ABSTRACT

Title of dissertation: Genome Assembly:

Novel Applications
by Harnessing Emerging Sequencing Technologies
and Graph Algorithms

Sergey Koren, Doctor of Philosophy, 2012

Dissertation directed by: Professor Mihai Pop
Department of Computer Science

Genome assembly is a critical first step for biological discovery. All current sequencing technologies share the fundamental limitation that segments read from a genome are much shorter than even the smallest genomes. Traditionally, whole-genome shotgun (WGS) sequencing over-samples a single clonal (or inbred) target chromosome with segments from random positions. The amount of over-sampling is known as the coverage. Assembly software then reconstructs the target. So called next-generation (or second-generation) sequencing has reduced the cost and increased throughput exponentially over first-generation sequencing. Unfortunately, next-generation sequences present their own challenges to genome assembly: (1) they require amplification of source DNA prior to sequencing leading to artifacts and biased coverage of the genome; (2) they produce relatively short reads: 100bp–700bp; (3) the sizeable runtime of most second-generation instruments is prohibitive for applications requiring rapid analysis, with an Illumina HiSeq 2000 instrument

requiring 11 days for the sequencing reaction.

Recently, successors to the second-generation instruments (third-generation) have become available. These instruments promise to alleviate many of the downsides of second-generation sequencing and can generate multi-kilobase sequences. The long sequences have the potential to dramatically improve genome and transcriptome assembly. However, the high error rate of these reads is challenging and has limited their use. To address this limitation, we introduce a novel correction algorithm and assembly strategy that utilizes shorter, high-identity sequences to correct the error in single-molecule sequences. Our approach achieves over 99% read accuracy and produces substantially better assemblies than current sequencing strategies.

The availability of cheaper sequencing has made new sequencing targets, such as multiple displacement amplified (MDA) single-cells and metagenomes, popular. Current algorithms assume assembly of a single clonal target, an assumption that is violated in these sequencing projects. We developed Bambus 2, a new scaffolder that works for metagenomics and single cell datasets. It can accurately detect repeats without assumptions about the taxonomic composition of a dataset. It can also identify biological variations present in a sample. We have developed a novel end-to-end analysis pipeline leveraging Bambus 2. Due to its modular nature, it is applicable to clonal, metagenomic, and MDA single-cell targets and allows a user to rapidly go from sequences to assembly, annotation, genes, and taxonomic info. We have incorporated a novel viewer, allowing a user to interactively explore the variation present in a genomic project on a laptop.

Together, these developments make genome assembly applicable to novel targets while utilizing emerging sequencing technologies. As genome assembly is critical for all aspects of bioinformatics, these developments will enable novel biological discovery.

Genome Assembly: Novel Applications by Harnessing Emerging Sequencing Technologies and Graph Algorithms

by

Sergey Koren

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
2012

Advisory Committee:
Professor Mihai Pop, Chair/Advisor
Professor Alan Sussman
Professor Carl Kingsford
Professor Hal Daumè III
Professor Najib M. El-Sayed (Dean's Representative)

Preface

This dissertation is based, in part, on the following publications, listed by chapter:

Chapter 1

Miller JR, **Koren S**, and Sutton G. Assembly algorithms for next-generation sequencing data. *Genomics* 95(6):315-327, 2010.

Chapter 2

Koren S, Miller JR, Walenz BP, and Sutton G. An algorithm for automated closure during assembly. *BMC Bioinformatics*, 11(1):457, 2010.

Chapter 3

Koren S, Schatz MC, Walenz BP, Martin J, Howard J, Ganapathy G, Wang Z, Raska DA, McCombie WR, Jarvis ED, and Phillippy AM. Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nature Biotechnology*. 2012. In Review.

Chapter 4

Koren S, Treangen TJ, and Pop M. Bambus 2: scaffolding metagenomes. *Bioinformatics*, 27(21):29642971, 2011.

Chapter 5

Treangen TJ*, **Koren S***, Sommer D, Liu B, Astrovskaya I, Darling AE, and Pop M. metAMOS: A modular open source metagenomic assembly and analysis pipeline. In Prep.

*Equal contribution.

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Mihai Pop for giving me an invaluable opportunity to work on challenging and extremely interesting projects. It has been a pleasure to work with and learn from such an extraordinary individual.

I would like to thank all of my collaborators who contributed to the work presented in this dissertation. Many helped coauthor the publications on which this dissertation is based. My coworkers at JCVI and TIGR who took a chance on a computer scientist and introduced me to bioinformatics, especially Jason Miller, Brian Walenz, and Granger Sutton. Thank you for sharing your wisdom and experience and guiding my first forays into biology and bioinformatics. It was through my experiences that I was inspired to pursue my PhD. Many thanks to my colleagues at NBACC, Brian Ondov and Nick Bergman. Thank you especially to Adam Phillippy for his guidance and support. A special thank you to both Adam and Mike Schatz for their thorough revisions to my manuscripts and advice, I have learned so much from both of you and am better for it.

My colleagues at the Pop lab have enriched my graduate life in many ways and deserve a special mention. Bo Liu, Henry Lin, Mohammadreza Ghodsi, Irina Astrovskaya, Dan Sommer, Christopher Hill, Chengxi Ye, Lee Mendelowitz, and Joseph Paulson. I will miss our invaluable discussions. I would also like to especially

thank Todd Treangen who helped polish and focus my thoughts on presenting results and pushed me to complete my publications.

I owe thanks to my family - my mother and father who have always stood by me and guided me through my career. I would also like to thank my wife, Kate Koren, who provided endless support throughout my time in graduate school and read countless drafts. Words cannot express the gratitude I owe them.

Finally, I'd like to thank my committee, Dr. Mihai Pop, Dr. Alan Sussman, Dr. Carl Kingsford, Dr. Hal Daumè III, and Dr. Najib M. El-Sayed for their support and careful reading of this dissertation.

It is impossible to remember all, and I apologize to those I've inadvertently left out.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

CCS	Circular Consensus Sequence
DBG	de Bruijn Graph
HMP	Human Microbiome Project
HTML	Hyper-Text Markup Language
MDA	Multiple Displacement Amplification
MetaHIT	Metagenomics of the Human Intestinal Tract
NGS	Next-generation Sequencing
OLC	Overlap/Layout/Consensus
PBcR	PacBio Corrected Reads
PCR	Polymerase Chain Reaction
WGA	Whole Genome Shotgun Assembly
WGS	Whole Genome Shotgun

Chapter 1

Introduction[†]

The advent of short-read sequencing machines gave rise to a new generation of assembly algorithms and software. Here, we introduce algorithms for *de novo* whole-genome shotgun assembly from next-generation sequencing data. We use a narrow definition of *de novo* whole-genome shotgun assembly. The shotgun process takes reads from random positions along a target molecule [138]. Whole-genome shotgun (WGS) sequencing samples the chromosomes that make up one genome. WGS assembly is the reconstruction of sequence up to chromosome length. The assembly task is relegated to computer software [155]. Assembly is possible when the target is over-sampled by the shotgun reads, such that reads overlap. *De novo* WGS assembly refers to reconstruction in its pure form, without consultation to previously resolved sequence including from genomes, transcripts, and proteins. Broader introductions can be found elsewhere, *e.g.* [125].

Today's commercial DNA sequencing platforms include the Genome Sequencer from Roche 454 Life Sciences [97], the Solexa Genome Analyzer from Illumina [8], the SOLiD System from Applied Biosystems. These platforms have been well reviewed, *e.g.* [96, 107, 156, 119]. A distinguishing characteristic of these platforms is that

[†]The text of this chapter is based on the publication J. Miller, S. Koren, and G. Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–327, 2010. Only sections coauthored by Sergey Koren and Jason Miller are included. Sections 1.7 onward represent unpublished text.

they do not rely on Sanger chemistry [139] as did first-generation machines including the Applied Biosystems Prism 3730 and the Molecular Dynamics MegaBACE. The second-generation machines are characterized by highly parallel operation, higher yield, simpler operation, much lower cost per read, and (unfortunately) shorter reads. Today's machines are commonly referred to as short-read sequencers or next-generation sequencers (NGS) though their successors have become available. Pacific Biosciences' machine [28] can produce reads longer than first-generation machines. First-generation reads were commonly 500*bp* to 1000*bp* long. Today's NGS reads are in the 700*bp* range (from 454 machines), the 100*bp* range (from the Illumina machines), or less. Shorter reads deliver less information per read, confounding the computational problem of assembly. Assembly of shorter reads requires higher coverage, in part to satisfy minimum detectable overlap criteria. High coverage increases complexity and intensifies computational issues related to large data sets. All sequencers produce observations of the target DNA molecule in the form of reads: sequences of single-letter base calls plus a numeric quality value (QV) for each base call [31]. Although QVs offer extra information, their use generally increases a program's CPU and RAM requirements.

The NGS platforms have characteristic error profiles that change as the technologies improve. Error profiles can include enrichment of base call error toward the 3' (terminal) ends of reads, compositional bias for or against high-GC sequence, and inaccurate determination of simple sequence repeats. There are published error profiles for the 454 GS 20 [60], the Illumina 1G Analyzer [24], and comparisons of three platforms [55]. Some NGS software is tuned for platform-specific error profiles.

Others may have unintentional bias where development targeted one data type.

Sanger platforms could deliver paired-end reads, that is, pairs of reads with a constraint on their relative orientation and separation in the target. Paired ends were essential to assembly of cellular genomes small [35] and large [2] due to their ability to span repeats longer than individual reads. Paired ends, also called mate pairs, have a separation estimate that is usually provided to software as the fragment size distribution measured on a so-called library of reads. A sufficient variety of paired end separations should help resolve large chromosomes to single scaffolds [150]. The exploitation of paired ends (as well as other contig constraints) within assembly is generally referred to as the scaffolding problem and is discussed in Section 1.7.

1.1 What is an assembly?

An assembly is a hierarchical data structure that maps the sequence data to a putative reconstruction of the target. It groups reads into contigs and contigs into scaffolds. Contigs provide a multiple sequence alignment of reads plus the consensus sequence. The scaffolds, sometimes called supercontigs or metacontigs, define the contig order and orientation and the sizes of the gaps between contigs. Scaffold topology may be a simple path or a network. Most assemblers output, in addition, a set of unassembled or partially assembled reads. The most widely accepted data file format for an assembly is FastA, wherein contig consensus sequence can be represented by strings of the characters A, C, G, T, plus possibly other characters with special meaning. Dashes, for instance, can represent extra bases omitted from

the consensus but present in a minority of the underlying reads. Scaffold consensus sequence may have N's in the gaps between contigs. The number of consecutive N's may indicate the gap length estimate based on spanning paired ends.

Assemblies are measured by the size and accuracy of their contigs and scaffolds. Assembly size is usually given by statistics including maximum length, average length, combined total length, and N50. The contig N50 is the length of the smallest contig in the set that contains the fewest (largest) contigs whose combined length represents at least 50% of the assembly. The N50 statistics for different assemblies are not comparable unless each is calculated using the same combined length value. Assembly accuracy is difficult to measure. Some inherent measure of accuracy is provided by the degrees of mate-constraint satisfaction and violation [123]. Alignment to reference sequences is useful whenever trusted references exist.

1.2 The challenge of assembly

DNA sequencing technologies share the fundamental limitation that read lengths are much shorter than even the smallest genomes. WGS overcomes this limitation by over-sampling the target genome with short reads from random positions. Assembly software reconstructs the target sequence.

Assembly software is challenged by repeat sequences in the target. Genomic regions that share perfect repeats can be indistinguishable, especially if the repeats are longer than the reads. For repeats that are inexact, high-stringency alignment can separate the repeat copies. Careful repeat separation involves correlating reads

by patterns in the different base calls they may have [66]. Repeat separation is assisted by high coverage but confounded by high sequencing error. For repeats whose fidelity exceeds that of the reads, repeat resolution depends on spanners, that is, single reads that span a repeat instance with sufficient unique sequence on either side of the repeat. Repeats longer than the reads can be resolved by spanning paired ends, but the analysis is more complicated. Complete resolution usually requires two resources: pairs that straddle the repeat with each end in unique sequence, and pairs with exactly one end in the repeat. The limit of repeat resolution can be explored for finished genomes under some strict assumptions. For instance, it was shown that the theoretical best assembly of the *E. coli* genome from 20 bp unpaired reads would put 10% of bases in contigs of 10 Kbp or longer given infinite coverage and error-free reads [178]. The limit calculation is not straightforward for reads with sequencing error, paired-end reads, or unfinished genomes. Careful estimates of repeat resolution involve the ratio of read length (or paired-end separation) to repeat length, repeat fidelity, read accuracy, and read coverage. In regard to NGS data, shorter reads have less power to resolve genomic repeats but higher coverage increases the chance of spanning short repeats.

Repeat resolution is made more difficult by sequencing error. Software must tolerate imperfect sequence alignments to avoid missing true joins. Error tolerance leads to false positive joins. This is a problem especially with reads from inexact (polymorphic) repeats. False-positive joins can induce chimeric assemblies. In practice, tolerance for sequencing error makes it difficult to resolve a wide range of genomic phenomena: polymorphic repeats, polymorphic differences between non-

clonal asexual individuals, polymorphic differences between non-inbred sexual individuals, and polymorphic haplotypes from one non-inbred individual. If the sequencing platforms ever generate error-free reads at high coverage, assembly software might be able to operate at 100% stringency.

WGS assembly is confounded by non-uniform coverage of the target. Coverage variation is introduced by chance, by variation in cellular copy number between source DNA molecules, and by compositional bias of sequencing technologies. Very low coverage induces gaps in assemblies. Coverage variability invalidates coverage-based statistical tests, and undermines coverage-based diagnostics designed to detect over-collapsed repeats.

WGS assembly is complicated by the computational complexity of processing larger volumes of data. For efficiency, all assembly software relies to some extent on the notion of a k-mer. This is a sequence of k base calls, where k is any positive integer. In most implementations, only consecutive bases are used. Intuitively, reads with high sequence similarity must share k-mers in their overlapping regions, and shared k-mers are generally easier to find than overlaps. Fast detection of shared k-mer content vastly reduces the computational cost of assembly, especially compared to all-against-all pair-wise sequence alignment. A tradeoff of k-mer based algorithms is lower sensitivity, thus missing some true overlaps. The probability that a true overlap spans shared k-mers depends on the value of k, the length of the overlap, and the rate of error in the reads. An appropriate value of k should be large enough that most false overlaps do not share k-mers by chance, and small enough that most true overlaps do. The choice should be robust to variation in read coverage and

accuracy.

WGS assembly algorithms, and their implementations, are typically complex. The general assembly problem has been shown to be NP-hard [111]. Several problems within assembly, such as scaffolding, are also NP-hard [42]. Therefore, most software relies on heuristics and known approximations to solve the problem.

1.3 Graph algorithms for assembly

We organize the assemblers into three categories, all based on graphs. The Overlap/Layout/Consensus (OLC) methods rely on an overlap graph. The de Bruijn Graph (DBG) methods use some form of k-mer graph. The greedy graph algorithms may use OLC or DBG.

A graph is an abstraction used widely in computer science. It is a set of nodes plus a set of edges between the nodes. Nodes and edges may also be called vertices and arcs, respectively. Importantly, each directed edge represents a connection from one source node to one sink node. Collections of edges form paths that visit nodes in some order, such that the sink node of one edge forms the source node for any subsequent nodes. A simple path may not intersect itself, by definition, and one may additionally require that no other paths intersect it. The nodes and edges may be assigned a variety of attributes and semantics.

An overlap graph represents the sequencing reads and their overlaps [108]. The overlaps must be pre-computed by a series of (computationally expensive) pair-wise sequence alignments. Conceptually, the graph has nodes to represent the reads and

edges to represent overlaps. In practice, the graph might have distinct elements or attributes to distinguish the 5' and 3' ends of reads, the forward and reverse complement sequences of reads, the lengths of reads, the lengths of overlaps, and the type of overlap (suffix-to-prefix or containment). Paths through the graph are the potential contigs, and paths can be converted to sequence. Paths may have mirror images representing the reverse complement sequence. There are two ways to force paths to obey the semantics of double-stranded DNA. If the graph has separate nodes for read ends, then paths must exit the opposite end of the read they enter. If the graph has separate edges for the forward and reverse strands, then paths must exit a node on the same strand they enter.

The de Bruijn graph was developed outside the realm of DNA sequencing to represent strings from a finite alphabet. The nodes represent all possible fixed-length strings. The edges represent suffix-to-prefix perfect overlaps.

A k-mer graph is a form of de Bruijn graph. Its nodes represent all the fixed-length subsequences drawn from a larger sequence. Its edges represent all the fixed-length overlaps between subsequences that were consecutive in the larger sequence. In one formulation [62], there is one edge for the k-mer that starts at each base (excluding the last $k-1$ bases). The nodes represent overlaps of $k-1$ bases. Alternately [183], there is one node representing the k-mer that starts at each base. The edges represent overlaps of $k-1$ bases. By construction, the graph contains a path corresponding to the original sequence (Fig 1.1). The path converges on itself at graph elements representing k-mers in the sequence whose multiplicity is greater than one.

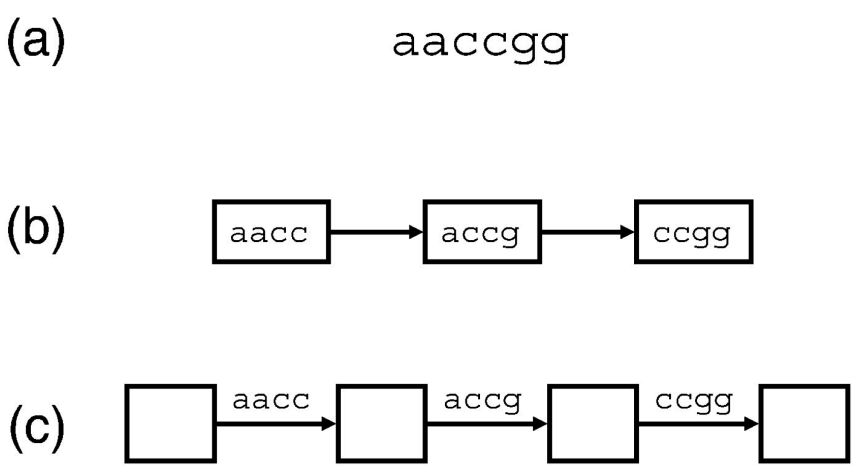


Figure 1.1: **A read represented by k-mer graphs.** (a) The read is represented by two types of k-mer graph with $k = 4$. Larger values of k are used for real data. (b) The graph has a node for every k-mer in the read plus a directed edge for every pair of k-mers that overlap by $k - 1$ bases in the read. (c) An equivalent graph has an edge for every k-mer in the read and the nodes implicitly represent overlaps of $k - 1$ bases. In these examples, the paths are simple because the value $k = 4$ is larger than the 2 bp repeats in the read. The read sequence is easily reconstructed from the path in either graph.

A k-mer graph may represent many sequences instead of one. In its application to WGS assembly, the k-mer graph represents the input reads. Each read induces a path. Reads with perfect overlaps induce a common path. Thus, perfect overlaps are detected implicitly without any pair-wise sequence alignment calculation (Fig 1.2). Compared to overlap graphs, k-mer graphs are more sensitive to repeats and sequencing errors. Paths in overlap graphs converge at repeats longer than a read, but paths in k-mer graphs converge at perfect repeats of length k or more, and k must be less than the read length. Each single-base sequencing error induces up to k false nodes in the k-mer graph. Each false node has a chance of matching some other node and thereby inducing a false convergence of paths.

Real-world WGS data induces problems in overlap graphs and k-mer graphs.

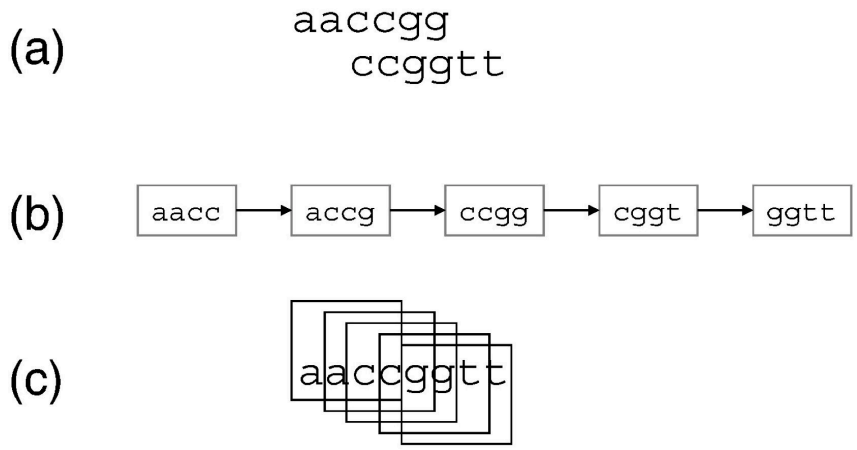


Figure 1.2: **A pair-wise overlap represented by a k-mer graph.** (a) Two reads have an error-free overlap of 4 bases. (b) One k-mer graph, with $k = 4$, represents both reads. The pair-wise alignment is a by-product of the graph construction. (c) The simple path through the graph implies a contig whose consensus sequence is easily reconstructed from the path.

- Spurs are short, dead-end divergences from the main path (Fig 1.3(a)). They are induced by sequencing error toward one end of a read. They can be induced by coverage dropping to zero.

- Bubbles are paths that diverge then converge (Fig 1.3(b)). They are induced by sequencing error toward the middle of a read, and by polymorphism in the target. Efficient bubble detection is non-trivial [32].

- Paths that converge then diverge form the frayed rope pattern (1.3(c)). They are induced by repeats in the target genome.

- Cycles are paths that converge on themselves. They are induced by repeats in the target. For instance, short tandem repeats induce small cycles.

In general, branching and convergence increases graph complexity, leading to tangles that are difficult to resolve. Much of the complexity is due to repeats in the

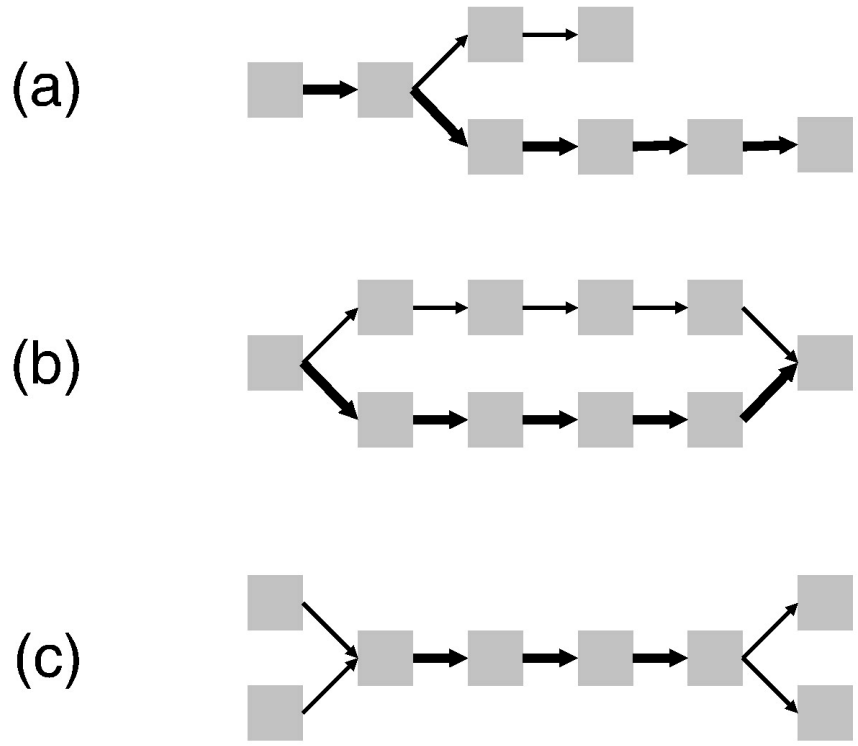


Figure 1.3: **Complexity in l-mer graphs can be diagnosed with read multiplicity information.** In these graphs, edges represented in more reads are drawn with thicker arrows. (a) An errant base call toward the end of a read causes a spur or short dead-end branch. The same pattern could be induced by coincidence of zero coverage after polymorphism near a repeat. (b) An errant base call near a read middle causes a bubble or alternate path. Polymorphisms between donor chromosomes would be expected to induce a bubble with parity of read multiplicity on the divergent paths. (c) Repeat sequences lead to the frayed rope pattern of convergent and divergent paths.

target and sequencing error in the reads. In the graph context, assembly is a graph reduction problem. Assemblers often rely on heuristic algorithms and approximation algorithms to remove redundancy, repair errors, reduce complexity, enlarge simple paths and otherwise simplify the graph.

1.4 Greedy Graph-based Assemblers

The first NGS assembly packages used greedy algorithms. These have been reviewed elsewhere [125, 128]. The greedy algorithms apply one basic operation: given any read or contig, add one more read or contig. The basic operation is repeated until no more operations are possible. Each operation uses the next highest-scoring overlap to make the next join. The scoring function measures, for instance, the number of matching bases in the overlap. Thus the contigs grow by greedy extension, always taking on the read that is found by following the highest-scoring overlap. The greedy algorithms can get stuck at local maxima if the contig at hand takes on reads that would have helped other contigs grow even larger.

The greedy algorithms are implicit graph algorithms. They drastically simplify the graph by considering only the high-scoring edges. As an optimization, they may actually instantiate just one overlap for each read end they examine. They may also discard each overlap immediately after contig extension. Like all assemblers, the greedy algorithms need mechanisms to avoid incorporating false-positive overlaps into contigs. Overlaps induced by repetitive sequence may score higher than overlaps induced by common position of origin. An assembler that builds on false-positive

overlaps will join unrelated sequences to either side of a repeat to produce chimera.

1.5 Overlap/Layout/Consensus assemblers

The OLC approach was typical of the Sanger-data assemblers. It was optimized for large genomes in software including Celera Assembler [110], Arachne [7, 64], and CAP and PCAP [59]. The OLC approach has been reviewed elsewhere [124, 6, 157].

OLC assemblers use an overlap graph. Their operation has three phases:

1. Overlap discovery involves all-against-all, pair-wise read comparison. The seed and extend heuristic algorithm is used for efficiency. The software precomputes k-mer content across all reads, selects overlap candidates that share k-mers, and computes alignments using the k-mers as alignment seeds. Overlap discovery is sensitive to settings of k-mer size, minimum overlap length, and minimum percent identity required for an overlap. These three parameters affect robustness in the face of base calling error and low-coverage sequencing. Larger parameter values lead to more accurate but shorter contigs. Overlap discovery can run in parallel with a matrix partition.

2. Construction and manipulation of an overlap graph leads to an approximate read layout. The overlap graph need not include the sequence base calls, so large-genome graphs may fit into practical amounts of computer memory.

3. Multiple sequence alignment (MSA) determines the precise layout and then the consensus sequence. There is no known efficient method to compute the

optimal MSA [172]. Therefore, the consensus phase uses progressive pairwise alignments guided by, for instance, the approximate read layout. The consensus phase must load the sequence base calls into memory. It can run in parallel, partitioned by contig.

1.6 The de Bruijn graph approach

The third approach to assembly is most widely applied to the short reads from the Illumina and SOLiD platforms. It relies on k-mer graphs, whose attributes make it attractive for vast quantities of short reads. The k-mer graph does not require all-against-all overlap discovery, it does not (necessarily) store individual reads or their overlaps, and it compresses redundant sequence. Conversely, the k-mer graph does contain actual sequence and the graph can exhaust available memory on large genomes. Distributed memory and compression approaches may alleviate this constraint.

The k-mer graph approach dates to an algorithm for Sanger read assembly [62] based on a proposal [120] for an even older sequencing technology; see [125] for review. The approach is commonly called a de Bruijn graph (DBG) approach or an Eulerian approach [122] based on an ideal scenario. Given perfect data (error-free k-mers providing full coverage and spanning every repeat) the k-mer graph would be a de Bruijn graph and it would contain an Eulerian path, that is, a path that traverses each edge exactly once. The path would be trivial to find making the assembly problem trivial by extension. Of course, k-mer graphs built from real

sequencing data are more complicated. To the extent that the data is ideal, assembly is a by-product of the graph construction. The graph construction phase proceeds quickly using a constant-time hash table lookup for the existence of each k-mer in the data stream. Although the hash table consumes extra memory, the k-mer graph itself stores each k-mer at most once, no matter how many times the k-mer occurs in the reads. In terms of computer memory, the graph is smaller than the input reads to the extent that the reads share k-mers.

Pevzner [120] explored problems that genomic repeats introduce. Repeats induce cycles in the k-mer graph. These would allow more than one possible reconstruction of the target. Idury and Waterman [62] also explored problems of real data. They added two extra types of information to the k-mer graph and named the result a sequence graph. Each edge was labeled with the reads, and positions within each read, of the sequences that induced it. Where nodes had one inbound and one outbound edge, the three elements could be compressed into one edge. This was called the elimination of singletons. Further research led to the Euler software implementation [121] for Sanger data. Impractical for large-scale Sanger sequencing projects, Euler and the DBG approach were well positioned when the Illumina platform started to produce data composed of very short unpaired reads of uniform size.

Three factors complicate the application of k-mer graphs to DNA sequence assembly.

1. DNA is double stranded. The forward sequence of any given read may overlap

the forward or reverse complement sequence of other reads. One k-mer graph implementation contains nodes and edges for both strands, taking care to avoid output of the entire assembly twice [62].

2. Real genomes present complex repeat structures including tandem repeats, inverted repeats, imperfect repeats, and repeats inserted within repeats. Repeats longer than k lead to tangled k-mer graphs that complicate the assembly problem. Perfect repeats of length k or greater collapse inside the graph, leaving a local graph structure that resembles a rope with frayed ends (Fig 1.3(c)); paths converge for the length of the repeat and then they diverge. Successful assembly requires separation of the converged path, which represents a collapsed repeat. The graph contains insufficient information to disambiguate the repeat. Assemblers typically consult the reads, and possibly the mate pairs, in attempts to resolve these regions.

3. A palindrome is a DNA sequence that is its own reverse complement. Palindromes induce paths that fold back on themselves.

4. Real data includes sequencing error. DBG assemblers use several techniques to reduce sensitivity to this problem. First, they pre-process the reads to remove error. Second, they weight the graph edges by the number of reads that support them, and then erode the lightly supported paths. Third, they convert paths to sequences and use sequence alignment algorithms to collapse nearly identical paths. Many of these techniques derive from the Euler family of assemblers.

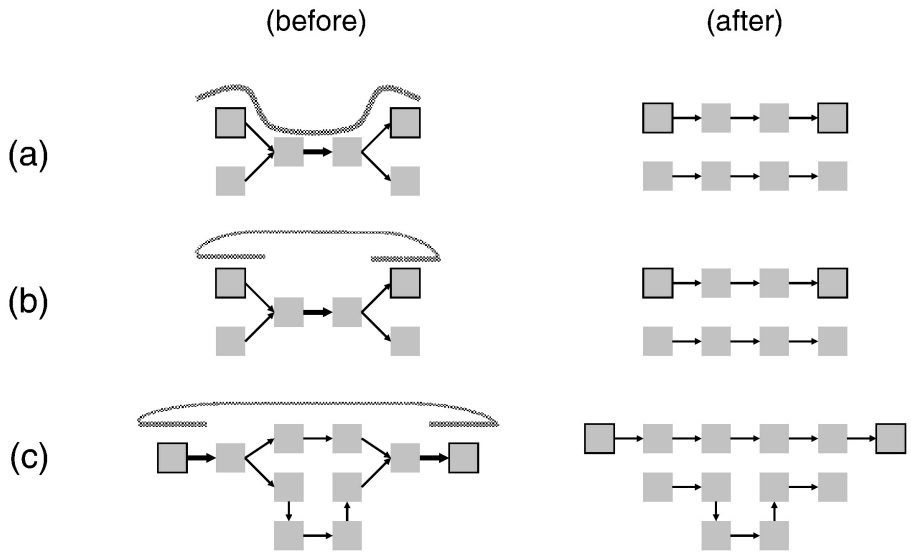


Figure 1.4: **Three methods to resolve graph complexity.** (a) Read threading joins paths across collapsed repeats that are shorter than the read lengths. (b) Mate threading joins paths across collapsed repeats that are shorter than the paired-end distances. (c) Path following chooses one path if its length fits the paired-end constraint. Reads and mates are shown as patterned lines. Not all tangles can be resolved by reads and mates. The non-branching paths are illustrative; they could be simplified to single edges or nodes.

By processing k-mers not reads, the k-mer graph construction discards long-range continuity information in the reads. This defect can be repaired by threading the reads through the graph. Reads ending inside a repeat are consistent with any path exiting the repeat, but reads spanning a repeat are consistent with fewer paths. For the latter, read threading pulls out one piece of string from a frayed rope pattern, thus resolving one copy of the collapsed repeat (Fig 1.4(a)). Thus read threading constrains the set of valid paths through the graph. This allows resolution of repeats whose length is between k and the read length. A paired-end read is effectively a long read that is missing base calls in the middle. Paired ends that span a repeat provide the evidence to join one path that enters a repeat to one path that exits the repeat (Fig 1.4(b)). Paired ends can also resolve some complex

tangles induced by repeats. A complex graph may have multiple paths between two nodes corresponding to opposite ends of a mate pair. Each path implies a putative DNA sequence. In many cases, only one of the paths implies a sequence whose length satisfies the paired-end constraint (Fig 1.4(c)). Between any mate pair, there could be too many paths for exhaustive search to be feasible, requiring heuristics.

Regardless of the assembly method used, the resulting contigs are generally short. Assemblers rely on paired-end sequences to build gapped sequence and scaffolds. We explore scaffolding in detail in the next section.

1.7 Scaffolding

After the unitigs are constructed, they may be viewed as the nodes in a new graph. At this stage, the nodes are commonly referred to as contigs. A contig may represent a single unitig or an un-gapped combination. Using the set of paired-end reads in a contig, assemblers bundle consistent pairs together into a weighted edge. Since each of the read pairs provides a distance and an orientation, each edge between two contigs has an orientation and a mean distance. The resulting graph is bi-directed. That is, there are four possible edges between any two nodes (Fig 1.5):

- Both arrows pointing inwards, away from the nodes. This is generally known as an innie edge.
- Both arrows pointing outward, toward the nodes. This is generally known as an outie edge.
- Left arrow pointing inward, right arrow pointing outward. This is generally

(a) Innie (b) Outie (c) Normal (d) Anti-Normal

Figure 1.5: The four types of edges present in a bi-directed graph of contigs.

known as a normal edge.

- Right arrow pointing inward, left arrow pointing outward. This is generally known as an anti-normal edge.

Generally, scaffolding starts by assigning an arbitrary orientation to a node and placing it at position 0. Nodes are then oriented consistent with their edges. Each node is also assigned a position along a linear axis. Whenever the edges indicate an overlap, two nodes may be joined. As a result of this operation the bi-directed graph is transformed into a directed one. Note that the innie and outie edges indicate that the two nodes are in opposite orientations (Fig 1.5(a), 1.5(b)). These edges are known as reverse-edges and it is not possible to properly orient the nodes if they are part of a cycle with an odd number of reverse-edges. A scaffolding module must minimize the number of edges removed from the graph. That is, we wish to ignore the minimum number of edges so that there are no remaining cycles with an odd number of reverse-edges. Unfortunately, finding such a minimum set is equivalent to the Maximal Bipartite Subgraph problem which is NP-hard [42]. Each connected component in a graph becomes a scaffold. Next, we need to generate a layout of contigs on a number line. We want to maximize the number of satisfied edges by

placing nodes as close to the specified position as possible. This problem is similar to the Optimal Linear Arrangement problem which is also NP-hard [42]. A final step is to project the graph form of a scaffold onto a linear sequence. Whenever two contigs occupy the same positions, they must be reconciled using sequence analysis. This can often lead to scaffolds being broken when two contigs cannot be reconciled.

1.8 Genome finishing

Despite the advances in genome assembly, the result is almost never a single contig per chromosome. The task of genome finishing is the process of "closing" a genome by filling in sequence missing from the assembly, confirming repeat sequence, and correcting assembly errors. Due to the expense, genome finishing is commonly only applied to prokaryotic genomes and often requires months of manual effort [36]. One common strategy for genome finishing is targeted sequencing. In contrast to WGS sequencing which samples the genome uniformly at random, targeted sequences are known to come from a specific region of the genome. These sequences are subsequently manually incorporated into an existing assembly to close gaps and resolve repeats.

1.9 Challenges of third-generation sequencing instruments

Recently, successors to next-generation sequencing instruments have become commercially available, starting with the PacBio RS [28]. The PacBio RS is a real-time single-molecule sequencer. It requires no amplification, eliminating potential

bias, while producing long sequences (*e.g.* median $\geq 2Kbp$) [130]. The unbiased long sequences hold great promise for simplifying genome assembly by resolving difficult repeats. The long sequences also hold the potential to sequence abundant transcripts in a single sequence. Unfortunately, the instrument averages only 82.1%[16]–84.6%[130] nucleotide accuracy. There is currently a lack of bioinformatics tool designed to handle this level of sequencing error, limiting the impact of this technology.

1.10 Metagenomics and single-cell genomics

Metagenomics is the sequencing of DNA in an environmental sample. Whereas WGS targets one clonal population, metagenomics usually targets several. By contrast, single-cell genomics targets the genome of a single cell. However, both single-cell and metagenomics present similar challenges to assembly. The assembly of metagenomic data is complicated by several factors such as: (i) widely different levels of representation for different organisms in a community; (ii) genomic variation between closely related organisms; (iii) conserved genomic regions shared by distantly related organisms; and (iv) repetitive sequences within individual genomes. Single-cell genomics relies on whole genome amplification by the multiple displacement amplification (MDA) method to allow sequencing of single cells [22, 149]. The assembly of single-cells is complicated by (i) widely different levels of representation for different portions of a genome (ii) amplified contaminating sequences from non-target genomes (iii) chimeric sequences introduced by the single-cell process

[85].

Current assemblers, developed for single-organism assembly, have been successfully used to assemble metagenomic data sets [136], [168]. However, these assemblies are not well specified, require manual intervention and undocumented settings, and are error prone. Simulations indicate high rates of chimera, especially in short contigs assembled from complex mixtures [98]. As discussed above, assemblers try to identify the genomic repeats for special handling. For example, the Celera Assembler [110] uses coverage, computed from the average number of bases between read start positions, to identify repeats. An increase in coverage is presumed to be an indicator of a repeat. The repeats may also be identified using local graph features such as convergent and divergent paths. However, in a metagenomic and single-cell context, these are not indicative of repeat structure.

Metagenomics is still a manually-intensive process. Biologists need to view how contigs are connected to see the paths representing organisms as well as areas of divergence. Current visualization is static [126] and the complex graph structure as well as poor node layout makes it difficult to interpret.

1.11 Previous work

We have previously explored the application of heuristic graph algorithms to the assembly problem. In Miller *et. al.* [105], we rely on a heuristic to simplify the overlap graph. Given the overlaps for a set of reads R, we construct a multi-graph G with both directed and undirected edges. Each read r is represented by a pair of

nodes $r_{5'}$ and $r_{3'}$, representing the two ends of a read, connected by an undirected edge. Directed edges represent dovetail overlaps, that is, those that span one end of each read. We simplified the graph using a simple heuristic, all directed edges except those that represent the mutually-best edge between any pair of read ends are removed. A mutually-best edge is defined as one that spans the most bases. This heuristic reduces the overlap graph to require storing only a single edge for each node. Finally, heuristics are applied to split contigs that were built incorrectly. This heuristic is shown to perform well in practice [105].

For the problem of multiple-sequence alignment, we also use a graph-based heuristic in Rausch *et. al.* [131]. A multi-alignment can be represented as a graph G. Vertices represent non-overlapping sequence segments, edges connect vertices and represent un-gapped alignments between reads while gaps are implicit. For a set S of n reads, G is n-partite. Every position in a read r is represented in exactly one vertex. Since some edges may be contradictory, only a subset of the edges E can be satisfied in a given alignment. A proper subset $E' \subset E$, called a trace, can be found efficiently using heuristics. The consensus tool built on this concept generates superior multi-alignments, especially in the case of high-error and short reads [131].

1.12 Contributions

In Chapter 2 we present a novel method for incorporating finishing sequences to improve genome assembly. The finishing of prokaryotic genomes is an expensive and time consuming manual process. We have automated an important step, the

incorporation of sequences targeted to fill gaps in a draft assembly. The resulting algorithm closes more gaps and achieves better repeat resolution in assemblies than previous approaches.

In Chapter 3 we present a novel method for utilizing high-error third-generation sequencing data to significantly improve genome assembly. In the best case, our method produces a 5–fold improvement over second-generation sequencing alone while reducing assembly error. Our method is widely applicable, and we demonstrate its use on assembly of both prokaryotic and eukaryotic genomes, as well as on eukaryotic RNA-seq analysis. Our results show that single-molecule long sequences can be accurately assembled, potentially ushering in the era of single-chromosome prokaryotic assembly.

In Chapter 4 we present a novel scaffolding method to address the challenges of uneven representation as well as genomic diversity. Our scaffolder, Bambus 2, performs as well as the state of the art assemblers targeting clonal datasets while significantly improving metagenomic assembly. Our repeat detection method is sensitive and accurate, without knowledge of the taxonomic composition of a dataset. We demonstrate increased assembly contiguity on three simulated and five real metagenomic datasets. Our method identifies biologically relevant genomic variations that have been previously observed. Finally, we demonstrate our method's applicability to single-cell genomes.

In Chapter 5, we present an end-to-end metagenomic analysis pipeline. We have developed a pipeline incorporating many third-party tools to assemble sequences, identify and annotate genes, analyze composition, and classify the assem-

bled results into their appropriate taxonomic class. The pipeline includes interactive visualization of taxonomic composition, using Krona [114], as well as assembly sequences. We include a novel visualization tool for the genomic variants present in a dataset that can allow a user to navigate a metagenome on their laptop.

Part I

Improving Clonal Genome Assembly

Chapter 2

An algorithm for automated closure during assembly[†]

2.1 Overview

The shotgun method generates reads randomly in high volumes by Sanger and next-generation sequencing platforms. Whole-genome shotgun assembly (WGA) is the process of constructing a draft assembly of a genome from whole-genome shotgun reads (WGS). WGA software constructs a read layout by inference from shared sequence between reads and constraints between pairs of reads from the same DNA fragment (paired-ends). The randomness of WGS can be exploited in software by adopting uniformity of read coverage as an objective function to be maximized by the assembly. For instance, the Celera Assembler software [110] invokes the A-stat coverage statistic to assign lower confidence to higher-coverage mini-assemblies. The Velvet software [183] invokes low-coverage to trim branches of its de Bruijn graph.

Finishing is the process of improving the quality and utility of a draft genome sequence. Finishing aims to fill gaps between contigs, enlarge contigs, or provide deeper coverage for the contigs in the draft. Some finishing is accomplished without sequencing by manually editing an automatically generated draft. Most finishing requires additional sequence referred to as finishing reads. Finishing reads derive

[†]The text of this chapter is based on the publication S. Koren, J. Miller, B. Walenz, and G. Sutton. An algorithm for automated closure during assembly. *BMC Bioinformatics*, 11(1):457, 2010.

from PCR, primer walking, transposon bombing, shotgun of individual clones, and other techniques. See [36] for a review.

Intuitively, a PCR experiment provides evidence that reads generated from within the PCR product should be assembled between instances of the primer sequences. The set of finishing reads derived from an individual amplicon, or clone, have a co-location requirement. End-reads from amplicons or clones provide the boundaries between which the finishing reads should assemble. Manual inspection and placement of finishing reads is expensive [115], even when assisted by software. The widely-used Consed package [51] provides assembly editing functionality through a graphical user interface. Dupfinisher [53] automates several finishing procedures, including homology-based search to identify repeats in the draft assembly. As an example, Consed and Dupfinisher were invoked during the finishing stage of the *Pedobacter heparinus* genome project [54]. After 44K WGS reads had been assembled with Phrap (http://www.phrap.org), Dupfinisher corrected mis-assemblies. Then, with 1 897 finishing reads, Consed and manual editing were used to close gaps and improve quality. Both of these methods are *a posteriori*, running after an assembly has been generated. They require users to specify a gap for each finishing read. In contrast, we introduce an algorithm for placing finishing reads within the context of a WGA.

Our method generates a *de novo* assembly in one process that integrates the WGS and finishing reads. We only require the identifiers for the end-reads from the amplicons or clones, potentially improving usability. The algorithm uses the sets of finishing reads and placement bounds for each set to incorporate finishing reads

during the assembly process. Our algorithm targets repetitive genomic regions, seeking to close and thicken repeat-induced gaps, as well as to locate repeat copies missed in the initial assembly.

The algorithm was challenged to assemble five prokaryotes and one eukaryotic genome from WGS and finishing reads. The results were compared to assemblies of all the reads input as WGS reads. The results were also compared to an alternate assembly pipeline.

2.2 Methods

The input has three components: WGS reads, finishing reads, and bounding constraints. The reads may include paired-end reads, such that any given read pair consists of two WGS reads or two finishing reads. The bounding constraint is usually a paired end thought to span the target of the finishing reactions, based on a WGS assembly. Alternately, it could be any two reads whose sequence encompasses the PCR primers that generated the template for the finishing reads, or the PCR primers themselves. Formally, given the set of bounded finishing reads F and WGS reads W, the finishing reads and bounding constraints must satisfy the following conditions:

- For each finishing read $f \in F$, there exists at most one pair of sequences, $f_a \in (W \cup F)$ and $f_b \in (W \cup F)$ s.t. $f_a \neq f_b$, $f \neq f_a$, and $f \neq f_b$ (referred to as the bounding constraint for f).

- For each bounding constraint (f_a, f_b), there exists an non-empty set of finishing

reads $\overline{(f_a, f_b)} = f_1 \ldots f_n$.

- For any two bounding constraints (f_a, f_b) and (g_a, g_b), the intersection $\overline{(f_a, f_b)} \cap \overline{(g_a, g_b)} = \emptyset$.

2.2.1 Assemble the reads

The algorithm uses the hierarchy of overlaps, unitigs, contigs, and scaffolds as used by Celera Assembler [110]. The WGS and finishing reads are processed to detect pair-wise overlaps. Reads and overlaps are compressed into unitigs, also called chunks [108]. The unitigs are low-risk assemblies consistent with nearly all of the detectable pair-wise read overlaps. Unitigs that are deemed repetitive, V, are not trusted. All the others, U, are trusted. The trusted unitigs are assembled into contigs and scaffolds using detected pair-wise overlaps and paired-end constraints. Untrusted unitigs are incorporated last. The contigs represent contiguous assembly. The scaffolds consist of contigs separated by gaps whose length is estimated from paired-end constraints.

2.2.2 Fill the gaps

The algorithm applies aggressive techniques to fill gaps in scaffolds. It starts by assigning left over unitigs (including individual reads) to specific gaps. It generates a list G of gaps in scaffolds, a list U of unique trusted unitigs not placed in scaffolds, and a list V of all untrusted repeat unitigs. For each unitig $u \in U \cup V$, and a gap $g \in G$, define a placement score $P(u, g) = M + B$ where M=(number of WGS reads

in u whose paired-end constraint would be satisfied by placement in gap $g \in G$) and B=(number of finishing reads in u whose bounding constraint would be satisfied by placement in gap $g \in G$). A paired-end constraint is satisfied when one read is within u while the second is not and the placement of the contig in the gap matches the expected orientation and distance of the paired-end library. A bounding constraint is satisfied when the bounding reads (f_a, f_b) are both placed in a pair of contigs and span only gap $g \in G$. For each unique contig $u \in U$, assign it one gap that maximizes $P(u, g) > 0$, if one exists. For each repeat contig $u \in V$, assign it all gaps for which $P(u, g) > Q$ for some threshold Q.

Once the unitigs are placed within gaps, we apply a miniature assembly process. Estimate the length of each gap from spanning paired-end constraints. For each gap, construct a graph whose nodes are unitigs assigned to the gap. Add a node for both of the contigs that bound the gap. Add an edge for every unitig pair, or unitig/contig pair, that shares paired-ends or has a detectable sequence overlap. Search for any path through the graph that satisfies the size estimate and visits each node at most once. If such a path is found, the gap is filled with unitigs from the path. If no path spans the gap, the algorithm will settle for less: adding a unitig to each contig end so as to reduce the size of the remaining gap. This gap-filling step runs twice: first with unassembled unitigs from U, then with all unassembled unitigs from U and V. For all unitigs in scaffolds, a multiple sequence alignment of reads is determined. Repeat unitigs, V, can have zero, one, or many placements in the assembly, but their reads can have at most one placement (Figure 2.1(a)). For each unitig $u \in V$ with at least one placement in a scaffold, every read $r \in u$ is assigned a

specific location, if possible. Read r is placed only if there is unambiguous support from the placement of u, and the mate of r, if any, or the bounding constraint of r, if any.

2.2.3 Implementation

In order to exploit the maturity of WGS assembly software, we implemented the algorithm inside an existing package, Celera Assembler. Originally designed for Sanger data [110], the software now incorporates alternate modules into a pipeline called CABOG [105] specifically for data from the 454 Life Sciences next-generation sequencing platform [97].

The bounding read behavior is offered as a run-time option in Celera Assembler. When turned on, it executes as part of the scaffold module, specifically during the gap filling stage called rocks and stones [110]. To search the set of unitigs that could fill a gap, Celera Assembler uses a breadth first search algorithm. As always, Celera Assembler calculates the scaffold consensus sequence given the hierarchical layout of contigs, unitigs, and reads. Where the new code needs to partition the unitigs into U and V, it re-uses Celera Assembler's partition, which is based on empirical observation of the distribution of unitig coverage levels and calculation of the A-stat log-odds ratio [110].

The implementation allows that some constraints are inherently unsatisfiable. Constraints where the bounds span multiple gaps or are not part of the assembly are considered unsatisfiable and the current implementation does not use them. The

Table 2.1: **Test datasets for the closure algorithm.** Sanger WGS Reads: number of paired or unpaired reads from WGS on a Sanger platform. 454 PE WGS Reads: number of reads from WGS pyrosequencing of a paired-end library, after processing to split linker-positive sequences into two reads. 454 UP WGS Reads: number of reads from WGS pyrosequencing of an unpaired library. Features (Features with bounds): regions targeted by finishing that present at least one finishing read bounded by two other reads. Bounded (Bounded finishing reads): number of finishing reads provided with a bounding constraint. These reads are the focus of the bounding read algorithm. For *E. coli* K12, a 454 library was used in combination with finishing reads and their bounds generated for the *E. coli* O157:H7 project. WGS: whole-genome shotgun. PE: paired end (counting two reads per pair). UP: unpaired read.

Species	Sanger WGS	454 PE WGS	454 UP WGS	Features	Bounded
E. coli O157:H7	59 749	0	0	516	1 041
E. coli K12	1 032	67 910	185 682	516	1 041
S. enterica	51 199	0	0	608	891
B. mallei	54 980	0	0	825	1 235
C. amycolatum	3 410	0	193 092	37	45
I. multifiliis	232 924	11 332	2 606 081	734	1 233

threshold Q, used to place unitigs from V in scaffold gaps, is calculated automatically at run time for each unitig. The threshold defaults to the number of unsatisfiable constraints. Intuitively, this value of Q requires more evidence for a unitig placement than against.

2.2.4 Test data

Six genomes were selected for testing (Table 2.1). The genomes include five bacteria and one protozoan. All six WGS data sets include Sanger sequence. Three are predominantly pyrosequencing data. All six finishing read sets include Sanger reads from selected WGS clones.

2.3 Results

2.3.1 Test of the algorithm

Contiguity statistics, such as N50, are course-grained and mask the improvements that a small numbers of finishing reads can provide. Therefore, we introduce a concept of candidate regions and we measure how many candidates are improved. The candidates are genome repeats for which we have finishing reads and bounding constraints. Potential improvements consisted of closing a gap (Fig 2.1(b)), adding read coverage across a repeat, and fixing the consensus sequence (Fig 2.1(a)). The candidates are exclusive of non-repeat regions (Fig 2.1(c)) for which the control algorithm is sufficient. Improvement was measured by comparing the bounded read assemblies against control assemblies. The controls used the same software and reads without the bounding constraints.

2.3.2 Comparison to alternate assemblers

Dupfinisher is a pipeline for assembly, repeat identification, finishing read generation, and re-assembly. It is integrated with phrap, BLAST, Consed, and Autofinish [52]. It was not feasible or meaningful to snap Dupfinisher into our assembly pipeline or to snap our assembler into its pipeline. Also, it was not possible to compare published experiments. Papers that describe Dupfinisher report combined gains of the software plus manual editing (*e.g.* [54]) or present results on projects [53] for which we could not obtain finishing reads.

The Newbler *de novo* assembly software [97] is designed specifically for py-

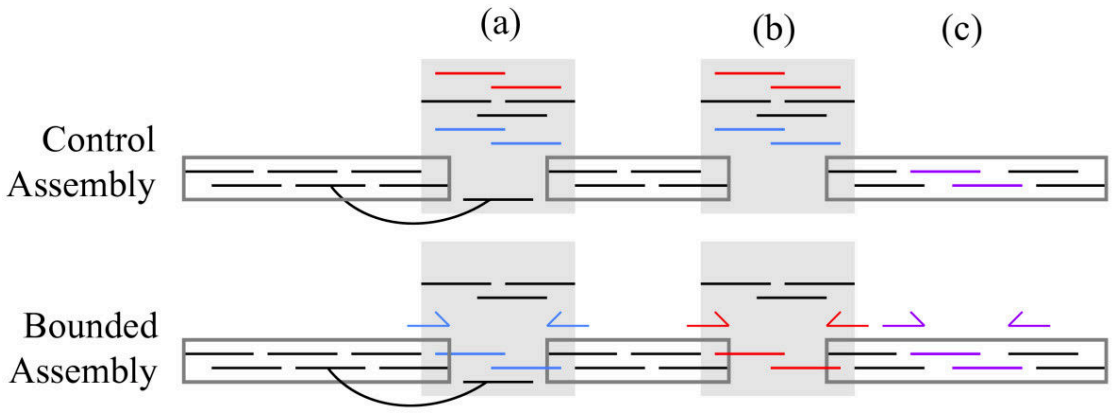


Figure 2.1: **Resolution of repeats using finishing reads.** Two algorithms for assembling shotgun reads and finishing reads. The control treats both read types equally. The bounded algorithm attempts to assemble finishing reads consistently with their bounding constraints. For each algorithm, the figure shows its construction of a scaffold from contigs (rectangles) with 2X in shotgun reads (black lines). Each finishing read (colored line) has a corresponding pair of PCR primer sites (arrows of same color). External to the scaffold is a unitig (grey area) deemed repetitive due to high coverage. (a) A mate pair constraint (curve) localizes one read and the unitig to this gap. Nevertheless, the control algorithm cannot tile this gap with reads. The bounded algorithm localizes two finishing reads by their primer sites. The bounded algorithm does tile the gap with reads, enabling a more accurate consensus sequence. (b) The control cannot localize the unitig or any reads to this gap. It does not close the gap. The bounded algorithm localizes the unitig by finishing reads and their primer sites. It tiles the gap with finishing reads from the unitig. (c) Both algorithms assemble finishing reads from a gap that is not a genomic repeat. In our data sets, most finishing reads fit gaps of this type.

rosequencing reads alone or in combination with Sanger data. It also supports incremental assembly, whereby additional reads are added to a previous assembly result. The incremental assembly feature can be used to add finishing reads to assemblies of WGS reads. Newbler was tested on three genomes for which we had pyrosequencing reads.

Table 2.2: **Comparison of three closure algorithms.** Control uses finishing reads like WGS reads. Bounded uses finishing reads with placement constraints. Alt (Alternate) uses finishing reads in a second round of assembly without constraints. Candidate gaps include both regions in the control assembly between finishing constraints with zero coverage and a consensus sequence derived from a repeat unitig or no consensus sequence in the control assembly. The parentheses indicate the number of gaps with no consensus sequence in the control assembly. The gap and spanning constraint are not necessarily 1-to-1. Bounds: The total number of bounding constraints that span the repeat gap or were not satisfied in both control and bounded assemblies. Finishing: The total number of finishing reads generated for the bounds in the table. Placed finishing reads: The total number of finishing reads placed in the assembly by each of the assembly algorithms. Gaps closed: The number of gaps closed by filling in missing consensus sequence or by tiling repeat instances with reads. By definition, the control assembly always closes 0 gaps. The bounded assembly joins were verified by alignment to finished reference, where available.

				# Placed Finishing Reads			Gaps closed		
Species	Candidates	# Bounds	# Finishing	Control	Bounded	Alt	Control	Bounded	Alt
E. coli O157:H7	14 (1)	56	128	26	92	-	0	11	-
S. enterica	2 (0)	18	33	14	23	-	0	1	-
B. mallei	9 (0)	23	40	4	27	-	0	4	-
I. multifiliis	11 (2)	14	21	3	21	17	0	6	4
E. coli K12	49 (2)	23	60	12	49	11	0	29	0
C. amycolatum	4 (0)	3	3	0	2	0	0	1	0
Total	89 (5)	137	285	59	214	28	0	52	4

2.3.3 Assembly results

The bounded algorithm closes 52 candidate regions, previously having either no sequence or no read coverage. The control, by definition, closes 0 and the alternate pipeline closes 4. This large gain can be attributed to incorporating more finishing reads. The algorithm incorporates 75.10% of finishing reads. That is 54% more than control and 42% more than the alternate. The assemblies show a gain of $0.09 \pm 0.09\%$ in percent of the genome with > 1X read coverage versus control. Full details of the per-genome assembly improvements are in Table 2.2. The bounded algorithm also fixes 63 consensus bases while introducing only 3 incorrect calls in comparison to the control versus reference. Details are available in Table 2.3.

Table 2.3: **Consensus quality of the bounded read placement algorithm.** Performance of the same three algorithms described in Table 2.2. Number of consensus differences: The total number of bases in consensus that are different between the bounded and control assemblies versus reference. True positive: Number of consensus base changes that are supported by the reference. False positive: The number of consensus base changes that differ from the reference. The finishing reads used for *E. coli* K12 did not come from the same strain as the reference. We cannot validate whether a consensus discrepancy between an assembly and the reference is due to assembly error or to strain-level differences. Consensus quality could not be measured on the two genomes that lack a reference.

Species	# Consensus differences	True positives	False negatives
E. coli O157:H7	14	14	0
S. enterica	5	5	0
B. mallei	47	44	3
Total	66	63	3

Table 2.4: **Contiguity metrics for the bounding read placement algorithm.** Performance of the same three algorithms described in Table 2.2. Contig count: number of contigs whose consensus is at least 2Kbp. Contig bases: sum of consensus lengths for contigs at least 2Kbp long.

	Contig count			Contig N50			Contig bases		
Species	Control	Bounded	Alt	Control	Bounded	Alt	Control	Bounded	Alt
E. coli O157:H7	6	5	-	2 315 032	4 484 293	-	5 656 811	5 661 119	-
S. enterica	6	6	-	3 620 140	3 620 144	-	4 813 438	4 813 442	-
B. mallei	19	19	-	424 003	424 003	-	5 835 215	5 834 616	-
I. multifiliis	4 273	4 273	5 765	12 070	12 070	11 444	37 616 884	37 616 884	47 976 992
E. coli K12	313	314	387	27 255	27 255	16 838	4 679 711	4 679 711	4 441 778
C. amycolatum	26	26	38	307 040	307 040	152 524	2 525 388	2 525 392	2 507 351

In three of the six genomes, the algorithm places additional copies of repeat unitigs, filling in missing consensus sequence. As a consequence, ten contigs are merged. Due to the coarseness of the contig metrics, the bounded assemblies show improvement (fewer contigs and higher N50) in one genome but no change on the rest. In two genomes, the merging involved one contig greater than 2Kbp and one smaller than 2Kbp and is therefore not reflected in Table 2.4.

The assemblies of *E. coli* O157:H7 were examined and compared to the avail-

able reference. The bounded assembly was confirmed, having eight alignments of 99% identity over 99% length of the assembly covering 99% of the reference. By the same measure, the control assembly had nine alignments. Inspection revealed two high-coverage unitigs each placed six times in the control and seven times in the bounded. Together, the two unitigs make up a seventh repeat instance that was missing from the control. The repeat, which we characterized by NCBI BLAST [3] as an rRNA operon, is known to occur seven times in the wild-type genome. [19]. Other joins in the bounded assembly were also verified by comparison to the reference, indicating no mis-assembly (Section A.2.1). The bounding read algorithm placed a majority of the finishing reads available for each genome. On no genome did the algorithm close all the gaps or tile all the candidate regions. The algorithm closed 52 out of 89 possible candidates, this may be due to limitations of the finishing read set rather than the algorithm as no assembly was able to close all candidates. In summary, the bounding read algorithm consistently augmented repeat resolution and gap closure by finishing read placement and improved the consensus.

2.4 Discussion

We implemented our algorithm within the Celera Assembler software for whole-genome shotgun (WGS) assembly. The implementation placed more finishing reads than two alternate methods: *de novo* assembly of WGS reads and finishing reads together (our control), or by adding finishing reads to the initial assembly (with Newbler). This result was not surprising since only our algorithm exploited the

finishing read placement constraint data associated with finishing reads.

All of our test data sets included some Sanger WGS reads. Future genome projects are unlikely to present Sanger WGS data due to the lower cost of high-throughput, next-generation sequencing (NGS). Such projects will require clone-free finishing reads generated from genomic template. In this case, each amplicon's end reads can serve as bounds for the other reads derived from that amplicon. Thus, our approach should apply to 100% NGS WGS data sets.

We have presented a novel algorithm for automated re-assembly to exploit finishing reads and placement constraints. Our implementation out-performed two other automated approaches on real data. An alternate approach to finishing, relying on NGS data to correct assembly errors, shows average gains of $0.16 \pm 0.15\%$ apart from a single outlier with 6.73% gain [115]. By comparison, our algorithm achieves a gain of $0.09 \pm 0.09\%$ through the careful use of existing finishing data, without relying on any additional sequencing. Both methods are valuable to correctly assemble the final pieces of a genome and demonstrate the difficulty involved.

2.5 Conclusions

The finishing process has rate-limiting manual components. Here we demonstrate automation of one finishing component, the careful placement of finishing reads whose position is known relative to other reads. We described the Bounding read algorithm that could be incorporated in a 4-part finishing pipeline: WGS reads are assembled with an assembler; the assembly is scanned for low-quality regions

and gaps; finishing reads are generated to target each region; the WGS and finishing reads are re-assembled with the bounding read assembly algorithm.

Earlier approaches to automated finishing use *a posteriori* methods that add finishing reads to assembled contigs. Dupfinisher was the first. Newblers iterative assembly method demonstrates another. Our approach incorporates finishing reads *a priori* in a *de novo* assembly with the WGS reads. The finishing reads are exploited throughout the assembly construction, possibly generating a different result than the WGS-only assembly. Additionally, our algorithm can identify new instances of recognized repeats and tile reads across them. The algorithm outperformed two alternate methods, filling more gaps, placing more reads, and improving consensus. Our algorithm is a valuable tool to assist the automation and improvement of genome finishing projects.

Chapter 3

Hybrid error correction and *de novo* assembly of single-molecule sequencing reads[†]

3.1 Overview

Second-generation sequencing technologies, starting with 454 pyrosequencing [97] in 2004, Illumina sequencing-by-synthesis [8] in 2007 and others, have revolutionized DNA sequencing by reducing cost and increasing throughput exponentially over first-generation Sanger [139] sequencing. Despite the great gains provided by second-generation instruments, they have several drawbacks. First, they require amplification of source DNA prior to sequencing, leading to amplification artifacts [113] and biased coverage of the genome related to the chemical-physical properties of the DNA [24]. Secondly, current second-generation technologies produce relatively short reads: typically 100 bp for Illumina (up to 150 bp) and $\approx$ 700 bp mode (up to 1 000 bp) for 454. Short-reads make assembly and related analyses difficult, with theoretical modeling suggesting that decreasing read lengths from 1 000 bp to 100 bp can lead to a six-fold or more decrease in contiguity [73]. Finally, the sizeable runtime of most second-generation instruments is prohibitive for applications

[†]The text of this chapter is based on the publication S. Koren, M. Schatz, B. Walenz, J. Martin, J. Howard, G. Ganapathy, Z. Wang, D. Rasko, W. McCombie, E. Jarvis, and A. Phillippy. Hybrid error correction and *de novo* assembly of single-molecule sequencing reads. *Nature Biotech*, In Review, 2012

requiring rapid analysis, with an Illumina HiSeq 2000 instrument requiring 11 days for the sequencing reaction alone (http://www.illumina.com/).

Pacific Biosciences recently released their first commercial "third-generation" sequencing instrument, the PacBio RS: a real-time, single-molecule sequence. It aims to address the problems outlined above by requiring no amplification and reducing compositional bias [140, 16], producing long sequences (*e.g.* median = 2 246, max = 23 000 bp using the latest PacBio chemistry which has not yet been publicly released) [130], and supporting a short turn-around time (24 hrs sample to sequence) [28, 16]. The long read lengths would be beneficial for *de novo* genome and transcriptome assembly as they have the potential to resolve complex repeats or span entire gene transcripts. However, the instrument generates reads that average only 82.1% [16]–84.6% [130] nucleotide accuracy, with uniformly distributed errors dominated by point insertions and deletions. The high error rate obscures the alignments between reads and complicates analysis since the pairwise differences between two reads will be approximately twice the individual error rate. This error is far beyond the 5%–10% error rate [105, 97, 137] that most genome assemblers can tolerate and simply increasing the alignment sensitivity of traditional assemblers is computationally infeasible (Section 3.3.4). Additionally, the PacBio technology utilizes hairpin adaptors for sequencing double stranded DNA, which can result in chimeric reads if the sequencing reaction processes both strands of the DNA (first in the forward and then reverse direction). While it is possible to generate accurate sequences on the PacBio RS by reading a circularized molecule multiple times (circular consensus or CCS), this approach reduces read length by a factor

equal to the number of times the molecule is traversed (resulting in sequence with an example median = 423 bp, max = 1 915 bp). Thus, there is a great potential advantage to the long, single-pass reads if the error rate can be algorithmically managed.

To overcome the limitations of single-molecule sequencing data and unlock its full potential for *de novo* assembly, we developed an approach that utilizes short, high-identity sequences to correct errors in long, single-molecule sequences (Fig 3.1). Our pipeline, PacBio corrected Reads (PBcR), implemented as part of the Celera Assembler [105], trims and corrects individual long-read sequences by first mapping short-read sequences to them and computing a highly-accurate hybrid consensus sequence: improving accuracy from as low as 80% to over 99.9%. The corrected "hybrid" PBcR reads may then be *de novo* assembled alone or in combination with other data, or exported as FastA sequences for other applications. As demonstrated below for several important genomes, including the previously un-sequenced ~ 1.2 Gbp genome of the parrot *Melopsittacus undulatus*, incorporation of PacBio data using this method leads to greatly improved assembly quality versus either first or second-generation sequencing, indicating the true dawn of a "third generation" of sequencing and assembly.

3.2 Methods

Our strategy consists of two phases: a long-read correction phase and an assembly phase. Both are implemented as part of the Celera Assembler [105], but

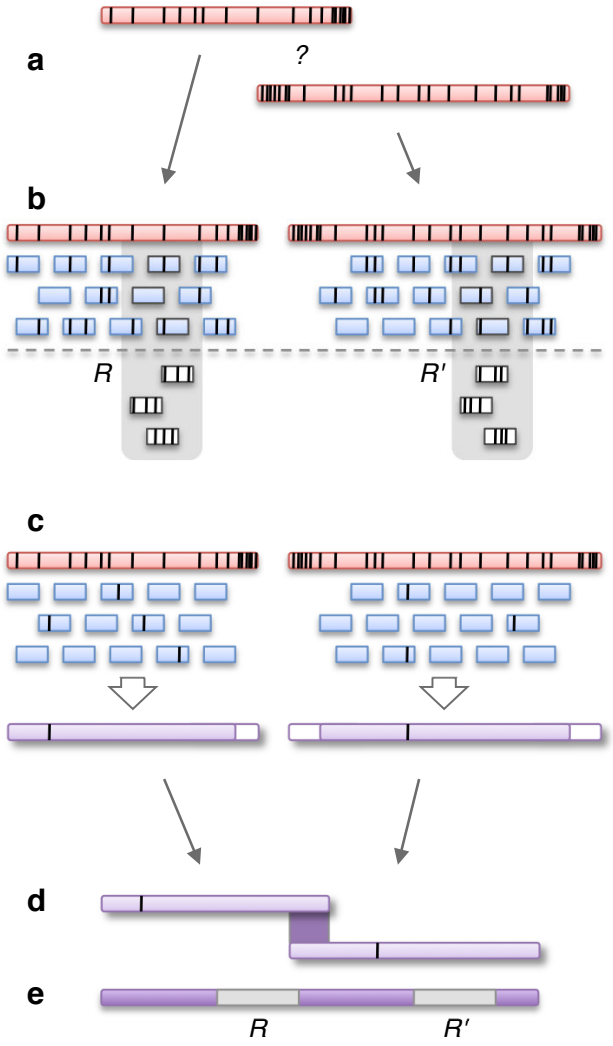


Figure 3.1: **The PBcR single-molecule read correction and assembly pipeline.** a) The high-error, indicated by black vertical bars, in single-pass PacBio RS sequences obscures overlaps. b) Given a high-accuracy sequence ($\approx 99\%$ identical to the truth), the error between it and a PacBio RS sequence is half the error between two PacBio RS sequences. Therefore, accurate alignments can be computed. In this example, black bars in the short-reads indicate "mapping errors" that are a combination of the sequencing error in both the long and short reads. In addition, a two-copy inexact repeat is present (outlined in gray) leading to "pile-ups" of reads at each copy. To avoid mapping reads to the wrong repeat copy, the pipeline selects a cutoff, C, and only the top C hits for each short read are used. The spurious mappings (in white) are discarded. c) The remaining alignments are used to generate a new consensus sequence, trimming and splitting long reads whenever there is a gap in the short-read tiling. Sequencing errors, indicated in black, may propagate to the PBcR read in rare cases where sequencing error co-occurs. d) After correction, overlaps between long PBcR sequences can be easily detected. e) The resulting assembly is able to span repeats that are unresolvable using only the short reads.

the output of the correction phase can be used as input to any other analysis or assembler capable of utilizing long FastA sequences. The outline of the correction algorithm is as follows: (1) high-identity short-read sequences are simultaneously mapped to all long-read sequences, (2) repeats are resolved by placing each short-read sequence in its highest identity repeat copy, (3) chimera and trimming problems are detected and corrected within the long-read sequences, and (4) a consensus sequence is computed for each long-read sequence based on a multiple alignment of the short-read sequences.

3.2.1 Simulated overlap error

We developed a simulated alignment program to calculate expected overlap error between PacBio sequences. The program assumed 83.7% accuracy, with a 11.5% insertion, 3.4% deletion, and 1.4% substitution rate. Reads were simulated from the same position of a reference *E. coli* K12 and randomly mutated. The resulting sequences were aligned and cumulative overlap error computed. Whenever two sequences had the same type of error in the same position, the error was ignored (that is if both reads had the same insertion at a single position). The simulation shows that the overlap error is approximately additive (1.87 times the single-sequence error) with the average error in a single sequence being 16.8% and the total error of 31.55% (versus 33.76% if the error were exactly additive) due to some pairs of sequences sharing an error at the same position. As most second-generation sequence overlaps are found below 3% error (Fig 3.2), we expect the average over-

lap between PacBio reads and high-identity short-read sequences should be at most 17.5% (16.5% + 1%). Simulating PacBio to Illumina overlaps, where Illumina has 99% accuracy (with all errors being substitutions), results in a total error of 17.45% (versus 17.83% if the error were exactly additive). Therefore, the expected overlap error between a high-accuracy technology (such as Illumina) and PacBio is approximately half (1.8 times) of one between two PacBio sequences. This observation is supported by real *E. coli* K12 data in Figures 3.2 and 3.3 below. Thus, our approach relies on the fact that overlaps between long-read sequences and high-identity sequences are detectable by current alignment methods.

3.2.2 Overlap detection

The algorithm begins by computing all-vs-all overlaps between the low-accuracy single-pass (PacBio) long-read sequences and high-identity short-read sequences (Illumina, 454, PacBio CCS). The overlaps are computed only between fragments that have shared seed sequences of a pre-defined length (14 bp by default), and only short-read sequences aligned across their entire length to a long-read sequence are considered; support for partial overlaps to the ends of long reads is left for future versions. For efficiency, overlaps between reads of the same technology (*e.g.* short to short) are not computed during this phase.

Next, overlaps are converted into a tiling of short-read sequences along each long-read sequence. Each short-read sequence is permitted to map to more than one long-read sequence, since the long-read sequences are expected to cover the

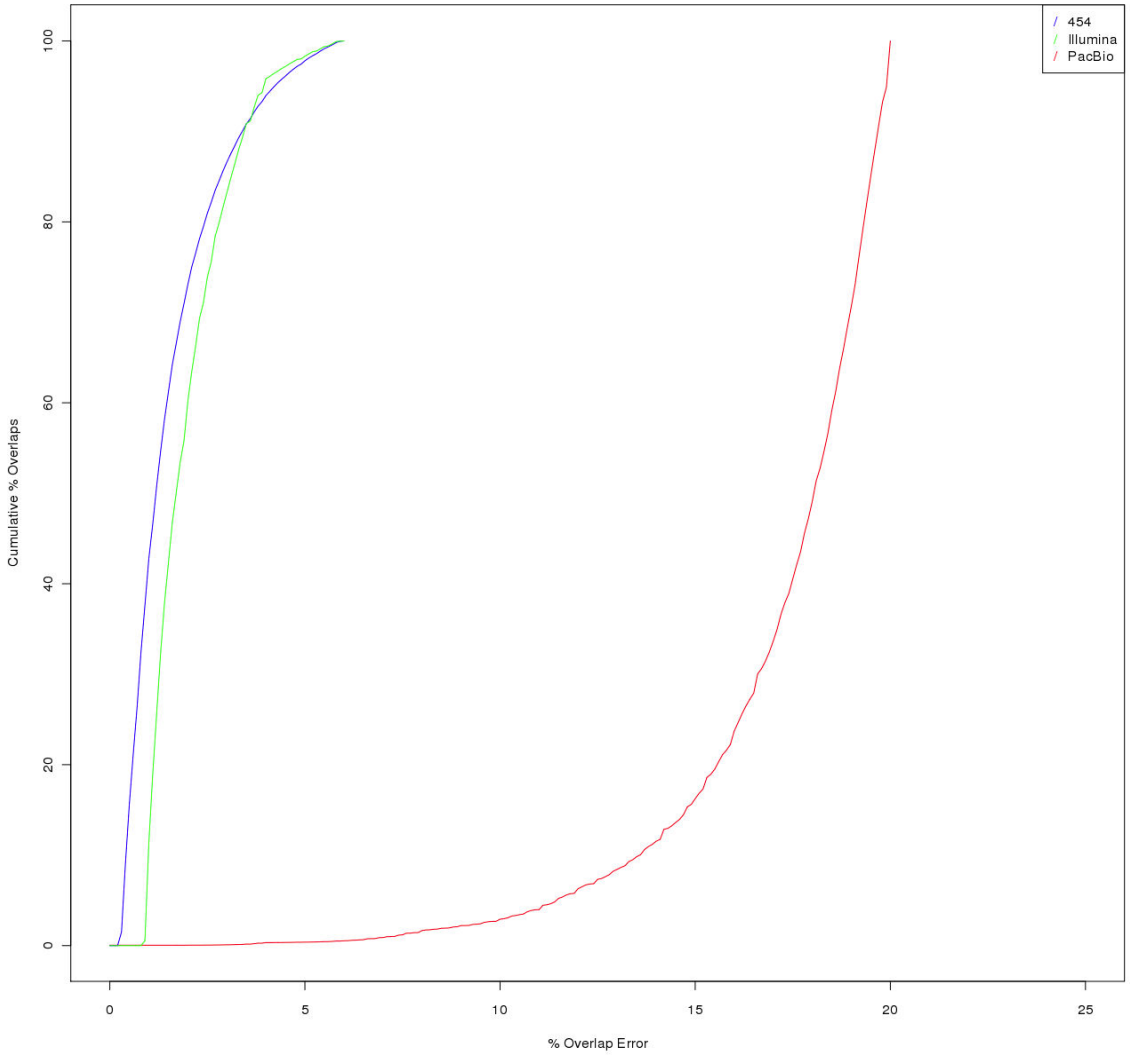


Figure 3.2: **PacBio RS sequences are too high error to accurately detect overlaps.**The cumulative percentage of overlaps detected below a given overlap error threshold is shown. The cumulative % of overlaps is calculated relative to the total number of overlaps detected below 25% error. As PacBio overlaps are expected to be 31.55% error (beyond the maximum limit of the overlapper), the curve above overestimates the percentage of true PacBio overlaps detected. For both 454 and Illumina, over 80% of the overlaps are detected by 3% error. By contrast, on PacBio, only 10% are detected at 15% error. Results shown using *E. coli* K12.

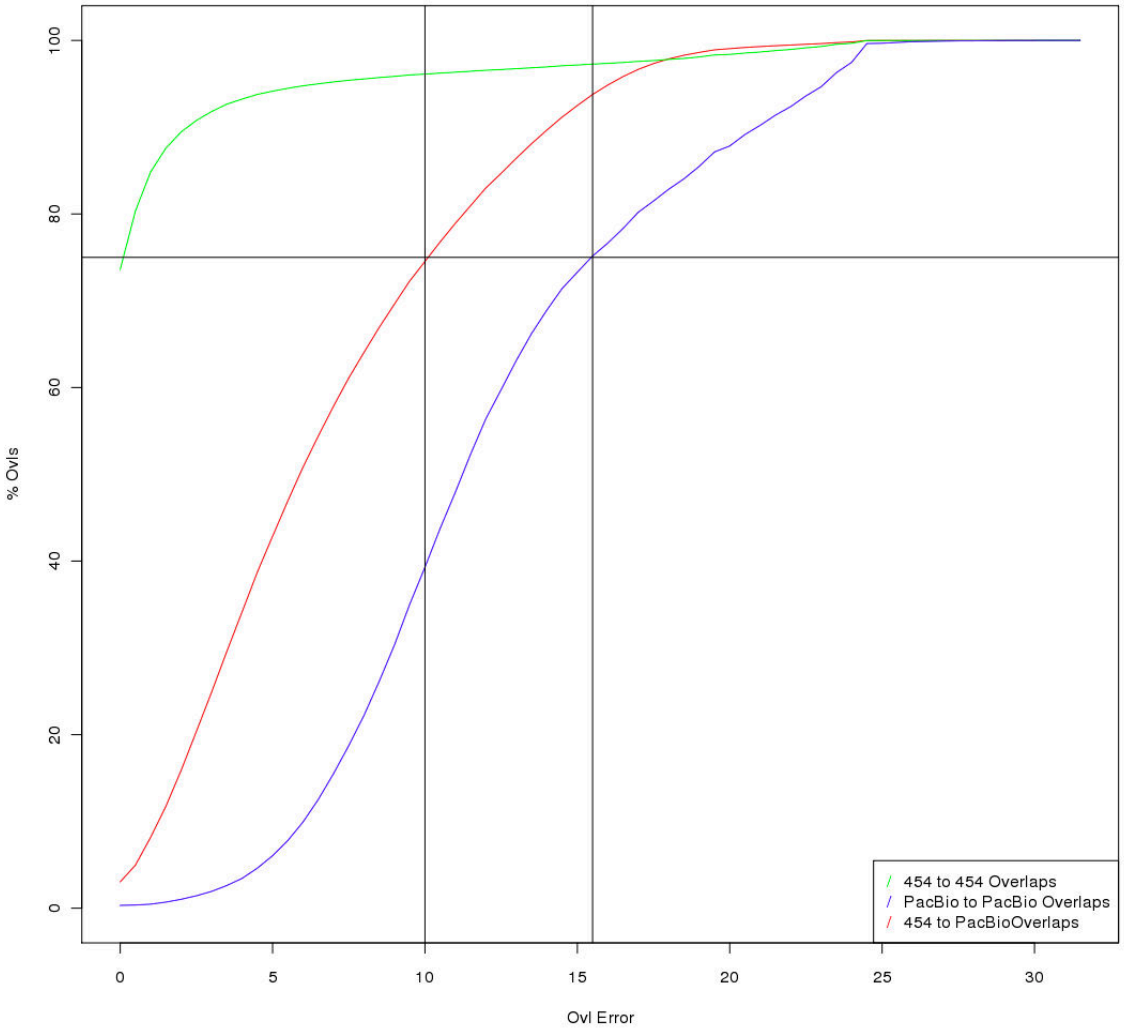


Figure 3.3: **Overlaps between PacBio RS and high-identity sequences are detectable at lower error than PacBio RS to PacBio RS overlaps.** The cumulative percentage of overlaps between 454 and PacBio is shown. As in Figure 3.2, the % of overlaps is computed out of the total overlaps detected below 25% error, overestimating the percentage of PacBio overlaps detected. As expected, the 454-PacBio overlaps are found much faster than PacBio-PacBio overlaps with approximately 75% detected by 10% error (versus less than 40% for PacBio-PacBio) and approximately 90% at 16% error. This corresponds well with our prediction that expected Illumina-PacBio overlaps error rate is 17.45%. Results shown using *E. coli* K12.

genome at more than $1X$ coverage. However, within a single long-read sequence, a short-read sequence is placed only in its highest identity location with ties broken randomly.

3.2.3 Repeat separation

In the case of repeats distributed across multiple long-read sequences, short-read sequences from all repeat copies will map to each copy of the repeat. To avoid tiling each repeat copy with the same set of reads, short-read sequences are separated into their appropriate copies by ranking their mappings by identity and permitting each short-read sequence to map only to its top C hits, where C is roughly defined to be the expected long-read sequencing depth. This effectively separates repeat copies when sequencing coverage and error is uniform. The value of C, a repeat threshold, is defined as follows:

Given a histogram $H = \sum_{n=1}^{n \leq max(n_i)}(n_i) \forall i$, and a threshold $0 \leq T \leq 1$

$$
\begin{aligned}
slope(K) =& \frac{H_K}{H_{K-1}} \forall K \geq 2 \\
total(K) =& \sum_{k=1}^{k \leq K} (\frac{H_K}{\sum_{k=1}^{k \leq max(n_i)} H_k}) \\
C =& min(K)\ s.t.\ total(K) \geq T\ and \\
& slope(K) > slope(K-1)\ and\ H_K < H_{K-1}
\end{aligned}
$$

Where n_i is the number of long-read sequences a short-read sequence i maps to $\forall i$. Theoretically, the histogram, H, has a peak equal to the long-read depth of coverage. It can be expected that a unique short-read sequence will map to, on average, this many long-read sequences. Thus, a short-read sequence from a two-copy repeat will map to roughly double this number. The chosen repeat threshold is the point in the curve past this peak that includes at least $T\%$ of the high-identity reads. Figure 3.4 shows an example histogram on *E. coli* K12. The histogram has a pronounced peak at 20X, corresponding to the PacBio coverage of this dataset. The vertical line shows the cutoff chosen by our algorithm. Figure 3.5 shows the coverage of the corrected PacBio RS sequence by Illumina data. The histogram has a peak at 50X, the Illumina coverage used for correction.

In this way, each repeat copy will only be tiled by its best representative reads for correction. This approach can sometimes place reads in the wrong repeat copy. For instance, in cases where the error rate of two PacBio RS sequences from two separate repeat instances is significantly different, such that one is higher, Illumina sequences may preferentially map to the lower-error PacBio read. This would increase the mapped coverage of the low-error read by including some reads from the alternate copy, while decreasing the coverage of the high-error read. However, this problem should be alleviated as overall PacBio coverage is increased, because the read accuracy distribution in the different repeat copies will converge after a few fold redundancy. As evidence, systematic misplacement of reads in repeats, leading to inaccurate correction, coverage fluctuations, or decreased throughput, has not been observed in any of our experiments (*e.g.* Table 3.3, Table 3.4, Fig 3.5).

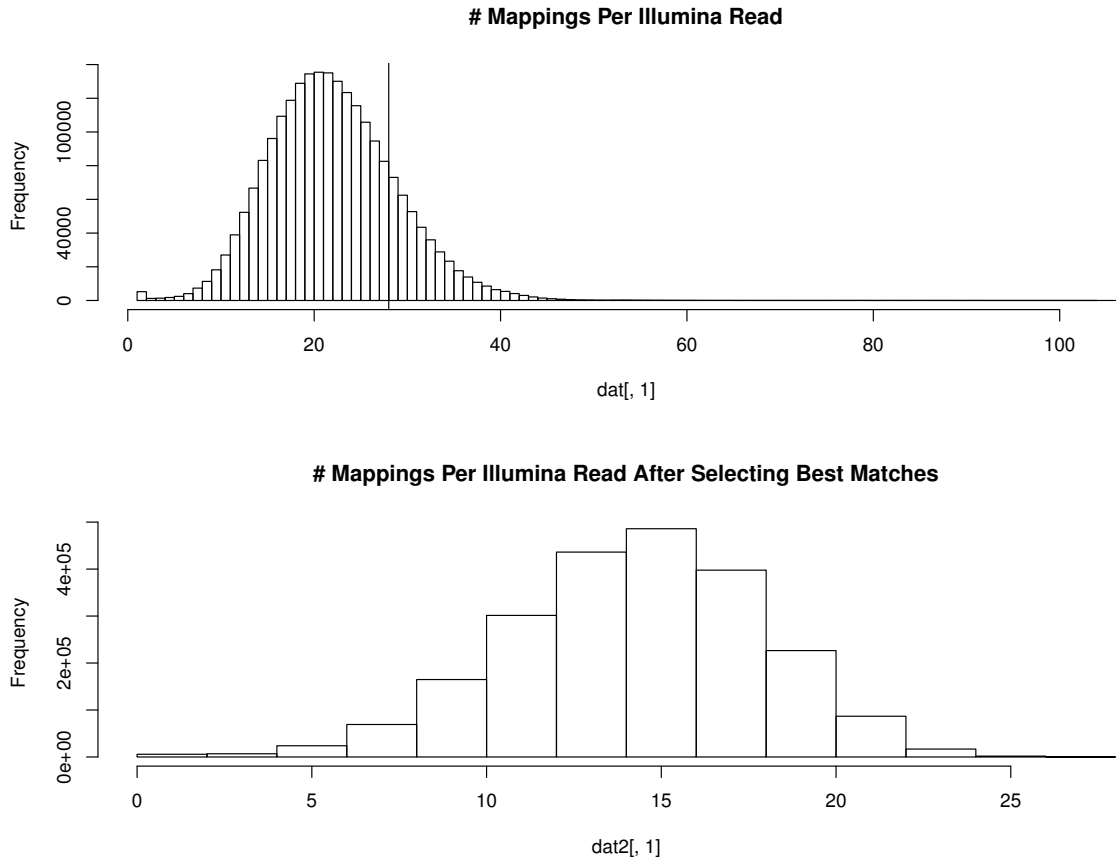


Figure 3.4: **Histogram for the number of PacBio long-read sequences each Illumina short-read sequence maps to for *E. coli* K12.** The peak is at 20, the coverage of the reference in long reads, and there is a long tail of reads with many matches, coming from repeat regions of the genome. The vertical line shows the repeat threshold identified by our algorithm, 28 in this case. Only the top 28 matches for each Illumina sequence will be used for correction. The remaining matches are assumed to be spurious due to a single Illumina sequence mapping to multiple instance of a genomic repeat. b) The histogram of Illumina read mappings after removing spurious repeat-induced mappings and short PacBio RS sequences.

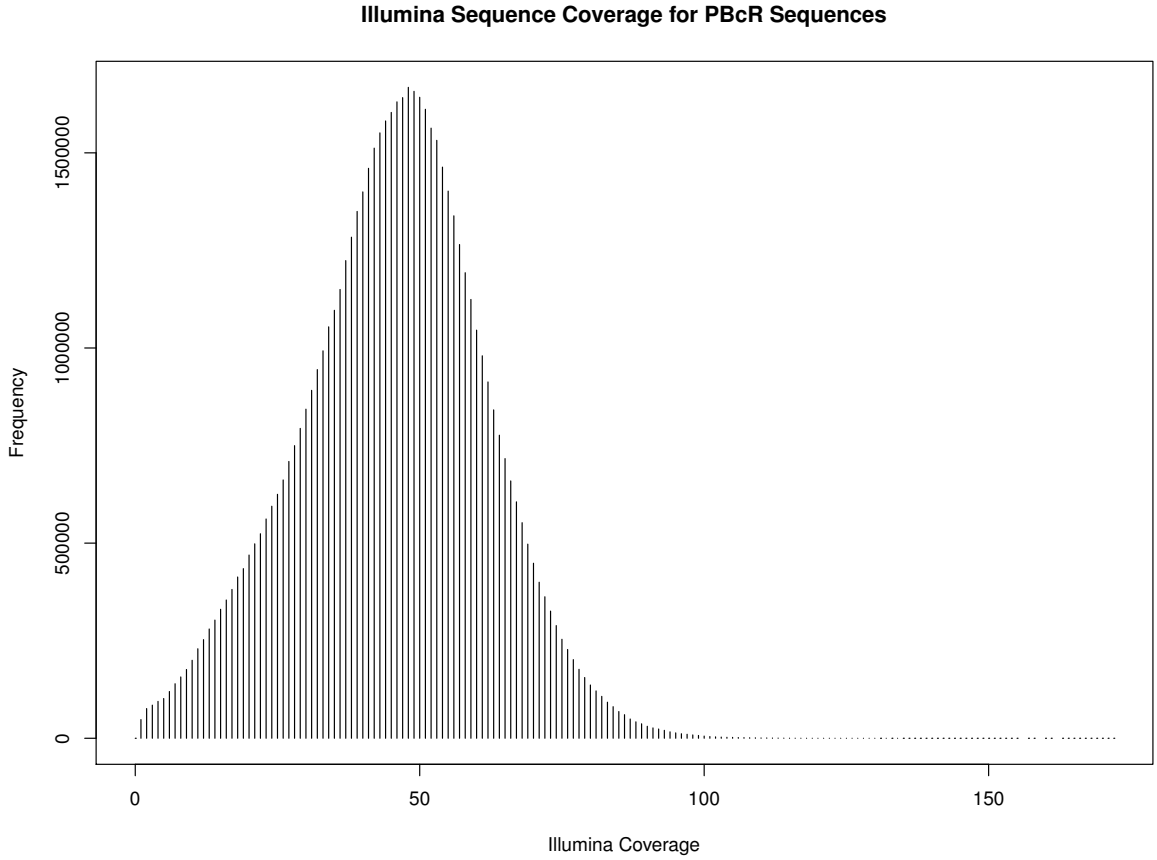


Figure 3.5: **Histogram of the Illumina short-read coverage for each corrected position of a PacBio sequence for *E. coli* K12.** The peak is at 50, the coverage of the Illumina short-reads used for correction. To correct for PBcR read ends, coverage is not computed for the first and last 100bp of each PBcR sequence (100 bp corresponds to the Illumina sequence length). The Y-axis shows the frequency while the X axis shows the coverage. The normal shape of the distribution shows the PacBio reads are uniformly covered by Illumina sequences at the expected depth, with very few regions of unusually high or low Illumina coverage.

3.2.4 Consensus generation

Finally, from the multiple-alignment of the tiled short-read sequences, the correction algorithm generates a new consensus sequence for each long-read sequence using the AMOS consensus module [127]. In the consensus, if there is a gap in the layout between adjacent overlapping short reads, this is considered a irreconcilable discrepancy between the short and long-read sequences, especially since the reads are generated from the same biological sample, and it is assumed there is sufficient coverage in the short sequences to tile each long-read sequence. Therefore, any gap in coverage is indicative of either improper trimming of the long-read sequence or chimera formation, and the long-read sequence is broken at this point. If instead there is merely insufficient coverage leading to a true sequencing gap for the short-read sequences, this will result in an unnecessary split. However, the correction algorithm errs on the side of caution. Future work remains to resolve any unnecessary gaps caused by the conservative trimming, such as by recognizing and filling these gaps during scaffolding.

The corrected, now high-identity, long-read sequences can be assembled alone or co-assembled with other read types using standard OLC assembly techniques. To support *de novo* assembly using Celera Assembler we have increased the input size limitation, applying it successfully to sequences up to 30 000 bp in length.

3.2.5 Implementation

The correction algorithm implementation is designed to be easily parallelizable, both using shared memory (via POSIX threads) and distributed architectures (using SGE). There are two important parameters to specify: 1) the number of parallel consensus jobs N. 2) the number of threads t to use for correction. A set of recommended parameters for SGE and shared-memory systems is provided in the code distribution. The correction step splits the long-read sequences into the user specified number of partitions N. The correction is parallelized in two blocks. The first streams through the overlaps computed for each long-read sequence and generates N intermediate files specifying the layout of the short-read sequences. The repeat threshold C is then computed as above. The overlaps are examined again (this time serially) for each short-read sequence and at most C best hits are stored for each. The best hits are recorded into N files, sorted by long-read sequence. Thus, the final parallel block uses a pool of t worker threads to operate on N partitions, selecting the next partition, $1 \leq n \leq N$ from a queue. As the intermediate results have already been sorted by long-read sequence and only matching high-identity short-read sequences remain, each thread can generate the output for a partition independently of the other threads. Finally, the consensus is computed in parallel on each of the N partitions.

Table 3.1: **Sequence data used to test PBcR correction/assembly pipeline** The eight datasets used for testing our assembly and correction pipeline. The PacBio RS lengths are reported before correction. The simulated data was generated by wgsim from the SAMTools package (version 0.1.16) [87]. Simulated sequences as well as the *Lambda* reference genome can be downloaded from http://www.cbcb.umd.edu/∼sergek/PacBio/index.html. The *Zea mays* project is hosted at http://www.maizesequence.org.

Genome	Institute	Tech	# Seqs	Mated	Med (bp)	Max (bp)
Lambda NEB3011	PacBio	PacBio RS	7 550	-	548	3 440
	simulated	Illumina	25 000	200bp	100	100
Escherichia coli K12	PacBio	PacBio RS	251 762	-	540	3 787
	Illumina UK	Illumina	22 720 100	500bp	100	100
Escherichia coli C227-11	PacBio	PacBio RS	258 301	-	2 098	22 841
	PacBio	PacBio CCS	617 561	-	423	1 915
	wgsim	Illumina	4 125 500	500bp	100	100
	wgsim	Illumina	2,749,218	3Kbp	100	100
	wgsim	Illumina	2 749 218	6Kbp	100	100
Escherichia coli 17-2	PacBio	PacBio RS	212 399	-	2 188	17,696
	IGS	Illumina	30 282 936	300bp	100	100
Escherichia coli JM221	PacBio	PacBio RS	211 366	-	2 553	18 564
	IGS	454 FLX Titanium	1 174 121	-	470	612
Saccharomyces cerevisiae S228c	CSHL	PacBio RS	969 445	-	588	8 495
	CHSL	Illumina	57 886 340	300bp	76	76
Melopsittacus undulatus	PacBio	PacBio RS	4 176 242	-	1,308	16,947
	Duke University	Illumina	660 997 244	400bp	101	101
	Roche/ Duke University	454 FLX Titanium(+)	48 337 115	3,8,20Kbp	385	2 038
	BGI	Illumina	2 031 639 664	0.22,0.5,0.8, 2,5,10Kbp	90	150
Zea mays B73	DOE JGI	PacBio RS	131 257	-	1 027	5 613
	DOE JGI	Illumina	230 000 000	-	250	250

3.3 Results

3.3.1 Test data

We used eight datasets to test and validate our correction and assembly pipeline, shown in Table 3.1

3.3.2 Analysis of PacBio RS sequences

The error distribution of PacBio RS sequences was evaluated using the *S. cerevisiae* S228c genome. The error at each base position was tabulated for all

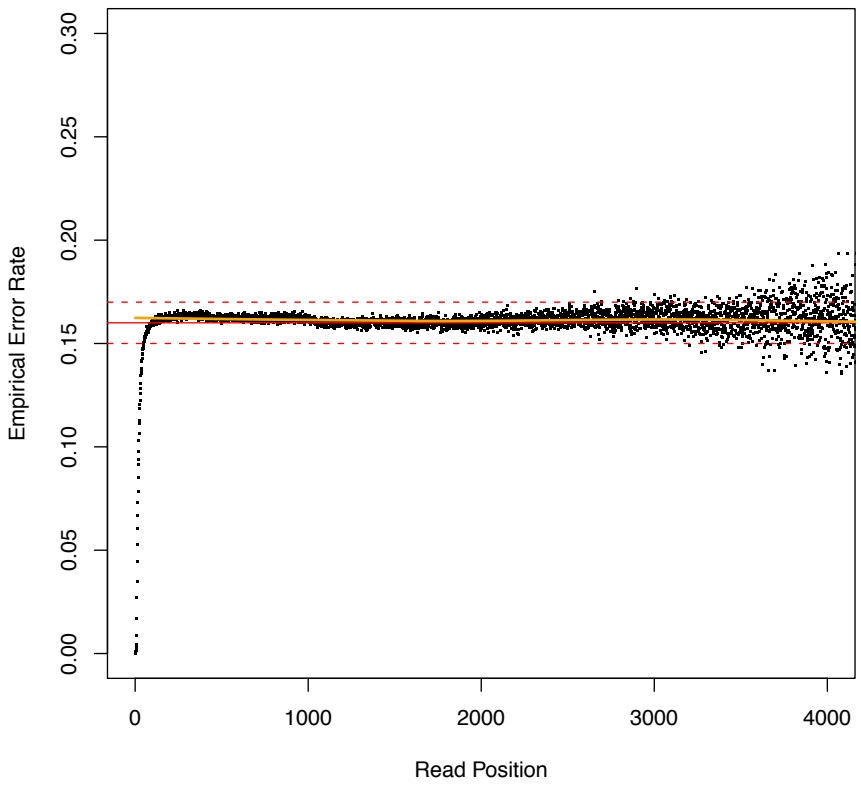


Figure 3.6: **The PacBio RS sequence error profile.** The positional error profile in PacBio RS sequences. The read error rate is tightly distributed around 16% (as expected) with very little deviation until 2.5Kbp. This pre-release sequencing data does not have many sequences over 2.5Kbp in length, limiting the sample size for the error rate calculation. However, the divergence does not show a drop in accuracy, instead there is an equally likely probability of higher or lower accuracy, reflecting the sampling effect.

sequences. Figure 3.6 shows the resulting distribution. Unlike all other sequencing technologies currently available, the PacBio RS shows a normal error distribution with no positional bias. The only deviation from the expected 16% error rate is visible after 2.5Kbp when a low sample size (due to the exponential read length distribution) causes the per-base positional accuracy to diverge from the mean. Note that this divergence does not show decreasing accuracy but is instead equally distributed both above and below the mean as would be expected with a sampling artifact.

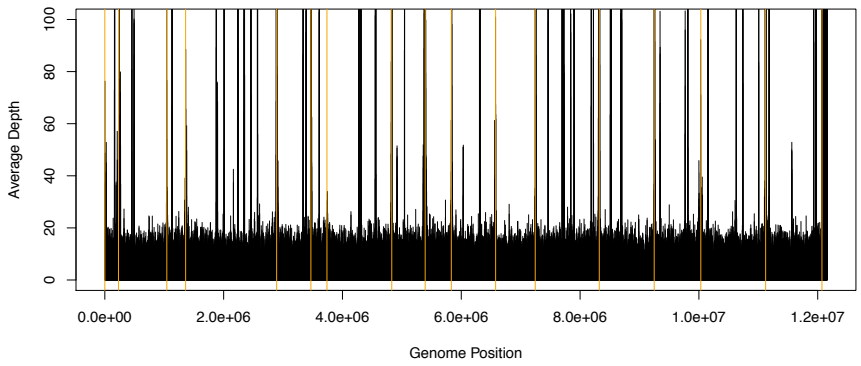


Figure 3.7: **PacBio RS sequencing depth by genome position.**The gold vertical lines separate the chromosome (chrI-chrXVI, then the mitochondrion genome). The plot shows that most of the genome is evenly covered with minimal bias. The mitochondrial genome shows higher depth because it is present in the cell at a greater copy number than the chromosomes. Coverage spikes in the chromosomes are mapping artifacts caused by repeats, because we report the ten best hits for each read.

We examined the induced coverage along the reference *Saccharomyces cerevisiae* S228c genome considering the top ten best alignments for each read and only considering alignments $\geq 1\,000$ bp. Figure 3.7 shows the average coverage of 1 000 bp bins along the genome, with gold vertical lines separating the chromosome (chrI-chrXVI, then the mitochondrion genome). The even coverage is consistent with Pacific Biosciences claim that their technology removes amplification bias and greatly reduces GC bias, leading to more uniform coverage of the genome than other technologies [16].

The coverage distribution is also displayed in Figure 3.8, which shows the number of bases of the genome at each coverage level. The distribution closely matches the expected Poisson distribution with a lambda of ~ 12.5 (shown in red) although the variance is slightly higher than predicted by a Poisson process. In particular, from the Poisson distribution 1.48% of the genome is expected to be at

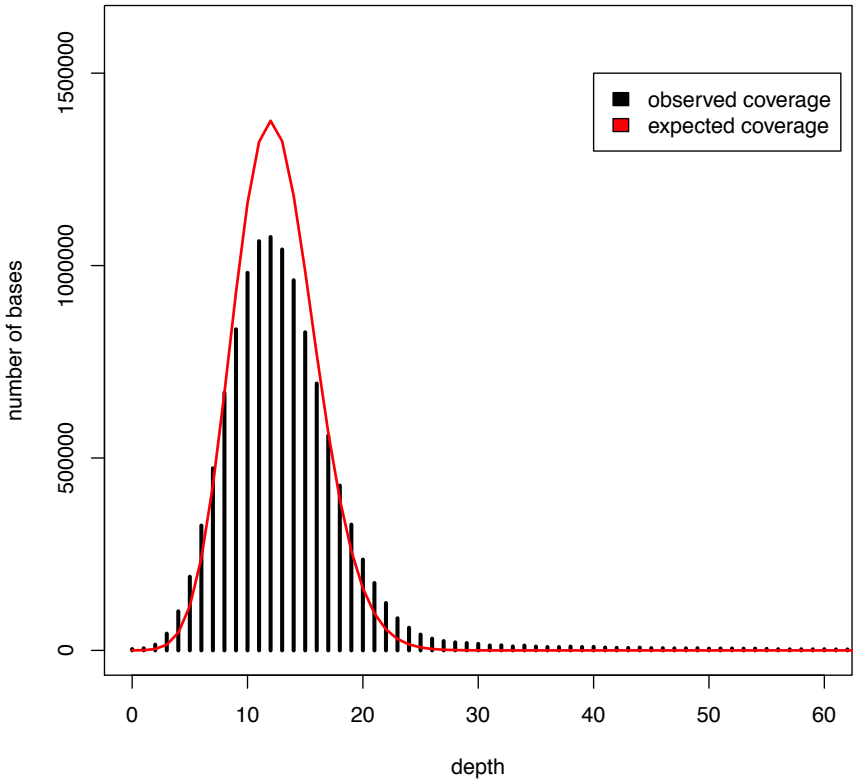


Figure 3.8: **PacBio RS coverage matches Poisson expectation.** The distribution closely matches the expected Poisson distribution with a lambda of ~ 12.5 (shown in red) although the variance is slightly higher than predicted by a Poisson process.

5-fold or lower coverage, but instead 2.97% has 5-fold of lower coverage coverage. Furthermore, only .0000187% of the genome is expected to have 30 fold or greater coverage, but instead 5.97% has high coverage with a max coverage of 2 024X.

We examined the regions with zero coverage and found they fell into 6 contiguous segments, although 3 were 13bp or less, concordant with the Poisson expectation. The remaining 3 consist of a 2 861 bp region of chr III (148 616–151 476), 326 bp of chr IX (119 987–120 312), and the last 158 bp of chr VI (270 002-270 160). The longest region contains a cluster of 3 LTRs and several tRNAs. The next largest segment consists of an exon of the STH1 gene, and the last segment contains part of

the telomeric arm of the chromosome. These results suggest the sequencing process may have small biases against certain repetitive sequences, although given the very small numbers of events and non-zero coverage across the other > 100 annotated teleomeric repeats, any biases must have marginal effects.

The highest coverage regions ($\geq$ 1 000 fold) consisted of a single 16 895 bp segment (451 625–468 519) on chr XII, which contains many genes from the 35S and 18S ribosomal RNA transcripts. The coverage spikes are likely a mapping artifact resulting from reporting the top ten matches for each sequence.

3.3.3 *De novo* assembly of long reads

Genome assembly is the computational problem of reconstructing a genome from sequencing reads [125, 104]. It and the closely related problem of *de novo* transcriptome assembly are critical tools of genomics required to make order from a myriad of short fragments. The assembly problem is frequently formulated as the problem of finding a traversal of an appropriately defined graph derived from the sequencing reads. Two commonly used formulations are: the Overlap-Layout-Consensus (OLC or string graph) paradigm [108, 67, 109, 105] where the graph is constructed from overlapping shared sequences (edges) between sequence reads (nodes), and the Eulerian/de Bruijn graph formulation [62, 122, 13, 15, 183] where the graph is constructed from substrings of a given length k, called k-mers, derived from the set of reads.

The optimal value of k for a de Bruijn assembler is dependent of the length of

the read, the genome coverage, and the error rate. In particular, the value k must be small enough so that reads with a true overlap share many error-free matching sequences of at least k bases. In contrast, OLC assemblers form the assembly graph using overlaps, where the graph nodes represent sequencing reads and an edge indicates overlap between two reads. A string-graph formulation can be used to simplify the graph by removing all transitive edges. After transitive reduction, the remaining branching nodes indicate read disagreement, where a sequence a overlaps both sequences b and c, but b and c do not overlap each other. The majority of assemblers developed for second-generation sequencing rely on the de Bruijn graph formulation because it is computationally simpler to identify length-k exact matches between reads, making it better suited for high-coverage, short-read sequencing. In contrast, OLC assemblers start by finding overlaps using pair-wise sequence alignment.

Assembly graph complexity is determined by both sequencing error and repeats, but repeats are the single biggest impediment to all assembly algorithms and sequencing technologies [123]. Under a de Bruijn graph formulation, repeats longer than k form branching nodes that must be resolved by "threading" reads through the graph or by applying other constraints, such as mate-pair relationships [101]. In contrast, within OLC assemblers, only repeats longer than $\bar{r} = r_{max} - 2 \times o$ cause unresolved branches in the graph, where r_{max} is the read length and o is the minimum acceptable overlap length. For short-read sequences, k and $\bar{r}$ are very similar, so the corresponding graphs are nearly equivalent. However, for long reads, $\bar{r}$ may be substantially longer than feasible values of k due to the limiting factors of sequencing error. Therefore, the availability of long sequences has a greater po-

tential to simplify the OLC assembly problem. In the extreme case, if all repeats in a genome were spanned by reads of greater length, OLC assembly of the genome into its constituent chromosomes and/or plasmids would be trivial. In practice, longer reads increase the probability of spanning repeats, and thus produce better assemblies at lower sequencing coverage than short reads.

3.3.4 Assembly of uncorrected PacBio data

For these reasons, OLC would seem to be superior for assembling long reads. Figure 3.9 reports the estimated k-mer size required to detect overlaps at various error rates. For PacBio RS uncorrected sequences, a k-mer size of 10 or lower is required to detect overlaps. We hypothesize that PacBio RS sequence overlap detection is computationally expensive and below the sensitivity of most assembles.

To test our hypothesis, we measured the relative performance of *de novo* assembly of PacBio RS reads from *Lambda* phage using OLC and de Bruijn methods. As expected, the assemblers are unable to deal with the high-error present in the (uncorrected) PacBio RS data, producing "shattered" assemblies (Table 3.2).

3.3.5 Comparison of assemblies using high-identity long-read data

Previous analysis confirmed the difficulty of finding both matching k-mers and overlaps at high-error rates (Fig 3.9, Fig 3.2). We next evaluated the performance of assembly after error correcting the *Lambda* phage PacBio RS sequences with high-accuracy short-read sequencing (Table 3.2). All assemblies were more contigu-

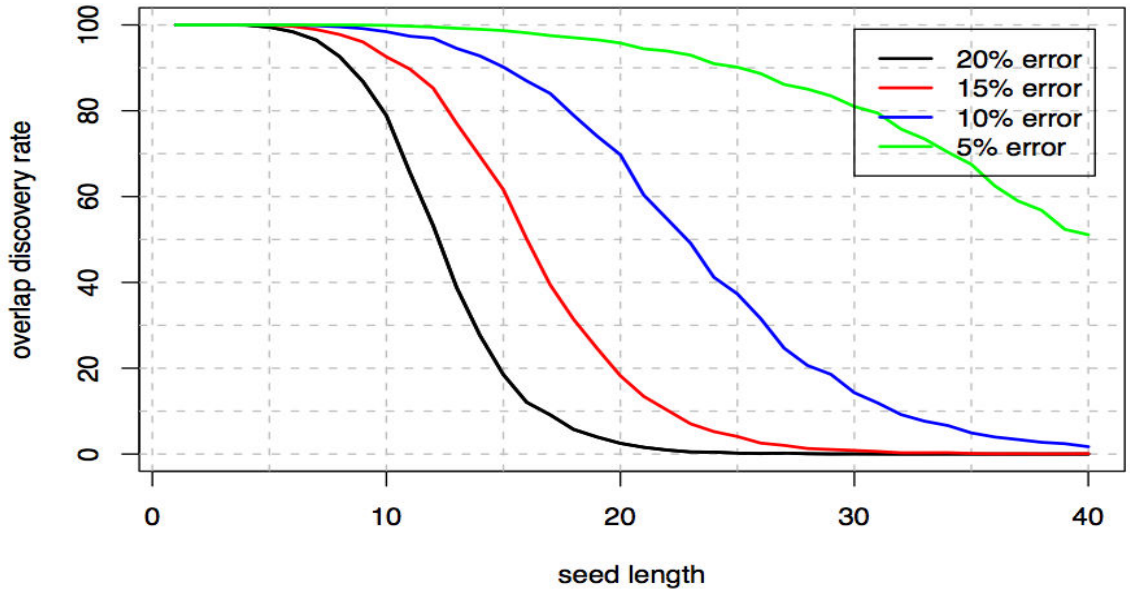


Figure 3.9: **Random errors obscure overlap seeds.** 20X coverage of 1 000bp reads was simulated for *E. coli* K12 at four error rates and the fraction of known overlapping reads sharing an exact match of at least seed-length bases was measured. The current PacBio error rate falls between the black bar (20% error) and the red bar (15% error). At this rate, a seed length of approximately 10 is required for good overlap discovery. Shorter seed sizes complicate the assembly graph (since any repeat longer than seed size is must be resolved via read threading or paired-ends).

Table 3.2: **Assemblers cannot utilize PacBio RS high-error sequences directly.** The most popular OLC and de Bruijn graph assemblers were compared on uncorrected PacBio data. % Cvg is the percentage of the reference covered by the assembly and % Idy is the average identity of the uniquely-mapping regions of the assembly to the reference. For CA, the k-mer size specified is the minimum length used to seed an overlap. Not surprisingly, neither the OLC nor the de Bruijn graph assemblers are able to deal with the high rate of sequencing error present in PacBio RS data, even on this simple phage genome. All assemblies cover only a fraction of the genome at low identity while making many errors. After correction, the assemblers are able to reconstruct the genome accurately, however, only the OLC assembler is able to reconstruct the entire genome in a single contig (versus 5 for SOAPdenovo and 23 for Velvet).

Assembler	k-mer	N50	Max	Corrected N50	Total BP	% Cvg	% Idy
CA	14	0	1 095	0	4 734	6.33%	98.92%
SOAPdenovo	39	2 239	2 682	133	4 526 193	60.01%	95.79%
Velvet	37	1 544	2 640	1 544	2 285 250	75.47%	96.20%
Corrected sequences							
CA	14	48 452	48 452	48 452	48 452	99.90%	99.93%
SOAPdenovo	87	26 424	26 424	26 424	48 850	99.90%	99.93%
Velvet	87	26 430	26 430	26 430	52 886	99.90%	99.93%

ous but only the OLC assembler reconstructed the phage in a single contig. To test the benefits of increasing read lengths, we simulated error-free data of varying length from the *Saccharomyces cerevisiae* S228c genome and compared the resulting assemblies (Fig 3.10(a)). OLC assembly becomes progressively more powerful for longer reads, displaying a nearly linear increase in contig size as read lengths grow. In contrast, the de Bruijn assemblies plateau and cannot effectively utilize the long reads without increasing k beyond practical values due to the inherent limitations of the graph construction and the complexity of the read-threading problem [122, 111]. Therefore, we developed a pipeline to correct and assemble PacBio RS sequences using an OLC approach.

3.3.6 Correction accuracy and performance

We evaluated the PBcR correction and assembly algorithm on multiple short and long read datasets generated by Illumina, 454, and PacBio sequencing instruments, including three data sets with available reference sequences: *Lambda* NEB3011, *Escherichia coli* K12, and *Saccharomyces cerevisiae* S228c (Table 3.1). Using $50X$ of Illumina data to correct PacBio reads for each reference organism, the characteristics of the corrected data is examined by comparison to a reference genome (Table 3.3). Figure 3.11(a) shows the raw sequences produced by the PacBio instrument. The accuracy of reads has a peak at 89.01% (median = 89.13%), as expected. A significant fraction (50%) of the PacBio sequence cannot be accurately mapped to the reference. Figure 3.11(b) shows that the accuracy of the corrected

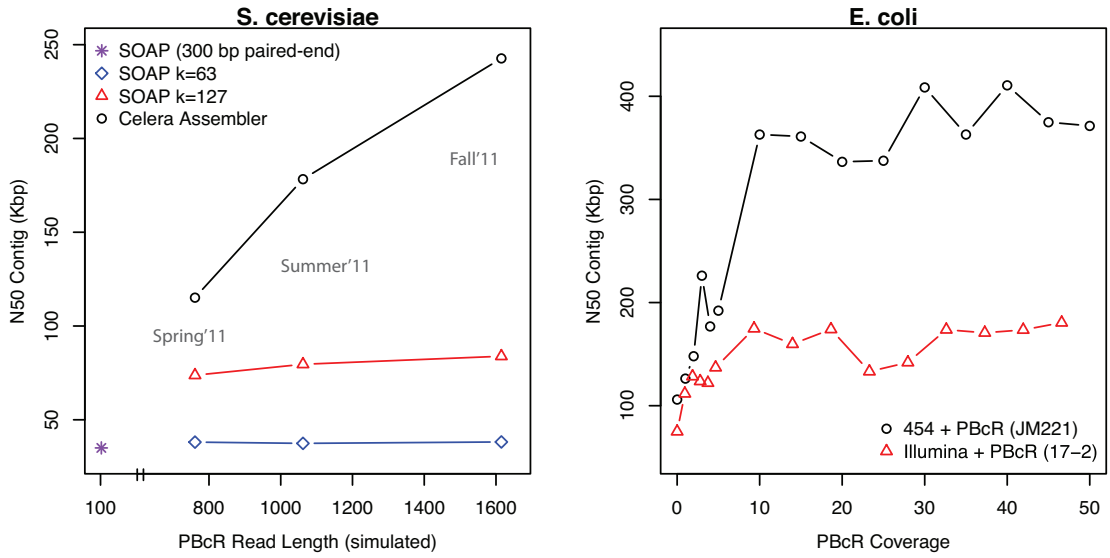


Figure 3.10: **Long-reads yield assembly improvements, even at low coverage.** a) Effect of PacBio corrected read length (PBcR) on contig size is measured for the OLC assembler Celera Assembler [105] and the de Brujin assembler SOAPdenovo [89]. Contig size, after breaking contigs at mis-joins, is measured using the standard N50 metric (N such that 50% of the genome is contained in contigs $\geq N$). The baseline SOAPdenovo assembly (purple star) represents an assembly of $50X$ of real 76 bp Illumina paired-end (300 bp) reads from *S. cerevisiae* S228c. Increasing PBcR read length was simulated and assembled using $10X$ of error-free reads generated from the reference at three additional length distributions: the pre-release PacBio instrument (Spring '11), the first publicly available instrument (Summer '11), and the "C2" chemistry upgrade (Fall '11). b). Effect of PBcR coverage is measured for *Escherichia coli*, sequenced with a combination of PacBio and second-generation sequencing. The benefit of the PBcR sequences is visible even below $5X$, which leads to a 50%–100% increase in N50. Maximum contig N50 is reached by $\approx 10X$, where adding $10X$ of PBcR increases the N50 by up as much as 3.5-fold (250%). The larger gain versus the 454-only assembly is due the longer PBcR sequences available for *E. coli* JM221. The variation in N50 is due to random subsampling of sequencing data.

reads with respect to the reference is 99.99% (median = 100%). The length of the sequences is shorter since chimeric sequences have been split during correction, but median length is not drastically affected (median = 848 pre-correction vs median = 767 post-correction). The corrected reads are also 99.96% (median = 100%) covered by a single match to the reference (Figure 3.11(b)).

The correction has a low rate of chimeric and improperly trimmed reads, measuring $< 2.5\%$ and $< 1\%$, respectively (Table 3.3). The concurrence of the corrected reads with their references is testament to the automated trimming process, which is necessary for the removal of adapter sequences that can be otherwise difficult to identify. During correction, reads may be discarded due to unusually low quality or short length, and the percentage of reads that are successfully corrected and output by the pipeline is termed *throughput*. The observed throughput is generally around 60%, but varies significantly depending on the quality of the individual runs. For example, throughput for the *S. cerevisiae* S228c reads appears unusually low, and is likely because much of this sequencing was performed using a pre-release PacBio RS instrument during testing at Cold Spring Harbor Laboratory. Nevertheless, in all cases the pipeline successfully identifies the usable data and outputs highly accurate long reads.

To further evaluate correctness accuracy, we selected regions of the genome that appear repetitive and compared the correction error rates within repeat regions to error rates in the full genome. PBcR quality was evaluated for the full PBcR set, and the repeat-only PBcR set for both *E. coli* K12 as well as *S. cerevisiae* S228c. To control for differences between the sequenced genome and the reference, the original

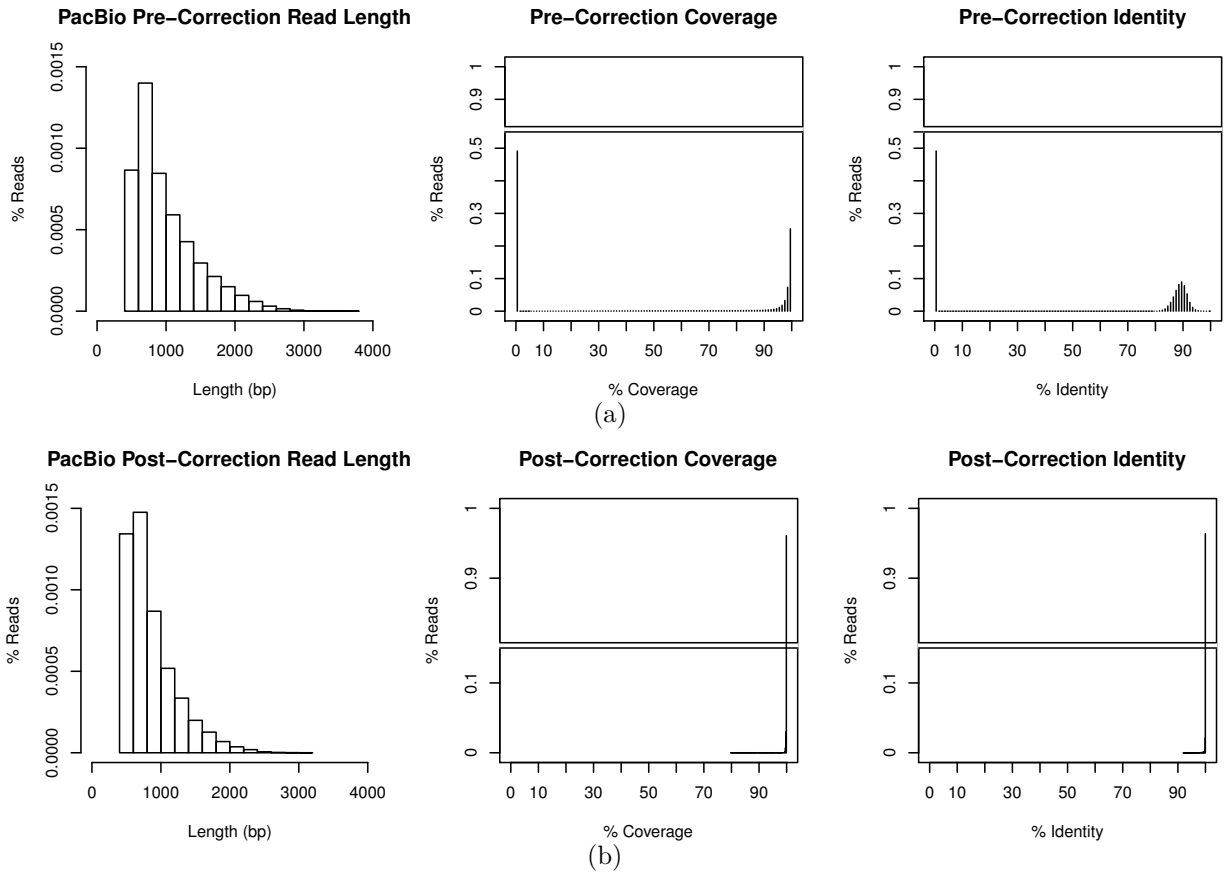


Figure 3.11: **A comparison of PacBio length, coverage, and identity versus a reference before (a) and after (b) correction.** Here, *E. coli* K12 is shown. Alignment was performed using MUMmer 3.23. Matched were filtered using `delta-filter q` to retain the best matches for each position of a sequence. a) the raw PacBio reads after quality filtering generated by the instrument. The % coverage is calculated as the total fraction of the fragment that could be mapped (in any number of matches) to the reference. The % identity is calculated as the average (weighted by match length) of all matches for a sequence. A significant fraction of sequences could not be aligned to the sequence and are reported as having 0% for coverage and identity. b) the same sequences after correcting using 50X of Illumina sequencing data. The resulting sequences are shorter (having a maximum of 3Kbp versus 4Kbp) due to breaking at positions with no short-read coverage. However, all corrected reads can be mapped to the reference, with the vast majority (over 95%) mapping at 100% identity over 100% of their length.

Table 3.3: **PacBio correction accuracy** Corrected (PBcR) read accuracy as compared to reference sequence. Reads were mapped using NUCmer 3.23 [82]. For all statistics, only reads $> 500bp$ were included. % TP (Throughput): the percentage of raw uncorrected bases that are in non-chimeric, correctly-trimmed sequences after correction. % Idy (R): average identity of good corrected reads to the reference. % Idy (A): average identity of assembled corrected reads to the reference. % Cov (Coverage): average coverage of good corrected reads by a single match to the reference. % Chimer: the percentage of corrected bases within reads with a split mapping to the reference. % Trim: the percentage of corrected bases within reads with a single match to the reference over less than 99.5% of their length. The time is reported in seconds and memory in gigabytes. The corrected sequences remain above 99% identity and 99% trim within the repetitive regions of the genome (Table 3.4).

Organism	% TP	% Idy	% Idy (A)	% Cov	% Chimer	% Trim	Time	Mem
Lambda NEB3011	74.03%	99.90%	100.00%	100.00%	1.82%	0.10%	121	0.12
E. coli K12	57.46%	99.99%	99.99 %	99.92%	2.02%	0.34%	1580	2.10
S. cerevisiae S228c	21.86%	99.90%	99.97%	99.93%	1.46%	0.33%	4357	5.90

uncorrected PacBio RS sequences were also evaluated. Only PacBio RS sequences with a mapping were used to tabulate statistics. The results are presented in Table 3.4. In all cases, the PBcR pipeline trims bad sequences while retaining over 99% identity and trim.

Sufficient coverage of short-read data was the primary factor for determining the effectiveness of the correction pipeline. As short-read coverage is increased, more long-read sequences can be accurately corrected, but the error correction runtime also increases linearly with the number of short-read sequences (Fig 3.12). To determine an appropriate tradeoff, *E. coli* K12 Illumina sequences were subsampled from $5X$ to $200X$ and used for correction (Fig 3.13, Table 3.5).

The long read accuracy greatly improves as Illumina coverage increases from 5 to 50X but improvements continue with diminishing returns at higher coverage. Furthermore, the correction pipeline required 135.46 CPU hours (2.5hrs wall-clock

Table 3.4: **PBcR correction is accurate within genomic repeats.** The repeat regions within genomes were selected by mapping. Both original PacBio RS sequences as well as the PBcR sequences intersecting those regions were selected and their quality evaluated as in Table 3.3. It is expected that selecting for repeat regions biases the selection towards naturally variable regions of the genome. Therefore, to identify correction errors versus true variation in the reads, the error rates were compared to the original PacBio RS reads. As expected, selecting repeat regions selects for variable regions within genomes, leading to higher rates of chimera and trim errors in the original PacBio RS data. Columns are defined as in Table 3.3: % Good Bases: the percentage of total sequence in non-chimera and non-trim sequences. % Idy (Identity): average identity of good corrected reads to the reference. % Cov (Coverage): average coverage of good corrected reads by a single match to the reference. % Chimer: the percentage of corrected bases within reads with a split mapping to the reference. % Trim: the percentage of corrected bases within reads with a single match to the reference over less than 99.5% of their length.

		All sequences					Repeat region sequences				
Genome	Seq	% Good	% Idy	% Cov	% Chim	% Trim	% Good	% Idy	% Cov	% Chim	% Trim
E. coli K12	Orig	30.46%	89.18%	99.77%	2.02%	60.96%	44.06%	89.01%	99.78%	4.12%	71.64%
	PBcR	97.61%	99.99%	99.92%	2.02%	0.33%	96.04%	99.94%	99.80%	3.37%	0.57%
S. cerevisiae S228c	Orig	13.78%	88.10%	99.63%	1.23%	22.81%	38.59%	88.23%	99.58%	4.56%	40.91%
	PBcR	98.27%	99.90%	99.93%	1.46%	0.33%	94.52%	99.51%	99.24%	3.15%	2.85%

time) and 7.5GB of peak memory for the 200X correction for an effective parallelism of 56 cores, and only 13.64 CPU hrs (0.5hrs wall-clock time) and 2.1GB of peak memory for the 50X case (for an effective parallelism rate of 32 cores). As the correction accuracy and assembly contiguity show diminishing returns after 50X, this coverage is recommended as a compromise between performance and accuracy.

3.3.7 Hybrid *de novo* assembly

We evaluated the impact of PBcR reads on whole-genome assembly, either alone or in combination with the complementary reads. Two other assemblers are also reported to support PacBio reads: ALLPATHS-LG [45] and ALLORA [130]. However, neither program performs correction or *de novo* assembly from uncorrected

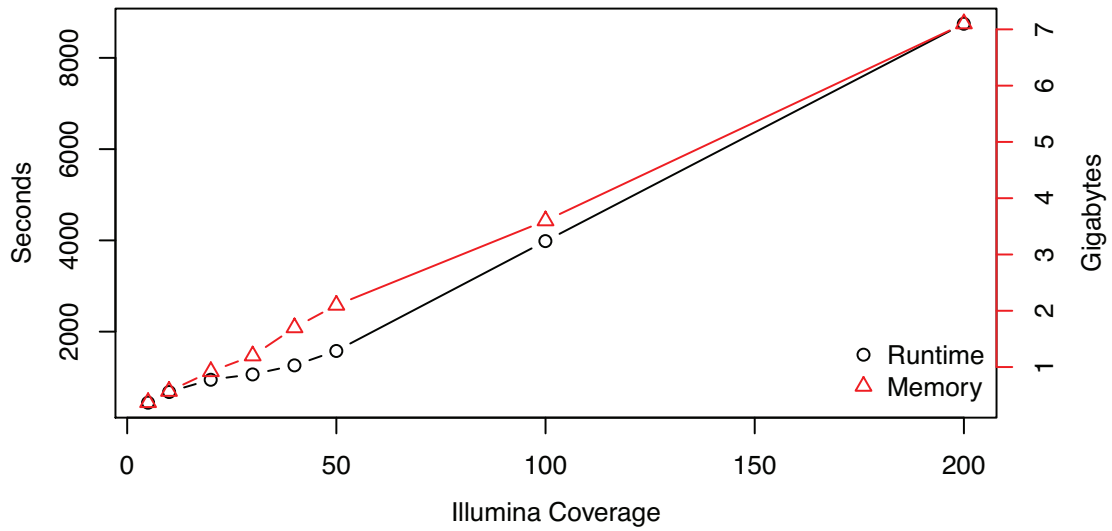


Figure 3.12: **Performance of the correction algorithm scales linearly with increasing coverage.** Performance of the correction pipeline as Illumina coverage is varied from 5X to 200X. The left vertical axis shows the time (in seconds) for the pipeline to complete. The right vertical axis shows (in gigabytes) the peak memory used by the pipeline. The peak memory is the maximum memory in use on a single machine by the pipeline. An average of 41.5 overlap jobs (min = 10, max = 97) were created and submitted to an SGE grid. For the correction step, we used 16 threads and 200 parallel consensus jobs to generate the corrected sequence.

Table 3.5: **PBcR correction is accuracy is independent of coverage.** The coverage of high-identity sequences does not have a significant impact on correction accuracy. While the throughput is lower (as shown in Fig 3.13), the identity and coverage remains above 99.9%.

Genome	Coverage	% Idy (Reads)	% Cov	% Chimera	% Trim
E. coli K12	5X	99.95%	99.91%	0.74%	0.12%
	10X	99.98%	99.95%	1.17%	0.13%
	20X	99.99%	99.96%	1.24%	0.14%
	30X	99.99%	99.93%	1.35%	0.47%
	40X	99.99%	99.93%	1.62%	0.50%
	50X	99.99%	99.92%	1.72%	0.49%
	60X	99.98%	99.91%	1.94%	0.52%
	70X	99.98%	99.92%	1.96%	0.57%
	80X	99.98%	99.93%	2.03%	0.59%
	90X	99.98%	99.91%	1.83%	0.75%
	100X	99.98%	99.92%	1.91%	0.62%
	200X	99.96%	99.91%	3.21%	0.67%

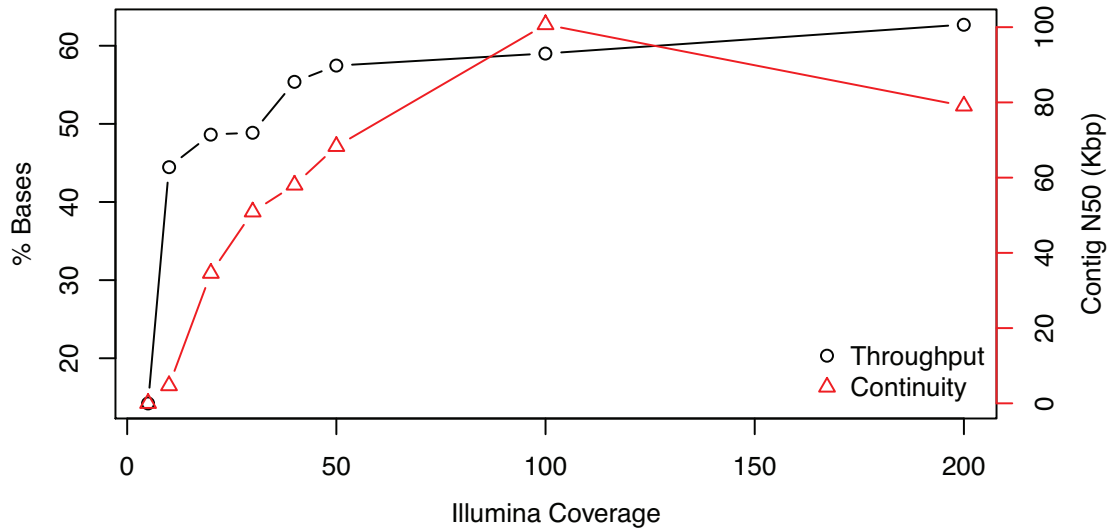


Figure 3.13: **Increased coverage with Illumina sequences allows increased error correction.** The percentage of original PacBio reads remaining after correction as Illumina sequence coverage is increased. Results are presented for E. coli K12. For assembly contiguity, the contig N50 (assembly only the PBcR sequences, after breaking at mis-joins) is reported. As the figure shows, there is a large gain as coverage increases from 5X to 30X, after which the return from additional sequencing begins to diminish, leveling off at 50X. The lower assembly contiguity at 200X represents a minor 4.86% percent drop in uncorrected N50. This lower contiguity is due to a 1.3% increase in chimera (to 3.61%) at 200X Illumina coverage. At this extreme depth, erroneous Illumina sequences begin to confirm native PacBio chimeras by random chance and these chimeras negatively affect the OLC assembly. More aggressive trimming of Illumina sequences before correction (for example by Quake [69]) reduces the chimera rate to 1.89% and eliminates the N50 drop. However, correction at this level of coverage is unnecessary and not recommended.

reads. Instead, ALLPATHS-LG uses the raw reads to assist in scaffolding and gap closure of short-read de Bruijn assemblies. The downside of this approach is that errors introduced in the short-read contigs will go uncorrected, and this function is currently available only for genomes < 10 Mbp with an Illumina paired-end library < 200 bp as well as a long-range Illumina jump library. Only the parrot genome presented here includes this required combination of Illumina and PacBio reads, but is larger than the stated size limit and could not be evaluated. ALLORA, a long-read assembler based on AMOS [127, 154, 142], is computationally limited to small genomes and requires high-accuracy PacBio sequences, such as CCS, to operate. Inspired by our initial results, low-accuracy PacBio sequences from the 2011 German *E. coli* outbreak were manually corrected using our consensus module and iteratively assembled with ALLORA [130]. We have now evaluated our automated correction and assembly pipeline on the same *E. coli* C227-11 genome, and have found it outperforms the previously published assembly (Table 3.8).

Being the only OLC assembler currently capable of assembling giga-base genomes from long reads (*e.g.* $> 2\,000$ bp), the Celera Assembler was used for all experiments unless otherwise noted. Current algorithms are capable of assembling large and accurate scaffolds using solely Illumina short and long-rage pairs, but leave many repeats unresolved, resulting in short contigs [26, 137]. Thus, we focus results on the area of the assembly most likely to gain from long reads: contig size.

In all cases, from bacterial to eukaryotic, the incorporation of PBcR data produces substantially better assemblies than any other sequencing strategy tested—in the best cases, more than tripling the N50 contig size for equivalent depths of cover-

Table 3.6: **PBcR sequences increase assembly accuracy and contiguity.** Organism: The genome being assembled. The median and max lengths of corrected PacBio sequences (PBcR) are given in parenthesis. The corrected length is shorter than original PacBio RS sequences due to trimming and splitting chimeric sequences. Table 3.1 reports the original PacBio RS sequence lengths before correction. The three reference data sets (*Lambda* NEB3011, *E .coli* K12, and *S. cerevisiae* S228c) were generated using the pre-release PacBio RS. Technology: the read data used for assembly. Pair separation (if applicable) is listed immediately after the coverage. Reference bp: the assumed genome size used for the N50 calculation. Assembly bp: the total number of base pairs in all contigs (only contigs $\geq 10\,000$ bp are included in all results). # Contigs: The number of contigs comprising the assembly. Max/CMax: The maximum contig length, before and after breaking at assembly errors following the methodology in [137]. N50/CN50: N such that 50% of the genome is contained in contigs of length $\geq N$, again before and after correction. Assemblies for next-generation (Illumina/454) data were generated by Celera Assembler [105] and SOAPdenovo [89]. Only the best assembly (based on contiguity) in each case was reported. The ratio is measured between corrected and original N50. A higher ratio indicates a more correct assembly. Full assembly quality statistics are listed in Table 3.7, following the GAGE assembly evaluation methodology [137].

Organism	Technology	Ref bp	Asm bp	Contigs Count	Contigs Max	Contigs CMax	N50	CN50	Ratio
Lambda NEB3011 med:727 max:3 280	Ill 100X 200bp	48 502	48 492	1	48 492	48 492	48 492	48 492	100%
	PBcR 25X		48 440	1	48 444	48 444	48 444	48 440	100%
E .coli K12 med:747 max:3 068	Ill 100X 500bp	4 639 675	4 462 836	61	221 615	221 553	100 338	83 037	82.76%
	PBcR 18X		4 465 533	77	239 058	238 224	71 479	68 309	95.57%
	18X PBcR + Ill 50X 500bp		4 576 046	65	238 272	238 224	93 048	89 431	96.11%
S. cerevisiae S228c med:674 max:5 994	Ill 100X 300bp	12 157 105	11 034 156	192	266 528	227 714	73 871	49 254	66.68%
	PBcR 13X		11 110 420	224	224 478	217 704	62 898	54 633	86.86%
	PBcR 13X + Ill 50X 300bp		11 286 932	177	262 846	260 794	82 543	59 792	72.44%

Table 3.7: **PBcR sequences generate accurate assemblies.** N50: Contig N50 size. CN50 (Corrected N50): corrected N50 length computed as in GAGE [137]. Ratio: The fraction of the corrected N50, relative to the originala larger percentage indicates a more correct assembly. Idy (Identity): The average identity of the assembly to the reference. Inv (Inversions): an inverted assembly with respect to the reference. Reloc (Relocation): chimeric assembly region corresponding to a large jump in the reference sequence. Trans (Translocation): A combination of sequences from two different chromosomes into a single assembled sequence. While the absolute number of errors is sometimes higher in the PBcR hybrid assemblies, the ratio of corrected N50 to original is always higher. This means the errors are in short contigs (chaff) that is any contig over 200bp. Since every PBcR read is over 200bp, chaff will include any originally chimeric untrimmed PacBio RS sequence. Future work remains to identify and remove this chaff from the assembly output.

Genome	Assembly	N50	CN50	Ratio	% Idy	Inv	Reloc	Trans	Total
Lambda NEB3011	Illumina	48 492	48 492	100.00%	99.92%	0	0	0	0
	PBcR	48 444	48 444	100.00%	99.93%	0	0	0	0
E. coli K12	Illumina	100 338	83 037	82.76%	99.99%	0	7	0	7
	PBcR	71 479	68 309	95.57%	99.99%	1	2	0	3
	Illumina + PBcR	93 048	89 431	96.11%	99.99%	7	3	0	10
S. cerevisiae S228c	Illumina	73 871	49 254	66.68%	99.99%	0	5	8	13
	PBcR	62 898	54 633	86.86%	99.97%	2	7	14	23
	Illumina + PBcR	82 543	59 792	72.44%	99.97%	2	5	25	32

age (Table 3.8, Fig 3.14). These improvements also come without the introduction of additional assembly error, as measured against the three available reference genomes (Table 3.6).

Figure 3.14 summarizes the N50 results for various technologies and coverages for the *E. coli* genome. The three "short-read" alternatives of $50X$ 454, $50X$ PacBio CCS, and $100X$ Illumina paired-ends all produce similar assemblies. However, substituting half of the 454 coverage with corrected PacBio reads increases the N50 contig by 3 fold (*e.g.* $25X$ 454 + $25X$ PBcR); matching $50X$ short-read CCS coverage with $50X$ of PBcR reads results in a 5 fold increase. Because PacBio sequencing can be completed in just hours, this example provides a promising method for rapid genotyping and sequencing in time-critical situations, such as for an emerging disease outbreak.

Table 3.8: **PBcR assembly is more contiguous than second-generation sequencing.** Organism: The genome being assembled. The median and max lengths of corrected PacBio sequences (PBcR) are given in parenthesis. The corrected length is shorter than original PacBio RS sequences due to trimming and splitting chimeric sequences. Table 3.1 reports the original PacBio RS sequence lengths before correction. Technology: the read data used for assembly. Pair separation (if applicable) is listed immediately after the coverage. Reference bp: the assumed genome size used for the N50 calculation. Assembly bp: the total number of base pairs in all contigs (only contigs $\geq 10\,000$ bp are included in all results). # Contigs: The number of contigs comprising the assembly. Max Contig Length: The maximum contig length. N50: N such that 50% of the genome is contained in contigs of length $\geq N$. Assemblies for next-generation (Illumina/454) data were generated by Celera Assembler [105], SOAPdenovo [89], and ALLPATHS-LG [45] (where possible). Only the best assembly (based on contiguity) in each case was reported.

E. coli C227-11 (med: 1 217 max: 14 901)	PacBio CCS 50X	5 504 407	4 917 717	76	249 515	100 322
	PacBio 25X PBcR (corrected by 25X CCS)		5 207 946	80	357 234	98 774
	PacBio PBcR 25X + CCS 25X		5 269 158	39	647 362	227 302
	PacBio PBcR 50X (corrected by 50X CCS)		5 445 466	35	1 076 027	376 443
	PacBio PBcR 50X + CCS 25X		5 453 458	33	1 167 060	527 198
	Manually Corrected ALLORA Assembly [130]		5 452 251	23	653 382	402 041
E. coli 17-2 (med: 886 max: 10 069)	Illumina 100X 300bp	5 000 000	4 975 331	62	226 141	74 940
	PacBio PBcR 50X		4 981 368	58	318 969	143 307
	PacBio PBcR 50X + Illumina 50X 300bp		5 022 503	55	367 911	180 932
E. coli JM221 (med: 1 216 max: 12 552)	454 50X	5 000 000	4 714 344	66	308 063	106 034
	PacBio PBcR 25X		5 005 429	30	631 286	314 500
	PacBio PBcR 25X + 454 25X		5 008 824	30	633 667	314 500
Melopsittacus undulatus (med: 997 max: 13 079)	Illumina 194X 220/500/800 paired-end 2/5/10Kb mate-pairs	1.23 Gbp	1 023 532 850	24 181	1 050 202	47 383
	454 15.4X FLX + FLX Plus + 3/8/20Kbp paired-ends		999 168 029	16 574	751 729	75 178
	454 15.4X + PacBio PBcR 3.75X		1 071 356 415	15 081	1 238 843	99 573

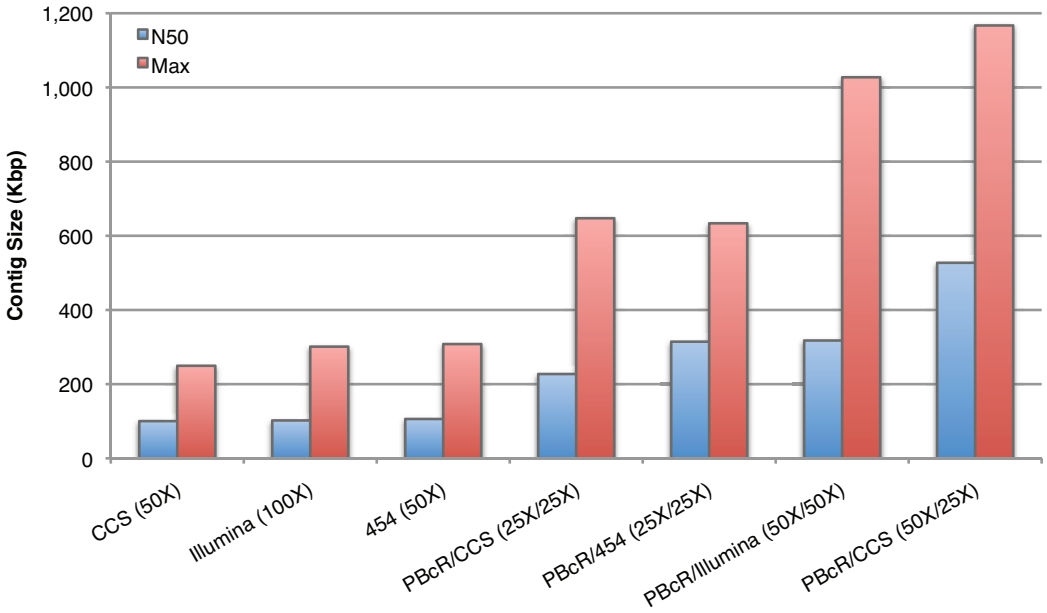


Figure 3.14: **Contig sizes for various combinations of sequencing technologies.** Assemblies are for *E. coli* C227-11 (assemblies using Illumina, PBcR (corrected by Illumina and PacBio CCS), and PacBio CCS) and *E. coli* JM221 (assemblies using 454, PBcR (corrected by 454)). Both genomes have similar repeat content, PacBio read length, and coverage. Assemblies of only second-generation data are comparable and average $N50 \approx 100$ Kbp. By comparison, adding 25 or $50X$ of PBcR to these data sets increases N50 as much as 5 fold and push the maximum contig size greater than 1 Mbp (for the PBcR/CCS combination).

Unlike de Bruijn approaches, which often benefit from high depth of coverage, extreme coverage (*e.g.* > 100X) can be detrimental to OLC assemblers. Too little coverage leads to a fragmented assembly because of sequencing gaps, and too much coverage accumulates sequencing errors in the string graph, which can fragment the assembly. To seek an appropriate coverage balance, single-pass PacBio reads for *E. coli* C227-11 were corrected using both 25X and 50X of CCS data. The hybrid reads were then assembled at 25, 50, and 75X coverage. The assembly quality plateaus when hybrid coverage matches the correction read coverage (*e.g.* 50X CCS plus 50X hybrid, Table 3.9). Intuitively, this is because the correction pipeline splits sequences at short-read coverage gaps. Therefore, the hybrid assembly is inherently limited by the correction read coverage. This potential limitation could be overcome in later stages of assembly by using the uncorrected PacBio reads for scaffolding, for example.

While more contiguous than second-generation sequencing, the assembly contiguity in Table 3.8 varies between genomes. We hypothesize that the performance difference is due to varying PBcR read lengths. To test our hypothesis, the average read length for each assembly in Tables 3.6 and 3.8 was calculated and plotted vs N50 (scaled by genome size). The result is shown in Figure 3.15. The degree of improvement correlates with the median length of the corrected reads, with the newer, longer reads yielding the biggest gains and the older technology producing only modest gains. Two clear outliers, *Lambda* and *M. undulatus*, are visible. This is expected, as *Lambda* is a phage genome with a low repeat content while *M. undulatus* is a complex eukaryote. For the others, the graph indicates strong agreement

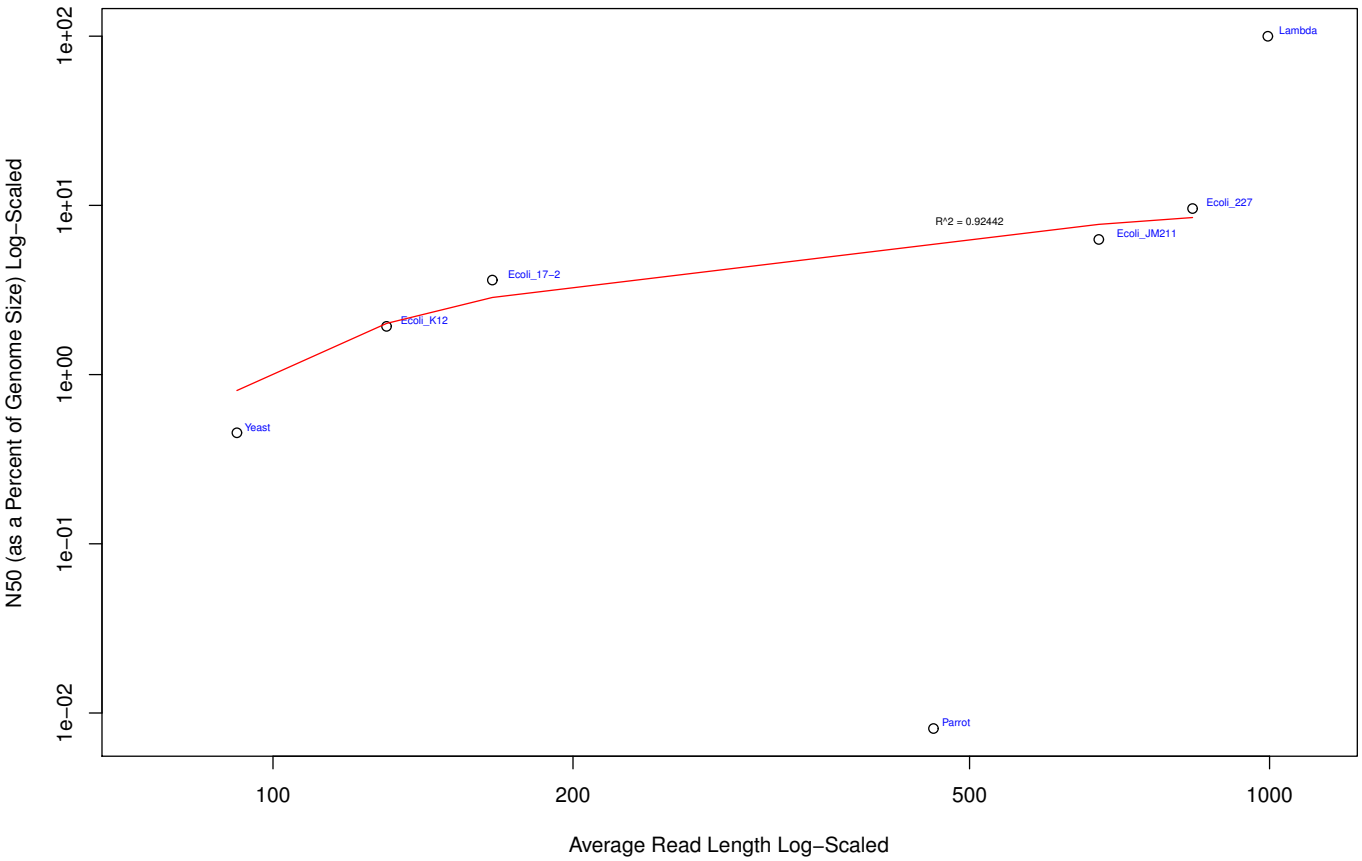


Figure 3.15: **Assembly contiguity is highly correlated with average read length.** The average read length for assemblies from Tables 3.6 and 3.8 is plotted against assembly contiguity (calculated as N50 scaled by genome size). There is a clear visual trend (supported by a log-linear model) of increasing contiguity as read lengths increase. The two outliers are present due to their relative genome complexity when compared with the other assemblies

between average read length and assembly contiguity. While there are only five samples used for the model, four are similar *E. coli* genomes with PacBio read length being the major variable. We believe this trend explains much of the variation in contiguity observed in Tables 3.6 and 3.8.

3.3.8 PBcR assembly outperforms long-rage paired-end libraries and second-generation long-read sequencing

Arguably, a diverse range of long insert sizes might be used to produce an assembly of comparable continuity. However, the available data listed in Tables 3.6 and 3.8 is limited to unpaired 454 and short-insert Illumina libraries (200–500 bp), with the exception of *M. undulatus*. To compare the effect of PBcR reads versus long-range pairs, we simulated ideal 3 Kbp and 6 Kbp Illumina long-range libraries at $50X$ coverage each for the *E. coli* C227-11 genome (with 10% standard deviation on insert length and no chimeric pairs or size/orientation artifacts). Even in this ideal case, the pure PacBio assembly of the PBcR (corrected by CCS) and CCS reads outperforms the Illumina-only assembly by 44% with an N50 of 527 198 versus 364 181. In addition, the combination of PBcR reads and Illumina short-range paired data produces an assembly nearly identical to the idealized Illumina long-range libraries (Table 3.9). As Illumina short-range libraries double the sequencing time and long-range libraries are difficult to construct, these results suggest long, single-molecule sequencing is a practical alternative to both.

The 454 sequencing platform has recently been upgraded to generate sequences up to 1 Kbp in length. The only genome for which both FLX+ and long PacBio sequences are available is *M. undulatus*, precluding a comparison due to the low and unequal coverage. To compare assembly contiguity between these long 454 sequences, called FLX+, and PacBio sequences, we generated error-free simulated sequences for the *E. coli* K12 genome. Using the observed read distributions in *M.*

Table 3.9: **PBcR corrected sequences outperform Illumina jumping libraries.** Technology: the read data used for assembly. Reference bp: the number of base pairs in the reference sequence used for N50 calculation. Total bp: the total number of base pairs in all contigs. # Contigs: The number of contigs comprising the assembly. Only contigs $\geq$ 10 000bp are included in results. Max Contig Length: The length of the max contig in the assembly. N50: Contig N50 size.

Organism	Technology	Ref BP	Asm BP	# Contigs	Max	N50
E. coli C227-11	Ill 100X 500bp	5 504 407	5 010 115	68	301 145	102 139
	Ill 50X 500bp + 50X 3Kbp		5 268 399	44	521 615	273 314
	Ill 50X 3Kbp + 50X 6Kbp		5 267 648	36	763 958	364 181
	Ill 50X 500bp + 50X 3Kbp + 50X 6Kbp		5 288 424	38	546 066	287 929
	PacBio 50X (Corrected by 50X Illumina)		5 342 166	35	915 367	318 612
	PacBio 50X + Illumina 50X 500bp		5 490 446	44	1 1027 387	317 661
	PacBio 25X (Corrected by 25X CSS)		5 207 946	80	357 234	98 774
	Pacbio 50X (Corrected by 25X CSS)		5 204 812	83	340 018	89 556
	PacBio 75X (Corrected by 25X CSS)		5 249 417	87	343 158	84 817
	PacBio 25X (Corrected by 50X CSS)		5 397 525	41	569 739	216 129
	PacBio 50X (Corrected by 50X CSS)		5 476 824	39	1 057 326	365 964
	PacBIo 75X (Corrected by 50X CSS)		5 601 310	55	642 068	308 312

undulatus for the FLX+ sequences as well as the post-correction PBcR sequences, we generated 18X coverage for each dataset. We then calculated assembly metrics as in Table 3.6. The FLX+ simulated assembly generated a total of 42 contigs with a maximum contig of 343Kbp and an N50 of 179Kbp. In contrast, the PacBio simulated assembly generated a total of 11 contigs with a maximum of 1 261Kbp and an N50 of 1 204Kbp. The increased contiguity of the PBcR assembly is due to the exponential read distribution generated by the PacBio RS sequencer, with over 30% of the PBcR bases being in reads of 2Kbp or greater. By contrast, all FLX+ sequences were shorter than 2Kbp.

3.3.9 Prokaryotic repeat resolution

The assembly gains are striking because they are entirely a result of resolving repeat structure rather than closing so-called "sequencing gaps" in the short-read coverage (Section 3.2.4). Thus, the PBcR reads are uniquely suited for closing

difficult gaps left by second generation technologies, such as complex tandem repeats (*e.g.* VNTRs and STRs).

Repeat resolution occurs when a read spans a repeat and is anchored by the surrounding unique sequence. Longer reads are capable of spanning a greater variety of repeats, leading to better assemblies. Repeat classes fall in two broad categories: interspersed and tandem. Paired reads, either with Illumina or 454, can be used to resolve many simple interspersed repeats. Here, an advantage of long reads is in library prep, by removing the need for paired libraries, which can be difficult and costly to construct. Paired ends fail to resolve repeats when their short ends cannot be uniquely anchored. Similarly, tandem repeats can be very difficult to resolve using only read pairing. For example, a 10 bp element repeated 100 times is too long (1 000 bp) to be spanned by a second generation read, and pair libraries do not have the resolution to determine the number of copies (the difference between 99 and 100 copies is only 10bp, which is shorter than the typical size variation seen in 1 000bp insert libraries). These types of repeats, such as VNTRs and STRs, have important biological functions and make powerful genotyping tools, so their correct assembly is important. The long, continuous PBcR reads allow the assembly of such sequences, which is not always possible with other technologies. Figure 3.16 shows three common types of repeats resolved by PBcR reads in bacterial genomes that were left un- or mis-assembled using 454 reads: interspersed, inverted, and tandem.

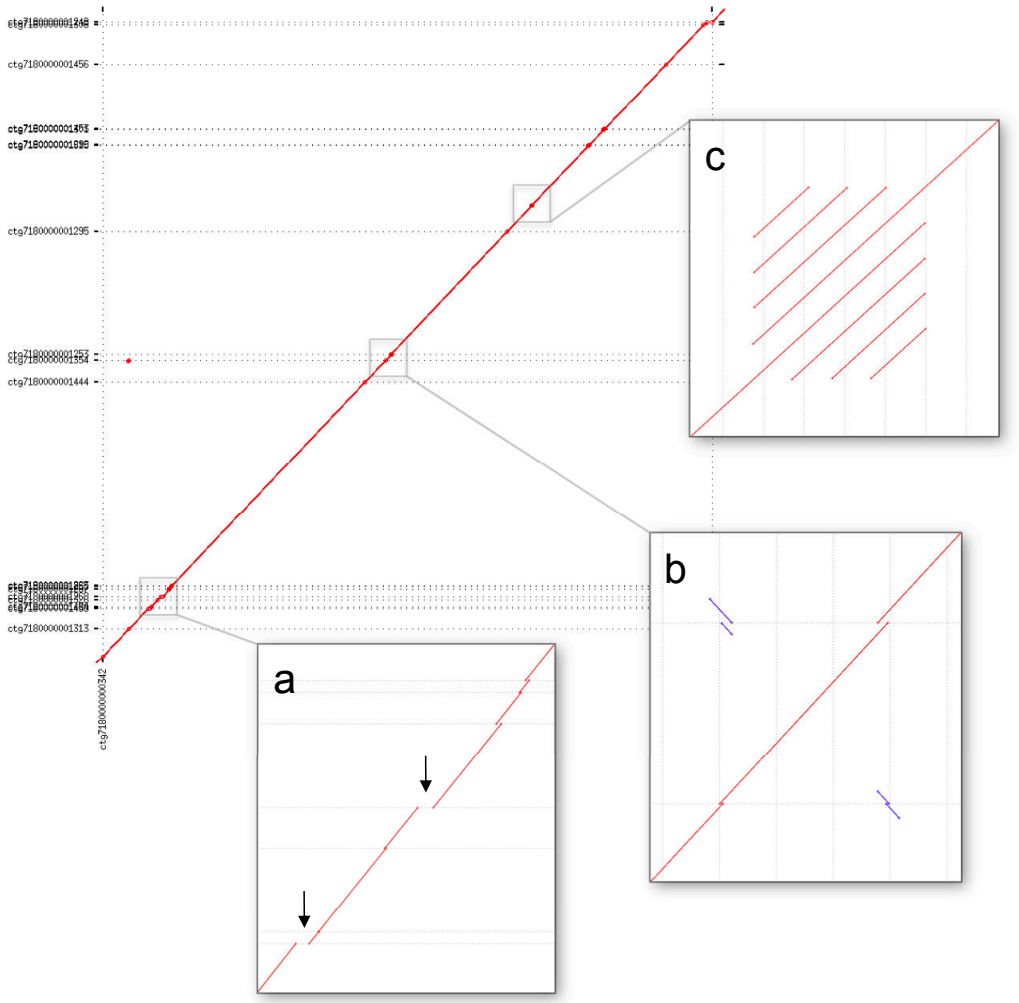


Figure 3.16: **Example repeats resolved by PBcR sequences in *E. coli* JM221.** A dotplot shows the alignment of a single 1 Mbp hybrid PBcR contig to the corresponding 454 contigs. This single PBcR contig closes 18 gaps left in the 454 assembly. Each horizontal dotted line indicates the boundary of a 454 contig and the contigs are arranged in order of their appearance in the PBcR contig. Three repeats resolved by PBcR but not 454 are highlighted. a) The two black arrows point to 1.4 and 1.8 Kbp gaps in the 454 assembly. These represent two different interspersed repeat families that appear in the genome in multiple copies, but were collapsed into single contigs elsewhere in the 454 assembly. Because the long PBcR reads were able to span these repeats, the gaps were closed. b) The blue, negative diagonal alignments indicate an inverted repeat of $\sim$ 800 bp bounding a region of 5 Kbp. The 454 reads were unable to resolve this repeat structure, but the region was closed by PBcR reads. c) This alignment motif represents a tandem repeat with a unit length of $\sim$ 100 bp, repeated 4 times, spanning $\sim$ 400*bp*. The 454 assembly has mis-assembled the region by inserting an extra copy of the repeat. This type of tandem repeat slippage (either expansion or collapse) is a common mis-assembly seen in second-generation data and is very difficult to resolve without a full read spanning the entire region.

3.3.10 Low long-read coverage impact on assembly

Long reads are capable of producing better assemblies, even at greatly reduced coverages. A comparison of the literature shows that a 10–20X Sanger assembly is better than a 100X Illumina assembly, albeit with prohibitively greater sequencing costs using the older technology [86, 141, 45]. We found that for *S. cerevisiae* S228c, an assembly using 13X of PBcR data (corrected by 50X Illumina) is comparable to an assembly of 100X of paired-end Illumina data (Table 3.6). This is true despite the fact that sequencing was performed using a pre-release instrument. The corrected PacBio sequences also generate a more accurate assembly: while the 100X of Illumina produces a slightly longer raw N50, after splitting contigs at assembly errors, the N50 is larger for the PBcR assembly. Another striking example is *E. coli* JM221, for which the 25X PBcR assembly triples the N50 of the 50X 454 assembly.

Given the evident ability of PBcR reads to improve assemblies, the additive benefit of supplementing second-generation data was measured using *E. coli*. Between 1X to 50X of corrected PacBio data was added to the short read data for an existing assembly (Fig 3.10(b)). The large and rapid gains after the addition of long-read sequencing are readily apparent. At just 10X coverage, nearly the maximum N50 is reached for the second-generation/PBcR assembly. The N50 measures a 2.5 and 3.5-fold improvement over the Illumina and 454 assemblies, respectively. These results demonstrate significant improvements in continuity without the need for paired libraries and at relatively minor coverages. Thus, one might expect roughly double the N50 contig size with the addition of just 20X raw PacBio sequencing

(assuming a throughput of $> 50\%$ during correction).

3.3.11 Assembling the parrot genome

Demonstrating applicability to vertebrate genomes, we successfully assembled the *Melopsittacus undulatus* genome using the PBcR pipeline. A total of 5.5X PacBio sequences (4M) was corrected using 54X of Illumina paired-end reads (660M), producing 3.75X of PBcR sequences for a throughput of 68.38%. The correction required 20K CPU hours (6.75 days wall-clock time) using a peak of $176GB$ of memory and 121 effective cores. Based on these performance results, correcting a human genome with matching coverage would take 61K CPU hours, which can be completed in 10.16 days with an effective parallelism rate of 250 cores. By comparison an ALLPATHS-LG Illumina assembly and a Celera Assembler 454 assembly each took over 1 week to complete, with the Celera Assembler using the same number of cores as PBcR. Thus, the correction represents an approximate doubling of the total assembly time.

Following the method presented in [141], we evaluated the repeat complexity of the parrot genome in comparison with several other genomes in Figure 3.17 using the tallymer tool [82]. This analysis is sensitive to assembly quality, so we measured both the Illumina and PBcR assemblies. If an assembly over-collapses repeats, for example, the assembly will measure more unique than the truth. The figure shows that the *M. undulatus* genome is at least as complex to assemble as the human genome (as measured for the Illumina assembly) and is likely more comparable to

the fruit fly and mouse genomes (as measured for the PBcR assembly). The mouse genome, for example, is known to have 2.25 to 3.25 fold more simple sequence repeats than the human genome [175], even though it has fewer interspersed repeats. In many cases, it is the simple repeats that are the most difficult to assemble. Even though *S. cervisiae* has a small genome, it is also complex for long k relative to its genome size. This demonstrates that because the PBcR pipeline performed well on both *S. cervisiae* and *M. undulatus*, it can be expected to perform well on other high-complexity genomes, such as human.

Because *M. undulatus* is a novel genome without an available reference, correction accuracy was estimated by mapping PBcR reads to all assemblies (except our own) submitted as part of the Assemblathon 2 [26]. For this diploid genome, each assembly can be thought of as a mosaic of the two haplotypes, so only the assembly containing the best mapping for each PBcR read was considered in an effort to match the correct haplotype. Using this method 99.7% of the PBcR reads had at least one mapping, with 93.7% mapping end-to-end with an average identity of 99.5%. Of the 6.2% of reads with partial or fragmented mappings, 2.5% have breakpoints internal to a contig, which provides a rough estimate of chimerism. The remaining 3.7% map to contig boundaries and their accuracy cannot be determined. Considering likely haplotype switching within Illumina contigs, the slight increase in estimated error is not unexpected. Overall, the PBcR reads show good congruence with the independent assemblies.

The PBcR reads were then co-assembled with $15.4X$ of 454 reads, which included 3, 8, and 20 Kbp libraries, to leverage the complementarity of the three

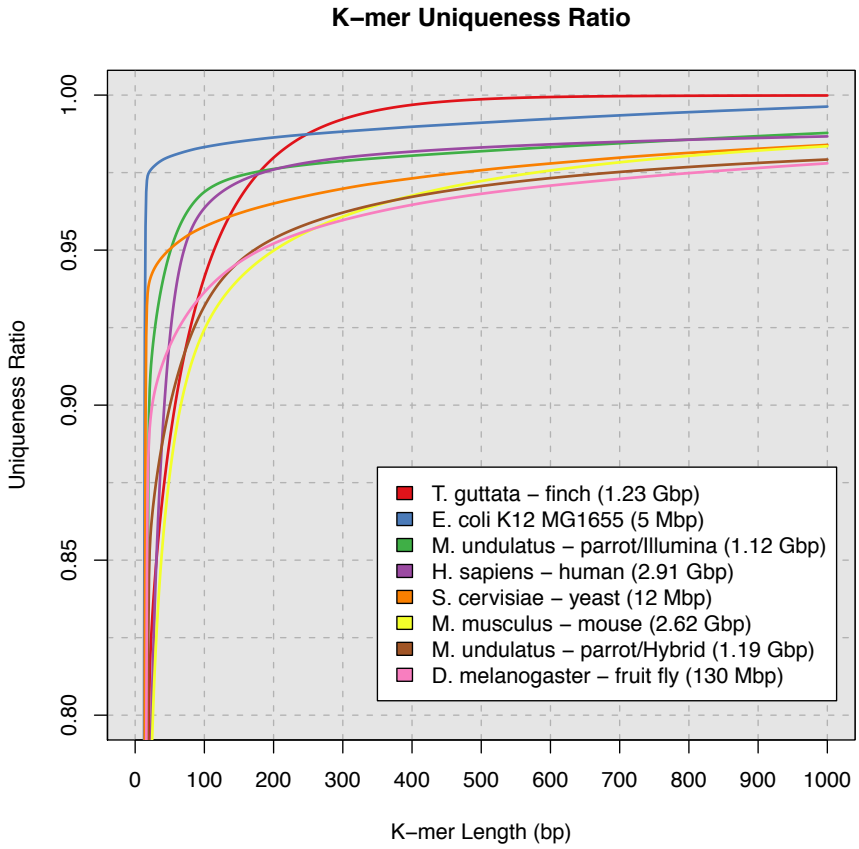


Figure 3.17: **k-mer uniqueness for five well-known eukaryotes.** The ratio is defined here as the percentage of the genome that is covered by unique sequences of length k or longer. The horizontal axis shows the length in base pairs of the sequences. For example, 97.5% of the human genome is contained in unique sequences of 200bp or longer. In contrast, only 95% of the parrot genome is contained in unique sequences at the same 200bp length. The figure demonstrates that parrot repeat complexity is higher than that of human, closer to *D. melonogaster* and *M. musculus* rather than *H. sapiens*.

sequencing technologies and provide a diverse set of insert lengths. For comparison, two additional assemblies were generated. One running Celera Assembler with identical parameters but on the 454 data only, and a second running ALLPATHS-LG on 194X of Illumina data, including 0.22, 0.5, 0.8, 2 Kbp, 5 Kbp, and 10 Kbp libraries. ALLPATHS-LG has been shown to be an effective short read assembler for large genomes [137, 26], and serves as an appropriate benchmark for assembling this genome using only Illumina data. The results of the PBcR, 454, and Illumina assemblies are included in Table 3.8. However, a co-assembly of just the 454 and Illumina data was not possible because Celera Assembler does not support high-coverage Illumina data and ALLPATHS-LG does not support 454.

The PBcR/454 assembly, with an N50 contig size of 100 Kbp, is more contiguous than the second generation assemblies in Table 3.8 and compares favorably with previous avian genome assemblies sequenced using the "gold-standard" Sanger method. The zebra finch (*Taeniopygia guttata*) was sequenced to $6X$ coverage using Sanger sequencing, generating a maximum contig of 424 635 and an N50 of 38 549 [174]. The chicken *Gallus gallus* was also sequenced using Sanger to $7.1X$, resulting in a maximum contig of 624 663 and an N50 of 45 280 [58]. In contrast, for genomes assembled using only short-read sequencing, the N50 contig size rarely exceeds 30 000 bp [141, 45]. Much of the parrot genome continuity can be attributed to the long-read 454 data, including a mix of library sizes and the latest 454 chemistry nearing 1 Kbp reads, but the addition of just $3.75X$ PBcR sequences results in a 65% increase in maximum contig length and 32% increase in N50 (Table 3.8), on par with the projections from Figure 3.10(b). However, these PBcR sequences

were corrected using Illumina data, so the gains are not directly comparable to the 454-only assembly. Instead, the PBcR hybrid assembly demonstrates the utility of combining a diverse set of sequencing technologies, something that is not currently supported by most large-genome assemblers.

In addition to the continuity, the overall quality of the contigs remains high after the addition of the PBcR reads. Improvement to long-range accuracy is supported by satisfaction of both assembled 454 pairs and mapped Illumina mate-pairs. The fraction of unsatisfied 454 pairs drop from 0.056% in the 454 assembly to 0.051% for PBcR. The fraction of satisfied 10Kbp Illumina mate-pairs increases from 50.80% to 51.62% in the PBcR versus the 454-only assembly (an increase of 1M or 0.82%).

The Illumina mate-pairs, not utilized in the PBcR assembly, were also used to test for the presence of chimeric joins. We calculated clone coverage for each base of the assembly. The clone coverage is incremented for any bases that are spanned by a satisfied mate, along with the bases within the mate sequences themselves. Unsatisfied (wrong orientation, stretched, or compressed) mates do not contribute to the clone coverage. If an assembly contains a chimeric joins, no pairs are expected to span the join with the correct separation and orientation. This has previously been shown to be an effective indicator of mis-assembly [123, 169]. Confirming correctness, the percentage of bases not covered by satisfied 10Kbp mates was tabulated to be 0.11% in the 454-only assembly versus 0.16% in the PBcR assembly, indicating almost no change.

The gaps closed by PBcR reads were also evaluated for correctness and their clone coverage was compared to the overall assembly (Fig 3.18). Both histograms

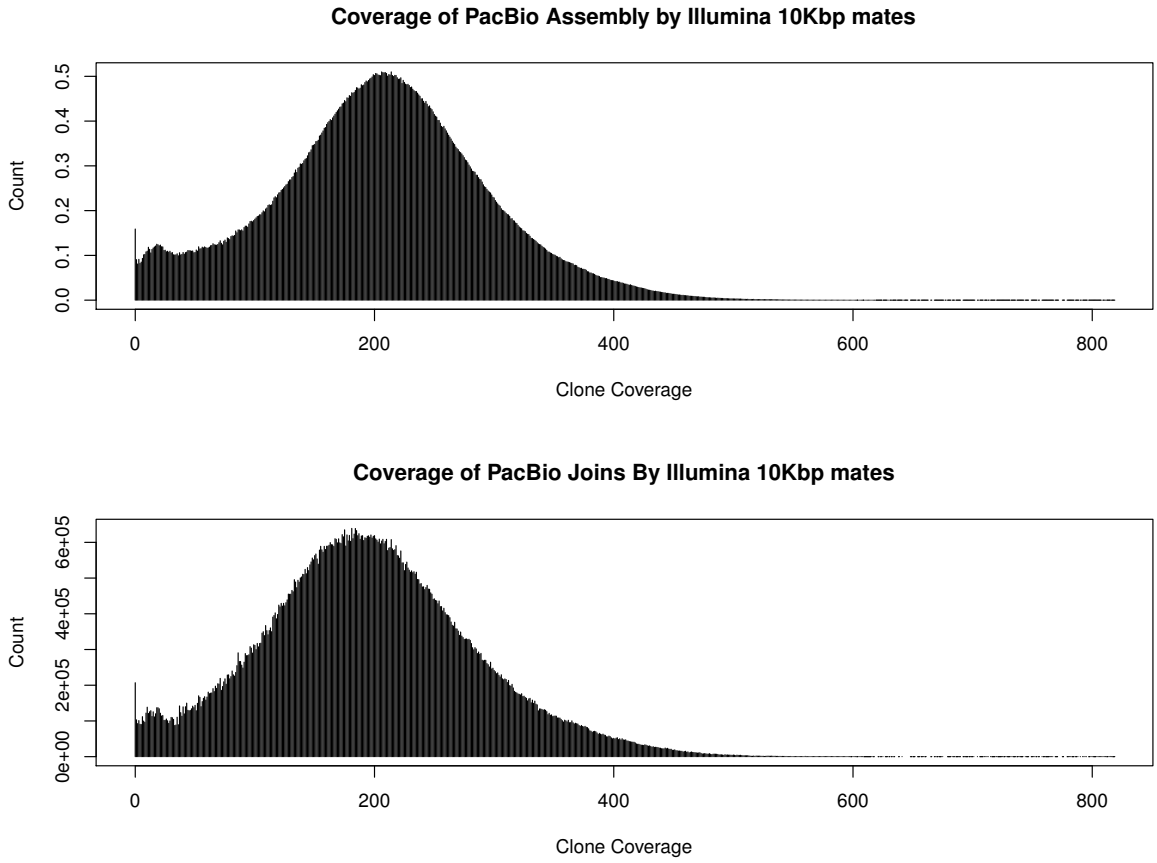


Figure 3.18: **PBcR contigs are supported by Illumina mate-pairs.** The histograms show the per-base clone coverage by satisfied mate-pairs for the full PBcR/454 assembly compared to junction regions in the PBcR assembly versus the 454-only assembly. The histograms both show a strong peak at 200X clone coverage. There is a low rate of 0X coverage regions in both histograms (corresponding to 0.16% of the bases in both cases). The peak at 20X coverage could correspond to low-complexity regions that could not be sequenced by Illumina or a mapping artifact.

have a peak at approximately 200X with the same percentage of bases with no clone coverage (0.16%) and no indication of a higher rate of bad paired-end sequences surrounding PBcR junctions. The clone coverage by satisfied pairs across PBcR junctions by an independent library confirms that the assembled contigs are well supported.

Completeness and correctness of the PBcR assembly is also supported by aligned transcripts. Of the 15 275 zebra finch mRNA sequences currently anno-

tated in GenBank, approximately 95% are partially mappable to the three parrot assemblies using the gmap spliced aligner [181]. All three assemblies show good structural agreement with the transcripts with only 83, 86, and 85 chimeric mappings to the PBcR, 454, and Illumina scaffolds, respectively. Despite its smaller contigs, ALLPATHS-LG appears very effective at assembling and scaffolding exons, and its assembly contains an additional 1–2% of the transcript bases than the others (23.94Mbp PBcR, 23.78% 454, 24.26Mbp Illumina). However, the long read assemblies excel at reconstructing the often complex intronic sequences, with PBcR and 454 splitting 23% and 18% fewer transcripts across contigs than Illumina (1 320 PBcR, 1 405 454, 1 708 Illumina). This results in a greater fraction of each transcript being contained on a single contig (Fig 3.19, Table 3.10), and the number of transcripts mapped to a single contig at over 80% length is 15% and 6% greater for the PBcR and 454 versus Illumina assemblies (8 876 PBcR, 8 156 454, 7 719 Illumina). We extracted the mapping for the well-studied FOXP2 mRNA transcript (NM_001048263.1) and found that 94.1% maps to a single 504 945bp contig in the PBcR assembly, while only 80.5% is contained in smaller 163 917bp and 119 070bp contigs in the 454 and Illumina assemblies, respectively. Thus, the PBcR assembly shows improved intron reconstruction, but the Illumina assembly maintains slightly higher exon coverage. This suggests that high-coverage Illumina sequencing may be a good complement to low-coverage PacBio RS sequencing for complex genomes, but additional development is needed to adapt assemblers to this challenge.

This demonstrates the utility of corrected, long sequences to improve both contig length and quality, and better resolve the complex sequence of non-coding

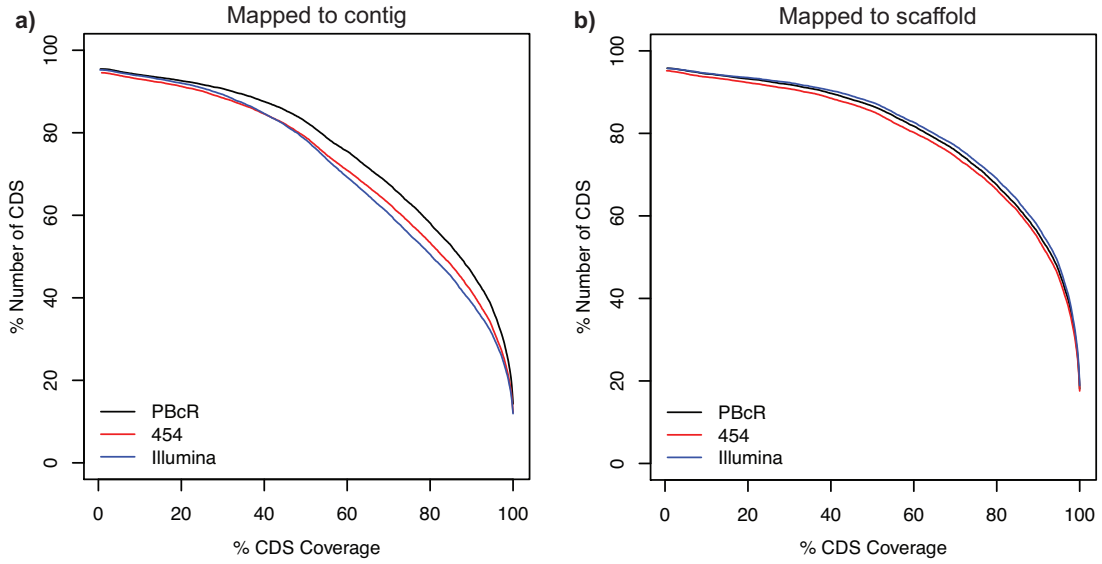


Figure 3.19: **PBcR assembly demonstrates better eukaryotic transcript coverage.** The cumulative plot shows the percentage of transcripts mapped at a minimum percentage in a single contig (left) and scaffold (right) in each parrot assembly. The Y axis is the total number of transcripts mapped at or below the coverage on the X axis. Perfect mapping is not possible between different species, but curves shifted closest to Y=100% represent assemblies with the best transcript coverage. The Illumina assembly shows the best transcript coverage in scaffolds, with the PBcR assembly having 1–2% less transcripts at a given coverage. This demonstrates that the high-coverage Illumina assembly is able to accurately reconstruct and scaffold the exons. In contrast, the PBcR assembly has a higher percentage of transcripts covered in a single contig at any given coverage than either the Illumina-only or 454-only assembly, with the Illumina-only assembly having the lowest percentage coverage. This suggests that the high-coverage Illumina assembly is able to accurately reconstruct and scaffold the exons, while the advantage of the long reads lies in resolving complex intron and non-coding sequences.

Table 3.10: **PBcR demonstrates better eukaryotic transcript coverage.** The table shows the values plotted in Figure 3.19 above. The Min % Coverage corresponds to the number of transcripts with at least this coverage in a single mapping to a contig or scaffold.

	Contigs			Scaffolds		
Min % Coverage	PBcR	454	Illumina	PBcR	454	Illumina
10	14 366	14 204	14 326	14 424	14 308	14 445
20	14 147	13 939	14 058	14 240	14 101	14 287
30	13 853	13 511	13 641	14 030	13 871	14 103
40	13 373	12 918	12 929	13 702	13 520	13 806
50	12 628	12 049	11 962	13 233	13 033	13 365
60	11 536	10 830	10 575	12 489	12 256	12 631
70	10 327	9 605	9 225	11 584	11 365	11 758
80	8 876	8 156	7 719	10 316	10 139	10 547
90	7 056	6 333	5 927	8 515	8 320	8 759
100	2 181	1 867	1 821	2 757	2 683	2 886

regions, even when long-range libraries are available.

3.3.12 Towards single-contig chromosomes

All our datasets are based on the second-generation PacBio chemistry, with an uncorrected median read length of $\approx$ 2 Kbp. To estimate effects of increasing read length for *E. coli* K12, we generated several exponential distributions with increasing medians and assembled the resulting error-free read sets for each. We found that the exponential distribution with a median of 1 600 assembles *E. coli* K12 into a single chromosome. Given our current observed sequence loss due to trimming (median 2 553 trimmed to 1 216 for *E. coli* JM221, or 48% after trim), this corresponds to a median, uncorrected length of approximately 3 350. However, PacBio read lengths do not exactly follow an exponential distribution and the simulations always assemble better than real data. Thus, we roughly estimate that a median, uncorrected length of 3.5 Kbp will enable single contig assemblies for E. coli K12 at 50X coverage. At this level of coverage, a sufficient fraction of the reads are long

enough to span the largest repeat family in *E. coli*, which is approximately 5.5 Kbp. A median read length of 3.5 Kbp represents a seemingly achievable 40% length increase over the current median length produced by the PacBio RS. However it is difficult to predict when this threshold will be reached because it is unclear how the read length distribution may scale for future chemistries.

3.3.13 Single-molecule RNA-Seq correction

Finally, we explored the application of PBcR in transcriptome analysis. Since the length of the single-molecule PacBio reads (ranging from a few hundred bases to several kilobases) from RNA-Seq experiments is within the size distribution of most transcripts, we expect many PacBio reads will represent full-length or near full-length transcripts. These long reads can therefore greatly reduce the need for transcript assembly, which requires complex algorithms for short reads [160]. By aligning long PacBio reads to the reference genome to reveal the exon structure, alternatively spliced isoforms should be easily detected. However, aligning these reads to the reference genome is problematic because of the predominance of indel errors and the lack of algorithms to efficiently align PacBio RNA-Seq reads to the genome. For example, in this study we generated 50 130 PacBio reads with a median size of 817 bp from a *Zea mays* B73 seedling mRNA sample, but only 11.6% (15 173) of the reads can be aligned to the reference genome by BLAT [72] at $> 90\%$ sequence identity. In contrast, for the corrected PBcR sequences (using 17.8X of Illumina data completing in 3.56 days of wall-clock time and a peak of 225GB of memory), the

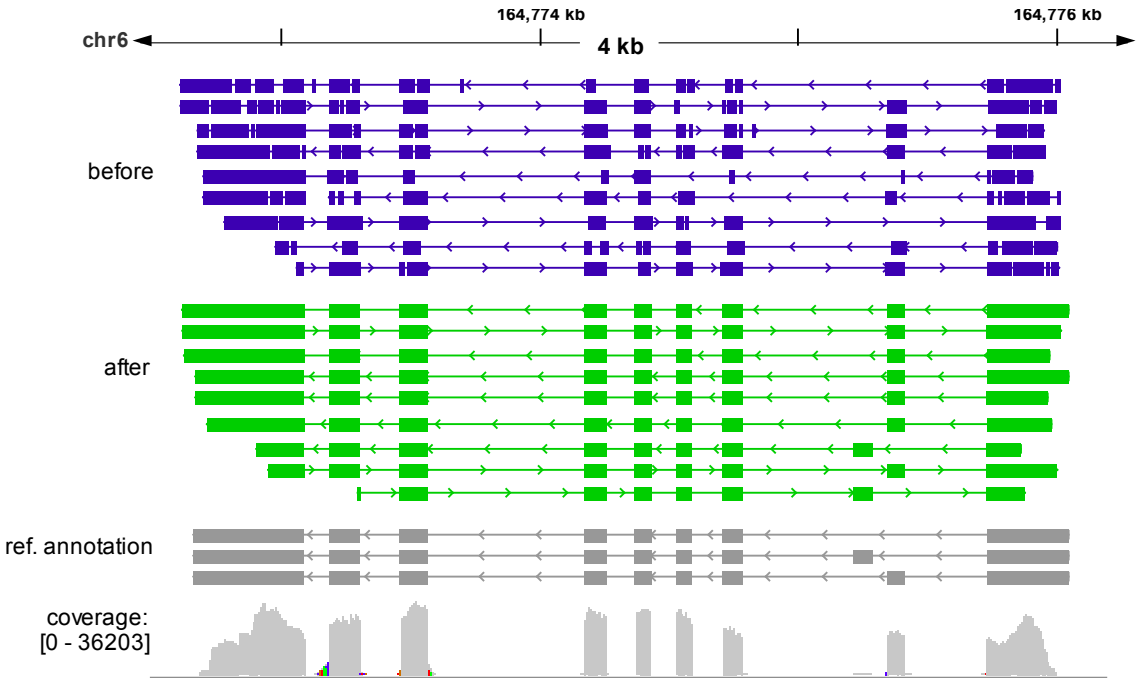


Figure 3.20: **Error correction of RNA-Seq data provides more accurate mapping of novel transcripts.** A genome browser view of alignments using uncorrected (purple) and Illumina-corrected (green) PacBio reads generated from *Zea mays* B73 cDNAs. The splice-aware aligner, BLAT [72], was used for aligning PacBio reads to the genome. Long gaps in the alignment correspond to introns in the PacBio reads but not the reference genome, and short gaps (only visible in the pre-corrected PacBio reads) are putative indel errors. The read coverage of the Illumina reads used for correction is also shown, along with the current reference gene annotation for the displayed reference locus. The colored bars in read coverage are an artifact of the aligner, indicating reads which have overhangs across exon junctions. The corrected PBcR sequences match the reference annotations end-to-end. Genome coordinates for chr6 are shown from the RefGen v2 genome assembly (http://maizesequence.org/).

percentage of sequences that align at $> 90\%$ identity increases dramatically to 99.1% (49 679 reads). Consistent with the results reported above for genome assembly, the corrected RNA-Seq sequences have very low error rates, with only 0.06% insertion and 0.02% deletion rates.

As shown in Figure 3.20, many PacBio reads indeed represent close to full-length transcripts. However, the exon structure is not evident before the error

correction by PBcR. The post-correction sequences have virtually no errors and precisely identify splicing junctions. As a result, two of the isoforms at this locus in the reference annotation were confirmed by PacBio RNA-Seq reads. To systematically test the ability of PacBio reads to validate annotated gene structure, we aligned the PacBio reads to the reference genome, and looked for PacBio reads that matched the exon structure over the entire length of the annotated transcripts. Before correction, only 41 (0.1%) of the PacBio reads exactly match the annotated exon structure. This number rises sharply after correction to 12,065 (24.1%), suggesting that PBcR can greatly increase the usefulness of the PacBio RNA-Seq reads for transcript structure annotation or validation.

3.4 Discussion

Current *de novo* assemblers are unable to effectively use the long-read sequencing data generated by present single-molecule sequencing technologies primarily because of the considerable error rate. Our approach exploits this technology by complementing it with shorter, high-identity sequences resulting in long, accurate transcripts and improved assemblies for both large and small genomes. Since the average contig size produced by our approach correlates with read length, assembly results are expected to improve as the read lengths of the technology improve. This strategy also benefits from the complementarity of multiple technologies, which proved powerful when combining Sanger sequencing with second-generation data when it first became available [47]. The result of our hybrid approach is higher

quality assemblies with fewer errors and gaps, which will drive down the expensive cost of genome finishing and enable more accurate downstream analyses.

High-quality assemblies are critical for all aspects of genomics, especially genome annotation and comparative genomics. For example, many microbial genomic analyses depend on finished genomes [37], but producing finished sequence remains prohibitive with the cost of finishing proportional to the number of gaps in the original assembly. Eukaryotic genomics requires continuous assemblies to capture long, multi-exon genes and to determine genome organization and structural polymorphisms. In addition, recent work has suggested *de novo* assembly may be superior to read mapping approaches for discovering large structural variations, even when a reference genome is available [90]. This is especially significant for understanding the genetic variations of cancer genomes and other human diseases such as autism that frequently contain gene fusions, copy number variations, and other large scale structural variations [33, 146]. It is clear that higher quality assemblies, with long unbroken contigs, will have a positive impact on a wide range of disciplines.

Future improvements to the PBcR pipeline would benefit from the addition of a gap closure routine to fill potential sequencing gaps in the short-read data, and appropriately weighing the single-molecule base calls during consensus calling. This is particularly important for GC-rich sequences that tend to be underrepresented by second-generation sequencers, and for metagenomic and amplified samples that have severe coverage fluctuations. Non-uniform coverage will also require modifications to the repeat separation algorithm, since the current heuristic assumes uniform long-read coverage and error. For example, by utilizing paired-end information or variant

clustering to constrain the mapping during correction.

3.5 Conclusion

We have demonstrated that high error rates need not be a barrier to assembly. High-error, long reads can be efficiently assembled in combination with complementary short-reads to produce assemblies not possible with any prior technology, bringing us one step closer to the goal of "one chromosome, one contig." The rapid turnaround time possible with PacBio and other technologies such as Ion Torrent [135] will make it possible to produce high-quality genome assemblies at a fraction of the time once required. Future studies are needed to explore the relative costs and trade-offs of the available technologies, but from our results we anticipate future sequencing projects will consist of a combination of both long and short-read sequencing. Today this is particularly necessary for effective long-range scaffolding ($\geq$ 9 Kbp pairs), for which the current PacBio reads provide limited assistance. However, if single-molecule technology continues to advance and reads begin to exceed the lengths of typical bacterial repeats ($\approx$ 6 Kbp) at reasonable cost and throughput, single-contig assemblies of some bacterial chromosomes will be possible. Additionally, we believe many long sought capabilities will be enabled, such as haplotype separation in eukaryotes, accurate transcriptome annotation, and true comparative genomics that extends beyond an exon-centric view to include the whole genome.

Part II

Applying Assembly to Novel Targets

Chapter 4

Towards Automated Metagenomic and Single-cell Analysis[†]

4.1 Overview

Metagenomics, the direct sequencing of DNA from all organisms in an environment without culturing, has recently emerged as a new scientific field that enables the discovery of novel organisms and genes [182] - as well as the study of population structure and dynamics [5, 74]. Metagenomic studies have greatly expanded the understanding of microbial diversity. For example, viral quasi-species have been shown to affect pathogenicity in the poliovirus due to cooperation between differently adapted individuals in a population, as well as between co-infecting viruses [170]. Other recent studies have relied on metagenomics to identify novel genes and uncultured microbes [57].

The assembly of metagenomic data is complicated by several factors such as: (i) widely different levels of representation for different organisms in a community; (ii) genomic variation between closely related organisms; (iii) conserved genomic regions shared by distantly related organisms; and (iv) repetitive sequences within individual genomes. Similar challenges occur in the assembly of polymorphic eukaryotes, a challenging domain for existing assembly algorithms. For example, the

[†]The text of this chapter is based on the publication S. Koren, T. Treangen, and M. Pop. Bambus 2: scaffolding metagenomes. *Bioinformatics*, 27(21):2964–2971, 2011. Sections 4.4.2, 4.4.3, 4.4.7, 4.3.1, and 4.3.6 represent unpublished work.

assembly of the sea squirt genome *Ciona savignyi* required extensive manual intervention and customized scripts despite the fact that this genome is fairly "simple" – there were only two haplotypes of roughly equal coverage [171]. Metagenomic data are considerably more complex. Due to the lack of assembly tools specifically targeted at metagenomic projects, studies rely on existing assemblers and attempt to mitigate some of the challenges posed by the data through iterative adjustment of assembly parameters and post-processing. Tuning is critical as existing assemblers make frequent errors even in simulated datasets with significantly lower complexity than true environments [98]. At the same time, current assemblers produce fragmented assemblies, hampering downstream analysis. For example, in the analysis of the Global Ocean Survey data, the Celera Assembler [110] was heavily modified to allow high error rates in order to account for strain variation, and to overcome the effects of varied coverage levels on the statistical repeat detection procedure [168, 136]. Only two assemblers were developed specifically for metagenomic datasets [84, 117]. However, neither utilizes mate-pairs, our focus in this work.

We present novel scaffolding algorithms optimized for non-clonal assembly. Though our algorithms are also applicable to polymorphic genomes, the primary focus of this paper is on metagenomic analysis. These algorithms are implemented in a software tool called Bambus 2. Bambus 2 supersedes our previous scaffolder, Bambus [126], which was targeted at clonal Sanger data. We will show that, when applied to metagenomic datasets, Bambus 2 generates large scaffolds while avoiding false joins between distantly related organisms. Furthermore, our software can automatically identify genomic regions of variation that correspond to previously

characterized polymorphic loci.

4.1.1 Metagenomic scaffolding

In our opinion, the main challenge in metagenomic assembly is to develop an assembler that can automatically generate contiguous assemblies yet accurately capture genomic variation information throughout the assembly process.

It is important to first define the basic concepts underlying genome scaffolding. Most modern genome assemblers start by reconstructing segments of the genome that are unambiguously defined by the set of reads. These segments – called unitigs – are sections of the genome entirely contained in either unique regions, or in repeats, *i.e.* they do not span the boundary between individual repeats or between repeats and unique regions. The nucleic acid sequence of unitigs can, therefore, be unambiguously reconstructed.

Irrespective of the assembly algorithm employed, the unitigs themselves are generally small and assembly software must use additional information to increase the size of the contigs produced. Commonly, assemblers leverage the information contained in mate-pairs – information constraining (in orientation – the DNA strand from which the sequence originated – and approximate distance) the pairwise position of reads along the genome. The process through which mate-pair information is used to increase contig sizes, as well as to determine a global arrangement of contigs along the genome, is called scaffolding. Note that longer contigs can also be constructed by careful analysis of the assembly graph without the use of mate-

pair information [111, 73] – we broadly consider scaffolding to also include such analyses. Most existing genome assemblers contain dedicated scaffolding modules (*e.g.* [110, 13, 184, 89]). The unitig graph is output by a variety of modern assemblers such as Newbler [97], Celera Assembler [110], and SOAPdenovo [89], allowing scaffolding tools to operate as a stand-alone module post-assembly [126, 21, 41]. Throughout the paper we will assume that the unitig graph is given and will demonstrate how this information can be used to effectively analyze metagenomic datasets.

Genomic repeats are the major challenge when assembling isolate genomes, and their effect is compounded in metagenomic datasets. Repeats link together disparate sections of the genome. As the number of reconstructions grows exponentially with the number of repeats [73], it is intractable to find the one correct reconstruction. Therefore, most assemblers start by masking out unitigs that appear to represent repetitive segments of a genome. Celera Assembler, for example, uses depth of coverage statistics to determine whether a particular unitig represents a repeat, then ignores these unitigs until the later stages of scaffolding [110]. Coverage statistics are also used in other assemblers [13, 184, 21]. An alternative approach relies on topological information: unitigs that have multiple conflicting neighbors [89] can be inferred to represent repeats.

While the approaches described above work well in isolate genomes, they can lead to false positives in metagenomic datasets. Coverage-based methods can classify abundant organisms as repeats, preventing the assembly of exactly those segments of the community that should be easily assembled [168]. Distinguishing between repeats within the same genome and conserved genomic segments shared by closely

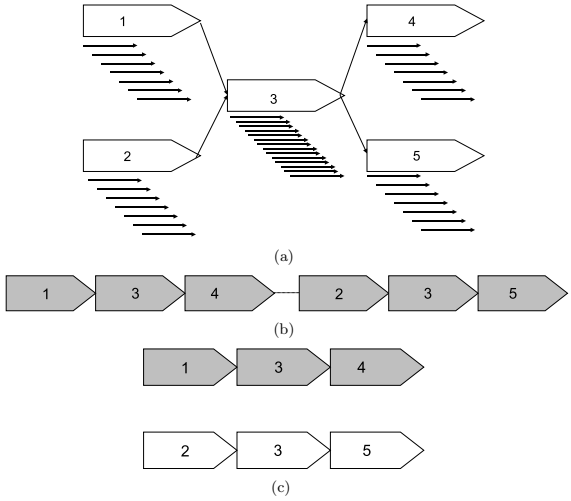


Figure 4.1: **Local coverage metrics are misleading in a metagenomic assembly.** (a) The unitig graph representation of a single unitig, 3, having double the coverage of the surrounding unitigs. Solid black arrows correspond to reads comprising a unitig. (b) One of the possible resolutions of the graph presented in (a). This example places unitig 3 in two locations along a single genome. (c) A second of the possible resolutions of the graph presented in (a). This example places unitig 3 at the same location in two genomes (highlighted in different colors).

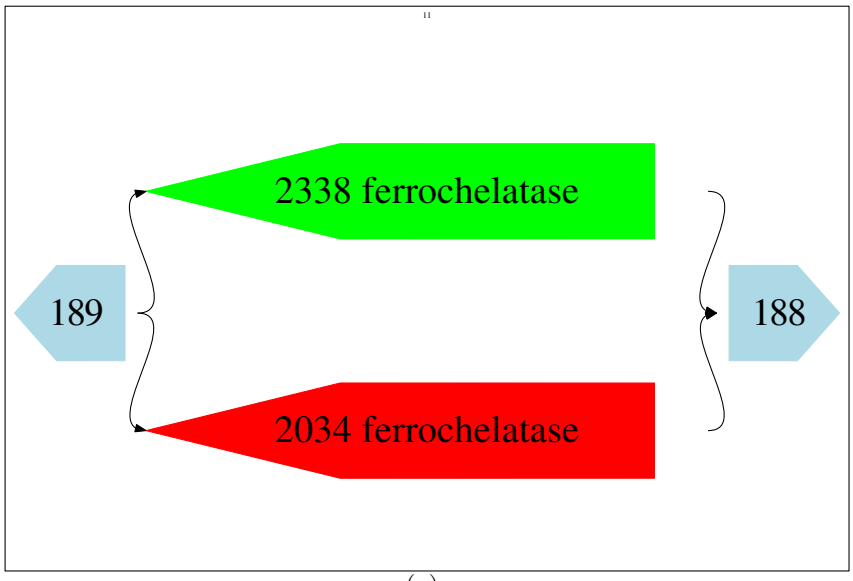
related organisms can be difficult. As seen in Figure 4.1, the local unitig graphs and coverage look identical in both cases. Below we will describe new approaches for repeat detection that work well in metagenomic datasets.

Currently available scaffolders attempt to construct linear scaffolds, *i.e.* where unitigs can be placed in a linear, non-overlapping order. When multiple unitigs occupy the same genomic region, they are either collapsed into one or the scaffolds are broken apart. Collapsing unitigs assumes the differences are due to error [183]. Breaking scaffolds assumes the ambiguity is due to repeats, [21]. In metagenomic assembly, such bubbles (multiple contigs occupying the same position in the assembly) are common due to polymorphisms between closely related strains, and fracturing the scaffolds at such positions leads to fragmented assemblies. Collapsing unitigs

can lead to a "mosaic" consensus sequence. If the variation occurs within genes, the consensus may contain frame-shifts and even make it difficult to determine whether a gene exists. Previous attempts at untangling the genomic variation information from assembly data have relied on visualization techniques [29]. While valuable insights have been obtained through such studies, these approaches are manually intensive and not scalable to large metagenomic datasets. In this paper we propose an approach that can preserve polymorphic bubbles within the assembly yet allows long-range scaffolds to be constructed.

4.2 Our approach

We propose that repeats and genomic variation can be distinguished from each other by examining the unitig graph. Repeats appear to "tangle" the unitig graph, thereby masking the global structure of the genome. Genomic variants, on the other hand, lead to localized motifs in the graph. For example, assume several strains of an organism are virtually identical with the exception of a region of variation (*e.g.* a locus of antigenic variation). The graph pattern corresponding to this situation in Figure 4.2(a) appears as a bubble in the unitig graph. We suggest that the global structure of the genome can be best recovered if the ambiguity due to genomic variation is maintained throughout the scaffolding process. Specifically, motifs due to genomic variation do not affect the long-range structure of the common backbone shared by related genomes. Instead of resolving the bubbles, we detect regions of variation and replace each of them with a single graph node, simplifying the graph



(a)

```
189_2034_188                          TTCCGCAAGCGATAGAGGAAATTGTGGCTGCTTGTTGTTATTTCCAC
189_2338_188 REPLACE RANGE (0, 18299) TTCCGCAAGCGATAGAGGAAATTGTGGCTGCTTGTTGTTATTTCCAC
                                      **********************************************

189_2034_188                          ATTATCAAATCAATATGTCCCGCATTGGCTTTGCCGGTGATTCCGCA
189_2338_188 REPLACE RANGE (0, 18299) TTAAGCGCCGGAATATATTCATATTTTTTCCCGCCGGCACCGAGGAA
                                       * * *     ***** * *    **      *****       * *

189_2034_188                          CGCTCGCCAGTGCGTTGTGGTTGCGTGATAAACAGATCGATTGCGGT
189_2338_188 REPLACE RANGE (0, 18299) TGCTCGGCAATCTCTTCCAGC------GTCTCCAGACAATCCGCAGC
                                       ***** ** *   **   *        *   ****      ** * *
189_2034_188                          TTTTGCTGTGGTATGGGCTTTACGGATTAC---GGGATTCCGTGACT
189_2338_188 REPLACE RANGE (0, 18299) CATCACCTGAATATGACCTACGCCTTTTTCTCCGAGCATTTTCAGCG
                                        *  *     ****  **   *   ** *   * *  *      * *
```

(b)

Figure 4.2: **Bambus 2 is able to identify biologically relevant variations in a metagenome.**(a) A variant motif detected on the Sim3 dataset. The motif corresponds to a *ferrochelatase* gene in *Escherichia coli*. There are two alternate versions of the gene within the *E. coli* K12 (2338) and *E. coli* O157:H7 (2034) genomes. (b) A CLUSTAL W [159] alignment of a subset of the FastA output from Bambus 2, with an edit region corresponding to (a).

without obscuring the structure. Through the iterative application of this process, interleaved with standard graph simplification procedures we can obtain scaffolds that capture a large fraction of the common genome structure of closely related organisms. For each variant, we output a main sequence along with alternatives corresponding to the haplotypes in the data. Fasulo *et. al.* [32] have previously presented an approach for detecting and representing variant bubbles during the assembly process, primarily targeting short-range variation that can be found within a single sequencing read. Our approach is more general and can tolerate larger scale variants (our approach detected variants with an average size of 5606.2 ± 8868.26 when scaffolding 75bp reads). Used in concert with the algorithm described by Fasulo *et al.*: our method will detect large-scale polymorphisms in addition to the short-range within-read variants.

Underlying the procedure above is the assumption that the ambiguity in the assembly graph is primarily caused by genomic variants, *i.e.* repeats have been detected and removed from the graph. We will describe two approaches for finding repeats in metagenomic samples. The first approach is based on the observation that repeat nodes appear to "tangle" the graph structure – *i.e.* these nodes look like focal points in the graph, as in Figure 4.3. We detect such repeats using a measure of node centrality similar to the vertex-betweenness centrality measure used in social network analysis [38, 39]. We also propose a variant of coverage-based repeat detection that tracks the change in coverage within graph components instead of using a global coverage statistic. We will show that this localized coverage measure is less sensitive to coverage differences between organisms in the sample.

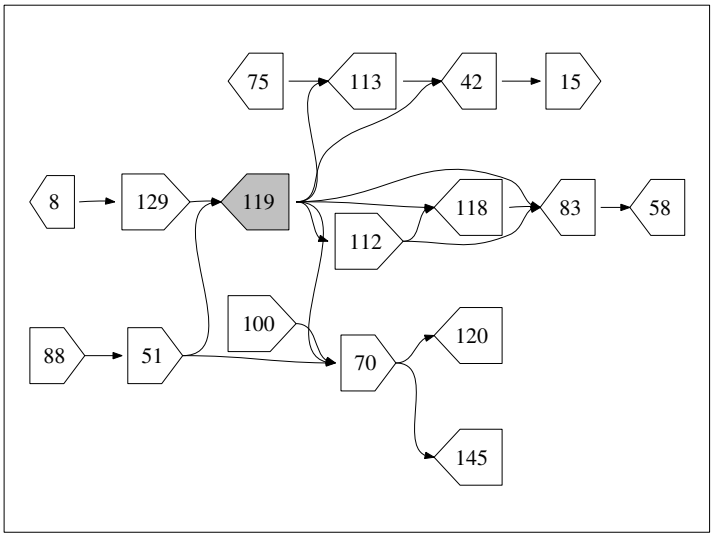


Figure 4.3: **Example of a genomic repeat tangling the contig graph.** The figure shows a subset of a bacterial assembly where nodes are connected if they share paired-end reads. The shaded node, 119, is a repeat that occurs on many shortest paths.

4.3 Methods

Our algorithms operate on a contig graph. A contig may represent a single unitig or an un-gapped concatenation of multiple contigs. For each mate-pair connecting pairs of contigs, we generate a link l with length $d(l)$ and orientation computed from the orientation and positions of the reads in the contigs. The standard deviation $\sigma(l)$ is provided as input to Bambus 2. Using the set of links between pairs of contigs, the orientation is set as the orientation of the majority of the links. Once an orientation is selected, we check whether the distance constraints implied by the links are consistent with each other. If not, we discard the smallest number of links that results in a consistent set S (the largest consistent set can be found in $nlogn$ time using an algorithm for maximal clique finding in an interval graph). Each consistent set is output as an edge e with weight $w(e) = |S|$. The average

length $l(e) = \frac{\sum \frac{d(l)}{\sigma(l)^2}}{\sum \frac{1}{\sigma(l)^2}}$ and standard deviation $\sigma(e) = \frac{1}{\sum \frac{1}{\sigma(l)^2}}$ as suggested in [61]. Additional information, such as overlaps between adjacent contigs (contigs sharing common sequence), is also included when constructing the edges. The resulting graph is bi-directed [101].

Scaffolding consists of three operations: orientation, positioning, and simplification. Throughout the process, we prune the graph by removing contradictory edges and recording their reason for removal.

To avoid the ambiguity introduced by repeats we start with a repeat detection step, then exclude all repeat contigs and incident edges from scaffolding. The (possibly multiple) placement of these nodes can be determined after the initial scaffolding is complete.

4.3.1 Centrality-based repeat detection

We calculate the all-pairs-shortest paths with each edge having weight $w = 1$. For each node, v, we calculate the number of times it appears on a shortest path: P_v. Note that larger contigs are expected to have a higher degree because they contain more reads and, therefore, have a higher chance of being the end-point of a mate-pair link. To correct for this, we linearly scale P_v by the contig length. Such a length-dependent correction has been previously proposed in the context of estimation of gene abundance in metagenomic samples [148]. A node is declared repetitive if the scaled $P_v > \bar{x} + c \times \sigma$ where c is a constant (usually set to 3), $\bar{x}$ is the mean of all scaled P_v $\forall v \in V$, and σ is the standard deviation of all scaled P_v

$\forall v \in V$.

The algorithm above for arbitrary edge weights runs in $O(V^3)$ using the Floyd-Warshall algorithm. It also only tabulates the single shortest path between a pair of nodes, not all shortest paths. As an alternative, we also implement the breadth-first search based algorithm first proposed by Brandes [12] with a complexity of $O(VE)$. This algorithm, while not allowing weights on edges, does count all shortest paths between pairs of nodes. It has also been extended to count not only the shortest path but the k–shortest path as well. The k–shortest path is defined as a path between nodes v and u with length equal to $d(u, v) + k$ where $d(u, v)$ is the length of the shortest path between u and v.

In Algorithm 4.1 we present the cache-coherent lock-free pseudocode for the betweenness centrality algorithm, implemented in OpenMP [20] which requires $O(e^{\sqrt{k}}VE)$ time and $O(VE)$ space. The algorithm relies on two atomic functions, $compare\_and\_swap(variable, originalvalue, newvalue)$ which will update *variable* to *newvalue* if it currently equals *originalvalue*. In either case, it will return the value stored in *variable*. It also requires an $atomic\_add(variable, value)$ which increments the *variable* by *value*. The algorithm is adapted from [95, 65]. The recursive function $W(k, d, w, \sigma)$ is defined below in Algorithm 4.2.

4.3.2 Local coverage statistic

For each connected component S and for each node $v \in S$, we compute the A-stat value [110]. An abundant organism is less likely to appear repetitive in our approach as the connected component is more homogeneous. The calculation

Algorithm 4.1 Compute Vertex Betweenness In Parallel: Part 1, Initialization.

```
B[v] = 0∀v ∈ V
S[v] = 0∀v ∈ V
{Part 1: Initialization}

{#pragma omp parallel for schedule(static)}
for k = 1 → K do
    child[k][e] = ∅∀e ∈ E
for s = 1 → V do
    dist[v] ← ∞∀v ∈ V
    sigmaSums[v] ← 0∀v ∈ V
    {#pragma omp parallel for}
    for k = 1 → K do
        σ[k][v] ← 0∀v ∈ V
        δ[k][v] ← 0∀v ∈ V
    phase ← 0
    σ[0][s] ← 1
    dist[s] ← 0

    {Part II: Graph Traversal}
    count ← 1
    while count > 0 do
        count ← 0
        {#pragma omp parallel for schedule(dynamic)}
        for all v ∈ S[phase] do
            for all w ∈ V such that ∃e(v, w) ∈ E do
                ← compare_and_swap(dist[w], ∞, phase + 1)
                deltaW ← dist[v] − distW + 1
                if distW = ∞ then
                    atomic_add(count, 1)
                    S[phase + 1] ← S[phase + 1] ∪ w
                    distW ← phase + 1
                    deltaW ← dist[v] − distW + 1
                if deltaW < K then
                    child[deltaW][v] ← child[deltaW][v] ∪ w
                if deltaW ≤ min(K − 1, 1) then
                    atomic_add(σ[deltaW][w], σ[0][v])
        phase ← phase + 1
    for k = 1 → K − 1 do
        for i = 1 → phase do
            {#pragma omp parallel for schedule(dynamic)}
            for all v ∈ S[i] do
                for all w ∈ child[0][v] do
                    atomic_add(σ[k][w], σ[k][v])
                if k < (K − 1) then
                    for dj = 1 → k + 1 do
                        for all w ∈ child[dj][v] do
                            atomic_add(σ[k + 1][w], σ[k + 1 − dj][v])
    {#pragma omp parallel for schedule(static) collapse(2)}
    for k = 1 → K do
        for j = 1 → V do
            atomic_add(sigmaSums[k], σ[k][j])

    {Part 3: Back-propagation}
    phase ← phase − 1
    for k = 1 → K do
        {#pragma omp parallel for schedule(dynamic)}
        for p = phase → 0 do
            for all v ∈ S[p][i] do
                for d = 0 → K do
                    for all w ∈ child[d][v] do
                        for di = 0 → (k − d) do
                            sum ← 0
                            e ← k − d − di
                            for dj = 0 → e do
                                sum ← sum + W(e − dj, e, w, σ) ∗ σ[dj][v]
                            δ[k][v] ← δ[k][v] + (sum ∗ (δ[di][w]/(σ[0][w])^{e+1}))
                        δ[k][v] ← δ[k][v] + (σ[k − d][v]/sigmaSums[w])
                B[v] ← B[v] + δ[k][v]
```

Algorithm 4.2 Function W(k, d, w, σ)

```
if k = 0 then
  return σ[0][w]^d
else
  sum ← 0
  for i = 0 → k do
    sum ← sum − σ[i][w] ∗ W(k − i, d − 1, w, σ)
  return sum
```

on each connected component is performed in parallel using OpenMP [20]. This operation is carried out after the repeat nodes identified by all-pairs-shortest paths have been removed.

4.3.3 Orientation

We must first convert the bi-directed graph into a directed graph by choosing an orientation for each node in the graph. We call **reverse edges** any pairwise constraints that require the adjacent contigs to be in opposite orientations. It is impossible to assign a consistent order to nodes involved in a cycle with an odd number of reverse-edges without discarding edges. We attempt to remove a minimum number of edges to allow a consistent orientation to be assigned. Finding such a minimum set is equivalent to the Maximal Bipartite Subgraph problem which is NP-hard [42]. We rely on a greedy heuristic proposed by Kececioglu *et al.* [67] that achieves a 2-factor approximation. The algorithm runs in $O(V + E)$ time.

4.3.4 Positioning

In addition to assigning an edge direction, we want to assign a position for each contig. There may be multiple edges assigning contradictory positions to a node. This imperfect data is the result of experimental errors and repeats (ambiguities in the placement of reads along a genome). We want to maximize the number of satisfied edges by placing nodes as close to the specified position as possible. This

problem is similar to the Optimal Linear Arrangement problem which is also NP-hard [42]. We rely on the following greedy extension heuristic to linearly order the contigs: Scaffolding starts by placing an arbitrary node at position 0. For each node without a position, compute an initial position based on all already-placed neighbors as a weighted average. Subsequent edges can reposition the node within a limit of $3\sigma(e)$ where $\sigma(e)$ is the standard deviation of the edge. The extension stops when the ratio of an edge weight $w(e(u,v))$ to the maximum weight edge incident on node u or v is below a threshold. Edges eliminated from the graph due to invalid orientation are not used in this step. The algorithm runs in $O(V+E)$ time. This heuristic is sufficient once the graph is simplified as above and repeat contigs removed.

4.3.5 Simplification

A transitive reduction is applied to the contig graph and redundant edges are removed. Transitive edges (an edge $e(u,v)$ such that there is a path p with a set of edges $p_e \subset E$ incident on nodes $p_v \subset V$ between u and v not including $e(u,v)$) are removed from acyclical components of the graph by performing a depth-first search from each node in topological order. Given the sequence lengths of contig in the graph $l(v)\ \forall v \in V$ and a path p, we define the length of the path as $l(p) = \sum_{\forall \text{ contigs } v \in p_v} l(v) + \sum_{\forall \text{ edges } e \in p_e} l(e)$. Define the standard deviation of the path as $\sigma(p) = \sum_{\forall \text{ edges } e \in p_e} \sigma(e)$. A transitive edge is removed when $|l(e) - l(p)| \leq \sigma(e) + \sigma(p)$. These edges can be removed without loss of information. Simple paths (all nodes have in- and out-degree equal to 1) are then collapsed: the nodes on the path are replaced with a single node representing the concatenation of the original nodes, and the intervening edges are removed from the graph. Finally, each simplified connected component in the graph gets reported as a scaffold.

4.3.6 Variant detection

Once we have oriented and positioned the contigs and simplified the graph, we iteratively search for variation motifs. We search for subgraphs where multiple paths begin at a source node and collapse to one sink node within a certain number of hops. To allow for artifacts due to incomplete coverage, we allow subgraphs where paths terminate before reaching the sink.

Given graph $G = (V, E)$ and motif set $S \subset V$

$$\text{incoming edges} = S_{in}(u, v) \subset E \text{ s.t. } u \in V - S \text{ and } v \in S$$

$$\text{outgoing edges} = S_{out}(x, w) \subset E \text{ s.t. } x \in S \text{ and } \in V - S$$

$$\forall_{e \in S_{in}(u,v)}, v = source, \forall_{e \in S_{out}(x,w)}, x = sink$$

That is, the incoming edges may only be incident on the source node and the outgoing edges may only be incident on the sink node. The pseudocode to identify 1-deep motifs is given in Algorithm 4.3.

Finally, to avoid false positives due to layouts that satisfy edge constraints but where nodes can be placed in a linear, non-overlapping order, we calculate the overlap ratio.

Algorithm 4.3 Identify and Simplify Motifs in Contig Graph

Given a directed, multi-graph $G(V, E)$
Each vertex $v \in V$ has a position and orientation.
Each edge $e(u, v) \in E$ has an orientation and a distance.
Set M initialized to empty.
while a motif is found **do**
  for all $v \in V$ **do**
    $sink \leftarrow$ uninitialized
    for all $u \in V$ such that $\exists e(v, u) \in E$ **do**
      $neighbor \leftarrow$ uninitialized
      $numNeighbors \leftarrow 0$
      for all $w \in V$ such that $\exists e(u, w) \in E$ **do**
        if $w \neq$v **then**
          $numNeighbors = numNeighbors + 1$
          $neighbor \leftarrow w$
      if ($numNeighbors = 1$ AND ($sink = neighbor$ OR $sink =$ uninitialized)) **then**
        $sink \leftarrow neighbor$
        $M.add(v)$, $M.add(u)$, $M.add(sink)$
    for all $u \in V$ such that $\exists e(v, u) \in E$ **do**
      if $u \notin M$ **then**
        $M.clear()$
    for all $u \in V$ such that $\exists e(u, sink) \in E$ **do**
      if $u \notin M$ **then**
        $M.clear()$
    for all $m \in$M such that $m \neq v$ AND $m \neq sink$ **do**
      for all $u \in V$ such that $\exists e(m, u) \in E$ or $\exists e(u, m) \in E$ **do**
        if $u \notin M$ **then**
          $M.clear()$

Given $S \subset V$, node $v \in S$, start coordinate of v, $B(v)$, and end coordinate of v, $E(v)$

$$length(S) = abs(E(sink) - B(source))$$

$$overlap(S) = \sum_{\forall_{(u,v)\in S}} (min(E(u), E(v)) - max(B(u), B(v)) + 1)$$

$$\text{s.t. } min(E(u), E(v)) - max(B(u), B(v)) + 1) > 0$$

The overlap ratio is then $\frac{overlap(S)}{length(S)}$. Intuitively, it is the total number of bases covered by two or more nodes, divided by the total number of bases in the motif. Motifs whose overlap ratio exceeds a threshold are marked as a polymorphism. To make the problem tractable, only subgraphs with a diameter of 2 are detected in the current implementation of our algorithm. Each iteration of motif detection has a runtime of $O(|V| \times (\Delta(G)^3 + 3\Delta(G)))$ where $\Delta(G)$ is the maximum degree of G. This algorithm has a worst-case runtime of $O(|V| \times (|E|^3 + 3|E|))$. However, in a contig graph it is likely that $\Delta(G) << |E|$. Every level of depth multiplies the runtime by a factor of $\Delta(G)$.

4.3.7 Output

Bambus 2 supports several output formats. Since we do not linearize scaffolds and maintain ambiguity due to variation in the graph, the native output is a graph (in Graphviz format [40]). Bambus 2 also finds the longest sequence reconstruction through each scaffold. That is, it will ignore variant motifs and generate a single self-consistent sequence for each scaffold. Additionally, Bambus 2 outputs each variation motif as a set of sequences. For each motif, S, we start from the *source* node, as defined above. For each child node c of *source*, we recursively compute the sequences starting at c. The longest sequence starting at *source* is the master sequence of the motif. The alternate sequences found in the graph are also output,

including edit positions specifying where within the master sequence they belong. Figure 4.2(b) shows an example alignment of the FastA output for a variant region within *Escherichia coli*.

4.3.8 Test data

We tested the algorithm using the following datasets: *Brucella suis* 1330 comprised of 36 080 reads and available as NCBI Trace Archive Project ID 320. The reference includes: AE014291:AE014292 (2 107 792bp, 1 207 381bp). Three simulated datasets were generated using MetaSim [133] (Table 4.1). The Acid Mine drainage dataset was generated by [165, 151], consists of 179 770 reads and is available as NCBI Trace Archive Project ID 13696. The reference AMD dataset includes: *Ferroplasma acidarmanus* Type I, *Ferroplasma sp.* Type II, *Leptospirillum sp.* Group II 5-way CG, *Leptospirillum sp.* Group III, and *Thermoplasmatales archaeon* Gpl and is available as CH003520:CH004435. The Twin Gut data was generated by [163] and is available as SRA002775 (8.30M GS FLX fragments). The MetaHIT datasets were generated by the MetaHIT consortium [129] and are available as ERS006526, ERS006594, and ERS006494. The GAGE dataset is used in [137] and available online. The single cell genomes are published in [17] and available online.

4.4 Results

In the following section, we demonstrate the performance of Bambus 2 by comparing it with two assemblers used in recent metagenomic projects (Celera Assembler [110], and SOAPdenovo [89]). We also compare Bambus 2 to seven assemblers targeting clonal genomes and one assembler designed for single-cell sequencing data. We have not included a comparison with our previous scaffolder, Bambus [126], as it lacks the functionality necessary in a metagenomic setting. Also, we have omitted

Table 4.1: **Four reference genomes used to generate three simulated metagenomic datasets.** Organism: The reference used to generate simulated data. Reference Size: The size (in bp) of the reference. Identifier: The identifier of the reference in the NCBI Entrez database. # Reads: Total number of reads simulated from the reference for a simulated dataset. The effective coverage for each reference is listed in each dataset.

Organism	Reference Size	Identifier
Psychromonas sp CNPT3	3 052 410	AAPG00000000
Porphyromonas gingivalis W83	2 343 476	AE015924
Escherichia coli K-12 MG1655	4 639 675	U00096
Escherichia coli O157:H7 EDL933	5 528 445	AE005174

Dataset	# Reads	Simulated paired-end size	*P. sp* CNPT3	*P. gingivalis* W83	*E. coli* K-12	*E. coli* O157:H7
Sim1	10,000	50% 5Kbp, 50% 10Kbp	1.97X	2.00X	2.01X	0.00X
Sim2	10,000	50% 5Kbp, 50% 10Kbp	5.30X	0.55X	0.56X	0.00X
Sim3	10,000	50% 5Kbp, 50% 10Kbp	0.55X	0.57X	1.68X	1.65X

comparisons to Genovo [84] and Meta-IDBA [117] as neither of these use mate-pair information during the assembly process.

4.4.1 Repeat detection

We benchmarked our algorithms for repeat detection using artificial and real datasets by comparing repeats identified by Bambus 2 with those identified by the Celera Assembler [105] with metagenomic settings [168, 136] (referred to as CA-met). The CA-met settings increase the tolerance for mismatches when building unitigs, providing longer-range contiguity but possibly leading to mis-assembly. The repeat detection from Celera Assembler relies on coverage, a common approach, and procedures for tuning this assembler for both isolate and metagenomic assemblies have been documented (http://wgs-assembler.sf.net). Figure 4.4 shows the results.

Ideally, the repeat detection should have both high sensitivity and specificity. Sensitivity reflects how many true repeats are detected. Detecting too few repeats can lead to assembly errors in scaffolding. Specificity reflects the false-positives. Detecting too many repeats leads to a sub-optimal assembly as these contigs do not fully participate in scaffolding. In the case of *Brucella suis* 1330, both methods have

high sensitivity and specificity. Celera Assembler repeat detection was designed for clonal organisms. Since the *B. suis* dataset is clonal, CA can accurately detect repeats. In all other cases, Bambus 2 has a higher sensitivity and specificity than Celera Assembler. The default genome size estimates in CA are too sensitive, identifying too many repeats. While varying the genome size improves repeat detection, it is at the expense of sensitivity or specificity. On all datasets, this tuning, which is difficult when the true taxonomic distribution is unknown, still does not match Bambus 2's automated sensitivity and specificity result.

4.4.2 Parallel repeat detection

We evaluate the performance of the OpenMP [20] algorithm and compare it to the Floyd-Warshall $O(V^3)$ implementation. Figure 4.5(a) shows the absolute time taken for repeat detection. Switching to the $O(VE)$ algorithm improves runtime from 137 185s to 3 835.8s, a 35.8–fold improvement in the best case. The minimum gain is a 1.7–fold improvement. The parallel algorithm also shows good scaling as the number of processors is increased (Fig 4.5(b)). The speedup is 7.55 (94.38% efficiency), 6.01 (75.13%), and 7.39 (92.38%) for eight threads and 14.12 (88.81%), 9.1 (56.88%), and 16 (100%) for sixteen threads on UC8, VC1, and MH12, respectively. The best-case efficiency corresponds to MH12, by far the largest graph in the test set with 378 128 nodes and 38 475 edges. The wost-case efficiency corresponds to VC1, the smallest test set with only 90 563 nodes and 8 754 edges. As our algorithm's parallelism is limited by the average out-degree of a node in the graph, the smaller graph is likely reaching saturation by 4 threads while the larger graph continues to benefit. Future work remains to automatically scale the number of threads based on the graph size. The parallel algorithm provides significant gains in speed while providing good speedup efficiency through the use of lock-free operations.

As mentioned above, the Floyd-Warshall algorithm only computes a single

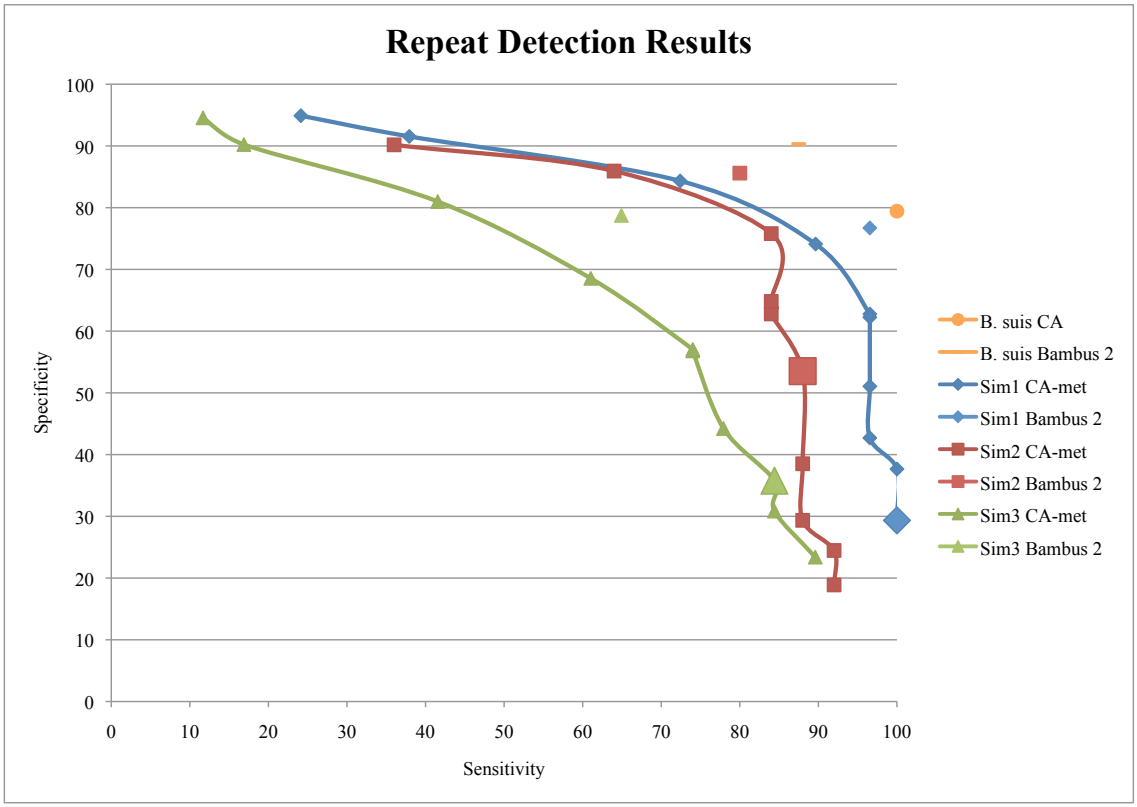
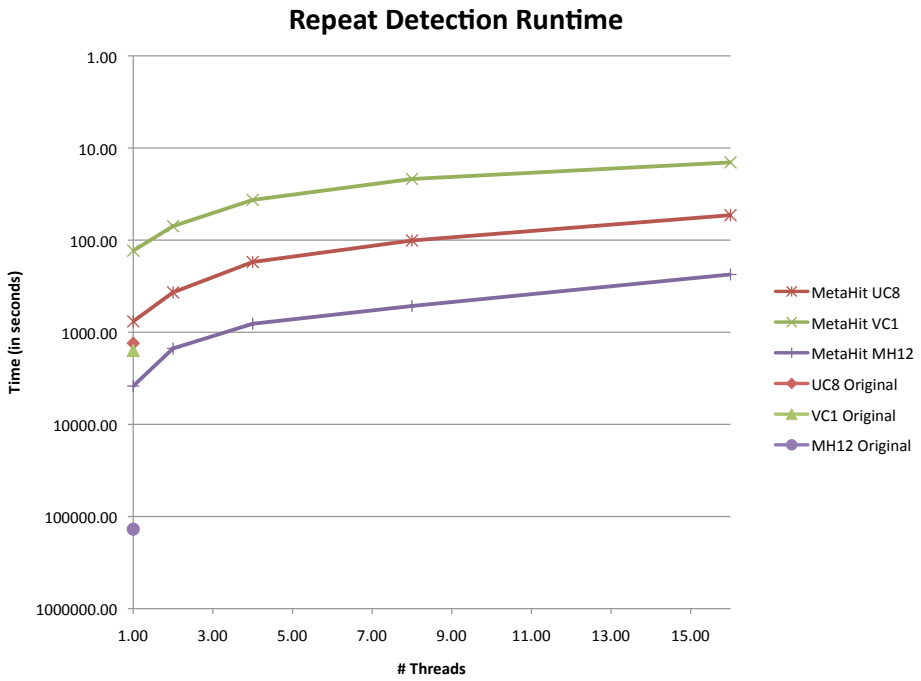
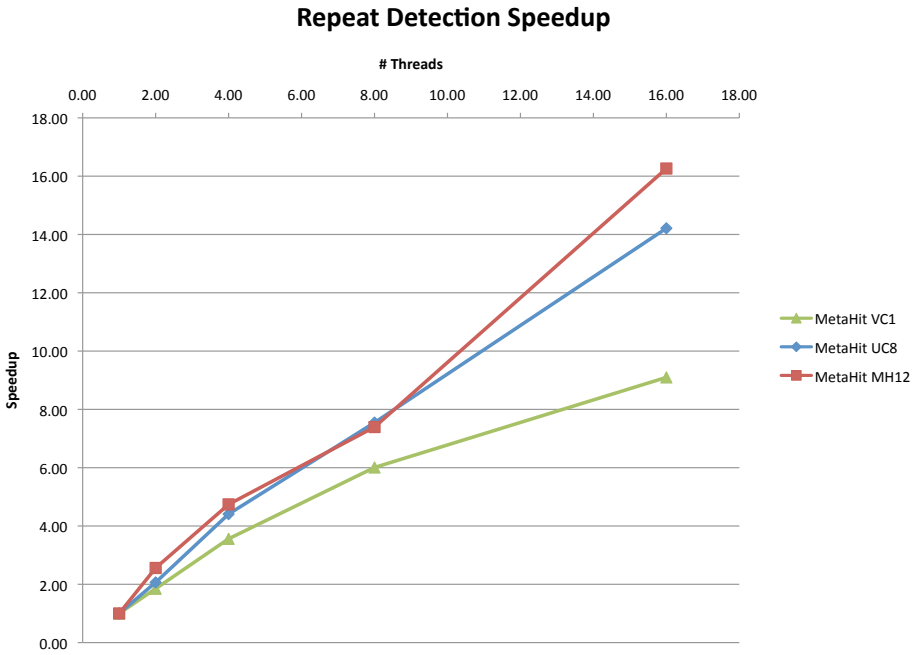


Figure 4.4: **Bambus 2 shows higher sensitivity and specificity when identifying repeats.** Ideal repeat detection corresponds to the top-right corner of the graph, with 100% sensitivity and specificity. We vary the genome size estimate (a critical parameter in the procedure for detecting repeats) for CA, generating a curve for each dataset. The CA-met default is indicated by large shaded points. The Bambus 2 repeat detection is fully automated, generating a single point. As CA is designed for clonal organisms, only the default genome size estimate is used for *B. suis*. The gold standard is built from REPuter. All tests are run using the set of unitigs generated by CA-met. Sensitivity: $\frac{TruePositives}{TruePositives+FalseNegatives}$. Specificity: $\frac{TrueNegatives}{TrueNegatives+FalsePositives}$.



(a)



(b)

Figure 4.5: **Parallel repeat detection decreases runtime as much as** 35–**fold at over** 88% **efficiency.** (a) The repeat performance of the original $O(V^3)$ algorithm as compared to the parallel $O(VE)$ algorithm, in seconds. The parallel algorithm provides as much as a 35–fold improvement in runtime. (b) The efficiency of the parallel algorithm with increasing processors. Perfect efficiency corresponds to a diagonal line. The algorithm demonstrates good efficiency (over 88% at 16 threads) on two of the three datasets. The smallest dataset (VC1) experiences diminishing returns after 4 threads.

Table 4.2: **Repeat detection accuracy on unweighted graphs in parallel matches weighted single-core results.** Datasets and columns are defined in Figure 4.4. Path-only refers to only the betweenness centrality based repeat detection. The Total includes repeats found by both betweenness centrality as well as by local coverage metrics.

	Dataset	Repeat Detection	Sensitivity	Specificity
Path-only	*B. suis*	Weighted (original)	25.00%	98.96%
		Unweighted (original)	25.00%	98.96%
		Unweighted (parallel)	25.00%	96.96%
	Sim1	Weighted (original)	24.13%	98.12%
		Unweighted (original)	24.13%	98.12%
		Unweighted (parallel)	24.13%	98.39%
	Sim2	Weighted (original)	8.00%	99.61%
		Unweighted (original)	12.00	100.00%
		Unweighted (parallel)	12.00%	100.00%
	Sim3	Weighted (original)	9.09%	98.14%
		Unweighted (original)	7.79%	98.09%
		Unweighted (parallel)	10.38%	98.45%
Total	*B. suis*	Weighted (original)	83.33%	92.72%
		Unweighted (original)	83.33%	92.72%
		Unweighted (parallel)	83.33%	92.22%
	Sim1	Weighted (original)	96.55%	76.40%
		Unweighted (original)	96.55%	76.72%
		Unweighted (parallel)	96.55%	76.79%
	Sim2	Weighted (original)	80.00%	85.31%
		Unweighted (original)	80.00%	85.60%
		Unweighted (parallel)	80.00%	85.60%
	Sim3	Weighted (original)	64.93%	78.75%
		Unweighted (original)	64.93%	78.75%
		Unweighted (parallel)	64.93%	79.01%

shortest path between a set of nodes. However, it allows arbitrary weights on the edges. To compare the benefits of weighted graphs we ran the Floyd-Warshall algorithm, both weighted and unweighted and compare it to the Brandes [12] algorithm (Table 4.2). While the weighted graph is sometimes beneficial, the gain from computing all shortest paths (not just a single one) outweighs it and the parallel algorithm is able to generate similar sensitivity and specificity results in a fraction of the time.

Table 4.3: **Scaffolding results for *Brucella suis* 1330 using Celera Assembler and Bambus 2.** Scaffolds and contigs do not include scaffolds comprising a single contig. N50 Scaffold: represents the scaffold such that at least 50% of the genome bases are in scaffolds of that size or greater. Scaffold Span: the total number of bases in all scaffolds, including gaps between contigs.

ASM	# Unitigs	# Scaffolds	Scaffold N50	Scaffold Span
CA	20	3	1 822 827	3 322 447
Bambus2	151	4	1 807 874	3 298 843

4.4.3 Scaffolding of clonal data

While metagenomic assembly violates assumptions commonly made by assemblers targeting clonal populations, a metagenomic assembler without those assumptions is still a superset of the clonal assembly algorithm. Therefore, a metagenomic assembler (or scaffolder) should be able to perform well in the context of clonal assembly.

We assembled the *Brucella suis* 1330 genome and compared the Bambus 2 assembly to the Celera Assembler [110] (Table 4.3). The results show that Bambus 2 produces scaffolds comparable to those generated by Celera Assembler. Bambus 2 determined the correct orientation for all the unitigs, and accurately positioned 145 out of 151 unitigs (96%).

We further evaluated Bambus 2 in the GAGE [137] assembly bake–off. Table 4.4 shows the resulting assembly contig and scaffold contiguity for eight assemblers. Note that Bambus 2 is a close second in scaffold length on *S. aureus*, only being beaten by Allpaths-LG [45]. It is third on *R. sphaeroides*. Bambus 2 is also fifth and second on *S. aureus* and *R. sphaeroides* in contig length, respectively. Table 4.5 shows the errors assemblers make in both contigs and scaffolds. Once again, Bambus 2 performs well, tying for first in both contig and scaffold mis-joins on *S. aureus* and coming in first and third in contig and scaffold mis-joins on *R. sphaeroides*, respectively. These results are especially impressive since Bambus 2 only performs

Table 4.4: **Assembly contiguity on two NGS prokaryotic datasets.** Assemblies of *Staphylococcus aureus* (genome size 2 872 915) and *Rhodobacter sphaeroides* (genome size 4 603 060). For all assemblies, N50 values are based on the same genome size. The Errors column contains the number of mis-joins plus indel errors > 5bp for contigs, and the total number of mis-joins for scaffolds. Corrected N50 values were computed after correcting contigs and scaffolds by breaking them at each error. See the GAGE publication [137] for details on how errors were identified.

Genome	Assembler	Contigs				Scaffolds			
		Num	N50	Errors	N50 corr (kb)	Num	N50	Errors	N50 corr. (kb)
S. aureus	ABySS	301	29.2	14	24.8	246	34	1	28
	Allpaths-LG	60	96.7	16	66.2	12	1 092	0	1 092
	Bambus 2	164	29.5	15	26.0	16	1 089	0	1 089
	CABOG	Could not run: incompatible read lengths in one library.							
	MSR-CA	94	59.2	22	48.2	17	2 412	3	1 022
	SGA	252	4.0	6	4.0	456	208	1	208
	SOAPdenovo	107	288.2	48	62.7	99	332	8	284
	Velvet	162	48.4	28	41.5	45	762	17	162
R. sphaeroides	ABySS	1915	5.9	55	4.2	1 701	9	3	5
	Allpaths-LG	204	42.5	43	34.4	34	3 192	0	3 092
	Bambus 2	376	21.0	25	19.5	92	2 478	2	2 478
	CABOG	322	20.2	34	17.9	130	66	5	55
	MSR-CA	395	22.1	42	19.1	43	2 976	5	2 966
	SGA	3 067	4.5	8	2.9	2 096	51	0	51
	SOAPdenovo	204	131.7	414	14.3	166	660	3	658
	Velvet	583	15.7	35	14.5	178	353	6	270

scaffolding, it depends on an initial assembly. Comparing the starting point for Bambus 2 to Allpaths-LG (the best performer), it builds scaffolds from a corrected N50 that is 4.3 times smaller for *S. aureus* yes produces a nearly-identical scaffold. On *R. sphaeroides*, Bambus 2 scaffolds starting with a corrected N50 that is 1.9 times smaller yet produces the second-largest contigs and third-largest scaffold. Having established that Bambus 2 is a competitive scaffolder on clonal datasets, we next evaluate it both on real and simulated metagenomic datasets.

4.4.4 Scaffolding of simulated metagenomic datasets

We compared Bambus 2 to CA with default settings and CA-met. While other assemblers have been used in metagenomic studies (*e.g.* Phrap (http://www.phrap.org/) and Newbler [97]), as far as we are aware, they have not been extended to target metagenomic data. SOAPdenovo has also been used for metagenomic studies, how-

Table 4.5: **Assembly correctness on two NGS prokaryotic datasets.** Assemblies of *Staphylococcus aureus* (genome size 2 872 915) and *Rhodobacter sphaeroides* (genome size 4 603 060). See the GAGE publication [137] for details on how errors were identified.

Genome	Assembler	SNPs	Indels		Contigs			Scaffolds		
			$\leq$ 5bp	$>$ 5bp	Mis-joins	Inv	Reloc	Mis-joins	Inv	Reloc
S. aureus	ABySS	258	20	9	5	3	2	1	1	0
	Allpaths-LG	79	4	12	4	0	4	0	0	0
	Bambus 2	39	12	11	4	1	3	0	0	0
	MSR-CA	191	23	10	12	6	6	3	3	0
	SGA	32	2	2	4	1	3	0	0	0
	SOAPdenovo	246	25	31	17	1	16	8	1	7
	Velvet	217	6	14	14	5	9	17	5	12
R. sphaeroides	ABySS	692	208	34	21	2	19	3	0	3
	Allpaths-LG	218	150	37	6	0	6	0	0	0
	Bambus 2	193	136	23	2	1	1	2	0	2
	CABOG	536	145	24	10	1	9	5	4	1
	MSR-CA	807	179	32	10	1	9	5	2	3
	SGA	336	116	4	4	0	4	0	0	0
	SOAPdenovo	527	155	406	8	0	8	3	1	2
	Velvet	413	148	27	8	0	8	6	6	7

ever, no scaffolding results were reported [129].

We ran Bambus 2 to scaffold unitigs from CA-met and Minimus [154] (Fig 4.6, Tables 4.6–4.8). While CA-met performs well for certain taxonomic compositions, for example Table 4.7, where one genome is dominant, it shows a greater variation between different compositions. When the assemblies of the genomes are averaged across the simulations, Bambus 2 outperforms CA (Fig 4.6). For all but one genome, Bambus 2 also outperforms CA-met. The only case where CA-met performs better than Bambus 2 is *Escherichia coli* O157:H7 EDL933. The closely related *E. coli* strains are present at sufficient combined coverage for CA-met to obtain large scaffolds. However, the low-abundance genomes in the same sample are not assembled. In scaffolds over $2Kbp$, CA-met only includes 10.90% and 13.31% of the low-abundance genomes, versus 17.24% and 18.37% for Bambus 2 .

Additionally, CA-met constructs a "mosaic" sequence of the two *E. coli* strains, masking variation and potentially introducing error. We evaluated the assemblies of the *E. coli* genomes in detail. There is a large rearrangement of the genome from 100Kbp to 250Kbp (Fig 4.7). In Figure 4.8, CA correctly reconstructs the

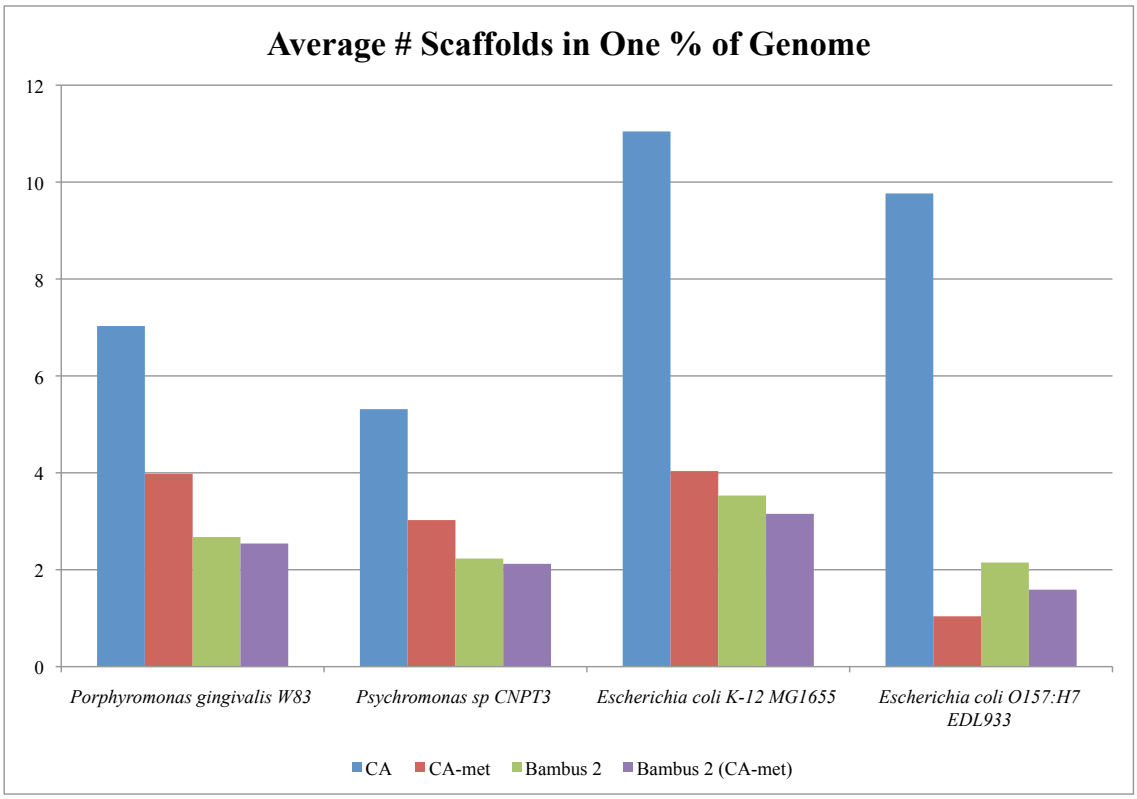


Figure 4.6: **Assembly results for three simulated metagenomic datasets.** The y-axis represents the minimum number of scaffolds that add up to one percent of the genome size. Lower bars represent a better assembly. Bambus 2 produces large scaffolds for a wide range of coverage levels in our simulated datasets. Bambus 2 (CA-met) is Bambus 2 run using CA-met instead of using Minimus unitigs. We aligned the assembly (all contigs $> 2Kbp$) to the reference and counted coverage by reciprocal best matches over 95% identity. We use reciprocal best matches to avoid double counting Bambus 2 motifs that cover the same genomic region. We divide the number of scaffolds by the genome coverage and average the results, by genome, on all three simulated datasets to evaluate performance across varying coverage.

Table 4.6: **Results on simulated metagenomic datasets using Celera Assembler and Bambus 2 on Sim1.** # Scaffolds: The number of scaffolds larger than 2Kbp comprising a given species. % Genome: The percentage coverage provided by the contigs in the scaffolds representing a species.

ASM	Organism	Coverage	# Scaffolds	% Genome
CA	*P. sp* CNPT3	2.00X	233	43.77%
	P. gingivalis W83	1.97X	180	46.26%
	E. coli K-12 MG1655	2.01X	339	43.48%
CA-met	*P. sp* CNPT3	2.00X	48	76.93%
	P. gingivalis W83	1.97X	50	76.45%
	E. coli K-12 MG1655	2.01X	72	77.57%
Bambus 2	*P. sp* CNPT3	2.00X	65	74.71%
	P. gingivalis W83	1.97X	54	72.47%
	E. coli K-12 MG1655	2.01X	116	74.05%
Bambus 2 (CA-met)	*P. sp* CNPT3	2.00X	66	73.96%
	P. gingivalis W83	1.97X	45	72.85%
	E. coli K-12 MG1655	2.01X	110	73.81%

Table 4.7: **Results on simulated metagenomic datasets using Celera Assembler and Bambus 2 on Sim2.** # Scaffolds: The number of scaffolds larger than 2Kbp comprising a given species. % Genome: The percentage coverage provided by the contigs in the scaffolds representing a species.

ASM	Organism	Coverage	# Scaffolds	% Genome
CA	*P. sp* CNPT3	5.30X	3	98.75%
	P. gingivalis W83	0.56X	29	3.64%
	E. coli K-12 MG1655	0.55X	48	2.95%
CA-met	*P. sp* CNPT3	5.30X	1	99.26%
	P. gingivalis W83	0.56X	74	12.59%
	E. coli K-12 MG1655	0.55X	129	12.27%
Bambus 2	*P. sp* CNPT3	5.30X	25	98.47%
	P. gingivalis W83	0.56X	67	19.93%
	E. coli K-12 MG1655	0.55X	130	18.02%
Bambus 2 (CA-met)	*P. sp* CNPT3	5.30X	13	97.93%
	P. gingivalis W83	0.56X	64	19.61%
	E. coli K-12 MG1655	0.55X	130	17.85%

genome but with is a very fractured assembly. In Figure 4.9, CA-met constructs a sequence representative of only one E. coli, with a large inversion at approximately 150Kbp. By contrast, in Figure 4.10, Bambus 2 has the forward reconstruction of

Table 4.8: **Results on simulated metagenomic datasets using Celera Assembler and Bambus 2 on Sim3.** # Scaffolds: The number of scaffolds larger than 2Kbp comprising a given species. % Genome: The percentage coverage provided by the contigs in the scaffolds representing a species.

ASM	Organism	Coverage	# Scaffolds	% Genome
CA	*P. sp* CNPT3	0.55X	18	1.70%
	P. gingivalis W83	0.57X	24	2.60%
	E. coli K-12 MG1655	1.68X	283	31.21%
	E. coli O157:H7 1.65X	362	37.07%	
CA-met	*P. sp* CNPT3	0.55X	92	10.90%
	P. gingivalis W83	0.57X	72	13.31%
	E. coli K-12 MG1655	1.68X	31	46.51%
	E. coli O157:H7	1.65X	54	51.96%
Bambus 2	*P. sp* CNPT3	0.55X	96	17.24%
	P. gingivalis W83	0.57X	72	18.37%
	E. coli K-12 MG1655	1.68X	84	46.33%
	E. coli O157:H7	1.65X	115	d 53.54%
Bambus 2 (CA-met)	*P. sp* CNPT3	0.55X	94	17.60%
	P. gingivalis W83	0.57X	69	18.44%
	E. coli K-12 MG1655	1.68X	58	46.50%
	E. coli O157:H7	1.65X	83	52.26%

both genomes thanks to the detected variant motifs. Finally, in Figure 4.11, we show a subset of the *E. coli* K12 genome (corresponding to Fig 4.2). Here, CA-met has a large insertion in the assembly due to variation within the strains. By contrast, both CA and Bambus 2 have a good reconstruction for the reference. Bambus 2 is able to accurately represent both strains through a combination of scaffolds and variation motifs. As we will show below, on the AcidMine dataset, this "mosaic" assembly leads CA-met makes more mistakes (chimeric scaffolds) than Bambus 2.

We examined all datasets for variation motifs detected by Bambus 2. A total of 16 motifs were found in the Sim1 dataset, and 6 motifs in the Sim2 dataset. Each of the motifs appear to be false positives (all the contigs comprising the motif originate from the same genome). The analysis of the sequence of the overlapping unitigs could be used to detect and correct such mistakes. Such analyses will be

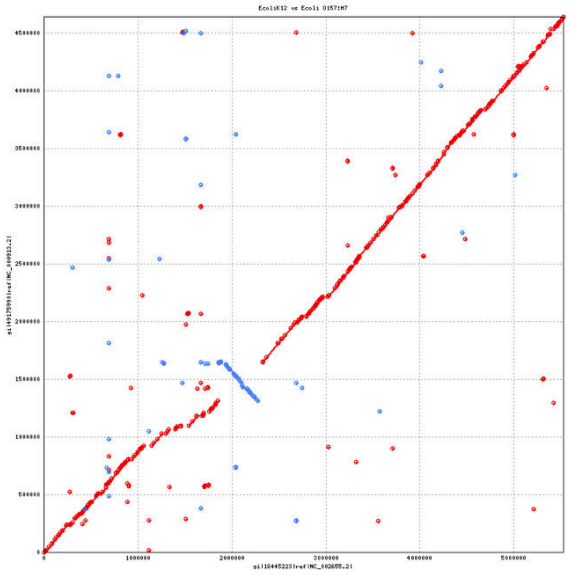


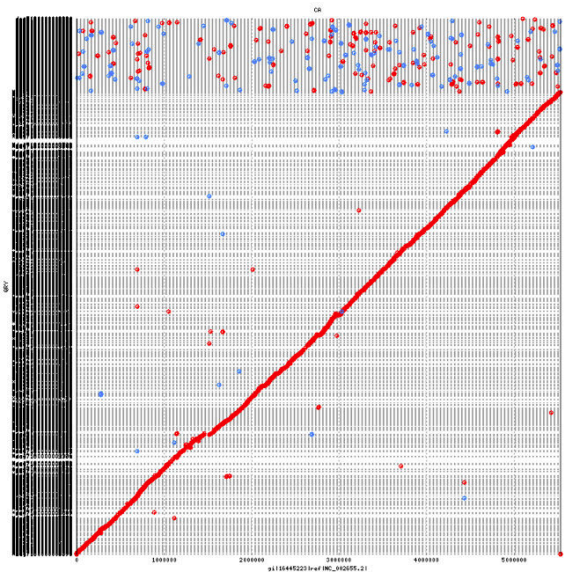
Figure 4.7: **Alignment of *E. coli* K12 (vertical) against *E. coli* O157:H7.**

included in future versions of our software.

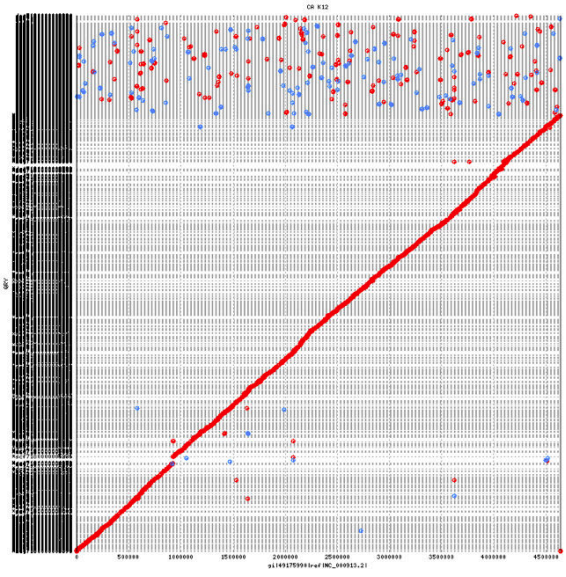
A total of 30 variation motifs were detected in the Sim3 dataset. The motifs detected include genes for *ferrochelatase* (Fig 4.2(a)) as well as *outer membrane proteins*, and *integrase for prophage* which are known to vary across strains of *E. coli* [118] and more broadly, across other enterobacteria.

4.4.5 Scaffolding of the Acid Mine Drainage metagenome

We tested Bambus 2 on an Acid Mine Drainage [165] metagenomic set. These data represent an ideal benchmark as they comprise a low number of organisms, and the genomic variation between related members of the community has been extensively studied. We generated unitigs using CA and scaffolded them with Bambus 2. The Bambus 2 assembly has fewer scaffolds in three of the five organisms present in this sample when compared to CA-met (Fig 4.12, Table 4.9). In two cases, *Leptospirillum sp* Group III and *Ferroplasma acidarmanus* Type I, Bambus 2 more than halves the number of scaffolds as compared to CA-met while reconstructing a larger percentage of the reference genome. In one case, *Ferroplasma sp* Type II, CA-met

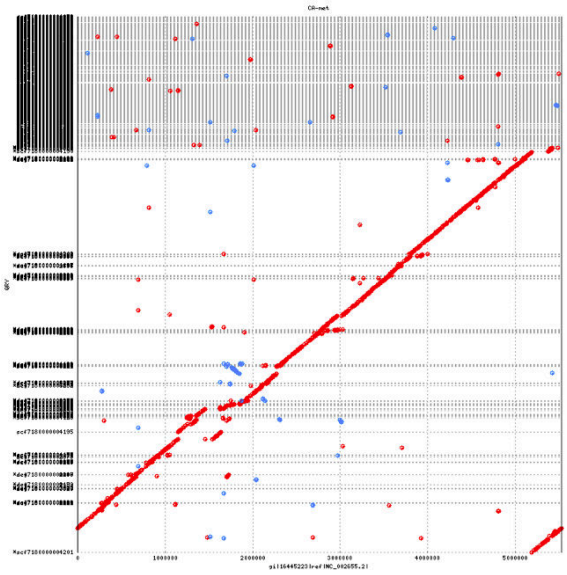


(a)

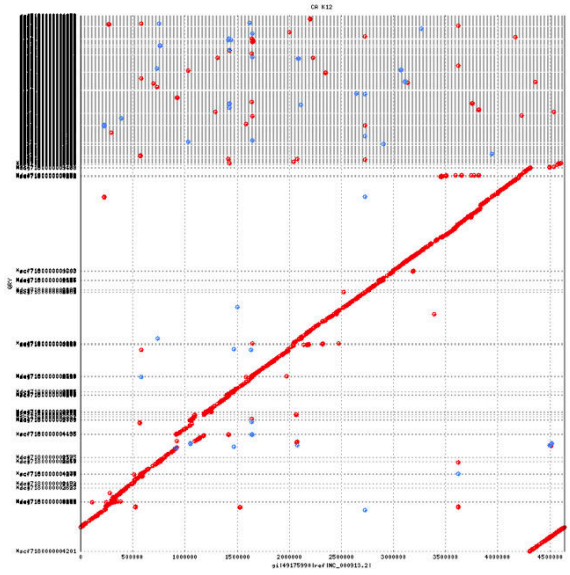


(b)

Figure 4.8: **The CA assembly mapping to *E. coli* O157:H7 (top) and *E. coli* K12 (bottom).** Both genomes are correctly reconstructed (indicated by forward matches in red). However, the matches are very short, indicated by the many scaffold names along the vertical axis in the plots.

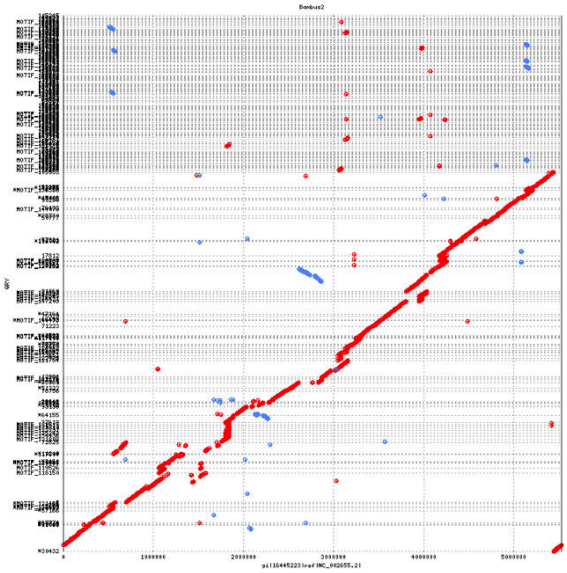


(a)

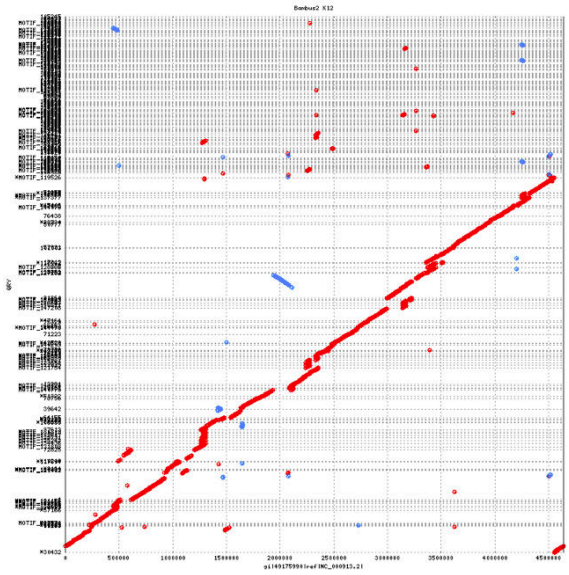


(b)

Figure 4.9: **The CA-met assembly mapping to *E. coli* O157:H7 (top) and *E. coli* K12 (bottom).** The matches are significantly longer than Figure 4.8. However, there is no correct reconstruction of *E. coli* O157:H7 (top). There is a large inversion in one scaffold corresponding to *E. coli* K12.

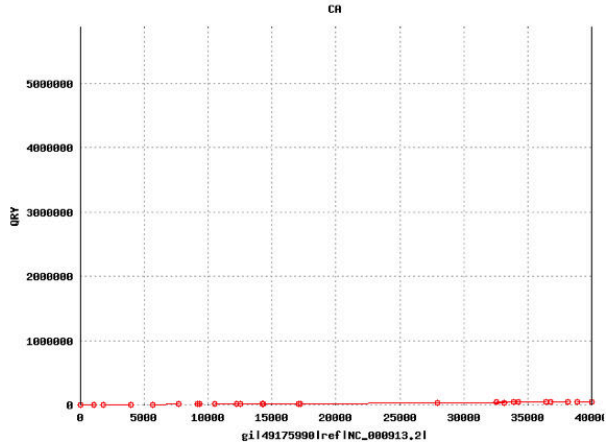


(a)

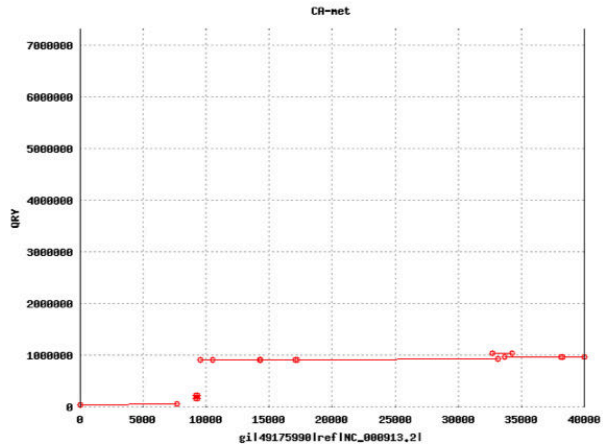


(b)

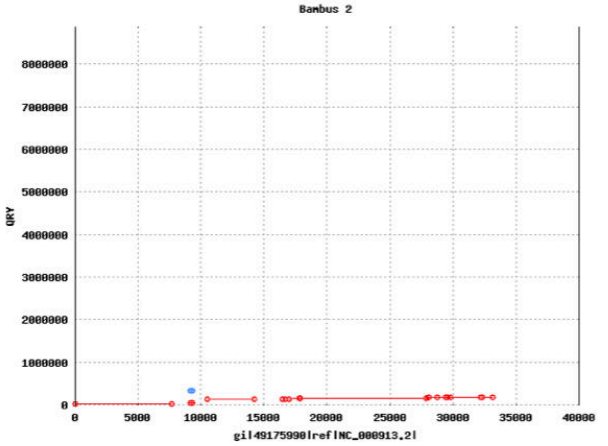
Figure 4.10: **The Bambus 2 assembly mapping to *E. coli* O157:H7 (top) and *E. coli* K12 (bottom).** The matches are longer than those of CA. There are still correct reconstructions of both genomes due to the differences in motif mappings (long names on vertical axis). Note several regions where large groups of motifs are visible in regions where CA-met has a representation of only a single genome version.



(a)



(b)



(c)

Figure 4.11: **A subset of the *E. coli* K12 genome corresponding to Figure 4.2 showing CA, CA-met, and Bambus 2 alignments.** Both CA and Bambus 2 have good alignment to the reference (x-axis) but CA-met has a large insertion in the assembly due to variation within the strains.

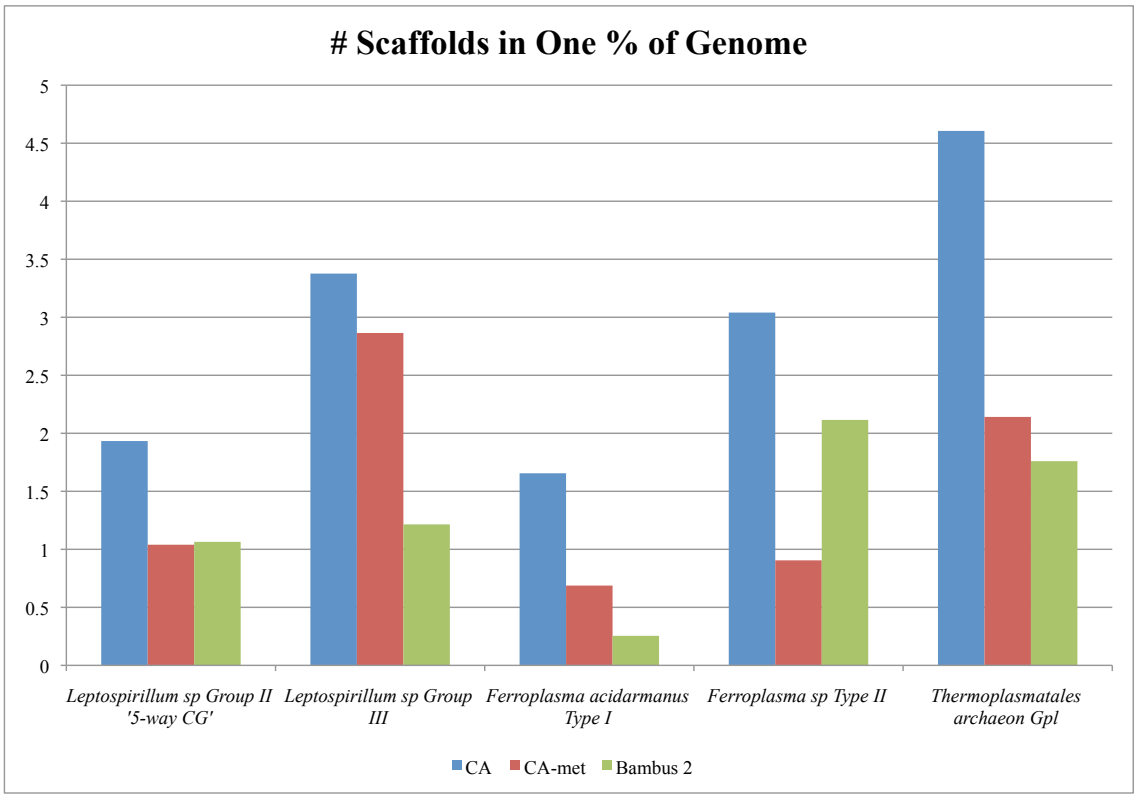


Figure 4.12: **Assembly results for the Acid Mine Drainage metagenome.** The y-axis represents the minimum number of scaffolds that add up to one percent of the genome size. Lower bars represent a better assembly. Bambus 2 produces larger scaffolds that CA-met in three of the five genomes. We calculated assembly statistics as in Figure 4.6. In three genomes, both CA and Bambus 2 produced slightly above 100% coverage. This is due to redundancy within the MUMmer alignments.

produces fewer scaffolds than Bambus 2. However, we found that over 61% of the contigs in the *Ferroplasma sp* Type II CA-met assembly cannot be uniquely assigned to a single reference genome. We hypothesize that CA-met combined the assemblies of *Ferroplasma acidarmanus* Type I and *Ferroplasma sp* Type II, creating chimeric contigs and scaffolds.

We validated our hypothesis by counting the fraction of contigs in chimeric scaffolds. Chimeric scaffolds either include a chimeric contig or contain contigs from different organisms (Section C.2.2). Bambus 2 had the lowest rate of chimeras, 5.66%, while CA-met had the highest at 23.07%. This is expected as CA-met was tuned to maximize scaffold size, possibly merging unrelated organisms. Bambus 2

Table 4.9: **Results on the Acid Mine drainage metagenome datasets using Celera Assembler and Bambus 2.** # Scaffolds: The number of scaffolds larger than 2Kbp comprising a given species. % Genome: The percentage coverage provided by the contigs in the scaffolds representing a species.

Acid Mine	Organism	# Scaffolds	% Genome Covered
CA	*Leptospirillum sp* Group II 5-way CG	198	102.42%
	Leptospirillum sp Group III	277	82.04%
	Ferroplasma acidarmanus Type I	151	91.21%
	Ferroplasma sp Type II	342	112.48%
	Thermoplasmatales archaeon Gpl	405	87.93%
CA-met	*Leptospirillum sp* Group II 5-way CG	101	97.14%
	Leptospirillum sp Group III	234	81.69%
	Ferroplasma acidarmanus Type I	62	90.15%
	Ferroplasma sp Type II	90	99.46%
	Thermoplasmatales archaeon Gpl	179	83.60%
Bambus 2	*Leptospirillum sp* Group II 5-way CG	109	102.40%
	Leptospirillum sp Group III	103	84.78%
	Ferroplasma acidarmanus Type I	26	102.13%
	Ferroplasma sp Type II	237	112.04%
	Thermoplasmatales archaeon Gpl	167	94.90%

built large scaffolds while making fewer mistakes.

The acid mine community used in our analysis is dominated by two genera: *Leptospirillum* bacteria, and *Ferroplasma* archaea. A large extent of genomic variation, primarily due to recombination, was characterized in both these groups of organisms [165, 151, 30]. Initial studies of this environment indicated that most genomic variation can be found in *Ferroplasma sp* Type II, with no predominant functional groups being associated with the variable regions [165]. Subsequent publications with additional sequencing (included in our test), also showed significant variation in *Leptospirillum sp* Group II '5-way CG' [151]. Here we evaluated whether Bambus 2 is able to rediscover these results. We detected a total of 99 motifs, of which 66 represented alternate sequences (two contigs occupying the same positions) and 33 represent insertion/deletion of sequence. The majority of motifs could be assigned to regions from the *Ferroplasma sp* Type II, as expected. However, as a

percentage of bases contained within variation motifs (the extent, rather than number of motifs), the most varied organisms appear to be *Leptospirillum sp* Group II '5-way CG' and *Leptospirillum sp* Group III, followed by *Ferroplasma sp* Type II. The difference in the patterns of variation (frequent but small in *Ferroplasma* and less frequent but large in *Leptospirillum*) was also observed by Simmons *et al.* [151] and could be explained by different biological mechanisms that drive the genomic variability. It was hypothesized [30] that recombination frequently occurs within *Ferroplasma* possibly due to the fact that that these organisms (as well as many other archaea) lack the *mutS* and *mutL* DNA repair systems. Conjugation or transduction, which produce large events (as they are dependent on the F-plasmid and phage size) were hypothesized to contribute to the genomic variation in *Leptospirillum* [151].

We compare the genes within variation motifs to those identified in previous publications. We annotated the assembly by taking non-overlapping best BLASTX [3] hits for each unitig and assigned a COG [158] functional category to each hit. We tabulated the counts of each COG category within the assembly and within the motifs. We then characterized the functional categories that are statistically enriched in motif regions (Section C.2.4, Table 4.10). The functional category corresponding to "DNA replication, recombination and repair" (category L) is significantly enriched (p=0.006, hypergeometric test). Also enriched, (p=0.25, hypergeometric test) is one of the poorly characterized COG categories, "general function prediction" (category R). Our results are consistent with previous analysis of the data [151]. One specific motif identified within *Leptospirillum sp* Group II '5-way CG', corresponds to *glycosyltransferase*, a gene previously characterized as occurring within a mobile region of *Leptospirillum sp* Group II and *Leptospirillum sp* Group III [49]. Thus it is expected that this mobile element would mutate and recombine independently within the members of the *Leptospirillum sp* population, giving rise to the motif.

Table 4.10: **COG categories in the Acid Mine Drainage dataset.**The table shows abundance of COG categories in the Acid Mine dataset. % In Asm: The percentage of each type of COG in the entire assembled dataset. % In Motifs: The number of each type of COG within the motifs detected by Bambus 2. RD (Ratio Difference): % of Motif - % of Assembly. ES (Enriched Significance): The hypergeometric P-values after Bonferroni correction for enrichment. DS (Depleted Significance): The hypergeometric P-values after Bonferroni correction for depletion. See Section C.2.4 for details on how the values are computed.

COG	% In ASM	% In Motifs	RD	ES	DS
A	0.0677%	0.0000%	-0.0677	1	1
B	0.3385%	0.0000%	-0.3385	1	1
C	6.7028%	6.1069%	-0.5959	1	1
D	1.2864%	1.1450%	-0.1414	1	1
E	8.5308%	6.8702%	-1.6606	1	1
F	3.5884%	2.6718%	-0.9166	1	1
G	4.3331%	2.6718%	-1.6614	1	1
H	5.8903%	6.1069%	0.2166	1	1
I	2.3697%	1.5267%	-0.8430	1	1
J	9.4110%	7.2519%	-2.1591	1	1
K	6.1611%	4.9618%	-1.1993	1	1
L	7.5829%	13.3588%	5.7758	0.006	1
M	3.7238%	3.8168%	0.0930	1	1
N	1.0833%	0.7634%	-0.3199	1	1
O	3.3852%	4.1985%	0.8132	1	1
P	3.7238%	3.4351%	-0.2886	1	1
Q	1.4218%	1.9084%	0.4866	1	1
R	12.7962%	17.5573%	4.7610	0.25	1
S	11.0359%	9.1603%	-1.8756	1	1
T	2.6405%	1.5267%	-1.1138	1	1
U	2.3697%	2.2901%	-0.0796	1	1
V	1.4218%	2.2901%	0.8683	1	1
W	0.0000%	0.0000%	0.0000	1	1
Y	0.0000%	0.0000%	0.0000	1	1
Z	0.1354%	0.3817%	0.2463	1	1

4.4.6 Scaffolding the output of NGS assemblers

Finally, we tested Bambus 2 on four dataset comprised of next-generation sequencing reads. The first dataset, comprising the gut microbiome of twins [163], was assembled using Newbler [97] followed by Bambus 2. Our assembly combined all 18

Table 4.11: **Contiguity results on four NGS datasets.** #Scaffolds: The number of scaffolds $> 2Kbp$. Mean: The average length of scaffolds. Max: The maximum length scaffold in the assembly. Scaffold at 5Mbp: We sort the scaffolds in decreasing order by length and count the number and size of the smallest scaffold required to reach 5Mbp. The #ORFs measures the number of ORFs identified by MetaGeneMark [93] in the assembly. The counts are normalized by total sequence length in the assembly. The # Scf in 1% is reported as in Figures 4.6 and 4.12 using a reference identified by BLAST [3], lower scores are better. The errors are reported by dnadiff from the MUMmer 3.20 package [82]. The GUT dataset did not include paired-end information and we relied on the Newbler contig graph to perform scaffolding with Bambus 2. Therefore, the Newbler results are reported on contigs not scaffolds as no scaffolds were generated by Newbler on this dataset.

Dataset	ASM	# Scfs	Mean	Max	# Scf	Len at 5Mbp	# ORFs	# Scf in 1%	# Errors
GUT	Newbler	11 012	4 115.1	46 150	275	12 769	0.00217	19.18	7
	Bambus2	11 450	4 778.9	80 512	134	25 370	0.00204	15.24	10
V1.CD-2	SOAPdenovo	5 794	4 889.0	84 000	230	14 207	0.00186	15.54	51
	Bambus2	4 057	5 680.6	237 167	166	18 210	0.00200	7.28	67
V1.UC-8	SOAPdenovo	15 029	5 371.3	176 511	87	39 282	0.00178	1.34	13
	Bambus2	12 952	5 954.0	257 939	58	55 905	0.00183	0.80	18
MH0012	SOAPdenovo	27 451	6 470.1	356 312	30	115 466	0.00172	2.67	33
	Bambus2	23 994	5 704.7	823 131	35	84 700	0.00180	0.99	43

individual samples from the original study. The assembly generated 3 230 variation motifs. Since we lacked a reference, we could not map our assembly and tabulate statistics as with previous datasets. Instead, we evaluated the assembly contiguity. We sorted the scaffolds in decreasing order by size and counted the number and size of the smallest scaffold required to reach 5Mbp (Table 4.11). To assess scaffold correctness, we used BLAST [3] to identify a dominant organism within the dataset. The best hit was *Bifidobacterium longum* NCC2705 (AE014295). We mapped the assembled scaffolds to the reference using MUMmer [82] and calculated the errors in scaffolds and the number of scaffolds to cover 1% of the reference. We also ran MetaGeneMark [93] to identify ORFs within the assemblies and include the results in Table 4.11. We annotated the assembly and evaluated COG functional category enrichment in motifs as before. The COG functional categories for "amino acid transport and metabolism" (category E), "nucleotide transport and metabolism" (category F), "carbohydrate transport and metabolism" (category G), "DNA repli-

cation, recombination and repair" (category L), and "cell envelope biogenesis, outer membrane" (category M) were enriched, while categories for "cell motility and secretion" (category N) and "unknown function" (category S) were depleted in the variation motifs found by Bambus 2. Interestingly, the enriched functional categories were characterized as "core" for the gut biome (categories universally found across all subjects) in [163]. Other categories classified by Turnbaugh *et al.* as core, such as "transcription" (category K), were also found enriched in our motifs, but not significantly (Table 4.12).

Turnbaugh *et al.* noted that while no core microbiome exists at a taxonomic level, a core can be detected at a functional level. The over-abundance of these core genes in the detected motifs may explain this observation. We hypothesize that the core genes can occur in different genomic contexts due to lateral transfer, allowing a diverse set of organisms to survive within the human distal gut, and thereby explaining an enrichment of such genes within variation hotspots. These results would not be apparent from the analysis of the contig consensus sequences and demonstrate the importance of performing detailed analyses of the data underlying the assembly (*i.e.* the assembly graph) to characterize an environment.

We selected three samples at random from the MetaHIT consortium (V1.CD-2, V1.UC-8, and MH0012) [129]. We re-ran SOAPdenovo to generate unitigs and scaffolds. We used Bambus 2 to scaffold the unitigs produced by SOAPdenovo [89] and compare Bambus 2 scaffolds to those generated by SOAPdenovo (Table 4.11). One dataset (V1.CD-2), comprising over 51M Illumina reads was analyzed in $\approx 3.5hrs$ with a peak RAM usage of $10.0GB$. The largest dataset (MH0012) comprising 186M reads was analyzed in $\approx 20hrs$.

In all cases, Bambus 2 produced more contiguous scaffolds than SOAPdenovo, in two cases more than doubling the largest scaffolds. We again identified a dominant organism within each dataset and map scaffolds to it. The best hits were *Bacteroides*

Table 4.12: **COG categories in Human Gut dataset.** The table shows abundance of COG categories in the Human Gut dataset. % In Asm: The percentage of each type of COG in the entire assembled dataset. % In Motifs: The number of each type of COG within the motifs detected by Bambus 2. RD (Ratio Difference): % of Motif - % of Assembly. ES (Enriched Significance): The hypergeometric P-values after Bonferroni correction for enrichment. DS (Depleted Significance): The hypergeometric P-values after Bonferroni correction for depletion. See Section C.2.4 for details on how the values are computed.

COG	% In ASM	% In Motifs	RD	ES	DS
A	0.2679%	0.0619%	-0.2059	1.0000	0.6600
B	0.2381%	0.0000%	-0.2381	1.0000	0.1400
C	5.9821%	6.0062%	0.0240	1.0000	1.0000
D	1.3988%	1.3622%	-0.0366	1.0000	1.0000
E	6.9048%	8.7926%	1.8878	0.0004	1.0000
F	2.4107%	3.3437%	0.9329	0.0110	1.0000
G	6.0714%	7.8638%	1.7923	0.0004	1.0000
H	4.2857%	4.8916%	0.6059	1.0000	1.0000
I	2.0536%	2.2291%	0.1755	1.0000	1.0000
J	5.2381%	6.3777%	1.1396	0.0580	1.0000
K	4.3750%	5.1393%	0.7643	0.5000	1.0000
L	5.2976%	7.1207%	1.8231	0.0001	1.0000
M	4.7917%	5.9443%	1.1526	0.0350	1.0000
N	1.9345%	1.1765%	-0.7581	1.0000	0.0390
O	3.6310%	3.0960%	-0.5350	1.0000	1.0000
P	4.4940%	4.7678%	0.2738	1.0000	1.0000
Q	1.5476%	0.9907%	-0.5569	1.0000	0.2200
R	14.6726%	13.0031%	-1.6695	1.0000	0.1600
S	17.0536%	10.9598%	-6.0938	1.0000	0.0000
T	3.5417%	3.4056%	-0.1361	1.0000	1.0000
U	2.4405%	1.8576%	-0.5829	1.0000	0.6200
V	1.0714%	1.4861%	0.4146	0.4300	1.0000
W	0.0298%	0.0619%	0.0322	1.0000	1.0000
Y	0.0298%	0.0000%	-0.0298	1.0000	1.0000
Z	0.2381%	0.0619%	-0.1762	1.0000	1.0000

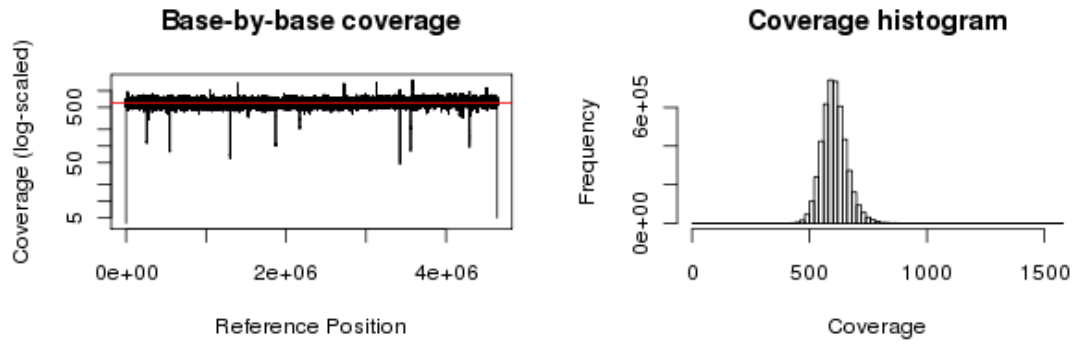
coprophilus DSM 18228 (NZ_ACBW00000000), *Methanobrevibacter smithii* ATCC 35061 (NC_009515), and *Akkermansia muciniphila* ATCC BAA-835 (CP001071.1) for V1.CD-2, V1.UC-8, and MH0012 respectively. Bambus 2 produced more ORFs per MB of assembly. It also required fewer scaffolds to cover the reference while not introducing many errors.

We hypothesize that the improvement in contiguity is due to Bambus 2 overcoming genomic variation within the data, where we identified 2 763 variation motifs. To evaluate our hypothesis we aligned Bambus 2 motifs to SOAPdenovo scaffolds and counted motifs which span multiple scaffolds. Out of the 2 763 motifs in the assemblies, 2 554 mapped to multiple scaffolds, confirming that Bambus 2 motifs correspond to scaffold breaks in SOAPdenovo.

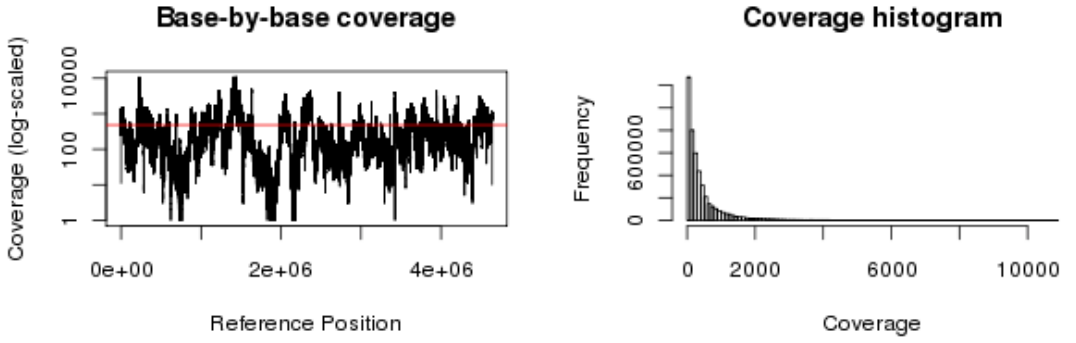
4.4.7 Scaffolding single-cells

Single-cell assembly is another example of an assembly target challenged by widely different levels of coverage for different portions of a genome (Fig 4.13). The Bambus 2 coverage-independent repeat identification can benefit scaffolding in this context. Recently, a novel assembler specifically targeting single-cell sequencing has been developed [17]. The algorithm modifies an existing assembler to tolerate coverage differences. Unfortunately, the published assembler does not generate scaffolds or make use of paired-end data. We assembled two datasets from the Velvet-SC publication [17] and scaffolded the results using Bambus 2. In Table 4.13, we present the results from the publication as well as our results running Velvet-SC alone and in combination with Bambus 2.

The table shows that Bambus 2 is able to generate longer contigs than those from Velvet-SC alone. The assemblies show a 4.4% and 22% corrected N50 gain while introducing one and zero errors in *E. coli* K12 and *S. aureus* USA300, respectively. While the gain is small, the datasets only have short Illumina paired-ends (260bp) and Bambus 2 is still able to improve on the state-of the-art. Based on the assembly of *S. aureus* from Salzberg *et. al.* [137], Bambus 2 contiguity increases 22–fold when long (3Kbp) Illumina mate-pairs are available (from 50Kbp to 1 082Kbp scaffold corrected N50). Therefore, we expect Bambus 2 can significantly improve single-cell assemblies.



(a)



(b)

Figure 4.13: **Single-cells sequenced using MDA have significant bias in coverage.** (a) The coverage of *E. coli* K12 using clonal sequencing. The mean (shown in red) is 607X. The coverage is evenly distributed throughout the genome and the histogram matches the expected Poisson distribution. (b) Coverage of *E. coli* K12 using MDA sequencing. The mean (shown in red) is 482X. There is significant variation in coverage (from 0X to 10 000X) with several regions of the genome at 0X coverage. The histogram has a very long tail, reflecting the biased sequencing of the genome.

Table 4.13: **Assembly contiguity on two single-cell prokaryotic datasets.** Assemblies of *Staphylococcus aureus* (genome size 2 872 915) and *Escherichia coli* (genome size 4 639 675). For all assemblies, N50 values are based on the same genome size. The Errors column contains the number of mis-joins plus indel errors > 5bp for contigs, and the total number of mis-joins for scaffolds. Corrected N50 values were computed after correcting contigs and scaffolds by breaking them at each error. See the GAGE publication [137] for details on how errors were identified.

Genome	Assembler	Contigs				Scaffolds			
		Num	N50	Errors	N50 corr (kb)	Num	N50	Errors	N50 corr. (kb)
E. coli Lane 6	Euler+Velvet-SC [17]	501	32.0	-	-	-	-	-	-
	Velvet-SC	220	56.5	23	52.1	-	-	-	-
	Velvet-SC+Bambus 2	204	60.7	24	54.4	193	65.0	0	59.6
S. aureus Lane 7	Euler+Velvet-SC [17]	355	32.3	-	-	-	-	-	-
	Velvet-SC	175	37.5	19	34.9	-	-	-	-
	Velvet-SC+Bambus 2	141	45.8	19	42.7	136	48.4	4	40.9

4.5 Discussion

The repeat detection procedures used in Bambus 2 are sensitive without sacrificing specificity, and have been applied to the assembly of clonal, metagenomic, and single-cell samples. The scaffolds generated by Bambus 2 cover a large percentage of the genomes in the samples, while largely avoiding mis-joins. The FastA output of variants motifs facilitates analysis of the full diversity in an environment. Furthermore, the ability to highlight regions of variation has proven useful in detecting biologically meaningful patterns that match previously published results.

Bambus 2 is not a stand-alone assembler. Instead, it is a drop-in scaffolding module optimized for non-clonal data and is compatible with the output of many modern assemblers. Thus, Bambus 2 can be applied to virtually all existing sequencing technologies – it is sufficient to start with an assembler that is best suited for that type of data.

4.6 Conclusion

Accurately assembling metagenomic datasets automatically is challenging with current assemblers, and often requires manual tuning of parameters and post-processing. Our work represents a first step towards automated metagenomic assembly, and is able to obtain long-range contiguity in metagenomic datasets while also characterizing regions of variation.

Chapter 5

An Automated End-end Metagenomic Analysis Pipeline[†]

5.1 Introduction

Metagenomics has opened the door to unprecedented comparative and ecological studies of microbial communities, ranging from the sea [136, 182, 180] to the Arctic [167]. Metagenomics has also enables new studies of the human body [4, 44, 75, 164, 163], eukaryotes and their flora [173, 43], and even a health-care facility [71]. Next-generation sequencing (NGS) single genome assemblers are rapidly improving [45], but due to the diverse mix of genomes in a given metagenomic sample, distinguishing polymorphisms from sequencing error [23] is less obvious. Paralogs can be easily mistaken for orthologs, leading to assembly errors [166]. Thus, due to the ubiquity of NGS data and shortage of metagenomic assemblers, analysis is commonly limited to metagenomic read classification. However, short reads offer very limited information to read-based classifiers [10, 11, 102, 99, 18, 79, 116]. As a result, these methods often classify fewer than 50% of reads and fail to outperform BLAST [112]. In addition, given the highly biased phylogenetic distribution of genome databases [179], classification methods relying on databases fail to acknowledge the majority of novel species present in the dataset. This is especially true in the case of viral metagenomes where often 60–99% of a sample is unknown [34, 106]. In an attempt to move away from read-centric analyses, computational tools based on promising algorithmic and statistical methods for metagenomic *de novo* assembly have begun to emerge [84, 94, 117]. However, to date they either

[†]The text of this chapter is based on the publication T. Treangen*, S. Koren*, D. Sommer, B. Liu, I. Astrovskaya, A. Darling, and M. Pop. metAMOS: A modular and open source metagenomic assembly and analysis pipeline. *Genome Biology*, In Prep, 2012.

are ill-suited for large datasets or lack a scaffolding module. In addition, the most recently developed methods have not been exhaustively compared to one another (within an assembly bake-off competition such as GAGE [137]) leaving their respective benefits and drawbacks unclear and hampering their adoption for metagenomic assembly.

De novo assembly is a small part of metagenomic analysis, which can also include gene-based analysis and classification methods. Until now, custom and manually intensive metagenomic analysis has impeded biological progress by making reproducibility difficult and limiting collaboration. The importance of reproducibility in analysis has been discussed in the context of microarray data [63] and general bioinformatics [46]. However, setting up a pipeline to analyze a metagenome still requires familiarity with tool-specific parameters and bioinformatics expertise. To the best of our knowledge there is no comprehensive end-to-end assembly and analysis toolkit for metagenomics. Thus, there is a current need for efficient yet comprehensive assembly and analysis pipelines for NGS data sequenced from metagenomes.

To address the need for metagenomic analysis, we present metAMOS, a modular and customizable approach to assembly and analysis of metagenomic samples. metAMOS can be viewed as an assembly-centric counterpart to QIIME [14] and mothur [145], which were designed to analyze 16S rRNA metagenomic sequences. metAMOS also shares similarities with SmashCommunity [4]. However, SmashCommunity supports only 454 and Sanger read data, does not make use of existing metagenomic assemblers and scaffolders (currently supports just three assemblers: Arachne, Celera Assembler, and Forge), and does not provide validation methods.

Figure 5.1 outlines the workflow of metAMOS. From our point of view, providing a customizable end-to-end analysis pipeline addresses the challenges outlined above. Metagenomic analysis is simply too complex for any individual laboratory to tackle alone, which is why we believe the right approach harnesses the collec-

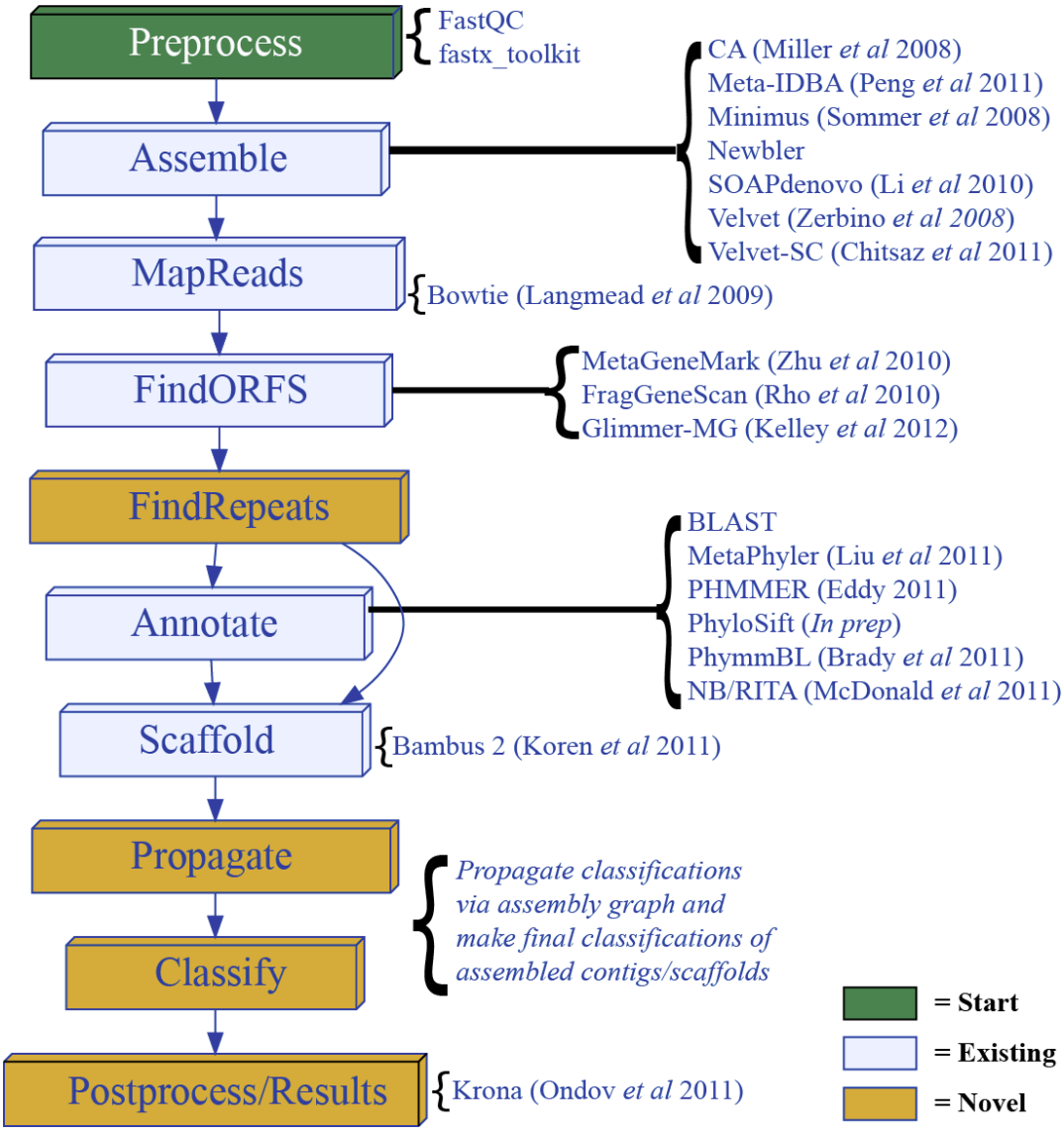


Figure 5.1: **metAMOS pipeline overview.**

tive expertise of the community into a single analysis pipeline. metAMOS is built upon AMOS [127, 154, 123, 142, 162] and follows the philosophy that creating an open-source and modular framework encourages collaboration in the community. This approach allows researchers to focus on their areas of expertise while benefiting from the collective expertise and advances from other research groups that have representation within metAMOS. A secondary but fundamental goal for metAMOS is to enable rapid analysis of hundreds of millions of sequences. Assembly is a fundamental component to achieve this goal as it reduces the data from millions of short sequences into a much smaller collection of unitigs and contigs, removing redundancy that costs computational cycles in downstream analyses. For example, for 690 Human Microbiome (HMP) samples [164], the average compression from sequences to assembled contigs is as high 160-fold and 3–5 fold when unassembled sequences are included in the analysis. Performing time-consuming analyses such as gene-calling and taxonomic classification after assembly allows metAMOS to exploit this reduction. metAMOS packages together over 20 existing metagenomic tools and combines them with novel analysis tools we have developed, enabling the user to analyze hundreds of millions of sequences in *hours* rather than *days* or *weeks*. Included visualization tools and HTML reports provide easy access to results. All steps are discussed in detail in Section 5.2.2.

5.2 Methods

Our design of the metAMOS pipeline was motivated by two guiding principles: modularity and robustness. These were inspired by the desire to allow users to tailor metAMOS to the biological questions they want to answer, not the inverse. Given that each metagenome analysis presents a unique set of challenges and goals, users will take advantage of this modularity and customize their own pipelines by combining the modules they deem necessary on the fly. metAMOS

leverages a previously published workflow management system (Ruffus [50]) to keep track of inputs/outputs/states and checkpoint while running through computationally intensive analyses. While metAMOS offers several novel features specific to metagenomic assembly, we also leverage existing methods and software for metagenomic analysis by creating a collaborative environment for metagenomic assembly to encourage cooperation among the community.

5.2.1 Install

To retrieve the source code, the user can either download it directly from the repository or execute following command from the command-line: `git clone git://github.com/treangen/metAMOS.git`. Upon download of the metAMOS source, the user simply has to have python installed (2.5.x-2.7.x) and then run the *INSTALL* script. This will automatically configure the pipeline to run within the users environment and also will fetch any externally required data if a connection to the internet is available. Once installed, there are two main executables that comprise metAMOS: *initPipeline* and *runPipeline*. *initPipeline* is used to create a project environment for *runPipeline* and is where the user describes their input files (454/Illumina reads, assembled contigs) and library types. *runPipeline* takes a project directory as the input and will initiate the entire metAMOS pipeline (Fig 5.1). Now we will describe each module in the pipeline in detail.

5.2.2 Workflow

5.2.2.1 Preprocess (required)

This is the starting point of all analyses in the pipeline. metAMOS can take a variety of inputs, including interleaved and non-interleaved FastQ/FastA files, SFF files, and even a set of pre-assembled contigs. metAMOS supports existing

read-analysis tools such as FastQC to evaluate the quality of the supplied read data. Preprocess includes an optional custom aggressive read filter that discards any read containing Ns or a base below a pre-defined quality value. The idea behind aggressive read filtration is that read coverage/depth is no longer at a premium and the quality of reads has a huge influence on the quality of assemblies [137]. This initially may seem extreme, especially since this can discard upwards of 25% of the reads; however, given the dependency of de Bruijn graph based assemblers on clean data, we anticipate that our assembly quality will be improved and have indeed observed this in practice. That said, we also include a read filtration step based on the fastx_toolkit that allows the user to trim and keep reads. Another important component of Preprocess is the library verification step that will check whether read pairs are properly aligned and also modify read headers to ensure they are compatible with downstream tools.

5.2.2.2 Assemble (optional)

Once reads are pre-processed they are passed onto the Assemble step for assembly. Currently metAMOS has support for seven assemblers including SOAPdenovo [88], Newbler [97], Velvet [183], Velvet-SC [17], Meta-IDBA [117], CABOG [105] and Minimus [154]. Each of these assemblers has its own (lengthy) set of parameters and required input format, all of which are automatically managed within the pipeline and transparent to the user. It is our goal to keep growing this list to include the plethora of existing assemblers and eventually allow the user to combine assemblies via a meta-assembly strategy (Schatz, Metassembler, In prep), combining the strengths of each assembler and hopefully avoiding the weaknesses of any single strategy. Note if the user specified contigs as input to metAMOS, this step is automatically disabled. Three types of assembly are possible in the current version of metAMOS: (a) single genome/isolate, (b) metagenomic, or (c) single cell. While

the main focus is on metagenomic assembly, thanks to the modular nature of metAMOS, all three types are supported via the mentioned assemblers and command-line options.

5.2.2.3 FindORFS (optional)

Immediately following assembly, we pass the contigs onto the metagenomic gene prediction module. Three metagenomic gene prediction tools that are currently supported include FragGeneScan [132], MetaGeneMark [9], and Glimmer-MG [68]. The rationale for calling genes after assembly is two-fold. Firstly, most metagenomic gene prediction tools have significantly increased sensitivity and accuracy once the fragment is more than 300bp, something that is reasonable to expect once we have assembled the reads. Secondly, mis-assembles can be potentially inferred from this stage given the output of the metagenomic gene prediction tool. For example, if we attempt to assemble contigs that belong to a bacterial genome and the contig spans several kilobases, we should see fairly regular gene predictions given the expected coding density in prokaryote genomes [80]. If we observe a region in a contig that that lacks ORF predictions over a large region, it is either (a) an error of the gene prediction method or (b) a mis-assembled contig that does not allow a gene to be called or (c) a pseudo-gene or eukaryotic DNA. Since we support multiple tools, we can utilize each of them and check if there is a consensus for the region lacking ORFs. If this is the case, we can preemptively break the contig at this region and assume chimerism. This indicator can be combined with other mis-assembly signals [123] to improve specificity. Even though these tools are efficient, we limit the work by only calling ORFS on contigs with more than 3X depth of coverage and larger than 300bp (both parameters are configurable by the user).

5.2.2.4 FindRepeats (optional)

A novel feature in metAMOS, this step takes ORFs (or contigs) as an input. The step serves two purposes. First, it annotates repetitive contigs that can be later used to bypass the MarkRepeat step of Bambus 2, which, depending on the sample, can become computationally expensive. Second, we would like to create multi-alignments of all predicted ORFs to avoid classifying several copies of near identical genes. Repeatoire does exactly this, as it is well-suited for identifying repeat families *de novo*.

5.2.2.5 Annotate (optional)

This step also takes ORFs (or contigs) as an input. Once we have assembled contigs and predicted ORFs, we likely want to determine which organisms comprise a given sample. In order to annotate our ORFs or contigs, we offer five classification methods such as: PhyloSift (Darling, in prep), BLAST [3], FCP, PHMMER [27], and PhymmBL [10, 11], which include homology based methods, phylogenetic (Bayesian) methods, and Iterative Markov Model training methods. Annotating each and every read in a several million or even billion read sample can be computationally prohibitive and lead to inaccurate classifications due to a lack of a discriminatory signal from such short sequences [112]. Thus, our philosophy is to assemble first and then classify contigs, or even better, ORFs. This allows a more focused approach to annotation that permits much more reliable classification on the predicted ORFs compared to individual short reads. However, as not all reads will find their way into the final assembly, we will still have to invest computational resources to classify singleton reads to get a complete picture of the composition.

5.2.2.6 Scaffold (required)

One of the main challenges in metagenomic assembly is the entanglement of repeats and genomic variation. In isolate bacterial genome assembly, any regions that tangle the assembly graph are necessarily repetitive regions in the genome; there exist a variety of strategies for disambiguating and resolving repeats in bacterial genomes [176]. However, in metagenomic assembly, one cannot assume that tangles in the graph are due to repeats. Indeed, it is highly likely that they could also be generated due to variable regions in closely related strains of a bacteria/virus commonly found in the targeted community. Fortunately, computational tools such as Bambus 2 [78] exist that allow for metagenomic scaffolding and can disentangle genomic variation from repeats. One option to the user is to enable a sequence-based repeat detection (the output of FindRepeats), which can offer a performance increase over graph-based repeat detection at the cost of an increased rate of false positives (variant regions instead of repeats). Once repeats have been identified and classified, Bambus 2 can focus on the variant regions in the assembly. For example, assume that several strains of a given bacterium are identical with the exception of a region of variation (*e.g.* a locus of antigenic variation). Since this structure is local in the assembly graph, Bambus 2 is able to distinguish it from a repeat which tangles the assembly graph. Bambus 2 is able to recover the global structure of the genome by maintaining the ambiguity due to genomic variation throughout the scaffolding process; motifs due to genomic variation do not affect the long-range structure of the common backbone shared by related genomes. In addition, Bambus 2 outputs these variant regions and makes them available to the user for downstream analysis.

5.2.2.7 Propagate (optional)

Another novel piece of the metAMOS pipeline is the annotation propagation (via mate-pairs) step. Given a traversable assembly graph that has been simplified

by scaffolding, we query the graph with high-confident annotations. The annotations are transferred to other previously unclassified regions that are supported via mate-pair links. Since high complexity metagenomic communities are expected to result in fragmented assemblies, consisting mainly of relatively small contigs or scaffolds, this step proves useful in classifying the smaller pieces without requiring more contiguous assemblies.

5.2.2.8 Classify (optional)

One of the final steps of the metAMOS pipeline is to assign final classifications to each and every contig/ORF/scaffold in the outputs produced by earlier steps and store them in subdirectories labelled with the pre-specified taxonomic level. In addition, all reads that were used in the assembly of the contigs/scaffolds are placed in each appropriate subdirectory. The benefit of this step is two-fold: (a) organisms of interest are searchable and able to be targeted for comparative assembly and (b) all unclassified and potentially novel organisms are grouped together and can be the starting point for further analysis.

5.2.2.9 Postprocess (required)

The metAMOS pipeline ends by generating an HTML formatted summary (including charts) of the assembly statistics and estimated abundance information in addition to an interactive report generated by Krona [114] (based on the Annotation step) in a single HTML file. The file is self-contained and can be shared with colleagues via email or a web-server. metAMOS also includes novel assembly visualization tools targeted to metagenomics. One viewer, a plugin for Cytoscape [147, 153], can dynamically highlight or hide subsets of the assembly and output a multi-FastA sequence corresponding to user-selected variants. It is described in detail below.

Once finished, if the user prefers to rerun any step (for instance, annotate with a different method), the pipeline can be re-run with the exactly same command that was previously used and, thanks to the Ruffus framework, it will only re-run the necessary steps. This allows for a quick exploratory run to be performed that is later refined once initial information is gathered on the composition and characteristics of the metagenomic community.

5.2.3 Visualization

We introduce a viewer, named ScaffViz, that is a plugin for Cytoscape [147, 153]. Assembler output is read in by ScaffViz using the factory design pattern, abstracting Cytoscape objects from the developer, enabling new assembler support to be easily added. A layout algorithm positions, orients, and sizes the nodes to reconstruct the last view an assembler had before termination. The assembly layout supports user preferences (Figure 5.2c) to narrow the view to only those scaffolds of interest. The viewer can dynamically highlight or hide subsets of the assembly and output a multi-FastA sequence corresponding to user-selected variants. Scaffolds are displayed in decreasing order from the top of the screen to the bottom. For each node, we compute the bounding rectangle based on its position in the scaffold. Intersecting nodes are shifted down (Figure 5.2a), forming a stair structure.

5.2.3.1 Motifs and FastA output

ScaffViz tracks subsets of the assembly that the user can dynamically show (or hide) (Figure 5.2b).

ScaffViz implements the data structures represented by Figure 5.3. The motifs output by Bambus 2 are automatically loaded [78]. ScaffViz also allows users to designate an arbitrary set S, at runtime. A subset of nodes S is replaced by a single node u. Recursion within sets is allowed so a node $s \in S$ may itself correspond to

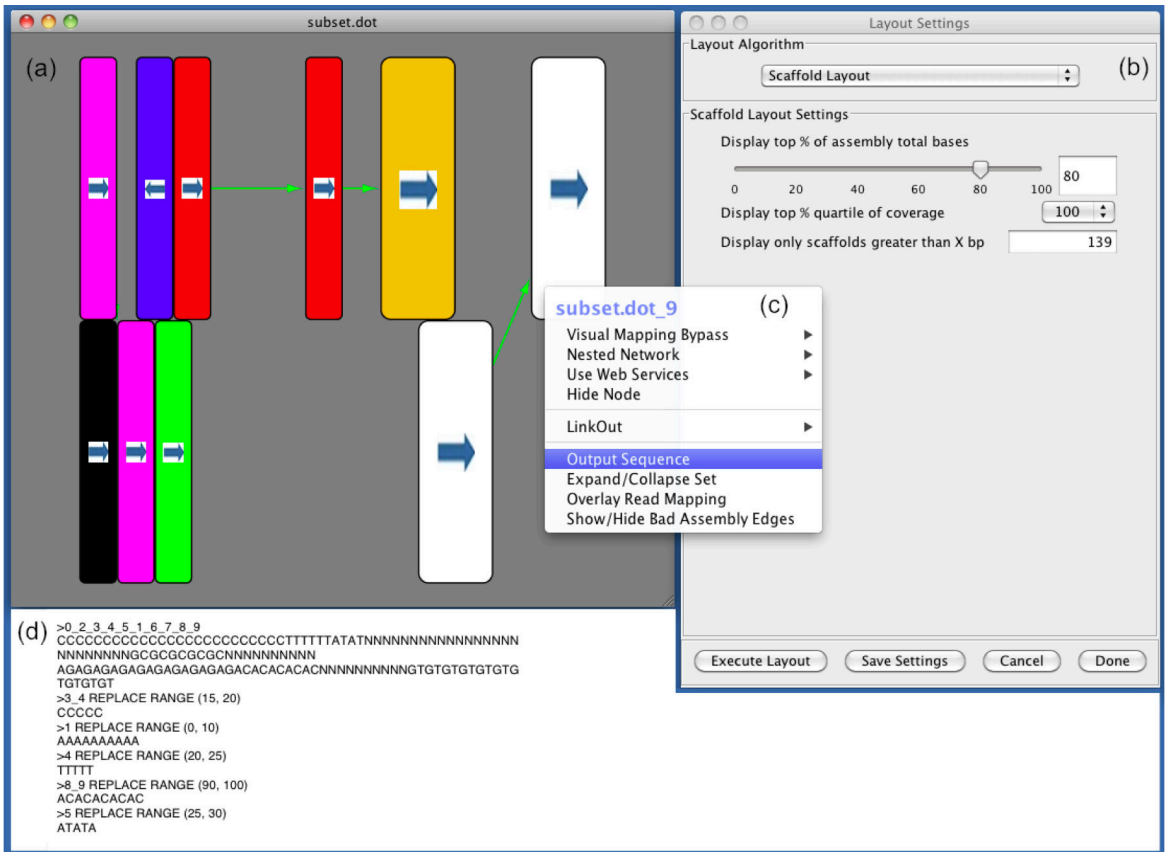


Figure 5.2: **ScaffViz metagenomic visualization screenshot.** a) shows the layout of contigs within a scaffold. Assembly metadata is displayed as node/edge attributes. These attributes can be used by built-in Cytoscape algorithms. Node width is proportional to the contig length. When available, node classification (for example taxonomic assignment) is indicated by color and node height is proportional to the coverage (the amount of oversampling of a sequence). Otherwise, as here, node color indicates coverage. b) Available layout settings. Here, the user can filter the view by scaffold length, total length, and coverage. c) The context menu supporting FastA output and the display/hiding of subsets of nodes. d) FastA output from the graph in a). The first sequence (master) is the longest traversal of the nodes. The remaining five records indicate variants within the set, corresponding to the five overlapping nodes in the graph.

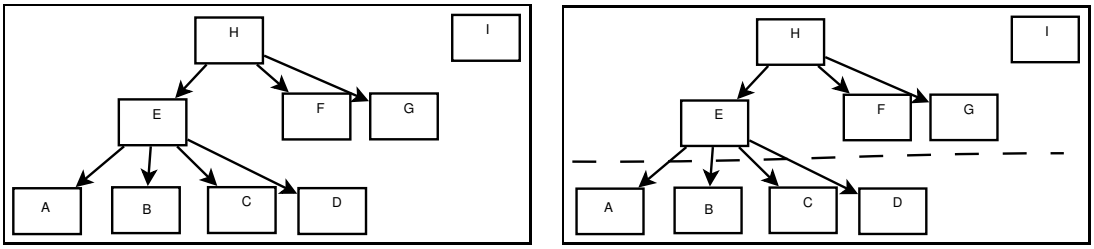


Figure 5.3: **ScaffViz representation of variations in a metagenome.** (left) Shows the in-memory representation of a motif forest. Nodes A, B, C, D, and I are contigs. Nodes E, F, G, and H are motifs encapsulating the nodes in their respective subtrees. (right) Shows an example cut in the motif forest. In this example, the user would only see nodes E, F, G (which would indicate they group into super-node H) and I. ScaffViz tracks the level of visibility the user has chosen. The user could expand node E as well, in which case ScaffViz would show nodes A, B, C, D (indicating they group into super-node E), and F, G (indicating they group into super-node H).

another set S'. The viewer outputs the longest path through a selected subset first. For any alternate sequence, it outputs an edit directive listing the alternate bases and their position within the longest sequence (Figure 5.2d), giving visual access to the motifs output by metAMOS.

5.2.4 Quality control

Given the relative immaturity of metagenomic genome assemblers, quality control is central for building a reliable assembly pipeline. An assembler may use a conserved region between multiple genomes to incorrectly join sequences, creating a chimer. We include optional QC modules in metAMOS to detect chimeric contig/scaffold formation and split them before downstream analysis. We borrow techniques from amos-validate [123] and utilize coverage, mate-pair information, and read mapping results to QC. In addition, we make use of existing gene calls to identify regions in assembled contigs or scaffolds that were greater than 1Kbp (in prokaryotes) and devoid of a gene call. While there could be false negatives in the gene caller, downstream steps (such as scaffolding) can re-join erroneously bro-

ken contigs. By contrast, false positives would allow errors to pass through and be compounded. Generally, metAMOS takes a cautious approach to avoid propagating assembly error through the pipeline. We feel this approach is a reference-free way to identify problematic regions in metagenomic assembly.

5.3 Results

5.3.1 metAMOS Performance

We begin by evaluating metAMOS performance on datasets varying from 10 million sequences up to 600 million (Fig 5.4). The runtime of metAMOS scales linearly with the number of input sequences. It is also able to process $>$ 100 million sequences in less than 3 days of runtime and 600 million in less than a week.

5.3.2 Visualization performance

We evaluated ScaffViz performance on seven datasets of varying size. The memory and time required is in Figure 5.5. As the figure shows, runtime is close to linear with respect to the number of elements in the graph (nodes + edges). The memory scaling is linear with respect to the number of nodes. The largest graphs on the plot represent real next-generation metagenomic datasets, including a viral metagenome, as well as a sample from the MetaHIT consortium [129]. ScaffViz can open all of them in under one minute using less than 1.5GB of memory. Therefore, ScaffViz is scalable to large graphs, allowing it to run on a laptop.

5.3.3 metAMOS assembly evaluation

Our goal is to analyze the feasibility of using metAMOS with genome assemblers that were not intended for use on metagenomic data (*e.g.* SOAPdenovo) and compare the results to those of produced by assemblers specifically designed for

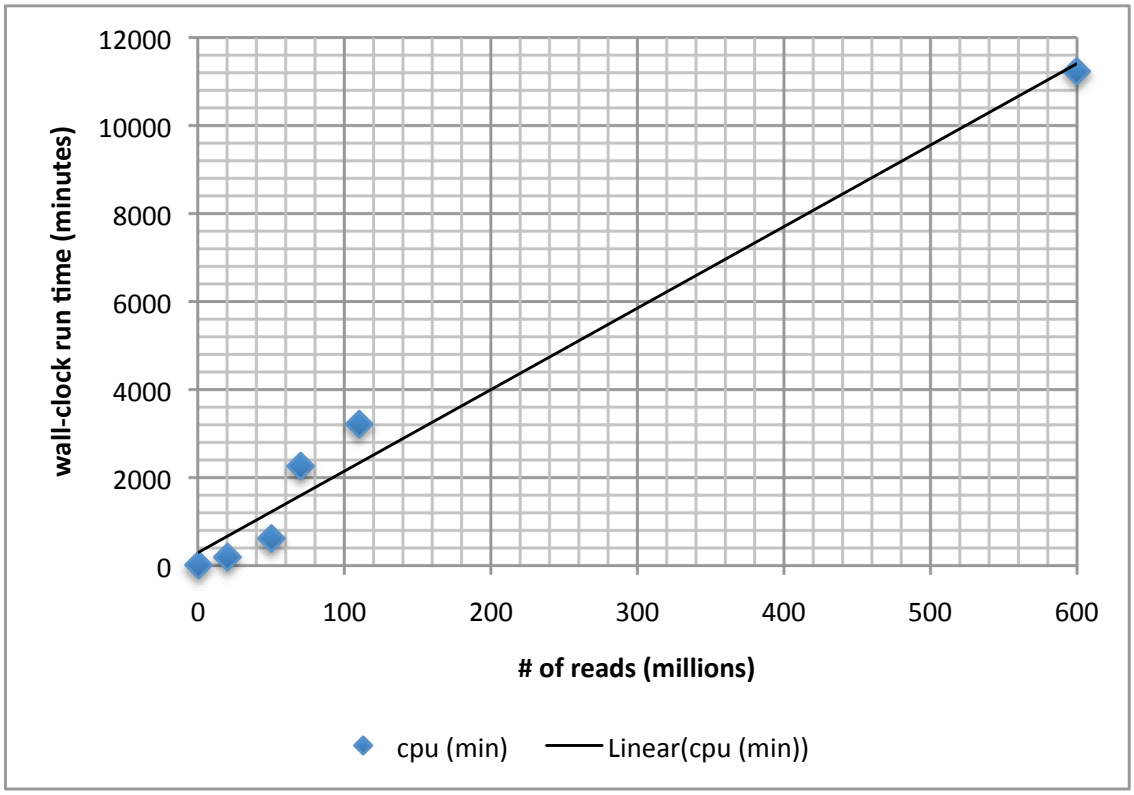


Figure 5.4: **Runtime of metAMOS scales linearly with number of sequences.** Six independent runs of metAMOS on datasets varying in size from 300,000 reads to 600 million. For all points, 16 threads were used and assembly was performed by SOAPdenovo.

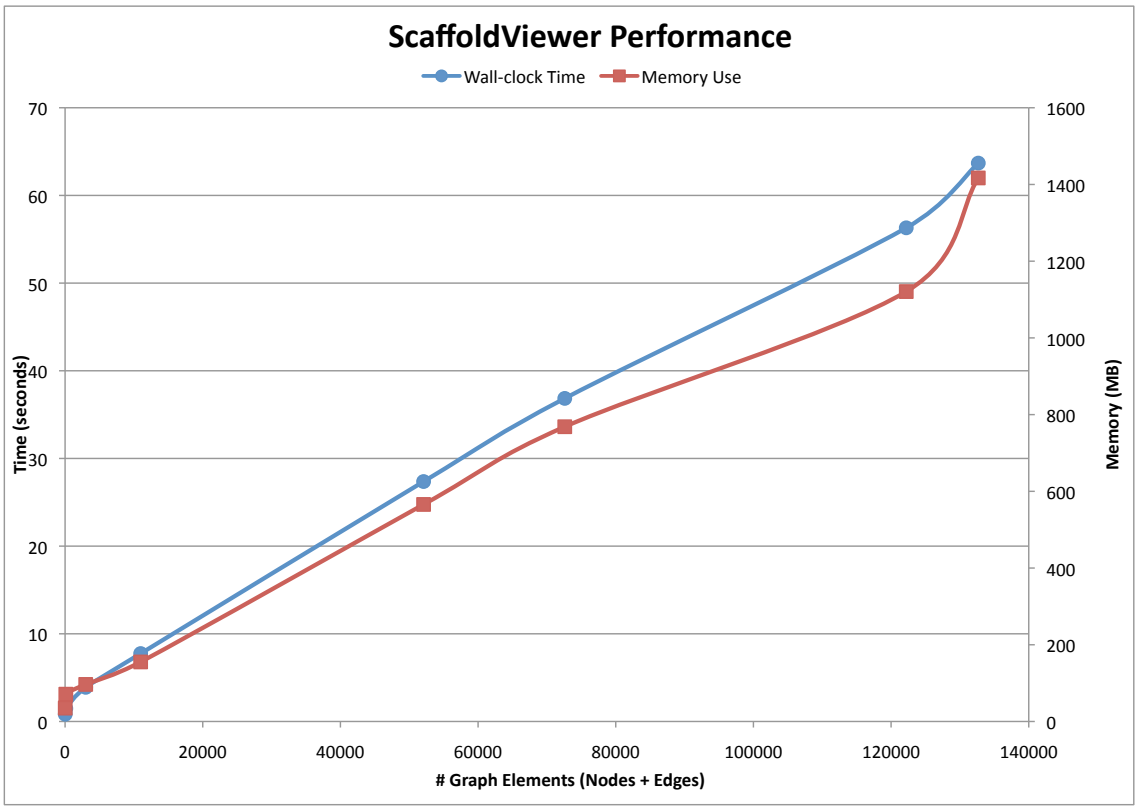


Figure 5.5: **Performance of ScaffViz (memory and runtime) scales linearly with graph size.** All tests were performed on a MacBook Pro (Intel *i*7 Dual Core 2.66GHz CPU, 8GB of RAM, and OSX 10.6.6). Built-in Java functions reported wall-clock time and memory. Cytoscape 2.8.0 was used, each dataset was loaded five times, and the time and memory averaged.

metagenomic data (*e.g.* Meta-IDBA). We use both mock datasets generated for the Human Microbiome Project (HMP) project [164] and real metagenomic samples from the HMP as well as Metagenomics of the Human Intestinal Tract (MetaHIT) [129] projects.

5.3.4 Simulated datasets

To compare and evaluate metagenomic assembly accuracy we first rely on metagenomic samples with known composition, specifically two mock communities created by the HMP. These communities represent the result of sequencing a mixture of known DNA fragments from organisms with known genomic sequences, comprising over 50 bacterial genomes and a few eukaryotes. The results obtained on real metagenomic samples are difficult to evaluate due to the absence of a golden truth reference. While not without limitations, this dataset has advantages over pure simulated data since the data were generated through an actual sequencing experiment from the mixed DNA sample, thereby capturing true artifacts of the sequencing technology.

Two HMP mock communities were generated: Even and Staggered. The reference genomes in these mock communities are precisely known, the abundances are fairly well known, and the reads were sequenced with the Illumina GAII instrument [8] replicating any biases and the error profile associated with the sequencing technology. Figure 5.6 (mock Staggered in blue color, and mock Even in red color) shows the average reference coverage (percent of the reference genome covered by good quality assembled contigs) for both of these mock communities when assembled with seven different methods: SOAP.utg (SOAPdenovo unitigs), MA (metAMOS+SOAPdenovo unitigs), midba (Meta-IDBA), Velvet, CA (Celera Assembler + metagenomic parameters), CAdeg (CA + degenerates). The CAdeg assembly warrants additional explanation. Celera Assembler marks certain contigs as de-

generates if they fail certain quality checks, including if they appear repetitive (in terms of depth-of-coverage statistics). In metagenomic samples contigs from abundant organisms may, therefore, appear repetitive and be marked as degenerates by CA. Including these allows us to capture more of the sequence of the community, however can also lead to increased errors (many degenerates are truly mis-assembled repeats). All assemblers have lower performance on the mock Staggered community, which is expected to better model the pattern of taxonomic diversity encountered in real data [70, 92, 144]. When taking correctness into account (Table 5.2), we observe that all assemblers make mistakes, especially in the category termed heavy mis-assembly. Heavy mis-assembly represents contigs with only one alignment in the reference genome covering less than 90% of the contig's length, or having multiple incompatible matches to a single reference. These results indicate that the choice of assembler has a strong influence on the final assembly results and choosing the ideal assembler requires taking into account both contiguity and correctness (Tables 5.1, 5.2). For example, while the CAdeg assemblies have the largest max contigs, it is at the expense of one of the highest error rates. Meta-IDBA has the largest contigs at 10Mbp but also generates more assembly errors than the more conservative SOAPdenovo assembly. metAMOS matches the best assemblers in terms of reference representation, having the most genomes over 90% covered with good contigs (Fig 5.7), while having the lowest rate of error (and chimera) in both datasets (Figs 5.8, 5.9).

It is important to point out that our goal is not to argue that the performance of metAMOS is better than that of other existing metagenomic assemblers. Our code, in fact, can use any assembler as a first step. Rather, metAMOS was designed to augment and improve upon existing methods, using specialized post-processing routines aimed at the unique set of challenges posed by metagenomic assembly. In this example, SOAP.utg (SOAPdenovo run without scaffolding information) is

Table 5.1: **Assembly contiguity and mapping statistics for the Mock Even and Mock Staggered datasets.** Assembly evaluation on mock Even (mockE) and mock Staggered (mockS) datasets. All contigs/scaffolds are mapped using MUMmer [82] at 97% ID and minimum matching length of 100bp. Contigs are considered good if they are completely covered except for 15bp on either end (to account for errors at end of reads). We also attempted to run Genovo [84] on these datasets. However, the program crashed after completing one iteration (out of a recommended 10 000). The single iteration required more than 24 hours to complete, indicating that Genovo is not scalable to large datasets.

Community	Assembler	# Ctg	Good	% Good	Total (MB)	Size @ 10Mbp	Num @ 10Mbp	Max Ctg
mockE	SOAP.utg	69 517	65 230	93.83%	51.9	28 208	195	249 819
mockE	MA (SOAP.utg)	63 440	62 971	99.26%	51.6	28 208	195	249 819
mockE	SOAPdenovo	64 345	63 796	99.14%	51.6	28 208	195	249 819
mockE	Velvet	11 770	11 354	96.47%	21.1	34 642	177	169 453
mockE	CA	7 164	6 293	87.84%	40.2	45 618	124	288 938
mockE	CAdeg	23 220	21 988	94.69%	48.4	45 618	124	288 938
mockE	Meta-IDBA	41 990	33 823	80.55%	49.0	50 595	116	204 186
mockS	SOAP.utg	69 633	54 088	77.68%	29.4	5 672	626	186 064
mockS	MA (SOAP.utg)	45 143	44 598	98.79%	29.2	5 672	626	186 064
mockS	SOAPdenovo	54 078	52 214	96.55%	29.2	5 672	626	186 064
mockS	Velvet	9 214	8 799	95.50%	10.6	6 441	499	186 086
mockS	CA	4 284	3 792	88.51%	18.4	7 594	475	219 878
mockS	CAdeg	17 407	16 534	94.98%	24.4	7 594	475	219 878
mockS	Meta-IDBA	72 526	27 285	37.62%	21.2	13 150	368	119 064

Table 5.2: **Assembly correctness on the Mock Even and Mock Staggered datasets.** All contigs are mapped as in Table 5.1. Contigs were extracted from scaffolds, if available by splitting at Ns. Slight mis-assemblies are alignments that cover 90% or more of the aligned contig in a single match. Heavy mis-assembly are alignments that cover less than 90% of the aligned contig in a single match or have two or more matches to a single reference. Chimeras are contigs with matches to two distinct reference genomes.

Community	Assembler	Slight Mis-assembly	Heavy Mis-assembly	Chimeras	Errors / MB
mockE	SOAP.utg	168	133	1	5.8
mockE	MA (SOAP.utg)	167	131	1	5.8
mockE	SOAPdenovo	179	135	1	6.1
mockE	Velvet	147	174	20	15.3
mockE	CA	499	305	12	20.3
mockE	CAdeg	648	374	27	21.6
mockE	Meta-IDBA	372	163	4	10.9
mockS	SOAP.utg	155	138	0	10.0
mockS	MA (SOAP.utg)	131	102	0	8.2
mockS	SOAPdenovo	161	135	0	10.1
mockS	Velvet	110	140	20	25.5
mockS	CA	228	173	12	22.4
mockS	CA deg	336	236	16	23.5
mockS	Meta-IDBA	150	158	1	14.6

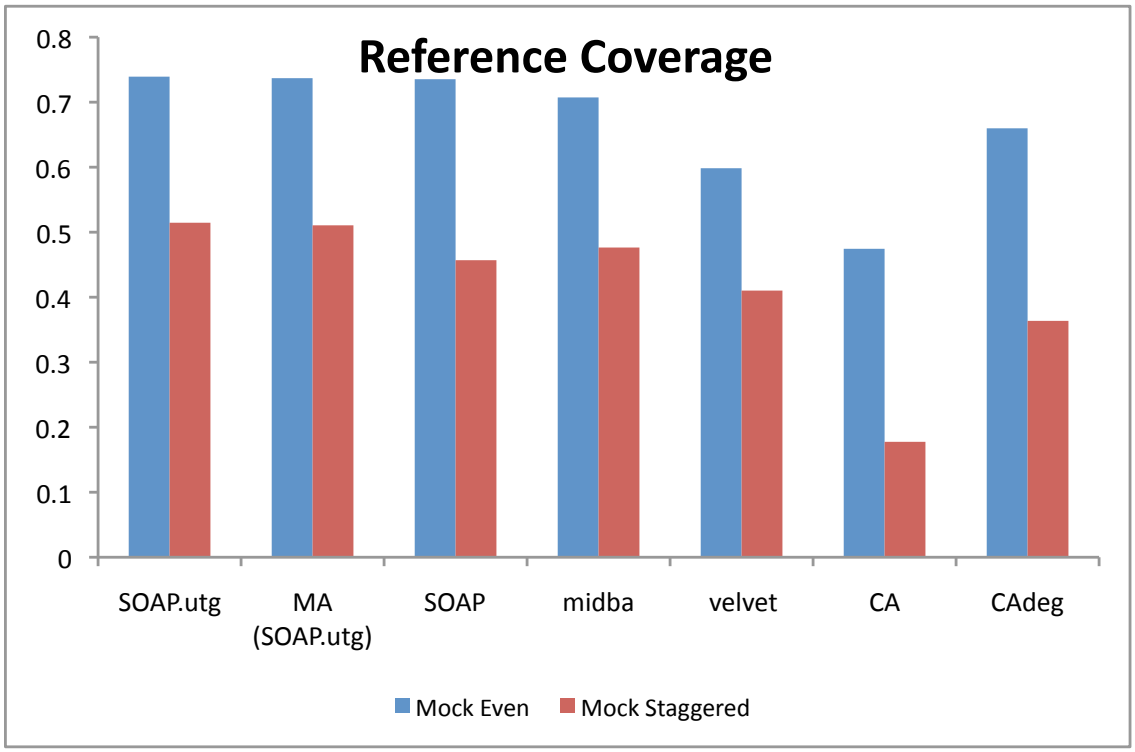


Figure 5.6: **Comparison of the average percentage of a reference correctly assembled in the Mock Even and Mock Staggered assemblies.** The average coverage is the percentage of a reference genome covered by correct contigs in the assembly. Only matches above a minimum length (100bp) with a high-quality single match at greater than 97% identity over 90% of the contig length are reported. In this way, single read matches and chimeric/mis-assembled contigs do not contribute to an assembler's coverage statistics.

the starting point for the metAMOS analysis. Compared to the full SOAPdenovo run, metAMOS has more genomes with 90% representation (or greater) in good contigs. metAMOS also introduces no new chimera while fixing assembly errors in both Mock and Even datasets (Table 5.2). Given that SOAPdenovo produced smaller and more accurate contigs while Meta-IDBA produced the largest contigs at 10Mbp and is second in terms of error to SOAPdenovo, we compare the two assembly approaches within metAMOS on a real metagenomic dataset.

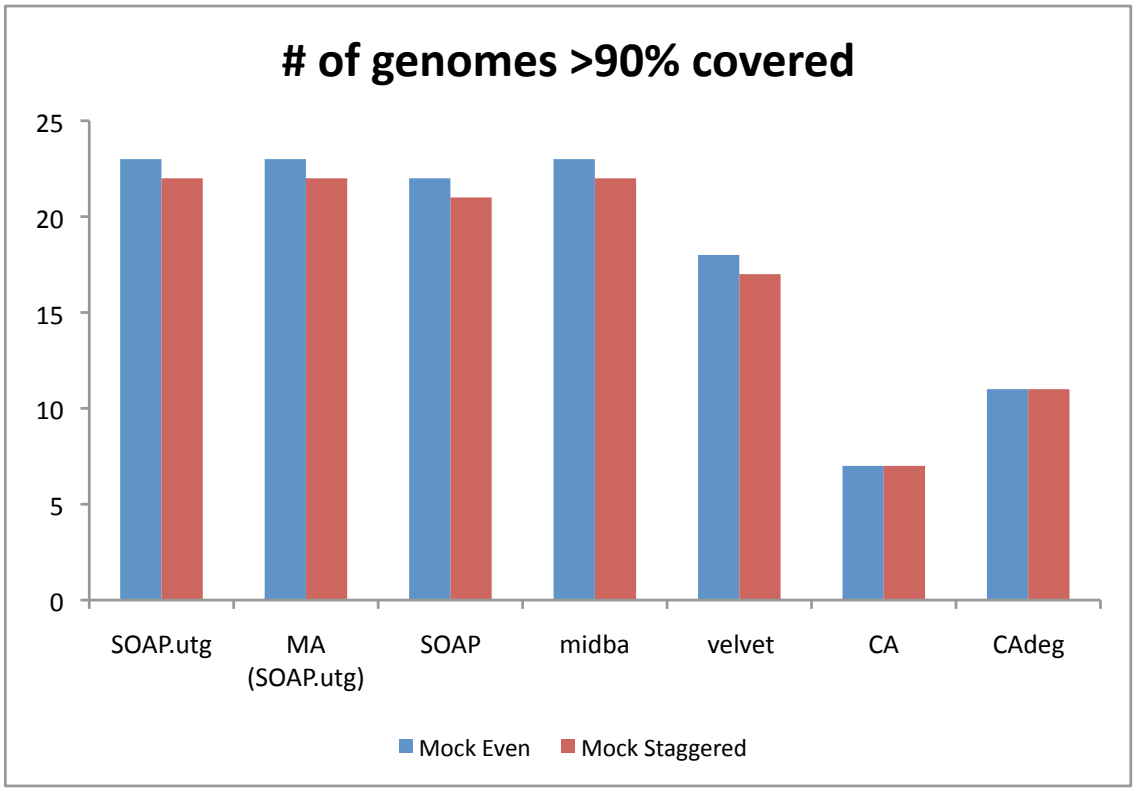


Figure 5.7: **Count of the reference organisms with over** 90% **of the genome correctly reconstructed by assembly.** Higher numbers are better. Matches are computed as in Figure 5.6.

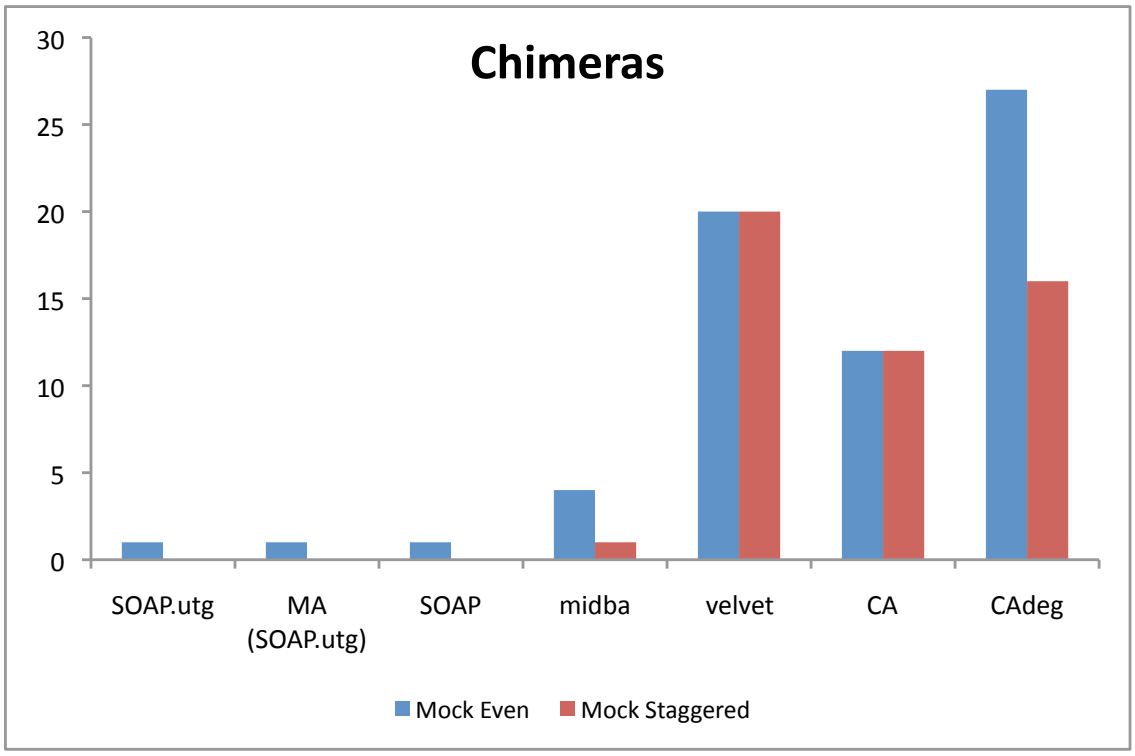


Figure 5.8: **Count of chimera in the Mock Even and Mock Staggered assemblies.** The number of chimeric contigs – contigs with mappings to more than a single reference genome – in the assemblies. Lower numbers are better, full assembly correctness metrics are in Table 5.2.

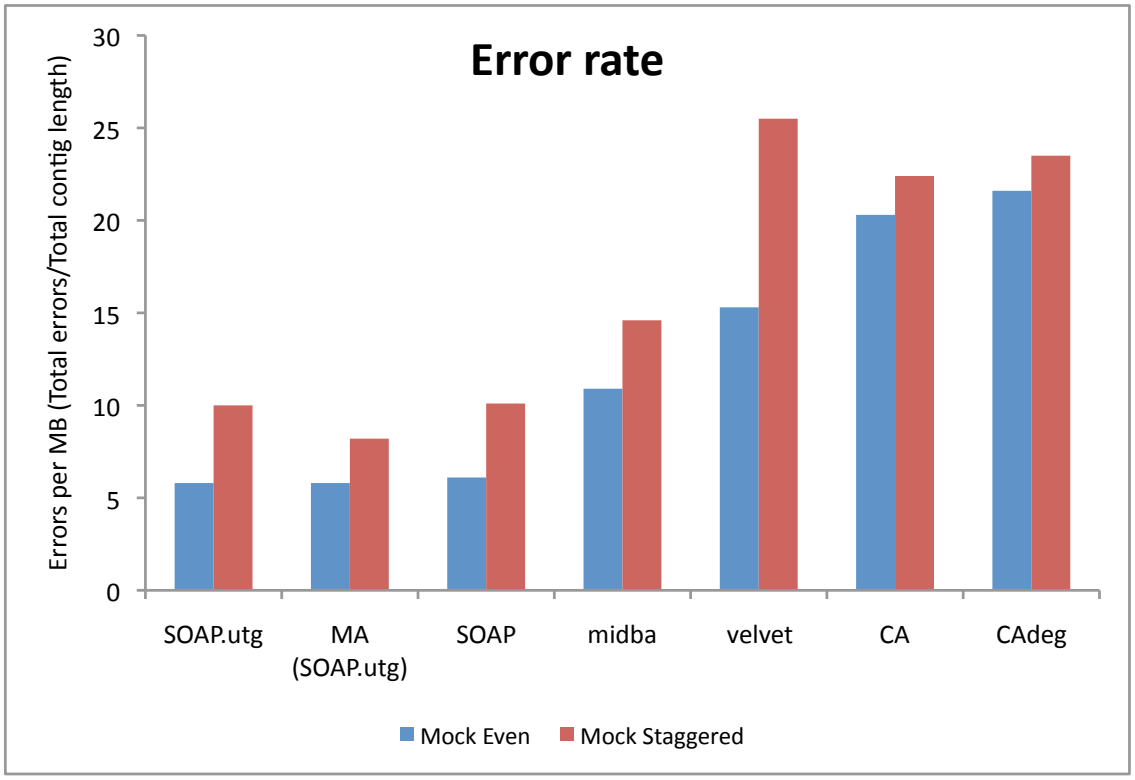


Figure 5.9: **Number of errors per megabase in the Mock Even and Mock Staggered assemblies.** The errors are reported as the sum of heavy mis-assemblies + chimera in Table 5.2. Lower numbers are better. metAMOS has the lowest rate of error, correcting the errors in SOAP.utg while not introducing new error to the assembly.

Table 5.3: **Assembly of the HMP tongue dorsum dataset with metAMOS.** Stats comparing two assemblies generated within metAMOS (using Meta-IDBA and SOAPdenovo, MetaGeneMark and Bambus 2) of tongue dorsum female sample (HMP, SRS077736). Unitigs indicate initial output of each assembler. Contigs are reported after scaffolding by Bambus 2 (splitting scaffolds at Ns).

		Unitigs		Contigs		Scaffolds	
Assembler	Total BP	#	Max	#	Max	#	Max
Meta-IDBA	119 075 843	678 034	220 488	673 291	220 488	644 997	443 823
SOAPdenovo	101 769 360	451 765	116 181	292 706	238 051	287 108	238 051

5.3.5 HMP tongue dorsum

Our second analysis was performed on real data (HMP tongue dorsum female sample). For this sample, the true and complete composition of the community is unknown; instead we constructed a reference genome set from the genomes identified by the HMP to have high similarity to the sequences within the sample. This dataset was previously assembled with Meta-IDBA and the published results demonstrated that Meta-IDBA was able to generate larger contigs than SOAPdenovo [117]. We used both SOAPdenovo and Meta-IDBA as starting points for the metAMOS pipeline. The results shown in Table 5.3 show that while Meta-IDBA produces a significantly larger maximum unitig (doubling that obtained by SOAP.utg), the resulting contigs and scaffolds are much closer in length. Focusing on producing larger unitigs in an initial assembly leads to higher error rates (Table 5.1) while metAMOS produces accurate unitigs and contiguous contigs/scaffolds. Figure 5.10 shows the Krona [114] plot for the sample which is automatically generated by metAMOS. The figure allows both for an overview of the taxonomic composition in a dataset as well as allowing interactive navigation to explore specific branches of the taxonomy.

To evaluate the correctness of these assemblies, we aligned them against our set of reference genomes. In Figure 5.11 we show the percentage of each reference genome covered by correctly assembled contigs. While both assemblers (SOAPdenovo and Meta-IDBA) vary in their ability to reconstruct individual genomes,

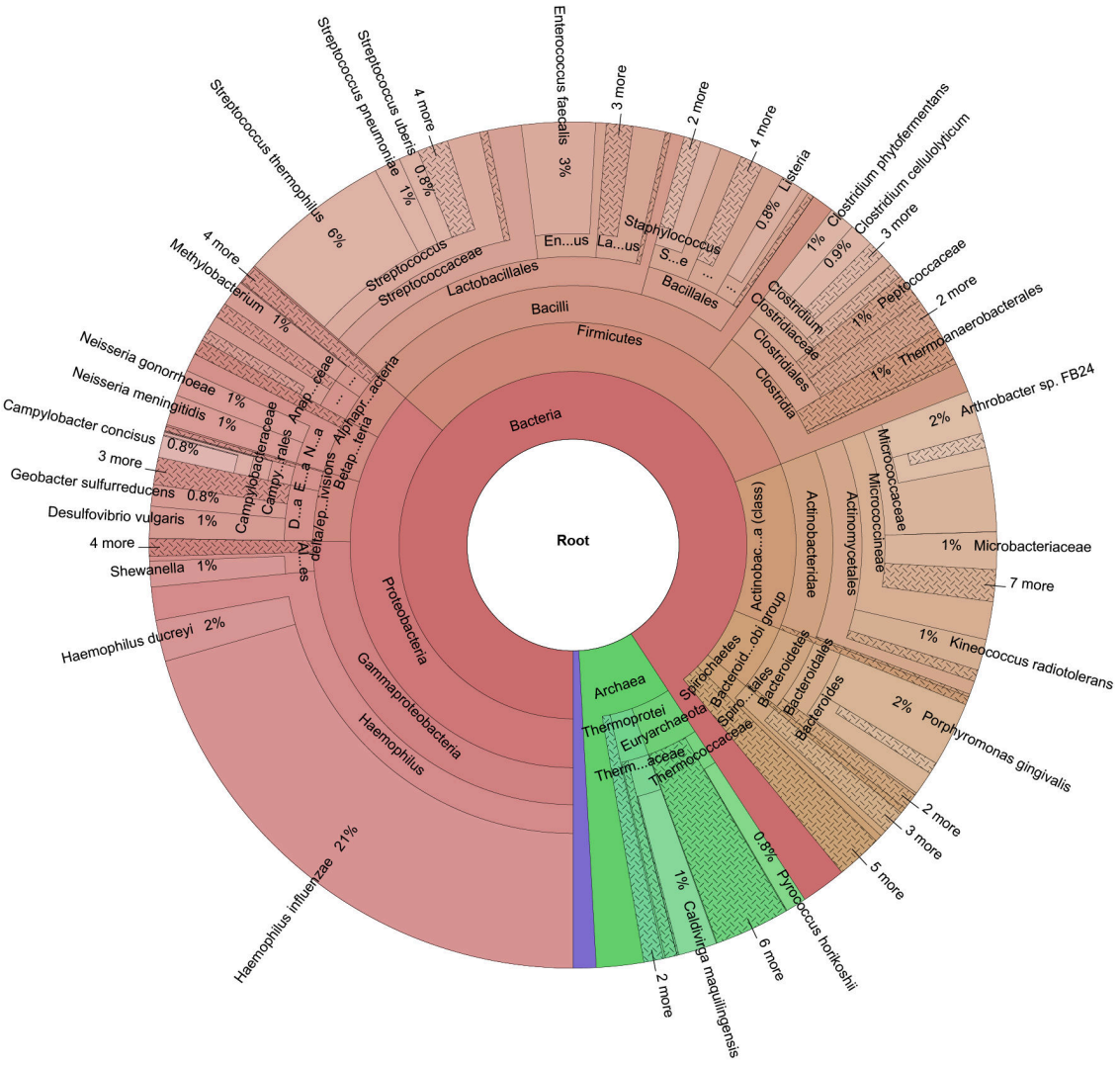


Figure 5.10: **Human tongue dorsum sample from the HMP project was classified using PhyloSift and displayed using Krona.**. Meta-IDBA was used for assembly, MetaGeneMark for ORF prediction, and PhyloSift for abundance estimates (based on predicted ORFs). The classifications can optionally be colored by average confidence, distinguishing uncertain from certain classifications.

metAMOS is able to improve on the starting assembly in all cases except one. In the case of *Rothia mucilaginosa*, which is present at over 200X in the sample, both SOAPdenovo and Meta-IDBA can accurately reconstruct the full genome. Most of the genomes have over 80% of their sequence correctly reconstructed. Five genomes from this reference set were poorly assembled (reference coverage of 40% or less) due their low abundance in the community. Figure 5.12 highlights the strong dependence between overall depth of coverage and ability to reconstruct the full genome sequence, indicating that genomes below 5X cannot be reconstructed by assembly. Figure 5.13 shows the same phenomenon using the relative abundance estimates of the individual organisms obtained with the taxonomic profiling tool MetaPhyler [91]. The results are highly consistent with those shown in Figure 5.12, and are nearly identical irrespective of the assembler used. This indicates that taxonomic profiling data can be used to help fine-tune the assembly parameters even when adequate reference genome sequences are not available.

5.3.5.1 Biological variant identification

metAMOS, through its use of the Bambus 2 scaffolder [78], is currently the only metagenomic assembly pipeline able to automatically identify assembly patterns indicative of genomic variation (termed variation motifs in the following). Figure 5.14 shows one of the top ranked variant motifs (spanning 1 212bp) reported by metAMOS. This motif is composed of two variant subregions (200bp in length, each), connecting to two larger nodes (500bp contigs) in the graph. Nucleotide alignments yield significant hits to *Streptococcus oralis* Uo5 and *Streptococcus sanguinis* SK36. The variant region in the middle contains 12 SNPs (Fig 5.14(b)). The SNPs fall within a poorly characterized hypothetical protein, distantly related to a *glutamic acid decarboxylase*. It is likely the protein varies between closely related strains of Streptococcus.

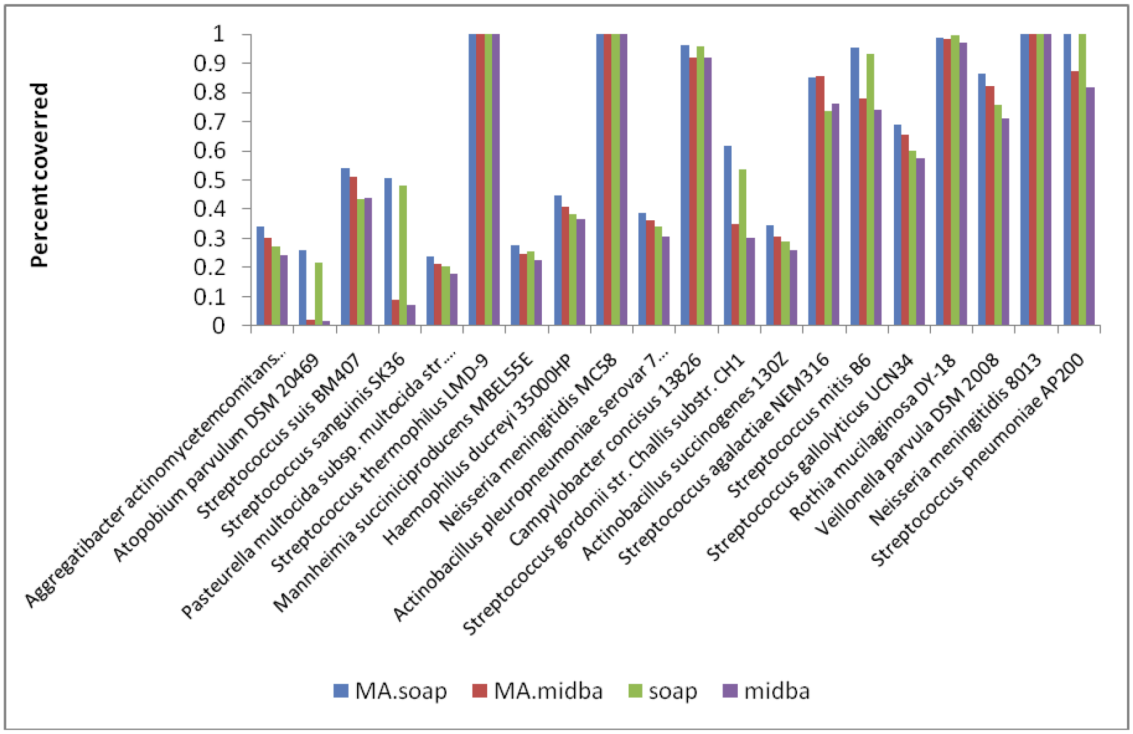


Figure 5.11: **Percentage coverage of the references by correctly assembled contigs in the HMP tongue dorsum sample.** Higher is better, matches are computed as in Figure 5.6. metAMOS improves the reference representation in all cases except one from the starting assembly.

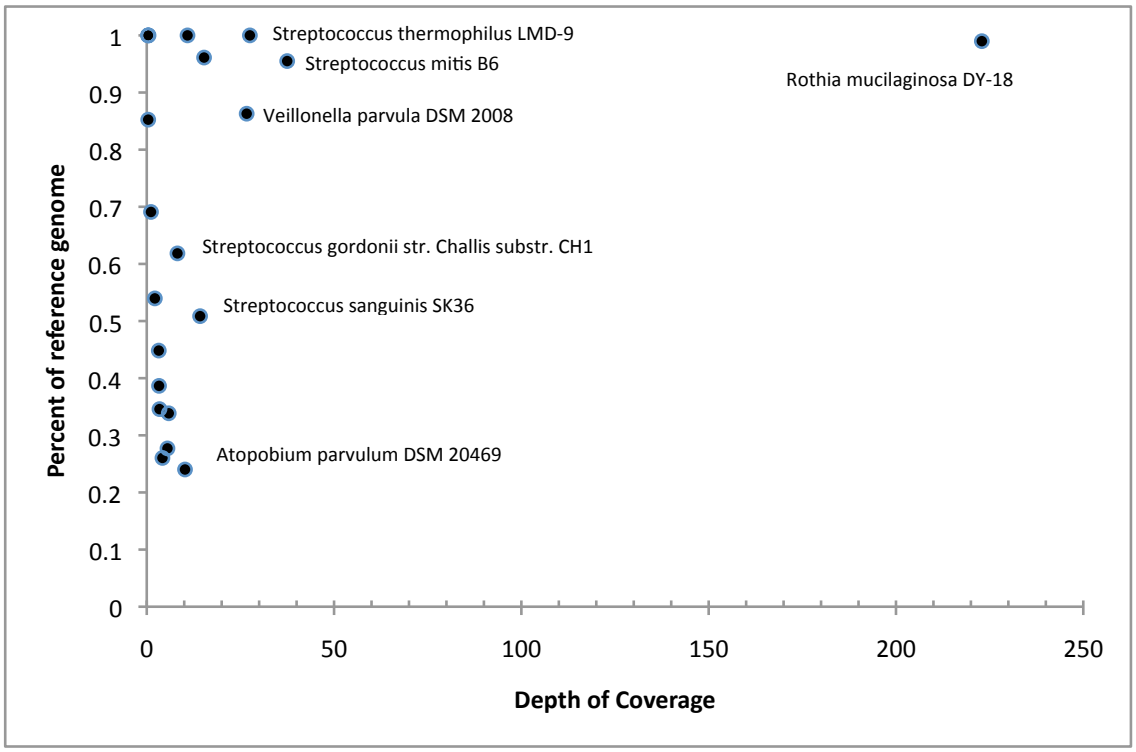


Figure 5.12: **Comparing depth of coverage versus the percentage of a reference that is correctly assembled.** Genomes above 5X can be accurately assembled, most below this threshold are not.

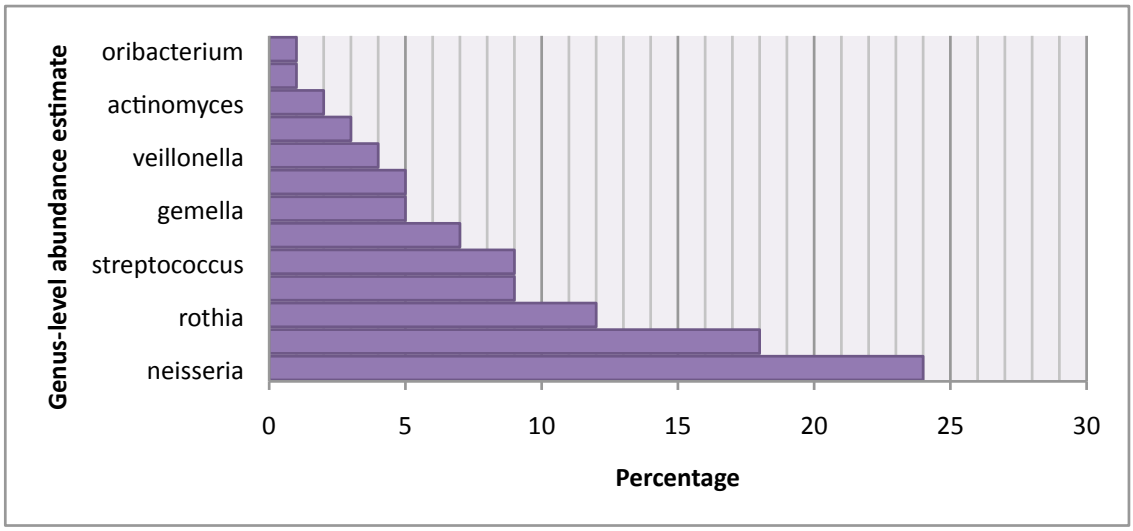
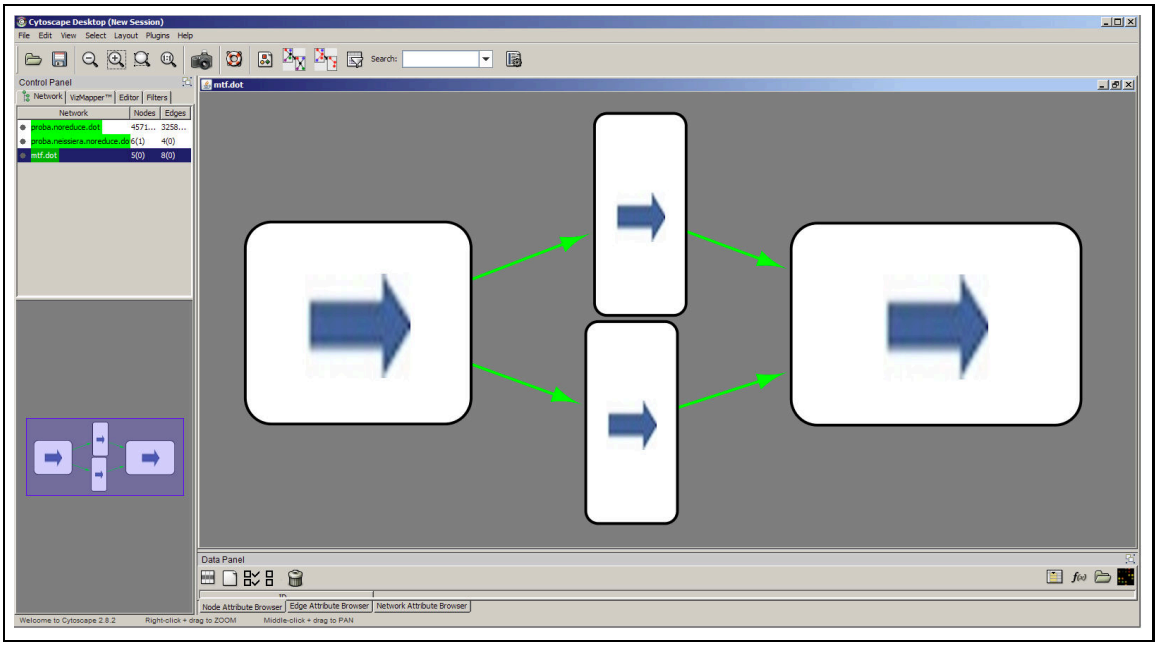


Figure 5.13: **metAMOS taxonomic abundance estimate on the HMP tongue dorsum sample**. Meta-IDBA was used for assembly, MetaGeneMark for ORF prediction, and MetaPhyler for abundance estimates (based on predicted ORFs).




```
Streptococcus_oralis_Uo5_mtf1     ATCTTGGAGAAGCTCAAGATAATCATCTGGATTGATGACTTTTAGATAACCATCTAAA
Streptococcus_oralis_Uo5_mtf2     ATCTTGGAGAAGCTCAAGATAATCATCTGGATTGATGACTTTTAGATAACCATCTAAA

Streptococcus_oralis_Uo5_mtf1     GGTTCCcAagCCATCtTCcTGCCAAATCTGtAaCAACTCAGCAGGGACTTGGTCCTTG
Streptococcus_oralis_Uo5_mtf2     GGTTCCaAgaCCATCcTCtTGCCAAATCTGaAcCAACTCAGCAGGGACTTGGTCCTTG

Streptococcus_oralis_Uo5_mtf1     TTTTTCAATaACTTCTTGGGGCATATCtGCCaCTTTgATAAAGTTTTCTAGCATgTTT
Streptococcus_oralis_Uo5_mtf2     TTTTTCAATcACTTCTTGGGGCATATCcGCCtCTTTtATAAAGTTTTCTAGCATaTTT

Streptococcus_oralis_Uo5_mtf1     CTCCGATTTGATTTTTAGCATCATTTCTCTACAACCATAGTATACCATAAACCATATG
Streptococcus_oralis_Uo5_mtf2     CTCCGATTTGATTTTTAGCATCATTTCTCTACAACCATAGTATACCATAAACCATATG
```

Figure 5.14: **ScaffViz visualization of a variation reported to metAMOS.** A motif within the HMP tongue dorsum dataset. a) The ScaffViz representation of the motif, indicating the overlapping 200bp contigs. b) The multi-alignment of the variants reported by Bambus 2 showing 12 SNPs.

This simple example highlights the utility of variation motifs – typical assembly software would break the assembly in this region or forcefully merge the two variants. Instead, metAMOS preserves the contiguity of the genomes backbone while also outputting the pattern of variation detected in this region. Note that such regions are difficult to identify within the output of existing assemblers – their identification requires substantial manual examination of the assembly output [29].

5.3.6 Sexual dimorphism in the human gut microbiome

To demonstrate the types of analyses enabled by metAMOS, we investigate sexual dimorphism in the human gut microbiome. Microbiome differences between different genders were previously demonstrated in macaques [100] and mice [143], and such differences have yet (to the best of our knowledge) to be explored in humans. To explore whether evidence of sexual dimorphism could be gleaned from metagenomic data analyzed with metAMOS, we focused on six subjects (three male and three female), all of the same age (59 years) and from the same country (Denmark), whose microbiome was sequenced as part of the MetaHIT project. Before we proceed with the description of the analysis we would like to stress the fact that conclusively assessing whether sexual dimorphism within gut bacteria is a real phenomenon in the human population requires extensive studies of much broader scope than that being performed here. This is especially true given that there is no known physiological basis for dimorphism. We simply focus on this problem because: (a) such an analysis has not been previously performed; and (b) the overall analysis approach is typical of a wide range of comparative metagenomic analyses that are commonly performed in a clinical setting.

Three male and female samples, comprising more than 70 million sequences each, were co-assembled with metAMOS in under four days, using 20 cores. The assembly statistics for both male and female samples are reported in Table 5.4.

Table 5.4: **Assembly of the MetaHIT gut microbiome data for Danish adults.** Assemblies were performed with SOAPdenovo. ORFs were predicted with MetaGeneMark. Scaffolding was done with Bambus 2.

							Contig		Scaffold	
Sex	# Samples	# reads	length	Age	Country	Total BP	#	Max	#	Max
Male	3	72M	75bp	59	Denmark	138 506 290	415 145	67 596	333 141	178 391
Female	3	79M	75bp	59	Denmark	108 975 669	350 327	90 489	279 061	238 222

While the maximum contig and scaffolds sizes are similar, the male sample has approximately 30 Mbp more assembled bases than the female sample. To explore the biological reason for this discrepancy, Metaphyler [91] was run both on the individual reads, pre-assembly, and on the final collection of ORFs, post-assembly. The taxonomic profiles pre-, and post-assembly are highly concordant (correlation coefficients of 0.998 and 0.993 for the male and female samples, respectively). We estimated that the equivalent Metaphyler analysis would require ≈ 300 times the computational time of the the post-assembly analysis. This result highlights the power of assembly as a data 'compression' tool, and suggests many analyses currently performed on the reads directly (*e.g.*, functional annotation [48, 103], or pathway analysis [1]) would be substantially sped up if performed on the assembled data instead.

In terms of taxonomic composition, male samples were dominated by the Bacteroidales order while female samples were dominated by the Clostridiales order. While not statistically significant at the order level due to our low sample size (Fisher exact test, $p = 0.1192$), the difference between male and female samples was significant at the family and genus levels (Fisher exact test, $p = 0.041$ and $p = 0.022$ on family and genus, respectively). Interestingly, an analysis of the macaque data [100] using MetaStats [177] shows that Bacteroidales is differentially abundant between males and females, concordant with our observations.

5.4 Discussion

Overall, our results indicate that the choice of assembler has a strong influence on the final assembly results and choosing the ideal assembler requires taking into account both contiguity and correctness. More aggressive assembly approaches can result in more contiguous assemblies, but that also generates more errors, often of the most severe kind (chimeras). Thus, given a novel metagenomic dataset with no knowledge of taxonomic composition, it can be difficult to choose the appropriate assembler. This motivates our focus on fast end-to-end analysis and inclusion of multiple assembly methods, allowing the user to tailor the pipeline to their data, and tailor tools to answer questions rather than answering a limited set of questions allowed by available tools.

Irrespective of the starting point, the assemblies generated by metAMOS are similar in quality, suggesting the effectiveness of our approach: conservative assembly followed by error correction and metagenomic scaffolding. The scaffolding analysis, performed with Bambus 2, is largely unaffected by the specific features of the underlying sequencing technology. The modular design of metAMOS enables its adaptation to new types of data by simply incorporating genome assembly tools tuned to the specific features of the new data. As an example, the combination of Velvet-SC (an assembler already integrated within metAMOS which is targeted at data derived from single–cell experiments) and the coverage–independent repeat detection procedures provided by Bambus 2 allows the use of metAMOS in single–cell projects. Furthermore, the ability to easily switch the metAMOS pipeline between different assembly and analysis tools allows users to fine–tune metAMOS to the characteristics of their environment and datasets being analyzed.

Finally, the modular design and open–source licensing model enables researchers to adapt metAMOS to new applications. We hope that the availability of metAMOS will encourage researchers to contribute their own analysis modules, and that

this framework will accelerate developments in this field by allowing scientists to focus their attention on individual components rather than having to re–implement all the components of a metagenomic pipeline. In addition to our primary goal of providing biologists with an integrated analysis pipeline for metagenomic data, we also hope that our software will help reduce the duplication of efforts in this field and allow a larger number of researchers to focus on the difficult analysis challenges posed by metagenomics and other emerging fields.

5.5 Conclusion

The goal of metAMOS is to provide an integrated environment for metagenomic assembly and analysis, relying both on existing and novel algorithms and software tools. Results on both mock communities with known sequence composition and real metagenomic data demonstrate that the metAMOS pipeline can generate accurate and contiguous assemblies of metagenomic datasets. metAMOS substantially improves upon the quality of initial assemblies constructed either with conservative assemblers developed for single genomes (*e.g.*, SOAPdenovo), or with assemblers specifically targeted at metagenomic data (*e.g.*, meta-IDBA).

In addition to assembly, metAMOS provides several features important for the downstream analysis of the resulting data, including taxonomic profiling, gene detection, and identification of motifs indicative of genomic variation. Future enhancements to our package will include integration with existing pipelines for functional annotation and metabolic analysis (such as HUMAnN [1]). We would like to stress that currently there are no other tools that provide such integration between the different steps of a metagenomic analysis while supporting metagenomic specific tools (such as gene finders, assemblers and scaffolders). Our pipeline enables the effective, accurate, and efficient conversion of the raw output of a sequencing instrument into refined biological entities (genes, genomic contigs, and genomic variants).

Furthermore, metAMOS is the only metagenomic analysis pipeline that provides the ability to automatically detect and analyze genomic variation within metagenomic assemblies. We have demonstrated above how this feature can be used to pinpoint biologically relevant differences between closely related genomes co–assembled within a sample.

Chapter 6

Conclusion

High-quality assemblies are critical for all aspects of genomics, especially genome annotation and comparative genomics. For example, many microbial genomic analyses depend on finished genomes [37], but producing finished sequence remains prohibitive with the cost of finishing proportional to the number of gaps in the original assembly. Eukaryotic genomics requires continuous assemblies to capture long, multi-exon genes and to determine genome organization and structural polymorphisms. In addition, recent work has suggested *de novo* assembly may be superior to read mapping approaches for discovering large structural variations, even when a reference genome is available [90]. This is especially significant for understanding the genetic variations of cancer genomes and other human diseases such as autism that frequently contain gene fusions, copy number variations, and other large scale structural variations [33, 146]. It is clear that higher quality assemblies, with long unbroken contigs, will have a positive impact on a wide range of disciplines.

The computational tools and methods presented significantly advance the state-of-the-art in genome assembly and are applicable to other computer science research areas. For example, the alignment and correct algorithm (Chapter 3) may be useful for natural language processing. Hazen *et al* [56] propose to combine automatically-generated speech transcription with a human-generated approximate transcript. These are aligned with substitutions, insertions, and deletions and the approximate transcript is corrected. Our algorithm could be used to combine multiple automatic transcripts with a human-generated approximate transcript. The modified betweenness centrality (Chapter 4), scaled by an appropriately defined expectation (length in our case) can be applied to social networks to find not only the

most connected nodes but those more connected than would be expected and is the first OpenMP [20] implementation of k–betweenness to the best of our knowledge. The viewer (Chapter 5) is applicable to any large graph with positional constraints and can automatically collapse subsections of a graph.

I have contributed novel computational methods both to utilize emerging sequencing for genome assembly and expand its efficiency on novel targets. Together, these advances form a comprehensive genome assembly and analysis toolset and enable new avenues of biological discovery.

Part III

Appendices

Appendix A

Supplementary Data for An algorithm for automated closure during assembly

A.1 Methods

A.1.1 Reads

The bacterium *Escherichia coli* O157:H7 str EC4115 was sequenced with Sanger chemistry and is deposited at the NCBI Trace Archive. The reference [GenBank: CP001163], [GenBank: CP001165], [GenBank: CP001164] consists of two circular plasmids and a circular genome of 94 644, 37 452, and 5 572 075 bases respectively. The bacterium *Escherichia coli* K12 substr MG1655 was sequenced using 454. The WGS data is available through the Short Read Archive [SRA:SRA001028]. The reference [GenBank: NC_000913] consists of one circular genome of 4 639 675 bases. The bacterium *Salmonella enterica* subsp. enterica serovar Schwarzengrund str. CVM19633 was sequenced with Sanger chemistry and is deposited at the NCBI Trace Archive. The reference [GenBank: CP001125], [GenBank: CP001126], and [GenBank: CP001127] consists of two circular plasmids and a circular genome of 110 227, 4 585, and 4 709 075 bases respectively. The bacterium *Burkholderia mallei* NCTC 10247 was sequenced using Sanger chemistry and deposited at the NCBI Trace Archive. The finishing reads are also available from the NCBI Trace Archive. The reference [GenBank: CP000548], and [GenBank: CP000547] consists of two circular chromosomes of 3 495 687 and 2 352 693 bases respectively. The bacterium *Corynebacterium amycolatum* SK46 HMP033 was sequenced using 454 and Sanger. The WGS data is available through the NCBI Trace Archive and the Short Read

Archive [SRA:SRR005142]. The ciliate protozoan *Ichthyophthirius multifiliis* G5 was sequenced using 454 and Sanger. [The data is scheduled to be deposited in the SRA and the NCBI Trace Archive and is available by contacting the authors] The 454 reads were generated by the GS FLX Titanium pyrosequencing platform. They were processed with Celera Assembler to remove duplicates, detect linker, and split paired ends. For comparison with Dupfinisher, NCBI trace archive was unsuccessfully searched for reads with trace_type_code other than WGS belonging to *Methanospirillum hungatei* JF-1, *Rhodoferax ferriducens* DSM 15236, or *Shewanella baltica* OS155.

A.1.2 Finishing reads

The finishing reads were obtained from JCVI databases. For finishing reads generated from a clone, the clone-end reads were provided to Celera Assembler as bounding reads. Not all finishing reads had bounding reads.

A.1.3 Software

Celera Assembler software was run using run-time parameters recommended for each sequencing technology. The Sanger-only assemblies used the unitigger module while the assemblies with 454 data used the BOG module from CABOG. The *I. multifiliis* assembly used a 10% error rate instead of defaults. The specific version is marked with CVS tag WGS_CLOSURE-6_00- BRANCH and will be packaged starting with the 6.1 release. Newbler version 2.3 was used with default run-time parameters as the Alternate pipeline. Dupfinisher was kindly provided by its authors.

A.2 Analysis

Continuity statistics were gathered from each assembly using analysis of the FastA output files. The gap statistics were gathered from each assembly using scripts for analyzing assembly output. The MUMmer package [82] was used to compare assemblies to the references by running `nucmer ---maxmatch`. To identify candidate gaps to evaluate the bounded assembly we focus on gaps caused by genomic repeats, both with and without consensus sequence in the control. First, we identify regions of the control assemblies that had zero coverage in reads, a consensus sequence due to placement of a (repeat) unitig, and coverage in the unitig at least twice that of the overall scaffold average (Fig 2.1(a)). Separately, we listed gaps that have no consensus sequence in the control assembly (Fig 2.1(b)). The assemblies were aligned by using `nucmer ---maxmatch` and `show-tiling` was used to look for split contigs in either assembly. We also looked for any gaps that have no consensus sequence in the bounded assembly but do in the control. There were none in our datasets. The `show-snps` program from the MUMmer package [82] was used to identify SNPs between the reference and both control and bounded assemblies. The matches were first filtered by running `delta-filter -1` and the results used as input for `show-snps` (with no parameters). Regions where the control assembly had gaps (Ns) in the sequence were not included in SNP counts. The total number of SNPs in the bounded assembly but not the control assembly and vice-versa were tabulated.

A.2.1 Comparison to reference

Comparison to Reference. The assemblies of *E. coli* O157:H7 were examined and compared to the available reference. Eight alignments of 99% identity over 99% length of the Bounded assembly contigs cover 99% of the *E. coli* genome. Nine

alignments of 99% identity over 99% length of the Standard assembly contigs cover 99% of the *E. coli* genome. In both cases, there is a mis-join at the end of one contig due to an inverted copy of the rRNA operon The Standard assembly contained two surrogate (repeat) unitigs, each placed six times. In the Bounded assembly both surrogates were placed seven times. An NCBI BLAST [3] search confirmed that the two surrogates make up the rRNA operon, known to occur seven times in the wild-type genome [19]. In addition, several other surrogates were placed one more time in the Bounded assembly than in the Standard assembly. Thus, the Bounded assembly seems to have a more complete representation of this repeated structure.

The assemblies of *E. coli* K12 were examined and compared to the available reference. A total of 313 alignments of 99% identity over 99% length of the Bounded assembly contigs cover 97% of the *E. coli* K12 genome. A total of 313 alignments of 99% identity over 99% length of the Standard assembly contigs cover 97% of the *E. coli* K12 genome. In this case, the Standard and Bounded assembly has the same representation of the reference genome, consistent with their assembly statistics in Table 2.3.

The assemblies of *S. enterica* were examined and compared to the available reference. Six alignments of 99% identity over 99% length of the Bounded assembly contigs cover 99% of the *S. enterica* genome. Six alignments of 99% identity over 99% length of the Standard assembly contigs cover 99% of the *S. enterica* genome. Once again, the Standard and Bounded assembly has the same representation of the reference genome.

The assemblies of *B. mallei* were examined and compared to the available reference. Nineteen alignments of 99% identity over 99% length of the Bounded assembly contigs cover 99% of the *B. mallei* genome. Nineteen alignments of 99% identity over 99% length over 99% length of the Standard assembly contigs cover 94% of the *B. mallei* genome.

Appendix B

Supplementary Data for Hybrid error correction and de novo assembly of single-molecule sequencing reads

B.1 Analysis

B.1.1 Analysis of PacBio sequences

Sequences were aligned to the reference using `blasr` with default parameters.

B.1.2 Assembly of uncorrected PacBio data

Three commonly available OLC and de Bruijn assemblers were used to assemble the uncorrected PacBio sequences for the *Lambda* phage. We ran SOAPdenovo v1.05, Velvet v1.1.06 and CA with the bogart unitigger. For SOAPdenovo, the k-mer size was varied from 3 to 127 mer and the assembly that covered the largest percentage of the reference was picked. For Velvet, VelvetOptimizer v2.2.0 was used to vary the k-mer size from 5 to 63 and the assembly covering the largest percentage of the reference was picked. For CA, merSizes from 10 (which produced no assembly) to 22 (the default) were used. The CA unitig, overlap, consensus, and cgw error rates were all set to 25%. The PacBio sequences were also corrected using our algorithm via 50X of Illumina data and assembled by SOAPdenovo, Velvet, and CA (Table 3.2). The CA assembly used the parameters (`overlapper=ovl merSize=14 unitigger=bogart`). As the SOAPdenovo assembly was representative of de Bruijn assemblers, we used it for subsequent experiments in the paper.

B.1.3 Comparison of OLC and de Bruijn assemblers

To test the effect of read length on assembly we used real+simulated PacBio+Illumina data for *Saccharomyces cerevisiae* S228c. We used SOAPdenovo v1.05 to demonstrate the de Bruijn approach. Parameters `-all -K 63`, and `-all -K 127` were used. Celera Assembler was used to demonstrate the OLC approach using parameters `merSize=14 unitigger=bogart`. Contigs were broken on error as outlined in GAGE [137]. The baseline SOAPdenovo assembly had an N50 of 35Kbp.

B.2 Test Data

We have tested the algorithm using eight hybrid data sets. Whenever subsets of coverage were used, a random subset was selected using the CA gatekeeper command. First a new gatekeeper store was created using the command `gatekeeper -T -F -o tmp.gkpStore pacbio.frg`. A subset was created using the command `gatekeeper -allreads -dumpfrg -randomsubset 0 <total bp> tmp.gkpStore`. A genome size of 5.5Mbp was used for *E. coli* C227-11 and 5.0Mbp for *E. coli* 17-2 and *E. coli* JM221. For the subset tests of *E. coli* 17-2 and *E. coli* JM221, a random subsets was selected as a percent of total available sequence (up to a max of 275Mbp corresponding to 50X of a 5Mbp genome).

Lambda PacBio RS sequences and simulated data are available from http://www.cbcb.umd.edu/∼sergek/PacBio/index.html.

Escherichia coli PacBio RS sequence is available from the PacBio DevNet Portal (http://www.pacbiodevnet.com/Share/Datasets/E-coli-K12-Resequencing). The Illumina sequences used for correction are available under SRX000429.

The genomes *Escherichia coli* C227-11, *Escherichia coli* 17-2, and *Escherichia coli* JM221 PacBio RS and PacBio CCS sequences are available from the PacBio DevNet Portal (http://www.pacbiodevnet.com/Share/Datasets/E-coli-Outbreak) [130].

The University of Maryland Institute for Genome Sciences generated the Illumina/Roche 454 sequences. The UMD SOM data as well as the simulated Illumina sequences are available at http://www.cbcb.umd.edu/∼sergek/PacBio/index.html. For correction, we generated Illumina data from the assembly published in [130] using wgsim. The simulated Illumina mate-pairs and paired-ends for Illumina assembly were generated from the completed outbreak genome by BGI (ftp://ftp.genomics.org.cn/pub/Ecoli_TY-2482/Escherichia_coli_TY-2482.chromosome.20110616.fa.gz) using wgsim.

Saccharomyces cerevisiae S228c Illumina and PacBio RS sequences were generated by Cold Spring Harbor Laboratory and can be downloaded at http://www.cbcb.umd.edu/∼sergek/PacBio/index.html.

Melopsittacus undulatus consisted of Illumina, 454, and PacBio sequencing. Duke University and Roche generated the 454 sequences. The Illumina sequencing used for correction was generated by Illumina UK. The Illumina sequence used for ALLPATHS-LG assembly was generated by the Beijing Genome Institute (BGI). Pacific Biosciences generated the PacBio RS sequences. The sequences are available from the assemblathon project (http://assemblathon.org/).

RNA-Seq sequencing of *Zea mays* B73 was performed using both Illumina and PacBio RS at the DOE Joint Genome Institute. A total of 125M Illumina GAII paired-end reads and 388M Illumina HiSeq 150bp reads were generated with a mean insert size of 248bp. The overlapping paired-end reads were joined together to form 250bp unpaired fragments using the method of Rodrigue *et al.* [134]. A total of 230M pairs (460M reads) could be confidently joined. These 250bp sequences were used for correction. The maize RefGen v2 assembly was used for accuracy assessments and is available from http://www.maizesequence.org.

B.3 Assembly

B.3.1 Correction and assembly

The correction pipeline was run using the command `pacBioToCA` with the `parameters -length 500 -partitions 200 -l pacbio -t 16 -s pacbio.spec` (available as wgs-correction.tar.gz). For short high-identity sequences (< 100bp, only *S. cerevisiae* in our dataset) the parameters to the consensus module were modified to be `make-consensus -x removed.seq -w 5 -e 0.03`, as suggested by the AMOS documentation. A maximum of 100X of raw PacBio sequences was used for correction. Illumina-only assemblies were generated using the Celera Assembler (with the parameters `overlapper=ovl merSize=14 unitigger=bogart`) and SOAPdenovo v1.05 (with parameters `all -K 63`) followed by GapCloser (with default parameters) with only the best reported in the text. For *Melopsittacus undulates*, we also ran ALLPATHS-LG using the commands `PrepareAllPathsInputs.pl PHRED_64=True PLOIDY=2 and RunAllPathsLG THREADS=32 PRE=allpaths-lg DATA_SUBDIR=assembly RUN=myrun REFERENCE_NAME=.`. Hybrid assemblies were generated using Celera Assembler modified to accept sequences up to 30 000bp using the bogart unitigger (`overlapper=ovl merSize=14 unitigger=bogart`) (available as wgs-assembler.tar.gz). The Celera Assembler includes three unitigger options: utg, bog, and bogart. The utg unitigger is originally developed for Sanger sequences. BOG was developed to handle 454 pyrosequencing data [105]. The bogart unitigger (Walenz personal communications) has been developed to better handle high-coverage datasets, such as those generated by Illumina instruments while matching BOGs performance on pyrosequencing data. Thus we have focused our modifications/testing of long-read support within the Celera Assembler on bogart. Our correction pipeline (as well as bogart) will be distributed with the Celera Assembler starting with version 7.0.

B.3.2 Repeat region identification

To further evaluate correctness accuracy, we selected regions of the genome that appear repetitive and compared the correction error rates within repeat regions to error rates in the full genome. Repeat regions were identified by mapping Illumina sequences used for correction to the reference using bowtie 0.12.7 (`bowtie --all -p 16`) [83]. Any sequence with more than a single mapping was assumed to originate from a repeat. All genomic regions covered by a mapping of one of these multiply-placed read were considered repetitive. Any PBcR reads intersecting these regions were extracted from the full PBcR set.

B.3.3 Correction by coverage

Figure B.1, evaluates the ability of high-coverage to correct for sequencing errors. With sufficient coverage, even high error rates can be compensated for.

B.3.4 Validation of contigs

The contigs from each assembly were aligned to a reference. For SOAPdenovo, contigs were obtained by splitting scaffolds at each N. Statistics were tabulated using custom scripts using a fixed genome size (equal to the reference length when available) across all assemblies. For evaluating correctness, alignment statistics and mis-assemblies were tallied using the program `dnadiff` [123] from MUMmer v3.23 [82]. `dnadiff` operates by constructing local pairwise alignments between a reference and query genome using the Nucmer aligner. The aligned segments are then filtered to obtain a globally optimal mapping between the reference and query segments, while allowing for rearrangements, duplications, and inversions. This technique was later described in detail by Dubchack *et al.* as the SuperMap algorithm [25]. Conveniently, this method identifies both a one-to-one mapping of segments as well

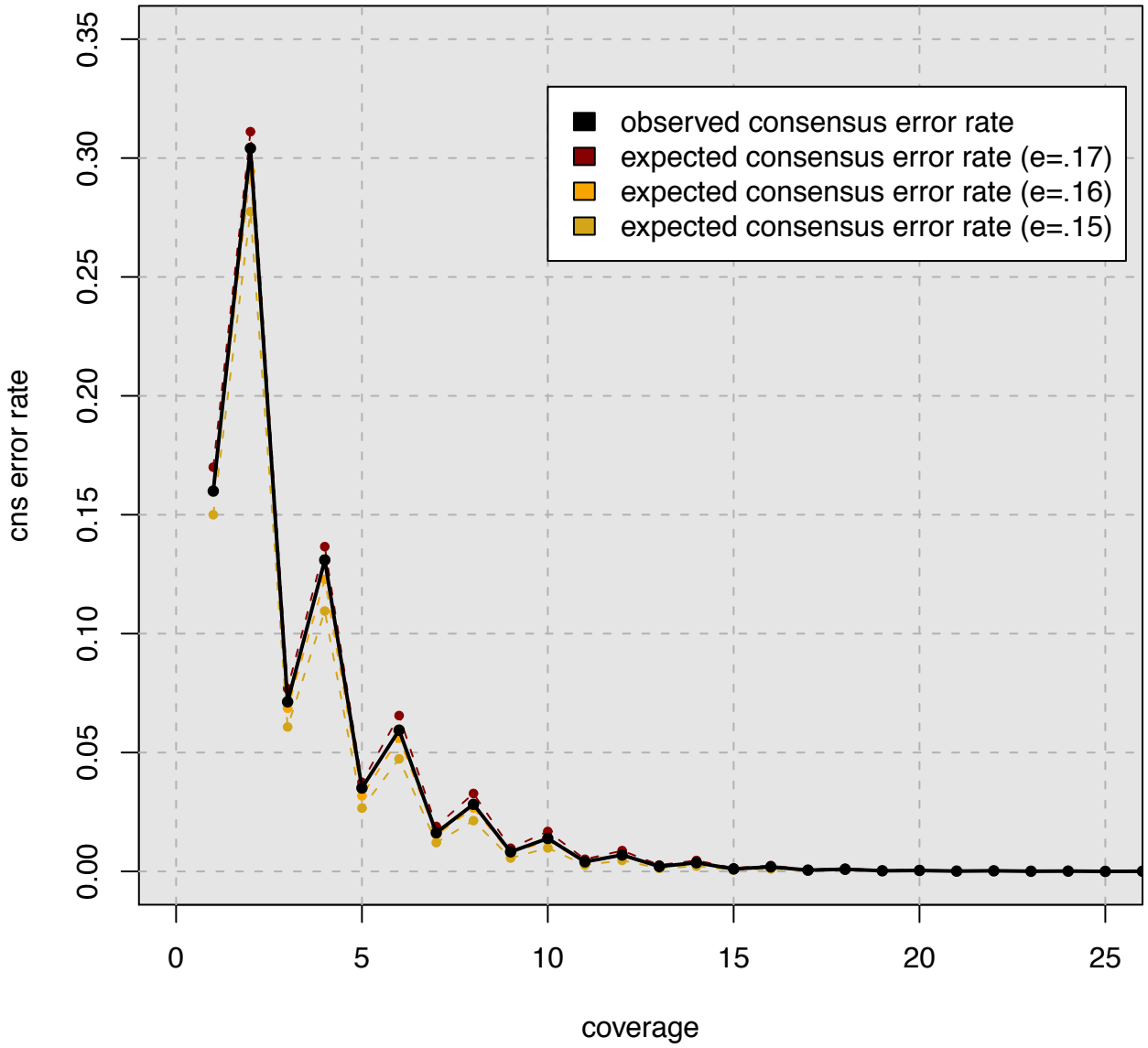


Figure B.1: **Coverage can overcome most random errors.** 1000bp reads for *E. coli* K12 were simulated with random errors and the resulting consensus accuracy was measured. Even with high errors, coverage over 10X is sufficient to generate an accurate consensus. The periodic fluctuation in consensus error rate is an artifact of the tie-breaking scheme used in the consensus simulation (even numbers of reads can have ties and odd cannot).

as any duplicated sequences. When applied to assembly mapping, it can be used to measure the quantity and types of common mis-assemblies.

To create the alignments contigs were aligned using `nucmer` with the options (`--maxmatch -l 30 -banded -D 5`). Combined with its default options, this in-

vocation requires a minimum exact-match anchor size of 30 bp, and a minimum combined anchor length of 65 bp per cluster. Clusters are further required to have no more than 90 bp separation or more than 5 inserted bases between any two adjacent anchors. Acceptable clusters are then used to seed banded Smith-Waterman alignments [152]. After running nucmer, alignments with less than 95% identity or more than 95% overlap with another alignment were discarded using `delta-filter`. `dnadiff` [123] was then executed on the remaining alignments with default parameters and correctness statistics were tabulated from its output. Average identity was computed on the one-to-one aligned segments, ignoring duplicated bases. To calculate a corrected N50, the resulting one-to-one alignment lengths were used. As alignments are broken at any alignment error, the alignment sizes correspond to the pieces of the assembly that are error-free. This correctness method has been previously used to evaluate assemblies for the GAGE project [137].

B.3.5 Zebra finch transcripts

The 15 275 zebra finch mRNA sequences *Taeniopygia guttata*, NCBI build 1, assembly version 3.2.4, genome accession ABQF00000000.1 were downloaded. They were mapped to all contigs in each of the three assemblies, regardless of length, using gmap [181] with the parameters `--cross-species`.

B.3.6 Paired-end satisfaction

The Celera Assembler generates a file named asm.posmap.mates specifying the status of each paired-end within the assembly. There is also an asm.posmap.frgctg listing each fragments location within the assembled contig. The output files were parsed to extract paired-ends where both sequence ends fell in one contig (denoted by a suffix of a or b in Celera Assembler) from the asm.posmap.frgctg file. Next, the status for each selected pair was extracted from the asm.posmap.mates file.

The possible statuses include good, badLong (above 3 standard deviations from the mean), badOuttie (incorrect orientation), badSame (incorrect orientation), badShort (below 3 standard deviations). All pairs not marked as good were considered bad for assembly correctness. A total of 3 242 006 paired-ends were marked good in the 454-only assembly versus 3 281 360 in the PBcR/454 hybrid, an increase of 39 354. Additionally, 1 806 paired-end were bad in the 454-only assembly, vs 1 688 in the PBcR/454 hybrid.

To test the assembly correctness using an independent technology, we mapped the BGI 10Kbp jumping library to the 454 and PBcR/454 assemblies using bowtie 0.12.7 with the command `bowtie --best -M 1 -p 16`.

Appendix C

Supplementary Data for Bambus 2: Scaffolding Metagenomes

C.1 Methods

C.1.1 Unitig generation

We used the Minimus assembler available as `bambus2_v1_0_BRANCH` from the AMOS package to generate unitigs with the command line `minimus <prefix>`. The Minimus assembler does not scale to large datasets, therefore, we used the Celera Assembler version 5.1 with default settings (`runCA.pl -p asm -d ca`) referred to as CA below for the Acid Mine drainage dataset. We used the Newbler assembler software version 2.3 – PreRelease – 9/14/2009 from 454 Life Sciences with default settings (`runAssembly <fastaFileName>`) for datasets containing pyrosequencing reads. When no paired-end data was available for pyrosequencing reads, the contig graph generated by Newbler was converted to Bambus 2 links with mean of 0 and weight equal to the number of reads joining the contigs. For analysis of Illumina sequencing data, we used SOAPdenovo contigs as input to Bambus 2. Reads were mapped to the contigs using `bowtie` version 0.12.7.

C.1.1.1 Bambus 2 parameters

All tests except the comparison to SOAPdenovo used Bambus 2 available as `bambus2_v1_0 BRANCH` from the AMOS package. Unitigs were generated as above and scaffolded by Bambus 2 using default parameters with the minimum edge-weight set to one (`-redundancy 1`). The *Brucella suis* 1330 dataset set the minimum edge-weight to two (`-redundancy 2`). The pipeline to generate results is

(1) `clk -b data.bnk` 2 (2) `Bundler -b data.bnk` (3) `MarkRepeats -b data.bnk -redundancy X -all  repeats` (4) `OrientContigs -b data.bnk -redundancy X -repeats repeats -all`. The analysis of Illumina data as well as GUT sequence used for validation used Bambus 2 available from the AMOS git repository as of 8/17/11. For SOAPdenovo comparison, the repeat redundancy was set to 50 and OrientContigs to redundancy 5 (`-redundancy 5`) due to the high coverage of the Illumina sequences. Due to the large dataset size, on MH0012, repeat detection was restricted to the connected component based approach. The latest Bambus 2 (available from git) includes an automated module to choose the appropriate redundancy threshold parameter.

C.1.1.2 Celera Assembler parameters

Celera Assembler version 5.1 was run both with default settings (`runCA.pl -p asm -d ca`) referred to as CA below and with the metagenomic settings (`runCA.pl -p asm -d ca-met utgErrorRate=120 ovlErrorRate=0.14 cnsErrorRate=0.14 cgwErrorRate=0.14 merSize=14 utgGenomeSize=2000000 utgBubblePopping=0 doFragmentCorrection=0`) referred to as CA-met.

C.1.1.3 Newbler parameters

Newbler software version 2.3 – PreRelease – 9/14/2009 from 454 Life Sciences was run with default settings (`runAssembly <fastaFileName>`)

C.1.1.4 SOAPdenovo parameters

SOAPdenovo software version 1.04 was used to assemble the MetaHit sample with parameters recommended by the MetaHit consortium [129]. The first 25bp of each read was mapped to contigs using bowtie[83] version 0.12.7 and converted to

AMOS using custom scripts.

C.2 Analysis

C.2.1 Annotation of repeats in known genomes

We used REPuter [81] version 3.0 with the parameters `-f -p -r -l 50 -best 50000` to identify repeats in each genome. Overlapping repeat regions were merged. The detected repeats for all genomes in a dataset are the gold standard. Finished reference genomes were used when available. For draft genomes, the entire set of assembled contigs was provided to REPuter. The repeat sequences were mapped onto the set of unitigs from an assembly. Those with high-identity (95% identity, 90% coverage) mappings to repeats were marked as repeat. Unitigs composed of only one read were not used in the analysis.

C.2.2 Validation of scaffolds

We used the MUMmer package, version 3.20. The contigs generated for the Acid Mine dataset were mapped to the set of references. The best match for each position of the query was saved (`delta-filter -q`). Each contig was assigned to the organism it matched. Any contig having a match to more than one organism was marked as chimeric.

C.2.3 *E. coli* mosaic sequence

We used the MUMmer package, version 3.20. The contigs generated for the CA, CA-met, and Bambus 2 (with motifs) were mapped to *E. coli* O157:H7. The best match for each position of the query was saved (`delta-filter -q`). We also mapped *E. coli* K12 to *E. coli* O157:H7 to obtain comparative structure. The mummerplot for each run was generated using `mummerplot --layout`.

C.2.4 Functional annotation of variation motifs

We used the blastx program from NCBI BLAST version 2.2.23 to annotate assemblies against the refseq protein database [3]. An e-value cutoff of 0.01 was used for datasets with simulated data and an e-value of 5 was used for the rest. All non-overlapping best hits (sorted by e-value) were saved. The COG database [158] was downloaded on May 8th, 2010. For each COG, we used an indicator variable set to true if at least one protein from the COG was present in the metagenomic data, and false otherwise. COGs were assigned to their functional category and the total number of COGs in each functional category was tabulated. A COG assigned to more than one category contributed to the count for all its categories. A COG with no functional category was counted as N/A. We measured the enrichment of a given functional category F with the hypergeometric test. The enrichment p-value is computed by $\sum_{i=k}^{m} \left(\frac{\binom{f}{i}\binom{|N|-f}{|M|-i}}{\binom{|N|}{|M|}} \right)$ where $|N|$ is the numbers of COGs in the assembly, $|M|$ is the number of COGs in the motifs, f is the number of COGs in the assembly from functional category F, and k is the number of COGs in the motifs from functional category F. The depletion p-value is computed by $\sum_{i=0}^{k} \left(\frac{\binom{f}{i}\binom{|N|-f}{|M|-i}}{\binom{|N|}{|M|}} \right)$. The computed hypergeometric p-values are Bonferroni corrected to account for multiple testing. A functional category with a p-value of less than 0.05 was considered significant.

C.2.5 Motif classification

We classified motifs as either in/del or substitution variants. For each motif, we evaluate all pairs of overlapping contigs. Motifs where at least one pair of contigs positions overlap by 80% of the smaller contigs length were considered as substitutions (since the two matching contigs represent alternate sequence). Otherwise, motifs were considered in/dels since there is a significant difference of sequence length within the motif.

C.2.6 SOAPdenovo scaffold comparison

We mapped the Bambus 2 motifs to SOAPdenovo scaffolds using MUMmer 3.20 with the command `nucmer -maxmatch`. Then we counted the number of Bambus 2 motifs with mappings to more than one scaffold, implying they cross SOAPdenovo scaffold ends. We detected 2 544 such motifs. Alternatively, we also evaluated coverage by matches. We counted matches where a SOAPdenovo scaffold contained an entire Bambus 2 motif as those with $> 95\%$ identity and the match coverage of the motif was $> 95\%$. There were only 80 such scaffolds. By contrast, looking at matches with $> 95\%$ identity and SOAPdenovo scaffold coverage of $> 95\%$ (but motif coverage $\leq 95\%$), we counted 1 512 of 2 763 motifs. We used NCBIs nucleotide web-based BLAST with default parameters against the nr database on the longest scaffold and picked the best matching reference. We used `dnadiff` from the MUMmer 3.20 package [82] to map the assembly scaffolds to the reference. The errors were counted by `dnadiff` (sum of Relocations, Translocations, and Inversions). The total number of scaffolds mapping to the reference after filtering (`delta-filter q`) were divided by the bases of the reference covered as reported by dnadiff.

C.3 Results

C.3.1 Scaffolding of isolate genomes

We use the Minimus assembler - a conservative unitigger to assemble *Brucella suis* 1330. This dataset was used to tune Bambus 2's repeat and motif detection. The dataset included 36 080 fragments and 16 850 mate pairs (33 700 mated reads) for a total coverage of 8X and is available from the NCBI Trace Archive (Project ID 320). The dataset was filtered to include only WGS-reads and to remove reads not matching the reference sequence (presumed contaminants). The reference comprises two circular chromosomes, accessions AE014291 and AE014292, of 2 107 792 bp and

1 207 381 bp respectively. We applied Bambus 2 and compared the results to the results generated with Celera Assembler.

We tested assemblies of *Rhodobacter sphaeroides* and *Staphylococcus aureus* from the GAGE publication [137]. All results, except Bambus 2 were taken from tables in the paper. Bambus 2 was run as specified in the appendix of GAGE [137], which we summarize below. An initial assembly was generated by Celera Assembler using the BOG unitigger [105]. Illumina jumping (mate-pair) libraries were input to CA as unpaired sequences. Both mate-pair sequences were reverse complemented to convert from outie to innie orientation. Finally, all pairing information and CA unitigs were input to Bambus 2 for scaffolding with the parameters of `-noPathRepeats` to MarkRepeats and `-redundancy 0 -maxOverlap 500` to OrientContigs. In the case of *Staphylococcus aureus*, the mate-pair sequences were too short to be input to CA directly. They were mapped to assembled unitigs using bowtie [83] and scaffolded by Bambus 2 as above.

C.3.2 Scaffolding of single-cell

Datasets were generated by the Velvet-SC authors in [17]. Three datasets were downloaded and analyzed: (1) *E. coli* K12 from clonal sequencing (2) *E. coli* K12 lane6 from an MDA sample, and (3) *S. aureus* US300 from lane 7.

Coverage plots were generated by mapping the sequencing reads to the reference genome (NC_000913) using bowtie version 0.12.7 with the commands `-p 16 --sam -q --best --fr --minins=180 --maxins=250 ecoli_normal.fastq` and `-p 16 --sam -q --best --fr --minins=180 --maxins=360 ecoli_mda_lane6.fastq`. Coverage was tabulated for each position. Histograms were generated using R with the command `hist(coverage[,2], breaks=100)` and plots were generated via the command `plot(coverage[,1], coverage[,2], typ="l", log="y")`.

The genomes were assembled using Velvet-SC with the parameters `-k 55`

`-cov_cutoff 300` (for *E. coli* K12) and `-k 55 -cov_cutoff 600` (for *S. aureus*) followed by Bambus 2 with the parameters `-redundancy 0 -maxOverlap 500 -all` (which were previously used in the GAGE study [137]. Correctness statistics were tabulated as reported in [137].

Appendix D

Supplementary Data for An Automated End-end Metagenomic Analysis Pipeline

D.1 Results

D.1.1 Validation of contigs

MUMmer [82] version 3.23 was used to align contigs to the references. Whenever scaffolds were available, contigs were extracted by splitting the scaffolds at Ns. The last 15bp of contig ends were trimmed before aligning. Alignments were filtered using `delta-filter -q` to keep only the best matches for each contig. Finally, statistics were tabulated based on the contig matches. A contig matching a single reference at over 97% identity over full length was considered good. A contig matching a reference over 90% of its length was considered a slight mis-assembly (but still contributed to computed reference coverage). Any contig with a match over $< 90\%$ of its length was considered a heavy mis-assembly. Any contig with multiple incompatible matches to a single reference was also considered a heavy mis-assembly. Finally, any contig with matches to two or more references was considered a chimera. None of the heavily mis-assembled contigs nor the chimeric contigs were included when tabulating reference coverage.

D.1.2 HMP mock samples

The samples are available from the NCBI as BioProject ID 48475. The mock samples were assembled using metAMOS (available in git as of 02/29/2012) with the commands `initPipeline` and `runPipeline` with default parameters. The same pa-

rameters were used for both mock Even and mock Staggered. SOAPdenovo version 1.05 was with the parameters `-D -d -R -M 3`. CA version 6.1 was run with metagenomic settings as `doMerBasedTrimming=1 doOverlapBasedTrimming=0 utgErrorRate=0.12 utgErrorLimit=6.5 cnsErrorRate=0.14 cgwErrorRate=0.14 ovlErrorRate=0.14 unitigger=bog utgBubblePopping=0 bogBadMateDepth=30 merSize=14 utgGenomeSize=10000000`. Velvet version 1.1.05 was run with $k = 63$. Meta-IDBA version 0.19 was run with default parameters.

D.1.3 HMP tongue dorsum

The sample was downloaded from the SRA using ID SRS077736. The assemblers were run within metAMOS using the same parameters as above. The motif was aligned using `blastn` to identity top-scoring genes.

D.1.4 Sexual dimorphism

We selected three males and three females randomly from the MetaHit project of the same age (59 years) and the same country (Denmark). We chose samples with the same enterotype (ET1) [5]. We also chose the samples to have approximately equal BMI (26.19 for males vs 24.12 for females). The chosen samples were MH0041, MH0045, and MH0055 for males and MH0002, MH0024, and MH0082 for females. metAMOS was run on all three samples of each sex using the longer paired libraries for each sample (ERR011181, ERR011189, ERR011209 for males and ERR011091, ERR011149, ERR011264 for females), giving a total of 72M sequences for males and 79M sequences for females.

To test for concordance between pre– and post–assembly annotations, we selected the order level classifications and compared the percentage classified at each order in the pre- and post–assembly male and female samples independently. We used R (version 2.11.1) and the command `cor.test(preAsm, postAsm)`. Both sam-

ples were over 99.9% correlated. By comparison, when the male and female samples were compared to each other they were only 94.28% correlated. To test for significance of the difference between samples we used the Fisher exact test on the order, family, and genus levels with the command `fisher.test(x)`.

D.1.5 Metaphyler runtime estimate

We ran two versions of Metaphyler, one based on the BLAST [3] analysis and a new version using MUMmer [82]. The new Metaphyler is significantly faster but at the expense of being unable to process sequences shorter than 60bp. The new Metaphyler ran in 720 CPU minutes versus 1 506 for the post-assembly analysis (which used the original Metaphyler). We estimated the runtime of the original Metaphyler on the pre-assembly dataset based on the published runtime of 70K reads in 8 hours [91], giving an estimated runtime of 8 000 CPU hours for 70M sequences. Comparing the 8 000 CPU hours to 1 506 gives the estimated speedup.

Bibliography

[1] S. Abubucker, N. Segata, J. Goll, A. Schubert, J. Izard, B. Cantarel, B. Rodriguez-Mueller, J. Zucker, M. Thiagarajan, B. Henrissat, O. White, S. Kelley, B. Methé, P. Schloss, D. Gevers, M. Mitreva, and C. Huttenhower. Metabolic reconstruction for metagenomic data and its application to the human microbiome. *PLoS Comp Bio*, In review, 2012.

[2] M. D. Adams, S. E. Celniker, R. A. Holt, C. A. Evans, J. D. Gocayne, P. G. Amanatides, S. E. Scherer, P. W. Li, R. A. Hoskins, R. F. Galle, R. A. George, S. E. Lewis, S. Richards, M. Ashburner, S. N. Henderson, G. G. Sutton, J. R. Wortman, M. D. Yandell, Q. Zhang, L. X. Chen, R. C. Brandon, Y. H. Rogers, R. G. Blazej, M. Champe, B. D. Pfeiffer, K. H. Wan, C. Doyle, E. G. Baxter, G. Helt, C. R. Nelson, G. L. Gabor, J. F. Abril, A. Agbayani, H. J. An, C. Andrews-Pfannkoch, D. Baldwin, R. M. Ballew, A. Basu, J. Baxendale, L. Bayraktaroglu, E. M. Beasley, K. Y. Beeson, P. V. Benos, B. P. Berman, D. Bhandari, S. Bolshakov, D. Borkova, M. R. Botchan, J. Bouck, P. Brokstein, P. Brottier, K. C. Burtis, D. A. Busam, H. Butler, E. Cadieu, A. Center, I. Chandra, J. M. Cherry, S. Cawley, C. Dahlke, L. B. Davenport, P. Davies, B. de Pablos, A. Delcher, Z. Deng, A. D. Mays, I. Dew, S. M. Dietz, K. Dodson, L. E. Doup, M. Downes, S. Dugan-Rocha, B. C. Dunkov, P. Dunn, K. J. Durbin, C. C. Evangelista, C. Ferraz, S. Ferriera, W. Fleischmann, C. Fosler, A. E. Gabrielian, N. S. Garg, W. M. Gelbart, K. Glasser, A. Glodek, F. Gong, J. H. Gorrell, Z. Gu, P. Guan, M. Harris, N. L. Harris, D. Harvey, T. J. Heiman, J. R. Hernandez, J. Houck, D. Hostin, K. A. Houston, T. J. Howland, M. H. Wei, C. Ibegwam, M. Jalali, F. Kalush, G. H. Karpen, Z. Ke, J. A. Kennison, K. A. Ketchum, B. E. Kimmel, C. D. Kodira, C. Kraft, S. Kravitz, D. Kulp, Z. Lai, P. Lasko, Y. Lei, A. A. Levitsky, J. Li, Z. Li, Y. Liang, X. Lin, X. Liu, B. Mattei, T. C. McIntosh, M. P. McLeod, D. McPherson, G. Merkulov, N. V. Milshina, C. Mobarry, J. Morris, A. Moshrefi, S. M. Mount, M. Moy, B. Murphy, L. Murphy, D. M. Muzny, D. L. Nelson, D. R. Nelson, K. A. Nelson, K. Nixon, D. R. Nusskern, J. M. Pacleb, M. Palazzolo, G. S. Pittman, S. Pan, J. Pollard, V. Puri, M. G. Reese, K. Reinert, K. Remington, R. D. Saunders, F. Scheeler, H. Shen, B. C. Shue, I. Sidén-Kiamos, M. Simpson, M. P. Skupski, T. Smith, E. Spier, A. C. Spradling, M. Stapleton, R. Strong, E. Sun, R. Svirskas, C. Tector, R. Turner, E. Venter, A. H. Wang, X. Wang, Z. Y. Wang, D. A. Wassarman, G. M. Weinstock, J. Weissenbach, S. M. Williams, WoodageT, K. C. Worley, D. Wu, S. Yang, Q. A. Yao, J. Ye, R. F. Yeh, J. S. Zaveri, M. Zhan, G. Zhang, Q. Zhao, L. Zheng, X. H. Zheng, F. N. Zhong, W. Zhong, X. Zhou, S. Zhu, X. Zhu, H. O. Smith, R. A. Gibbs, E. W. Myers, G. M. Rubin, and J. C. Venter. The genome sequence of drosophila melanogaster. *Science*, 287(5461):2185–2195, Mar 2000.

[3] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *J mol Biol*, 215(3):403–410, 1990.

[4] M. Arumugam, E. D. Harrington, K. U. Foerstner, J. Raes, and P. Bork. Smashcommunity: a metagenomic annotation and analysis tool. *Bioinformatics*, 26(23):2977–2978, 2010.

[5] M. Arumugam, J. Raes, E. Pelletier, D. Le Paslier, T. Yamada, D. R. Mende, G. R. Fernandes, J. Tap, T. Bruls, J.-M. Batto, M. Bertalan, N. Borruel, F. Casellas, L. Fernandez, L. Gautier, T. Hansen, M. Hattori, T. Hayashi, M. Kleerebezem, K. Kurokawa, M. Leclerc, F. Levenez, C. Manichanh, H. B. Nielsen, T. Nielsen, N. Pons, J. Poulain, J. Qin, T. Sicheritz-Ponten, S. Tims, D. Torrents, E. Ugarte, E. G. Zoetendal, J. Wang, F. Guarner, O. Pedersen, W. M. de Vos, S. Brunak, J. Dore, J. Weissenbach, S. D. Ehrlich, and P. Bork. Enterotypes of the human gut microbiome. *Nature*, 473(7346):174–180, 05 2011.

[6] S. Batzoglou. Algorithmic challenges in mammalian genome sequence assembly. *Encyclopedia of genomics, proteomics and bioinformatics. John Wiley and Sons*, 2005.

[7] S. Batzoglou, D. B. Jaffe, K. Stanley, J. Butler, S. Gnerre, E. Mauceli, B. Berger, J. P. Mesirov, and E. S. Lander. Arachne: a whole-genome shotgun assembler. *Genome research*, 12(1):177–189, Jan 2002.

[8] D. Bentley. Whole-genome re-sequencing. *Current Opinion in Genetics & Development*, 16(6):545–552, 2006.

[9] M. Borodovsky, R. Mills, J. Besemer, and A. Lomsadze. Prokaryotic gene prediction using GeneMark and GeneMark.hmm. *Curr Protoc Bioinformatics*, Chapter 4:Unit4.5, May 2003.

[10] A. Brady and S. Salzberg. Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models. *Nature Methods 2009*, 6:673–610, 2009.

[11] A. Brady and S. Salzberg. PhymmBL expanded: confidence scores, custom databases, parallelization and more. *Nature methods*, 8(5):367–367, 2011.

[12] U. Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[13] J. Butler, I. MacCallum, M. Kleber, I. A. Shlyakhter, M. K. Belmonte, E. S. Lander, C. Nusbaum, and D. B. Jaffe. ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome Research*, 18(5):810–820, 2008.

[14] J. Caporaso, J. Kuczynski, J. Stombaugh, K. Bittinger, F. Bushman, E. Costello, N. Fierer, A. Peña, J. Goodrich, J. Gordon, et al. QIIME allows

analysis of high-throughput community sequencing data. *Nature methods*, 7(5):335–336, 2010.

[15] M. J. Chaisson and P. A. Pevzner. Short read fragment assembly of bacterial genomes. *Genome Research*, 18(2):324–330, 2008.

[16] C. S. Chin, J. Sorenson, J. B. Harris, W. P. Robins, R. C. Charles, R. R. Jean-Charles, J. Bullard, D. R. Webster, A. Kasarskis, P. Peluso, E. E. Paxinos, Y. Yamaichi, S. B. Calderwood, J. J. Mekalanos, E. E. Schadt, and M. K. Waldor. The origin of the Haitian Cholera outbreak strain. *New England Journal of Medicine*, 364(1):33–42, 2011.

[17] H. Chitsaz, J. L. Yee-Greenbaum, G. Tesler, M.-J. Lombardo, C. L. Dupont, J. H. Badger, M. Novotny, D. B. Rusch, L. J. Fraser, N. A. Gormley, O. Schulz-Trieglaff, G. P. Smith, D. J. Evers, P. A. Pevzner, and R. S. Lasken. Efficient de novo assembly of single-cell bacterial genomes from short-read data sets. *Nature Biotech*, 29(10):915–921, 10 2011.

[18] J. Clemente, J. Jansson, and G. Valiente. Flexible taxonomic assignment of ambiguous sequencing reads. *BMC Bioinformatics*, 12(1):8, 2011.

[19] C. Condon, J. Philips, Z. Fu, C. Squires, and C. Squires. Comparison of the expression of the seven ribosomal RNA operons in *Escherichia coli*. *Embo Journal*, 11(11):4175–4185, 1992.

[20] L. Dagum and R. Menon. OpenMP: An Industry-Standard API for Shared-Memory Programming. *IEEE COMPUTATIONAL SCIENCE AND ENGINEERING*, pages 46–55, 1998.

[21] A. Dayarian, T. Michael, and A. Sengupta. SOPRa: Scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics*, 11(1):345, 2010.

[22] F. Dean, J. Nelson, T. Giesler, and R. Lasken. Rapid amplification of plasmid and phage DNA using phi29 DNA polymerase and multiply-primed rolling circle amplification. *Genome Research*, 11(6):1095–1099, 2001.

[23] P. H. Degnan and H. Ochman. Illumina-based analysis of microbial community diversity. *ISME J*, 6(1):183–194, 01 2012.

[24] J. C. Dohm, C. Lottaz, T. Borodina, and H. Himmelbauer. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Research*, 36(16):e105, Sep 2008.

[25] I. Dubchak, A. Poliakov, A. Kislyuk, and M. Brudno. Multiple whole-genome alignments without a reference organism. *Genome research*, 19(4):682–689, 2009.

[26] D. Earl, K. Bradnam, J. St. John, A. Darling, D. Lin, J. Fass, H. O. K. Yu, V. Buffalo, D. R. Zerbino, M. Diekhans, N. Nguyen, P. N. Ariyaratne, W.-K. Sung, Z. Ning, M. Haimel, J. T. Simpson, N. A. Fonseca, I. Birol, T. R. Docking, I. Y. Ho, D. S. Rokhsar, R. Chikhi, D. Lavenier, G. Chapuis, D. Naquin, N. Maillet, M. C. Schatz, D. R. Kelley, A. M. Phillippy, S. Koren, S.-P. Yang, W. Wu, W.-C. Chou, A. Srivastava, T. I. Shaw, J. G. Ruby, P. Skewes-Cox, M. Betegon, M. T. Dimon, V. Solovyev, I. Seledtsov, P. Kosarev, D. Vorobyev, R. Ramirez-Gonzalez, R. Leggett, D. MacLean, F. Xia, R. Luo, Z. Li, Y. Xie, B. Liu, S. Gnerre, I. MacCallum, D. Przybylski, F. J. Ribeiro, S. Yin, T. Sharpe, G. Hall, P. J. Kersey, R. Durbin, S. D. Jackman, J. A. Chapman, X. Huang, J. L. DeRisi, M. Caccamo, Y. Li, D. B. Jaffe, R. E. Green, D. Haussler, I. Korf, and B. Paten. Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome Research*, 21(12):2224–2241, 2011.

[27] S. R. Eddy. Accelerated profile HMM searches. *PLoS Comput Biol*, 7(10):e1002195, 10 2011.

[28] J. Eid, A. Fehr, J. Gray, K. Luong, J. Lyle, G. Otto, P. Peluso, D. Rank, P. Baybayan, B. Bettman, A. Bibillo, K. Bjornson, B. Chaudhuri, F. Christians, R. Cicero, S. Clark, R. Dalal, A. Dewinter, J. Dixon, M. Foquet, A. Gaertner, P. Hardenbol, C. Heiner, K. Hester, D. Holden, G. Kearns, X. Kong, R. Kuse, Y. Lacroix, S. Lin, P. Lundquist, C. Ma, P. Marks, M. Maxham, D. Murphy, I. Park, T. Pham, M. Phillips, J. Roy, R. Sebra, G. Shen, J. Sorenson, A. Tomaney, K. Travers, M. Trulson, J. Vieceli, J. Wegener, D. Wu, A. Yang, D. Zaccarin, P. Zhao, F. Zhong, J. Korlach, and S. Turner. Real-time DNA sequencing from single polymerase molecules. *Science*, 323(5910):133–138, Jan 2009.

[29] J. Eppley, G. Tyson, W. Getz, and J. Banfield. Strainer: software for analysis of population variation in community genomic datasets. *BMC bioinformatics*, 8(1):398, 2007.

[30] J. M. Eppley, G. W. Tyson, W. M. Getz, and J. F. Banfield. Genetic Exchange Across a Species Boundary in the Archaeal Genus Ferroplasma. *Genetics*, 177(1):407–416, 2007.

[31] B. Ewing and P. Green. Base-calling of automated sequencer traces using phred II error probabilities. *Genome research*, 8(3):186–194, Mar 1998.

[32] D. Fasulo, A. Halpern, I. Dew, and C. Mobarry. Efficiently detecting polymorphisms during the fragment assembly process. *Bioinformatics*, 18 Suppl 1:S294–S302, 2002.

[33] L. Feuk, A. R. Carson, and S. W. Scherer. Structural variation in the human genome. *Nat Rev Genet*, 7(2):85–97, 02 2006.

[34] S. R. Finkbeiner, A. F. Allred, P. I. Tarr, E. J. Klein, C. D. Kirkwood, and D. Wang. Metagenomic analysis of human diarrhea: Viral detection and discovery. *PLoS Pathog*, 4(2):e1000011, 02 2008.

[35] R. D. Fleischmann, M. D. Adams, O. White, R. A. Clayton, E. F. Kirkness, A. R. Kerlavage, C. J. Bult, J. F. Tomb, B. A. Dougherty, and J. M. Merrick. Whole-genome random sequencing and assembly of haemophilus influenzae rd. *Science*, 269(5223):496–512, Jul 1995.

[36] L. Frangeul, K. Nelson, C. Buchrieser, A. Danchin, P. Glaser, and F. Kunst. Cloning and assembly strategies in microbial genome projects. *Microbiology*, 145(Pt 10):2625–2634, 1999.

[37] C. M. Fraser, J. A. Eisen, K. E. Nelson, I. T. Paulsen, and S. L. Salzberg. The value of complete microbial genome sequencing (you get what you pay for). *J. Bacteriol.*, 184(23):6403–6405, 2002.

[38] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[39] L. Freeman. Centrality in social networks conceptual clarification. *Social networks*, 1(3):215–239, 1979.

[40] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software - Practice and Experience*, 30(11):1203–1233, 2000.

[41] S. Gao, N. Nagarajan, and W.-K. Sung. Opera: Reconstructing Optimal Genomic Scaffolds with High-Throughput Paired-End Sequences. *Lecture Notes in Computer Science*, 6577:437–451, 2011.

[42] M. Garey and D. Johnson. Computers and intractability: a guide to NP-completeness. *WH Freemanand Company, San Francisco, California*, 1979.

[43] X. Ge, Y. Li, X. Yang, H. Zhang, P. Zhou, Y. Zhang, and Z. Shi. Metagenomic Analysis of Viruses from the Bat Fecal Samples Reveals Many Novel Viruses in Insectivorous Bats in China. *J Virol*, Feb 2012.

[44] S. R. Gill, M. Pop, R. T. DeBoy, P. B. Eckburg, P. J. Turnbaugh, B. S. Samuel, J. I. Gordon, D. A. Relman, C. M. Fraser-Liggett, and K. E. Nelson. Metagenomic analysis of the human distal gut microbiome. *Science*, 312(5778):1355–1359, 2006.

[45] S. Gnerre, I. MacCallum, D. Przybylski, F. J. Ribeiro, J. N. Burton, B. J. Walker, T. Sharpe, G. Hall, T. P. Shea, S. Sykes, A. M. Berlin, D. Aird, M. Costello, R. Daza, L. Williams, R. Nicol, A. Gnirke, C. Nusbaum, E. S. Lander, and D. B. Jaffe. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *PNAS*, 2010.

[46] J. Goecks, A. Nekrutenko, J. Taylor, and The Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):R86, 2010.

[47] S. M. Goldberg, J. Johnson, D. Busam, T. Feldblyum, S. Ferriera, R. Friedman, A. Halpern, H. Khouri, S. A. Kravitz, F. M. Lauro, K. Li, Y. H. Rogers, R. Strausberg, G. Sutton, L. Tallon, T. Thomas, E. Venter, M. Frazier, and J. C. Venter. A sanger/pyrosequencing hybrid approach for the generation of high-quality draft assemblies of marine microbial genomes. *Proceedings of the National Academy of Sciences*, 103(30), 2006.

[48] J. Goll, D. B. Rusch, D. M. Tanenbaum, M. Thiagarajan, K. Li, B. A. Methe, and S. Yooseph. METAREP: JCVI metagenomics reports–an open source tool for high-performance comparative metagenomics. *Bioinformatics*, 26:2631–2632, Oct 2010.

[49] D. S. Goltsman, V. J. Denef, S. W. Singer, N. C. VerBerkmoes, M. Lefsrud, R. S. Mueller, G. J. Dick, C. L. Sun, K. E. Wheeler, A. Zemla, B. J. Baker, L. Hauser, M. Land, M. B. Shah, M. P. Thelen, R. L. Hettich, and J. F. Banfield. Community Genomic and Proteomic Analyses of Chemoautotrophic Iron-Oxidizing "Leptospirillum rubarum" (Group II) and "Leptospirillum ferrodiazotrophum" (Group III) Bacteria in Acid Mine Drainage Biofilms. *Appl. Environ. Microbiol.*, 75(13):4599–4615, 2009.

[50] L. Goodstadt. Ruffus: a lightweight python library for computational pipelines. *Bioinformatics*, 26(21):2778, 2010.

[51] D. Gordon, C. Abajian, and P. Green. Consed: a graphical tool for sequence finishing. *Genome Res*, 8(3):195–202, 1998.

[52] D. Gordon, C. Desmarais, and P. Green. Automated finishing with autofinish. *Genome Research*, 11(4):614–625, 2001.

[53] C. Han and P. Chain. Finishing repetitive regions automatically with dupfinisher. *Proceedings of 2006 International Conference on Bioinformatics and Computational Biology: 2006; Las Vegas, Nevada, USA*, 2006.

[54] C. Han, S. Spring, A. Lapidus, T. Rio, H. Tice, A. Copeland, J.-F. Cheng, S. Lucas, F. Chen, M. Nolan, D. Bruce, L. Goodwin, S. Pitluck, N. Ivanova, K. Mavrommatis, N. Mikhailova, A. Pati, A. Chen, K. Palaniappan, M. Land, L. Hauser, Y.-J. Chang, C. Jeffries, E. Saunders, O. Chertkov, T. Brettin, M. Goker, M. Rohde, J. Bristow, J. Eisen, V. Markowitz, P. Hugenholtz, N. Kyrpides, H.-P. Klenk, and J. Detter. Complete genome sequence of *Pedobacter heparinus* type strain (him 762-3t). *Standards in Genomic Sciences*, 1(1):54, 2009.

[55] O. Harismendy, P. C. Ng, R. L. Strausberg, X. Wang, T. B. Stockwell, K. Y. Beeson, N. J. Schork, S. S. Murray, E. J. Topol, S. Levy, and K. A. Frazer.

Evaluation of next generation sequencing platforms for population targeted sequencing studies. *Genome Biology*, 10(3):R32, Mar 2009.

[56] T. Hazen. Automatic alignment and error correction of human generated transcripts for long speech recordings. In *Ninth International Conference on Spoken Language Processing*, 2006.

[57] M. Hess, A. Sczyrba, R. Egan, T. W. Kim, H. Chokhawala, G. Schroth, S. Luo, D. S. Clark, F. Chen, T. Zhang, R. I. Mackie, L. A. Pennacchio, S. G. Tringe, A. Visel, T. Woyke, Z. Wang, and E. M. Rubin. Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science*, 331(6016):463–467, 2011.

[58] L. Hillier, W. Miller, E. Birney, W. Warren, R. Hardison, C. Ponting, P. Bork, D. Burt, M. Groenen, M. Delany, et al. Sequence and comparative analysis of the chicken genome provide unique perspectives on vertebrate evolution. *Nature*, 432(7018):695–716, 12 2004.

[59] X. Huang, J. Wang, S. Aluru, S. P. Yang, and L. Hillier. PCAP: A Whole-Genome Assembly Program. *Genome Research*, 13(9):2164–2170, 2003.

[60] S. M. Huse, J. A. Huber, H. G. Morrison, M. L. Sogin, and D. M. Welch. Accuracy and quality of massively parallel DNA pyrosequencing. *Genome biology*, 8(7):R143, 2007.

[61] D. Huson, K. Reinert, and E. Myers. The greedy path-merging algorithm for sequence assembly. In *Proceedings of the fifth annual international conference on Computational biology*, RECOMB '01, pages 157–163, New York, NY, USA, 2001. Association for Computing Machinery.

[62] R. Idury and M. Waterman. A new algorithm for DNA sequence assembly. *Journal of Computational Biology*, 2(2):291–306, 1995.

[63] J. P. A. Ioannidis, D. B. Allison, C. A. Ball, I. Coulibaly, X. Cui, A. C. Culhane, M. Falchi, C. Furlanello, L. Game, G. Jurman, J. Mangion, T. Mehta, M. Nitzberg, G. P. Page, E. Petretto, and V. van Noort. Repeatability of published microarray gene expression analyses. *Nat Genet*, 41(2):149–155, 02 2009.

[64] D. B. Jaffe, J. Butler, S. Gnerre, E. Mauceli, K. Lindblad-Toh, J. P. Mesirov, M. C. Zody, and E. S. Lander. Whole-genome sequence assembly for mammalian genomes: Arachne 2. *Genome research*, 13(1):91–96, Jan 2003.

[65] K. Jiang, D. Ediger, and D. Bader. Generalizing k-betweenness centrality using short paths and a parallel multithreaded implementation. In *Parallel Processing, 2009. ICPP'09. International Conference on*, pages 542–549. IEEE, 2009.

[66] J. Kececioglu and J. Ju. Separating repeats in DNA sequence assembly. In *Proceedings of the fifth annual international conference on Computational biology*, pages 176–183. ACM, 2001.

[67] J. Kececioglu and E. Myers. Combinatorial algorithms for DNA sequence assembly. *Algorithmica*, 13(1):7–51, 1995.

[68] D. Kelley, B. Liu, A. Delcher, M. Pop, and S. Salzberg. Gene prediction with Glimmer for metagenomic sequences augmented by classification and clustering. *Nucleic acids research 2011*, 10., 2011.

[69] D. Kelley, M. Schatz, and S. Salzberg. Quake: quality-aware detection and correction of sequencing errors. *Genome Biology*, 11(11):R116, 2010.

[70] S. W. Kembel, J. A. Eisen, K. S. Pollard, and J. L. Green. The phylogenetic diversity of metagenomes. *PLoS ONE*, 6(8):e23214, 08 2011.

[71] S. W. Kembel, E. Jones, J. Kline, D. Northcutt, J. Stenson, A. M. Womack, B. J. Bohannan, G. Z. Brown, and J. L. Green. Architectural design influences the diversity and structure of the built environment microbiome. *ISME J*, 2012.

[72] W. J. Kent. Blat–the blast-like alignment tool. *Genome Research*, 12(4):656–664, 2002.

[73] C. Kingsford, M. Schatz, and M. Pop. Assembly complexity of prokaryotic genomes using short reads. *BMC bioinformatics*, 11(1):21, 2010.

[74] J. E. Koenig, A. Spor, N. Scalfone, A. D. Fricker, J. Stombaugh, R. Knight, L. T. Angenent, and R. E. Ley. Succession of microbial consortia in the developing infant gut microbiome. *Proceedings of the National Academy of Sciences*, 108(Supplement 1):4578–4585, 2011.

[75] H. H. Kong, J. Oh, C. Deming, S. Conlan, E. A. Grice, M. Beatson, E. Nomicos, E. Polley, H. D. Komarow, N. C. S. Program, P. R. Murray, M. L. Turner, and J. A. Segre. Temporal shifts in the skin microbiome associated with atopic dermatitis disease flares and treatment. *Genome Research*, 2012.

[76] S. Koren, J. Miller, B. Walenz, and G. Sutton. An algorithm for automated closure during assembly. *BMC Bioinformatics*, 11(1):457, 2010.

[77] S. Koren, M. Schatz, B. Walenz, J. Martin, J. Howard, G. Ganapathy, Z. Wang, D. Rasko, W. McCombie, E. Jarvis, and A. Phillippy. Hybrid error correction and *de novo* assembly of single-molecule sequencing reads. *Nature Biotech*, In Review, 2012.

[78] S. Koren, T. Treangen, and M. Pop. Bambus 2: scaffolding metagenomes. *Bioinformatics*, 27(21):2964–2971, 2011.

[79] L. Krause, N. N. Diaz, A. Goesmann, S. Kelley, T. W. Nattkemper, F. Rohwer, R. A. Edwards, and J. Stoye. Phylogenetic classification of short environmental dna fragments. *Nucleic Acids Research*, 36(7):2230–2239, 2008.

[80] C.-H. Kuo, N. A. Moran, and H. Ochman. The consequences of genetic drift for bacterial genome complexity. *Genome Research*, 2009.

[81] S. Kurtz, J. V. Choudhuri, E. Ohlebusch, C. Schleiermacher, J. Stoye, and R. Giegerich. REPuter: the manifold applications of repeat analysis on a genomic scale. *Nucleic Acids Research*, 29(22):4633–4642, 2001.

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

[83] B. Langmead, C. Trapnell, M. Pop, and S. Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[84] J. Laserson, V. Jojic, and D. Koller. Genovo: de novo assembly for metagenomes. *Journal of Computational Biology*, 18(3):429–443, 2011.

[85] R. Lasken and T. Stockwell. Mechanism of chimera formation during the multiple displacement amplification reaction. *BMC Biotechnology*, 7(1):19, 2007.

[86] S. Levy, G. Sutton, P. C. Ng, L. Feuk, A. L. Halpern, B. P. Walenz, N. Axelrod, J. Huang, E. F. Kirkness, G. Denisov, Y. Lin, J. R. MacDonald, A. W. Pang, M. Shago, T. B. Stockwell, A. Tsiamouri, V. Bafna, V. Bansal, S. A. Kravitz, D. A. Busam, K. Y. Beeson, T. C. McIntosh, K. A. Remington, J. F. Abril, J. Gill, J. Borman, Y. H. Rogers, M. E. Frazier, S. W. Scherer, R. L. Strausberg, and J. C. Venter. The diploid genome sequence of an individual human. *PLoS Biol*, 5(10):e254, 09 2007.

[87] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup. The sequence alignment/map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[88] R. Li, W. Fan, G. Tian, H. Zhu, L. He, J. Cai, Q. Huang, Q. Cai, B. Li, Y. Bai, Z. Zhang, Y. Zhang, W. Wang, J. Li, F. Wei, H. Li, M. Jian, J. Li, Z. Zhang, R. Nielsen, D. Li, W. Gu, Z. Yang, Z. Xuan, O. A. Ryder, F. C.-C. Leung, Y. Zhou, J. Cao, X. Sun, Y. Fu, X. Fang, X. Guo, B. Wang, R. Hou, F. Shen, B. Mu, P. Ni, R. Lin, W. Qian, G. Wang, C. Yu, W. Nie, J. Wang, Z. Wu, H. Liang, J. Min, Q. Wu, S. Cheng, J. Ruan, M. Wang, Z. Shi, M. Wen, B. Liu, X. Ren, H. Zheng, D. Dong, K. Cook, G. Shan, H. Zhang, C. Kosiol, X. Xie, Z. Lu, H. Zheng, Y. Li, C. C. Steiner, T. T.-Y. Lam, S. Lin, Q. Zhang, G. Li, J. Tian, T. Gong, H. Liu, D. Zhang, L. Fang, C. Ye, J. Zhang, W. Hu, A. Xu,

Y. Ren, G. Zhang, M. W. Bruford, Q. Li, L. Ma, Y. Guo, N. An, Y. Hu, Y. Zheng, Y. Shi, Z. Li, Q. Liu, Y. Chen, J. Zhao, N. Qu, S. Zhao, F. Tian, X. Wang, H. Wang, L. Xu, X. Liu, T. Vinar, Y. Wang, T.-W. Lam, S.-M. Yiu, S. Liu, H. Zhang, D. Li, Y. Huang, X. Wang, G. Yang, Z. Jiang, J. Wang, N. Qin, L. Li, J. Li, L. Bolund, K. Kristiansen, G. K.-S. Wong, M. Olson, X. Zhang, S. Li, H. Yang, J. Wang, and J. Wang. The sequence and de novo assembly of the giant panda genome. *Nature*, 463(7279):311–317, Jan 2010.

[89] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, and J. Wang. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20(2):265–272, 2010.

[90] Y. Li, H. Zheng, R. Luo, H. Wu, H. Zhu, R. Li, H. Cao, B. Wu, S. Huang, H. Shao, H. Ma, F. Zhang, S. Feng, W. Zhang, H. Du, G. Tian, J. Li, X. Zhang, S. Li, L. Bolund, K. Kristiansen, A. J. de Smith, A. I. Blakemore, L. J. Coin, H. Yang, J. Wang, and J. Wang. Structural variation in two human genomes mapped at single-nucleotide resolution by whole genome de novo assembly. *Nature Biotech*, 29(8):723–730, 08 2011.

[91] B. Liu, T. Gibbons, M. Ghodsi, and M. Pop. Metaphyler: Taxonomic profiling for metagenomic sequences. In *Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on*, pages 95–100. IEEE, 2011.

[92] C. A. Lozupone and R. Knight. Species divergence and the measurement of microbial diversity. *FEMS Microbiology Reviews*, 32(4):557–578, 2008.

[93] A. V. Lukashin and M. Borodovsky. Genemark.hmm: New solutions for gene finding. *Nucleic Acids Research*, 26(4):1107–1115, 1998.

[94] C. Luo, D. Tsementzi, N. Kyrpides, and K. Konstantinidis. Individual genome assembly from complex community short-read metagenomic datasets. *The ISME Journal*, 2011.

[95] K. Madduri, D. Ediger, K. Jiang, D. Bader, and D. Chavarria-Miranda. A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets. In *Parallel and Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE, 2009.

[96] E. R. Mardis. The impact of next-generation sequencing technology on genetics. *Trends in genetics : TIG*, 24(3):133–141, Mar 2008.

[97] M. Margulies, M. Egholm, W. E. Altman, S. Attiya, J. S. Bader, L. A. Bemben, J. Berka, M. S. Braverman, Y. J. Chen, Z. Chen, S. B. Dewell, L. Du, J. M. Fierro, X. V. Gomes, B. C. Godwin, W. He, S. Helgesen, C. H. Ho, C. H. Ho, G. P. Irzyk, S. C. Jando, M. L. Alenquer, T. P. Jarvie, K. B. Jirage, J. B. Kim, J. R. Knight, J. R. Lanza, J. H. Leamon, S. M. Lefkowitz, M. Lei,

J. Li, K. L. Lohman, H. Lu, V. B. Makhijani, K. E. McDade, M. P. McKenna, E. W. Myers, E. Nickerson, J. R. Nobile, R. Plant, B. P. Puc, M. T. Ronan, G. T. Roth, G. J. Sarkis, J. F. Simons, J. W. Simpson, M. Srinivasan, K. R. Tartaro, A. Tomasz, K. A. Vogt, G. A. Volkmer, S. H. Wang, Y. Wang, M. P. Weiner, P. Yu, R. F. Begley, and J. M. Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437:376–380, 2005.

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

[99] A. C. McHardy, H. G. Martin, A. Tsirigos, P. Hugenholtz, and I. Rigoutsos. Accurate phylogenetic classification of variable-length DNA fragments. *Nature Methods*, 4(1):63–72, 01 2007.

[100] P. McKenna, C. Hoffmann, N. Minkah, P. P. Aye, A. Lackner, Z. Liu, C. A. Lozupone, M. Hamady, R. Knight, and F. D. Bushman. The macaque gut microbiome in health, lentiviral infection, and chronic enterocolitis. *PLoS Pathog.*, 4:e20, Feb 2008.

[101] P. Medvedev, K. Georgiou, G. Myers, and M. Brudno. Computability of models for sequence assembly. *Algorithms in Bioinformatics*, pages 289–301, 2007.

[102] P. Meinicke, K. Ahauer, and T. Linger. Mixture models for analysis of the taxonomic composition of metagenomes. *Bioinformatics 2011*, 27:1618–1624, 2011.

[103] F. Meyer, D. Paarmann, M. D'Souza, R. Olson, E. M. Glass, M. Kubal, T. Paczian, A. Rodriguez, R. Stevens, A. Wilke, J. Wilkening, and R. A. Edwards. The metagenomics RAST server - a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics*, 9:386, 2008.

[104] J. Miller, S. Koren, and G. Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–327, 2010.

[105] J. R. Miller, A. L. Delcher, S. Koren, E. Venter, B. P. Walenz, A. Brownley, J. Johnson, K. Li, C. Mobarry, and G. Sutton. Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics*, 24(24):2818–2824, 2008.

[106] J. L. Mokili, F. Rohwer, and B. E. Dutilh. Metagenomics and future perspectives in virus discovery. *Current Opinion in Virology*, 2(1):63–77, 2012.

[107] O. Morozova and M. A. Marra. Applications of next-generation sequencing technologies in functional genomics. *Genomics*, 92(5):255–264, Nov 2008.

[108] E. Myers. Toward simplifying and accurately formulating fragment assembly. *Journal of Computational Biology*, 2(2):275–290, 1995.

[109] E. W. Myers. The fragment assembly string graph. *Bioinformatics*, 21 Suppl 2:79–85, 2005.

[110] E. W. Myers, G. G. Sutton, A. L. Delcher, I. M. Dew, D. P. Fasulo, M. J. Flanigan, S. A. Kravitz, C. M. Mobarry, K. H. Reinert, K. A. Remington, E. L. Anson, R. A. Bolanos, H. H. Chou, C. M. Jordan, A. L. Halpern, S. Lonardi, E. M. Beasley, R. C. Brandon, L. Chen, P. J. Dunn, Z. Lai, Y. Liang, D. R. Nusskern, M. Zhan, Q. Zhang, X. Zheng, G. M. Rubin, M. D. Adams, and J. C. Venter. A whole-genome assembly of drosophila. *Science*, 287(5461):2196–2204, Mar 2000.

[111] N. Nagarajan and M. Pop. Parametric complexity of sequence assembly: Theory and applications to next generation sequencing. *Journal of Computational Biology*, 16(7):897–908, 2009.

[112] O. Nalbantoglu, S. Way, S. Hinrichs, and K. Sayood. Raiphy: Phylogenetic classification of metagenomics samples using iterative refinement of relative abundance index profiles. *BMC Bioinformatics*, 12(1):41, 2011.

[113] B. Niu, L. Fu, S. Sun, and W. Li. Artificial and natural duplicates in pyrosequencing reads of metagenomic data. *BMC Bioinformatics*, 11(1):187, 2010.

[114] B. Ondov, N. Bergman, and A. Phillippy. Interactive metagenomic visualization in a web browser. *BMC Bioinformatics*, 12(1):385, 2011.

[115] T. Otto, M. Sanders, M. Berriman, and C. Newbold. Iterative correction of reference nucleotides (iCORN) using second generation sequencing technology. *Bioinformatics*, 26(14):1704, 2010.

[116] K. R. Patil, P. Haider, P. B. Pope, P. J. Turnbaugh, M. Morrison, T. Scheffer, and A. C. McHardy. Taxonomic metagenome sequence assignment with structured output models. *Nature Methods*, 8(3):191–192, 03 2011.

[117] Y. Peng, H. C. M. Leung, S. M. Yiu, and F. Y. L. Chin. Meta-IDBA: a de novo assembler for metagenomic data. *Bioinformatics*, 27(13):i94–i101, 2011.

[118] N. T. Perna, G. Plunkett, V. Burland, B. Mau, J. D. Glasner, D. J. Rose, G. F. Mayhew, P. S. Evans, J. Gregor, H. A. Kirkpatrick, G. Posfai, J. Hackett, S. Klink, A. Boutin, Y. Shao, L. Miller, E. J. Grotbeck, N. W. Davis, A. Lim, E. T. Dimalanta, K. D. Potamousis, J. Apodaca, T. S. Anantharaman, J. Lin, G. Yen, D. C. Schwartz, R. A. Welch, and F. R. Blattner. Genome sequence of enterohaemorrhagic *Escherichia coli* O157:H7. *Nature*, 409(6819):529–533, 2001.

[119] E. Pettersson, J. Lundeberg, and A. Ahmadian. Generations of sequencing technologies. *Genomics*, 93(2):105–111, Feb 2009.

[120] P. A. Pevzner. 1-tuple dna sequencing: computer analysis. *Journal of biomolecular structure & dynamics*, 7(1):63–73, Aug 1989.

[121] P. A. Pevzner and H. Tang. Fragment assembly with double-barreled data. *Bioinformatics*, 17 Suppl 1:S225–S233, 2001.

[122] P. A. Pevzner, H. Tang, and M. S. Waterman. An Eulerian path approach to DNA fragment assembly. *PNAS*, 98(17):9748–9753, 2001.

[123] A. Phillippy, M. Schatz, and M. Pop. Genome assembly forensics: finding the elusive mis-assembly. *Genome Biology*, 9(3):R55, 2008.

[124] M. Pop. Shotgun sequence assembly. *Advances in Computers*, 60(1):193–248, 2004.

[125] M. Pop. Genome assembly reborn: recent computational challenges. *Briefings in bioinformatics*, 10(4):354–366, Jul 2009.

[126] M. Pop, D. S. Kosack, and S. L. Salzberg. Hierarchical Scaffolding With Bambus. *Genome Research*, 14(1):149–159, 2004.

[127] M. Pop, A. Phillippy, A. L. Delcher, and S. L. Salzberg. Comparative genome assembly. *Briefings in Bioinformatics*, 5(3):237–248, 2004.

[128] M. Pop and S. L. Salzberg. Bioinformatics challenges of new sequencing technology. *Trends in genetics : TIG*, 24(3):142–149, Mar 2008.

[129] J. Qin, R. Li, J. Raes, M. Arumugam, K. S. Burgdorf, C. Manichanh, T. Nielsen, N. Pons, F. Levenez, T. Yamada, D. R. Mende, J. Li, J. Xu, S. Li, D. Li, J. Cao, B. Wang, H. Liang, H. Zheng, Y. Xie, J. Tap, P. Lepage, M. Bertalan, J. M. Batto, T. Hansen, D. Le Paslier, A. Linneberg, H. B. Nielsen, E. Pelletier, P. Renault, T. Sicheritz-Ponten, K. Turner, H. Zhu, C. Yu, S. Li, M. Jian, Y. Zhou, Y. Li, X. Zhang, S. Li, N. Qin, H. Yang, J. Wang, S. Brunak, J. Dore, F. Guarner, K. Kristiansen, O. Pedersen, J. Parkhill, J. Weissenbach, P. Bork, S. D. Ehrlich, J. Wang, M. Antolin, F. Artiguenave, H. Blottiere, N. Borruel, T. Bruls, F. Casellas, C. Chervaux, A. Cultrone, C. Delorme, G. Denariaz, R. Dervyn, M. Forte, C. Friss, M. van de Guchte, E. Guedon, F. Haimet, A. Jamet, C. Juste, G. Kaci, M. Kleerebezem, J. Knol, M. Kristensen, S. Layec, K. Le Roux, M. Leclerc, E. Maguin, R. M. Minardi, R. Oozeer, M. Rescigno, N. Sanchez, S. Tims, T. Torrejon, E. Varela, W. de Vos, Y. Winogradsky, and E. Zoetendal. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285):59–65, 03 2010.

[130] D. A. Rasko, D. R. Webster, J. W. Sahl, A. Bashir, N. Boisen, F. Scheutz, E. E. Paxinos, R. Sebra, C. S. Chin, D. Iliopoulos, A. Klammer, P. Peluso, L. Lee, A. O. Kislyuk, J. Bullard, A. Kasarskis, S. Wang, J. Eid, D. Rank, J. C. Redman, S. R. Steyert, J. Frimodt-M?ller, C. Struve, A. M. Petersen,

K. A. Krogfelt, J. P. Nataro, E. E. Schadt, and M. K. Waldor. Origins of the *E. coli* strain causing an outbreak of hemolytic–uremic syndrome in Germany. *New England Journal of Medicine*, 365(8):709–717, 2011.

[131] T. Rausch, S. Koren, G. Denisov, D. Weese, A.-K. Emde, A. Döring, and K. Reinert. A consistency-based consensus algorithm for de novo and reference-guided sequence assembly of short reads. *Bioinformatics*, 25(9):1118–1124, 2009.

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

[133] D. C. Richter, F. Ott, A. F. Auch, R. Schmid, and D. H. Huson. MetaSim: A sequencing simulator for genomics and metagenomics. *PLoS ONE*, 3(10):e3373, 10 2008.

[134] S. Rodrigue, A. C. Materna, S. C. Timberlake, M. C. Blackburn, R. R. Malmstrom, E. J. Alm, and S. W. Chisholm. Unlocking short read sequencing for metagenomics. *PLoS ONE*, 5(7):e11840, 07 2010.

[135] J. M. Rothberg, W. Hinz, T. M. Rearick, J. Schultz, W. Mileski, M. Davey, J. H. Leamon, K. Johnson, M. J. Milgrew, M. Edwards, J. Hoon, J. F. Simons, D. Marran, J. W. Myers, J. F. Davidson, A. Branting, J. R. Nobile, B. P. Puc, D. Light, T. A. Clark, M. Huber, J. T. Branciforte, I. B. Stoner, S. E. Cawley, M. Lyons, Y. Fu, N. Homer, M. Sedova, X. Miao, B. Reed, J. Sabina, E. Feierstein, M. Schorn, M. Alanjary, E. Dimalanta, D. Dressman, R. Kasinskas, T. Sokolsky, J. A. Fidanza, E. Namsaraev, K. J. McKernan, A. Williams, G. T. Roth, and J. Bustillo. An integrated semiconductor device enabling non-optical genome sequencing. *Nature*, 475(7356):348–352, 07 2011.

[136] D. B. Rusch, A. L. Halpern, G. Sutton, K. B. Heidelberg, S. Williamson, S. Yooseph, D. Wu, J. A. Eisen, J. M. Hoffman, K. Remington, K. Beeson, B. Tran, H. Smith, H. Baden-Tillson, C. Stewart, J. Thorpe, J. Freeman, C. Andrews-Pfannkoch, J. E. Venter, K. Li, S. Kravitz, J. F. Heidelberg, T. Utterback, Y. H. Rogers, L. I. Falcon, V. Souza, G. Bonilla-Rosso, L. E. Eguiarte, D. M. Karl, S. Sathyendranath, T. Platt, E. Bermingham, V. Gallardo, G. Tamayo-Castillo, M. R. Ferrari, R. L. Strausberg, K. Nealson, R. Friedman, M. Frazier, and J. C. Venter. The Sorcerer II Global Ocean Sampling expedition: northwest Atlantic through eastern tropical Pacific. *PLoS Biol*, 5(3):e77, 03 2007.

[137] S. L. Salzberg, A. M. Phillippy, A. V. Zimin, D. Puiu, T. Magoc, S. Koren, T. Treangen, M. C. Schatz, A. L. Delcher, M. Roberts, G. Marcais, M. Pop, and J. A. Yorke. GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Research*, 2011.

[138] F. Sanger, A. R. Coulson, B. G. Barrell, A. J. Smith, and B. A. Roe. Cloning in single-stranded bacteriophage as an aid to rapid DNA sequencing. *Journal of molecular biology*, 143(2):161–178, Oct 1980.

[139] F. Sanger, S. Nicklen, and A. Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977.

[140] E. E. Schadt, S. Turner, and A. Kasarskis. A window into third-generation sequencing. *Human Molecular Genetics*, 19(R2):R227–R240, 2010.

[141] M. Schatz, A. Delcher, and S. Salzberg. Assembly of large genomes using second-generation sequencing. *Genome Research*, 20(9):1165–1173, 2010.

[142] M. C. Schatz, A. M. Phillippy, D. D. Sommer, A. L. Delcher, D. Puiu, G. Narzisi, S. L. Salzberg, and M. Pop. Hawkeye and AMOS: visualizing and assessing the quality of genome assemblies. *Briefings in Bioinformatics*, 2011.

[143] P. D. Schloss and J. Handelsman. Introducing SONS, a tool for operational taxonomic unit-based comparisons of microbial community memberships and structures. *Appl. Environ. Microbiol.*, 72:6773–6779, Oct 2006.

[144] P. D. Schloss and J. Handelsman. Toward a census of bacteria in soil. *PLoS Comput Biol*, 2(7):e92, 07 2006.

[145] P. D. Schloss, S. L. Westcott, T. Ryabin, J. R. Hall, M. Hartmann, E. B. Hollister, R. A. Lesniewski, B. B. Oakley, D. H. Parks, C. J. Robinson, J. W. Sahl, B. Stres, G. G. Thallinger, D. J. Van Horn, and C. F. Weber. Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and Environmental Microbiology*, 75(23):7537–7541, December 1, 2009.

[146] J. Sebat, B. Lakshmi, D. Malhotra, J. Troge, C. Lese-Martin, T. Walsh, B. Yamrom, S. Yoon, A. Krasnitz, J. Kendall, A. Leotta, D. Pai, R. Zhang, Y. H. Lee, J. Hicks, S. J. Spence, A. T. Lee, K. Puura, T. Lehtimaki, D. Ledbetter, P. K. Gregersen, J. Bregman, J. S. Sutcliffe, V. Jobanputra, W. Chung, D. Warburton, M. C. King, D. Skuse, D. H. Geschwind, T. C. Gilliam, K. Ye, and M. Wigler. Strong association of de novo copy number mutations with autism. *Science*, 316(5823):445–449, 2007.

[147] P. Shannon et al. Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. *Genome Research*, 13(11):2498–2504, 2003.

[148] I. Sharon, A. Pati, V. Markowitz, and R. Pinter. A statistical framework for the functional analysis of metagenomes. *Research in Computational Molecular Biology*, 5541:496–511, 2009.

[149] M. Shoaib, S. Baconnais, U. Mechold, E. Le Cam, M. Lipinski, and V. Ogryzko. Multiple displacement amplification for complex mixtures of DNA fragments. *BMC Genomics*, 9(1):415, 2008.

[150] A. F. Siegel, G. van den Engh, L. Hood, B. Trask, and J. C. Roach. Modeling the feasibility of whole genome shotgun sequencing using a pairwise end strategy. *Genomics*, 68(3):237–246, Sep 2000.

[151] S. L. Simmons, G. Dibartolo, V. J. Denef, D. S. Goltsman, M. P. Thelen, and J. F. Banfield. Population genomic analysis of strain variation in Leptospirillum group II bacteria involved in acid mine drainage formation. *PLoS Biol*, 6(7):e177, 07 2008.

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

[153] M. Smoot, K. Ono, J. Ruscheinski, P. Wang, and T. Ideker. Cytoscape 2.8: New Features for Data Integration and Network Visualization. *Bioinformatics*, 2010.

[154] D. Sommer, A. Delcher, S. Salzberg, and M. Pop. Minimus: a fast, lightweight genome assembler. *BMC Bioinformatics*, 8(1):64, 2007.

[155] R. Staden. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Research*, 6(7):2601–2610, Jun 1979.

[156] R. L. Strausberg, S. Levy, and Y.-H. Rogers. Emerging DNA sequencing technologies for human genomic medicine. *Drug Discovery Today*, 13(13-14):569–577, Jul 2008.

[157] G. Sutton and I. Dew. Shotgun Fragment Assembly. *Systems Biology: Genomics*, 3:79, 2007.

[158] R. L. Tatusov, M. Y. Galperin, D. A. Natale, and E. V. Koonin. The COG database: a tool for genome-scale analysis of protein functions and evolution. *Nucleic Acids Research*, 28(1):33–36, 2000.

[159] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22(22):4673–4680, 1994.

[160] C. Trapnell, B. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. Van Baren, S. Salzberg, B. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotech*, 28(5):511–515, 2010.

[161] T. Treangen*, S. Koren*, D. Sommer, B. Liu, I. Astrovskaya, A. Darling, and M. Pop. metAMOS: A modular and open source metagenomic assembly and analysis pipeline. *Genome Biology*, In Prep, 2012.

[162] T. Treangen, D. Sommer, F. Angly, S. Koren, and M. Pop. Next generation sequence assembly with AMOS. *Current Protocols in Bioinformatics*, 11:1–11, 2011.

[163] P. J. Turnbaugh, M. Hamady, T. Yatsunenko, B. L. Cantarel, A. Duncan, R. E. Ley, M. L. Sogin, W. J. Jones, B. A. Roe, J. P. Affourtit, M. Egholm, B. Henrissat, A. C. Heath, R. Knight, and J. I. Gordon. A core gut microbiome in obese and lean twins. *Nature*, 457(7228):480–484, 2008.

[164] P. J. Turnbaugh, R. E. Ley, M. Hamady, C. M. Fraser-Liggett, R. Knight, and J. I. Gordon. The human microbiome project. *Nature*, 449(7164):804–810, 10 2007.

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

[166] P. A. Vaishampayan, J. V. Kuehl, J. L. Froula, J. L. Morgan, H. Ochman, and M. P. Francino. Comparative metagenomics and population dynamics of the gut microbiota in mother and infant. *Genome Biology and Evolution*, 2:53–66, 2010.

[167] T. Varin, C. Lovejoy, A. D. Jungblut, W. F. Vincent, and J. Corbeil. Metagenomic analysis of stress genes in microbial mat communities from Antarctica and the High Arctic. *Appl. Environ. Microbiol.*, 78:549–559, Jan 2012.

[168] J. C. Venter, K. Remington, J. F. Heidelberg, A. L. Halpern, D. Rusch, J. A. Eisen, D. Wu, I. Paulsen, K. E. Nelson, W. Nelson, D. E. Fouts, S. Levy, A. H. Knap, M. W. Lomas, K. Nealson, O. White, J. Peterson, J. Hoffman, R. Parsons, H. Baden-Tillson, C. Pfannkoch, Y. H. Rogers, and H. O. Smith. Environmental genome shotgun sequencing of the Sargasso Sea. *Science*, 304(5667):66–74, 2004.

[169] F. Vezzi, G. Narzisi, and B. Mishra. Feature-by-Feature – evaluating *de Novo* sequence assembly. *PLoS ONE*, 7(2):e31002, 02 2012.

[170] M. Vignuzzi, J. K. Stone, J. J. Arnold, C. E. Cameron, and R. Andino. Quasispecies diversity determines pathogenesis through cooperative interactions in a viral population. *Nature*, 439(7074):344–348, 2005.

[171] J. P. Vinson, D. B. Jaffe, K. O'Neill, E. K. Karlsson, N. Stange-Thomann, S. Anderson, J. P. Mesirov, N. Satoh, Y. Satou, C. Nusbaum, B. Birren, J. E. Galagan, and E. S. Lander. Assembly of polymorphic genomes: algorithms and application to *Ciona savignyi*. *Genome Research*, 15(8):1127–1135, 2005.

[172] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

[173] F. Warnecke, P. Luginbuhl, N. Ivanova, M. Ghassemian, T. H. Richardson, J. T. Stege, M. Cayouette, A. C. McHardy, G. Djordjevic, N. Aboushadi, R. Sorek, S. G. Tringe, M. Podar, H. G. Martin, V. Kunin, D. Dalevi, J. Madejska, E. Kirton, D. Platt, E. Szeto, A. Salamov, K. Barry, N. Mikhailova, N. C. Kyrpides, E. G. Matson, E. A. Ottesen, X. Zhang, M. Hernandez, C. Murillo, L. G. Acosta, I. Rigoutsos, G. Tamayo, B. D. Green, C. Chang, E. M. Rubin, E. J. Mathur, D. E. Robertson, P. Hugenholtz, and J. R. Leadbetter. Metagenomic and functional analysis of hindgut microbiota of a wood-feeding higher termite. *Nature*, 450:560–565, Nov 2007.

[174] W. C. Warren, D. F. Clayton, H. Ellegren, A. P. Arnold, L. W. Hillier, A. Kunstner, S. Searle, S. White, A. J. Vilella, S. Fairley, A. Heger, L. Kong, C. P. Ponting, E. D. Jarvis, C. V. Mello, P. Minx, P. Lovell, T. A. Velho, M. Ferris, C. N. Balakrishnan, S. Sinha, C. Blatti, S. E. London, Y. Li, Y. C. Lin, J. George, J. Sweedler, B. Southey, P. Gunaratne, M. Watson, K. Nam, N. Backstrom, L. Smeds, B. Nabholz, Y. Itoh, O. Whitney, A. R. Pfenning, J. Howard, M. Volker, B. M. Skinner, D. K. Griffin, L. Ye, W. M. McLaren, P. Flicek, V. Quesada, G. Velasco, C. Lopez-Otin, X. S. Puente, T. Olender, D. Lancet, A. F. Smit, R. Hubley, M. K. Konkel, J. A. Walker, M. A. Batzer, W. Gu, D. D. Pollock, L. Chen, Z. Cheng, E. E. Eichler, J. Stapley, J. Slate, R. Ekblom, T. Birkhead, T. Burke, D. Burt, C. Scharff, I. Adam, H. Richard, M. Sultan, A. Soldatov, H. Lehrach, S. V. Edwards, S. P. Yang, X. Li, T. Graves, L. Fulton, J. Nelson, A. Chinwalla, S. Hou, E. R. Mardis, and R. K. Wilson. The genome of a songbird. *Nature*, 464:757–762, 2010.

[175] R. H. Waterston, K. Lindblad-Toh, E. Birney, J. Rogers, J. F. Abril, P. Agarwal, R. Agarwala, R. Ainscough, M. Alexandersson, P. An, S. E. Antonarakis, J. Attwood, R. Baertsch, J. Bailey, K. Barlow, S. Beck, E. Berry, B. Birren, T. Bloom, P. Bork, M. Botcherby, N. Bray, M. R. Brent, D. G. Brown, S. D. Brown, C. Bult, J. Burton, J. Butler, R. D. Campbell, P. Carninci, S. Cawley, F. Chiaromonte, A. T. Chinwalla, D. M. Church, M. Clamp, C. Clee, F. S. Collins, L. L. Cook, R. R. Copley, A. Coulson, O. Couronne, J. Cuff, V. Curwen, T. Cutts, M. Daly, R. David, J. Davies, K. D. Delehaunty, J. Deri, E. T. Dermitzakis, C. Dewey, N. J. Dickens, M. Diekhans, S. Dodge, I. Dubchak, D. M. Dunn, S. R. Eddy, L. Elnitski, R. D. Emes, P. Eswara, E. Eyras, A. Felsenfeld, G. A. Fewell, P. Flicek, K. Foley, W. N. Frankel, L. A. Fulton, R. S. Fulton, T. S. Furey, D. Gage, R. A. Gibbs, G. Glusman, S. Gnerre, N. Goldman, L. Goodstadt, D. Grafham, T. A. Graves, E. D. Green, S. Gregory, R. Guigo, M. Guyer, R. C. Hardison, D. Haussler, Y. Hayashizaki, L. W. Hillier, A. Hinrichs, W. Hlavina, T. Holzer, F. Hsu, A. Hua, T. Hubbard, A. Hunt, I. Jackson, D. B. Jaffe, L. S. Johnson, M. Jones, T. A. Jones, A. Joy, M. Kamal, E. K. Karlsson, D. Karolchik, A. Kasprzyk, J. Kawai, E. Keibler, C. Kells, W. J. Kent, A. Kirby, D. L. Kolbe, I. Korf, R. S. Kucherlapati,

E. J. Kulbokas, D. Kulp, T. Landers, J. P. Leger, S. Leonard, I. Letunic, R. Levine, J. Li, M. Li, C. Lloyd, S. Lucas, B. Ma, D. R. Maglott, E. R. Mardis, L. Matthews, E. Mauceli, J. H. Mayer, M. McCarthy, W. R. McCombie, S. McLaren, K. McLay, J. D. McPherson, J. Meldrim, B. Meredith, J. P. Mesirov, W. Miller, T. L. Miner, E. Mongin, K. T. Montgomery, M. Morgan, R. Mott, J. C. Mullikin, D. M. Muzny, W. E. Nash, J. O. Nelson, M. N. Nhan, R. Nicol, Z. Ning, C. Nusbaum, M. J. O'Connor, Y. Okazaki, K. Oliver, E. Overton-Larty, L. Pachter, G. Parra, K. H. Pepin, J. Peterson, P. Pevzner, R. Plumb, C. S. Pohl, A. Poliakov, T. C. Ponce, C. P. Ponting, S. Potter, M. Quail, A. Reymond, B. A. Roe, K. M. Roskin, E. M. Rubin, A. G. Rust, R. Santos, V. Sapojnikov, B. Schultz, J. Schultz, M. S. Schwartz, S. Schwartz, C. Scott, S. Seaman, S. Searle, T. Sharpe, A. Sheridan, R. Shownkeen, S. Sims, J. B. Singer, G. Slater, A. Smit, D. R. Smith, B. Spencer, A. Stabenau, N. Stange-Thomann, C. Sugnet, M. Suyama, G. Tesler, J. Thompson, D. Torrents, E. Trevaskis, J. Tromp, C. Ucla, A. Ureta-Vidal, J. P. Vinson, A. C. Von Niederhausern, C. M. Wade, M. Wall, R. J. Weber, R. B. Weiss, M. C. Wendl, A. P. West, K. Wetterstrand, R. Wheeler, S. Whelan, J. Wierzbowski, D. Willey, S. Williams, R. K. Wilson, E. Winter, K. C. Worley, D. Wyman, S. Yang, S. P. Yang, E. M. Zdobnov, M. C. Zody, and E. S. Lander. Initial sequencing and comparative analysis of the mouse genome. *Nature*, 420:520–562, 2002.

[176] J. Wetzel, C. Kingsford, and M. Pop. Assessing the benefits of using mate-pairs to resolve repeats in de novo short-read prokaryotic assemblies. *BMC Bioinformatics*, 12(1):95, 2011.

[177] J. R. White, N. Nagarajan, and M. Pop. Statistical methods for detecting differentially abundant features in clinical metagenomic samples. *PLoS Comput. Biol.*, 5:e1000352, Apr 2009.

[178] N. Whiteford, N. Haslam, G. Weber, A. Prügel-Bennett, J. W. Essex, P. L. Roach, M. Bradley, and C. Neylon. An analysis of the feasibility of short read sequencing. *Nucleic Acids Research*, 33(19):e171, Nov 2005.

[179] D. Wu, P. Hugenholtz, K. Mavromatis, R. Pukall, E. Dalin, N. N. Ivanova, V. Kunin, L. Goodwin, M. Wu, B. J. Tindall, S. D. Hooper, A. Pati, A. Lykidis, S. Spring, I. J. Anderson, P. D/'haeseleer, A. Zemla, M. Singer, A. Lapidus, M. Nolan, A. Copeland, C. Han, F. Chen, J.-F. Cheng, S. Lucas, C. Kerfeld, E. Lang, S. Gronow, P. Chain, D. Bruce, E. M. Rubin, N. C. Kyrpides, H.-P. Klenk, and J. A. Eisen. A phylogeny-driven genomic encyclopaedia of bacteria and archaea. *Nature*, 462(7276):1056–1060, 12 2009.

[180] D. Wu, M. Wu, A. Halpern, D. Rusch, S. Yooseph, M. Frazier, J. Venter, and J. Eisen. Stalking the fourth domain in metagenomic data: Searching for discovering, and interpreting novel, deep branches in marker gene phylogenetic trees. *PLoS ONE 2011*, 6:12, 2011.

[181] T. D. Wu and C. K. Watanabe. GMAP: a genomic mapping and alignment program for mRNA and EST sequences. *Bioinformatics*, 21(9):1859–1875, 2005.

[182] S. Yooseph et al. The Sorcerer II global ocean sampling expedition: Expanding the universe of protein families. *PLoS Biol*, 5(3):e16, 03 2007.

[183] D. R. Zerbino and E. Birney. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821–829, 2008.

[184] D. R. Zerbino, G. K. McEwen, E. H. Margulies, and E. Birney. Pebble and rock band: Heuristic resolution of repeats and scaffolding in the velvet short-read de Novo assembler. *PLoS ONE*, 4(12):e8407, 12 2009.