads are sequenced differs from a reference genome to which the reads are compared. The first step is to align the reads to the reference genome using specialized methods [51] that will only allow a few mismatches between the read and reference, such as up to two mismatches in a recent study [74]. A read containing a SNP will start with one mismatch already, and any additional differences from the reference due to sequencing errors will make alignment difficult (see Figure 2.1). Furthermore, the distribution of SNPs in a genome is not uniform and clusters of SNPs tend to appear [75]. Reads from such regions may contain multiple SNPs. If these reads contain any sequencing errors, they will not align causing the highly polymorphic region to be overlooked.

To explore the benefit that error correction with Quake may have on SNP detection, we randomly sampled reads representing 35x from the *E. coli* K12 reads used above. To call SNPs, we aligned the reads to a related reference genome (*E. coli* 536 [GenBank:NC_008253]) with Bowtie [76] using two different modes. We first mapped reads allowing up to two mismatches to resemble the SNP calling pipeline in a recent, large study [74]. We also mapped reads using Bowtie's default mode, which allows mismatches between the reference and read until the sum of the quality values at those mismatches exceeds 70 [76]. We called SNPs using the SAMtools pileup program [77], requiring a Phred-style base call quality $\geq$40 and a coverage of $\geq$3 aligned reads. Having a reliable reference genome for both strains of *E. coli* allowed us to compare the SNPs detected using the reads to SNPs detected by performing a whole genome alignment. To call SNPs using the reference genomes, we used the MUMmer utility *dnadiff* which aligns the genomes with MUMmer, identifies the optimal alignment for each region, and enumerates SNPs in aligning regions [73]. We treat these SNPs as the gold standard (though there may be some false positives in improperly aligned regions) in order to compute recall and precision statistics for the read-based SNP calls.

In the first experiment, 128K additional reads of 4.12M aligned after correcting

Method	Reads mapped	SNPs	Recall	Precision
Two mismatch uncorrected	3.39M	79748	0.746	0.987
Two mismatch corrected	3.51M	80796	0.755	0.987
Quality-aware uncorrected	3.56M	85071	0.793	0.984
Quality-aware corrected	3.55M	85589	0.798	0.984

Table 2.5: We called SNPs in 35x coverage of 36 bp reads from *E. coli* K12 by aligning the reads to a close relative genome *E. coli* 536 with Bowtie using both a two mismatch and quality-aware alignment policy and calling SNPs with SAMtools pileup. SNPs were validated by comparing the *E. coli* K12 and *E. coli* 536 reference genomes directly. Under both alignment policies, correcting the reads with Quake helps find more true SNPs.

with Quake, of which 110K (85.8%) aligned to SNPs, demonstrating the major benefit of error correction before SNP calling. As seen in Table 2.5, with these reads mapped, we discovered more SNPs and recall increased at the same level of precision. Supporting the hypothesis that many of these newly discovered SNPs would exist in SNP-dense regions, we found that 62% of the new SNPs were within 10 bp of another SNP, compared to 38% for the entire set of SNPs. On the uncorrected reads, Bowtie's quality-aware alignment policy mapped 165K (4.9%) more reads than a two mismatch policy. Similarly, many of these new alignments contained SNPs, which led to more SNPs discovered, increasing recall with only a slight drop in precision. Using the quality-aware policy, slightly fewer reads mapped to the reference after error correction because some reads that could not be corrected and were removed could still be aligned. However, 33.7K new read alignments of corrected reads were found, which allowed the discovery of 518 additional SNPs at the same level of precision. Thus, error correction of the reads using Quake leads to the discovery of more true SNPs using two different alignment policies.

In order to demonstrate the ability of Quake to scale to larger datasets and benefit re-sequencing studies of humans, we corrected 1.7 billion reads from a Korean individual [SRA:SRA008175] [78]. This set includes 1.2B 36 bp reads and 504M 75 bp reads. Quake corrected 206M (11.9%) of these reads, trimmed an additional 75.3M (4.4%), and removed 344M (19.9%). Before and after error correction, we aligned the reads to the human genome (NCBI build 37) and called SNPs with Bowtie allowing two mismatches and SAMtools as described above (though requiring the diploid genotype to have quality $\geq$40 implicitly requires coverage $\geq$4). Because some putative SNPs had read coverage indicative of a repeat, we filtered out locations with read coverage greater than three times the median coverage of 19, leaving 3,024,283 SNPs based on the uncorrected reads. After error correction, we found 3,083,481 SNPs, an increase of 2.0%. The mean coverage of these SNPs was 20.1 reads, an increase of 4.8% over the coverage of these locations in the alignments of uncorrected reads, which should provide greater accuracy. Thus, Quake helps detect more SNPs in larger diploid genomes as well.

2.2.4 Data quality

Our experiences correcting errors in these datasets allowed us to assess the quality of the sequencing data used in a number of interesting ways. First, as has previously been established, nucleotide-specific error rates in Illumina sequencing

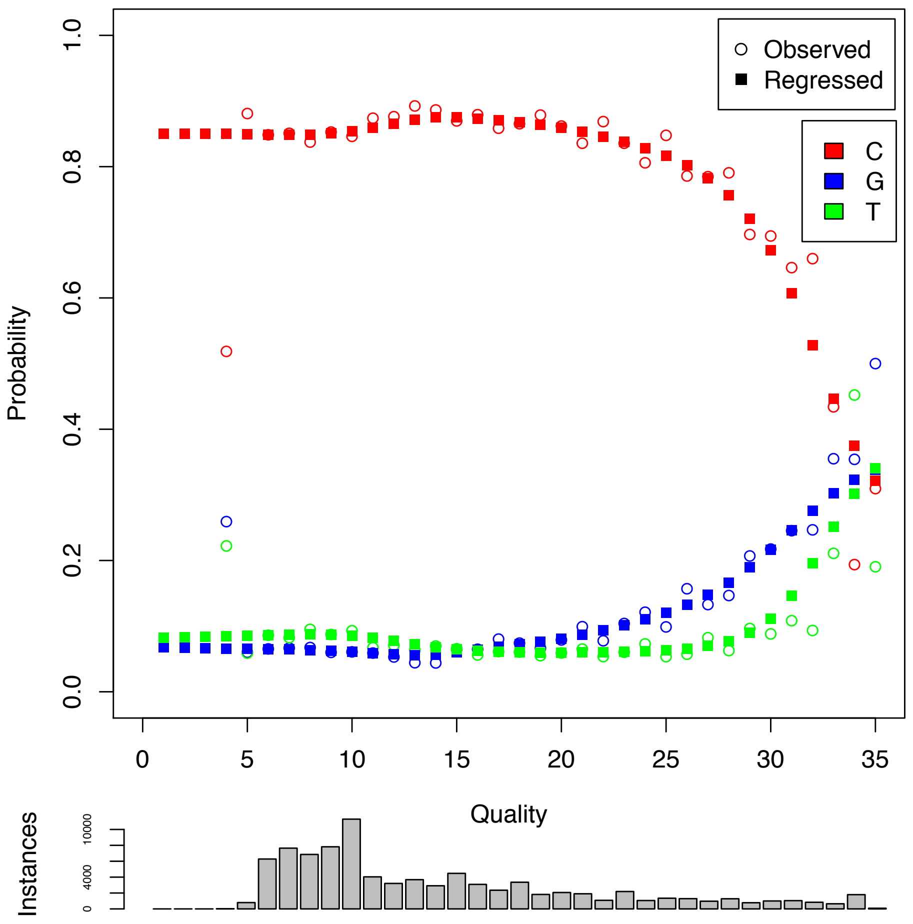


Figure 2.2: The observed error rate and predicted error rate after nonparametric regression are plotted for adenine by quality value for a single lane of Illumina sequencing of *Megachile rotundata*. The number of training instances at each quality value are drawn as a histogram below the plot. At low and medium quality values, adenine is far more likely to be miscalled as cytosine than thymine or guanine. However, the distribution at high quality is more uniform.

reads are not uniform [63]. For example, adenines were miscalled far more often as cytosine than thymine or guanine in *Megachile rotundata* (see Figure 2.2). As exemplified in the figure, error rates also differ significantly by quality value. While miscalls at adenines were highly likely to be cytosines at low quality, errors were closer to uniform at high quality positions in the read. Finally, error rates varied from lane to lane within a sequencing project. For example, the multinomial samples of nucleotide to nucleotide miscall rates for every pair of six lanes from the *Megachile rotundata* sequencing reads differed with unquestionably significant p-values using two sample chi square tests.

As sequencing becomes more prevalent in biological research, researchers will want to examine and compare the quality of an instance (single lane, machine run, or whole project) of data generation. Error correction with Quake provides two simple measures of data quality in the number of reads corrected and the number of reads removed. Furthermore, Quake allows the user to search for biases in the data like those described above using bundled analysis scripts on the log of all corrections made. Thus, researchers can detect and characterize problems and biases in their data before downstream analyses are performed.

2.3 Conclusions

The low cost and high throughput of 2^{nd}-generation sequencing technologies are changing the face of genome research. Despite the many advantages of the new technology, sequencing errors can easily confound analyses by introducing false polymorphisms and fragmenting genome assemblies. The Quake system detects and corrects sequencing errors by using the redundancy inherent in the sequence data. Our results show that Quake corrects more reads more accurately than previous methods, which in turn leads to more effective downstream analyses.

One way Quake improves over prior corrections methods is by q-mer counting, which uses the quality values assigned to each base as a means of weighting each k-mer. The coverage distributions of error and true k-mers cannot be separated perfectly according to their number of appearances due to high coverage errors and low coverage genomic regions. Yet, the choice of a cutoff to determine which k-mers will be trusted in the correction stage can have a significant affect on downstream applications like genome assembly. Weighting k-mer appearances by quality puts more distance between the two distributions because erroneous k-mers generally have lower quality than true k-mers. Furthermore, with q-mers, the cutoff value separating the two distributions no longer needs to be an integer. For example, at low coverage we might use 0.95 as a cutoff, such that k-mers that appear once with high quality bases would be trusted, but those with lower quality would not. Such fine-grained cutoff selection is impossible with simple k-mer counting.

Quake includes a sophisticated model of sequencing errors that allows the correction search to examine sets of corrections in order of decreasing likelihood, thus correcting the read more accurately. The model also helps to better identify reads with multiple sets of equally good corrections, which allows the system to avoid mis-correcting and creating a chimeric read. At a minimum, quality values should be included in error correction as a guide to the likely locations of sequencing errors.

In each dataset we examined, the rates at which each nucleotide was mis-called to other nucleotides were not uniform and often varied according to quality. Adjusting for these rates provides further improvements in error correction, and distinguishes our method.

We expect Quake will be useful to researchers intersted in a number of downstream applications. Correcting reads with Quake improves genome assembly by producing larger and more accurate contigs and scaffolds using the assemblers Velvet [12] and SOAPdenovo [46]. Error correction removes many of the false paths in the assembly graphs caused by errors and helps the assembler to detect overlaps between reads that would have been missed. Eliminating erroneous k-mers also significantly reduces the size of the assembly graph, which for large genomes may be the difference between being able to store the graph in a computer's memory or not [46]. In a re-sequencing application, correcting reads with Quake allows Bowtie [76] to align many more reads to locations in the reference genome where there is one or more SNPs. Reads containing variants already have differences from the reference genome; correcting additional differences caused by sequencing errors makes these reads easier to align and then available as input for the SNP calling program. Finally, Quake offers a unique perspective into the quality of the data from a sequencing experiment. The proportion of reads corrected, trimmed, and removed are useful statistics with which experiments can be compared and data quality can be monitored. The output log of corrections can be mined for troubling biases.

On microbial sized genomes, error correction with Quake is fast and unobtrusive for the researcher. On larger datasets, such as a human re-sequencing, it is computationally expensive and requires substantial resources. For the Korean individual reads, we counted k-mers on a 20-core computer cluster running Hadoop [79], which required 2–3 days. For error correction, the data structure used to store trusted k-mers requires 4^k bits, which is 32 GB for human if $k = 19$. Thus, the correction stage of Quake is best run on a large shared memory machine, where correction is parallelized across multiple threads using OpenMP [80]. Running on 16 cores, this took a few days for the Korean individual dataset. Future work will explore alternative ways to perform this step that would require less memory. This way correction could be parallelized across a larger computer cluster and made more accessible to researchers without a large shared memory machine.

K-mer based error correction programs are affected significantly by the cutoff separating true and error k-mers. Improvements in k-mer classification, such as the q-mer counting introduced by Quake, improve the accuracy of error correction. Coverage biases in 2^{nd}-generation sequencing technologies, which are largely inexplicable outside of the affect of local GC content, add to the difficulty [63]. Further characterization of these biases would allow better modeling of k-mer coverage and better classification of k-mers as true or error. In more repetitive genomes, the probability increases that a k-mer that is an artifact of an error actually does occur in the genome. Such k-mers are not really misclassified, but may cause Quake to ignore a sequencing error. To improve error correction in these cases, the local context of the k-mer in the sequencing reads must be taken into account. Though this was done for Sanger read error correction [16, 54, 55], it is not currently computationally and algorithmically feasible for high throughput datasets containing many more reads.

Quake's model for sequencing errors takes into account substantial information about which types of substitution errors are more likely. We considered using Quake to re-estimate the probability of a sequencing error at each quality value before using the quality values for correction. Doing so is difficult because Quake detects many reads that have errors for which it cannot find a valid set of corrections and pinpoint the errors' locations. If Quake re-estimated quality value error probabilities without considering these reads, the error probabilities would be underestimated. Additionally, the benefit of re-estimation is minimal because quality values are mainly used to determine the order in which sets of corrections are considered. Alternatively, passing on more information from the base calling stage, such as the probability that each individual nucleotide is the correct one, would be very helpful. Quake's error model could be made more specific, the need to learn nucleotide specific error rates would be alleviated, and more accurate error correction could be expected.

2.4 Methods

Quake detects and corrects errors in sequencing reads by using k-mer coverage to differentiate k-mers trusted to be in the genome and k-mers that are untrustworthy artifacts of sequencing errors. For reads with untrusted k-mers, Quake uses the pattern of trusted and untrusted k-mers to localize the errors and searches for the set of corrections with maximum likelihood that make all k-mers trusted. The likelihood of a set of corrections to a read is defined by a probabilistic model of sequencing errors incorporating the read's quality values as well as the rates at which nucleotides are miscalled as different nucleotides. Correction proceeds by examining changes to the read in order of decreasing likelihood until a set of changes making all k-mers trusted is discovered and found to be sufficiently unambiguous.

2.4.1 Counting k-mers

Counting the number of occurrences of all k-mers in the sequencing reads is the first step in the Quake pipeline. K must be chosen carefully, but a simple equation suffices to capture the competing goals. Smaller values of k provide greater discriminative power for identifying the location of errors in the reads and allow the algorithm to run faster. However, k cannot be so small that there is a high probability that one k-mer in the genome would be similar to another k-mer in the genome after a single nucleotide substitution because these occurrences confound error detection. We recommend setting k such that the probability that a randomly selected k-mer from the space of $\frac{4^k}{2}$ (for odd k considering reverse complements as equivalent) possible k-mers occurs in a random sequence of nucleotides the size of the sequenced genome G is $\sim$0.01. That, is we want k such that

$$\frac{2G}{4^k} \simeq 0.01 \qquad (2.2)$$

which simplifies to

$$k \simeq \log_4 200G \qquad (2.3)$$

For a $\sim$5 Mbp such as *E. coli*, we set k to 15, and for the $\sim$3 Gbp human genome, we set k to 19 (rounding down for computational reasons). For the human genome, counting all 19-mers in the reads is not a trivial task, requiring >100GB of RAM to store the k-mers and counts, many of which are artifacts of sequencing errors. Instead of executing this computation on a single large memory machine, we harnessed the power of many small memory machines working in parallel on different batches of reads. We execute the analysis using Hadoop [79] to monitor the workflow, and also to sum together the partial counts computed on individual machines using an extension of the MapReduce word counting algorithm [81]. The Hadoop cluster used in these experiments contains 10 nodes, each with a dual core 3.2 gigahertz Intel Xeon processors, 4 GB of RAM, and 367 GB local disk (20 cores, 40 GB RAM, 3.6TB local disk total).

In order to better differentiate true k-mers and error k-mers, we incorporate the quality values into k-mer counting. The number of appearances of low coverage true k-mers and high copy error k-mers may be similar, but we expect the error k-mers to have lower quality base calls. Rather than increment a k-mer's coverage by 1 for every occurrence, we increment it by the product of the probabilities that the base calls in the k-mer are correct as defined by the quality values. We refer to this process as *q-mer counting*. Q-mer counts approximate the expected coverage of a k-mer over the error distribution specified by the read's quality values. By counting q-mers, we are able to better differentiate between true k-mers that were sequenced to low coverage and error k-mers that occurred multiple times due to bias or repetitive sequence.

2.4.2 Coverage cutoff

A histogram of q-mer counts shows a mixture of two distributions— the coverage of true k-mers, and the coverage of error k-mers (see Figure 2.3). Inevitably, these distributions will mix and the cutoff at which true and error k-mers are differentiated must be chosen carefully [82]. By defining these two distributions, we can calculate the ratio of likelihoods that a k-mer at a given coverage came from one distribution or the other. Then the cutoff can be set to correspond to a likelihood ratio that suits the application of the sequencing. For instance, mistaking low coverage k-mers for errors will remove true sequence, fragmenting a *de novo* genome assembly and potentially creating mis-assemblies at repeats. To avoid this, we can set the cutoff to a point where the ratio of error k-mers to true k-mers is high, e.g. 1000:1.

In theory, the true k-mer coverage distribution should be Poisson, but Illumina sequencing has biases that add variance [63]. Instead, we model true k-mer coverage as Gaussian to allow a free parameter for the variance. K-mers that occur multiple times in the genome due to repetitive sequence and duplications also complicate the distribution. We found that k-mer copy number in various genomes has a "heavy tail" (meaning the tail of the distribution is not exponentially bounded) that is approximated well by the Zeta distribution [83], which has a single shape parameter. Our full model for true k-mer coverage is to sample a copy number from a Zeta distribution, and then sample a coverage from a Gaussian distribution with mean and variance proportional to the chosen copy number.

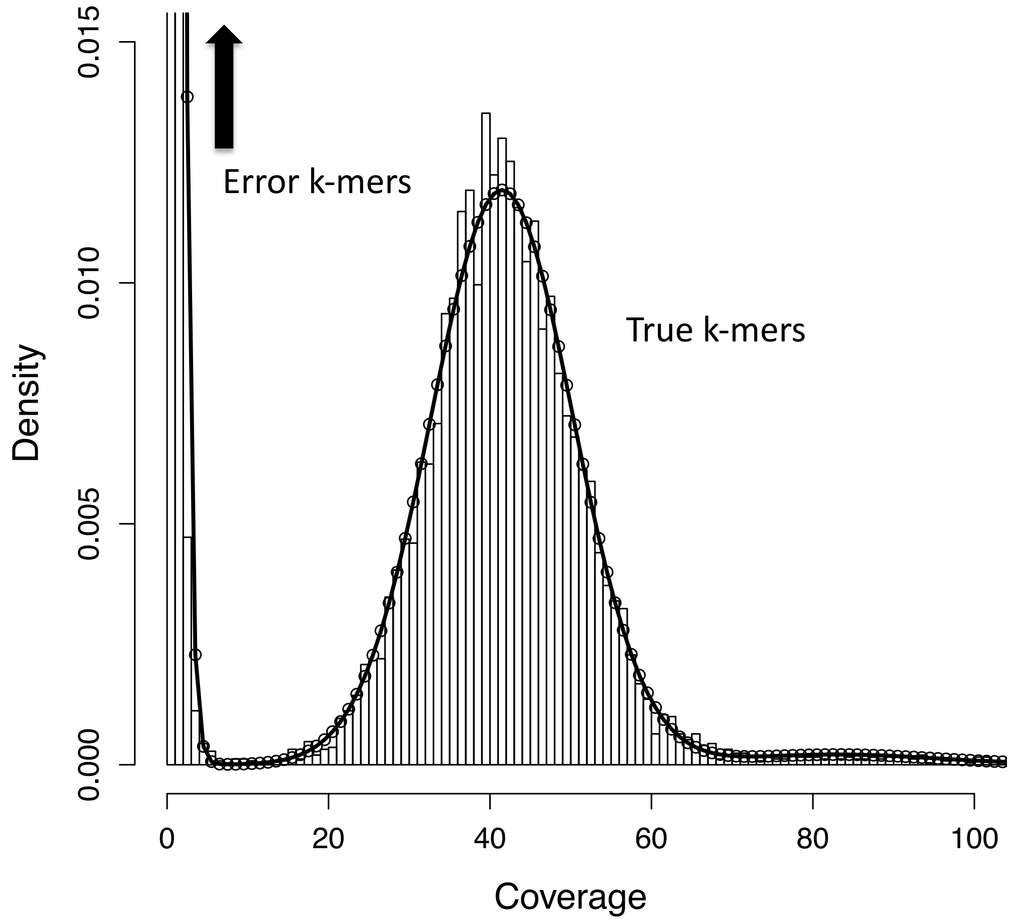


Figure 2.3: 15-mer coverage model fit to 76x coverage of 36 bp reads from *E. coli*. Note that the expected coverage of a k-mer in the genome using reads of length L will be $\frac{L-k+1}{L}$ times the expected coverage of a single nucleotide because the full k-mer must be covered by the read. Above, q-mer counts are binned at integers in the histogram. The error k-mer distribution rises outside the displayed region to 0.032 at coverage 2 and 0.691 at coverage 1. The mixture parameter for the prior probability that a k-mer's coverage is from the error distribution is 0.73. The mean and variance for true k-mers are 41 and 77 suggesting that a coverage bias exists as the variance is almost twice the theoretical 41 suggested by the Poisson distribution. The likelihood ratio of error to true k-mer is 1 at a coverage of 7, but we may choose a smaller cutoff for some applications.

The error k-mer coverage distribution has been previously modeled as Poisson [45]. In data we examined, this distribution also has a heavy tail, which could plausibly be explained if certain sequence motifs were more prone to errors than others due to sequence composition or other variables of the sequencing process. Additionally, by counting q-mers, we have real values rather than the integers that Poisson models. We examined a few options and chose the Gamma distribution with free shape and scale parameters to model error q-mer counts.

Finally, we include a mixture parameter to determine which of the two distributions a k-mer coverage will be sampled from. We fit the parameters of this mixture model by maximizing the likelihood function over the q-mer counts using the BFGS algorithm, implemented as the *optim* function in the statistical language R [84]. Figure 2.3 shows an example fit to 76x coverage of *E. coli*. Using the optimized model, we compute the likelihood ratio of error k-mer to true k-mer at various coverages and set the cutoff to correspond to the appropriate ratio.

2.4.3 Localizing errors

Once a cutoff to separate trusted and untrusted k-mers has been chosen, all reads containing an untrusted k-mer become candidates for correction. In most cases the pattern of untrusted k-mers will localize the sequencing error to a small region. For example, in Figure 2.4a, a single base substitution causes 15 adjacent untrusted 15-mers. To find the most likely region for the sequencing error(s), we take the intersection of a read's untrusted k-mers. This method is robust to a few

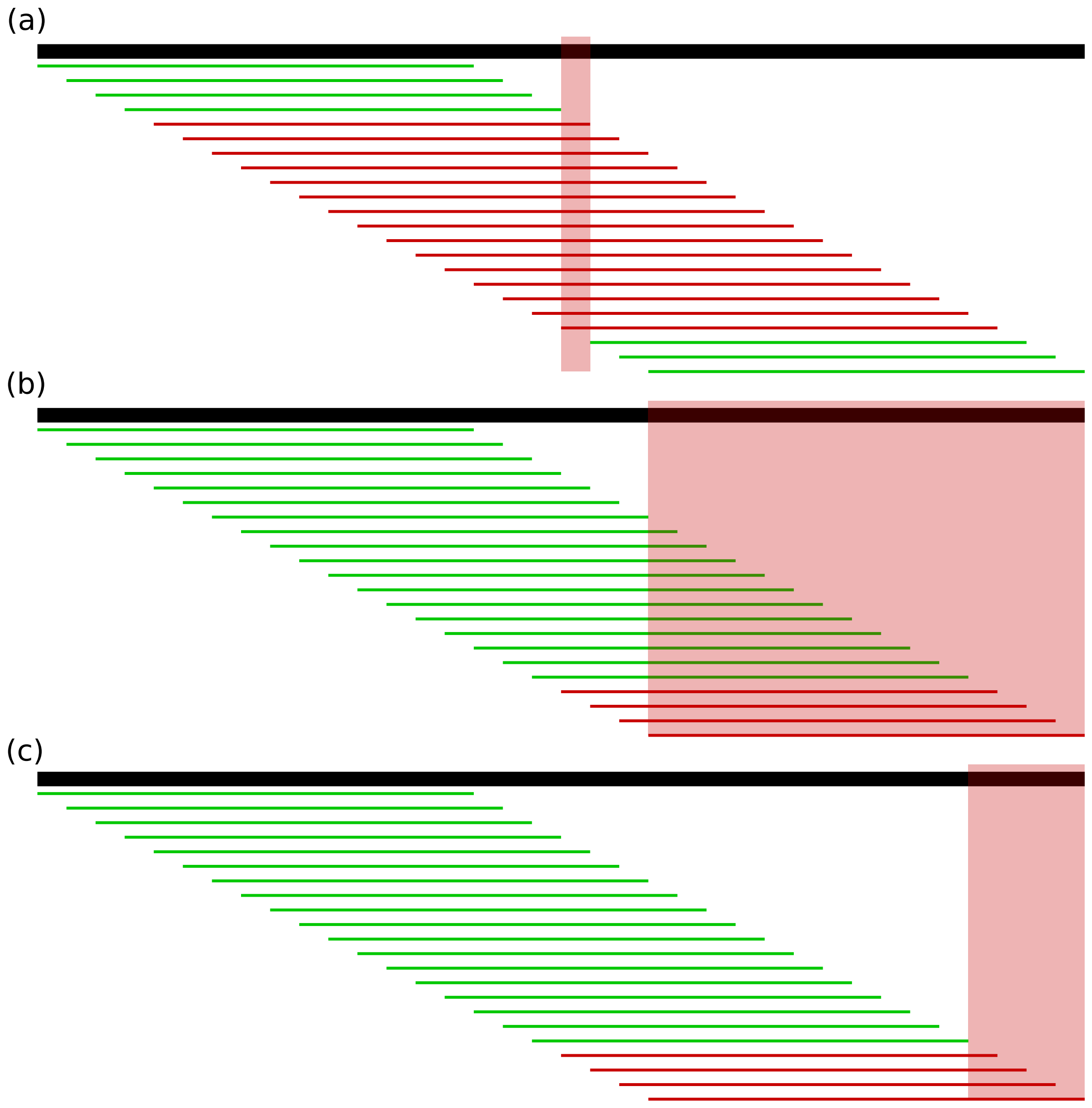


Figure 2.4: Trusted (green) and untrusted (red) 15-mers are drawn against a 36 bp read. In (a), the intersection of the untrusted k-mers localizes the sequencing error to the highlighted column. In (b), the untrusted k-mers reach the edge of the read, so we must consider the bases at the edge in addition to the intersection of the untrusted k-mers. However, in most cases, we can further localize the error by considering all bases covered by the right-most trusted k-mer to be correct and removing them from the error region as shown in (c).

misclassified error k-mers, but not to true k-mers with low coverage that are classified as untrusted. Thus, if the intersection of the untrusted k-mers is empty (which also occurs when there are multiple nearby errors) or a valid correction cannot be found, we try again localizing to the union of all untrusted k-mers.

A few more complications are worth noting. If the untrusted k-mers reach the edge of the read, there may be more sequencing errors at the edge, so we must extend the region to the edge, as in Figure 2.4b. In this case and in the case of multiple nearby sequencing errors, we may also benefit from considering every base covered by the right-most trusted k-mer and left-most trusted k-mer to be correct, and trimming the region as in Figure 2.4c. Because this heuristic is sensitive to misclassified k-mers, we first try to correct in the region shown in Figure 2.4c, but if no valid set of corrections is found, we try again with the larger region in Figure 2.4b. Finally, in longer reads we often see clusters of untrusted k-mers that do not overlap. We perform this localizing procedure and correction on each of these clusters separately. Altogether, these heuristics for localizing the error in a read vastly decrease the runtime of the algorithm compared to considering corrections across the entire read.

2.4.4 Sequencing error probability model

After finding a region of the read to focus our correction efforts on, we want to search for the maximum likelihood set of corrections that makes all k-mers overlapping the region trusted. First, we must define the likelihood of a set of corrections. Let $O = O_1,O_2,...O_N$ represent the observed nucleotides of the read, and $A = A_1,A_2,...A_N$ the actual nucleotides of the sequenced fragment of DNA. Given the observed nucleotides we would like to evaluate the conditional probability of a potential assignment to A. Assuming independence of sequencing errors at nucleotide positions in the read and using Bayes theorem, we can write

$$P(A = a|O = o) = \prod_{i=1}^{N} \frac{P(O_i = o_i|A_i = a_i)P(A_i = a_i)}{P(O_i = o_i)} \tag{2.4}$$

Because we compare likelihoods for a single observed read O at a time, $P(O_i = o_i)$ is the same for all assignments to A and is ignored. $P(A_i = a_i)$ is defined by the GC% of the genome, which we estimate by counting Gs and Cs in the sequencing reads. Let $p_i = 1 - 10^{-\frac{q_i}{10}}$ be the probability that the nucleotide at position i is accurate, where q_i is the corresponding quality value. Also, let $E_q(x, y)$ be the probability that the base call y is made for the nucleotide x at quality value q given that there has been a sequencing error. Then $P(O_i = o_i|A_i = a_i)$ can be specified as

$$P(O_i = o_i|A_i = a_i) = \left\{ \begin{array}{cc} p_i & \text{if } o_i = a_i \\ (1 - p_i)E_{q_i}(a_i, o_i) & \text{otherwise} \end{array} \right\} \tag{2.5}$$

Modeling sequencing errors with E allows for biases in base substitution that are known to exist for the Illumina platform. For example, one study found A to C was the most frequent error, likely because A and C are detected by one laser while G and T are detected by another [63]. Making the substitution distribution conditional upon the quality value allows this substitution bias to vary at different qualities, which was found to occur for Sanger sequencing [85] and here for Illumina (see Figure 2.2). Although some work has modeled error distributions conditionally on the position of the nucleotide in the read [86], we assume that quality values capture this sequencing cycle effect. Recent base-calling algorithms incorporate this effect on fluorescence intensity measurements explicitly in some way and generate quality values that satisfy our assumption [64–68].

The error matrices E are estimated from the sequencing reads as follows. First we initially set $E_q(x,y) = \frac{1}{3}\ \forall q, x, y$ and run the algorithm, counting the corrections by quality value and nucleotide to nucleotide type. During this initial pass, we only make simple, unambiguous corrections by abandoning low quality reads more aggressively and using a greater ambiguity threshold (described below). In order to reduce the variance of our estimate of E, we perform kernel smoothing across the quality q using a Gaussian kernel [87] with standard deviation 2. Let $C_q(x,y)$ be the number of times actual nucleotide x was observed as error nucleotide y at quality value q, $C_q(x)$ be the number of times actual nucleotide x was observed as an error at quality value q, and $N(q;u,s)$ be the probability of q from a Gaussian distribution with mean u and standard deviation s. Then E is defined by

$$E_q(x,y) = \frac{\sum_i C_{q_i}(x,y) N(q_i;q,2)}{\sum_i C_{q_i}(x) N(q_i;q,2)} \tag{2.6}$$

2.4.5 Correction search

Once we can assign a likelihood to a set of corrections and localize the error(s) to a specific region of the read, we must search for the set with maximum likelihood such that all k-mers in the corrected read are trusted. We refer to a set of corrections as *valid* if all resulting k-mers are trusted. In order to limit the search space, we

```
function SEARCH($R$)
    $P$.PUSH($\{\}, 1$)
    while $(C, L) \leftarrow P$.POP() do
        if VALID($R, C$) then
            return $C$
        else
            $i \leftarrow$ lowest quality unconsidered position
            for $nt \in [A, C, G, T]$ do
                if $R[i] == nt$ then
                    $C_{nt} = C$
                else
                    $C_{nt} = C + (i, nt)$
                $L_{nt} \leftarrow$ LIKELIHOODRATIO($R, C_{nt}$)
                if $L_{nt} > likelihood\_threshold$ then
                    $P$.PUSH($C_{nt}, L_{nt}$)
    return $\{\}$
```

Figure 2.5: Pseudocode for the algorithm to search for the most likely set of corrections that makes all k-mers in the read trusted. P is a heap-based priority queue that sorts sets of corrections C by theirlikelihood ratio L. The algorithm examines sets of corrections in decreasing order of their likelihood until a set is found thatconverts all k-mers in the read to trusted k-mers.

consider only sets of corrections for which the ratio of the likelihood of the corrected read to the original is above a fixed threshold (default 10^{-6}).

Figure 2.5 outlines the algorithm. To consider sets of corrections in order of decreasing likelihood, the algorithm maintains a heap-based priority queue P where each element contains a set of corrections C and the ratio of their likelihood to the original read's likelihood L. In each iteration through the main loop, the algorithm pops the maximum likelihood set of corrections C from the queue P. If C makes all k-mers in the region trusted, then it returns C. Otherwise, it examines the next lowest quality read position that has not yet been considered, which we track with minor additional bookkeeping. For each nucleotide substitution at this position, we compute a new likelihood and add the updated set of corrections to

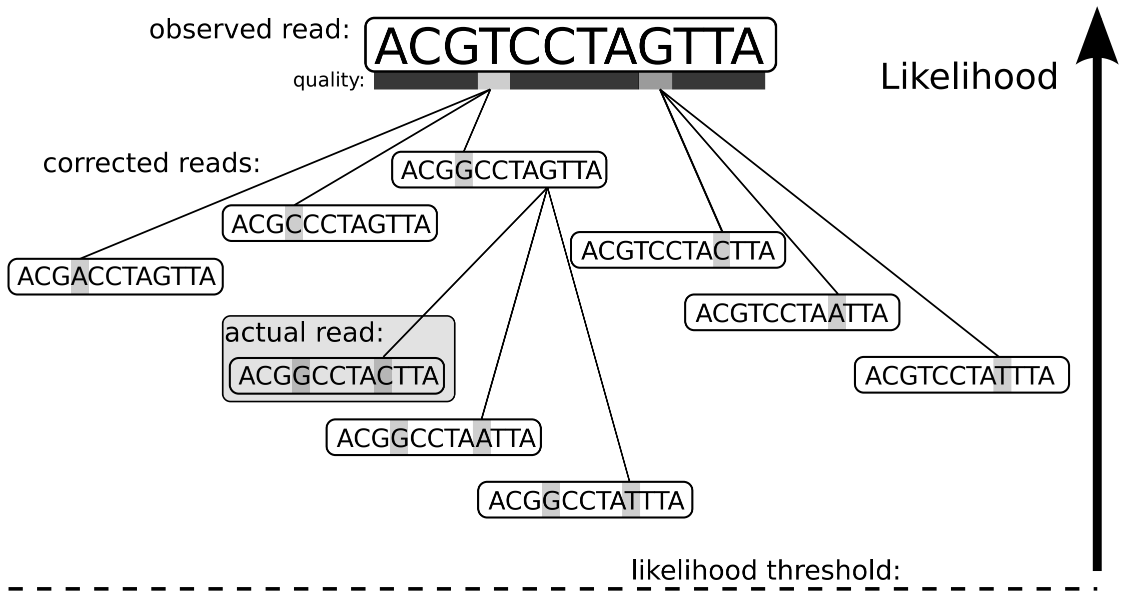


Figure 2.6: The search for the proper set of corrections that change an observed read with errors into the actual sequence from the genome can be viewed as exploring a tree. Nodes in the tree represent possible corrected reads (and implicitly sets of corrections to the observed read). Branches in the tree represent corrections. Each node can be assigned a likelihood by our model for sequencing errors as described in the text. Quake's algorithm visits the nodes in order of decreasing likelihood until a valid read is found or the threshold is passed.

the priority queue if its likelihood ratio is above the threshold. If the queue empties without finding a valid set of corrections, we abandon the read. This procedure could alternatively be viewed as searching a tree where nodes are corrected reads and branches represent corrections (see Figure 2.6).

In practice, we make a few adjustments to this procedure. Reads from repeats may have multiple sets of valid corrections separated by a small likelihood difference so that the true correction is ambiguous. Therefore, we actually continue past the point of finding a valid set of corrections to ensure that another valid set does not exist within a certain likelihood threshold (default 0.1). As described, the algorithm will devote a large majority of its computation effort to the lowest quality reads, which have many potential sets of corrections to consider. In order to balance correction sensitivity with speed, we pre-screen the error region and immediately abandon a read if its error region is filled with low quality base calls. More specifically, in our experiments we found that regions containing $\geq$13 positions with a probability of error >1% were difficult or impossible to correct quickly, and these reads are abandoned without further effort. For regions containing $\geq$9 such positions, we increase the likelihood ratio threshold to 10^{-3} so that we only consider a limited number of corrections before giving up.

In order to run Quake on very large datasets (e.g. containing billions of reads), we must be able to determine very quickly whether a set of corrections makes all k-mers trusted. We accomplish this by mapping all 4^k k-mers to an index in a bit array that is set to 1 if the k-mer is trusted and 0 otherwise. For 15-mers this bit array uses just 128 MB of space, while it requires 32 GB for 19-mers, which are needed for larger genomes. If memory usage must be reduced, a Bloom filter could be used to hash the trusted k-mers in <4 GB at the expense of occasional false positive queries [52].

2.5 List of abbreviations

bp: base pair, K: thousand, M: million, B: billion, MB: megabytes, GB: gigabytes, Mbp: megabases, Gbp: gigabases, SNP: single nucleotide polymorphism

2.6 Acknowledgements

Thanks to Gene Robinson at the University of Illinois for permission to use preliminary sequence data from *Megachile rotundata*. Thanks to Héctor Corrada Bravo and Darryl Reeves for helpful discussion and feedback.

This work was supported by NIH grants R01-LM006845 and R01-GM083873.

Chapter 3

Detection and correction of false segmental duplications caused by genome mis-assembly

The primary goal of a genome project is to produce large and accurate segments from the chromosomes that can then be used to analyze the genome. Because genome assembly software makes assumptions about the data and uses heuristics in order to make the problem feasible, sometimes mistakes are made. Additionally, these errors may be very misleading during downstream analysis of the genome.

This chapter describes work done with Steven Salzberg where we found a particular type of problematic mis-assembly in many important public vertebrate genome assemblies. In a heterozygous region of a diploid genome where there are many differences between the two copies of the chromsome, the assembler may construct two distinct contigs covering the region and place them nearby in the final assembly. This gives the illusion of a segmental duplication, which is a well-studied evolutionary event. We developed a method to detect such mis-assemblies based on sequence alignment and probabilistic modeling of paired-end read distances. Then we ran the method on a set of recent genome assemblies and analyzed the prevalence of mis-assemblies and their consequences. The following manuscript was published in March 2010 in Genome Biology [88].

3.1 Background

Ever since the publication of the Drosophila melanogaster genome [89], large-scale eukaryotic sequencing projects have increasingly used the whole-genome shotgun (WGS) strategy to sequence and assemble genomes. Algorithms to assemble a genome from WGS data have grown increasingly sophisticated, but problems nonetheless remain, and despite the ever-accelerating pace of "complete" genome announcements, not a single vertebrate genome is truly complete. While it is widely known that draft assemblies contain gaps, the extent of errors in published assemblies is less well known.

One particular type of error that confounds analysis is an erroneously duplicated sequence. Duplications involving large genomic regions, known as segmental duplications, have been the subject of intensive study in the human genome [90, 91] and other species (e.g. [92, 93]). Although much effort has gone into avoiding the problem of artificially collapsing duplicated regions [18], less attention has been paid to the assembly processes that improperly reconstruct duplicated regions from WGS data, which is a problem for assembly of diploid organisms. Genome assembly software is generally designed as if the sequencing data ("reads") were derived from a clonal, haploid chromosome. This was indeed the case for early WGS projects, which targeted bacteria [2] or archaea [94], but in general is not true for more genetically complex organisms like vertebrates. Diploid organisms inevitably have differences between their two copies of each chromosome, and these differences complicate assembly. This problem can be alleviated somewhat by choosing highly inbred individuals with few differences between chromosomes for sequencing. But for many species such inbred lines are not available, and for others the inbreeding has not resulted in the desired homozygosity [95]. Adding further to the confusion is the fact that virtually all DNA sequence databases (including GenBank, EMBL, and DDBJ) maintain only a single copy of each chromosome for all species.

When assembling a diploid genome with any significant variation between the two chromosomes, even the best assembly software may find it difficult to reconstruct a single sequence for heterozygous regions. As a result, genome projects in which a highly heterozygous individual was sequenced have documented problems with assembly, e.g. *Anopheles gambiae* [96], *Candida albicans* [97], and *Ciona savignyi* [98]. Even with highly inbred strains such as mouse, mis-assemblies due to heterozygosity have been described [93, 99].

Specifically, when two copies of a chromosome diverge sufficiently, an assembler will create two distinct reconstructions (contigs) of the divergent regions, using reads from each of the respective copies of the chromosome. If the sequencing project used paired-end sequences, as is commonly done, then both contigs are likely to have linking information from these reads to their “mates” in the same surrounding region. The duplicate contigs might then be placed into the genome at adjacent locations, possibly with some non-duplicated flanking sequence on either side. The incorporation of both haplotypes into the genome gives the illusion of a segmental duplication. In addition, single nucleotide polymorphisms (SNPs) and small indels captured in the differences between the two haplotype contigs are missed.

Segmental duplications and SNPs have been studied extensively for their important role in genome evolution [100–102] and for their associations with disease [103,104]. Previous attempts to accurately quantify the number of duplications in the human genome have briefly discussed the likelihood that highly similar (e.g. >98% identity) apparent intrachromosomal duplications may be erroneous [90,91]. We hypothesize that many duplicated regions in current, published genome sequences are in fact errors due to mis-assembly, and in this chapter we attempt to identify and quantify the frequency of this type of assembly error. To accurately detect mis-assembled haplotype sequence, we incorporate the reads' mate pair information, a data source that has not been previously used in duplication detection. Mate pair constraints, coverage data (the number of reads covering a particular locus in a genome), and read placement data are all valuable tools in validating assemblies [105–107].

In this chapter, we present a contig-centric analysis of mis-assembled segmental duplications. Our process begins by aligning every contig in an assembly to the surrounding sequence (see Methods for details). Those contigs that have strong similarity to nearby regions – apparent segmental duplications – are analyzed to determine whether the reads' mate pairs would be more consistent if the duplicated segments were merged into one copy. In cases where this is true, the genome can be corrected by re-computing the consensus sequence using all reads, which then uncovers polymorphisms between the two haplotypes that had previously been overlooked.

3.2 Results and Discussion

3.2.1 Genomes

We ran our mis-assembly detection pipeline on the genomes of domestic cow, *Bos taurus* (UMD1.6, a precursor to UMD2 where all detected mis-assemblies were fixed [108]); chimpanzee, *Pan troglodytes* (panTro2 assembly [109]); chicken, *Gallus gallus* (galGal3 assembly [110]); and dog, *Canis familiaris* (canFam2 assembly [111]). These genomes were assembled with three different assemblers: Celera Assembler [15], Arachne [16], and PCAP [112]. We selected them based on their large size, biological significance, range of assembly software, and (most critically) the availability of low level assembly data including the placements of reads in contigs. We chose to analyze the UMD2 cow assembly over the BCM4 assembly [113, 114] because placement of reads in contigs is a requirement of our method and such information is not available for BCM4.

Table 3.1 displays the results of running our pipeline on these four genomes. Contigs that align to nearby sequence appear as duplicated contigs, and those that appear to be erroneous (see Figure 3.1) are summarized in the table as mis-assembled contigs. For a significant number of apparent duplications, especially in chicken and chimpanzee, the mate pairs are more consistent when the contig is superimposed on a nearby duplication, suggesting that the sequence in the contig and the nearby sequence represent two slightly divergent haplotypes that belong to the same chromosomal position. These results demonstrate that published whole-genome assemblies of diploid species contain mis-assemblies due to heterozygosity.

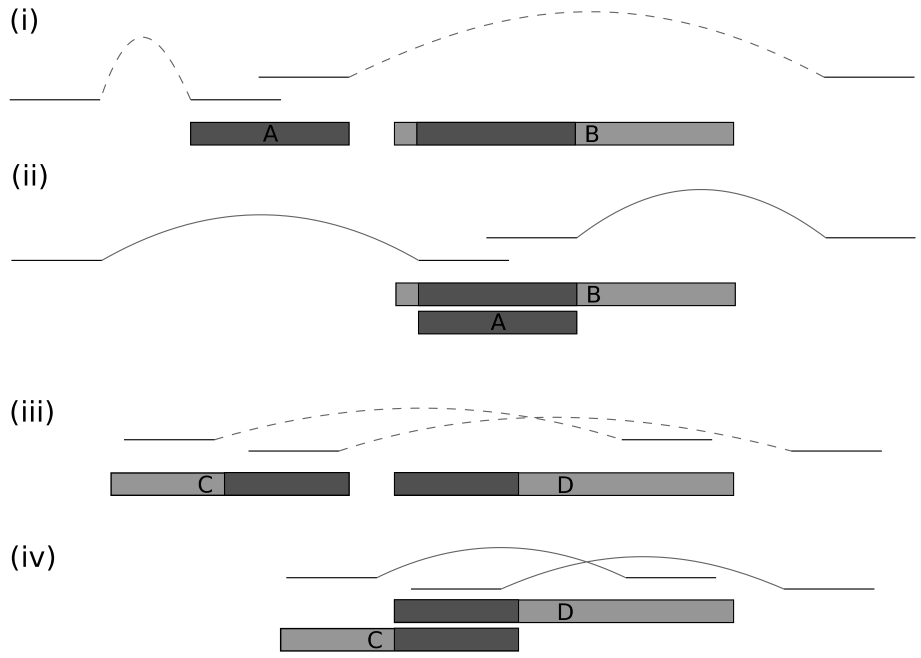


Figure 3.1: Mis-assembled DCC and DOC. Assemblers may mistakenly form two contigs from the two haplotypes, as shown in (i) where contig A contains heterozygous sequence and contig B contains homozygous sequence (light) on both sides of a matching heterozygous region (dark) (with sequencing reads as lines above them). We refer to A as a duplicated contained contig (DCC). We can identify this situation by finding an alignment between contigs A and B that completely covers contig A and comparing contig A's mate pair links in the original location to those same links when contig A is overlaid on contig B at the location of its alignment, as shown in (ii). Dashed curves in (i) indicate distances that are significantly shorter (left side of figure) or longer (right) than expected; solid curves indicate distances that are consistent with specifications. In the situation shown here, we would designate contig A as an erroneous duplication likely to have been caused by haplotype differences. Alternatively, heterozygous sequence may be separated into two contigs that each include some homozygous sequence on opposite ends, as in contigs C and D in (iii), which we refer to as duplicated overlapping contigs. If a significant alignment exists between the ends of these contigs and the distances between mate pairs pointing right from contig C and left from contig D better match their expected fragment sizes when the contigs are joined, we designate the region as an erroneous duplication and join the contigs as in (iv).

	Gallus gallus (chicken)	*Pan troglodytes* (chimpanzee)	*Bos taurus* (cow)	*Canis familiaris* (dog)
Assembled genome size	1.00 Gb	2.89 Gb	2.57 Gb	2.33 Gb
DCCs	4418 (7.6 Mb)	5467 (8.0 Mb)	1297 (3.71 Mb)	80 (170 Kb)
Mis-assembled DCCs	2303 (3.61 Mb)	2298 (2.97 Mb)	394 (1.18 Mb)	2 (1.8 Kb)
DOCs	5947 (11.2 Mb)	13571 (14.1 Mb)	1366 (1.88 Mb)	22 (34.7 Kb)
Mis-assembled DOCs	5698 (10.8 Mb)	13159 (13.7 Mb)	1094 (1.09 Mb)	8 (7.9 Kb)
Total mis-assemblies	8001 (14.4 Mb)	15457 (16.7 Mb)	1488 (2.27 Mb)	10 (9.7 Kb)

Table 3.1: Erroneously duplicated sequences in vertebrate genomes. Genome sizes were determined by summing the lengths of all contigs and linked gaps in each assembly. Duplicated contained contigs (DCCs) include all contigs that aligned to nearby sequence where the contig is completely contained within another contig, as shown in Fig. 3.1(ii). Mis-assembled DCCs are the subset of DCCs that we identified by mate pairs as erroneous duplications (assembly errors). Duplicated overlapping contigs (DOCs) include all pairs of nearby contigs that overlap at their ends, followed again by the subset found to have more consistent mate pairs when merged. Contigs that were not designated as mis-assembled either had consistent mate pairs in their original location, or else lacked sufficient mate-pair data to make a determination. Note that this analysis used the UMD 1.6 version of the *Bos taurus* genome, and based on these results, erroneous duplications were removed from the published UMD 2.0 assembly.

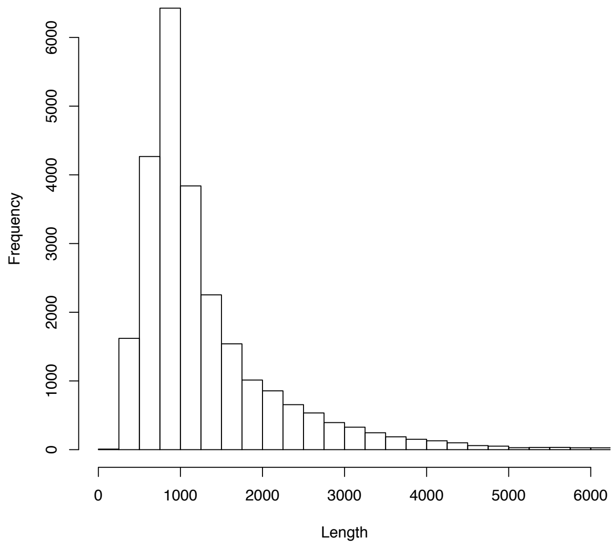


Figure 3.2: Erroneous duplication lengths. Contigs from chimpanzee, chicken, cow, and dog that are classified by our procedure as mis-assembled erroneous duplications were binned by length at 250 bp resolution. The distribution was similar for each individual species.

The four assemblies displayed a wide range of incorrectly assembled haplotype sequence. The assembly of the dog genome with Arachne had the fewest problems by far, which we attribute to the extensive post-assembly procedures that were applied to that genome [115] and to that group's experience with highly polymorphic genomes such as *Ciona savignyi* [98]. We therefore excluded the dog genome from the remainder of the experiments below. By contrast, chimpanzee and chicken, assembled with PCAP, contain 16.7 and 14.4 Mb of sequence, spread across thousands of contigs, that appears to represent erroneous segmental duplications. The cow genome assembly had fewer such regions (2.27 Mb), which are corrected in the publicly released version of the genome.

The distribution of sizes of mis-assembled contigs in the four genomes is de-

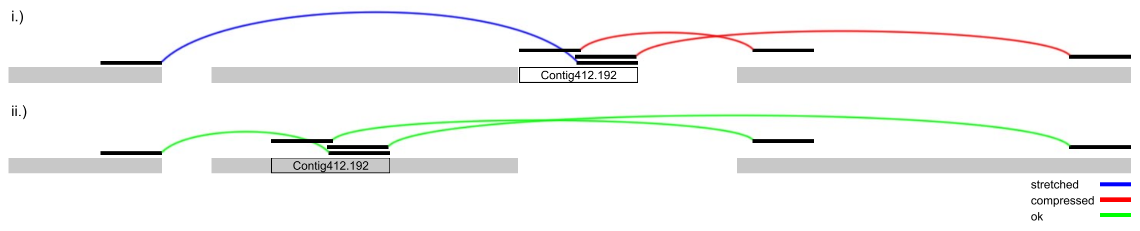


Figure 3.3: In (i), Contig412.192 is placed in the chimpanzee assembly on chromosome 1 such that mated reads pointing to the right have compressed mate pair distances and mated reads pointing to the left have stretched mate pair distances. By moving the 1537 bp contig to a nearby location where it aligns in its entirety at 98.9%, the distances between mated reads become far more consistent with their library insert lengths. Thus, Contig412.192 is classified as a spurious duplication.

picted in Figure 5.1. Most of the contigs are less than 2000 bp, though there are a few larger contigs up to 28 Kb in cow. The median alignment percent identity between a falsely duplicated contig and the nearby region to which it aligns is 98.1%. Few contigs align at greater than 99.5%. These statistics were similar in each genome. Figure 3.3 displays an example spurious duplication in chimpanzee detected by analyzing mate pairs.

3.2.2 Use of the human genome to check duplications

For the chimpanzee genome, we used the human genome as an independent resource to confirm that the contigs we identified as haplotype variants were likely to be mis-assemblies rather than true duplications. Because the human genome has been the subject of far more analysis and refinement than any other vertebrate genome, we made the simplifying assumption that it does not contain any mis-assembled segmental duplications. A recent study found that 83% of chimpanzee duplications are shared by human [116]; thus it is reasonable to assume that a large majority of the duplicated contigs we found in the chimpanzee assembly should be duplicated in human as well if they truly are duplications. We aligned all chimpanzee contigs classified as mis-assembled in Table 3.1 to the human genome (NCBI build 36) using MUMmer [73]. Many of the sequences contain high-copy repetitive elements, and to avoid confusion we first ran the program RepeatMasker [117], which screens the sequence against a database of known interspersed repeats and low complexity sequence, on the chimpanzee sequences and removed the 2962 contigs (out of 15457) that were more than 90% masked. Of the remaining 12495 contigs, only 486 (3.9%) were found in multiple copies in human. This is dramatically lower than the 83% rate reported in the Cheng et al. study, indicating that most of these contigs are likely to be single-copy. Furthermore, detection of a chimpanzee contig as multiple copies in human does not preclude the possibility of a mis-assembly in the location we identified.

3.2.3 Coverage depth

Another independent check on the accuracy of our mis-assembly detection method is the depth of coverage by WGS reads. Because WGS reads represent a random sample of the genome, the expectation of the coverage at any location is equal to the global average coverage. We measured coverage using the A-statistic [15], which computes the log of the ratio of the likelihood that a contig is a single-copy segment and the likelihood that it is duplicated. For all duplicated regions, we considered WGS reads from both of the contigs that were placed in the region covered by the span of the alignment of the contigs. We found that, for the regions identified as mis-assembled in Table 3.1, 77.2% of the chicken contigs, 76.3% of the chimpanzee contigs, and 94.1% of the cow contigs had A-statistics greater than zero, indicating that they were likely to be single-copy regions; i.e. that they were mis-assembled and falsely present in two copies.

Read coverage is a strong indicator of duplication, but is subject to considerable noise at the sequence lengths considered (see Figure 5.1). As a further check on our method, we examined several borderline cases where read coverage, as indicated by the A-statistic, suggested that a contig was duplicated even though our analysis of mate pairs indicated that it was spurious. In each case, the mated reads associated with the contig in question strongly suggested a mis-assembly. For example, Contig438.7 (2983 bp) in the chimpanzee assembly has an A-statistic strongly indicating that it is duplicated. However, the existing placement is supported by only a single pair of mated reads, while every other mate pair is stretched by $\sim$61000 bp. If instead we superimpose this contig on Contig 438.13, to which it aligns at 98.6%, 28 mated reads would be the correct distance from one another without a perceivable bias. Despite the read coverage, mate-pair data show that Contig 438.7 clearly represents a mis-assembly in the current placement. While depth of read coverage can be a very useful tool for detecting mis-assemblies [105, 106], cases like these where repetitive sequence is mis-assembled can only be detected by using the mate pairs.

3.2.4 Genes affected by erroneous duplications

We examined the annotations for the erroneous duplications found by our method using the NCBI Entrez Gene database [118] as a source for annotation. This analysis only examined the chicken and chimpanzee assemblies, because the intermediate UMD1.6 cow assembly used in this study was not annotated. For chicken, 3459 of the mis-assembled contigs overlap a gene model, and 585 of these contain protein-coding sequence. In chimpanzee, 6121 contigs overlap a gene model, with 381 containing coding sequence. A complete list of the particular genes affected is provided in Additional file 2.

In most cases, contigs containing coding sequence contained one or two exons, and removing the duplicated region would maintain the consistency of mRNA alignments. Specifically, no mRNA contained two copies of the exon even though it is duplicated nearby. If the exon prediction differed on the two copies of the duplication, we checked that no exons overlapped or changed order after moving the contig. In other words, the mRNA alignments support our hypothesis that the duplication is erroneous. This was the case for 316 of the 381 chimp contigs and 427 of the 585 chicken contigs that contained coding sequence. Figure 3.4 shows an example from the chimpanzee genome in which an erroneous duplication contains three exons, but none of the mRNA sequences contain duplicate copies of those exons as might be expected if the duplication were real.

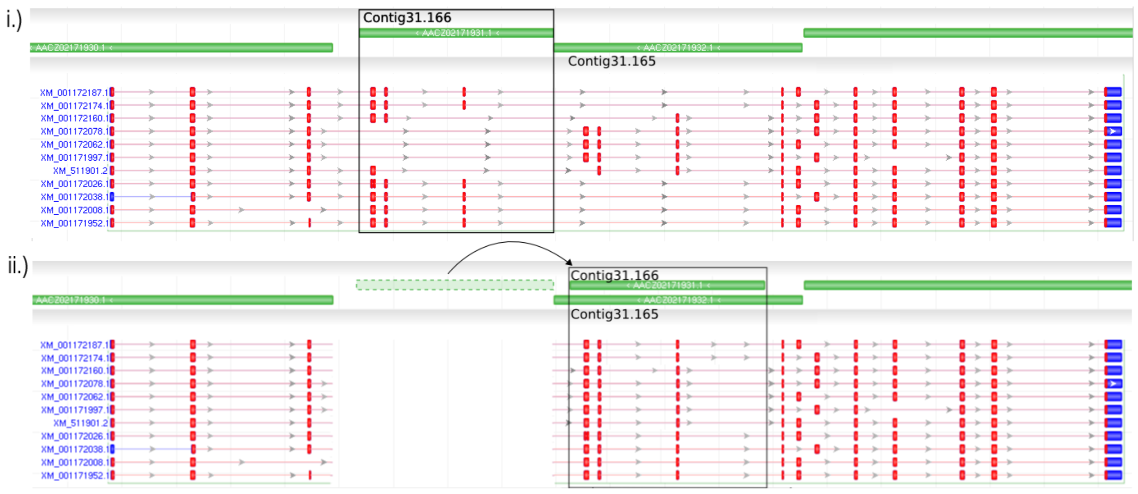


Figure 3.4: SCPEP1 consistent mRNA alignments. The screenshot above, taken from the NCBI Sequence Viewer, displays the gene model for serine carboxypeptidase 1 (SCPEP1) where green bars represent contigs and mRNA alignments are listed with red bars as alignments to exons. (i) Contig31.166 contains three putative exons. However, it overlaps neighboring Contig31.165 for all of its length (7162 bp) at 98.6% identity, and mate pairs indicate that the two contigs came from the same position. Every mRNA alignment takes a path through the exons such that only one copy of each duplicated exon is included. When the contig is moved (ii), the extra copies of these three apparently duplicated exons are removed, but all of the alignments remain consistent.

	Gallus gallus (chicken)	*Pan troglodytes* (chimpanzee)	*Bos taurus* (cow)	*Canis familiaris* (dog)
Unplaced contigs	25957 (56.8 Mb)	47549 (153 Mb)	133918 (307 Mb)	7551 (75.1 Mb)
Mis-assembled DCCs	8044 (16.3 Mb)	10407 (21.3 Mb)	1793 (4.92 Mb)	2 (2.92 Kb)
Mis-assembled DOCs	663 (1.23 Mb)	2204 (2.96 Mb)	751 (827 Kb)	15 (23.0 Kb)

Table 3.2: In each of the four genome assemblies, a large set of contigs that could not be placed on the chromosomes exists (summarized in the first row). We aligned these contigs against all placed contigs and identified those that were contained in placed sequence (DCCs) or overlapped a placed contig (DOCs). We checked mate pairs for evidence that these contigs should be merged with the placed contigs. The table shows the number of contigs of each type found to have a supported placement in the assembly for each alignment type. These unplaced contigs are likely haplotype variants of the sequence in the placed contigs.

3.2.5 Unplaced contigs

We developed a variation of our haplotype mis-assembly pipeline to identify likely haplotype variants among the unplaced contigs (those not assigned to a chromosome) in each genome, including dog. We aligned all unplaced contigs to all placed contigs, identified alignments indicative of a mis-assembly, and checked for consistent mate pairs for the unplaced contig in the location implied by the alignment (see Methods for details). The results are displayed in Table 3.2. As with the placed contigs, the amount of unplaced haplotype sequence varied considerably among genomes. In all but the dog genome, a significant number of contigs were identified as haplotype variants by this procedure.

3.2.6 SNPs and indels

The mis-assembled contigs detected by our pipeline represent distinct sequences that should have been assembled into a single consensus. We recomputed the multiple alignment of all reads from both haplotypes for each erroneous duplication using Seq-Cons [119]. With a new multiple alignment of reads to represent the region, polymorphisms that went unnoticed when the haplotypes were separated could be detected. To be conservative, we only count polymorphisms for pairs of contigs with read coverage indicative of a single-copy segment in order to filter out mis-assembled repetitive sequence. After filtering for high quality neighboring sequence, we report 124432 SNPs and 22960 indels in chimpanzee, 188617 SNPs and 16840 indels in chicken, and 50209 SNPs and 10764 indels in cow. For chimpanzee and chicken, we submitted these SNPs to the public SNP database dbSNP (submitted SNP numbers 181362056 to 181746453) [120]. To assess the number of novel SNPs contributed for each organism, we aligned the sequence surrounding each SNP against entries for that organism in dbSNP. 26451 chimpanzee SNPs, 21646 chicken SNPs, and 1727 cow SNPs matched entries in the database. Thus, a significant number of novel polymorphisms would have been lost due to mis-assembly but were recovered by our pipeline.

3.3 Conclusions

Assembling the genome of a diploid organism remains a formidable task, especially in the presence of heterozygosity. Most genome sequencing projects to date have attempted to create a single reference genome, which has involved merging the two copies of each chromosome into one consensus sequence. Assembly algorithms use a variety of strategies to avoid collapsing highly similar copies of repetitive sequences (e.g. strict requirements for an overlap between two reads), which is of utmost concern when detecting duplications [90,91]. However, these very same algorithmic techniques can separate two haplotype variants – which ought to be merged – creating an erroneous duplication. No assembly algorithm yet invented does a perfect job of balancing these competing goals.

A number of assembly methods have been designed to avoid mis-assemblies due to haplotype divergence. In *Anopheles gambiae*, a conservative scaffold layout algorithm was implemented to reduce placement of redundant sequence [96]. A procedure to filter out overlaps between reads originating from different chromosomes was used before assembling *Ciona savignyi* [98]. For the grapevine genome, scaffolds that aligned for >40% of their length at high identity were visually inspected and in most cases, one copy was removed [121]. In the assembly of *Candida albicans*, significant heterozygosity and the aggressive assembly strategy of the Phrap assembler created numerous mis-assembled contigs, which needed to be carefully stitched back together [97].

At the post-assembly analysis stage, a number of reports have indicated problems with false duplications, but no previous work has reported an algorithmic solution. For example, two independent assessments of duplications in a previous build of the human genome reported nearly identical intrachromosomal duplications [90,91] and questioned their reliability. More recently, researchers found that significant erroneous duplications – due to haplotype differences – permeate nematode genome assemblies [95].

The work described here presents an algorithm to detect erroneous duplications that are caused by heterozygosity between haplotypes. Our pipeline relies not only on sequence alignments among contigs but also a novel, detailed analysis of mate pair constraints that provides fine-scale resolution of the evidence for each duplication. We ran our pipeline on a set of vertebrate genomes that represent a sample of different assembly methods. Our results demonstrate some published assemblies, including chimpanzee and chicken, are riddled with erroneous duplications, with >14 Mb of problematic sequence in each. Uncovering these mis-assemblies requires a revision of the amount of sequence covered by segmental duplications in these genomes. Segmental duplications have proven to be relevant to disease [103] and integral to studies on genome evolution [100, 101], and proper identification of duplications is a necessity for investigations into their role in these phenomena. Our results remove thousands of duplications from the chimpanzee, chicken, and cow genomes. In most cases, the false duplications described here are highly similar, making it appear that they are very recent events, which have been of great interest, particularly in primates [122, 123].

In addition, when the sequences from a heterozygous region are erroneously assembled into two separate contigs, we lose information about the heterozygosity in that region. Single nucleotide polymorphisms (SNPs) and insertions/deletions (indels) are valuable for many reasons, including genotyping, evolutionary analysis, and the relation of genotype to phenotype [104, 124, 125]. For example, we must know which of the SNPs between chimpanzee and humans are due to intra-species diversity in order to accurately model evolution in the primate lineage [102]. By recomputing the multiple alignment of reads in the mis-assembled duplications, we were able to find tens of thousands of additional polymorphisms that were overlooked in the original analyses of the genomes. In the past, discovery of this number of polymorphisms has required expensive efforts to sequence many different individuals [124, 126, 127].

Numerous recent human genome resequencing projects have performed a diploid assembly where both chromosomes are described [128, 129]. These projects begin by assembling a single reference genome and then perform a post-processing step called "haplotype assembly" where the assembly is assumed to be correct and variations in the consensus multiple alignment of reads are used to pull apart the two haplotypes for stretches of sequence as long as possible [130–132]. In fact, "haplotype assembly" algorithms will not succeed unless the two haplotypes are assembled into a single contig. Thus, correcting mis-assemblies of haplotype sequence is an integral first step that has not previously been considered and would certainly result in longer stretches of haplotype sequence since these regions are replete with informative variations.

Due to their greatly lower cost and higher throughput, next-generation sequencing technologies are rapidly being adopted for large genome projects. The limitations of short reads in resolving repetitive areas of the genome due to the absence of reads that cover the entire region have been discussed previously [13], and resolving haplotype differences will be difficult for similar reasons. Most of the programs to assemble short reads incorporate a procedure to attempt to rid the assembly of these contigs; e.g., by detecting bubbles in the de Bruijn graph of the reads [12]. However, similar algorithms have been used for many years [133], but have not been able to rid large genome assemblies of false duplications due to haplotype differences, as demonstrated here. Accurate assembly of segmental duplications, and the avoidance of false duplications, is likely to remain a difficult problem for the foreseeable future.

3.4 Methods

We developed a pipeline to identify mis-assemblies due to haplotype differences. First, all contigs placed in the assembly are aligned to the surrounding sequence. Then, those contigs that have strong similarity to nearby regions – apparent segmental duplications – are analyzed using the methods described below to determine if they are misassembled. The analysis examines the mate pairs of the reads contained in these contigs to determine whether the assembly would be more consistent if the apparent duplicates were merged together.

The pipeline requires as input the contig sequences, an AGP file or other description of the placement of contigs along the chromosomes, placements of reads within the contigs, and mate pair and library information for the sequencing reads. In our experiments, ancillary read data was downloaded from the NCBI ftp site. Contig sequences, AGP files, and read placement information were downloaded from the ftp sites of the Genome Center at Washington University in St. Louis for chimpanzee and chicken, the Broad Institute for dog, and the Center for Bioinformatics and Computational Biology at the University of Maryland for cow.

3.4.1 Detection of potential haplotype mis-assemblies

Haplotype sequence that is placed twice in the assembly will have one of two signatures depending on how the flanking homozygous sequence (that is merged by the assembler) is placed. One possibility, illustrated in Figure 3.1(i), is that a long contig contains heterozygous sequence surrounded by homozygous sequence on both sides and another shorter contig contains only the heterozygous sequence. In this case, the shorter contig will align in its entirety to the heterozygous region in the longer one. Another possibility, shown in Figure 3.1(iii), is that both contigs contain matching heterozygous sequence as well as homozygous sequence on opposite ends. Here, the contigs will align only at their heterozygous ends. We call these cases mis-assembled duplicated contained contigs (DCCs) and mis-assembled duplicated overlapping contigs (DOCs) respectively. We restrict our analysis to duplications on separate contigs. Duplications also occur within a single contig, but these are rarely mis-assembled single copy sequence because the overlap graph of reads must have contained an unambiguous path through the two putative copies. Intra-contig mis-assemblies can be detected by other means, such as by computing the compression-expansion statistic across the contig [107].

Detection of DCCs and DOCs requires first finding the alignments. We aligned every contig to other contigs within 50 kilobases (Kb) using the MUMmer program [73]. We chose 50 Kb because this distance includes all common fragment insert sizes for the four genomes in our study. (Longer inserts based on bacterial artificial chromosomes were used in some projects, but they represented a small fraction of the sequence data.) In theory, a smaller distance might suffice, but our strategy was to identify a superset of possible erroneous duplications and filter the results in subsequent steps. Alignments that cover >93% of the contig's length at >95% identity are saved as DCCs. Alignments of size >300 base pairs (bp) and >95% identity that are consistent with the layout of DOCs and within 300 bp of the ends of both contigs are considered as DOCs. Again these parameters were chosen conservatively to allow more cases to be examined for mate pair consistency. Lowering them any further resulted in few extra alignments, which then passed the mate pair tests at a sufficiently decreased rate to cause concerns of false positives. Most examples found tended towards the ideal problematic case, e.g. 11113 of 13576 (82%) DOCs in chimpanzee had alignments within 10 bp of the ends of the contigs. DOC alignments were further filtered to only consider cases where the contigs are placed adjacently on the chromosome or there is a single contig in between that was classified as a mis-assembled DCC by the tests described below.

3.4.2 Analysis of mate pairs

These contigs, which align closely to a nearby location in the genome, were then analyzed further using the mate pairs of their reads to determine if they are true segmental duplications or two divergent haploid copies of the same chromosome region. A pair of mated reads is generated by sequencing both ends of a long fragment of DNA. The size of this fragment determines the distance we expect between the mated reads in the assembly. If a contig is truly duplicated, then the distances between mate pairs of relevant reads should better match their fragment sizes when the contig is in its current location in the assembly. But if the contig represents an erroneous duplication, we expect a better match when the contig is merged with the nearby copy. See Figure 3.1 for an illustration.

Within a library of reads, the fragment size is intended to fall within a tight distribution. The NCBI Trace Archive assumes that the distribution of fragment sizes within a library is normal and allows for sequencing centers to submit a mean and standard deviation for the fragment size of every read. However, this is an approximation (see Figure 3.5) and the real distribution may be considerably skewed from normal. Therefore we empirically measure the distribution of fragment sizes from the other reads placed in the assembly, thus alleviating the need to make any potentially biased assumptions. Though every assembly has its problems, a large majority of the sequence will be very accurate, and the vast majority of mated reads will be placed accurately with respect to each other. For each library, we find all mate pairs placed in the assembly, measure the distance between their 5' ends, and construct a histogram of the insert size distribution using a cubic smoothing spline function to alleviate noise (as implemented with smooth.spline in R with default parameters [84]). This nonparametric regression of the data does not assume a model distribution. When there are ample mated reads in the library, the result is a very accurate measurement of the distribution of fragment sizes, but not all

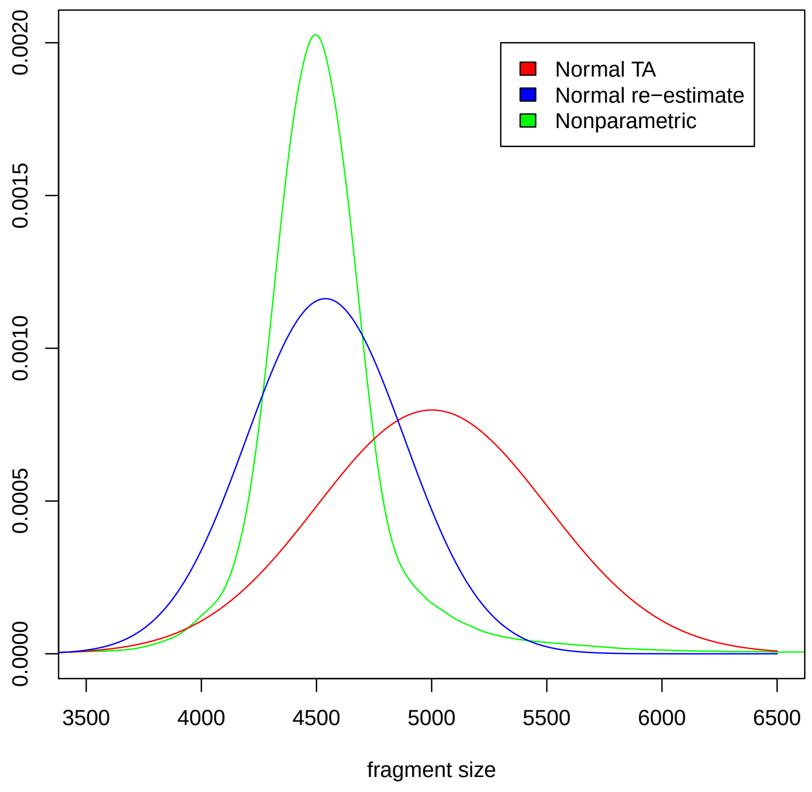


Figure 3.5: Re-estimated fragment size distribution. The distribution of fragment sizes for chimpanzee library G591P4 is plotted above under three models. The normal distribution with mean and standard deviation given by the NCBI Trace Archive is plotted as “Normal TA”. A normal distribution re-estimated from the placement of mated reads from the library is plotted as “Normal re-estimate”. To lessen the effect of outliers, we did an initial estimation of the parameters, filtered out any mate pair distances that were greater than 4 standard deviations away, and then estimated the parameters again. “Nonparametric” plots the actual density of mate pair distances after running a cubic smoothing spline. The actual fragment distribution has a mean of 4500 rather than the 5000 listed in the Trace Archive and is far tighter around the mean than suggested by the other models. In particular, the “Normal TA” model would have given us a very inaccurate view of this library, which is one of the largest for chimpanzee with over 2.3 million reads.

libraries contain a sufficient number of reads. Therefore, for each library, we compute a Kolmogorov-Smirnov goodness of fit test of the fragment sizes implied by the library's mated reads against the normal distribution with parameters given by the Trace Archive. If we can reject the null hypothesis that the distributions are the same with a p-value <0.01, we perform the re-estimation procedure above. If not, which will be the case if there are too few reads, we keep the normal distribution model.

For each contig, we determined the chromosomal location of each of its relevant reads and their mates. For a DCC, all reads in the contig with a mate pair outside of the contig are relevant. For DOCs, only reads with mate links that cross the overlap are relevant. Mated reads pointing away from the overlap are assumed to have had a significant enough influence in determining the size of the adjacent gap that these gaps, as well as the mate pair distances for reads crossing them, should remain unchanged. We consider reads with mates in both directions for DCCs because they are generally smaller and less influential in determining the size of surrounding gaps and the contigs tend to be considered for more distant and complicating moves than the DOCs. Both of these methods are imperfect, and ideally we would completely re-scaffold the region (i.e. position contigs and recompute gaps) and re-map it back to the chromosome. However, we do not attempt this at this time because different assembly projects may use many different mapping data types with specialized requirements. Nevertheless, our methods capture the most important information in the region's mated reads without having to resort to such a complicated extreme.

Given the library distributions and positions of the relevant mates, we can compute the likelihood of the insert sizes at the current contig position and the alternative, merged location. Each pair of mates is assumed to be independent, and thus the likelihood of contig c in chromosomal location l is given by

$$L(c, l) = \prod_{r \in reads(c)} P(frag(r, l) | lib(r)) \tag{3.1}$$

Here $reads(c)$ is the set of relevant reads for c, $frag(r, l)$ is the fragment size implied by read r and its mate in location l, and $lib(r)$ is the fragment distribution model for r's library. If the library has been re-estimated, the function is given by the smoothed frequency function. If not, the probability is given by the probability density function of the normal distribution with the Trace Archive parameters. Though density functions are reserved for continuous distributions, it serves as an approximation of discretizing the continuous normal distribution to integer values. A final complication is that we force a library-specific minimum value on the probability of any given fragment size. Doing so prevents highly improbable distant fragment sizes from dominating the likelihood comparison and allows us to include disoriented mate pairs (e.g. reads pointing away from each other) in the likelihood by giving them the minimum value. The minimum value was set such that the cumulative probability of all fragment sizes with probability less than the minimum value (not including far distant outliers) was .0001. For the normal distribution, this corresponds to an interval of $\sim$4 standard deviations.

For each contig that has been flagged as a DCC or DOC, we compute the likelihood function defined above at its original location and at the location suggested by its alignment to a nearby contig. If the likelihood is greater at the new location, then the mate pairs suggest that location is more appropriate for the contig and its reads. We classify such contigs as mis-assembled erroneous duplications.

3.4.3 Unplaced contigs

In addition to the contigs placed on the chromosomes, each of the four genome assemblies in this study contains a set of contigs that could not be placed. We used a similar procedure to find unplaced contigs that are likely to be haplotype variants of sequence that was placed. A stricter set of criteria was used to classify an unplaced contig as a haplotype variant, because unlike placed contigs, these contigs cannot be localized to a chromosome region. For each genome, all unplaced contigs were aligned with MUMmer to all placed contigs. An alignment of 96% identity spanning 94% of the length of the unplaced contig was required to consider it as a DCC and an alignment of 96% identity spanning 400 bp was required to consider it as a DOC. Contigs were classified as haplotype variants if at least two mate pairs were consistent and at least 30% of the mate pairs with a mate outside of the contig were consistent. Here consistent was defined as having an implied fragment length for which the probability is greater than the minimum value, with the minimum value set as above but eliminating 0.05 of cumulative probability (to correspond to being within $\sim$2 standard deviations for the normal distribution).

3.4.4 Haplotype polymorphisms

SNPs and indels are major contributing factors to the variation within a species and are highly sought after in the human genome [124]. Because two different copies of each chromosome are sequenced, genome sequencing projects are an incredible resource for finding SNPs and indels. However, when the homologous sequence from each chromosome is assembled into two separate contigs, this polymorphism information is lost. By detecting and correcting mis-assemblies that create erroneous duplications, we recover the information. During each of the respective genome projects, a consensus step to compute a multiple alignment of the reads was performed for each contig with the reads separated. For every pair of contigs designated as a false duplication by the procedure outlined, the mated reads suggest the contigs belong together. Thus, we recompute the consensus sequence with all reads covering the region using the Seq-Cons program [119]. On the new multiple alignment of reads, we implemented a Bayesian procedure to call SNPs and indels. However, to be conservative and eliminate the possibility of calling SNPs on mis-assembled repetitive sequence, we only count polymorphisms for pairs of contigs with read coverage indicative of a single-copy segment (negative A-statistic [15]).

At each position i in the new consensus sequence, we determine the most probable genotype G (e.g. AA if both chromosomes have adenine, AG if one chromosome has adenine and the other has guanine). Given the column i of the multiple sequence alignment of reads and using Bayes rule, we write the probability as

$$P(G_i|reads) = \frac{P(reads|G_i)P(G_i)}{P(reads)} \tag{3.2}$$

We need only concern ourselves with the numerator since the denominator is the same for every genotype. For each genome, we searched the literature for an appropriate estimate of the rate of polymorphism to use as the prior probability of a SNP. Because a sequenced individual is likely to be biased towards less polymorphism due to inbreeding, we err on the conservative side. Estimates range from 0.13-0.17% for chimpanzee, [134, 135] so we choose 0.1%. For cow, the rates for a number of breeds were recently estimated at 0.14-0.27% so we again use 0.1% [136]. The best estimates for chicken resulted from comparing domestic breeds to the wild reference genome (0.5%) and domestic breeds to each other (0.4-0.5%) [127]. To account for this inexact estimate and significant inbreeding, we conservatively set the prior probability of a SNP in chicken to 0.2%. To demonstrate the robustness of the SNP counts to these numbers, we also report statistics using prior probabilities that are 50% less than the chosen values in Table 3.3.

We set the prior probability of a homozygous genotype to the frequency of that base in the rest of the assembled sequence multiplied by one minus the SNP rate. We set the prior probability of a heterozygous genotype to the proportion of SNPs with that pair in the public SNP database dbSNP [120] for that organism multiplied by the SNP rate.

Let each read r extend to cover the entire consensus sequence and contain

	chimpanzee	chicken	cow
placed SNPs	67529	106691	20165
placed SNPs, prior/2	61550	104765	19172
placed SNPs, NQS	44884	90392	12515
placed SNPs, prior/2, NQS	43278	87940	11974
unplaced SNPs	98446	114840	48020
unplaced SNPs, prior/2	95842	112888	46628
unplaced SNPs, NQS	79548	98225	37694
unplaced SNPs, prior/2, NQS	77592	95929	37046

Table 3.3: Number of SNPs found from recomputing the multiple alignment of reads for haplotype variant contigs using different criteria. The prior probability of a SNP is set to 0.001 for chimpanzee and cow and 0.002 for chicken based on estimates from the literature. To demonstrate the robustness of the counts to the prior, we also count the number of SNPs at a prior that is half of the estimates (0.0005 for chimpanzee and cow and 0.001 for chicken). We also filter SNPs using a method similar to the widely used Neighborhood Quality Standard (NQS).

null characters except where the read aligns. Thus, r_i refers to the position in the read that aligns to the i^{th} position in the consensus. Let p_i be the probability that the base called at that position in the read is correct, which is determined by the quality value. Finally, let $reads(i)$ refer to the set of reads that cover position i in the consensus. Then the probability of the reads given a homozygous genotype at i is

$$P(reads|G_i = g_1g_1) = \prod_{r \in reads(i)} 1\{r_i = g_1\}p_i + 1\{r_i \neq g_1\}(1 - p_i)/3 \tag{3.3}$$

The first term corresponds to an accurate base call and the second term corresponds to a sequencing error, under the simplifying assumption that the error base call will be each of the other three bases with equal probability. The probability of the reads given a heterozygous genotype at i is

$$P(reads|G_i = g_1g_2) = \prod_{r \in reads(i)} 1\{r_i = g_1, g_2\}\frac{p_i + (1 - p_i)/3}{2} + 1\{r_i \neq g_1, g_2\}(1 - p_i)/3 \tag{3.4}$$

The first term corresponds to a base call that matches one of the bases of the genotype. Within the first term, r_i could have arisen via an accurate base call or an error from the other possible base. For example, if the genotype is AC, an observed A could have arisen from sequencing the A chromosome accurately or the C chromosome inaccurately. The last term represents a sequencing error away from either base of the genotype.

We calculate the conditional probability of each genotype given the reads and choose the most probable genotype at every position. If the genotype is heterozygous, a potential SNP is reported.

Following prior work on resequencing studies to discover SNPs, we filter the heterozygous sites for high quality surrounding sequence. The Neighborhood Quality Standard (NQS), calls for the base pair at which the SNP is called to have quality value $\geq$20 and the neighboring 5 base pairs on each side to have quality value $\geq$15 [126]. Though we are dealing with the reads from the original assembly, we can apply a similar filter by calculating quality values for each position based on the conditional probabilities of genotypes and requiring the most probable genotype to meet the NQS.

Indels are more difficult to model probabilistically. Instead, we report an indel for every column in the multiple sequence alignment where at least 3 reads have a gap and at least 3 reads have sequence. Continuous stretches of indel columns are merged into a single indel event.

3.5 List of abbreviations

WGS: whole-genome shotgun. SNP: single nucleotide polymorphism. Bp: base pairs. Kb: kilobases. Mb: megabases. DCC: duplicated contained contig. DOC: duplicated overlapping contig.

3.6 Acknowledgements

The authors thank Michael Schatz and Adam Phillippy for helpful discussion and comments on the manuscript. This work was supported in part by the National Institutes of Health under grants R01-LM006845 to SLS.

Chapter 4

Clustering metagenomic sequences with interpolated Markov models

The traditional method of sequencing a microbe's genome involves isolating the organism in culture, but a large number of interesting free-living microbes cannot be cultured. Shotgun sequencing of environmental DNA is one way to obtain genomic sequence from these organisms. However, the sequencing reads produced by this type of metagenomics experiment require more computational processing to determine what organisms were found in the mixture. Supervised classification approaches have proven useful for some well-studied environments, but lacking when known reference genomes are not close matches for the sequences obtained from the sample.

This chapter describes work done with Steven Salzberg to develop a composition-based unsupervised clustering approach to determining the relationships between metagenomic sequences. Our method, called SCIMM, represents clusters by interpolated Markov models and then optimizes the clustering objective function with a variant of the iterative k-means algorithm. SCIMM clusters simulated metagenomic reads more accurately than previous unsupervised approaches. The following manuscript was published in November 2010 in BMC Bioinformatics [137].

4.1 Background

Over the last 15 years, DNA sequencing technologies have advanced rapidly, allowing sequencing of over one thousand microbial genomes [138]. Still, this accounts for only a sliver of the fantastic diversity of microbes on the planet [139]. Sequencing of environmental DNA (often called metagenomics) has shown tremendous potential to drive the discovery and understanding of the "unculturable majority" of species — the vast number of unknown microbes that cannot be cultured in the laboratory [140]. Successful metagenomics projects have sequenced DNA from ocean water sampled from around the world [43], microbial communities in and on humans [141–144], and acid drainage from an abandoned mine [41]. These and many other projects (e.g. [145–147]) promise to uncover the true extent of microbial diversity and give us a better understanding of how these unknown microbes live.

However, progress has been slowed by the difficulty of analysis of metagenomic data. The output from an environmental shotgun sequencing project is a large set of DNA sequence "reads" of unknown origin. Because these reads come from a diverse population of microbial strains, assembly produces a large collection of small contigs (contiguous stretches of unambiguously overlapping reads) [44,148]. Two important goals of metagenomics are to determine what species are in the mixture in what proportions and to assemble substantial portions of individual genomes. A fragmented assembly of short sequences makes attaining these goals difficult. Advances in computational analysis techniques are essential to move the field forward.

To uncover what microbes are in a metagenomic sample, we must determine

(1) which sequencing reads came from the same microbial strain, and (2) where those strains fit into the phylogenetic tree of life [149]. Methods to solve these two problems are related. Clustering methods solve the former problem by binning sequences into clusters that represent a single taxonomic class. Classification methods aim to solve the latter problem by assigning a specific taxonomic class to every sequence.

In some cases, the presence of marker genes like 16S rRNA, which is very highly conserved across species but has variable regions, can be used to assign a taxonomic class to sequence fragments [150, 151], but this typically pertains to only a very small percentage of the reads. For example, ~0.1% of reads in a typical metagenomics project carry rRNA genes [149]. More general sequence similarity-based methods align reads with BLAST [152] to known genomes deposited in public databases like GenBank [153] and use those alignments to assign a taxonomic classification [154–156]. However, sequence alignment can only classify reads from organisms with a close evolutionary relative that has already been sequenced [157]. In most environments, this will not be the case for many of the reads; e.g. 70% of Sargasso Sea reads had a BLAST hit using "extremely lenient" search parameters, and only 30% aligned for nearly their whole length [43].

Composition-based methods for clustering and classification use properties of the DNA sequence such as oligonucleotide frequencies. These "genome signatures" are influenced by a variety of factors including codon usage, DNA structure, replication and repair processes, and evolutionary pressures [158–160]. They are fairly constant within a genome [161–163] and can be useful for inferring phylogenies [164].

Crucially for the use of genome signatures for clustering and classification, they persist even in conserved [165] or horizontally transferred regions (after a sufficient period of time) [166] and remain diverse between species despite shared environmental pressures and interactions [167]. Composition-based classification methods typically train on the oligonucleotide frequencies of all known genomes, and then classify sequences using supervised machine learning such as kernelized nearest neighbor [168], support vector machines [169], self-organizing maps [170], and naive Bayesian classifiers [171]. Phymm, a recently developed composition-based approach developed in our group [172], trains interpolated Markov models (IMMs) on known genomes in order to classify sequences.

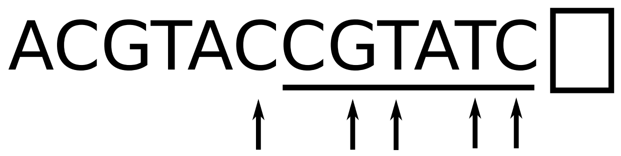
While supervised learning has proven useful in practice, shortcomings exist. Methods trained on the genomes in GenBank make an implicit assumption that those genomes are representative of microbes waiting to be found by metagenomics projects. This assumption is clearly violated by many if not most metagenomic samples. Supervised learning methods that tread carefully with respect to the potential biases caused by this assumption can still be useful analytical tools for many environments. Alternatively, genome signatures can be used for unsupervised clustering by learning the signatures from the set of sequences without the use of known genomes [167, 173–176]. Such approaches may be required when publicly available genomes are a poor fit to the data.

As an alternative to oligonucleotide frequencies, Markov chain models have shown great promise for characterizing genomic content [177], and have been implemented for both supervised classification [172] and unsupervised clustering [174] methods. In this chapter, we cluster sequences using interpolated Markov models (IMMs), a type of Markov chain model that adapts the model complexity to take advantage of variable amounts of training data. This strategy is well suited to metagenomics clustering problems, where the amount of sequencing performed and the relative abundances of the species in the mix can vary widely. Our clustering framework proceeds similarly to one used to cluster sequences using hidden Markov models where optimization is performed iteratively by a relative of the k-means clustering algorithm [178]. We refer to our method as SCIMM (Sequence Clustering with Interpolated Markov Models).

We test SCIMM on simulated metagenomic datasets of fragments from mixtures of randomly selected known genomes and demonstrate improvement on the performance of the metagenomic sequence clustering programs CompostBin [173] and LikelyBin [174]. We also assess the limitations of unsupervised learning on complex datasets, and describe how a combination of SCIMM and Phymm, which we call PHYSCIMM, clusters more accurately when useful training data is available.

4.2 Methods

Markov models have proven to be an invaluable tool for sequence analysis [179], including capturing genome signatures [177]. Here we present a clustering algorithm called SCIMM in which we use interpolated Markov models (IMMs) to model clusters of sequences. Clustering of sequences is performed using a variant of the k-means algorithm.



b	P(b)
A	.4
C	.2
G	.2
T	.1

Figure 4.1: In a standard w^{th}-order Markov chain model, the next base b in the DNA sequence is assigned a probability that is conditioned on the previous w bases (underlined above for $w = 6$). w should be chosen so that the data contains a sufficient number of instances of all 4^w substrings of length w. An IMM uses all of the Markov models from order 0 to w and computes the probability of the next base by interpolating among them. Our version of the IMM takes this a step further: rather than using the w immediately preceding positions, we use the most "informative" positions (shown above with arrows) of the previous w according to a recursive mutual information calculation.

4.2.1 Interpolated Markov models

A fixed-order Markov chain is a model for generating a sequence of outputs (in this case, nucleotides in a DNA molecule) in which the i^{th} element in the sequence has a distribution that is conditional on the previous w elements. Thus, given a sequence s and a model m, we can compute the probability that s was generated by m by walking along the sequence and multiplying the conditional probabilities.

$$P(s|m) = \prod_{i=w+1}^{|s|} P_m(s_i|s_{i-1}s_{i-2}...s_{i-w}) \tag{4.1}$$

Alternatively, IMMs are variable-order Markov chains, a strict generalization of fixed-order Markov chains, and interpolate between multiple models of fixed size via weights (also referred to as "model averaging"). Past work has found that increasing the order of the Markov model (e.g., using a 3rd-order model instead of a 2nd-order model) leads to more accurate predictions as long as there is sufficient training data. IMMs dynamically adjust the order of the models based on the data, which allows them to make the most of whatever information is available. This is particularly useful for clustering of metagenomic sequences where the amount of sequence from each species may differ widely due to differential abundance of organisms and the amount of sequencing performed on the sample. The variant of IMMs used in our system, introduced in the Glimmer 2.0 gene prediction software [180], is even more general as it allows the nucleotide distributions to be conditional on a subset of indexes in the preceding size w window (see Figure 4.1).

To train an IMM on a set of sequences, consider each $w+1$ sized window in the sequences and let the distribution of nucleotides at position i in the windows define random variable X_i. Training creates a probabilistic decision tree using information gain as the splitting criteria where each node specifies certain nucleotides at a subset of the window positions and defines a probability distribution for the final nucleotide in the windows. To construct this tree, first, compute the mutual information $I(X_i; X_{w+1})$ between the final position in the window and positions $i \in 1..w$. Define the initial split in the tree at the position with the greatest mutual information. Create branches to new nodes for all four nucleotides at this position. Next, perform a similar procedure for each branched node considering only windows containing the specific nucleotide at the position chosen. For these windows, compute the conditional mutual information of the remaining positions and choose the most informative position for the next split. Repeat this procedure to fill in the full decision tree, stopping early on paths where data becomes too sparse. At some point walking down each path, additional nucleotide positions may fail to be informative. We recognize this by computing a chi-square test between each node's distribution and its parent node's distribution. If the distributions are sufficiently similar, we stop branching and interpolate between the node and its parent's distributions, weighting each one based on the chi-square test result and the number of training windows mapped to the node.

To compute the likelihood that a novel sequence was generated by this IMM, consider each window of size $w + 1$ in the sequence as in Equation 4.1. For each window, follow a path through the decision tree to a leaf node according to the nucleotides at the positions defined by the nodes and branches. Score the next nucleotide in the novel sequence using the leaf node's interpolated probability distribution. More details of the training of IMMs and sequence likelihood computations can be found in the Glimmer descriptions [21, 180].

4.2.2 K-means clustering framework

The k-means algorithm is a widely used, simple and effective method for clustering data points. We review that algorithm before introducing our own approach to clustering sequences. Points are modeled as having come from k sources, each represented by a cluster mean. The algorithm begins by initializing these cluster means, e.g. by randomly choosing k data points. Next, one repeats the following steps. First, compute the distance between all points and the k cluster means. Second, assign each point to its nearest cluster. Finally, recompute the cluster means using the current assignment of points to clusters. After a number of iterations, one arrives at a stable partitioning of data points that approximates the minimum sum of squared distances between data points and their assigned cluster means.

An alternative formulation of the algorithm leads more directly to our approach. The k-means algorithm has also been referred to as Classification Expectation-Maximization (CEM) to optimize the Classification Maximum Likelihood (CML) criterion for data points generated from k Gaussian distributions with equal variance and zero covariance mixed in equal proportions [181]. For data points $x_1...x_n$ sampled from clusters $C_1...C_K$ and Gaussian density f parameterized by mean vectors $u_1...u_K$, CML is defined as

$$CML(C,u) = \sum_{k=1}^{K} \sum_{x_i \in C_k} log(f(x_i;u_k)) \tag{4.2}$$

That is, CML approximates the log likelihood that the cluster models generated the data points, but with each data point assigned a hard classification to a single cluster. CML can be further generalized to the case where data points are sampled from the clusters according to a multinomial distribution parameterized by $p_1...p_K$. Here CEM assigns each data point x_i to the cluster C_k that provides the greatest posterior probability $log(p_k f(x_i;u_k))$, and CML is defined as

$$CML(C,p,u) = \sum_{k=1}^{K} \sum_{x_i \in C_k} log(p_k f(x_i;u_k)) \tag{4.3}$$

Using CEM, the CML criterion converges to a local maximum [181].

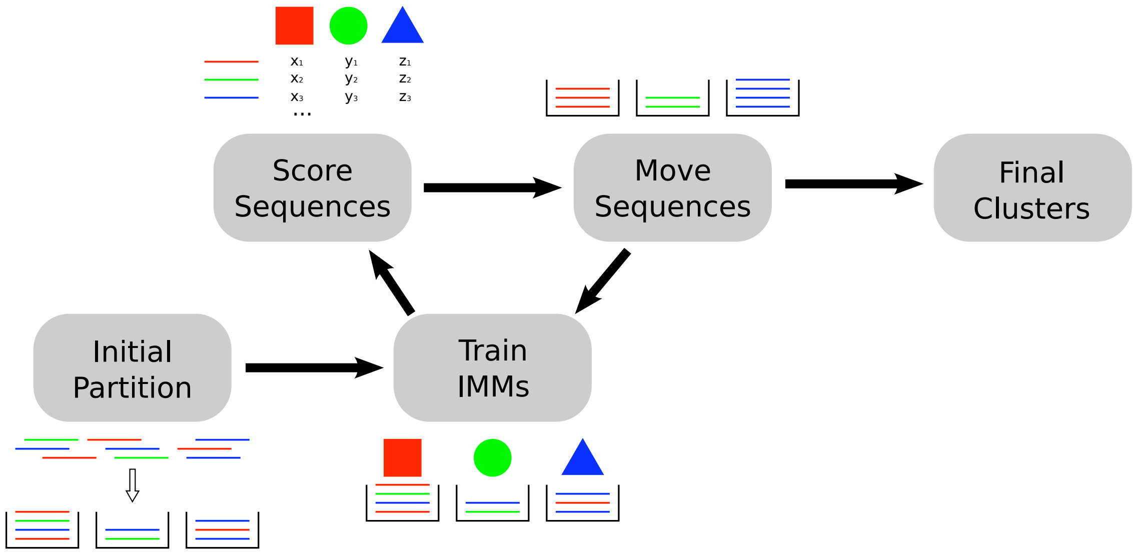


Figure 4.2: To initialize the IMMs, we initially partition a subset of the sequences into k clusters with a previously published method such as CompostBin [173] or LikelyBin [174]. We train an IMM on each cluster, and then compute the likelihood that each sequence was generated by each IMM for all sequences and all IMMs. Next, we reassign each sequence to the cluster corresponding to the IMM which generated it with greatest likelihood. If $> 0.1\%$ of the sequences changed clusters, we repeat the process. Otherwise we consider the clusters to be stable and halt.

4.2.3 Scimm

Scimm uses the same general algorithm as CEM, where the data points are DNA sequences and the cluster models are IMMs. Here the goal is to find the IMMs and multinomial probabilities that maximize the CML criterion, which approximates the log likelihood that the mixture of cluster models generated the sequences. The algorithm begins by initializing k IMMs (discussed in detail below). Then the following steps are repeated until convergence. First, for all sequences s and all IMMs m, compute the log likelihood that s was generated by m. Second, assign each sequence to the cluster corresponding to the IMM m that maximizes the posterior probability $log(p_m) + log(P(s|m))$. Finally, re-train the IMMs on the sequences currently assigned to their corresponding clusters. This loop is depicted in Figure 5.3. Over the course of the iterations, the IMMs converge to a set that should represent the phylogenetic sources.

Because the Maximization step is not straightforward maximum likelihood estimation (instead using IMM training's highly effective heuristics to choose a model order and interpolate between models), we lose the theoretical guarantee of CML convergence [181]. In practice, we did not find this to be a problem as the algorithm converged in all experiments. However, Scimm halts when fewer than 0.1% of the sequences change clusters in order to reduce computation time because the last few stages of this procedure tend to shuffle a small number of sequences with a negligible effect on clustering accuracy.

4.2.4 Initial partitioning

Scimm inherits the simplicity and effectiveness of the k-means algorithm, but also its sensitivity to initial conditions. We found that the likelihood landscape is riddled with local maxima from which the optimization cannot escape. Initially partitioning the sequences by very simple clustering algorithms yielded insufficient results.

To improve performance, we tried using previous methods for unsupervised clustering of metagenomic sequences to initialize the IMMs. We focused on two particularly successful approaches, LikelyBin and CompostBin. LikelyBin models sequences using k fixed 2nd-4th order Markov models learned by counting oligonucleotides [174]. Because LikelyBin uses simpler models with far fewer parameters than IMMs, a Markov chain Monte Carlo algorithm is used to search the parameter space for the parameters that maximize the likelihood of generating the sequences. LikelyBin is publicly available [182]. The second approach, CompostBin [173], works as follows. For each sequence, count oligonucleotide frequencies and project the frequency vectors into three dimensions using principal component analysis. Next, create a graph where each sequence is represented by a vertex and edges are placed between a sequence and its six nearest neighbors. Finally, split the sequences into two partitions by finding a minimum normalized cut in this graph across which few edges exist [183]. This process is repeated until the desired number of clusters is reached. Though CompostBin is publicly available [184], we re-implemented the main unsupervised ideas of the algorithm to better fit in our pipeline and refer to our version as CBCBCompostBin. One notable adjustment to the method was to make the number of nearest neighbors with which to build the graph a function of the number of sequences, because fewer sequences required a less connected graph for good performance. Choosing the number of nearest neighbors is a difficult subproblem of the normalized cut clustering method upon which CompostBin is based [183]. We found that the function $f(n) = 2 + \frac{1}{2}\lfloor \ln(n) \rfloor$, where f returns the number of nearest neighbors and n is the number of input sequences, worked well in practice, but did not address this problem in depth because experimental results demonstrated that SCIMM's accuracy did not depend significantly on the parameter choice.

To initialize the IMMs for SCIMM, we can run either LikelyBin or CBCB-CompostBin on a random subset of the sequences with a user-specified number of clusters k and train an IMM on every cluster returned. We used a random subset because both algorithms can be slow for large data sets, and 2-3 Mb of sequence was sufficient to train the IMMs to begin the iterative clustering procedure. Because the two programs approach sequence clustering differently, they tend to succeed on different datasets — e.g. for mixtures of 10 genomes, the standard deviation of the difference between LikelyBin's and CBCBCompostBin's precision (defined below) is 8.3% and recall is 6.5%. Therefore, we initially partition the sequences with both LikelyBin and CBCBCompostBin and perform one iteration of SCIMM on each. For each partitioning, we compute the CML criterion and continue iterating on only the partitioning with the greater value.

4.2.5 Supervised initial partitioning

As we will show, unsupervised clustering methods are very effective on low complexity datasets, but less accurate on metagenomic samples with many (e.g. >20) microbial strains. With more strains, the genome signatures may blend together and become difficult to properly discern. Alternatively, classification methods like Phymm are immune to the complexity of the dataset because each sequence is classified independently of the others [172]. Sequence classifications can be interpreted as implying a clustering, for instance by forming clusters from all sequences classified to the same genus. Therefore, a classification method can also be used to obtain an initial partitioning for Scimm.

We considered a hybrid of supervised and unsupervised learning referred to as PhyScimm where we obtained an initial partitioning of the sequences with Phymm. First, we randomly chose a subset of sequences (again due to computation time concerns and the sufficiency of a subset), classified the sequences, and clustered at a certain taxonomic level (family in our tests). Due to misclassification noise, Phymm will usually return too many clusters. To filter out clusters of misclassified sequences, we found only keeping clusters containing $> \frac{20}{k}\%$ of the sequenced bases where k is the number of genomes in the mixture (e.g. $> 1\%$ for 20 genomes) to be a useful heuristic. Note that Phymm is not limited to returning k clusters, and the number of clusters returned depends on the strictness of filtering, which the user would need to specify in a novel environment. After filtering clusters, we moved all unclustered sequences to an additional cluster, otherwise Scimm tended to incorrectly force these sequences into the generally high quality clusters from Phymm classifications. Finally, we iterated IMM clustering as in the standard Scimm algorithm.

Scimm and PhyScimm are available open source from [185] under the Perl Artistic License [70].

4.3 Results and Discussion

4.3.1 Simulated reads

To assess the performance of Scimm and PhyScimm, we simulated sequencing reads from mixtures of 1028 sequenced genomes in GenBank [153] as of 2009 and clustered the reads with each method. The degree to which the diversity of a random mixture of these genomes is representative of a real metagenomic environment has not been explored in depth. We make two points in support of this experimental setup. First, because certain model and disease-related organisms are of particular interest to researchers, GenBank contains many clusters of extremely closely-related genomes that make clustering difficult and may be representative of a heterogeneous species population from a real metagenomic environment; for example, 29 *Escherichia coli*, 16 *Salmonella enterica*, and 15 *Staphylococcus aureus* genomes were included. Second, while the expected clustering accuracy of any single method on a novel metagenomic environment may not exactly match the statistics reported in our tests, the simulations still serve to rank Scimm and previous unsupervised approaches based on clustering accuracy.

For each test, we randomly chose k genomes and k corresponding uniformly distributed random numbers in the interval (0,1). We simulated 30000 reads of length 800 base pairs (bp) so that the percentage of reads from each genome in the sample was proportional to that genome's random number. We clustered the reads with SCIMM, LikelyBin, and CBCBCompostBin. LikelyBin runs used 2 MCMC start points and a 3rd order Markov model. CBCBCompostBin runs used 5-mers.

Clustering accuracy can be quantified using a variety of measures [186]. Sequences from the same genome should be placed in the same cluster, which is measured by *recall*. Let c_{ij} be the number of sequenced nucleotides from genome j placed in cluster i. Then the recall for genome j is computed as

$$recall(j) = \frac{\max_i c_{ij}}{\sum_i c_{ij}} \tag{4.4}$$

Sequences placed in a cluster should belong to the same genome. This is measured as *precision* and computed for cluster i as

$$precision(i) = \frac{\max_j c_{ij}}{\sum_j c_{ij}} \tag{4.5}$$

In order to obtain global performance statistics, precision and recall were combined across clusters and genomes by weighting each term by the number of sequenced nucleotides from the cluster or genome. We also measured accuracy using the adjusted Rand index. The Rand index is the proportion of pairs of data points that are correctly placed together or apart, and the adjusted Rand index modifies this statistic based on the sizes of the clusters [187].

We tested the unsupervised methods with mixtures of 2, 5, 10, and 20 genomes, performing 40 trials of each, which resulted in standard deviations of ∼1.0% for precision and recall and ∼1.5% for adjusted Rand index. SCIMM achieved superior

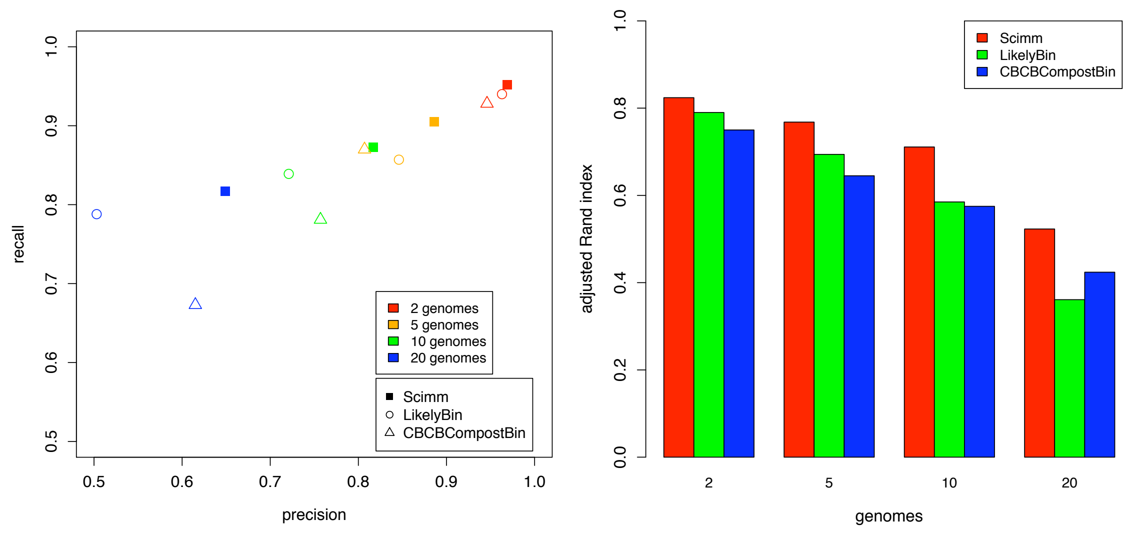


Figure 4.3: Cluster accuracy statistics for unsupervised methods on simulated 800 bp reads from random mixtures of genomes in random proportions. Scimm outperforms the other methods on all tests by all measures.

performance over the other methods by all measures, as shown in Figure 4.3. In addition to having a greater average adjusted Rand index, Scimm had the highest adjusted Rand index for 93% of the trials with ten genomes and 90% of the trials with twenty genomes. All methods were able to effectively partition sequences from two genomes. As we increased the number of genomes, performance degraded, but recall and precision >80% on average can be expected for mixtures of up to ten genomes.

We also examined the effect of sequence length on Scimm's performance (see Table 4.1) by sampling mixtures of five and ten genomes and varying the length of the simulated reads while holding the total number of sequenced bp constant; e.g. doubling the number of reads while halving the sequence length. Second generation sequencing technology from Roche/454 produces 400 bp reads, which should be

Length	Genomes	Recall	Precision	Adj Rand
400	5	0.858	0.863	0.689
800	5	0.905	0.886	0.768
1600	5	0.936	0.905	0.817
400	10	0.801	0.756	0.606
800	10	0.869	0.810	0.696
1600	10	0.911	0.821	0.738

Table 4.1: In tests with mixtures of five and ten random genomes in random proportions, increasing read length leads to greater accuracy. Even with 400 bp reads, the clusters are accurate enough to be useful for some applications.

sufficient for clustering sequences from low complexity environments with five or fewer strains as precision and recall are >85%. Accuracy continues to improve with 1600 bp fragments in both the five and ten genome tests, suggesting that longer read lengths or assembly of reads into contigs should be beneficial to metagenomic analysis.

All computational methods working with DNA sequencing reads must account for sequencing errors. We expect IMMs to be robust to such errors. A mis-sequenced nucleotide may affect the probabilities of up to $w+1$ nucleotides for window size w. However, the IMM will learn which positions in the window are informative for the distribution of the next nucleotide, and errors at uninformative nucleotides will have a negligible effect. Furthermore, even at what are considered high error rates, sequencing errors are rare enough to not overwhelm the genome signatures found in the sequences. To measure the effect of sequencing errors, we sampled

Error rate	Recall	Precision	Adj Rand
0.000	0.869	0.810	0.696
0.005	0.852	0.794	0.672
0.010	0.856	0.780	0.665
0.020	0.861	0.782	0.668

Table 4.2: In tests with mixtures of ten random genomes in random proportions, sequencing errors lead to decreased accuracy. However, the rate at which accuracy decreases as errors increases is slow so that SCIMM is fairly robust to error rates of 2.0%.

mixtures of ten genomes and mis-called nucleotides in the reads at rates of 0.5%, 1.0%, and 2.0%. Table 4.2 summarizes 40 iterations of this test, such that the standard deviations of precision and recall are ∼1.0% and adjusted Rand index is ∼1.5%. Though clustering accuracy decreases slightly with errors, increasing the error rate further has a negligible effect, and altogether SCIMM appears to be fairly robust to sequencing errors.

Unsupervised clustering performance degrades as the number of genomes reaches twenty or more, but classification methods like Phymm are largely unaffected by the number of genomes. We re-ran the experiment above using PHYSCIMM for mixtures of 5, 10, and 20 genomes. In order to thoroughly evaluate the performance of this supervised initial partitioning of the sequences, we performed separate tests of PHYSCIMM where Phymm's trained IMMs were held out if they were based on the same strain, species, and genus classification as the genomes from which the reads were simulated. For example, if we held out IMMs at the genus level, no IMMs were

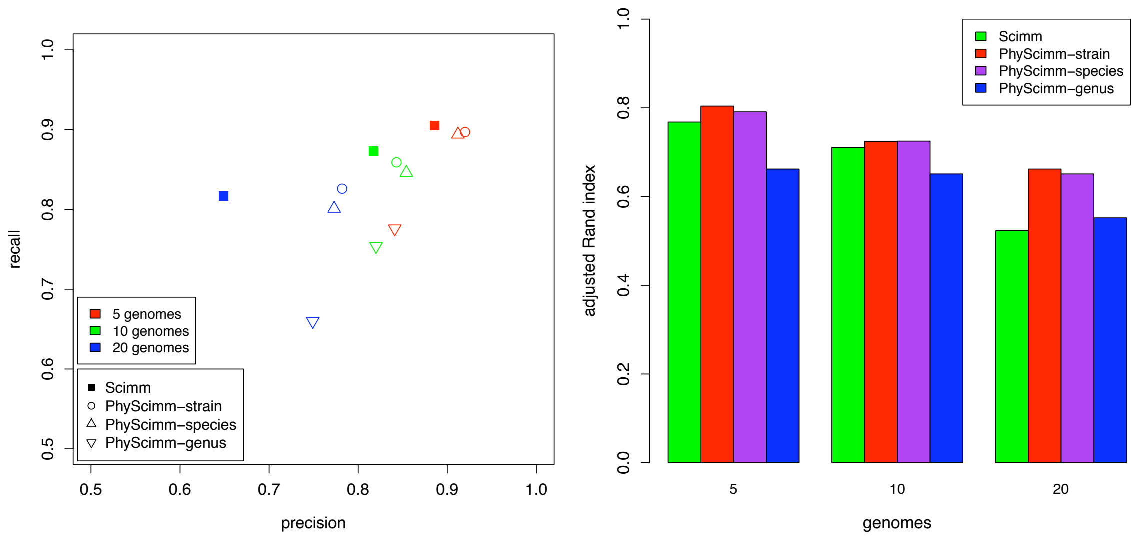


Figure 4.4: Cluster accuracy statistics for PhyScimm on simulated 800 bp reads from random mixtures of genomes in random proportions. PhyScimm-strain indicates that IMMs were held out if they matched the strain of a genome in the sample; similarly with PhyScimm-species and PhyScimm-genus. PhyScimm outpeforms Scimm in terms of precision unless IMMs are held out at the genus level and the mixture contains ten or fewer genomes.

used from microbial strains matching the genus of any of the genomes from which the reads were simulated. When IMMs from the same genus as those in the sample can be expected, PhyScimm produces accurate clusters (see Figure 4.4). But performance suffers when IMMs are unavailable from the same genus, and unsupervised clustering appears to be more useful in this case at ten and fewer genomes. With few genomes, accuracy is comparable to unsupervised Scimm, but the value of PhyScimm is readily apparent on the twenty genome mixture representing a more diverse metagenomic sample where performance is better than Scimm even when IMMs are held out at the genus level.

4.3.2 FAMeS

Experiments clustering single datasets can teach us about specific strengths and weaknesses of the methods and how they can be applied most effectively. To use more realistic data, we clustered the Arachne-assembled contigs from the FAMeS simulated metagenomic datasets of low (simLC) and medium (simMC) complexity [148]. These were created by mixing real reads from the original sequencing projects of 113 organisms. The contigs are dominated by a few species, but have a long tail of very low abundance species. We clustered with Scimm using $k = 2$–6 clusters and with PhyScimm initializing clusters from genus level classifications assigned to >1% of the total bp. Because Phymm has trained IMMs for these publicly available genomes, we held out IMMs similar to organisms in the mixture at the strain and species levels. In a noisy dataset with many organisms like this one, sequences from different strains of the same species are effectively indistinguishable. Thus, we computed accuracy at the species level for the tests that follow.

The simLC dataset contains 2362 contigs of mean size 3417 bp from 47 different microbial strains, but is dominated by 1283 contigs from *Rhodopseudomonas palustris HaA2* that make up 73.8% of the nucleotides and 617 contigs from *Bradyrhizobium sp. BTAi1* that make up 16.3%. The clustering accuracy statistics depend significantly on the arrangement of contigs from these two strains. Because *Rhodopseudomonas palustris HaA2* and *Bradyrhizobium sp. BTAi1* are both from the family *Bradyrhizobiaceae* and have similar high GC content (66.0% and 64.9%), separating each strain into its own cluster is difficult. Figure 4.5 displays the results for both methods. When clustering with Scimm at $k = 2$ and 3, nearly all contigs from the same strain were kept together leading to 99% recall, but each cluster contained a mix of species giving 80% precision. At larger values of k, some reads from the *Bradyrhizobiaceae* strains break off into other clusters, reducing the recall, though at the benefit of increased precision. When holding out IMMs from the same strains, PhyScimm achieved very high accuracy (95% precision and 94% recall), as *Rhodopseudomonas palustris HaA2* and *Bradyrhizobium sp. BTAi1* were mostly separated from each other because Phymm had a trained IMM for each species. When IMMs from genomes matching species in simLC were removed, PhyScimm's precision dropped to 83%. If the initial clusters are formed from Phymm classifications at the family level, the *Bradyrhizobiaceae* strains cannot be separated and precision also drops to 83%.

SimMC has 7307 contigs of mean size 2332 bp from 51 microbial strains. These

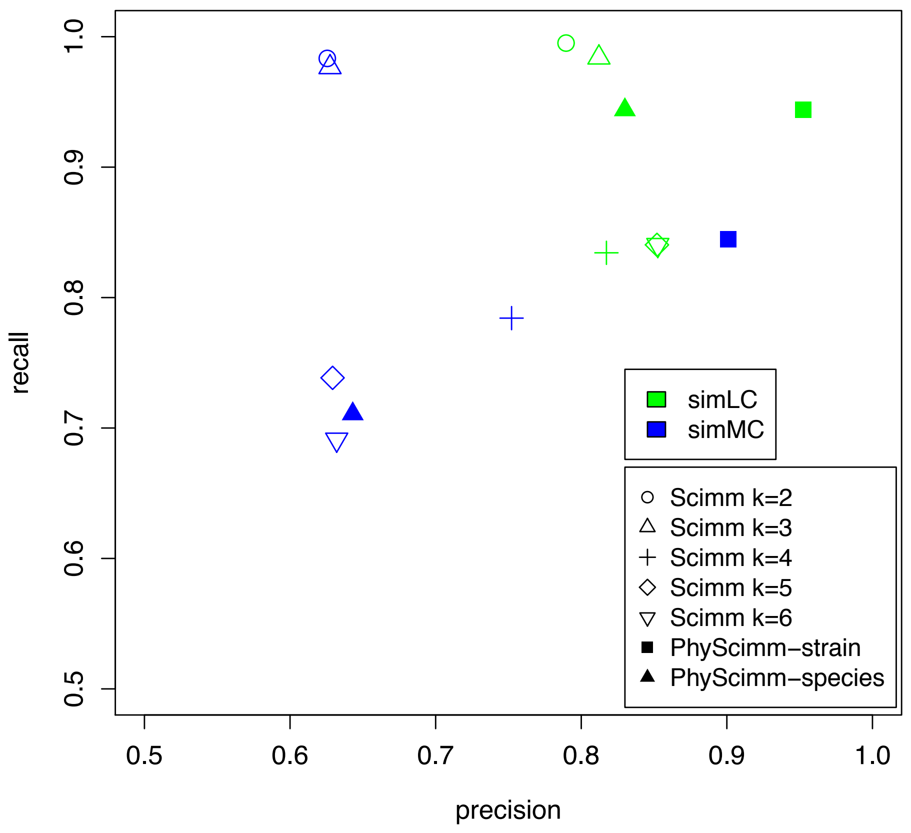


Figure 4.5: Cluster accuracy statistics for FAMeS Arachne-assembled contigs. We ran SCIMM at a range of values for k, the number of clusters, resulting in consistently high accuracy on the low complexity simLC dataset and variable accuracy on the medium complexity simMC for reasons discussed in the text. We ran PHYSCIMM, ignoring IMMs matching genomes in the mix at the strain and species levels (PHYSCIMM-strain and PHYSCIMM-species above). When some IMMs from the same species can be expected, accuracy is far greater, demonstrating that clustering with supervised help relies on models for similar genomes.

contigs are distributed among the strains slightly more uniformly, but still only six species account for 99.0% of the nucleotides. These six include two strains each from the species *Rhodopseudomonas palustris* and *Xylella fastidiosa*. *Bradyrhizobium sp. BTAi1* also appears and presents a challenge similar to that described above for simLC. For $k = 2$ and 3, SCIMM formed strong clusters for the the *Xylella fastidiosa* strains and the *Bradyrhizobiaceae* strains, leading to very high recall. As we increased k, these strains were split among the clusters, significantly decreasing the recall (see Figure 4.5). From this experiment and the last one, we see that clustering performance is best when k is set to the number of dominant phylogenetic sources. Increased values of k risk splitting a dominant species into multiple clusters rather than effectively clustering a far less abundant species. Precision did not increase with more clusters because when the *Bradyrhizobiaceae* strains split into multiple clusters, each one contained a mixture of both species. When IMMs from the same species were available, PHYSCIMM produced much better clusters with 90% precision and 85% recall. But, accuracy dropped precipitously when IMMs were held out at the species level. Thus, we see again that PHYSCIMM clusters more accurately if very related genomes are available for training, but pure unsupervised clustering is preferable for a metagenome containing organisms whose taxa are unsequenced.

Instead of computing accuracy at the species level, we could consider a higher level in the hierarchy of taxonomic classification, such as the family level. By doing so, we reward the clustering algorithm for clustering together two sequences that originated from different strains in the same family, such as *Bradyrhizobium sp. BTAi1* and the *Rhodopseudomonas palustris* strains. Family level precision is >97%

in all tests, meaning that generally when Scimm is merging two separate species into a cluster, they are phylogenetically related.

4.3.3 In vitro-simulated metagenome

To further explore the effectiveness of Scimm and PhyScimm on more realistic data, we clustered sequencing reads from an *in vitro*-simulated microbial community [188]. Here, ten microbes were mixed into a simulated metagenome and sequenced using a number of different protocols and sequencing techniques. These ten were chosen to cover a wide range of microbial diversity, but also to include closely related species, specifically two *Lactobacillus* strains and two *Lactococcus* strains. The resulting reads were then assigned to their source genome via BLAST [152] alignments to a database of the ten microbes' genomes. After combining classified reads from all non-454 datasets, we obtained 24410 mated reads and 3285 singleton reads.

We clustered the reads with Scimm into 8 clusters for the 8 species in the data and computed 87% recall and 88% precision at the species level. As expected, the *Lactobacillus* and *Lactococcus* strains each clustered together well. High quality clusters formed around the medium abundance strains *Shewanella amazonensis SB2B* and *Myxococcus xanthus DK 1622*, but Scimm split reads from the most abundant strain *Acidothermus cellulolyticus 11B* into mainly two clusters. Meanwhile, low abundance strains *Pediococcus pentosaceus ATCC 25745* and *Halobacterium sp. NRC-1* lacked the data to form their own pure clusters and co-clustered with the

Lactococcus strains and *Acidothermus cellulolyticus 11B* respectively. Knowing that Scimm can struggle with low abundance species and seeing that 2 of the 8 clusters were effectively unused and contained far fewer reads than the rest, we reduced the number of clusters to 6. Doing so brought the two *Acidothermus cellulolyticus 11B* clusters together and increased the recall to 94%.

Clustering the reads with PhyScimm led to further insight. The level at which IMMs were held out did not have a significant impact on clustering accuracy for this dataset, so we discuss the results from holding out IMMs from the same genus as the strains in the simulated metagenome. We initially clustered sequences using family classifications that were assigned to >3% of the sequences. Performance was considerably worse than unsupervised Scimm with a 79% recall and 83% precision on the 7 clusters. PhyScimm struggled with *Acidothermus cellulolyticus 11B* because there are no other trained IMMs in its family *Acidothermaceae*. Instead, Phymm assigned its reads to the families *Mycobacteriaceae*, *Microbacteriaceae*, and *Propionibacteriaceae*. Each of these families belong to the order *Actinomycetales*, and so PhyScimm performed far better when initialized using order classifications (6 clusters with 88% recall and 92% precision). Interestingly, Phymm misclassifies many reads from the *Lactococcus* strains, the *Lactobacillus* strains, and *Shewanella amazonensis SB2B* to the order *Enterobacteriales*, but iterative IMM clustering is able to effectively separate these species despite a poor initialization.

4.4 Conclusions

Determining the relationships between sequences is a crucial step in metagenomics analysis. In this chapter, we introduce Scimm, an unsupervised sequence clustering method based on interpolated Markov models (IMMs). Our experiments show that Scimm clusters sequences more accurately than previous unsupervised algorithms.

By demonstrating the ability of IMMs to successfully cluster sequences, we add to the growing evidence of the effectiveness of IMMs for modeling DNA sequences [23, 172]. Markov chain models have proven to be useful sequence modeling tools for many bioinformatics applications [179]. The increased modeling sophistication and ability to handle varying amounts of training data make IMMs preferable for many of these applications.

We compared two variations of clustering with IMMs. Scimm is purely unsupervised and makes use of the previously published methods LikelyBin and CompostBin to initially partition the sequences. PhyScimm partitions the sequences using supervised Phymm classifications before the unsupervised iterative IMM clustering stage. Supervised learning proved to be a valuable addition to the pipeline when genomes were available to train on from the same genus as the microbes in the mixture. Pure unsupervised learning is preferable when the available genomes to train on are not representative of those from which the sequencing reads originated. Because the classification accuracy of Phymm is independent of the complexity of the mixture, supervised learning also improves clustering of complex mixtures of twenty or more microbes. Developing more sophisticated combinations of classification and clustering methods may prove to be a fruitful line of research.

We believe SCIMM and PHYSCIMM will be valuable tools for researchers seeking to determine the relationships between sequencing reads from a metagenomics project. For environments with ten or fewer species, unsupervised clustering with SCIMM finds accurate clusters. However, the number of clusters k must be chosen carefully by the user. Specific knowledge about the environment, especially regarding the number of dominant microbes and their relationships to each other, can inform the choice of k and impact the utility of a clustering of the environment's sequences. Nevertheless, tests on the FAMeS dataset showed that various values of k can produce useful clusters. Lesser values of k tend to provide greater recall with lower precision. Greater values of k may decrease recall by dividing a particularly dominant species into more than one cluster but will usually improve precision.

When the microbes in an environment are thoroughly represented in public databases, PHYSCIMM finds even more accurate clusters. PHYSCIMM is also more effective for mixtures of twenty or more strains. The user does not need to choose the number of clusters with PHYSCIMM, but must choose the classification level and minimum support to initialize a cluster. The simulated read experiments offer guidelines for how to set these parameters effectively. Tests with the FAMeS and in vitro-simulated metagenome datasets demonstrated that incorporating knowledge about the dominant organisms in the environment can have a significant positive impact on the clusters.

Metagenomics projects are increasingly turning to less expensive and higher throughput second generation sequencing technologies such as those from Roche/454 and Illumina. Clustering of 400 bp read lengths is still reasonably effective for mixtures of five or fewer strains. Shorter reads, such as the 100-125 bp lengths currently available from Illumina, cannot be accurately clustered by our methods for environments with realistic complexity. However, if these reads can be assembled into larger contigs, then effective clustering of the contigs is possible.

A number of avenues appear worthwhile for further research. A principled method for setting parameters that affect the number of clusters would certainly aid researchers using the method. Preliminary sequencing of 16S rRNA or other marker genes followed by clustering may effectively achieve this goal [151, 189]. The k-means iterative clustering framework used by Scimm works well with a good initial partitioning of the sequences, but other optimization methods might prove more robust to the initial conditions and less prone to getting stuck in local maxima. Because Scimm nearly always improves on the clustering results of LikelyBin and CBCBCompostBin, there is reason to believe that it would also improve on initial clusters from more accurate future methods. We excluded an interesting feature from the original CompostBin in our experiments whereby reads containing informative marker genes were identified and classified using AMPHORA [151] and the classifications were used to add supplemental edges to the nearest neighbor graph. A similar semi-supervised scheme could be implemented in Scimm as well. Finally, assembly and clustering are both important steps in metagenomics pipelines, and further exploration of the relationship between the two has the potential to improve both tasks.

4.5 Acknowledgements

We thank Arthur Brady, Art Delcher, Mihai Pop, Carl Kingsford, Saket Navlakha, James White, and Adam Phillippy for valuable discussions on the method and the manuscript.

This work was supported in part by the National Institutes of Health grants R01-LM006845 and R01-LM083873 to SLS.

Chapter 5

Gene prediction with Glimmer for metagenomic sequences augmented by classification and clustering

Because environmental shotgun sequencing reads come from many populations of organisms, computational analysis of the data can be difficult, particularly assembly of the reads. Nevertheless, functional analysis of the metagenome is generally achievable by predicting genes on the contigs or raw reads. Much can be learned from the metagenome's genes, *e.g.*, by comparing them to protein databases to determine which have known homologues and which are novel.

Current methods for metagenomics gene prediction all use a simple method based on GC-content to address the major challenge– how should prediction models be parameterized for each new sequence? This chapter describes unpublished work with Bo Liu, Art Delcher, and Steven Salzberg to augment the prokaryotic gene finder Glimmer with classification and clustering of the sequences to parameterize prediction models. We also add a model to predict indel sequencing errors, which are prevalent and problematic in reads from the 454 sequencing technology. Glimmer-MG predicts genes more accurately on simulated and real datasets than all previous methods.

5.1 Introduction

Prokaryotic species inhabit an incredibly diverse array of environmental niches and account for most of the world's biomass [190–192]. They play an integral role in many ecosystems, including the human body in which a typical individual carries 10–100 times more prokaryotic cells than human cells [193]. The DNA sequences of these microorganisms provide us with important information about their identities, functions and evolution. The traditional method to obtain these sequences was to select a single microbe of interest, isolate it in culture, and sequence its genome to high coverage [2].

Because this process is costly and many microbes cannot be cultured, researchers have increasingly turned to sequencing DNA directly from environmental samples (often referred to as *metagenomics*) [40, 194]. Metagenomics has been shown to be an effective tool for exploring natural environments (*e.g.*, acid mine drainage [41], ocean water [43], and soil [195]) and environments on and within the human body [141]. With the development of improved sequencing technologies from companies like 454 Life Sciences and Illumina, DNA sequence reads can be obtained at increasingly higher throughput and lower cost. As these transformative technologies further develop, metagenomics will continue to be an important technique for analyzing genomes of entire communities of microbes in an ever-broadening collection of environments.

Identifying the protein-coding genes contained in the sequences is a fundamental step in any shotgun sequencing analysis, and metagenomics is no different.

If substantial segments of the genomes can be assembled from the data [**?**], then gene prediction helps to functionally annotate the genome. Even if the metagenome assembly is highly fragmented, many new and interesting genes may still be extracted [196]. Some metagenomic experiments aim to compare microbial communities in various environments [197, 198]. In these cases, accurate gene prediction allows a functional comparison [199, 200].

Because sequences coding for genes have statistical properties that differentiate them from noncoding sequences, computer software can be designed to select open reading frames (ORFs) that are more similar to known examples of coding than noncoding sequences based on statistical models [21, 22, 201]. Sequence composition is the most important discriminative feature due to the effects of natural selection on the DNA triplets coding for amino acids in a gene and can be captured by Markov-chain models. To predict genes on a novel genomic sequence, one would train these models either on a set of ORFs in the sequence that are highly likely to be genes or on a full set of genes from a close relative. State of the art methods currently achieve >99% sensitivity and high precision when predicting known genes [23] on full chromosome sequences. In metagenomics, we generally have only short sequence fragments of chromosomes, yet we still wish to find the genes contained in these data.

As environmental shotgun sequencing has become more prevalent, computational gene prediction approaches have adapted to the particular challenges of these data. The foremost problem is that, because the sequences represent an unlabeled sample from a mixture of organisms, training an appropriate prediction model for a given sequence is difficult. In addition, prediction must be performed on short sequence fragments that frequently capture only part of a gene. A finished genome assembly will have eliminated nearly all sequencing errors by computing a consensus base call using the many reads that cover a region. But, metagenomic assembly is more difficult and will result in many low-coverage contigs as well as unassembled singleton reads [44, 148], in which sequencing errors will be prevalent and problematic for gene prediction [202].

Despite these challenges, a number of methods for predicting genes in metagenomic sequences have been published, reporting varying degrees of success [203–206]. The GC-content of a genome provides a simple but effective way of parameterizing a predictive model such as a Markov model, and all of these previous methods incorporate GC-content into their prediction algorithms. MetaGeneAnnotator [203] also models the lengths of genes in order to appropriately score partial genes. FragGeneScan [206] allows frameshifts within protein coding regions, which allows it to tolerate insertion or deletion sequencing errors in the fragments.

For these programs, GC-content is a simple way to identify training genomes that are likely to be evolutionarily related, and whose genes might have similar sequence composition. This task is highly similar to taxonomic classification of sequences, which implicitly identifies close relatives, and for which much better statistics than simple GC-content have been developed [155, 168, 169, 172]. In this chapter, we explore the use of a more sophisticated classification scheme, based on the Phymm system [172], to parameterize gene prediction models for metagenomic sequences. Another related computational problem is unsupervised sequence clustering, in which the relationships between sequences are elucidated via a partition of the sequences into clusters, generally without the use of reference (or “training”) genomes [137, 173, 174]. Clusters have been shown to improve the identification of translation initiation sites [207]. Below, we use a more advanced clustering method Scimm [137] to assist in predicting genes via an unsupervised retraining step.

In previous work, our group has demonstrated that the Glimmer gene prediction program is highly effective, routinely identifying >99% of the genes in most complete prokaryotic genomes [23]. However, Glimmer was not designed for the highly fragmented, error-prone sequences that typify metagenomic sequencing projects today. In this chapter, we describe enhancements to Glimmer designed for metagenomic projects. First we describe several new algorithmic changes that improve Glimmer's precision and start-site prediction accuracy while maintaining its high sensitivity. We then describe a new software pipeline, Glimmer-MG, that incorporates classification and clustering of the sequences prior to gene prediction. Glimmer-MG achieves far greater accuracy than previous methods applied to the same metagenomic data. Glimmer-MG can also predict frameshifts resulting from insertions and deletions in short sequence fragments, and it achieves high prediction accuracy on simulated reads with errors. Finally, we demonstrate the Glimmer-MG pipeline on a set of real 454 reads from the human gut microbiome [208].

5.2 Methods

5.2.1 Glimmer

Glimmer's salient feature is its use of interpolated Markov models (IMMs) for modeling gene composition [21]. IMMs are variable-order Markov-chain models that maximize the model order for each specific oligonucleotide window based on the amount of training data available. IMMs then interpolate the nucleotide distributions between the maximum order and one greater. Thus, IMMs construct the most sophisticated composition model that the training data sequences support. To segment the sequence into coding and noncoding sequence, Glimmer uses a flexible open reading frame (ORF)-based framework that incorporates knowledge of how prokaryotic genes can overlap and upstream features of translation initiation sites (TIS) like the ribosomal binding site (RBS). Glimmer extracts every sufficiently long ORF from the sequence and scores it by the log-likelihood ratio of generating the ORF between models trained on coding versus noncoding sequence. The features included in the log-likelihood ratio are composition via the IMMs, RBS via a position weight matrix (PWM), and start codon usage. Then a dynamic programming algorithm finds the set of ORFs with maximum score subject to the constraint that genes cannot overlap for more than a certain threshold, *e.g.*, 50 bp.

5.2.2 Additional features

Glimmer is ineffective on metagenomic sequences because its gene composition model is trained under the assumption that the sequences all came from a single genome. Recent approaches both relax this assumption and add new features used to discriminate between coding and noncoding sequence. One approach called MetaGeneAnnotator (MGA) uses a similar framework to Glimmer by extracting ORFs, scoring them, and choosing a high scoring set using dynamic programming [203]. MGA incorporates additional gene features, of which we add three — ORF length, adjacent gene orientation, and adjacent gene distance — to Glimmer-MG. We describe how to compute models for these features assuming we have gene annotations to train on from a close evolutionary relative to our genome sequence of interest, an assumption that will be explained in more detail below.

To model ORF length, we seek probability distributions for the length of coding and noncoding ORFs. For the coding model, our sample data are the lengths of annotated genes in the training genome. For the noncoding model, the lengths of noncoding ORFs that meet a minimum length threshold (75 bp) and a maximum overlap threshold with a gene (30 bp) are considered. One can estimate the distributions using a nonparametric method based on the histogram of lengths or a parametric method where one assumes a well-studied probability distribution and computes the maximum likelihood parameters [87]. We use both methods to obtain our estimate. Where training data are plentiful, such as for common gene sizes, a nonparametric approach offers greater modeling specificity than any parameterized distribution. But when data are sparse, such as for very long ORFs, the nonparametric approach fails. For example, we cannot assign a useful probability to an ORF larger than any in our training set though it should obviously receive a large log-likelihood ratio score. A parameterized distribution can assign meaningful prob-

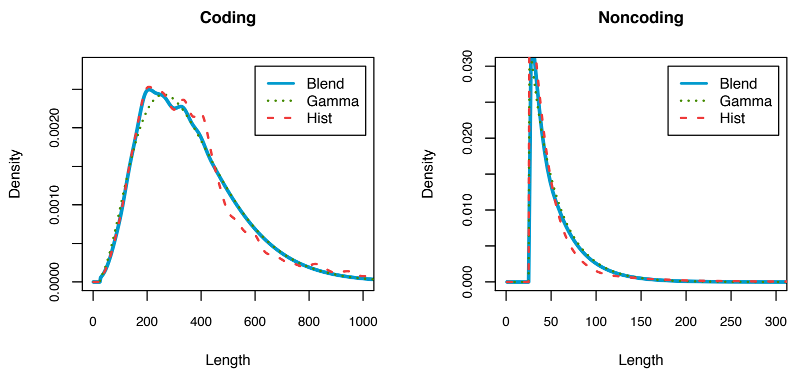


Figure 5.1: The distributions for coding and noncoding ORF lengths (in amino acids) from *Deinococcus radiodurans R1* estimated using the Gamma distribution "Gamma", a smoothed histogram "Hist", and a blend of the first two "Blend" that uses the histogram model for the first quartile, the Gamma model for the last quartile, and a linear combination in between. The "Hist" model offers greater specificity for short and medium sized ORFs, but is useless for very long ORFs, which "Gamma" can model more effectively.

abilities to any length ORF. This helps us choose the right start codon for large genes by properly rewarding the additional length provided by one start site versus another downstream. Thus, the length distributions are modeled by a histogram after kernel smoothing with a Gaussian kernel [87] for the first quartile (as determined by the raw counts), a Gamma distribution with maximum likelihood parameters for the last quartile, and a linear combination of the two in between with a linearly changing coefficient (*e.g.*, Figure 5.1).

ORFs truncated by the end of their fragments require adjustments to the length model. We know that the total length of a truncated ORF with X bp on a fragment is at least X and should therefore be scored higher than a full X bp ORF. We accomplish this by modeling the joint distribution of the length and the presence of start and stop codons (described below).

Features computed on pairs of adjacent genes also capture useful information. For example, genes are frequently arranged nearby in the same orientation to form transcriptional units called operons [25]. Alternatively, consecutive genes with opposing "head-to-head" orientations (where the 5' ends of the genes are adjacent) tend to be further apart to allow room for each gene's respective RBS. We added two features of adjacent genes: their orientation with respect to each other and the distance between them. Again, we need distributions for coding and noncoding ORFs to score a pair of adjacent genes by their log-likelihood ratio. The gene model uses all pairs of annotated genes. For the noncoding model, we consider noncoding ORFs that satisfy the length and overlap constraints with their adjacent annotated genes. For the distances, no parameterized distribution was a good fit for the data so we rely solely on a smoothed histogram. Because one gene's start codon often overlaps another gene's stop codon due to shared nucleotides, we do not smooth the histogram for distances implying overlapping start or stop codons.

5.2.3 Truncated length model

Log-likelihood ratio scores for gene length use two probability distributions that apply to coding and noncoding ORFs. To account for truncated ORFs, we jointly model the length with the presence of start and stop codons. In our experience, gene prediction accuracy, particularly for start sites, is highly dependent on an accurate formulation of this model. Our approach to constructing these probability distributions considers that, given detection of an ORF on the fragment, there are a finite number of places that the fragment could have landed in the genome. Thus, given the gene length and fragment length, we can compute the probability that we see the start codon and stop codon on the fragment.

Let $S(X, L, R)$ be the likelihood ratio for an ORF where X is the length and L and R are binary variables representing the presence of a start or stop codon at the left and right end of the fragment respectively. Also, let G be a binary variable representing whether the ORF is a coding or noncoding. Then for an ORF truncated at the left end with x nucleotides on the fragment, we score the ORF with the logarithm of

$$S(x, 0, 1) = \frac{P(X > x, L = 0, R = 1|G = 1)}{P(X > x, L = 0, R = 1|G = 0)} \tag{5.1}$$

Expanding this to specify the exact possible lengths gives

$$S(x, 0, 1) = \frac{\displaystyle\sum_{d=x+1} P(X = d, L = 0, R = 1|G = 1)}{\displaystyle\sum_{d=x+1} P(X = d, L = 0, R = 1|G = 0)} \tag{5.2}$$

Applying the definition of conditional probability gives

$$S(x, 0, 1) = \frac{\displaystyle\sum_{d=x+1} P(L = 0, R = 1|X = d, G = 1)P(X = d|G = 1)}{\displaystyle\sum_{d=x+1} P(L = 0, R = 1|X = d, G = 0)P(X = d|G = 0)} \tag{5.3}$$

The right term $P(X = d|G = g)$ is defined by our learned distributions for coding $(g = 1)$ and noncoding $(g = 0)$ ORFs. Given the ORF length, the probability that it is truncated on either side is independent of its coding potential so G can be

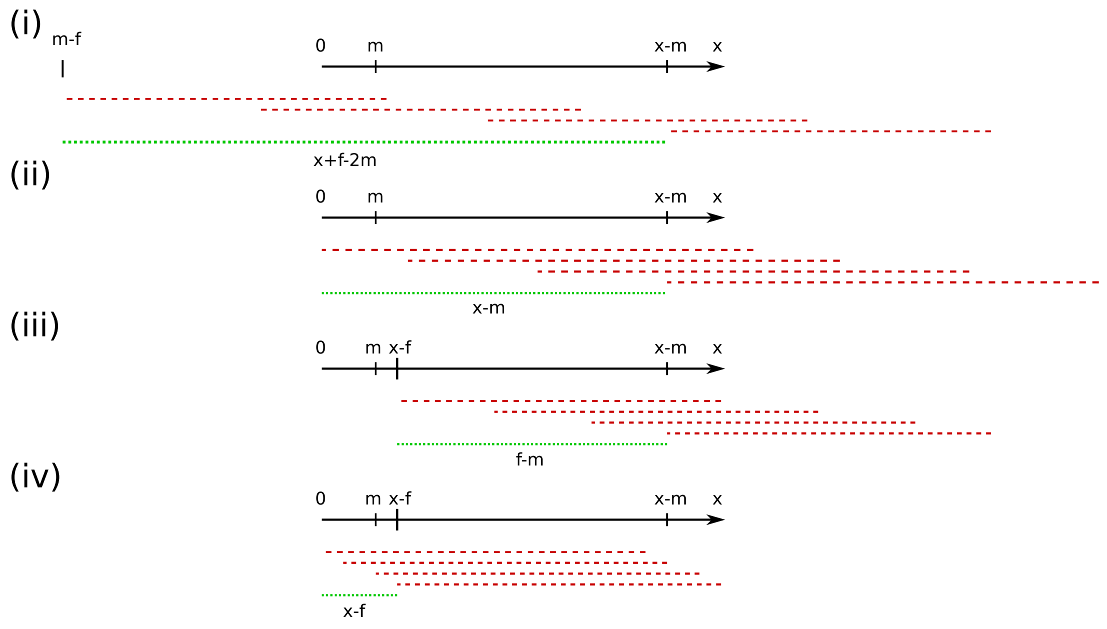


Figure 5.2: $P(L = l, R = r|X = x)$ scenarios. The solid black arrows show the gene, which is of length x and must be covered by a minimum gene length m to be detected. The dashed red lines show examples of reads with length f that would cover the gene with the L and R values specified. The dotted green lines show the region where a read can start and cover the gene with L and R values specified. (i) There are $x + f - 2m$ different read start points that would detect the ORF. (ii) If the read size f is greater than the gene size x, than $x - m$ of these start points lead to partial genes truncated on the right side. (iii) If f is less than x, there are $f - m$. (iv) $x - f$ lead to a partial gene truncated at both ends.

removed from the left term.

$$S(x,0,1) = \frac{\sum\limits_{d=x+1} P(L=0,R=1|X=d)P(X=d|G=1)}{\sum\limits_{d=x+1} P(L=0,R=1|X=d)P(X=d|G=0)} \tag{5.4}$$

We can define a probability distribution for start and stop codon presence by considering the genomic locations from which our fragment may have arisen given that we discovered the ORF. As shown in Figure 5.2(i), there are $x+f-2m$ positions where a fragment of length f could land to detect a minimum of m bp from a gene of length x. If $f > x$ as in Figure 5.2(ii), $x-m$ fragment positions truncate the ORF on one end. If $f < x$ as in Figure 5.2(iii), $f-m$ fragment positions truncate the ORF on one end. Thus, we can write the probability as

$$P(L=0,R=1|X=d) = \frac{\min(x,f)-m}{x+f-2m} \tag{5.5}$$

Because left and right truncation are symmetrical, $P(L=1,R=0|X=d)$ and thus $S(x,1,0)$ can be defined similarly. $S(x,0,0)$ corresponds to the case where both the left and right ends of the ORF are truncated. It can be defined using a similar series of steps where we see that $x-f$ fragment positions produce such an arrangement as in Figure 5.2(iv).

$$P(L=0,R=0|X=d) = \frac{x-f}{x+f-2m} \tag{5.6}$$

Also note that when both the start and stop codons do appear on the fragment, we lose the summations from Equation 5.4 and the left terms cancel, leaving the expected

$$S(x,1,1) = \frac{P(X=x|G=1)}{P(X=x|G=0)} \tag{5.7}$$

Given the fragment size and gene length probability distributions defined above, we can easily compute these scores. The model's dependence on the fragment size f is inconvenient, but necessary. For each dataset, we build models for all lengths present.

5.2.4 Classification

All previously published approaches to metagenomic gene prediction parameterize the gene composition models for each fragment as a function of its GC-content. For example, MetaGeneMark computes (offline) a logistic regression for each dicodon frequency as a function of GC-content for a large set of training genomes and sets its hidden Markov model parameters (online) according to the GC-content of the metagenomic sequence [205]. For whole genomes, gene composition model training has traditionally been performed on annotated close evolutionary relatives rather than genomes with similar GC-content [20]. Many methods for assigning a taxonomic classification to a metagenomic sequence are currently available [155, 168, 169, 172]. Here we suggest using one of these methods called Phymm [172] rather than GC-content to find evolutionary relatives of the metagenomic sequences on which to train gene composition models. Phymm trains an IMM on every reference genome in GenBank [209], scores each input sequence with all IMMs, and assigns a classification at each taxonomic level according to the reference genome of the highest scoring IMM. Phymm's IMMs are single-periodic and trained on whole genomes, in contrast to Glimmer-MG's IMMs which are 3-periodic and trained only on coding sequences.

Thus, before predicting genes, we run Phymm on the input sequences to score each sequence with each IMM. To train the gene prediction models, we use gene annotations for the genomes corresponding to the highest scoring IMMs. These annotations are taken from NCBI's Reference Sequence (RefSeq) database [210]. Though classification with Phymm is very accurate, the highest scoring IMM is rarely from the sequence's exact source genome. For this reason, we found that training over multiple genomes (*e.g.*, 3) captured a broader signal that improved prediction accuracy. Though most of the training can be performed offline, the models over multiple genomes must be combined online for each particular sequence. In order to realize a reasonable runtime, we must do this quickly. Features such as the length, start codon, and adjacent gene distributions are easy to combine across multiple training genomes by simply summing the feature counts.

IMMs cannot be combined quickly and saving trained IMMs for all combinations of 2 or 3 genomes would require too much disk space. In practice, pairs of genomes with similar composition are far more likely to be top classification hits together and we can restrict our offline training to only these pairs. We define an IMM composition distance on genomes X and Y that compares the likelihood that the genome's own IMM versus the other genome's IMM generated its sequence. If we let M_X and M_Y be the whole-genome Phymm IMMs and P_M give the probability that the IMM M generated the sequence, then we have

$$D_{\text{IMM}}(X,Y) = \frac{1}{|X|}\log\frac{P_{M_X}(X)}{P_{M_Y}(X)} + \frac{1}{|Y|}\log\frac{P_{M_Y}(Y)}{P_{M_X}(Y)} \tag{5.8}$$

If $X = Y$, the ratios are 1 and $D_{\text{IMM}} = 0$. If X and Y are very different, the ratios and the distance grow large. For each genome, we train Glimmer-MG gene IMMs on pairs of genomes for the 25 nearest genomes by our distance. If a metagenomic sequence's top hits do not contain a pre-trained pair, we default to a gene IMM trained on the single top classification for the sequence.

Glimmer-MG's RBS model trains using ELPH [211], a motif finder based on Gibbs sampling, to learn a 6 bp PWM from the 25 bp upstream of every gene in the training set. We train these PWMs offline for each individual reference genome, but like the other features, RBS modeling for metagenomic sequences benefits from the broader signal obtained by combining over multiple training genomes. Averaging PWMs for the top 3 Phymm classifications can be done quickly, but dilutes the signal. Instead, we generalized the RBS model in Glimmer-MG to score the upstream region of each start codon using a mixture of PWMs in equal proportions. Thus, a gene's RBS score is the probability that the best 6 bp motif in the 25 bp upstream of the start codon was generated by a mixture of 3 PWMs normalized by a null model based on GC-content to a log-likelihood ratio.

5.2.5 Clustering

On whole prokaryotic genomes, the following prediction pipeline has been applied successfully. First, train models on a finished and annotated close evolutionary relative. Make initial predictions, but then retrain the models on them and make a final set of predictions [20]. By using Phymm to find training genomes, we replicate the first step in this pipeline for application to metagenomics. However, retraining on the entire set of sequences would combine genes from many different organisms and yield a nonspecific and ineffective model. If the sequences could be separated by their source genome, retraining could be applied.

We accomplish this goal using Scimm, an unsupervised clustering method for metagenomic sequences that models each cluster with a single-periodic IMM [137]. After initially partitioning the sequences into a specified number of clusters, Scimm repeats the following three steps until the clusters are stable: train IMMs on the sequences assigned to their corresponding clusters, score each sequence using each cluster IMM, and reassign each sequence to the cluster corresponding to its highest scoring IMM. While Scimm may not partition the sequences exactly by their source organism, the mistakes that it tends to make do not create significant problems for retraining gene prediction models. In cases where Scimm merges sequences from two organisms together, they are nearly always phylogenetically related at the family level [137]. Scimm sometimes separates sequences from a single organism into multiple clusters, but this occurs most often for highly abundant organisms, in which case there will usually still be enough training data in each cluster to be informative. The already obtained Phymm classifications imply an initial clustering at a specified taxonomic level (*e.g.*, family), which can be used as an initial partition for the iterative clustering optimization in a mode of the program referred to as PhyScimm [137]. Using PhyScimm also implicitly chooses the number of clusters, removing this free variable.

After clustering the sequences, we focus on each cluster individually to retrain the coding IMM, RBS, and start codon models before making the final predictions within that cluster. The ORF length and adjacent ORF feature distributions are more difficult to estimate from short sequence fragments, so we still learn them using the Phymm classifications to whole annotated genomes. If the cluster is too small, retraining may not have enough data to capture the gene features, and prediction accuracy may decrease. We found 80 Kbp of predicted coding sequence was a useful threshold for retraining. For clusters with less, we do not retrain and instead finalize the gene predictions from the initial iteration. Accuracy may also decrease if the cluster is heterogeneous and does not effectively model some of its sequences. For each sequence, we compute the ratio between the likelihood that the cluster IMM versus its top scoring Phymm IMM generated the sequence. If the ratio is too low, we assume that the cluster does not represent this sequence well enough and finalize its initial predictions. The full pipeline for metagenomic gene prediction is depicted in Figure 5.3.

5.2.6 Sequencing errors

When predicting genes on the raw sequencing reads or contigs with low coverage, we must contend with sequencing errors. The most prevalent type of error made by the 454 sequencing technology is an insertion or deletion (indel) at a homopolymer run. Indels cause major problems for gene prediction by shifting the coding frame of the true gene, so that a method without a model for these errors could never predict the true gene. When Glimmer-MG encounters a shifted gene,

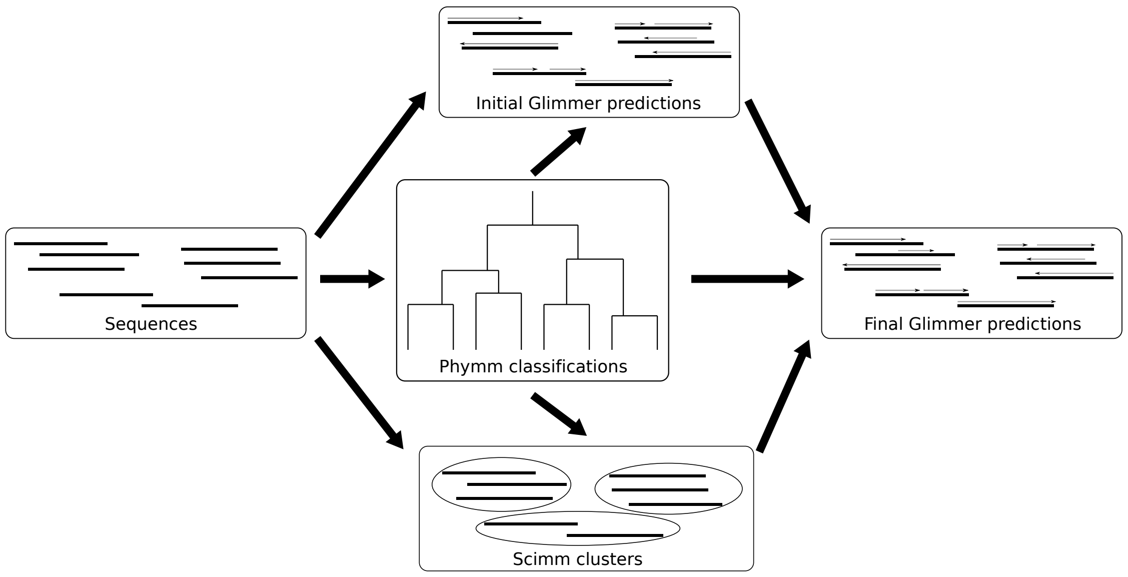


Figure 5.3: Glimmer-MG pipeline. First, we classify the sequences with Phymm in order to find related reference genomes to train the feature models. We use these to make initial gene predictions. Next, we cluster the sequences with SCIMM, starting at an initial partition from the Phymm classifications. Within each cluster, we retrain the models on the initial predictions before using all information to make the final set of predictions.

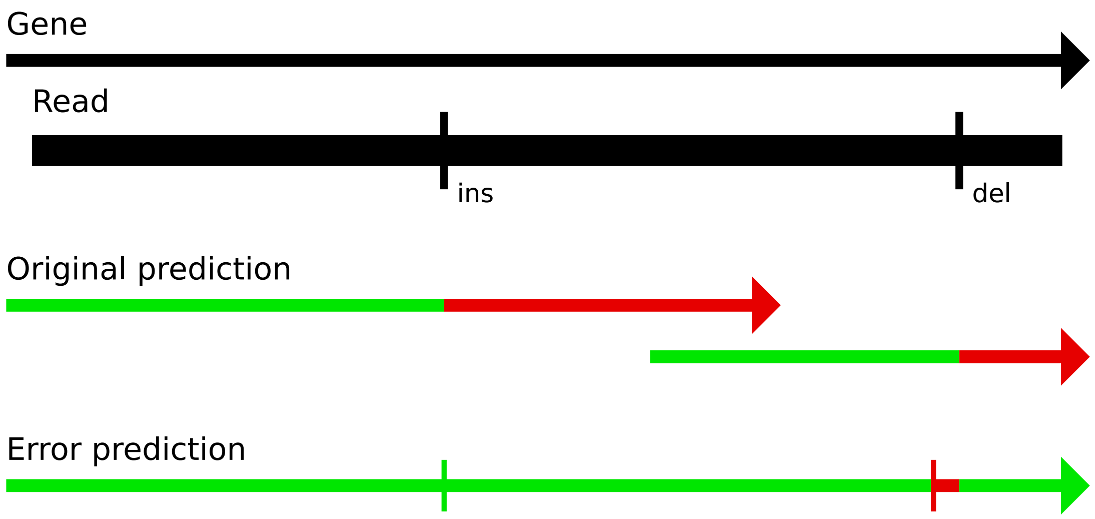


Figure 5.4: Depicted above is a common case where indel sequencing errors disrupt gene predictions. This 454-simulated 526 bp read falls within a gene in the forward direction, but has an insertion at position 207 and a deletion at position 480. Without modeling sequencing errors, Glimmer-MG begins to correctly predict the gene (shown in green), but is shifted into the wrong frame by the insertion (shown in red) and soon hits a stop codon. Downstream, Glimmer-MG makes another prediction in the correct coding frame, which is also forced into the wrong frame by the deletion. By allowing Glimmer-MG to predict frameshifts from sequencing errors, the prediction follows the coding frame nearly perfectly. The insertion site is exactly predicted and the deletion site is only off by 19 bp.

the most frequent outcome is two predictions, each of which covers half of the gene up to the point of the indel and then beyond (see Figure 5.4). Such predictions have limited utility.

While the problems caused by sequencing errors have been known for some time [148, 202], only recently has a good solution been published in the program FragGeneScan [206]. FragGeneScan uses a hidden Markov model where each of the three indexes into a codon are represented by a model state, but allows irregular transitions between the codon states that imply the presence of an indel in the sequence. On simulated sequences containing errors, FragGeneScan achieves far greater accuracy than previous methods that ignore the possibility of errors.

Because Glimmer-MG uses an ORF-based approach to gene prediction, we must take a more ad hoc approach to building an error model into the algorithm. When Glimmer-MG is scoring the composition of an ORF using the coding and noncoding IMMs, we allow branching into alternative reading frames. More specifically, we follow along the sequence and identify low quality base calls (defined below) that are strong candidates for a sequencing error. At these positions, we split the ORF into three branches. One branch scores the ORF as is. The other two switch into different frames to finish scoring, implying an insertion and deletion prediction. ORFs that change frames are penalized by the log-likelihood ratio of the predicted correction to the original base call probabilities. A maximum of two indel predictions per ORF is used to limit the computation time. After scoring all ORFs, ORFs with the same start and stop codon (but potentially different combinations of interior indels) are clustered and only the highest scoring is kept. All remaining

ORFs are pushed to the dynamic programming stage where the set of genes with maximum score subject to overlap constraints is chosen. However, the algorithm is further constrained to disallow an indel prediction in a region of overlapping genes.

Focusing on low quality base calls, which typically make up <5–10% of the sequence, makes the computation feasible. If quality values are available for the sequences, either from the raw read output or the consensus stage of an assembler, Glimmer-MG uses them and designates base calls less than a quality value threshold as potential branch sites. For 454 sequences that are missing quality values, we designate the final base of homopolymer runs longer than a length threshold as potential branch sites.

5.2.7 Whole genomes

Although we implemented the additional gene features with metagenomics in mind, they improve accuracy on whole genomes as well. In Glimmer3.0, the following pipeline was recommended [23]. First, using a program called *long-orfs*, find long non-overlapping ORFs in the sequence with amino acid composition that is typical of prokaryotic genomes. Train the coding IMM on these sequences, and predict genes on the genome. On the initial predictions, train the RBS and start codon models. Finally, make a second set of gene predictions incorporating all models.

For the new whole-genome version, designated as Glimmer3.1, we recommend a similar scheme. As before, we use *long-orfs* to train an IMM and predict an initial set of genes. Without a length model, these initial predictions tend to include many erroneous small gene predictions. We use a log-likelihood ratio threshold to filter out the lowest scoring ones. On the remaining genes, we retrain all models — IMM, RBS, start codons, length, and adjacency features — before predicting again. To eliminate any remaining bias from the initial prediction and filtering, we retrain and predict one final time.

The preceding pipeline is unsupervised, but we can do slightly better on average by following the Glimmer-MG version and using GenBank reference genomes. In this pipeline, we first classify the whole genome with Phymm to find similar reference genomes. Alternatively, a researcher may be able to specify these genomes based on prior knowledge. We train RBS, start codon, length, and adjacency models from the RefSeq annotations of these similar genomes as described. For the gene IMM, accuracy is better if we use *long-orfs* compared to an IMM trained on relative reference genomes. After making initial predictions, we retrain the IMM, RBS, and start codon models before predicting genes a final time.

5.2.8 Simulated metagenomes

We constructed simulated datasets from 1206 prokaryote genomes in GenBank [209] as of November 2010. Because Glimmer-MG involves clustering the sequences, it is important to have realistic simulated metagenomes. For each metagenome, we randomly chose 50 organisms and included all chromosomes and plasmids. We sampled organism abundances from the Pareto distribution, a power law probability distribution that has previously been used for modeling metagenomes [212]. Reference genomes included in the metagenome were removed from Phymm's database so that the sequences appeared novel and unknown. To simulate a single read, we selected a chromosome or plasmid with probability proportional to the product of its length and the organism's abundance and then chose a random position and orientation from that sequence. To enable comparison between experiments with different read lengths and error rates, we simulated 20 metagenomes (i.e. organisms, abundances, read positions, and read orientations) and used them to derive each experiment's dataset. We labeled the reads using gene annotations that are not described as hypothetical proteins from RefSeq [210].

In experiments where we considered sequencing errors, we focused on three prevailing technologies. Two varieties of high-throughput, short read technologies with very different characteristics have become ubiquitous tools for sequencing genomes, including metagenomics [26]. The Illumina sequencing platform generates 35–150 bp length reads with sequencing errors consisting almost entirely of substitutions [63]. The 454 sequencing platform generates 400–550 bp length reads where indels make up nearly all of the errors [30]. Less popular in recent studies due to greater expense and lesser throughput is Sanger sequencing with read lengths of 600–1000 bp and both substitution and indel sequencing errors. We include Sanger sequencing both because previous programs were designed and tested with the technology in mind and because the reads resemble contigs assembled from the more prevalent short read technologies with respect to length and errors tending to occur at the fragments' ends.

To imitate Sanger reads, we used the lengths and quality values from real reads taken from the NCBI Trace Archive [209] as templates. That is, for each fragment simulated from a genome as described above, we randomly chose a real Sanger read from our set to determine the length and quality values of the simulated read. Then we simulated errors into the read according to the quality values and using a ratio of five substitutions per indel. To achieve a specific error rate for a dataset, we multiplied the probability of error at every base by a factor defined by the desired rate. To generate simulated reads to imitate the Illumina platform, we similarly used real 124 bp reads as templates, but injected only substitution errors. For 454 reads, we used a read simulator called FlowSim which closely replicates the 454 stochastic sequencing process to generate the sequences and their quality values [213]. We conservatively quality trimmed all read ends to avoid large segments of erroneous sequence.

5.2.9 Accuracy

We computed accuracy a few ways to capture the multiple goals of gene prediction. Sensitivity is the ratio of true positive predictions to the number of true genes, and precision is the ratio of true positive predictions to the number of predicted genes. Because the RefSeq annotations tend to be incomplete after the removal of hypothetical proteins, which are unconfirmed computational predictions, we consider sensitivity to be the more important measure because "false positive" predictions may actually be real genes. For all experiments, we computed the sensitivity and precision of the 5' and 3' ends of the genes separately. Because there is only a single 3' site, 3' prediction is generally more valued. There are frequently many choices for the 5' end of the gene and a paucity of sequence information to discriminate between them. Adding to the difficulty, most of the 5' annotations in even the high quality RefSeq database are unverified.

In experiments with sequencing errors, indels shift the gene's frame and substitutions can compromise the start and stop codons. To measure the ability of the gene prediction to follow the coding frame, we compute sensitivity and precision at the nucleotide level. That is, every nucleotide is considered a unit and a true positive prediction must annotate the nucleotide as coding in the correct frame. A gene prediction that is correct until a sequencing error indel but predicted in the wrong frame beyond gets partial credit, whereas a gene prediction that identifies the error location and shifts the frame of the prediction gets full credit.

5.3 Results

5.3.1 Whole genomes

To evaluate the accuracy of the previous Glimmer3.0 iterated pipeline versus the proposed Glimmer3.1 and Glimmer-MG, we predicted genes in 12 reference genomes that cover a wide range of the prokaryotic phylogeny and were previously used to compare Glimmer3.0 to Glimmer2 [23]. Results for each of these genomes are displayed in Table 5.1. Note that the low precision values are not a concern because many genomes have a significant number of hypothetical proteins annotated and although predicting one of these genes is a false positive by our definition, some of them are likely to be real genes.

Glimmer3.1 maintains the high 3' sensitivity of Glimmer3, but improves the precision by 1.3% on average mainly by predicting fewer short genes (42 predictions <150 bp per genome versus 68) due to the length model. Glimmer-MG increases precision another 1.0% by using additional models, such as for gene length, learned accurately from close evolutionary relatives in the initial iteration. Glimmer3.1 also significantly improves TIS prediction as 5' sensitivity increases by 1.3% and precision by 1.8%. This improvement is attributable to its ability to assign greater scores to upstream start codons (which are longer genes) and penalize adjacent genes for unlikely arrangements like long overlaps. Glimmer-MG boosts sensitivity and precision relative to Glimmer3.1 by another 0.5% and 1.2% respectively.

5.3.2 Simulated metagenomes - perfect reads

To compare Glimmer-MG to previous methods for metagenomics gene prediction, we first predicted genes on simulated metagenomes with perfect read data without sequencing errors using Glimmer-MG, MetaGeneAnnotator [203], MetaGeneMark [205], and FragGeneScan [206]. MetaGeneAnnotator and MetaGeneMark runs used default parameters, and we set FragGeneScan's parameters for error-free sequences. Table 5.2 displays the programs' averaged accuracies over the 20 simulated metagenomes for each read technology.

Overall Glimmer-MG emerged as the clear best method, achieving the greatest

Table 5.1 ***(following page)*:** Accuracy on whole genomes. Sens - Sensitivity, Prec - Precision.

Genome			Glimmer3.0				Glimmer3.1				Glimmer-MG			
Organism	GC%	Genes	3’ Sens	3’ Prec	5’ Sens	5’ Prec	3’ Sens	3’ Prec	5’ Sens	5’ Prec	3’ Sens	3’ Prec	5’ Sens	5’ Prec
Archaeoglobus fulgidus	48%	1182	0.996	0.461	0.742	0.344	0.997	0.455	0.766	0.350	0.997	0.467	0.767	0.360
Bacillus anthracis	34%	3173	0.997	0.567	0.891	0.506	0.999	0.566	0.901	0.511	0.997	0.579	0.903	0.525
Bacillus subtilis	43%	3363	0.991	0.752	0.869	0.659	0.993	0.767	0.878	0.679	0.993	0.782	0.888	0.699
Chlorobium tepidum	55%	1303	0.998	0.596	0.688	0.411	0.997	0.636	0.731	0.466	0.997	0.646	0.723	0.469
Clostridium perfringens	28%	1690	0.999	0.626	0.918	0.575	0.999	0.624	0.924	0.576	0.999	0.629	0.925	0.583
Escherichia coli	50%	3367	0.983	0.745	0.871	0.660	0.983	0.780	0.880	0.699	0.983	0.783	0.890	0.709
Geobacter sulfurreducens	60%	2363	0.995	0.664	0.821	0.548	0.992	0.688	0.829	0.575	0.994	0.691	0.840	0.584
Helicobacter pylori	38%	945	0.994	0.555	0.865	0.483	0.996	0.513	0.873	0.450	0.992	0.560	0.872	0.493
Pseudomonas fluorescens	62%	4514	0.994	0.698	0.778	0.546	0.990	0.731	0.785	0.579	0.992	0.729	0.811	0.596
Ralstonia solanacearum	66%	2295	0.989	0.627	0.780	0.494	0.986	0.674	0.804	0.550	0.988	0.670	0.818	0.555
Staphylococcus epidermidis	31%	1674	0.997	0.684	0.919	0.631	0.998	0.687	0.930	0.640	0.996	0.692	0.929	0.645
Treponema pallidum	52%	582	0.986	0.519	0.677	0.356	0.981	0.549	0.692	0.388	0.981	0.562	0.701	0.402
Ureaplasma parvum	25%	402	1.000	0.657	0.983	0.645	1.000	0.654	0.973	0.636	1.000	0.661	0.970	0.641
Average	-	-	0.994	0.627	0.831	0.528	0.993	0.640	0.844	0.546	0.993	0.650	0.849	0.558

Tech	Method	3' Sens	3' Prec	5' Sens	5' Prec
Sanger	Glimmer-MG	0.987	0.702	0.901	0.641
(870 bp)	MetaGeneMark	0.969	0.707	0.857	0.625
	MetaGeneAnnotator	0.969	0.702	0.846	0.613
	FragGeneScan	0.962	0.667	0.823	0.570
454	Glimmer-MG	0.986	0.709	0.918	0.661
(535 bp)	MetaGeneMark	0.964	0.718	0.877	0.653
	MetaGeneAnnotator	0.966	0.707	0.853	0.625
	FragGeneScan	0.959	0.680	0.859	0.609
Illumina	Glimmer-MG	0.951	0.685	0.924	0.665
(120 bp)	MetaGeneMark	0.901	0.717	0.871	0.693
	MetaGeneAnnotator	0.915	0.686	0.839	0.629
	FragGeneScan	0.932	0.663	0.904	0.643

Table 5.2: Accuracy on simulated metagenomes with perfect reads. Tech - Sequencing technology, Sens - Sensitivity, Prec - Precision

sensitivity for every read length. Glimmer-MG's 3' sensitivity was nearly or exactly 2% greater than the second best method in each experiment, and its 5' sensitivity was better by margins up to 4.4% for Sanger reads. Glimmer-MG's precision on the longer 454 and Sanger reads was just behind MetaGeneMark for 3' prediction and exceeds all other programs for 5' prediction. On Illumina 120 bp reads, MetaGeneMark made much fewer predictions than the other programs leading to the greatest precision (3.2% greater than Glimmer-MG for 3'), but much lower sensitivity (5.0% less than Glimmer-MG for 3'). FragGeneScan was designed for these short reads and had better sensitivity than MetaGeneMark or MetaGeneAnnotator, but ∼2% less accuracy than Glimmer-MG by all measures.

By first classifying the reads, Glimmer-MG can identify sequences that are likely to use an irregular translation code, such as *Mycoplasma* bacteria where TGA codes for tryptophan rather than a stop codon. On the 0.35% of the reads in our simulated datasets that used irregular codes, Glimmer-MG predicted genes on the 454 reads with 91.1% sensitivity and 55.2% precision compared to the next best MetaGeneMark's 65.1% sensitivity and 37.6% precision. This difference was similar for other read lengths.

To assess the value of clustering and retraining, we also computed accuracy for Glimmer-MG's initial predictions. For each read type, retraining increased 3' sensitivity 0.6–2.0% while slightly decreasing 3' precision 0.4–0.6%. Illumina 3' sensitivity increased 2.0% because Phymm is less able to identify appropriate training genomes to aid the initial predictions; classification accuracy at the genus-level drops from 73.1% for Sanger reads to 34.3% for Illumina reads. After retraining, 5' sensi-

Tech	Error rate	Glimmer-MG		FragGeneScan	
		Sens	Prec	Sens	Prec
Sanger	0	0.989	0.756	0.977	0.740
(∼870 bp)	0.005	0.971	0.742	0.953	0.699
	0.010	0.955	0.731	0.938	0.687
	0.020	0.925	0.713	0.914	0.674
454	0	0.988	0.752	0.975	0.735
(∼535 bp)	0.005	0.899	0.679	0.846	0.621
	0.010	0.822	0.625	0.778	0.565
	0.020	0.711	0.545	0.678	0.501
Illumina	0	0.952	0.686	0.935	0.663
(∼120 bp)	0.005	0.938	0.682	0.923	0.640
	0.010	0.927	0.679	0.913	0.632
	0.020	0.910	0.673	0.900	0.625

Table 5.3: Accuracy on simulated metagenomes with error reads. Tech - Sequencing technology, Sens - Sensitivity, Prec - Precision. The read lengths are averages. Accuracy is computed at the nucleotide level.

tivity increased 1.5–1.9% with a similar level of precision, an improvement expected based on prior work [207].

5.3.3 Simulated metagenomes - error reads

Real metagenomic sequences will inevitably contain sequencing errors, and prior work showed that current gene prediction software struggles with these errors [202]. The recently published method FragGeneScan specifically models indel sequencing errors, which achieves far greater accuracy than other approaches when the sequences are short and error-prone [206]. To compare Glimmer-MG to FragGeneScan on reads containing errors, we simulated metagenomes as described using error rates ranging from 0–2%. We allowed Glimmer-MG to predict indels for Sanger and 454 reads, but not for Illumina. We ran FragGeneScan using predefined model parameters meant for the sequencing technology and the closest error rate. Table 5.3 displays the programs' averaged accuracies at the nucleotide level over the 20 simulated metagenomes for each read technology and error rate.

Glimmer-MG outperforms FragGeneScan with respect to both sensitivity and precision on all read lengths and error rates. The improvement is particularly evident for 454 reads where, for example, Glimmer-MG achieves 4.4% greater sensitivity and 5.8% greater precision than FragGeneScan at a 1.0% error rate. Glimmer-MG's limit of 2 indels per gene does not hinder gene prediction at a higher rate of 2.0% as accuracy remains greater than FragGeneScan.

Like prior work, our experiments demonstrate the difficulty of predicting genes on sequences with errors. For 454 reads where indel errors shift the gene frames, Glimmer-MG sensitivity plummets 9% for even a 0.5% error rate. The decrease in accuracy for Sanger or Illumina reads, where the errors are mostly from substitutions, should be less worrisome to researchers. Glimmer-MG sensitivity drops $\sim$2% for these reads when the error rate increases from 0 to 0.5%.

Modeling indel errors within Glimmer-MG significantly boosts performance for 454 reads. Without it, Glimmer-MG predicts with 41.9% sensitivity and 43.5%

precision at a 2.0% error rate, compared to 71.1% and 54.5% with indel prediction. Alternatively, on Illumina reads where the simulated sequencing errors are entirely substitutions, Glimmer-MG's indel prediction mode offers no benefit and slightly decreases precision. Sanger read prediction sees a meaningful 5.7% increase in sensitivity by modeling indels at a 2.0% error rate.

Comparison between Glimmer-MG initial and final prediction accuracy indicates that sequencing errors increase the value of retraining. For 454 reads, sensitivity increases 0.8% after retraining without errors, and 2.9% with 2.0% errors. Because retraining occurs on lower precision initial predictions, this result may be unintuitive. We can explain this as follows. Without sequencing errors, Glimmer-MG's predictions are very accurate so that the potential benefit of retraining and predicting again is limited. However, when there are sequencing errors, predicting coding sequence around indels is far more difficult, and the enhanced ability of Glimmer-MG's retrained models to identify coding sequence affects accuracy more significantly.

We measured both methods' accuracy predicting indels in the 454 simulated reads, to determine the degree to which it affects gene prediction accuracy. To do so, we computed a matching between the predicted and true indels in coding regions and called a pair separated by less than 15 bp a true positive. At a 1.0% error rate, Glimmer-MG correctly predicted 23.2% of the indels, with a 63.8% precision. FragGeneScan more readily shifts the gene frame and made 1.9 times more indel predictions. However, they resulted in fewer true positive predictions than Glimmer-MG's predictions (19.2% sensitivity) and far lower precision at 28.4%. For indels predicted correctly by both programs, Glimmer-MG's prediction was 2.3 bp away from the actual position on average, while FragGeneScan's was 5.2. Thus, by focusing on low quality nucleotides in the sequences, Glimmer-MG identifies indel positions more effectively than FragGeneScan. Sensitivity for both methods may seem low, but note that, in some cases, the frame of the coding sequence can still be closely followed without predicting the correct error. For example, two nearby insertions will generally result in a deletion prediction, which restores the proper frame more parsimoniously than two insertion predictions.

5.3.4 Real metagenomes

We evaluated the performance of Glimmer-MG on two real metagenomic datasets from a human gut microbiome study of obese and lean twins [208]. Sample TS28 consists of 303K reads sequenced by the 454 GS FLX Titanium with average length 335 bp, and sample TS50 consists of 550K reads sequenced by the 454 GS FLX with average length 204 bp. Evaluating prediction accuracy is more difficult for real metagenomes where there is no gold standard to compare against. We aligned the translated gene predictions made by Glimmer-MG and FragGeneScan against the NCBI nonredundant protein database with BLAST [152, 209], and considered a prediction to be a true positive if it matched a database protein with BLAST E-value less than 0.001.

Combining the two datasets, Glimmer-MG predicted 853K genes, of which 669K matched a known protein. Glimmer-MG's sensitivity was 4.5% greater than the 640K matches from FragGeneScan's 820K predictions. Precision has the caveat that a "false positive" prediction that does not match anything in the database will often represent a novel gene. Nevertheless, the two methods demonstrated a similar level of precision, 78.4% and 78.1% for Glimmer-MG and FragGeneScan respectively. For genes that were predicted by both methods, the aligned portions of Glimmer-MG predictions were 1.4% longer than those from FragGeneScan. Based on these results, Glimmer-MG is a better option for predicting genes on this human microbiome dataset.

5.4 Conclusion

A number of exciting projects over the last few years have demonstrated the value of environmental shotgun sequencing. As sequencing technologies are refined, the technique has the potential to make an even greater impact. But because the reads, having come from populations of usually unknown organisms, are difficult to analyze, metagenomics bioinformatics, including gene prediction, must improve in order to realize this potential. For example, projects seeking to discover new organisms such as the Global Ocean Sampling Expedition [42, 43] need accurate gene prediction to explore the functional repertoire of the many novel sequences obtained [196]. Projects focused on more well-known environments are also typically interested in characterizing the functional capabilities of the microbial communities, perhaps for comparison [197,198]. Methods to perform such functional comparisons benefit greatly from accurate identification of genes [199,200].

In this chapter, we introduced Glimmer-MG for metagenomics gene prediction. By modeling gene lengths and the presence of start and stop codons, Glimmer-MG successfully accounts for the truncated genes so common on metagenomic sequences. Where previous approaches parameterize prediction models using only the GC-content of the sequence, Glimmer-MG first classifies the sequences using a leading taxonomic classifier Phymm and trains models using the results. By clustering the reads using the unsupervised method Scimm, we elegantly allow retraining of prediction models on the sequences themselves. Augmenting Glimmer gene prediction with classification and clustering produces the most accurate gene predictions on our simulated metagenomes.

Sequencing errors in real metagenomics data wreak havoc on gene predictions. In Glimmer-MG, we can predict indels in error-prone sequences by considering frameshifts at low quality positions. In our experiments with real gut microbiome reads and simulated metagenomes with multiple types of sequencing technology, Glimmer-MG predicts genes on error-prone sequences more accurately than all other methods.

Overall, Glimmer-MG represents a substantial advance in metagenomics gene prediction, and should prove useful for a variety of applications.

5.5 Acknowledgements

We thank Derrick Wood for helpful TIS discussion.

Chapter 6

Conclusion

The work in this dissertation describes advances for a number of important computational problems related to the assembly and gene annotation of genomes. In each case, the biological results made from the analysis of the sequencing data are improved.

Chapter 2 introduced Quake, a method to detect and correct sequencing errors in high coverage Illumina datasets. Quake corrects errors more accurately than all previous approaches, and preprocessing data with it improves assembly and SNP finding results. The software is open source and available for use by the research community.

Chapter 3 uncovered a common mis-assembly in major genome assemblies where heterozygous sequence in diploid genomes was assembled as two contigs, creating a false duplication. We designed a method to detect these mis-assemblies and analyzed their impact, thus improving these genome assemblies that are widely used in comparative genomics.

In Chapter 4, we switched focus from traditional sequencing experiments to environmental shotgun sequencing, an exciting field in need of bioinformatics advances. The chapter described SCIMM, an unsupervised sequence clustering method, that clusters metagenomics sequences more effectively than previous programs. Scimm is available open source for researchers to analyze the composition of their metagenome.

Chapter 5 presents Glimmer-MG, an enhancement of the Glimmer gene prediction system for metagenomics sequences. By incorporating classification and clustering of the sequences, as well as modeling sequencing errors, Glimmer-MG predicts genes more accurately than all other approaches. Finding genes accurately on metagenomic sequences improves the functional analyses researchers want to perform on their metagenomic datasets.

As the technologies improve and sequencing becomes a ubiquitous tool for biological research, the field will depend on bioinformatics to efficiently and accurately process the data. The methods developed here are currently used by researchers for this purpose. I hope and expect that this work will also influence the development of the next generation of bioinformatics tools to meet the challenges posed by new and improved experiments and push forward this exciting line of research.

Bibliography

[1] Sanger F, Nicklen S, Coulson A: **DNA sequencing with chain-terminating inhibitors**. *Proc Natl Acad Sci USA* 1977, **74**(12):5463.

[2] Fleischmann R, Adams M, White O, Clayton R, Kirkness E, Kerlavage A, Bult C, Tomb J, Dougherty B, Merrick J, et al: **Whole-genome random sequencing and assembly of Haemophilus influenzae Rd**. *Science* 1995, **269**(5223):496.

[3] Lander E, Linton L, Birren B, Nusbaum C, Zody M, Baldwin J, Devon K, Dewar K, Doyle M, FitzHugh W, et al.: **Initial sequencing and analysis of the human genome**. *Nature* 2001, **409**(6822):860–921.

[4] Lander E, Waterman M: **Genomic mapping by fingerprinting random clones: a mathematical analysis**. *Genomics* 1988, **2**(3):231–239.

[5] Smith T, Waterman M: **Identification of common molecular subsequences**. *J Mol Biol* 1981, **147**:195–197.

[6] Karp R: **Reducibility among combinatorial problems**. *Complexity of Computer Computations* 1972.

[7] Kececioglu J, Myers E: **Combinatorial algorithms for DNA sequence assembly**. *Algorithmica* 1995, **13**:7–51.

[8] Green, P: **Phrap documentation** 1999. [Http://www.phrap.org/phredphrap/phrap.html].

[9] Myers E: **Toward simplifying and accurately formulating fragment assembly**. *J Comput Biol* 1995, **2**(2):275–290.

[10] Myers E: **The fragment assembly string graph**. *Bioinformatics* 2005, **21**(suppl 2):ii79.

[11] Pevzner P, Tang H, Waterman M: **An Eulerian path approach to DNA fragment assembly**. *Proc Natl Acad Sci USA* 2001, **98**(17):9748.

[12] Zerbino D, Birney E: **Velvet: algorithms for de novo short read assembly using de Bruijn graphs**. *Genome Res* 2008, **18**(5):821.

[13] Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK, Lander ES, Nusbaum C, Jaffe DB: **ALLPATHS: de novo assembly of whole-genome shotgun microreads.** *Genome Res* 2008, **18**:810–820.

[14] Pop M, Kosack D, Salzberg S: **Hierarchical scaffolding with Bambus**. *Genome Res* 2004, **14**:149.

[15] Myers EW, Sutton GG, Delcher AL, Dew IM, Fasulo DP, Flanigan MJ, Kravitz SA, Mobarry CM, Reinert KH, Remington KA, et al.: **A whole-genome assembly of Drosophila**. *Science* 2000, **287**(5461):2196–204.

[16] Batzoglou S, Jaffe D, Stanley K, Butler J, Gnerre S, Mauceli E, Berger B, Mesirov J, Lander E: **ARACHNE: a whole-genome shotgun assembler**. *Genome Res* 2002, **12**:177.

[17] Hughes JD, Estep PW, Tavazoie S, Church GM: **Computational identification of Cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae1**. *J Mol Biol* 2000, **296**(5):1205–1214.

[18] Salzberg S, Yorke J: **Beware of mis-assembled genomes**. *Bioinformatics* 2005, **21**(24):4320.

[19] Shine J, Dalgarno L: **The 3'-terminal sequence of Escherichia coli 16S ribosomal RNA: complementarity to nonsense triplets and ribosome binding sites**. *Proc Natl Acad Sci USA* 1974, **71**(4):1342.

[20] Majoros W: *Methods for computational gene prediction.* Cambridge University Press 2007.

[21] Salzberg SL, Delcher AL, Kasif S, White O: **Microbial gene identification using interpolated Markov models**. *Nucleic Acids Res* 1998, **26**(2):544–548.

[22] Borodovsky M, Mclninch JD, Koonin EV, Rudd KE, Médigue C, Danchin A: **Detection of new genes in a bacterial genome using Markov models for three gene classes**. *Nucleic Acids Res* 1995, **23**(17):3554–3562.

[23] Delcher AL, Bratke KA, Powers EC, Salzberg SL: **Identifying bacterial genes and endosymbiont DNA with Glimmer**. *Bioinformatics* 2007, **23**(6):673–679.

[24] Aziz R, Bartels D, Best A, DeJongh M, Disz T, Edwards R, Formsma K, Gerdes S, Glass E, Kubal M, et al.: **The RAST Server: rapid annotations using subsystems technology**. *BMC Genomics* 2008, **9**:75.

[25] Rocha E: **The organization of the bacterial genome**. *Annu Rev Genet* 2008, **42**:211–233.

[26] Shendure J, Ji H: **Next-generation DNA sequencing**. *Nat Biotechnol* 2008, **26**:1135–1145.

[27] **454 Life Sciences** [http://www.454.com].

[28] **Illumina** [http://www.illumina.com].

[29] Bentley D, Balasubramanian S, Swerdlow H, Smith G, Milton J, Brown C, Hall K, Evers D, Barnes C, Bignell H, et al.: **Accurate whole human genome sequencing using reversible terminator chemistry**. *Nature* 2008, **456**(7218):53–59.

[30] Margulies M, Egholm M, Altman W, Attiya S, Bader J, Bemben L, Berka J, Braverman M, Chen Y, Chen Z, et al.: **Genome sequencing in microfabricated high-density picolitre reactors**. *Nature* 2005, **437**(7057):376–380.

[31] Siva N: **1000 Genomes project**. *Nat Biotechnol* 2008, **26**:256.

[32] Mardis E: **Cancer genomics identifies determinants of tumor biology**. *Genome Biol* 2010, **11**:211.

[33] Haussler D, O'Brien S, Ryder O, Barker F, Clamp M, Crawford A, Hanner R, Hanotte O, Johnson W, McGuire J, et al.: **Genome 10K: A proposal to obtain whole-genome sequence for 10 000 vertebrate species**. *J Hered* 2009, **100**:659–674.

[34] **Illumina MiSeq** [http://www.illumina.com/systems/miseq.ilmn].

[35] **Ion Torrent** [http://www.iontorrent.com/about/overview].

[36] Wetterstrand K: **DNA Sequencing Costs: Data from the NHGRI Large-Scale Genome Sequencing Program** [http://www.genome.gov/sequencingcosts]. [Accessed Mar 14, 2011].

[37] Wang Z, Gerstein M, Snyder M: **RNA-Seq: a revolutionary tool for transcriptomics**. *Nat Rev Genet* 2009, **10**:57–63.

[38] Park P: **ChIP-seq: advantages and challenges of a maturing technology**. *Nat Rev Genet* 2009, **10**(10):669–680.

[39] Wooley J, Godzik A, Friedberg I: **A primer on metagenomics**. *PLoS Comput Biol* 2010, **6**(2):e1000667.

[40] Handelsman J: **Metagenomics: application of genomics to uncultured microorganisms**. *Microbiol Mol Biol Rev* 2004, **68**(4):669.

[41] Tyson G, Chapman J, Hugenholtz P, Allen E, Ram R, Richardson P, Solovyev V, Rubin E, Rokhsar D, Banfield J: **Community structure and metabolism through reconstruction of microbial genomes from the environment**. *Nature* 2004, **428**(6978):37–43.

[42] Venter J, Remington K, Heidelberg J, Halpern A, Rusch D, Eisen J, Wu D, Paulsen I, Nelson K, Nelson W, et al.: **Environmental genome shotgun sequencing of the Sargasso Sea**. *Science* 2004, **304**(5667):66.

[43] Rusch D, Halpern A, Sutton G, Heidelberg K, Williamson S, Yooseph S, Wu D, Eisen J, Hoffman J, Remington K, et al.: **The Sorcerer II Global Ocean Sampling Expedition: Northwest Atlantic through Eastern Tropical Pacific**. *PLoS Biol* 2007, **5**(3):e77.

[44] Chen K, Pachter L: **Bioinformatics for whole-genome shotgun sequencing of microbial communities**. *PLoS Comput Biol* 2005, **1**(2):106–12.

[45] Chaisson M, Brinza D, Pevzner P: **De novo fragment assembly with short mate-paired reads: Does the read length matter?** *Genome Res* 2009, **19**:336.

[46] Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, Li Y, Li S, Shan G, Kristiansen K, et al.: **De novo assembly of human genomes with massively parallel short read sequencing**. *Genome Res* 2010, **20**:265–272.

[47] Kelley DR, Schatz MC, Salzberg SL: **Quake: quality-aware detection and correction of sequencing errors**. *Genome Biol* 2010, **11**(11):R116.

[48] Hawkins R, Hon G, Ren B: **Next-generation genomics: an integrative approach**. *Nat Rev Genet* 2010, **11**:476–486.

[49] Robison K: **Application of second-generation sequencing to cancer genomics**. *Brief Bioinform* 2010, **11**:524–534.

[50] Palmer L, Dejori M, Bolanos R, Fasulo D: **Improving de novo sequence assembly using machine learning and comparative genomics for overlap correction**. *BMC Bioinformatics* 2010, **11**:33.

[51] Trapnell C, Salzberg S: **How to map billions of short reads onto genomes**. *Nat Biotechnol* 2009, **27**:455–7.

[52] Shi H, Schmidt B, Liu W, Muller-Wittig W: **A Parallel Algorithm for Error Correction in High-Throughput Short-Read Data on CUDA-Enabled Graphics Hardware**. *J Comput Biol* 2010, **17**:603–615.

[53] Yang X, Dorman K, Aluru S: **Reptile: Representative Tiling for Short Read Error Correction**. *Bioinformatics* 2010.

[54] Gajer P, Schatz M, Salzberg S: **Automated correction of genome sequence errors**. *Nucleic Acids Res* 2004, **32**:562.

[55] Tammi MT, Arner E, Kindlund E, Andersson B: **Correcting errors in shotgun sequences**. *Nucleic Acids Res* 2003, **31**:4663–4672.

[56] Schroder J, Schroder H, Puglisi SJ, Sinha R, Schmidt B: **SHREC: a short-read error correction method**. *Bioinformatics* 2009, **25**:2157–2163.

[57] Salmela L: **Correction of sequencing errors in a mixed set of reads**. *Bioinformatics* 2010, **26**:1284.

[58] Simpson J, Wong K, Jackman S, Schein J, Jones S, Birol I: **ABySS: A parallel assembler for short read sequence data**. *Genome Res* 2009, **19**:1117.

[59] Qu W, Hashimoto Si, Morishita S: **Efficient frequency-based de novo short-read clustering for error trimming in next-generation sequencing**. *Genome Res* 2009, **19**:1309–1315.

[60] Wijaya E, Frith M, Suzuki Y, Horton P: **Recount: Expectation maximization based error correction tool for next generation sequencing data**. In *Genome Inform*, *Volume 23* 2009:189–201.

[61] Zagordi O, Geyrhofer L, Roth V, Beerenwinkel N: **Deep sequencing of a genetically heterogeneous sample: local haplotype reconstruction and read error correction**. *J Comput Biol* 2010, **17**:417–428.

[62] Quince C, Lanzen A, Curtis T, Davenport R, Hall N, Head I, Read L, Sloan W: **Accurate determination of microbial diversity from 454 pyrosequencing data**. *Nat Methods* 2009, **6**:639–641.

[63] Dohm JC, Lottaz C, Borodina T, Himmelbauer H: **Substantial biases in ultra-short read data sets from high-throughput DNA sequencing**. *Nucleic Acids Res* 2008, **36**:e105+.

[64] Bravo H, Irizarry R: **Model-based quality assessment and base-calling for second-generation sequencing data**. *Biometrics* 2009.

[65] Kao W, Stevens K, Song Y: **BayesCall: A model-based base-calling algorithm for high-throughput short-read sequencing**. *Genome Res* 2009, **19**:1884.

[66] Erlich Y, Mitra P, de la Bastide M, McCombie W, Hannon G: **Alta-Cyclic: a self-optimizing base caller for next-generation sequencing.** *Nat Methods* 2008, **5**:679.

[67] Kircher M, Stenzel U, Kelso J: **Improved base calling for the Illumina Genome Analyzer using machine learning strategies**. *Genome Biol* 2009, **10**:R83.

[68] Rougemont J, Amzallag A, Iseli C, Farinelli L, Xenarios I, Naef F: **Probabilistic base calling of Solexa sequencing data**. *BMC Bioinformatics* 2008, **9**:431.

[69] **Quake** [http://www.cbcb.umd.edu/software/quake].

[70] **Perl Artistic License** [http://www.perl.com/pub/a/language/misc/Artistic.html].

[71] Li H, Ruan J, Durbin R: **Mapping short DNA sequencing reads and calling variants using mapping quality scores**. *Genome Res* 2008, **18**:1851.

[72] Cordaux R, Batzer M: **The impact of retrotransposons on human genome evolution**. *Nat Rev Genet* 2009, **10**:691–703.

[73] Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, Salzberg SL: **Versatile and open software for comparing large genomes**. *Genome Biol* 2004, **5**(2):R12.

[74] Pleasance ED, Cheetham RK, Stephens PJ, McBride DJ, Humphray SJ, Greenman CD, Varela I, Lin MLL, Ordonez GR, Bignell GR, et al.: **A comprehensive catalogue of somatic mutations from a human cancer genome.** *Nature* 2010, **463**:191–196.

[75] Kendal W: **An exponential dispersion model for the distribution of human single nucleotide polymorphisms**. *Mol Biol Evol* 2003, **20**:579.

[76] Langmead B, Trapnell C, Pop M, Salzberg S: **Ultrafast and memory-efficient alignment of short DNA sequences to the human genome**. *Genome Biol* 2009, **10**:R25.

[77] Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, Subgroup GPDP: **The sequence alignment/Map format and SAMtools**. *Bioinformatics* 2009, **25**:2078.

[78] Ahn SM, Kim TH, Lee S, Kim D, Ghang H, Kim DS, Kim BC, Kim SY, Kim WY, Kim C, et al.: **The first Korean genome sequence and analysis: Full genome sequencing for a socio-ethnic group**. *Genome Res* 2009, **19**:1622–1629.

[79] **Hadoop** [http://hadoop.apache.org].

[80] Dagum L, Menon R: **Open MP: An Industry-Standard API for Shared-Memory Programming**. *IEEE Computational Science and Engineering* 1998, **5**:46–55.

[81] Dean J, Ghemawat S: **MapReduce: simplified data processing on large clusters**. *Commun ACM* 2008, **51**:107–113.

[82] Chin F, Leung H, Li W, Yiu S: **Finding optimal threshold for correction error reads in DNA assembling**. *BMC Bioinformatics* 2009, **10**:S15.

[83] Johnson N, Kemp A, Kotz S: *Univariate discrete distributions.* Wiley-Interscience 2005.

[84] R Development Core Team: *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria 2010, [http://www.R-project.org]. [ISBN 3-900051-07-0].

[85] Li M, Nordborg M, Li LM: **Adjust quality scores from alignment and improve sequencing accuracy**. *Nucleic Acids Res* 2004, **32**:5183–5191.

[86] Li R, Li Y, Fang X, Yang H, Wang J, Kristiansen K, Wang J: **SNP detection for massively parallel whole-genome resequencing**. *Genome Res* 2009, **19**:1124.

[87] Hastie T, Tibshirani R, Friedman J: *The Elements of Statistical Learning*. Springer 2009.

[88] Kelley DR, Salzberg SL: **Detection and correction of false segmental duplications caused by genome mis-assembly**. *Genome Biol* 2010, **11**(3):R28.

[89] Adams M, Celniker S, Holt R, Evans C, Gocayne J, Amanatides P, Scherer S, Li P, Hoskins R, Galle R, et al.: **The genome sequence of Drosophila melanogaster**. *Science* 2000, **287**(5461):2185.

[90] Bailey JA, Gu Z, Clark RA, Reinert K, Samonte RV, Schwartz S, Adams MD, Myers EW, Li PW, Eichler EE: **Recent segmental duplications in the human genome**. *Science* 2002, **297**(5583):1003–7.

[91] Cheung J, Estivill X, Khaja R, MacDonald JR, Lau K, Tsui LC, Scherer SW: **Genome-wide detection of segmental duplications and potential assembly errors in the human genome sequence**. *Genome Biol* 2003, **4**(4):R25.

[92] Nicholas TJ, Cheng Z, Ventura M, Mealey K, Eichler EE, Akey JM: **The genomic architecture of segmental duplications and associated copy number variants in dogs**. *Genome Res* 2009, **19**(3):491–9.

[93] Cheung J, Wilson MD, Zhang J, Khaja R, MacDonald JR, Heng HH, Koop BF, Scherer SW: **Recent segmental and gene duplications in the mouse genome**. *Genome Biol* 2003, **4**(8):R47.

[94] Bult C, White O, Olsen G, Zhou L, Fleischmann R, Sutton G, Blake J, FitzGerald L, Clayton R, Gocayne J, et al.: **Complete genome sequence of the methanogenic archaeon, Methanococcus jannaschii**. *Science* 1996, **273**(5278):1058.

[95] Barriere A, Yang SP, Pekarek E, Thomas CG, Haag ES, Ruvinsky I: **Detecting heterozygosity in shotgun genome assemblies: Lessons from obligately outcrossing nematodes**. *Genome Res* 2009, **19**(3):470–80.

[96] Holt RA, Subramanian GM, Halpern A, Sutton GG, Charlab R, Nusskern DR, Wincker P, Clark AG, Ribeiro JM, Wides R, et al.: **The genome sequence of the malaria mosquito Anopheles gambiae**. *Science* 2002, **298**(5591):129–49.

[97] Jones T, Federspiel NA, Chibana H, Dungan J, Kalman S, Magee BB, Newport G, Thorstenson YR, Agabian N, Magee PT, et al.: **The diploid genome sequence of Candida albicans**. *Proc Natl Acad Sci U S A* 2004, **101**(19):7329–34.

[98] Vinson JP, Jaffe DB, O'Neill K, Karlsson EK, Stange-Thomann N, Anderson S, Mesirov JP, Satoh N, Satou Y, Nusbaum C, et al.: **Assembly of polymorphic genomes: algorithms and application to Ciona savignyi**. *Genome Res* 2005, **15**(8):1127–35.

[99] Bailey JA, Church DM, Ventura M, Rocchi M, Eichler EE: **Analysis of segmental duplications and genome assembly in the mouse**. *Genome Res* 2004, **14**(5):789–801.

[100] Sharp AJ, Locke DP, McGrath SD, Cheng Z, Bailey JA, Vallente RU, Pertz LM, Clark RA, Schwartz S, Segraves R, et al.: **Segmental duplications and copy-number variation in the human genome**. *Am J Hum Genet* 2005, **77**:78–88.

[101] Teichmann SA, Babu MM: **Gene regulatory network growth by duplication**. *Nat Genet* 2004, **36**(5):492–6.

[102] Varki A, Altheide TK: **Comparing the human and chimpanzee genomes: searching for needles in a haystack**. *Genome Res* 2005, **15**(12):1746–58.

[103] Conrad B, Antonarakis SE: **Gene duplication: a drive for phenotypic diversity and cause of human disease**. *Annu Rev Genomics Hum Genet* 2007, **8**:17–35.

[104] De Gobbi M, Viprakasit V, Hughes JR, Fisher C, Buckle VJ, Ayyub H, Gibbons RJ, Vernimmen D, Yoshinaga Y, de Jong P, et al.: **A regulatory SNP causes a human genetic disease by creating a new transcriptional promoter**. *Science* 2006, **312**(5777):1215–7.

[105] Phillippy A, Schatz M, Pop M: **Genome assembly forensics: finding the elusive mis-assembly**. *Genome Biol* 2008, **9**:R55.

[106] Choi JH, Kim S, Tang H, Andrews J, Gilbert DG, Colbourne JK: **A machine-learning approach to combined evidence validation of genome assemblies**. *Bioinformatics* 2008, **24**(6):744–50.

[107] Zimin AV, Smith DR, Sutton G, Yorke JA: **Assembly reconciliation**. *Bioinformatics* 2008, **24**:42–5.

[108] Zimin AV, Delcher AL, Florea L, Kelley DR, Schatz MC, Puiu D, Hanrahan F, Pertea G, Van Tassell CP, Sonstegard TS, et al.: **A whole-genome assembly of the domestic cow, Bos taurus**. *Genome Biol* 2009, **10**(4):R42.

[109] Sequencing TC, Consortium A: **Initial sequence of the chimpanzee genome and comparison with the human genome**. *Nature* 2005, **437**(7055):69–87.

[110] Consortium ICGS: **Sequence and comparative analysis of the chicken genome provide unique perspectives on vertebrate evolution**. *Nature* 2004, **432**(7018):695–716.

[111] Lindblad-Toh K, Wade CM, Mikkelsen TS, Karlsson EK, Jaffe DB, Kamal M, Clamp M, Chang JL, Kulbokas r E J, Zody MC, et al.: **Genome sequence, comparative analysis and haplotype structure of the domestic dog**. *Nature* 2005, **438**(7069):803–19.

[112] Huang X, Wang J, Aluru S, Yang SP, Hillier L: **PCAP: a whole-genome assembly program**. *Genome Res* 2003, **13**(9):2164–70.

[113] Elsik CG, Tellam RL, Worley KC, Gibbs RA, Muzny DM, Weinstock GM, Adelson DL, Eichler EE, Elnitski L, Guigo R, et al.: **The genome sequence of taurine cattle: a window to ruminant biology and evolution**. *Science* 2009, **324**(5926):522–8.

[114] Liu Y, Qin X, Song XZ, Jiang H, Shen Y, Durbin KJ, Lien S, Kent MP, Sodeland M, Ren Y, et al.: **Bos taurus genome assembly**. *BMC Genomics* 2009, **10**:180.

[115] Jaffe DB, Butler J, Gnerre S, Mauceli E, Lindblad-Toh K, Mesirov JP, Zody MC, Lander ES: **Whole-genome sequence assembly for mammalian genomes: Arachne 2**. *Genome Res* 2003, **13**:91–6.

[116] Cheng Z, Ventura M, She X, Khaitovich P, Graves T, Osoegawa K, Church D, DeJong P, Wilson RK, Paabo S, et al.: **A genome-wide comparison of recent chimpanzee and human segmental duplications**. *Nature* 2005, **437**(7055):88–93.

[117] Smit A, Hubley R, Green P: **RepeatMasker Open-3.0** 1996-2004.

[118] Maglott D, Ostell J, Pruitt KD, Tatusova T: **Entrez Gene: gene-centered information at NCBI**. *Nucleic Acids Res* 2007, **35**(Database issue):D26–31.

[119] Rausch T, Koren S, Denisov G, Weese D, Emde AK, Doring A, Reinert K: **A consistency-based consensus algorithm for de novo and reference-guided sequence assembly of short reads**. *Bioinformatics* 2009, **25**(9):1118–24.

[120] Sherry ST, Ward MH, Kholodov M, Baker J, Phan L, Smigielski EM, Sirotkin K: **dbSNP: the NCBI database of genetic variation**. *Nucleic Acids Res* 2001, **29**:308–11.

[121] Jaillon O, Aury JM, Noel B, Policriti A, Clepet C, Casagrande A, Choisne N, Aubourg S, Vitulo N, Jubin C, et al.: **The grapevine genome sequence suggests ancestral hexaploidization in major angiosperm phyla**. *Nature* 2007, **449**(7161):463–7.

[122] She X, Liu G, Ventura M, Zhao S, Misceo D, Roberto R, Cardone MF, Rocchi M, Green ED, Archidiacano N, et al.: **A preliminary comparative analysis of primate segmental duplications shows elevated substitution rates and a great-ape expansion of intrachromosomal duplications**. *Genome Res* 2006, **16**(5):576–83.

[123] Marques-Bonet T, Kidd JM, Ventura M, Graves TA, Cheng Z, Hillier LW, Jiang Z, Baker C, Malfavon-Borja R, Fulton LA, et al.: **A burst of segmental duplications in the genome of the African great ape ancestor**. *Nature* 2009, **457**(7231):877–81.

[124] Frazer KA, Ballinger DG, Cox DR, Hinds DA, Stuve LL, Gibbs RA, Belmont JW, Boudreau A, Hardenbol P, Leal SM, et al.: **A second generation human haplotype map of over 3.1 million SNPs**. *Nature* 2007, **449**(7164):851–61.

[125] Vignal A, Milan D, SanCristobal M, Eggen A: **A review on SNP and other types of molecular markers and their use in animal genetics**. *Genet Sel Evol* 2002, **34**(3):275–305.

[126] Altshuler D, Pollara VJ, Cowles CR, Van Etten WJ, Baldwin J, Linton L, Lander ES: **An SNP map of the human genome generated by reduced representation shotgun sequencing**. *Nature* 2000, **407**(6803):513–6.

[127] Wong GK, Liu B, Wang J, Zhang Y, Yang X, Zhang Z, Meng Q, Zhou J, Li D, Zhang J, et al.: **A genetic variation map for chicken with 2.8 million single-nucleotide polymorphisms**. *Nature* 2004, **432**(7018):717–22.

[128] Levy S, Sutton G, Ng PC, Feuk L, Halpern AL, Walenz BP, Axelrod N, Huang J, Kirkness EF, Denisov G, et al.: **The diploid genome sequence of an individual human**. *PLoS Biol* 2007, **5**(10):e254.

[129] Wang J, Wang W, Li R, Li Y, Tian G, Goodman L, Fan W, Zhang J, Li J, Guo Y, et al.: **The diploid genome sequence of an Asian individual**. *Nature* 2008, **456**(7218):60–5.

[130] Denisov G, Walenz B, Halpern AL, Miller J, Axelrod N, Levy S, Sutton G: **Consensus generation and variant detection by Celera Assembler**. *Bioinformatics* 2008, **24**(8):1035–40.

[131] Bansal V, Bafna V: **HapCUT: an efficient and accurate algorithm for the haplotype assembly problem**. *Bioinformatics* 2008, **24**(16):i153–9.

[132] Kim JH, Waterman MS, Li LM: **Diploid genome reconstruction of Ciona intestinalis and comparative analysis with Ciona savignyi**. *Genome Res* 2007, **17**(7):1101–10.

[133] Fasulo D, Halpern A, Dew I, Mobarry C: **Efficiently detecting polymorphisms during the fragment assembly process**. *Bioinformatics* 2002, **18 Suppl 1**:S294–302.

[134] Yu N, Jensen-Seaman MI, Chemnick L, Kidd JR, Deinard AS, Ryder O, Kidd KK, Li WH: **Low nucleotide diversity in chimpanzees and bonobos**. *Genetics* 2003, **164**(4):1511–8.

[135] Fischer A, Wiebe V, Paabo S, Przeworski M: **Evidence for a complex demographic history of chimpanzees**. *Mol Biol Evol* 2004, **21**(5):799–808.

[136] Gibbs RA, Taylor JF, Van Tassell CP, Barendse W, Eversole KA, Gill CA, Green RD, Hamernik DL, Kappes SM, Lien S, et al.: **Genome-wide survey of SNP variation uncovers the genetic structure of cattle breeds**. *Science* 2009, **324**(5926):528–32.

[137] Kelley DR, Salzberg SL: **Clustering metagenomic sequences with interpolated Markov models**. *BMC Bioinformatics* 2010, **11**:544.

[138] Liolios K, Chen I, Min A, Mavromatis K, Tavernarakis N, Hugenholtz P, Markowitz V, Kyrpides N: **The Genomes On Line Database (GOLD) in 2009: status of genomic and metagenomic projects and their associated metadata**. *Nucleic Acids Res* 2010, **38**(Database issue):D346.

[139] Wu D, Hugenholtz P, Mavromatis K, Pukall R, Dalin E, Ivanova NN, Kunin V, Goodwin L, Wu M, Tindall BJ, et al.: **A phylogeny-driven genomic encyclopaedia of Bacteria and Archaea**. *Nature* 2009, **462**(7276):1056–60.

[140] Eisen JA: **Environmental shotgun sequencing: its potential and challenges for studying the hidden world of microbes**. *PLoS Biol* 2007, **5**(3):e82.

[141] Costello E, Lauber C, Hamady M, Fierer N, Gordon J, Knight R: **Bacterial community variation in human body habitats across space and time**. *Science* 2009, **326**(5960):1694.

[142] Grice EA, Kong HH, Conlan S, Deming CB, Davis J, Young AC, Program NCS, Bouffard GG, Blakesley RW, Murray PR, et al.: **Topographical and temporal diversity of the human skin microbiome**. *Science* 2009, **324**(5931):1190–1192.

[143] Qin J, Li R, Raes J, Arumugam M, Burgdorf KS, Manichanh C, Nielsen T, Pons N, Levenez F, Yamada T, et al.: **A human gut microbial gene catalogue established by metagenomic sequencing**. *Nature* 2010, **464**(7285):59–65.

[144] Hamady M, Knight R: **Microbial community profiling for human microbiome projects: Tools, techniques, and challenges**. *Genome Res* 2009, **19**(7):1141–1152.

[145] Rodriguez-Brito B, Li L, Wegley L, Furlan M, Angly F, Breitbart M, Buchanan J, Desnues C, Dinsdale E, Edwards R, et al.: **Viral and microbial community dynamics in four aquatic environments**. *ISME J* 2010.

[146] Kosakovsky Pond S, Wadhawan S, Chiaromonte F, Ananda G, Chung W, Taylor J, Nekrutenko A: **Windshield splatter analysis with the Galaxy metagenomic pipeline.** *Genome Res* 2009, **19**(11):2144.

[147] Weinberg Z, Perreault J, Meyer M, Breaker R: **Exceptional structured noncoding RNAs revealed by bacterial metagenome analysis**. *Nature* 2009, **462**(7273):656–659.

[148] Mavromatis K, Ivanova N, Barry K, Shapiro H, Goltsman E, McHardy AC, Rigoutsos I, Salamov A, Korzeniewski F, Land M, Lapidus A, Grigoriev I, Richardson P, Hugenholtz P, Kyrpides NC: **Use of simulated data sets to evaluate the fidelity of metagenomic processing methods**. *Nat Methods* 2007, **4**(6):495–500.

[149] McHardy A, Rigoutsos I: **What's in the mix: phylogenetic classification of metagenome sequence samples**. *Curr Opin Microbiol* 2007, **10**(5):499–503.

[150] Navlakha S, White J, Nagarajan N, Pop M, Kingsford C: **Finding biologically accurate clusterings in hierarchical tree decompositions using the variation of information**. In *Research in Computational Molecular Biology* 2009:400–417.

[151] Wu M, Eisen J: **A simple, fast, and accurate method of phylogenomic inference**. *Genome Biol* 2008, **9**(10):R151.

[152] Altschul S, Madden T, Schäffer A, Zhang J, Zhang Z, Miller W, Lipman D: **Gapped BLAST and PSI-BLAST: a new generation of protein database search programs**. *Nucleic Acids Res* 1997, **25**(17):3389.

[153] Benson DA, Karsch-Mizrachi I, Lipman DJ, Ostell J, Sayers EW: **GenBank**. *Nucleic Acids Res* 2010, **38**(Database issue):D46–D51.

[154] Gerlach W, Junemann S, Tille F, Goesmann A, Stoye J: **WebCARMA: a web application for the functional and taxonomic classification of unassembled metagenomic reads**. *BMC Bioinformatics* 2009, **10**:430.

[155] Haque MM, Ghosh T, Komanduri D, Mande S: **SOrt-ITEMS: Sequence orthology based approach for improved taxonomic estimation of metagenomic sequences**. *Bioinformatics* 2009, **25**(14):1722–1730.

[156] Huson D, Auch A, Qi J, Schuster S: **MEGAN analysis of metagenomic data**. *Genome Res* 2007, **17**(3):377–386.

[157] Koski LB, Golding GB: **The closest BLAST hit is often not the nearest neighbor**. *J Mol Evol* 2001, **52**(6):540–2.

[158] Karlin S, Mrazek J, Campbell AM: **Compositional biases of bacterial genomes and evolutionary implications**. *J Bacteriol* 1997, **179**(12):3899–913.

[159] Bohlin J, Skjerve E, Ussery D: **Analysis of genomic signatures in prokaryotes using multinomial regression and hierarchical clustering**. *BMC Genomics* 2009, **10**:487.

[160] Mann S, Chen YP: **Bacterial genomic G+C composition-eliciting environmental adaptation**. *Genomics* 2010, **95**:7–15.

[161] Abe T, Kanaya S, Kinouchi M, Ichiba Y, Kozuki T, Ikemura T: **Informatics for unveiling hidden genome signatures**. *Genome Res* 2003, **13**(4):693–702.

[162] Teeling H, Meyerdierks A, Bauer M, Amann R, Glockner F: **Application of tetranucleotide frequencies for the assignment of genomic fragments**. *Environ Microbiol* 2004, **6**(9):938–947.

[163] Bohlin J, Skjerve E, Ussery D: **Investigations of Oligonucleotide Usage Variance Within and Between Prokaryotes**. *PLoS Comput Biol* 2008, **4**(4):e1000057.

[164] Mrazek J: **Phylogenetic signals in DNA composition: limitations and prospects**. *Mol Biol Evol* 2009, **26**(5):1163–1169.

[165] Lee SJ, Mortimer JR, Forsdyke DR: **Genomic conflict settled in favour of the species rather than the gene at extreme GC percentage values**. *Appl Bioinformatics* 2004, **3**(4):219–28.

[166] Lawrence JG, Ochman H: **Amelioration of bacterial genomes: rates of change and exchange**. *J Mol Evol* 1997, **44**(4):383–97.

[167] Dick G, Andersson A, Baker B, Simmons S, Thomas B, Yelton P, Banfield J: **Community-wide analysis of microbial genome sequence signatures**. *Genome Biol* 2009, **10**(8):R85.

[168] Diaz N, Krause L, Goesmann A, Niehaus K, Nattkemper T: **TACOA - Taxonomic classification of environmental genomic fragments using a kernelized nearest neighbor approach**. *BMC Bioinformatics* 2009, **10**:56.

[169] McHardy A, Martin H, Tsirigos A, Hugenholtz P, Rigoutsos I: **Accurate phylogenetic classification of variable-length DNA fragments**. *Nat Methods* 2006, **4**:63–72.

[170] Abe T, Sugawara H, Kinouchi M, Kanaya S, Ikemura T: **Novel phylogenetic studies of genomic sequence fragments derived from uncultured microbe mixtures in environmental and clinical samples**. *DNA Res* 2005, **12**(5):281.

[171] Sandberg R, Winberg G, Branden CI, Kaske A, Ernberg I, Coster J: **Capturing Whole-Genome Characteristics in Short Sequences Using a Naive Bayesian Classifier**. *Genome Res* 2001, **11**(8):1404–1409.

[172] Brady A, Salzberg S: **Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models**. *Nat Methods* 2009, **6**(9):673–676.

[173] Chatterji S, Yamazaki I, Bai Z, Eisen J: **CompostBin: A DNA Composition-Based Algorithm for Binning Environmental Shotgun Reads**. In *Research in Computational Molecular Biology* 2008:17–28.

[174] Kislyuk A, Bhatnagar S, Dushoff J, Weitz J: **Unsupervised statistical clustering of environmental shotgun sequences**. *BMC Bioinformatics* 2009, **10**:316.

[175] Chan CKK, Hsu A, Tang SL, Halgamuge S: **Using growing self-organising maps to improve the binning process in environmental whole-genome shotgun sequencing**. *J Biomed Biotechnol* 2008, **2008**.

[176] Wu YW, Ye Y: **A Novel Abundance-Based Algorithm for Binning Metagenomic Sequences Using l-Tuples**. In *Research in Computational Molecular Biology*, *Volume 6044 of* Lecture Notes in Computer Science. Edited by Berger B, Springer Berlin / Heidelberg 2010:535–549.

[177] Bohlin J, Skjerve E, Ussery D: **Reliability and applications of statistical methods based on oligonucleotide frequencies in bacterial and archaeal genomes**. *BMC Genomics* 2008, **9**:104.

[178] Smyth P: **Clustering sequences with hidden Markov models**. In *Advances in Neural Information Processing Systems*, *Volume 9* 1997:648–654.

[179] Durbin R, Eddy S, Krogh A, Mitchison G: *Biological sequence analysis*. Cambridge University Press 1998.

[180] Delcher AL, Harmon D, Kasif S, White O, Salzberg SL: **Improved microbial gene identification with GLIMMER**. *Nucleic Acids Res* 1999, **27**(23):4636–4641.

[181] Celeux G, Govaert G: **A classification EM algorithm for clustering and two stochastic versions**. *Computational Statistics and Data Analysis* 1992, **14**(3):315–332.

[182] **LikelyBin** [http://ecotheory.biology.gatech.edu/downloads/likelybin].

[183] Shi J, Malik J: **Normalized cuts and image segmentation**. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2000, **22**(8):888–905.

[184] **CompostBin** [http://bobcat.genomecenter.ucdavis.edu/souravc/compostbin].

[185] **Scimm** [http://www.cbcb.umd.edu/software/scimm].

[186] Tan P, Steinbach M, Kumar V: *Introduction to Data Mining*. Addison-Wesley 2006.

[187] Hubert L, Arabie P: **Comparing partitions**. *Journal of Classification* 1985, **2**:193–218.

[188] Morgan J, Darling A, Eisen J: **Metagenomic sequencing of an in vitro-simulated microbial community**. *PloS ONE* 2010, **5**(4):e10209.

[189] White J, Navlakha S, Nagarajan N, Ghodsi M, Kingsford C, Pop M: **Alignment and clustering of phylogenetic markers- implications for microbial diversity studies**. *BMC Bioinformatics* 2010, **11**:152.

[190] Lozupone C, Knight R: **Global patterns in bacterial diversity**. *Proc Natl Acad Sci USA* 2007, **104**(27):11436.

[191] Whitman W, Coleman D, Wiebe W: **Prokaryotes: the unseen majority**. *Proc Natl Acad Sci USA* 1998, **95**(12):6578.

[192] Curtis T, Sloan W, Scannell J: **Estimating prokaryotic diversity and its limits**. *Proc Natl Acad Sci USA* 2002, **99**(16):10494.

[193] Turnbaugh P, Ley R, Hamady M, Fraser-Liggett C, Knight R, Gordon J: **The human microbiome project**. *Nature* 2007, **449**(7164):804–810.

[194] Schloss P, Handelsman J: **Metagenomics for studying unculturable microorganisms: cutting the Gordian knot**. *Genome Biol* 2005, **6**(8):229.

[195] Tringe S, Von Mering C, Kobayashi A, Salamov A, Chen K, Chang H, Podar M, Short J, Mathur E, Detter J, et al.: **Comparative metagenomics of microbial communities**. *Science* 2005, **308**(5721):554.

[196] Yooseph S, Sutton G, Rusch D, Halpern A, Williamson S, Remington K, Eisen J, Heidelberg K, Manning G, Li W, et al.: **The Sorcerer II Global Ocean Sampling Expedition: expanding the universe of protein families**. *PLoS Biol* 2007, **5**(3):e16.

[197] Dinsdale E, Edwards R, Hall D, Angly F, Breitbart M, Brulc J, Furlan M, Desnues C, Haynes M, Li L, et al.: **Functional metagenomic profiling of nine biomes**. *Nature* 2008, **452**(7187):629–632.

[198] Brulc J, Antonopoulos D, Berg Miller M, Wilson M, Yannarell A, Dinsdale E, Edwards R, Frank E, Emerson J, Wacklin P, et al.: **Gene-centric metagenomics of the fiber-adherent bovine rumen microbiome reveals forage specific glycoside hydrolases**. *Proc Natl Acad Sci USA* 2009, **106**(6):1948.

[199] Kristiansson E, Hugenholtz P, Dalevi D: **ShotgunFunctionalizeR: an R-package for functional comparison of metagenomes**. *Bioinformatics* 2009, **25**(20):2737.

[200] Sharon I, Bercovici S, Pinter R, Shlomi T: **Pathway-Based Functional Analysis of Metagenomes**. *J Comput Biol* 2011, **18**(3):495–505.

[201] Fickett JW, Tung CS: **Assessment of protein coding measures**. *Nucleic Acids Res* 1992, **20**(24):6441–6450.

[202] Hoff KJ: **The effect of sequencing errors on metagenomic gene predictionw**. *BMC Genomics* 2009, **10**:520+.

[203] Noguchi H, Taniguchi T, Itoh T: **MetaGeneAnnotator: detecting species-specific patterns of ribosomal binding site for precise gene prediction in anonymous prokaryotic and phage genomes**. *DNA Res* 2008, **15**(6):387–396.

[204] Hoff KJ, Lingner T, Meinicke P, Tech M: **Orphelia: predicting genes in metagenomic sequencing reads**. *Nucleic Acids Res* 2009, **37**(suppl_2):W101–105.

[205] Zhu W, Lomsadze A, Borodovsky M: **Ab initio gene identification in metagenomic sequences.** *Nucleic Acids Res* 2010, **38**(12):e132+.

[206] Rho M, Tang H, Ye Y: **FragGeneScan: predicting genes in short and error-prone reads**. *Nucleic Acids Res* 2010, **38**(20):e191.

[207] Hu GQ, Guo JT, Liu YC, Zhu H: **MetaTISA: Metagenomic Translation Initiation Site Annotator for improving gene start prediction**. *Bioinformatics* 2009, **25**(14):1843–1845.

[208] Turnbaugh P, Hamady M, Yatsunenko T, Cantarel B, Duncan A, Ley R, Sogin M, Jones W, Roe B, Affourtit J, et al.: **A core gut microbiome in obese and lean twins**. *Nature* 2009, **457**(7228):480–484.

[209] Benson D, Karsch-Mizrachi I, Lipman D, Ostell J, Sayers E: **GenBank**. *Nucleic Acids Res* 2011, **39**(Database Issue):D32–D37.

[210] Pruitt K, Tatusova T, Klimke W, Maglott D: **NCBI Reference Sequences: current status, policy and new initiatives**. *Nucleic Acids Res* 2009, **37**(Database issue):D32.

[211] **ELPH** [http://cbcb.umd.edu/software/ELPH].

[212] Angly F, Willner D, Prieto-Davó A, Edwards R, Schmieder R, Vega-Thurber R, Antonopoulos D, Barott K, Cottrell M, Desnues C, et al.: **The GAAS metagenomic tool and its estimations of viral and microbial average genome size in four major biomes**. *PLoS Comput Biol* 2009, **5**(12):e1000593.

[213] Balzer S, Malde K, Lanzén A, Sharma A, Jonassen I: **Characteristics of 454 pyrosequencing data—enabling realistic simulation with flowsim**. *Bioinformatics* 2010, **26**(18):i420.