COPY NUMBER ESTIMATION FOR HIGH-THROUGHPUT SHORT READ SHOTGUN SEQUENCING *DE NOVO* WHOLE-GENOME ASSEMBLY CONTIGS

by

Yee Fay Lim

B.A., The University of Chicago, 2008

M.S., The University of Chicago, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Bioinformatics)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

April 2021

© Yee Fay Lim, 2021

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, a thesis entitled:

Copy number estimation for high-throughput short read shotgun sequencing *de novo* whole-genome assembly contigs

submitted by Yee Fay Lim in partial fulfillment of the requirements for

the degree of Master of Science

in Bioinformatics

Examining Committee:

Dr. İnanç Birol, Professor, Department of Medical Genetics, UBC
Supervisor

Dr. Alexandre Bouchard-Côté, Department of Statistics, UBC
Supervisory Committee Member

Dr. Gabriela V. Cohen Freue, Department of Statistics, UBC
Supervisory Committee Member

Abstract

High-throughput short shotgun sequencing reads, also known as second-generation sequencing (SGS) reads, continue to be prevalent for *de novo* whole-genome assembly, whether alone or in combination with long-range information. Knowledge of contig multiplicity (copy number) is acknowledged to improve assembly correctness, contiguity, and coverage for SGS reads. Despite that, a principled, general solution for contig copy number estimation in *de novo* whole-genome SGS assembly has been unavailable. In the literature, the problem is generally unaddressed or given heuristic treatment.

In this work, we introduce a novel, versatile statistically informed contig copy number estimator, based on mixture models, for high-throughput short read shotgun sequencing *de novo* whole-genome assembly. In particular, this tool targets de Bruijn graph assembly, the dominant paradigm for *de novo* whole-genome SGS assembly. We show that it performs reliably at resolving multiplicities up to low repeat copy numbers; it is also robust over a range of genome characteristics, sequencing coverage levels, and assembly settings. Moreover, it is far more versatile than the closest existing alternative tools and usually outperforms them, often by a wide margin. At the same time, somewhat reduced though still robust performance in a limited set of experiments using real sequencing data suggests fundamental limitations to its usage of only length and read coverage data; incorporating other types of information, e.g. GC content, may be necessary to improve performance. Our code is publicly available at https://github.com/bcgsc/wgs-copynum-est; we hope this effort will provide a useful reference for similar future work.

Lay Summary

High-throughput short sequencing reads are still widely used for reference-free whole-genome assembly. Information on the multiplicity, or copy number, of partially assembled sequences i.e. contigs improves final assembly quality. Despite that, a principled, general solution for contig copy number estimation in this class of assembly tasks has been unavailable.

We introduce a novel, versatile statistically informed contig copy number estimator for the setting described, targeting de Bruijn graph assembly, the dominant paradigm in this sphere, in particular. It shows reliable accuracy at resolving multiplicities up to a low maximum, over a range of genome, sequencing, and assembly conditions. Moreover, it is far more versatile than the closest existing alternative tools and often outperforms them by a wide margin. Nonetheless, there remains room for improvement, and we hope this effort proves useful for future work.

Preface

The project idea was conceived by my supervisor, Dr. İnanç Birol. The research design was largely mine, produced in consultation with Dr. Birol. I carried out all implementation tasks, including software programming and data analyses, with helpful and at times crucial input from Lauren Coombe, Dr. Shaun Jackman, and Dr. Birol.

Table of Contents

List of Tables

List of Figures

List of Supplementary Material

List of Abbreviations

AIC	Akaike Information Criterion
bp	Base pair
DBG	de Bruijn graph
DNA	Deoxyribonucleic acid
EM	Expectation-maximisation
GHz	Gigahertz
HTS	High-throughput sequencing
MPS	Massively parallel sequencing
NGS	Next-generation sequencing
NLLS	Non-linear least squares
OLC	Overlap-layout-consensus
RAM	Random-access memory
SGS	Second-generation sequencing
SMRT	Single-molecule real-time
SNP	Single-nucleotide polymorphism
TB	Terabyte
WGA	Whole-genome assembly
WGS	Whole-genome shotgun (sequencing)

Acknowledgements

I would like to thank my supervisor, Dr. İnanç Birol, for his patience and guidance during my studies.

I would also like to thank my thesis committee members, Prof. Alexandre Bouchard-Côté and Prof. Gabriela Cohen Freue for their contributions to the progress of my thesis. In particular, I appreciate Dr. Birol's and their highlighting the importance of communication in research.

Thank you also to Prof. Martin Hirst for chairing my thesis defence.

As well, I am grateful to various members of the Bioinformatics Technology Lab at the Genome Sciences Centre, whether or not their contributions made a direct impact on my thesis.

Thanks goes also to Sharon Ruschkowski, for being ever helpful with any matter related to the bioinformatics graduate program.

Finally, I would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for funding my studies.

Chapter 1: Introduction

1.1 Preliminaries

For most of the history of DNA sequencing, expense and difficulty rendered whole-genome sequencing and assembly attainable only for large, highly funded organisations and projects. However, in the last 15 to 20 years, massive advances in automation and speed, and reductions in cost of DNA sequencing have sparked a burst of mutually reinforcing growth in the accessibility of whole-genome sequencing and assembly, application domains, and bioinformatics tools for assembly and downstream analyses[1–3]. Whole-genome assembly (WGA), in particular *de novo* WGA i.e. reconstruction of an individual genome up to chromosome length without consulting previously resolved sequence[4], is a maturing and active field of research with well-established paradigms and a variety of tools addressing a range of needs. Nevertheless, genome assembly is an unsolved problem; due to the sheer difficulty of the task as well as the difficulty of finding realistic mathematical representations for it, assemblers still rely on heuristics and other ad hoc techniques instead of rigorous algorithms with provable performance guarantees[1]. Thus, there remains ample room at the time of writing for new approaches to continuing challenges in WGA. One of those, namely a statistical method for copy number estimation of contigs assembled from high-throughput sequencing data, shall be the subject of this thesis.

To set the stage for a more technical treatment of the subject matter, we shall start with a brief history of genome sequencing from the perspective of its usage in *de novo* WGA.

1.1.1 A brief history of genome assembly

Modern genome sequencing can be said to have begun with the introduction in 1977, by Frederick Sanger and colleagues, of a method for DNA sequencing with chain-terminating inhibitors[5], now commonly known as Sanger sequencing. This method became the most commonly used DNA sequencing technology for several years, spurring increasing automation and forming the basis of the first commercial DNA sequencing machines[6,7]; the data produced contains relatively long reads, typically between 300 and 1000 bps in length[8], at low read depth (note that "coverage" is often used instead of "depth"), i.e. each genomic locus is likely to be sampled in a relatively small number of reads.

During the "first generation" of genome sequencing represented by Sanger's method, the shotgun process, still prevalent today, was first developed and used to assemble long contiguous genomic sequences from shorter reads. The process randomly breaks a target molecule; the resulting fragments are sampled and sequenced to obtain reads[9,10]. In the case of whole-genome shotgun (WGS) sequencing, the target molecules consist of the chromosomes making up a genome. When a target is oversampled, the resulting reads overlap; they can then be computationally ordered and assembled[4]. The broad applicability of WGS was demonstrated by the 1995 completion of the 1.8-Mbp (million base pair) *Haemophilus influenzae* genome by WGS sequencing[11] and a number of subsequent projects[12]. Milestones were reached when almost all of the 120-Mbp euchromatic portion of the *Drosophila melanogaster* genome[13], and (in the Human Genome Project or HGP) a 2.91-Gbp consensus sequence of the euchromatic portion of the human genome[12], were determined through WGS sequencing supported by other techniques.

Concurrent with the spread and progress of first-generation sequencing technology, a luminescent method for measuring pyrophosphate synthesis was introduced[14]; its application to DNA sequencing via a technique called pyrosequencing was pioneered over the next decade[15,16] and licensed to the company 454 Life Sciences, where it was deployed in the first major commercially successful "next-generation sequencing" (NGS) machines[7]. These machines coupled pyrosequencing with massively parallelised sequencing reactions, producing up to a million reads ~200-500 bps long in each run, which represented an orders of magnitude increase in sequencing yield[7,17,18], and higher read depth relative to Sanger data. Massively parallel platforms from Solexa (now Illumina) output even shorter reads of 20 to 40 bps[19].

The massively parallel output of relatively short shotgun-generated reads is a shared, defining characteristic of several sequencing techniques that emerged over the next decade, which have come to be seen as second-generation sequencing (SGS). These techniques are also known by various other names, i.e. the aforementioned NGS, massively parallel sequencing (MPS), and high-throughput sequencing (HTS)[7,18,20]. Since the release of Illumina's Genome Analyzer II sequencer in 2006, competition between SGS technology manufacturers drove tremendous gains in output and reductions in cost, with raw per-base cost plummeting by four orders of magnitude between 2007 and 2012[21,22]. As a result, SGS had almost completely supplanted Sanger sequencing a decade after the HGP was completed in 2001, and Illumina has achieved a near-monopoly on the SGS market. At present, SGS reads are typically a few hundred bps long with low error rates (below 1% across Illumina platforms, consisting mostly of substitutions[18,20]), and sequencers produce an output of up to 1Tbps, or billions of reads, per run[7,20,22].

Alongside the rise to ubiquity of NGS, yet another class of sequencing technologies, sometimes called the third generation, has been advancing rapidly[7,22]. Characterised by long-range single-molecule resolution, these technologies do not require amplification in library preparation, and now routinely produce reads of average length around 10kb, in a range that can exceed 1Mb[23]. They consist of two main approaches, single-molecule real-time (SMRT) sequencing from PacBio[24] and nanopore-based sequencing from Oxford Nanopore Technologies[25]. Distinct from these true long-read platforms, synthetic long reads are created through library preparation protocols that associate NGS short reads derived from a single larger molecule with the same barcode; these are currently available on platforms from two vendors, Illumina and 10X Genomics[26]. Another NGS-compatible advance has been the creation of very long-range mate pair-like data from Hi-C and related chromatin crosslinking protocols[23,27]. Lastly, new optical mapping technology from BioNano Genomics uses nicking enzymes to create high-resolution sequence motif physical maps that can be *de novo* assembled into scaffolds to complement assembled genomic sequence[23,28].

In principle, if assembled using effectively tailored approaches, single-molecule data offer alternatives or solutions to the drawbacks of NGS technology for *de novo* WGA, including amplification artefacts from library preparation, and difficulty in characterising long repeats and large structural variation[23]. For example, high-quality long-read-only assemblies were created of microbial[29], maize[28], and human[30,31] genomes. However, in practice at the time of writing, the relatively high error rates (e.g. around 10% for PacBio reads[22]), right-skewed length distribution[23], and per-base cost of single-molecule reads make them impractical for many purposes unless used in conjunction with a short-read base assembly[32]: they require costly error-

correction procedures[33], can be very computationally expensive to assemble[23,34,35], or require prohibitively high depth to overcome the error rates and increase availability of the longest, most useful reads[23]—the aforementioned example assemblies used 65x to 103x SMRT read depth or, in one case, 22x and 24x SMRT with 80x optical mapping depth. Moreover, short reads, particularly Illumina reads, are widely used and likely to remain so for some time due to their high accuracy and low cost, especially for large-cohort studies and price-sensitive personalised medicine applications[32,36]. Not least, a NGS workflow can be leveraged to include linked reads and produce a high-quality assembly at modest extra cost[23,37,38]. Thus, NGS short-read *de novo* WGA is likely to remain highly relevant for some time.

1.1.2 An overview of *de novo* whole-genome shotgun sequence assembly

Of course, raw genomic sequence reads do us no good without tools, such as assemblers, that turn them into useful information. For much of the history of DNA sequencing—i.e. before the revolution in accessibility and applicability of the SGS era—genome sequencing and assembly was its primary purpose[22]; at the same time, its inaccessibility translated into a low level of activity in the field of assembly. The proliferation of SGS technologies and applications renewed interest in assembly, driving the rapid development of assemblers able to adapt to and leverage the characteristics of data produced by new sequencers.

This history is visible in the taxonomy of *de novo* WGS assemblers. At the most basic level, they belong to two paradigms, each corresponding to a distinct underlying search strategy: greedy and graph-based[39]. The earliest assemblers used greedy algorithms in response to the assumed computational intractability of the earliest formalisation of shotgun assembly as equivalent to the

shortest common superstring (SCS) problem for a set of sequences[40]; some well-known examples areTIGR[41] and phrap[42]. However, the difficulty of incorporating longer-range information into the inherently local assembly process of greedy assemblers is a major handicap. Thus, they have been superseded by graph-based assemblers[1], which in turn consist of two major subtypes: overlap-layout-consensus (OLC), and de Bruijn graph (DBG)[1,39].

Regardless of paradigm, all sequence assemblers operate on the general assumption that a sufficiently high degree of similarity between two sequences indicates that they are contiguous (or even coincident) in the genome used to generate the assembly dataset[1]. A graph-based assembler constructs a graph representing sequenced data to provide a basis for assembly decisions, with each node representing a sequence, and directed edges, each representing a suffix-to-prefix overlap between the nodes it connects.

The simplest definition of an OLC graph contains a node for every read, and an edge between every pair of reads with an overlap exceeding minimum thresholds in length and sequence identity[43]. This allows it to fully exploit information present in reads and supports error tolerance—characteristics well-suited for assembling longer, more inaccurate reads in lower-depth datasets, e.g. those from Sanger sequencing[1,4]. Unsurprisingly, the OLC paradigm was popularised by the Celera assembler, which helped produce the *Drosophila* genome[13,43] from Sanger reads. However, concerns about the inefficiency of computing overlaps between large numbers of reads have limited its application to SGS data, especially for larger genomes, though it is regaining ground with the emergence of long-read third-generation sequencing. Two well-known exceptions are the assembler Edena[44], which was published with results from bacterial

genomes, and SGA[45], which uses efficient string indexing data structures to enable assembly by a variant of OLC, the string graph, that simplifies the overlap graph by removing redundant information (transitive edges)[1].

In contrast, the DBG paradigm took off with the success of SGS. In the context of sequence assembly, a de Bruijn graph consists of all distinct substrings of some specified length, k, extracted from reads in a sequencing dataset, each uniquely represented by a node, and a directed edge between every pair of nodes with an exact k-1 bp suffix-to-prefix overlap, e.g. from ACGT to CGTA. First proposed as the "Eulerian approach" (i.e. representation as a search for an Eulerian path, a path traversing each edge once, in a graph) in relation to sequencing by hybridisation in 1989[46], it was shown to be applicable to Sanger read assembly in 1995[47]. The first DBG assembler, EULER, was introduced in 2001, with pre-assembly error correction to mitigate the drastic effects of base-calling error on its performance[48]. With the time and space required to build a DBG being linear in the size of the sequence underlying a dataset, as opposed to the quadratic increase with input size for building an OLC graph, DBGs scale well with the high depth of SGS datasets. As SGS became widely adopted over the next decade, several DBG-based assemblers were developed and improved to keep up with the characteristics of new sequencer outputs. Examples include EULER-USR (one of a few adaptations of EULER)[19], Velvet[49,50], ALLPATHS and ALLPATHS-LG[51,52], ABySS[53], SOAPdenovo[54,55], SPAdes (using variants on the standard DBG)[56], and DISCOVAR[57]. Moreover, the generally very low error rates of SGS support DBG reliance on exact overlaps. With advances in long-read sequencing, DBG assembly has also been leveraged as the basis of hybrid workflows incorporating long-read

data, e.g. in ABySS 2.0[36] and Supernova[38] (using linked reads), as well as Unicycler[32] and a recent unnamed diploid assembler[58] utilising SPAdes (using single-molecule reads).

The active research on genome assembly reflects the fundamental difficulty of the problem (at least with the technologies available to date), as revealed by the pioneering work of Esko Ukkonen et al. in the 1980s[59]. He demonstrated that sequence assembly, including genome assembly, ranges from trivial, to computationally intractable, to impossible[60] (i.e. information in sequencing data is insufficient to identify the correct genome reconstruction from an exponential number of equally good alternatives). This is certainly true of shotgun short-read assembly, even in hybrid workflows; of special interest here are some continuing challenges in the context of de Bruijn graph assembly of SGS data and how they can be addressed by the application of a statistical method for copy number estimation of partially assembled sequences. The rest of this thesis presents a technical treatment of its subject, starting in the next section with an introduction to DBG *de novo* WGS assembly of SGS data, followed by a survey of the issues of interest and existing approaches to addressing them.

1.2 Introduction to de Bruijn graph assembly

So far, DBG assembly has been presented as a monolithic, relatively unvarying entity; the reality is, of course, more complicated. That said, familiarity with a single, representative workflow shall suffice for an appreciation of the main subject matter of this thesis. Thus, while the exact terminology, graph structures, and workflow used vary considerably across assemblers, just one set of definitions (underlined, in bold) and descriptions, summarised and abstracted from the literature, shall serve as the conceptual basis for the rest of this material.

The k-mer de Bruijn graph, the foundational data structure of DBG assembly, is as defined earlier. The main stages of assembly are as follows:

1.2.1 k-mer de Bruijn graph construction

The canonical first step in the DBG assembly workflow. In practice, it is often preceded by an error correction step intended to fix or discard k-mers or reads containing sequencing errors, especially substitution errors[19,51,52,54,55,57,61]; alternatively, it can be accompanied by filtering out low-depth k-mers, i.e. those appearing in a number of reads below a specified threshold[36,38,53]. The construction is done as one might expect: given (as is often the case) a uniform input read length of some length L, extract all $L - k + 1$ k-mers present in each read, creating nodes to represent those newly encountered, and directed edges as implied by the definition given earlier. Thus, each node and each edge is unique; a k-mer is represented once regardless of how much it occurs in the genome, a key advantage in space complexity. Note that each node can have at most four incoming edges (corresponding to the letters of the DNA alphabet, A, C, G, and T), and similarly, at most four outgoing edges. Figure 1.1 gives a toy example of a k-mer DBG.

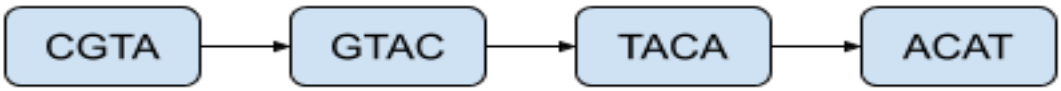

Figure 1.1 k-mer graph of the sequence "CGTACAT", with $k = 4$.

1.2.2 Contig building

Intuitively, nodes are merged along linear paths in the graph; the resulting string for each path is called a contig. For example, if a path consists of nodes representing "ACG", "CGT", and

"GTA", they would be merged to give a contig representing "ACGTA". More formally, each linear path has a source and a sink node, with all intervening nodes having both in-degree and out-degree 1; each node can belong to exactly one path. Thus, each path ends at a dead end, or at a branching point (at or adjacent to a node with total degree over 2). This step results in the contig graph, in which the k-mer nodes underlying each contig have been merged into a single node representing the contig, and each edge represents an overlap between two contigs. Figure 1.2 illustrates with an example.

Note that the contig graph contains no non-branching paths longer than a single node.

1.2.3 Scaffolding

This step is enabled by the ubiquity of paired-end sequencing in SGS (as well as the availability of other forms of long-range information), in which each read is located on one end of a sequence fragment, with a partner on the other end sequenced in the opposite orientation. The distance between the paired reads is drawn from a distribution and only approximately known[1].

In this step, long-range information, e.g. paired-end reads, mate pairs (paired reads providing the same type of information as paired-end reads at larger distances), linked reads, or single-molecule reads, are aligned or associated to the contig or k-mer graph to assess the degree of long-range support for graph connection and path candidates. Each contig is then merged or linked with other contigs judged to be connected, possibly with intervening gaps. This merging/linking process continues until a dead end is encountered, or a point is reached where no sufficiently supported path extension alternative is available, or an end-to-end path in the contig

graph is found between two contigs classified as single-copy (usually by heuristic criteria). The resulting sequence (which may contain gaps) is called a scaffold. This process is carried out on all contigs possible, taking care to avoid redundancy. Multiple types of long-range information may be used, often in successive stages in increasing order of distance.

Scaffolding encompasses repeat resolution, gap filling, and gap size estimation. The set of scaffolds obtained at the end of this stage represent the most contiguous reconstruction possible of a genome from a particular dataset, with a particular configuration of the assembler used. This can be considered the final output of assembly.

k-mer graph construction (1) and contig building (2) are often separated by graph simplification steps, in lieu of [36,49,50,53,56] or in addition to[54,55] pre-assembly error correction. Those steps typically consist of spur erosion (also called tip removal), bubble collapsing, and sometimes removal of low-coverage sequences; the next section shall cover them in more detail.

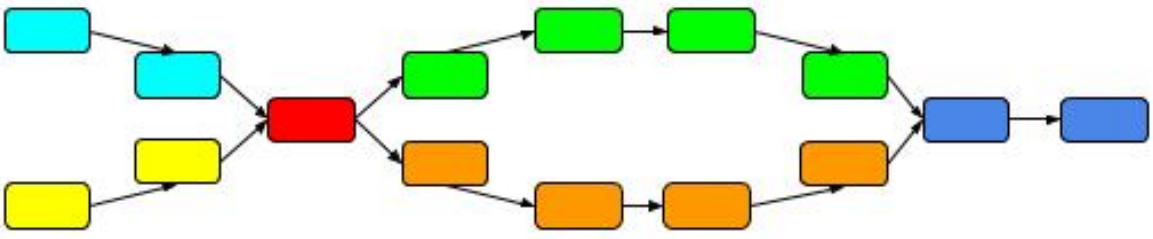

Figure 1.2(a) k-mer graph (k-mers omitted): Linear paths, each corresponding to a contig, colour coded.

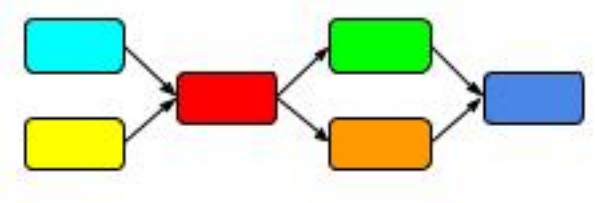

Figure 1.2(b) Contig graph: Linear path nodes merged into contigs; each node represents a contig.

1.3 Challenges in de Bruijn graph assembly of short reads

The goal of *de novo* genome assembly is to reconstruct a target genome to the highest possible degree of correctness, contiguity, and coverage (i.e. completeness). In particular, correctness comprises both base-level and structural accuracy, contiguity the lengths of reconstructed segments, and coverage the fraction of the genome reconstructed. Structural correctness and coverage additionally encompass assembly size accuracy, which in turn affects genome size estimation insofar as it is used for that purpose.

With respect to many organisms of interest, this task is fundamentally limited by the fact that the chromosomes constituting their genomes are longer than any sequencing read length possible to date. This is true of short-read assembly to an even greater degree, and for even more genomes. Furthermore, sequencing reads are subject to random error, i.e. mainly base call errors in the case of Illumina platforms, which effectively monopolise the present SGS market. Thus, the target genome is over-sampled to maximise genome coverage (with the degree of oversampling referred to as sequencing depth), and to minimise the impact of error through redundancy of correct base calls, though this strategy is complicated somewhat by chance and by the compositional bias of sequencing technologies against regions with more extreme levels (both low and high) of GC content[62,63].

1.3.1 Definition: Repeats

Crucially, given any particular level of sequencing depth, many genomes (i.e. those of nontrivial size and complexity) also possess characteristics that further confound sequence assembly, namely heterozygosity and repeats. For our purposes, let repeats first be defined as follows (using substring to denote a contiguous sequence of characters):

Given a set of sequences $S = \{ s_i \mid i \in \{ 1, 2, \ldots, n \} \text{ for some } n \in \mathbb{N}, \text{len}(s_i) \geq k \}$ from the alphabet $\{ A, C, G, T \}$, where $\text{len}(s_i)$ represents the length of s_i,

let $U = \{ u \mid u \text{ substring of } s_i \in S, i \in \{ 1, 2, \ldots, n \}, \text{len}(u) \geq k \}$, and

$\text{occur}(u)$ represent the multiplicity, i.e. number of occurrences, in S of a substring u.

$R = \{ u \mid u \in U, \exists\, v \in U, i \in \mathbb{N}$ such that u, v both substrings of s_i and $\text{occur}(u) > \text{occur}(v) \}$

R is the set of repeats of length at least k in S. A repeat is any $r \in$ R.

1.3.2 Effects of heterozygosity and repeats

Using this definition, a genome can be represented by S, with chromosomes as its members, and whether a sequence is a repeat relies on the existence of some sequence(s) that is connected to it and has lower multiplicity. A repeat (flanked by lower-multiplicity, i.e. lower copy number, sequences) induces path nonlinearity, and thus ambiguity, in the k-mer graph. For example:

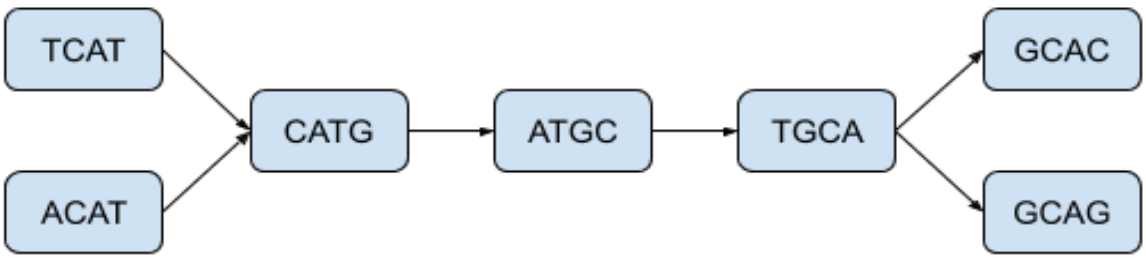

Figure 1.3 k-mer graph formed around the repeat motif "CATGCA", with $k = 4$.

There are two sets of two sequences compatible with the structure in Figure 1.3 (sometimes referred to as a "frayed rope" motif), one for each possible combination of the two entry and two exit paths: { TCATGCAC, ACATGCAG } and { TCATGCAG, ACATGCAC }. Without adequate supplementary information, e.g. from long-range data such as paired-end reads, the repeat cannot be resolved with reasonable certainty into the correct set of paths. In such a case, the contigs corresponding to the collapsed segment and the paths on either side cannot be merged and extended farther, affecting scaffold length and thus contiguity of assembly. That could have side effects on genome coverage, assembly size, and consequent genome size estimates, as illustrated in Figure 1.4: given that path copy number is unobserved, fully resolved repeats yield correct scaffolds with (locally) full genome coverage and correct assembly size, whereas unresolved repeats result in lower genome coverage and inaccurate assembly size.

In this thesis, we consider only haploid and diploid genomes. Continuing with the same example, without loss of generality, and using different multiplicities from those illustrated, suppose the structure in Figure 1.4 is induced by the set { TCATGCAC, ACATGCAG }. In a haploid genome, that corresponds to each of the sequences in the set occurring at separate loci in the genome. In a diploid genome, it could arise from the situation just described (e.g. at two separate loci on both members of a pair of homologous chromosomes), or from homozygosity (flanked by heterozygous sequence) at the same locus on homologous chromosomes, which is mathematically equivalent to a repeat. For compatibility with the standard practice of producing a haploid "consensus" instead of diploid (haplotype-aware) scaffolds, from now on "repeat" shall denote the shared motif only in the former. The two causes make a material distinction in

conventional assembly: for repeats, the structure should be separated into paths of the correct multiplicities; otherwise, it should be collapsed into a single path. Incorrect resolution contributes inaccuracy in genome coverage and assembly size, and decreases contiguity as well. However, graph topology alone does not provide enough information to disambiguate the two, creating another challenge to which current solutions could be improved upon.

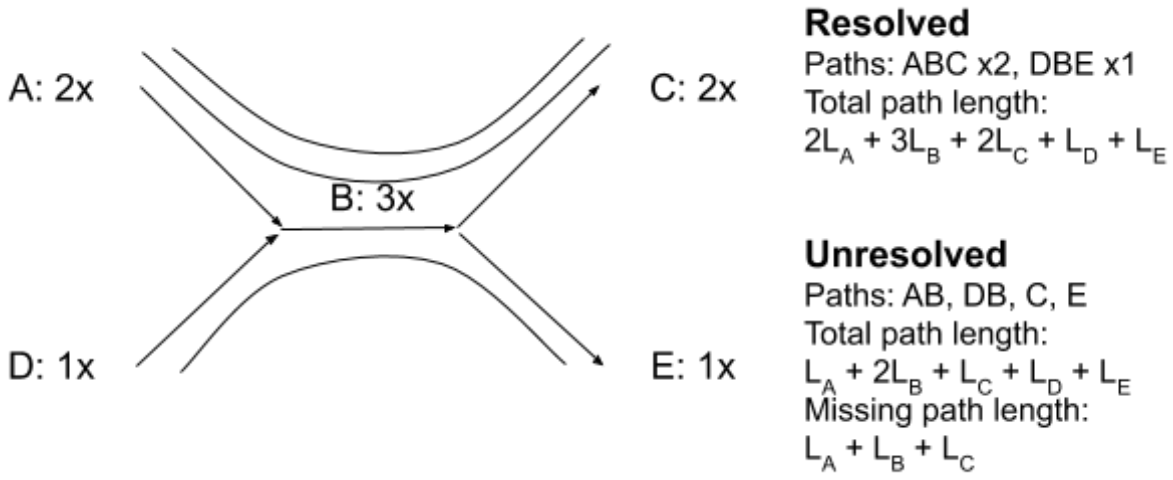

Figure 1.4 Effect of repeat resolution (or lack thereof) on assembly size, with contigs labelled A-E. Curves denote long-range support for the correct paths, ABC (of multiplicity 2) and DBE.

The effect of heterozygosity on the k-mer DBG is often exemplified by the "bubble" structure, i.e. a set of (at most four) disjoint paths sharing only their start and end nodes. Figure 1.5 shows a simple, balanced example (with equally long paths). The bubble can also result from sequence flanked by higher-copy-number segments, just as the frayed rope arises from multiple distinct causes. The two can be seen as duals, and often occur sequentially in k-mer DBGs. Indeed, the role of a bubble in confounding assembly is dependent on its overlapping with a frayed rope on one or both ends, causing the challenge of repeat resolution described earlier, possibly affecting assembly contiguity, coverage, and size. Moreover, the challenge of disambiguating between

homozygosity and repeats is equivalent to that of distinguishing heterozygous paths from those of lower multiplicity relative to flanking segments. Otherwise, absent sequencing error (addressed next), a solitary bubble can be confidently separated into two correct paths.

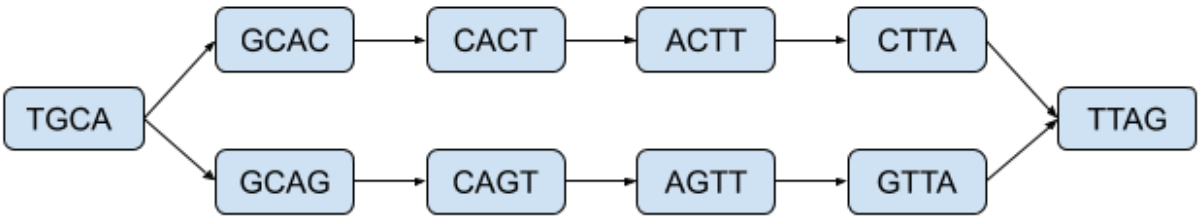

Figure 1.5 A simple bubble, induced by { TGCACTTAG, TGCAGTTAG }, equivalent to a SNP (single nucleotide polymorphism), or a base call error on the last character of GCAC or GCAG, or an inexact repeat (with paths having half the multiplicity of source and sink nodes).

1.3.3 Definition: Copy numbers

Apropos of the discussion about repeats, heterozygosity, and multiplicity in general, the concept of copy number (synonymous with multiplicity) is key to the rest of this thesis. We shall use the following operational definition for sequences of length greater than some reasonably large k (~20), which ignores some possibilities that are of no practical significance or extremely unlikely.

For a sequence from a haploid genome, the definition is simple: it has copy number 0 if it does not occur in the target genome; 1 if it occurs only once; 2 if it occurs twice (whether on the same or separate chromosomes); and so on. In other words, its copy number and number of occurrences are equal.

For a sequence from a diploid genome, there are complicating considerations. Technically, a unique heterozygous sequence occurs once in the genome, a unique homozygous sequence twice, a once-duplicated homozygous sequence with or without a heterozygous mutation in one locus three or four times respectively, and so on. However, beyond the important special cases of unique heterozygous and homozygous sequences, distinguishing between higher odd and even multiplicities is inconsistent with the standard practice of producing haploid "consensus" scaffolds. Thus, for present purposes, sequences having three occurrences are aggregated with those having four, and those having five with those having six, and so on. To maintain gap-free integer numbering, each unique sequence is assigned half its number of occurrences as its copy number; for non-unique sequences, this is rounded up to the nearest integer: 1 occurrence corresponds to copy number 0.5, 2 occurrences to 1, 3 and 4 to 2, 5 and 6 to 3, and so on.

1.3.4 Other complications

A bubble can also be caused by sequencing error; the discrepancy between correct and error reads induces sequences and k-mers resembling those from a heterozygous locus; in particular, a base call error looks like a SNP. Thus, an assembler needs to tell error apart from other competing reasons for the presence of a bubble and retain only the correct path in this case; alternatively, it can prevent bubble formation by error in many cases, albeit at a cost. Failure to do so results in base-level inaccuracy at best, or reduced contiguity, or possibly structural error.

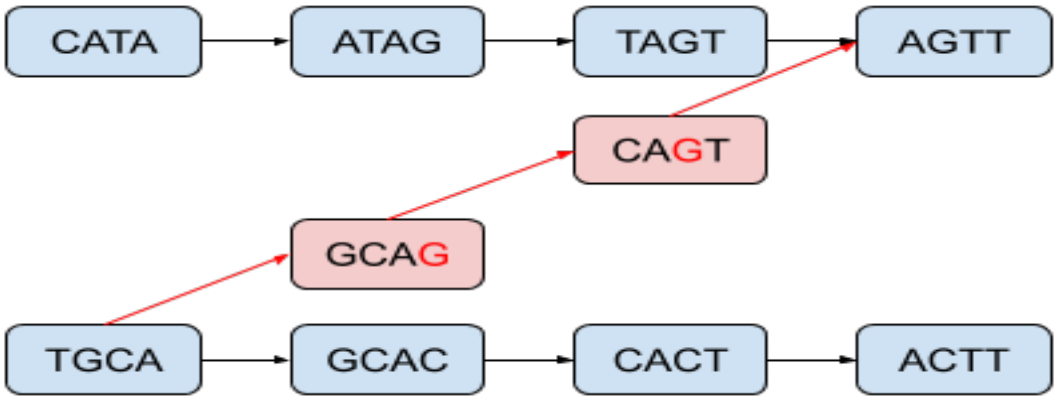

Figure 1.6 Spurious connection resulting from error-induced path, with base call error and spurious path in red. The spurious path could also reconverge on the correct path at the bottom to form a bubble, but that is omitted for clarity.

Last but not least, regions of low read depth also pose problems in assembly, affecting contiguity and coverage, and most insidiously, causing misassemblies. Detailed discussion is omitted here since statistical copy number estimation is expected to be of marginal utility in addressing these problems, both relative to the utility of other approaches, and to its utility in addressing some of the other issues discussed earlier. That said, it induces misassembly by causing the absence of k-mer graph edges that should exist in a correct representation of the target genome; that effect is illustrated in Figure 1.10 in relation to inappropriate error correction.

The (non-linear) DBG structures presented so far are by no means an exhaustive catalogue of what is encountered in practice. They merely illustrate the most basic outcomes possible from introducing vertices of in- or out-degree exceeding 1. In practice, DBGs are often far more complicated, with arbitrarily complex paths subject only to the constraint of node in- and out-degree not exceeding 4; for instance, exact tandem repeats induce cycles. Thus, a variety of approaches have been developed to address this complexity and its resulting problems, while room remains for additions to this toolbox.

1.4 Current approaches to challenges in short-read DBG assembly

Implicit in the discussion above is that the issues described (when not pre-empted—which comes with its own set of problems) apply to contigs in particular, and manifest during contig scaffolding. Some would benefit from accurate copy number estimation: Most simply, the effects of any given occurrence of sequencing error would disappear if spurious contigs arising from it were correctly identified as having copy number 0 and removed. A slightly less simple but arguably much more significant improvement on current practice would be, in general, reliable resolution of low copy number repeats into contigs of correct multiplicity, and in particular, the concomitant disambiguation of heterozygosity from other genomic features as a cause of DBG bubbles (contigs induced by the former have copy number 0.5, while the latter would be associated with higher copy numbers); these would increase accuracy of genome coverage and assembly size. Lastly, assembly contiguity could be a minor area of impact: First, an accurate repeat copy number estimate can be combined with graph topology to enable resolution of paths with insufficient long-range data support when an unambiguous solution to graph-based constraints exists; Figure 1.7 illustrates.

Perhaps more importantly, under certain conditions, accurate contig copy number estimation could improve scaffold contiguity, by preventing inappropriate bubble collapsing (merging separate paths into one), thus preserving long-range information associated with the separate paths for use in repeat resolution; Figure 1.8 illustrates.

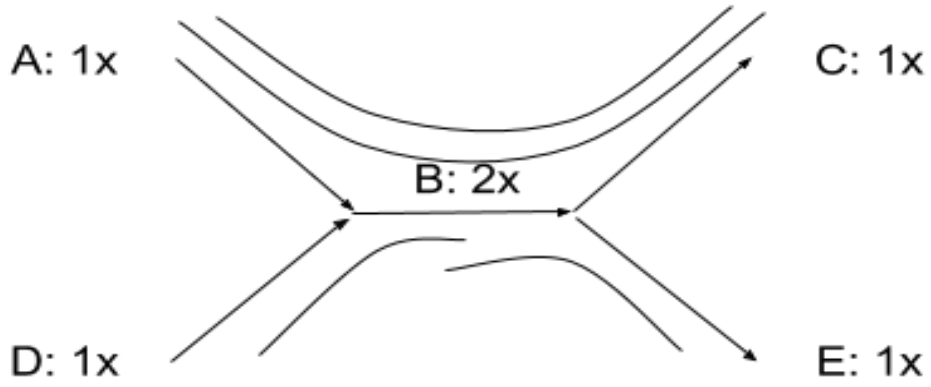

Figure 1.7 ABC has sufficient long-range support, but DBE doesn't. However, B has "remaining" copy number 1, which gives unambiguous support to DBE based on graph topology.

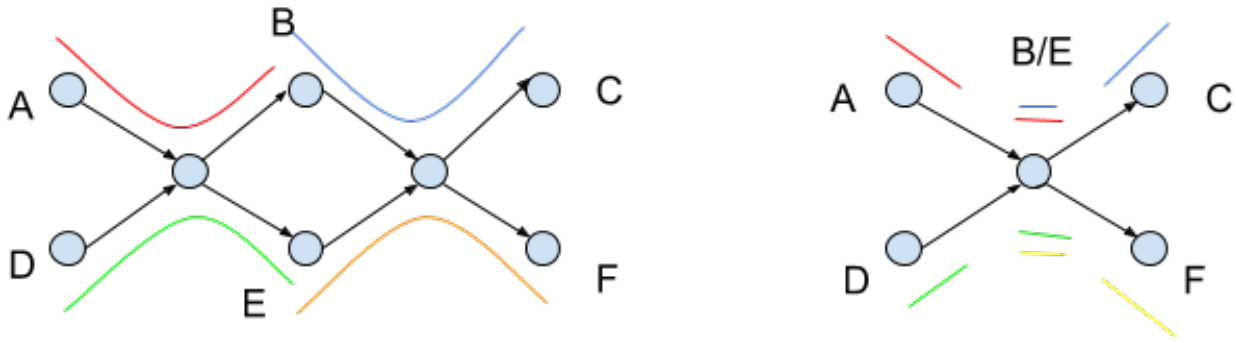

Figure 1.8 Nodes represent contigs. Each colour represents a linked read barcode (more barcodes are required in practice). Left: B has barcode overlap with both A and C, while E overlaps with D and F, thus resolving the subgraph into paths ABC and DEF. Right: with B, E, and their adjacent contigs merged, resulting node B/E has barcode overlaps with A, C, D, and F alike, so the paths can no longer be resolved. Note that this example does not apply to true long read information.

1.4.1 Sequencing error handling

Sequencing error-handling strategies can be classified into three categories, in increasing order of complexity: pre-assembly filtering of likely erroneous sequences, graph simplification, and pre-assembly error correction. The last is largely complementary to the first and second[19,56], though not always[54,55,57].

The simplest strategy, pre-assembly error filtering, excludes sequences from assembly by thresholding on read depth; it is used in conjunction with the other strategies. In ABySS[36] and Supernova[38], k-mers with read depth below an internal or user-specified threshold are excluded from k-mer DBG construction. This strategy is based on the likelihood that with high sequencing depth, most correct k-mers would be covered by a certain number of reads, and that given a low error rate, error-induced k-mers appear in a very small number of (spurious) reads[19]; of course, its sensitivity and precision also depend on an optimal threshold being chosen. However, it is also a dragnet that also removes many legitimate k-mers that happen to be in low-read-depth genomic regions, compromising graph continuity and even increasing the risk of misassembly, as alluded to in the last section.

Graph simplification consists mainly of dead-end branch removal (also called spur erosion or tip removal), bubble collapsing (of disjoint paths into one), and (lastly, in some assemblers) removal of low-depth graph connections; the depth of a path is usually taken to be the average depth of its constituent k-mers (nodes).

A dead-end branch (Figure 1.9) in the DBG is likely induced by a base call error in the first or last k bases of a read (otherwise it would diverge from the last correct k-mer preceding it in the read, and converge back onto the next error-free k-mer); those below some length (usually ~$2k$) threshold are removed[54,55], sometimes iteratively[36,49,50,53], and sometimes with the additional criterion that they have lower read depth than other paths diverging from their point of origin. Like a pre-assembly depth threshold, it is an effective tool for identifying and removing spurious sequences, but also risks removing legitimate sequences in low-read-depth regions, though likely

less so because it is additionally based on graph topology (even when the spur's depth relative to alternative paths is not considered, it is implicitly based on read depth since dead ends are caused by insufficient read depth for continuing k-mer overlap). Unlike that threshold, however, it disrupts connectivity only locally[49]. Bubble collapsing is used for the general purpose of graph simplification, such that its effects include removal of error-induced paths as well as merging of heterozygous paths into a consensus sequence; thus, it shall be addressed separately below as part of heterozygosity identification and repeat resolution. Removal of low-depth graph connections[54,55] below some hard threshold as the final step[49,50,56] of graph simplification targets low-depth paths remaining after prior steps, with the rationale that by then, most legitimate low-depth segments would be subsumed into long unique paths with little effect on average depth[49]. This is an effective strategy that can be further improved by enhancing its precision using statistically informed contig copy number estimates as a more flexible tool than a hard cutoff.

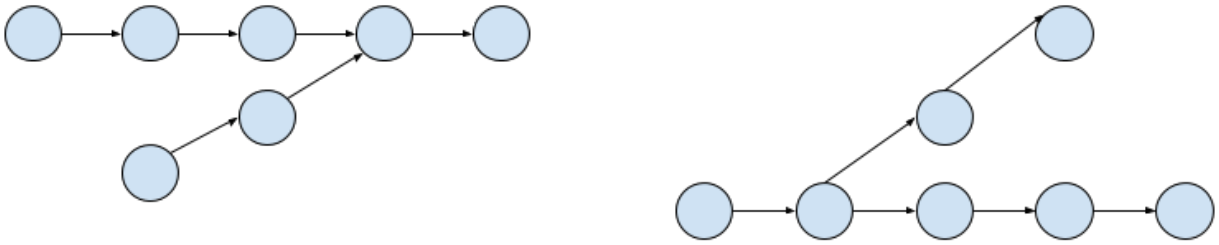

Figure 1.9 Dead-end branch examples

Pre-assembly error correction is a relatively sophisticated method for handling sequencing error, specifically by pre-empting its effects. Most implementations[19,48,51,54,55,61] of this approach are based on counting the occurrences (read depth) of all k-mers across an entire sequencing dataset, followed by an attempt to correct reads containing k-mers with occurrences under some heuristic

threshold (representing a minimum expected occurrence count for genuine k-mers). Reads for which the number of putative error k-mers can be reduced by a set of base-value changes meeting some criteria (e.g. sufficiently frequent alternatives, estimated likelihood of correctness) are changed; otherwise they are left unchanged or discarded. This procedure may be replaced in certain cases by more involved alternatives, e.g. a branch-and-bound tree (graph) algorithm[55]. However, these frequency-based procedures can discard true genetic variation when it coincides with low-depth regions. This effectively removes legitimate connections and paths, incurring the consequent effects on misassembly risk (see Figure 1.10) and contiguity (see Figure 1.8). Another error correction implementation[57] mitigates that by combining multiple sequence alignment of read pairs and base quality scores to identify low-frequency k-mers likely to represent genuine variation and exclude them from correction; however, as its authors state, this relies to some degree on PCR-free data.

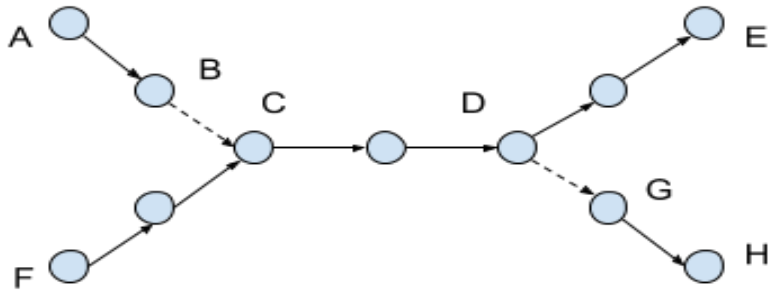

Figure 1.10 Nodes represent k-mers. Misassembly caused by missing DBG edges (dashed lines), resulting from low read depth or k-mer removal during error correction). With BC and DG missing, the contig FCDE is produced, resulting in a major misassembly if the correct set of paths through this region is { ACDE, FCDH } instead of { ACDH, FCDE }.

1.4.2 Heterozygosity identification, repeat resolution, contiguity, and genome coverage

As explained earlier, heterozygosity identification and distinguishing unique homozygous sequence from repeats are equivalent problems. Therefore, they would ideally be solved together, i.e. by copy number estimation. That done, bubbles can be collapsed or retained as appropriate (corresponding to paths of copy number 0.5 or otherwise), before repeat resolution finally takes place.

However, at present, explicit copy number estimation in *de novo* WGS assemblers is generally unaddressed or basic, with arguably two exceptions that shall be described later. Where it is addressed, heuristics are used, such as assuming even coverage[51], or a user-specified depth threshold[58], or some multiple of the mean depth of all contigs[55]. A slightly more sophisticated example[50] adapts the Celera assembler's A statistic[43] into log-odds ratio for the contig being unique vs. having copy number 2; however, it only models repeat status, not a more precise copy number, and assumes Poisson-distributed read occurrences over the genome, whereas real sequencing data, especially high-depth SGS data, is often relatively over-dispersed[64]. Of those, two[55,58] target heterozygosity.

Instead, many assemblers use collapsing of simple bubbles, i.e. bubbles in which all intermediate nodes on all paths have in- and out-degree of 1, as a graph simplification measure meant to increase contiguity by reducing path ambiguity, collapse heterozygous paths into a consensus and, where applicable, discard spurious paths[36,49,50,53,54,56]. This is usually subject to criteria e.g. sufficiently similar sequence identity across paths, and retains the path with highest read depth as the consensus. However, given the well-documented existence of nested repeats in genomes[65], it

can hardly be reliably assumed that bubble paths are heterozygous rather than of higher copy number. Besides, in a diploid genome, only two-path bubbles can indicate a correct choice of path in the presence of heterozygosity, which makes bubble collapsing inappropriate in other cases. The direct consequences of inappropriate bubble collapsing have been described, while its indirect effects are illustrated in Figure 1.8; not least, heterozygosity-induced bubbles that don't meet criteria for collapsing are retained and incorrectly treated as paths with higher copy number, affecting genome coverage and assembly size. Similarly, repeat resolution, genome coverage, and assembly size are determined entirely by contig scaffolding using long-range data, which as explained can yield inaccurate results for incorrectly or incompletely resolved repeats (see Figure 1.4).

1.5 Copy number estimation: related literature

There are few examples of any sort of copy number estimation that go beyond heuristics for assembly workflows. However, the sparse history of this endeavour reaches back at least as far as 2001, when an algorithm for a solution to the copy number problem was published along with EULER-DB[66]. It was formulated as the problem of finding a minimum flow in each connected component of the contig graph, satisfying unit lower capacity constraints on all edges (i.e. each edge has weight, corresponding to copy number, at least 1); for every node, the sum of the weights of inbound edges is constrained to equal that of outbound edges. This problem can be solved by an application of the min-flow max-cut theorem, a dual of the max-flow min-cut theorem. However, the time complexity of computing its solution is high, likely making it unsuitable for most or all SGS datasets.

To our knowledge, the only other example in a workflow for the express purpose of *de novo* WGS assembly arrived much later, when Unicycler, a tool for hybrid assembly of bacterial genomes from short and long reads, was published in 2017. Leveraging naïve usage of read depth information to inform a simple, computationally inexpensive graph-based approach, it starts by assigning copy number 1 to all ("seed") contigs satisfying both the following conditions: first, having median (as opposed to mean) read depth within 10% of the per-base median, and second, with both in- and out-degree at most 1. A greedy algorithm then propagates multiplicity where graph connections and depth are in close agreement, ending when a dead end is encountered or these conditions are no longer satisfied; this is repeated for all seed contigs. Figure 1.11 gives an abridged illustration. This is an elegant solution, but does not allow for copy number estimates that are independent of graph topology and make full use of depth distributional information. Its lack of independence and flexibility makes for many cases where estimates would be absent or incorrect, chiefly: its performance on repeat contigs is dependent on sufficient read depth across single-copy regions as well as connecting regions; no less, incorrect seed identification results in error propagation to all connected contigs.

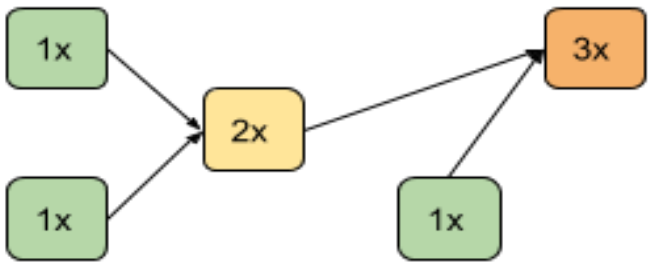

Figure 1.11 **Each colour indicates one round of copy number propagation (green -> yellow -> orange).**

There is, however, some relevant literature on *de novo* copy number estimation from whole-genome sequencing data, that takes more statistically informed approaches. One recent example is GenomeScope[67], a reference-free diploid genome profiler for short reads. It incorporates some mild assumptions regarding heterozygosity and repeats into a negative binomial mixture model of k-mer frequency as a function of depth, up to copy number 2:

$$f(x|\alpha,\beta,\gamma,\delta,\lambda,\rho,G) = G \cdot \{\alpha\mathrm{NB}(x|\lambda,\lambda/\rho) + \beta\mathrm{NB}(x|2\lambda,2\lambda/\rho) + \gamma\mathrm{NB}(x|3\lambda,3\lambda/\rho) + \delta\mathrm{NB}(x|4\lambda,4\lambda/\rho)\}$$

where

G is a scaling parameter w.r.t genome size,

α, β, γ, δ are mixing weights for each distribution (definitions omitted here)

λ is the mean read depth of the distribution of heterozygous k-mers

ρ is a common dispersion parameter

Usage of the negative binomial is motivated by the observation, mentioned earlier, that real sequencing data is often over-dispersed[64] relative to the Poisson distribution, which is popularly used to model sequencing read count data[43,68]; this allows variances to be controlled independently of the mean. This model is of limited utility for the purposes of contig copy number estimation due to its applicability to sequences of only one length, i.e. (some chosen) k, whereas contigs vary greatly in size; furthermore, it only models sequences up to copy number 2, whereas contig multiplicities can be arbitrarily high (within physical constraints). As well, it

does not address haploid genomes, which should be entirely feasible whenever diploid genomes can be modelled.

In relation to that, an effort to estimate copy numbers of arbitrarily long contigs is not unprecedented in the literature: that task has been studied and applied for the purpose of *de novo* detection of copy number variation by co-assembly, in Magnolya, a tool published in 2012[69]. They model the number of reads x_c that start on a contig *c* with a given copy number *i* as $p(x_c|i)$, with all contigs having x_c exceeding some threshold collapsed into a geometric distribution representing outliers:

$$p(x_c|\pi,\lambda,\alpha) = \sum_{i=1}^{M} \pi_i Pois(x_c|\theta_{i,c}) + \pi_{M+1} u(c) \mathrm{Geom}(x_c - \theta_{M+1,c}|\lambda,\alpha,M)$$

with

L_c the length of contig c

λ the number of reads starting at each base

$\theta_{i,c} = iL_c\lambda$ the Poisson rate parameter, with contig length L_c and integer copy number i

u(c) an indicator function, with value 1 for c with $x_c \geq (M+1)L_c\lambda$ and 0 otherwise

Geom() denotes the geometric distribution, with rate parameter α

This seems to be a relatively well-conceived model, though it still leaves room for improvement and adaptation. First, its assumption of shared mixture weights across contig lengths is incompatible with the probabilistically and empirically supported phenomenon that mixture

weights differ with contig length; specifically, repeat probability decreases as length increases. This constrains its applicability to contigs from a single mixture population; for example, the article's authors fit it only on contigs ≥ 500bp. Furthermore, its assignment of contigs to the geometric component based on an observable characteristic (read count value), is inconsistent with maximum-likelihood assignment to the other components. Lastly, it was designed for OLC contigs, for which read count data is naturally available; its discrete nature is unsuited to DBG assembly, for which collecting and retaining read counts or a similar discrete metric until the contig stage is infeasible.

1.6 Statistically informed copy number estimation: Statement and aims

Thus, there is ample basis for a novel statistically informed copy number estimator in the context of *de novo* whole-genome shotgun short-read assembly. Specifically, we develop a statistically informed tool to estimate copy number for contigs from *de novo* WGS DBG assembly of high-throughput short-read data for haploid and diploid genomes. This tool would supplement or supplant existing steps in assembly workflows as appropriate, for the following concomitant primary purposes:

1. Resolve multiplicity for heterozygous, unique, and low copy number repeat contigs.
2. Disambiguate heterozygosity-induced contigs from those of higher multiplicity.
3. Improve the accuracy of genome coverage, assembly size, and resulting genome size estimate in final assembly scaffolds. This would result from accomplishing (1) at the local (contig) level.

It would ideally fulfill the adjunct purpose of identifying spurious contigs (i.e. those having copy number 0), and thus potentially also serve as a computationally inexpensive alternative to error correction, particularly suitable for low-error datasets such as those from predominant Illumina machines. These improvements could also enhance assembly contiguity, as described earlier.

In order to achieve these aims, this tool would need to exploit the information from contig characteristics, primarily read depth and length, to the extent feasible. This can be done by building on existing examples in the literature, such as those just described, that approach problems related to those at hand by modelling copy number in sequencing data.

Chapter 2: Methods

2.1 Introduction

We introduce a fully automated copy number estimator for contigs in de Bruijn graph *de novo* WGS assembly of high-throughput short-read data for haploid and diploid genomes.

As mentioned in the preceding chapter, read count data, naturally available in OLC assembly, is infeasible in DBG assembly. Instead, per-contig mean k-mer read depth (coverage) is widely used as an indicator of contig coverage[49,50,54–56]. Since that metric is generated by read coverage at each occurrence of a contig sequence, it provides information about multiplicity. Using that along with length, we wish to estimate the copy number of each contig in an assembly dataset.

Across such datasets, a consistent relationship between contig length and mean k-mer read depth can be observed: the depth distribution shifts leftward and narrows with increasing length, i.e. the distribution differs substantively with contig length, as illustrated in Figure 2.1. Still, given that contig characteristics are affected by genome repeat structure and assembly parameters, it is difficult or impossible to choose a length range for which a single model applicable across many genomes and datasets would be feasible and useful; more generally, it is infeasible to choose a fixed partition by length of input contigs, and corresponding set of models, that would work well across datasets. A model or algorithm flexible enough to account for the joint distribution of contig length and depth within a dataset, as well as differences in that distribution between datasets, is needed.

There are two main ways that can be done. For one, a model explicitly including length can be used. Alternatively, input contigs can be partitioned by length, and a separate model fit within each range. In either case, a mixture model would be a natural representation of the data (or a subset of it), with each copy number corresponding to a mixture component; in addition, component dispersion should be allowed to vary inversely with length.

In many datasets, the depth distribution varies with length enough such that no single set of mixture weights would represent the entire dataset well. Given that, expressing component dispersion parameters (e.g. variances) and mixture weights as functions of length would be necessary, as the only way to accommodate the effects of length on both depth dispersion and copy number prevalence under a single mixture model. However, it is quite unclear what particular functional forms would provide an adequate approximation of joint length, depth, and multiplicity distributions across datasets.

That leaves fitting separate models on length-based input partitions as the most realistic approach. We now describe the partitioning method, then motivate model formulation.

2.2 Partitioning input contigs by length

If for each contig length value present in a dataset, enough observations were available to fit a model, it would be natural to simply split contigs into sets of equal length to be modelled separately. However, that is rarely if ever true, so a different procedure is needed.

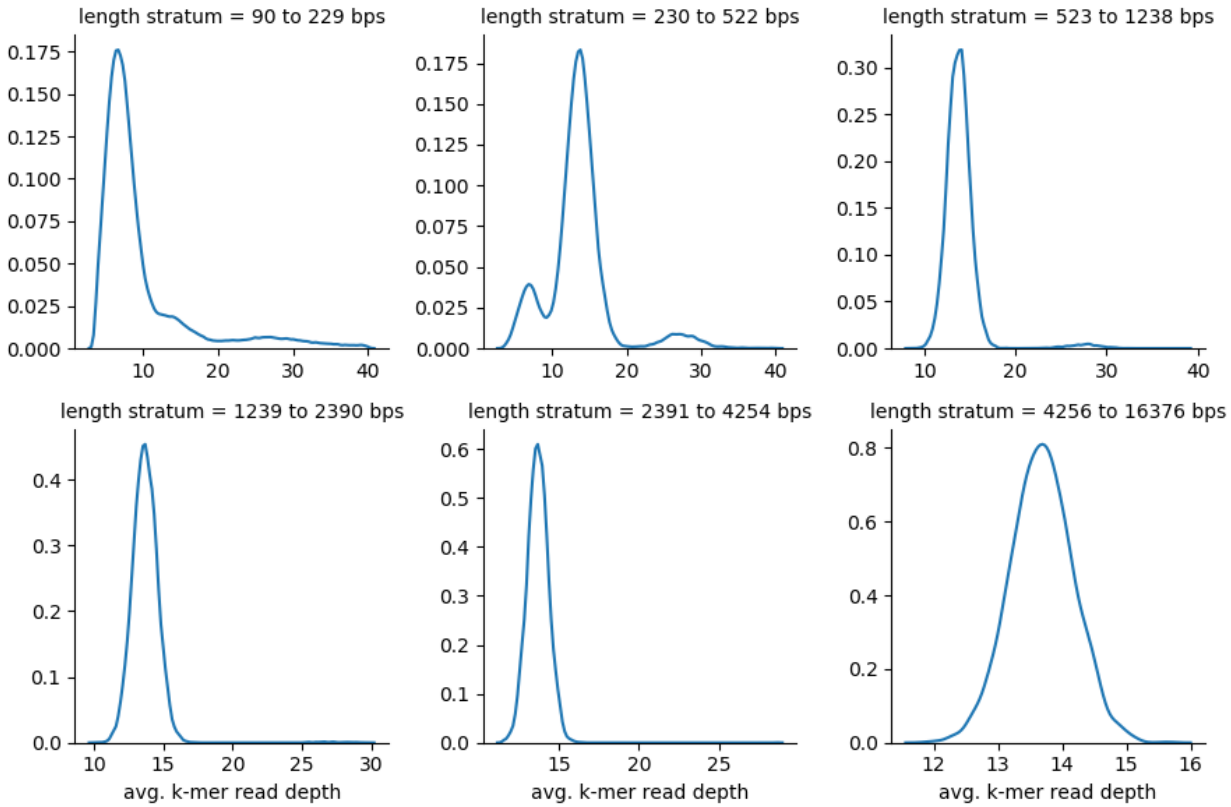

Figure 2.1 Example of differing average k-mer read depth distributions across contig length strata. Contigs assembled using k=90 from 150bp paired-end genomic reads simulated at 50X from the reference genome of *C. elegans* strain Bristol N2 (GCA_000002985.3).

We partition contigs into length strata each with at least a minimum specified number of observations, having the same depth distribution within-stratum and differing distributions across strata, as ascertained by Kolmogorov-Smirnov test p-value not exceeding 5%.

2.3 Model formulation

Read depth for a k-mer (i.e. the number of reads covering its entire length) is an integer-valued random variable; let it be represented by X_i. Thus, for a contig of any given length n (with $n - k + 1$ constituent k-mers), mean k-mer read depth can be represented by $\frac{1}{n-k+1}\sum_{i=1}^{n-k+1} X_i$, where the X_i can be regarded as identically distributed. Of course, unlike X_i, the mean is real-valued.

As illustrated in Figure 2.2, the read depths of neighbouring k-mers are highly correlated, since they share coverage by many of the same reads; thus, these read depths are not independently distributed. The correlation decays with distance, eventually reaching zero since each read is effectively generated by an independent process, so that dependence and correlation come only from coverage by shared reads, which sufficiently distant k-mers do not have. In other words, for some m (specifically, $m = L - k$), $(X_1, \ldots, X_i)$ and $\left(X_{i+j}, X_{i+j+1}, \ldots\right)$ are independent whenever $j > m$, i.e. that $X_1, X_2, \ldots$ is m-dependent. It is also effectively true that for each $k \geq 0$, the joint distribution of $(X_i, \ldots, X_{i+k})$ is the same for all $i \geq 1$, implying that the sequence $X_1, X_2, \ldots$ is stationary. Moreover, we know that the X_i have finite variance. Together, these imply through an extension of the central limit theorem that $\frac{1}{n-k+1}\sum_{i=1}^{n-k+1} X_i$ converges to a Gaussian distribution (with mean $E(X_i)$) as $n \to \infty$[70].

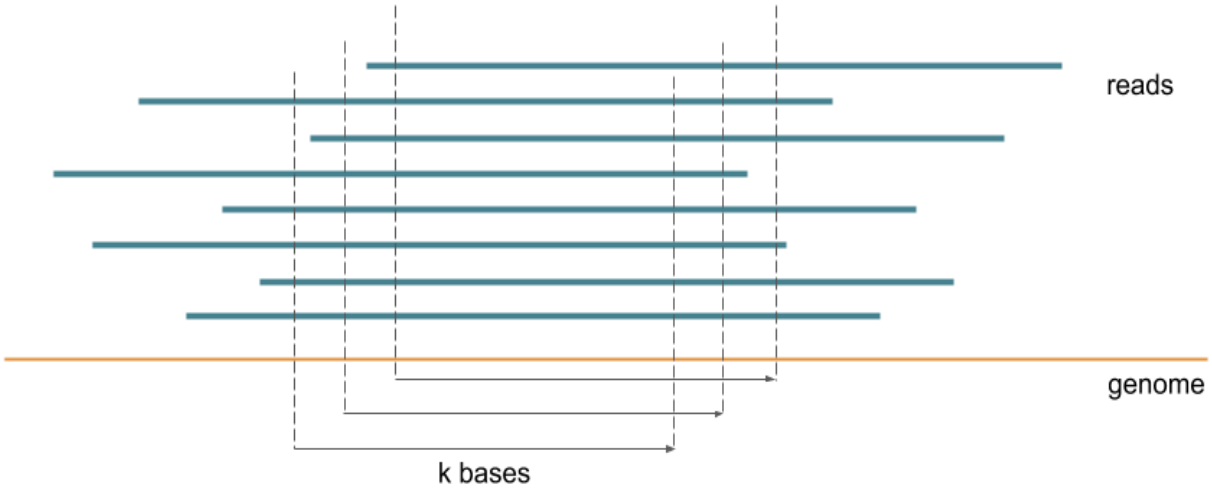

Figure 2.2 Illustration of read coverage correlation between adjacent k-mers. The leftmost k-mer has read depth 6, the middle k-mer 7, and the rightmost k-mer 7.

Therefore, the Gaussian is a good choice for modelling the distribution(s) of sufficiently long single-occurrence contigs; for consistency and simplicity, using it for all length strata (i.e. short contigs as well) would be preferable. In fact, the Gaussian has been used to model k-mer

coverage distributions (i.e. mean read depth distributions for the shortest possible contigs, in this context)[71]. That was done to allow for a free parameter for variance, since it has been observed that in practice, k-mer coverage distributions deviate from the Poisson[72], which would be the correct distribution if the read start rate were truly uniform across all bases in the genome[68].

As for repeats, they are overwhelmingly likely to be located at distances implying mutually independent read coverage. Thus, given the representation of copy number 1 contigs as normally distributed, for a given contig length L_c and $i > 1$, copy number i contig mean k-mer read depth is a sum of i independent normal variables, i.e. normally distributed with mean $i\mu$ and variance $i\sigma^2$, where μ and σ^2 are the mean and variance of copy number 1 contig depths. Similarly, in a diploid genome, copy number 0.5 contigs are normally distributed with mean 0.5μ and variance $0.5\sigma^2$ (recall that copy number 0.5 contigs occur once and copy number 1 twice).

Repeat prevalence decreases rapidly with contig length, such that in most length strata of most datasets (large enough to be partitioned), the maximum copy number specified in model fitting is 2. Nonetheless, in all but the simplest genomes, the depth distributions of short contigs is right-skewed, with density decaying slowly to maximum depth values far exceeding the mode. While simply fitting more Gaussians each representing a copy number would be conceptually valid, the increase in variance with copy number makes estimates for high-depth contigs increasingly uncertain. Therefore, in a stratum with a long-tailed depth distribution, rather than attempting to model and assign high copy numbers individually, we model the aggregation of all copy numbers above a heuristically determined threshold using a single mixture component; all

sufficiently high-depth contigs are assigned to it. For that purpose, we use the gamma distribution, which is known to be used effectively for analysing skewed non-negative data[73].

Naturally, the Gaussian and gamma distributions are also appropriate here i.e. for modelling real-valued contig mean k-mer read depths, because of their continuous domains.

Lastly, where low-depth density seems higher than would be expected from the lowest copy numbers alone, we try to model spurious (copy number 0) contigs using an exponentially distributed mixture component. It is equivalent to a gamma distribution with shape parameter 1; the gamma has been used to model the distribution of spurious k-mer read depths[71], but since a shape value other than 1 doesn't seem to be a realistic representation of the data, the exponential is a more parsimonious choice.

To summarise, for each length stratum within a dataset, a mixture model of copy number components with constrained parameters containing Gaussians, and optionally a gamma and/or exponential distribution, is fit to contig depth data.

2.4 Model statement and implementation

Mixture model parameters are usually estimated using the expectation-maximisation (EM) algorithm. There are reliable existing implementations of EM, but to the best of our knowledge, none allow for constraining parameters, as required here. Thus, as done in GenomeScope, we use non-linear least squares (NLLS) minimisation instead to take advantage of existing implementations accommodating that requirement. Instead of R functionality, however, we use

the Python package lmfit that provides a high-level interface to the Levenberg-Marquardt algorithm (used here) as well as other methods for curve fitting[74].

NLLS necessitates the usage of target values as objective function input. For the goal of estimating probability density parameters, the empirical density or frequency of the data seem like natural options. Using reasonable parameters in both cases (bandwidth and kernel for density, bin width for frequency), we have observed substantially worse results on average using empirical density, perhaps because it is itself estimated and thus a less reliable indicator of true density than empirical frequency is. Thus, we use empirical frequency of contig depth values at heuristically computed intervals as the objective function target variable.

Given that, we fit the following model of contig frequency within each length stratum:

$$f(x;\Theta) = G \cdot \{I_0 w_0 f_{exp}(x;r) + I_{0.5} w_{0.5} f_g\left(x; 0.5\mu, 0.5\sigma^2\right) + \sum_{i=1}^{c} w_i f_g(x; i\mu, i\sigma^2)$$

$$+ I_{c+1} w_{c+1} f_\gamma(x;\lambda,s,\theta)\}$$

where

x is contig mean k-mer read depth

$\Theta \equiv \{r, \mu, \sigma^2, \lambda, s, \theta, I_0, I_{0.5}, I_\gamma, G, \mathrm{w}\}$ is the parameter set, and

f_{exp}, f_g, f_γ are the density functions of the exponential, Gaussian, and gamma distributions,

r is the rate parameter of the exponential distribution,

μ, σ^2 are the mean and (within-stratum) variance respectively of copy number 1 contig depths,

c is the highest copy number represented by a Gaussian in the stratum,

λ, s, θ are the location, shape, and scale parameters respectively of the gamma distribution,

$I_0, I_{0.5}$ are indicator variables for components representing copy numbers 0 and 0.5,

I_{c+1} is an indicator variable for the component aggregating all copy numbers > c,

w is the component weight vector, containing $w_1, \ldots, w_c$, and $w_0, w_{0.5}, w_{c+1}$ where applicable,

G is a scaling parameter with respect to genome size.

Except for μ, which applies to all contigs in a dataset regardless of length, and thus is constrained to be almost identical across otherwise different length strata models, all parameters just listed are stratum-specific; we omit subscripts to keep the notation simple.

$c, I_0, I_{0.5}$, and I_{c+1} are determined at runtime, with the indicator variables taking the value 1 when their respective components are included in the model, and 0 otherwise ($I_{0.5}$ is always 0 for haploid genomes). As suggested earlier, I_{c+1} is usually 0 for length strata not containing the shortest contigs (and 1 for the stratum containing the shortest contigs).

In a diploid genome, the number of occurrences of a sequence is influenced by the heterozygosity level of the genome, in addition to duplication rate, which operates at any ploidy level. Low heterozygosity encourages high prevalence of even-numbered occurrences relative to odd-numbered ones, e.g. more sequences occurring twice instead of once, and four instead of three times; the reverse is true of high heterozygosity. (Relative prevalence of numbers of occurrences should be compared within pairs of each positive multiple of 2 and its preceding integer, e.g. 1 and 2, or 3 and 4, since the effects of duplication rate are largely controlled for within each such pair, but differ and likely outweigh the effects of heterozygosity level across such pairs.)

In other words, using the definition of copy number stated earlier, in a low-heterozygosity genome, at any given length, copy number 1 contigs are likely to be more numerous than copy number 0.5 contigs, and copy number 2 contigs than copy number 1.5 contigs, and so on; in a highly heterozygous genome, the reverse is likely. For present purposes, let's refer to lower-prevalence copy numbers as minor copy numbers, and higher-prevalence ones as major. Given these observations, the mixture component of each minor copy number higher than 1 would be flanked on both sides by major copy number components, causing a high degree of overlap. Due to the uncertainty of contig assignments to minor copy numbers that would ensue, we exclude minor copy numbers exceeding 1 from the model.

Earlier, we stated that repeat prevalence decreases rapidly with contig length. In particular, the longest contigs in any given dataset are exclusively single-copy, or almost so. That results in an empirical depth density with a peak corresponding to the mean of copy number 1 for haploid genomes, and 0.5 or 1 for diploid genomes—in which case making the correct choice is essential for applying the preceding measures appropriately, since the heterozygosity level of any given dataset is generally not known very precisely before assembly. One approach to choosing the copy number best representing the peak empirical depth density of the longest contigs, used for example in GenomeScope (albeit only for k-mers), is to fit a model corresponding to each assumption, and designate the one with lower AIC (Akaike Information Criterion) as correct. However, because both choices can sometimes be mathematically interchangeable, and fitting often converges on parameter values that make them seem similarly plausible, this often gives incorrect results. Thus, where (as is often the case) relevant information is available at the outset

regarding the correct choice, it should be used; we allow this copy number as user-specified input to that end.

These approaches assume that the peak empirical depth density of the longest contigs in a diploid dataset is located sufficiently close to the mean for either copy number 0.5 or 1, instead of far from it, e.g. around the midpoint between the two means. Fortunately, this assumption seems likely to hold under the vast majority of cases both in principle and in practice—probabilistically, the two copy number components are very unlikely to have similar enough prevalences that total density peaks far from either of their means, and in practice, we have never encountered it under reasonable assembly parameter specifications.

Also implicit above is the algorithmic feature of iterating over contig strata in decreasing order of length for model fitting, starting with the stratum containing the longest contigs. As suggested, this is done to take advantage in model fitting of long contigs' well-behaved mean k-mer depth distribution, which usually includes few if any duplicated sequences, and has low variance (which decreases as length increases). This provides the most precise estimate feasible in the initial unconstrained fit of copy number 1 mean depth, μ, which is then used to constrain model fitting for shorter contig strata. (Copy number 1 mean depth in these strata is constrained to be close to the initial estimate, while other parameters are allowed to vary freely, apart from the additional constraint that Gaussian component parameters be multiples of within-stratum copy number 1 component parameters.)

In a dataset too small to be partitioned into multiple length strata, all contigs belong to the same stratum, in which case the conditions supporting the assumption that peak density is close to a single-copy component mean may not hold. Multiple copy number components, including repeats, with relatively similar prevalences and larger variance (compared to a stratum with only long contigs) are likely, resulting in a less well-behaved density that might peak relatively far from a single-copy component mean. Constraining the copy-number-1 mean, unnecessary for fitting a well-behaved density, becomes inappropriate here. Thus, apart from multiplicative constraints between Gaussian parameters as described in the previous paragraph, parameter values are allowed to vary freely in such cases.

Finally, for each fitted copy number component, the probability density implied by parameter estimates is computed at each mean k-mer depth value found in a dataset, and the points where successive densities intersect taken as tentative copy number assignment boundaries. Any repeat (i.e. greater than 1) copy number components with heuristically estimated misclassification rates exceeding a threshold are collapsed into the lowest such copy number by summing the relevant densities. Each contig is then assigned to the copy number corresponding to the (possibly aggregated) component with the greatest density at its depth value; Figure 2.3 illustrates with a simple example.

Our code is publicly available at https://github.com/bcgsc/wgs-copynum-est.

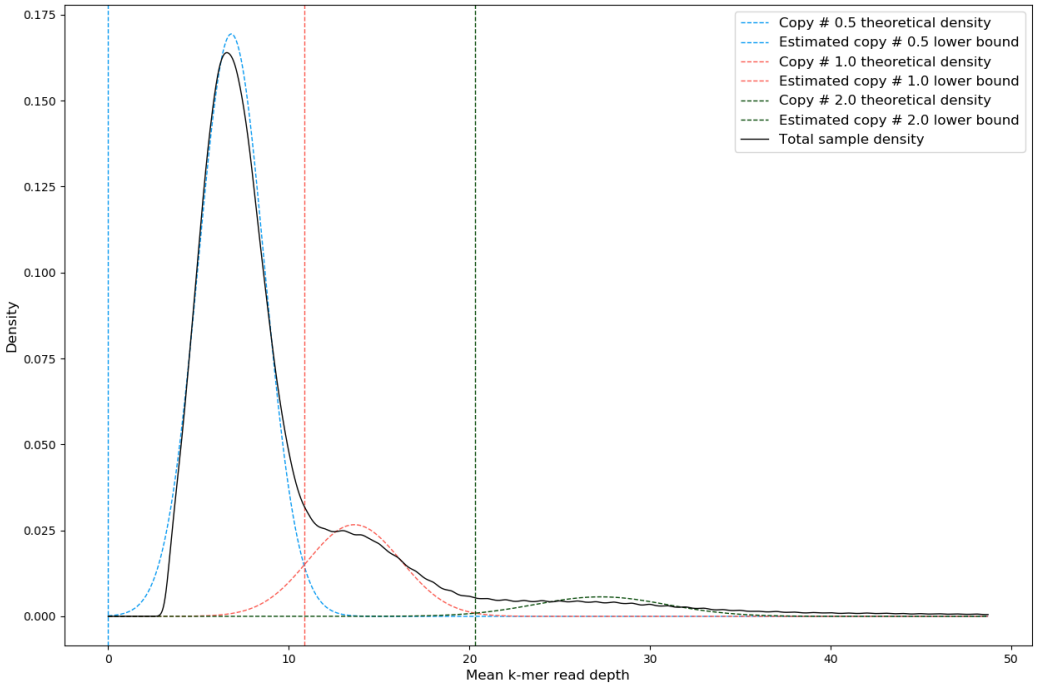

Figure 2.3 Example illustration of copy number assignment boundaries based on component probability densities implied by estimated parameter values.

Chapter 3: Experiments

This chapter describes and reports results from experiments on simulated and real read sets, with further technical detail available in the appendices. We relied primarily on simulated datasets because the partially known discrepancies between the genome underlying any given real dataset and a corresponding reference genome mean that true contig copy numbers cannot be reliably ascertained. This problem is exacerbated by sequence complexity in genomes of any appreciable size (over several megabases). Therefore, algorithm performance largely cannot be accurately assessed using real datasets.

3.1 Preliminaries

3.1.1 Read simulation

Reference genomes for *Escherichia coli*, *Caenorhabditis elegans* strain Bristol N2 (GCA_000002985.3), *Populus trichocarpa* (GCA_000002775.3), and *Homo sapiens* (GRCh38.p13/GCA_000001405.28) were obtained from the U.S. National Center for Biotechnology Information (NCBI) and EnsemblGenomes, from which 150bp Illumina paired-end reads were simulated using the short read simulator DWGSIM version 0.1.11[75] at various mean per-base read coverage levels and target per-base heterozygosity rates.

3.1.2 Real read dataset

A paired-end read set for *E. coli* strain K-12 substrain MG1655 (run accession ERR022075) with first read length 100bp and second read length 102bp produced on an Illumina Genome Analyzer IIx was used for our real dataset experiments.

3.1.3 Read assembly

For each read dataset, experiments were conducted using a range of *k* values, each using contigs from a separate assembly with the corresponding k-mer length. Each assembly was performed using ABySS 2.2.3 via the driver script `abyss-pe` up to the "shim" removal unitig stage, with bubble popping disabled, by specifying a `-2.dot` Make target. The term unitig refers to short contigs obtained directly from merging nodes in the k-mer de Bruijn graph; shims are unitigs that contribute no relevant information to assembly.

The term contig is used in this chapter for consistency with the rest of this work, but refers specifically to unitigs, unless stated or implied otherwise by context.

3.1.4 Summary of experimental settings

The following table summarises the settings of the experiments generated as described in sections 3.1.1 to 3.1.3. Heterozygosity rates are approximate per-base figures.

Data type	**Species**	**Heterozygosity rate, approx. (%)**	**Mean per-base read coverage**	***k* values**
Simulated	*E. coli*	N/A	30	60, 65, …, 115, 120
			50	65, 70, …, 120, 125
	C. elegans	0.0609	30	60, 65, …, 115, 120
			50	65, 70, …, 120, 125
		0.2430	30	60, 65, …, 115, 120
			50	65, 70, …, 120, 125
		1.1726	50	65, 70, …, 120, 125
	P. trichocarpa	0.0580	50	65, 70, …, 120, 125
		0.2338		
		0.5852		
	H. sapiens	0.0563	50	65, 70, …, 120, 125
		0.1148		
Real	*E. coli*	N/A	989	40, 45, …, 85, 90

Table 3.1 Summary of experimental settings

3.2 Copy number estimation

The contigs obtained from each read assembly, given in a -2.fa file, e.g. ecoli-2.fa, were used as input along with k-mer length for copy number estimation in a separate experiment. Program output lists each contig along with its most probable copy number estimated as described in Chapter 2. Effectively, the estimation algorithm classifies contigs by copy number (up to a low maximum, as also described earlier), e.g. 0, 0.5, 1, and 2+. As is, its output is a multiclass classification; from that, a one-versus-many (binary) classification is obtained by collapsing, where necessary or applicable, contigs with copy numbers 0.5 and 1 into the copy number 1 class, and all contigs with copy number over 2 into the "many" class. In our experiments, the multiclass case generally comprises copy numbers 0, 1, and 2+ for haploid-equivalent genomes (i.e. E. coli), or 0, 0.5, 1, and 2+ for diploid genomes.

3.3 Evaluation

Estimator performance was evaluated stand-alone, as well as against GenomeScope (version 2.0, the latest) and Unicycler where applicable. Evaluation was based on true copy numbers as ascertained by counting contig alignments to a corresponding reference genome. Performance in both full (multiclass) and one-many classification was evaluated.

3.3.1 Reference genomes

In simulated data experiments, contigs were aligned to the reference genome from which reads were simulated. In real data experiments (on *E. coli*), contigs were aligned to an *E. coli* strain K-12 substrain MG1655 reference genome (GCA_000005845); variant calling based on read

alignment to that reference indicates minimal difference between it and the genome underlying the read data, which lessens the uncertainty that would typically occur in performance assessment using real data.

3.3.2 Contig reference alignment

For each assembly, the contigs obtained were aligned to their corresponding reference genome using the sequence mapping algorithm BWA-MEM[76], and true copy numbers inferred.

3.3.3 GenomeScope

GenomeScope requires as input a *k* value and the corresponding k-mer profile (histogram) of an unassembled diploid whole-genome short read set. For each experiment using a diploid genome, ntCard[77] was used to compute that profile for GenomeScope execution. Where convergence to a solution occurred, parameter estimates were reported for negative binomial mixture components (as a function of k-mer read depth) equivalent to copy numbers 0.5, 1, 1.5, and 2 as used here. As in our estimation algorithm, the implied probability density was computed for each component at the read depth value of each k-base contig in a dataset, and copy number assigned accordingly. The performance of this procedure (taking copy numbers inferred by contig reference alignment as correct) was compared to that of our algorithm restricted to the same k-base contigs.

3.3.4 Unicycler

Evaluation against Unicycler was performed wherever feasible using *E. coli* reads, real or simulated. Specifically, Unicycler only accepts odd *k* values, so only a subset of those listed in Table 3.1 could be used for comparison.

Moreover, as its first step, it uses SPAdes to assemble Illumina short reads into an assembly graph. It then normalises contig median per-base read depths to the median of that quantity across the 10 longest contigs in the graph for use in its graph-based greedy copy number assignment prior to scaffolding. This results in somewhat different input depth distributions from those our algorithm was arguably designed for, but for the purpose of comparison, using the normalised depths is the most feasible option available; it may also provide some indication of algorithm robustness. Thus, the performance of Unicycler's copy number assignment was compared to that of our algorithm using its normalised contig depth values as input.

Lastly, Unicycler can be run with or without SPAdes' built-in pre-assembly read correction, according to user specification. For each case, a separate evaluation using each of those specifications was performed.

Chapter 4: Results

This chapter presents experimental performance stand-alone for the settings listed in Table 3.1, as well as compared to GenomeScope and Unicycler where feasible. The information shall mostly be presented in boxplots, with each box summarising the distribution of a performance metric (F1 or accuracy) across a range of *k* values with other settings fixed. As suggested in section 3.2, full classification comprises copy numbers 0, 0.5 for diploid genomes, 1, and 2+, except where stated otherwise.

The following should be noted about the tools being used for comparison:

1. As stated, Unicycler is designed to handle bacterial genomes, while GenomeScope only handles diploid genomes, so algorithm performance is compared against Unicycler only for *E. coli* experiments, and against GenomeScope only for other experiments.
2. GenomeScope often failed to converge to a solution. The settings under which that occurred will be stated where relevant; GenomeScope performance results and comparisons are taken only from other, unaffected settings, so that the number of experiments summarised is generally smaller in boxes summarising GenomeScope performance than in boxes summarising algorithm performance.

4.1 Simulated data performance summary

4.1.1 *E. coli*

As indicated in section 3.3.4, our copy number estimation algorithm and Unicycler's require different inputs, such that comparing them entailed using separate datasets from those used for

stand-alone evaluations. Thus, in contrast to the following sections that present experiments using other genomes, this section presents stand-alone and comparative results separately.

4.1.1.1 Stand-alone

We report results from experimental settings listed in Table 3.1 for simulated *E. coli* data.

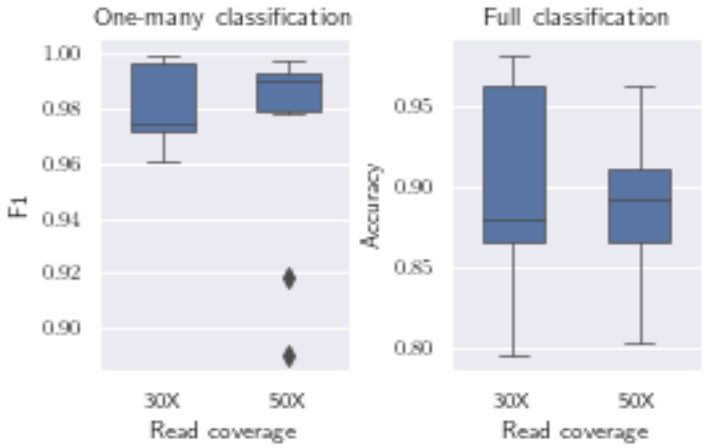

Figure 4.1 Algorithm performance results on assemblies of simulated *E. coli* reads.

4.1.1.2 Comparison with Unicycler

Of the *k* values listed in Table 3.1, only 65, 75, …, 105, 115 for 30X reads, and 65, 75, …, 115, 125 for 50X reads were accepted by Unicycler and thus used for comparison. We report results from experiments without and with SPAdes' built-in pre-assembly read correction in turn.

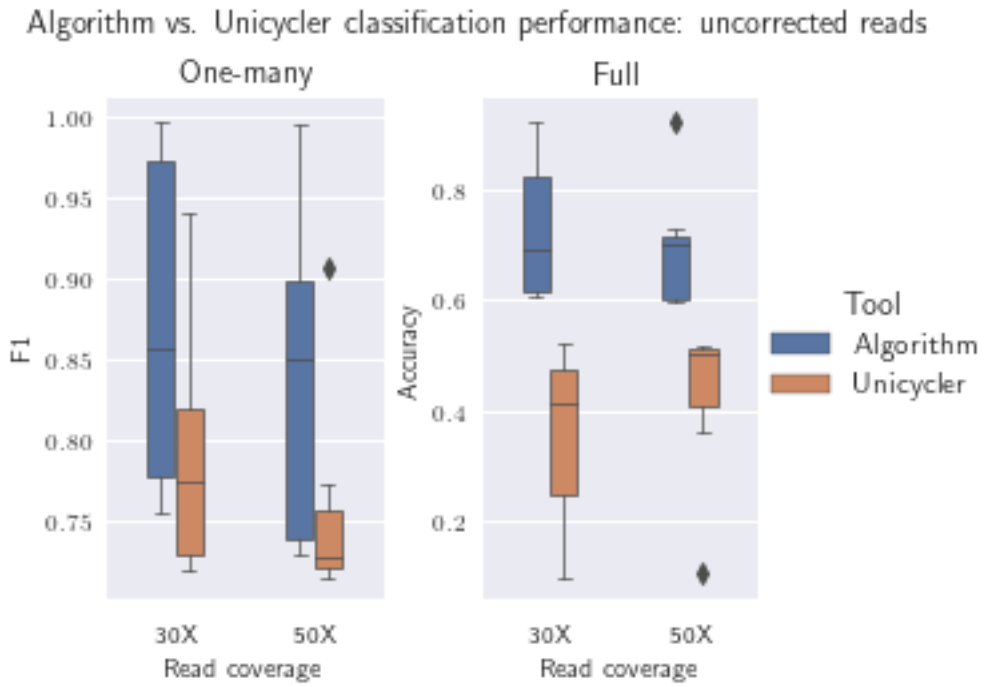

Figure 4.2 Algorithm and Unicycler performance results on SPAdes assemblies of simulated *E. coli* reads, without error correction.

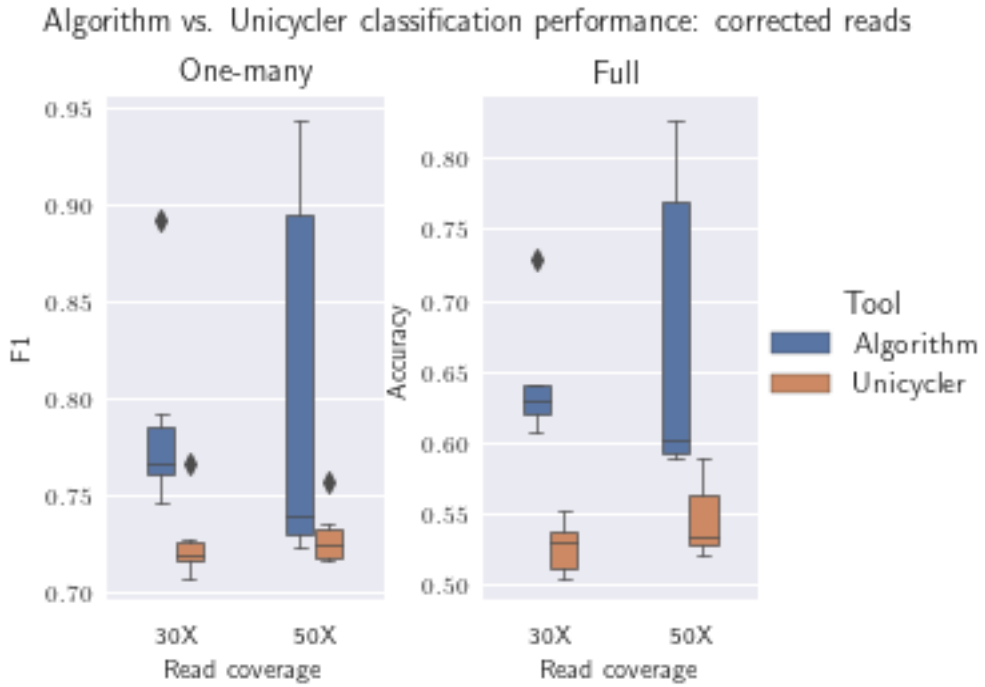

Figure 4.3 Algorithm and Unicycler performance results on SPAdes assemblies of simulated *E. coli* reads, with error correction.

4.1.2 *C. elegans*

Experiments were performed on contigs assembled from both 30X and 50X simulated *C. elegans* read sets with per-base heterozygosity level ~0.06% or ~0.24%. At 1.17% heterozygosity, however, only the 50X read set was used, due to an artefact likely resulting from ABySS assembly of relatively low-coverage reads from a highly heterozygous genome. Specifically, the empirical mean k-mer read depth density of the longest contigs is not well-behaved enough for highly precise estimation of copy number component parameters, while also violating the fundamental assumption that component means are constant across contig lengths. Since mean values for all shorter contig length strata are constrained based on those estimated for the longest stratum, that affects result validity throughout.

GenomeScope failed to converge to a solution under the following settings:

- ~0.06% per-base heterozygosity
    - 30X coverage: $k = 80, 90$
    - 50X coverage: $k = 65, 90, 95, 100, 105$
- ~0.24% per-base heterozygosity
    - 30X coverage: $k = 60, 65, 70, 75$
    - 50X coverage: $k = 65, 70, 75, 95$
- ~1.17% per-base heterozygosity, 50X: $k = 75$

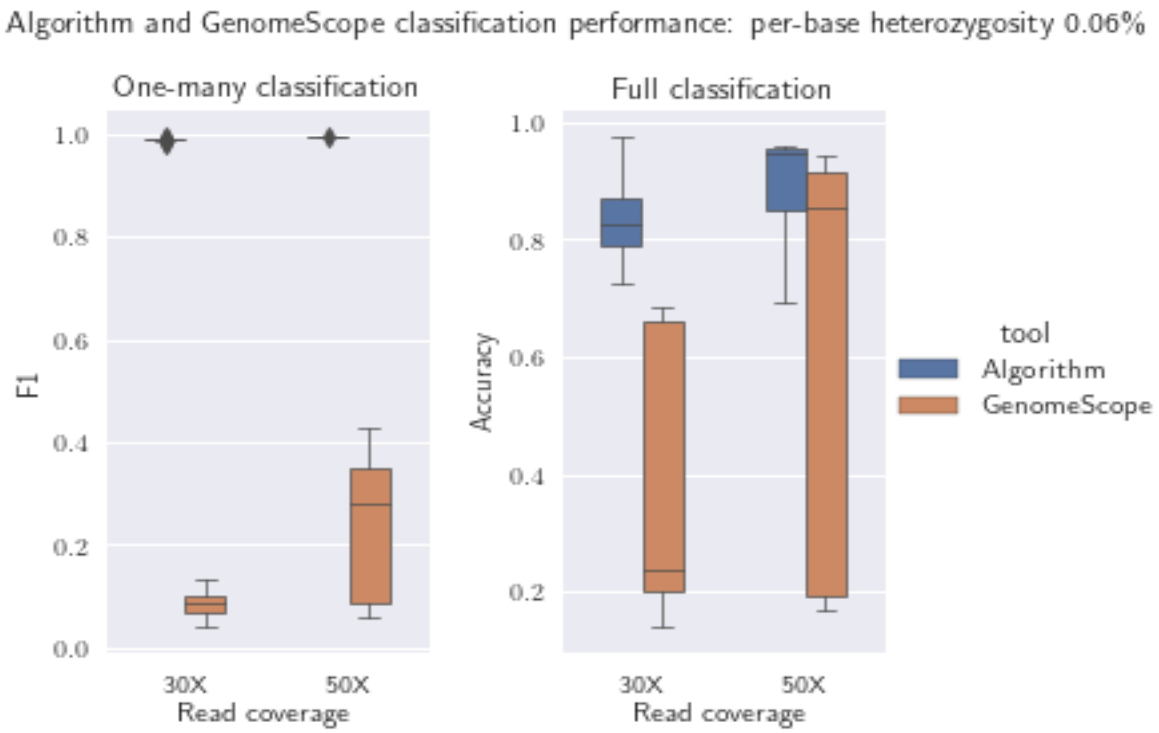

Figure 4.4 Algorithm and GenomeScope performance results on assemblies of simulated *C. elegans* reads at per-base heterozygosity ~0.06%.

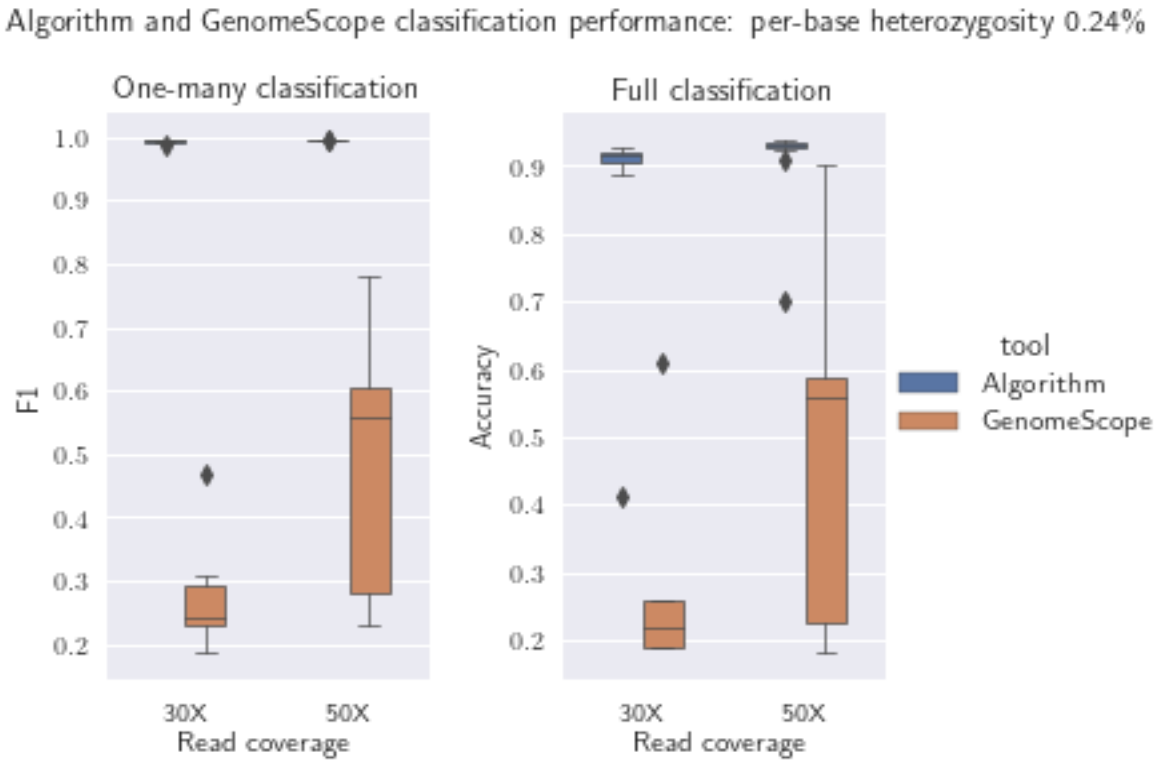

Figure 4.5 Algorithm and GenomeScope performance results on assemblies of simulated *C. elegans* reads at per-base heterozygosity ~0.24%. The lowest-performing outliers in the former's full classification occur at *k*=115 for 30X and *k*=125 for 50X reads.

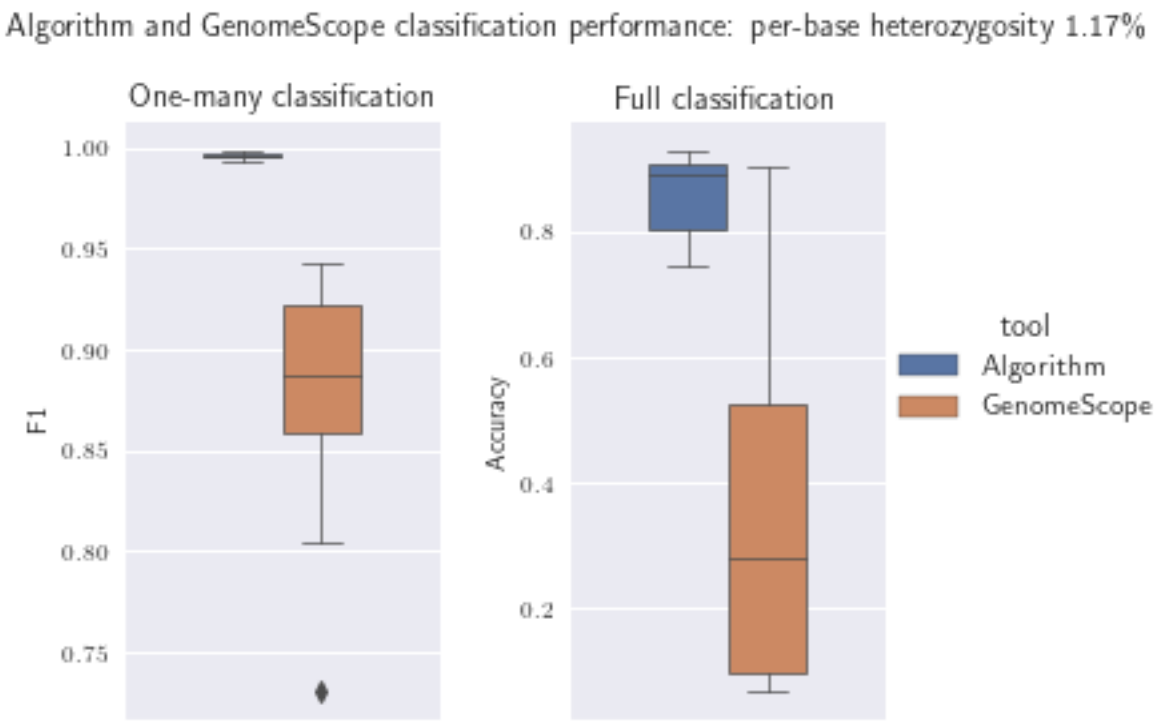

Figure 4.6 Algorithm and GenomeScope performance results on assemblies of simulated *C. elegans* reads at per-base heterozygosity ~1.17%. As explained, experiments on 30X reads were omitted.

4.1.3 *P. trichocarpa*

As stated in Table 3.1, only 50X reads were used for *P. trichocarpa* experiments.

GenomeScope failed to converge to a solution under the following settings:

- ~0.06% per-base heterozygosity: k = 80, 100
- ~0.23% per-base heterozygosity: k = 65, 70, 95, 100, 105
- ~0.59% per-base heterozygosity: k = 65, 70, 95

Our algorithm also failed during model fitting at k=115 with ~0.59% per-base heterozygosity; performance results come from remaining, unaffected settings.

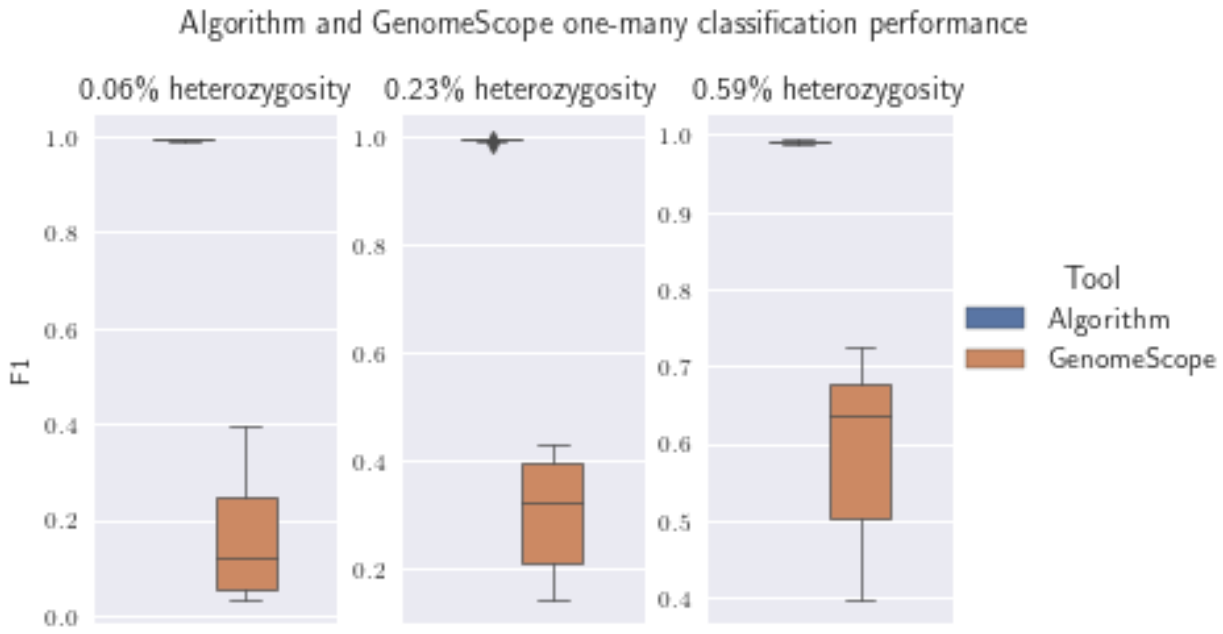

Figure 4.7 Algorithm and GenomeScope one-many classification performance results on assemblies of simulated *P. trichocarpa* reads.

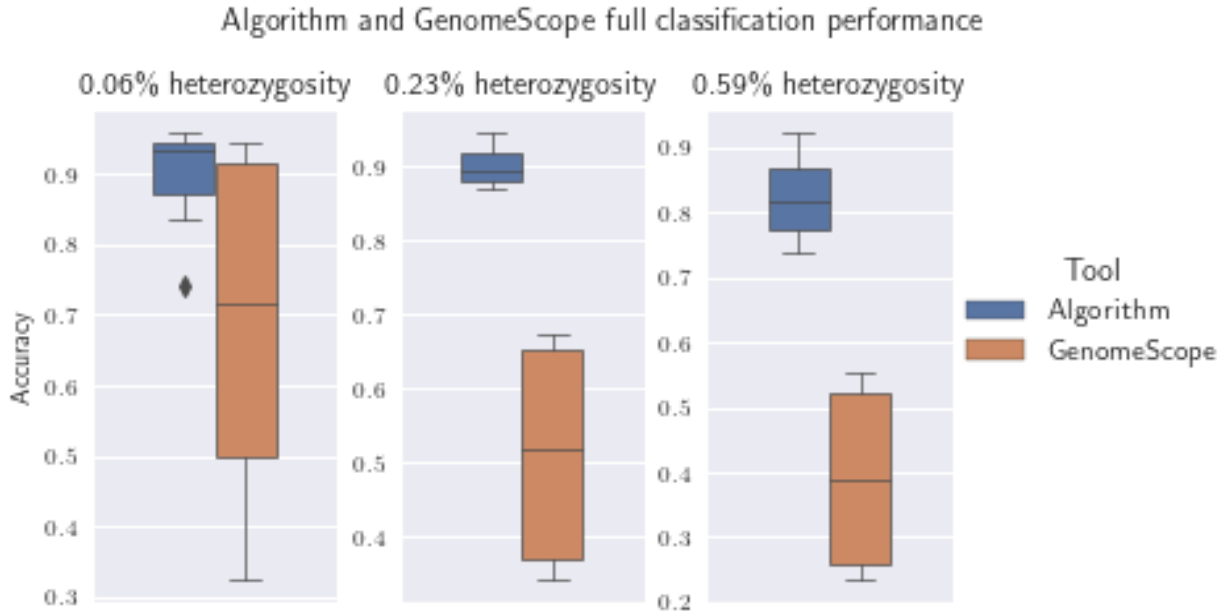

Figure 4.8 Algorithm and GenomeScope full classification performance results on assemblies of simulated *P. trichocarpa* reads. The outlier in algorithm performance on ~0.06% heterozygosity reads occurs at *k*=120.

4.1.4 *H. sapiens*

As stated in Table 3.1, only 50X reads were used for *H. sapiens* experiments.

GenomeScope failed to converge to a solution under the following settings:

- ~0.03% per-base heterozygosity: k = 65, 75, 90, 95, 100, 105
- ~0.11% per-base heterozygosity: k = 75, 80, 95, 100, 105, 110

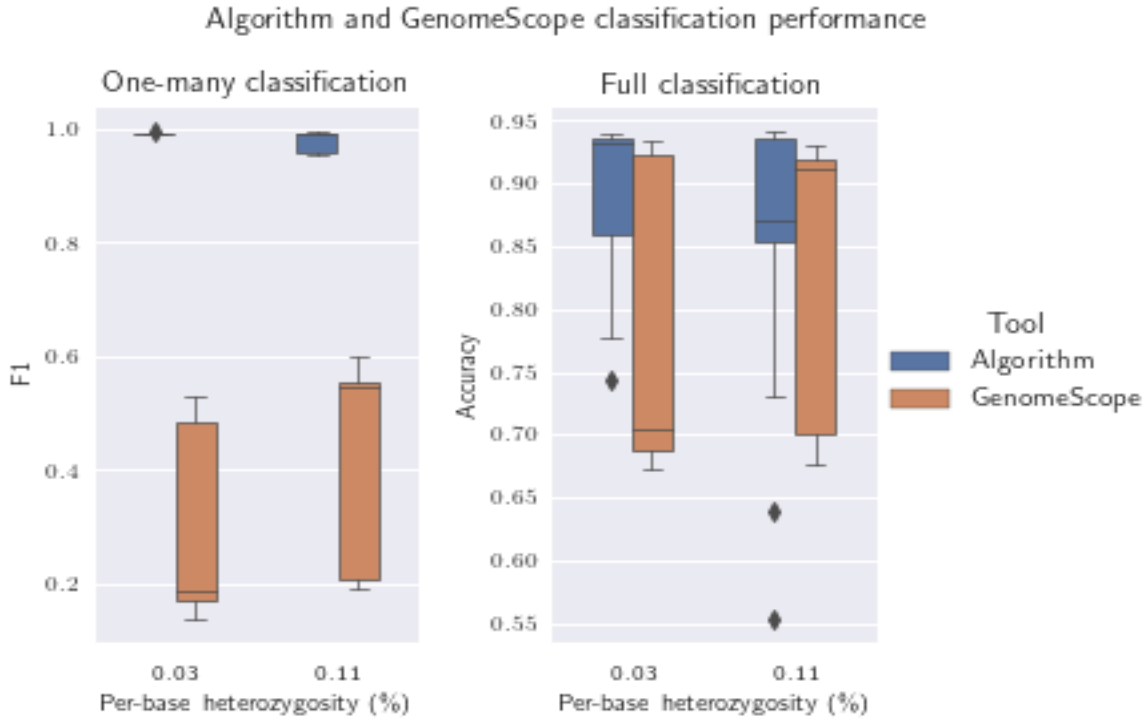

Figure 4.9 Algorithm and GenomeScope classification performance results on assemblies of simulated *H. sapiens* reads. The outliers in the algorithm's full classification occur at k=120 for heterozygosity ~0.03% reads, and k=120, 125 for heterozygosity ~0.11% reads.

4.2 Real (*E. coli*) data performance summary

As with section 4.1.1.1 on experiments using simulated *E. coli* data, this section presents stand-alone and comparative results separately.

4.2.1 Stand-alone

We report results from experimental settings listed in Table 3.1 for real *E. coli* data.

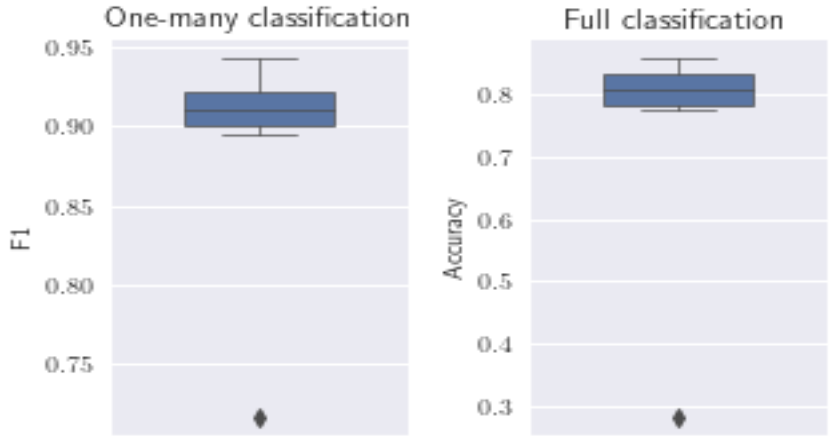

Figure 4.10 Algorithm performance results on assemblies of real *E. coli* reads using *k* = 40, 50, 55, …, 85, 90. (Model fitting failed at k=45.) The outliers for both types of classification occur at *k*=40.

4.2.2 Comparison with Unicycler

Of the *k* values listed above, only 45, 55, …, 75, 85 were accepted by Unicycler and thus used for comparison. As in section 4.1.1.2, we report results from experiments without and with SPAdes' built-in pre-assembly read correction in turn.

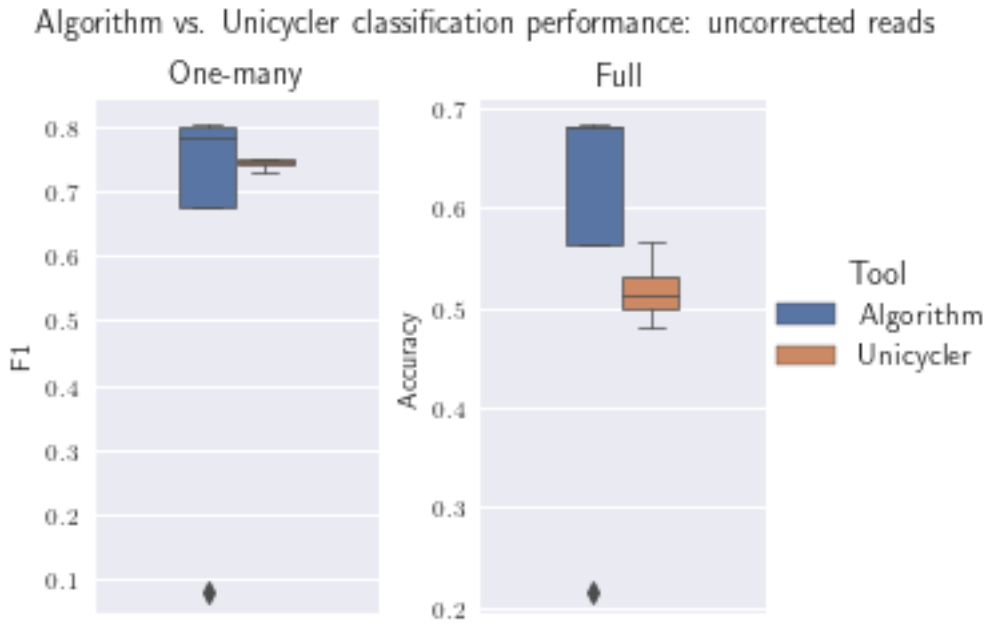

Figure 4.11 Algorithm and Unicycler performance results on SPAdes assemblies of simulated *E. coli* reads without error correction. The outliers in both one-many and full classification occur at *k*=85.

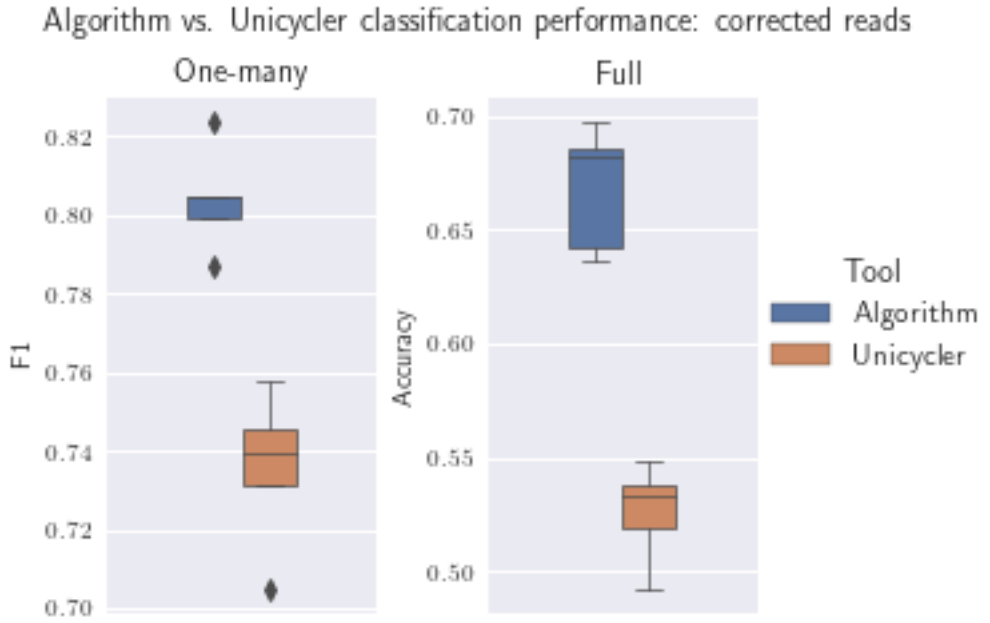

Figure 4.12 Algorithm and Unicycler performance results on SPAdes assemblies of simulated *E. coli* reads with error correction.

4.3 Disambiguating copy numbers 0.5 and 1

Earlier, we identified disambiguating heterozygosity-induced contigs from those of higher multiplicity (in diploid genome assemblies) as one of the primary purposes of copy number

estimation; achieving that aim requires correctly differentiating copy numbers 0.5 and 1. While the preceding sections summarised performance over entire datasets, here we focus on a specific aspect of full classification, namely algorithm performance in classifying contigs with true copy number 0.5 or 1.

4.3.1 *C. elegans*

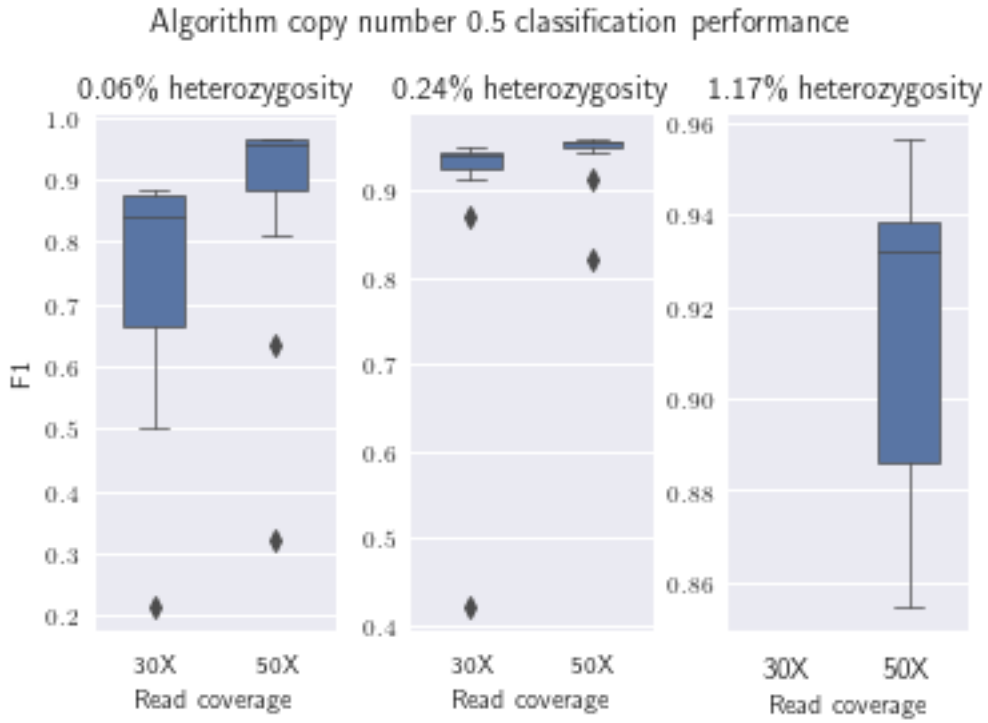

Figure 4.13 Algorithm performance results on copy number 0.5 contigs in assemblies of simulated *C. elegans* reads. The worst-performing outliers occur at *k*=115 for 30X and *k*=125 for 50X ~0.06%, and k=115 for 30X ~0.24% per-base heterozygosity.

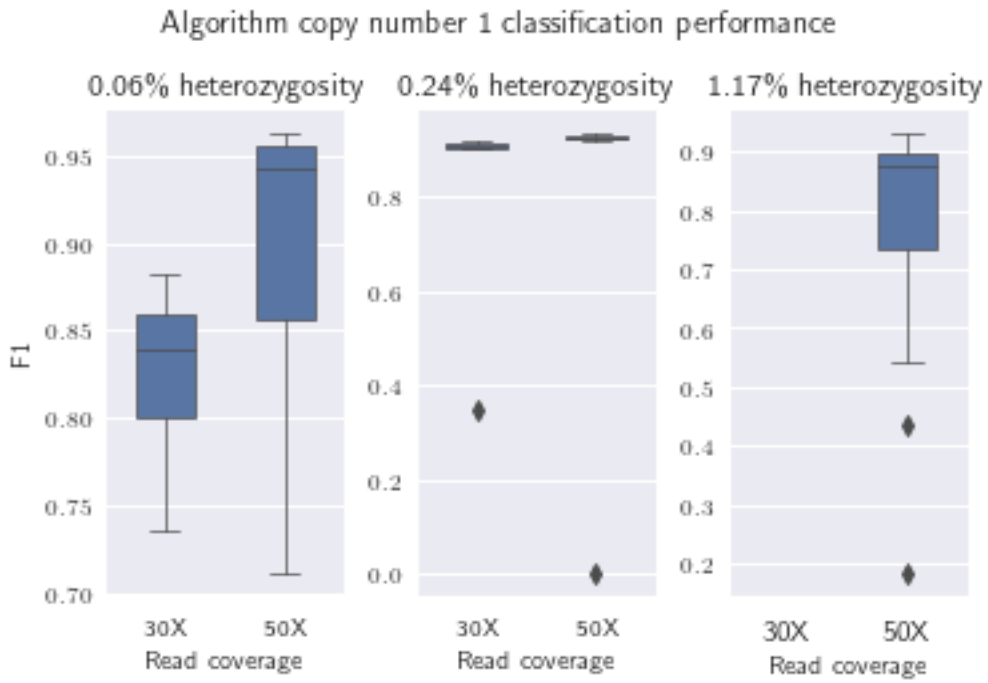

Figure 4.14 Algorithm performance results on copy number 1 contigs in assemblies of simulated *C. elegans* reads. The outliers occur at *k*=115 for 30X and *k*=125 for 50X ~0.24%, and *k*=115, 120 for ~1.17% per-base heterozygosity.

4.3.2 *P. trichocarpa*

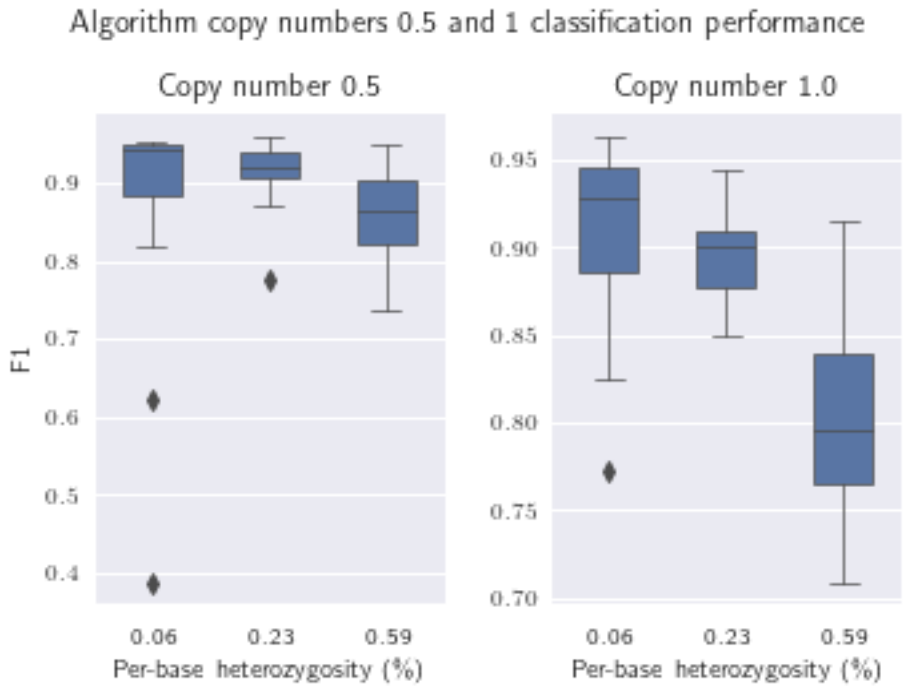

Figure 4.15 Algorithm performance results on copy numbers 0.5 and 1 contigs in assemblies of simulated *P. trichocarpa* reads. The worst-performing outliers occur at *k*=120, 125 in copy number 0.5 classification of ~0.06% per-base heterozygosity contigs.

4.3.3 *H. sapiens*

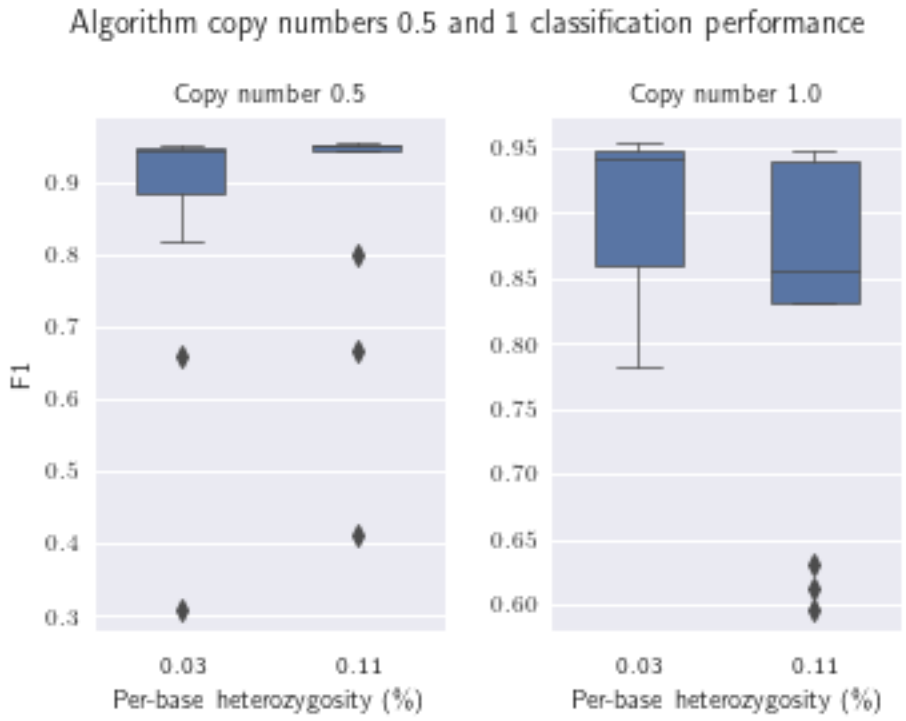

Figure 4.16 Algorithm performance results on copy numbers 0.5 and 1 contigs in assemblies of simulated *H. sapiens* reads. The worst-performing outliers occur at *k*=120, 125 for both ~0.06% and ~0.11% per-base heterozygosity in copy number 0.5 contig classification, and at *k*=115, 120, 125 (for ~0.11% per-base heterozygosity) in copy number 1 classification.

4.4 Computational runtime

As representative examples of runtime, a complete execution of our (single-threaded) program takes less than 5 minutes on *C. elegans* and about 75 minutes on *H. sapiens* datasets on a high-performance computing machine with 128 Intel Xeon E7-8860 processors at 2.2GHz and 1.6TB of RAM.

4.5 Discussion

The results presented demonstrate that our algorithm performs reliably across a range of conditions, including genome heterozygosity level and repeat content, sequencing read coverage, and assembly *k* value. It generally outperforms the closest existing alternative tools, often by a

wide margin, in addition to being more versatile (GenomeScope only handles k-mers, while Unicycler is designed only for bacterial genomes). Here, we discuss some factors that seem to influence copy number estimation performance.

As implied by the preceding figures and captions, the performance of our algorithm is relatively stable over a range of *k* values, but sometimes declines substantially at high extremes, which tend to induce high overlap between different copy number component depth distributions.

It is also obvious that our algorithm performs better at one-versus-many than full classification, which meets intuitive expectations since the depth distributions of copy number components 0.5 and 1 overlap more with each other than they do with the copy number 2 distribution; it is consequently harder to distinguish them from each other than from the latter.

As the previous paragraph implies, copy number 0.5 and 1 contig classification performance is indeed visibly lower than that of one-versus-many classification. However, as suggested by the stability of both types of classification performance, it is also reliable across a range of settings including *k*, albeit declining at high extremes as well.

In contrast, copy number 0 classification performance (not shown) is unfortunately abysmal in most cases—apart from contigs with very low mean k-mer depth, which do not occur during ABySS assembly anyway (except in cases of very high read coverage) due to default removal of low-depth k-mers, depth (and length) information alone is inadequate for distinguishing between spurious and bona fide contigs. This is unsurprising because such spurious contigs have a

substantial proportion of constituent k-mers that also belong to a bona fide contig, which contributes to a higher mean depth than would otherwise be expected.

All else being equal, our results also suggest that our algorithm's performance declines as heterozygosity increases, albeit not drastically. The evidence is less clear that the same is true for GenomeScope; in fact, it shows some support for the opposite, i.e. that it fares better with higher heterozygosity, which could be because its model of k-mer frequency as a function of depth better approximates the true distribution at higher levels of heterozygosity.

Not least, our algorithm performed fairly well across reasonable *k* values (and generally better than Unicycler) on real data, albeit somewhat less well than on simulated data—it fared worse on real data in the cases of ABySS contigs and SPAdes contigs from uncorrected reads, and comparably or slightly better in the case of SPAdes contigs from corrected reads. This suggests that simulated data experiments overstate the performance of our algorithm on real read data (assuming accurate assessment is possible). The single largest factor likely accounting for this difference is the artificially orderly nature of simulated data read depth distributions, in contrast to real data, which tend to have depth distributions that are messier in dataset-specific ways; this is separate from the obvious and common issue of contaminant reads. Taking our real-read contig datasets for example, assuming the generating genome is indeed effectively identical to the reference used, these contigs have depth distributions with much higher variance. On top of that, the data suggest that the discrepancies between their (partially observed) true copy number component distributions differ from those assumed in theory—both means and variances may be substantially removed from their theoretical values.

Chapter 5: Conclusion

In this work, we introduced a novel, statistically informed method for estimating the copy number of contigs in *de novo* WGS high-throughput short read DBG assembly.

We have shown that it performs fairly well in its primary aim of resolving multiplicity up to a low maximum for heterozygous, unique, and repeat contigs, and consequently disambiguating heterozygosity-induced contigs from those of higher multiplicity. Insofar as these are achieved, its final joint aim of improving the accuracy of genome coverage, assembly size, and resulting genome size estimate in final assembly scaffolds would follow. Its performance is also robust over a range of genome characteristics, sequencing coverage levels, and assembly *k* values. Moreover, it is far more versatile than the closest existing alternative tools and usually outperforms them, often by a wide margin.

Nonetheless, these observations are largely informed by experiments under relatively ideal conditions. Performance under less ideal, more realistic conditions could not be more fully characterised, due to the difficulty of assessing it on real datasets. The limited evidence gathered suggests that performance may decline in proportion to the departure of an observed read depth distribution from assumptions underlying the algorithm, though the change may be drastic only for notably irregular distributions.

This drawback may be mitigated somewhat by the simple improvement of providing copy number component density normalised by the sum of densities at any given contig depth as a classification confidence score, so that low-confidence labels may be excluded from use.

That said, it remains a fundamental limitation of using only contig depth and length data in the manner described, in addition to the observed inadequacy of these data alone for identifying spurious contigs. More data, and perhaps an essentially different model, are needed for substantial performance improvement. A model including one or more measures of GC content, e.g. over the length of a contig, or near its endpoints, is a good candidate for such an improvement or change.

In summary, we have shown that versatile, reliable statistically informed copy number estimation for *de novo* WGS short read assembly contigs, using only contig length and read coverage information, is feasible. There is, of course, room for improvement, and we hope that this work will provide a good basis or reference for similar future work.

Bibliography

1. Nagarajan, N. & Pop, M. Sequence assembly demystified. *Nature Reviews Genetics* **14**, 157–167 (2013).
2. Ekblom, R. & Wolf, J. B. W. A field guide to whole-genome sequencing, assembly and annotation. *Evolutionary Applications* **7**, 1026–1042 (2014).
3. Zhao, E. Y., Jones, M. & Jones, S. J. M. Whole-Genome Sequencing in Cancer. *Cold Spring Harb. Perspect. Med.* **9**, a034579 (2019).
4. Miller, J. R., Koren, S. & Sutton, G. Assembly algorithms for next-generation sequencing data. *Genomics* **95**, 315–327 (2010).
5. Sanger, F., Nicklen, S. & Coulson, A. R. DNA sequencing with chain-terminating inhibitors. *Proc. Natl. Acad. Sci.* **74**, 5463–5467 (1977).
6. Hunkapiller, T., Kaiser, R., Koop, B. & Hood, L. Large-scale and automated DNA sequence determination. *Science* **254**, 59–67 (1991).
7. Heather, J. M. & Chain, B. The sequence of sequencers: The history of sequencing DNA. *Genomics* **107**, 1–8 (2016).
8. Earl, D. *et al.* Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome Res.* **21**, 2224–2241 (2011).
9. Sanger, F., Coulson, A. R., Barrell, B. G., Smith, A. J. H. & Roe, B. A. Cloning in single-stranded bacteriophage as an aid to rapid DNA sequencing. *J. Mol. Biol.* **143**, 161–178 (1980).
10. Setubal, J. & Meidanis, J. *Introduction to computational molecular biology*. (PWA Publishing Company, 1997).
11. Fleischmann, R. D. *et al.* Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science* **269**, 496–512 (1995).
12. Venter, J. C. *et al.* The Sequence of the Human Genome. *Science* **291**, 1304–1351 (2001).
13. Adams, M. D. The Genome Sequence of Drosophila melanogaster. *Science* **287**, 2185–2195 (2000).
14. Nyrén, P. & Lundin, A. Enzymatic method for continuous monitoring of inorganic pyrophosphate synthesis. *Anal. Biochem.* **151**, 504–509 (1985).
15. Nyrén, P. Enzymatic method for continuous monitoring of DNA polymerase activity. *Anal. Biochem.* **167**, 235–238 (1987).
16. Ronaghi, M. A Sequencing Method Based on Real-Time Pyrophosphate. *Science* **281**, 363–365 (1998).
17. Margulies, M. *et al.* shendure Next-generation DNA Genome sequencing in microfabricated high-density picolitre reactors. *Nature* **437**, 376–380 (2005).
18. Shendure, J. & Ji, H. Next-generation DNA sequencing. *Nat. Biotechnol.* **26**, 1135–1145 (2008).
19. Chaisson, M. J., Brinza, D. & Pevzner, P. A. De novo fragment assembly with short mate-paired reads: Does the read length matter? *Genome Res.* **19**, 336–346 (2008).
20. Reuter, J. A., Spacek, D. V. & Snyder, M. P. High-Throughput Sequencing Technologies. *Mol. Cell* **58**, 586–597 (2015).
21. Wetterstrand, K. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP). Available at: https://www.genome.gov/sequencingcostsdata/. (Accessed: 17th April 2019)
22. Shendure, J. *et al.* DNA sequencing at 40: past, present and future. *Nature* **550**, 345–353 (2017).
23. Sedlazeck, F. J., Lee, H., Darby, C. A. & Schatz, M. C. Piercing the dark matter: Bioinformatics of long-range sequencing and mapping. *Nature Reviews Genetics* **19**, 329–346 (2018).
24. Roberts, R. J., Carneiro, M. O. & Schatz, M. C. The advantages of SMRT sequencing. *Genome Biol.* **14**, 405 (2013).
25. Jain, M., Olsen, H. E., Paten, B. & Akeson, M. The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. *Genome Biol.* **17**, 239 (2016).
26. Goodwin, S., McPherson, J. D. & McCombie, W. R. Coming of age: Ten years of next-generation

sequencing technologies. *Nature Reviews Genetics* **17**, 333–351 (2016).
27. Dudchenko, O. *et al.* De novo assembly of the Aedes aegypti genome using Hi-C yields chromosome-length scaffolds. *Science* **356**, 92–95 (2017).
28. Jiao, Y. *et al.* Improved maize reference genome with single-molecule technologies. *Nature* **546**, 524–527 (2017).
29. Chin, C.-S. *et al.* Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nat. Methods* **10**, 563–569 (2013).
30. Shi, L. *et al.* Long-read sequencing and de novo assembly of a Chinese genome. *Nat. Commun.* **7**, 12065 (2016).
31. Pendleton, M. *et al.* Assembly and diploid architecture of an individual human genome via single-molecule technologies. *Nat. Methods* **12**, 780–786 (2015).
32. Wick, R. R., Judd, L. M., Gorrie, C. L. & Holt, K. E. Unicycler: Resolving bacterial genome assemblies from short and long sequencing reads. *PLOS Comput. Biol.* **13**, e1005595 (2017).
33. Koren, S. *et al.* Hybrid error correction and de novo assembly of single-molecule sequencing reads. *Nat. Biotechnol.* **30**, 693–700 (2012).
34. Zimin, A. V. *et al.* The first near-complete assembly of the hexaploid bread wheat genome, Triticum aestivum. *Gigascience* **6**, (2017).
35. Koren, S. *et al.* Canu: scalable and accurate long-read assembly via adaptive k -mer weighting and repeat separation. *Genome Res.* **27**, 722–736 (2017).
36. Jackman, S. D. *et al.* ABySS 2.0: resource-efficient assembly of large genomes using a Bloom filter. *Genome Res.* **27**, 768–777 (2017).
37. Jiao, W.-B. & Schneeberger, K. The impact of third generation genomic technologies on plant genome assembly. *Curr. Opin. Plant Biol.* **36**, 64–70 (2017).
38. Weisenfeld, N. I., Kumar, V., Shah, P., Church, D. M. & Jaffe, D. B. Direct determination of diploid genome sequences. *Genome Res.* **27**, 757–767 (2017).
39. Narzisi, G. & Mishra, B. Comparing De Novo Genome Assembly: The Long and Short of It. *PLoS One* **6**, e19175 (2011).
40. Tarhio, J. & Ukkonen, E. A greedy approximation algorithm for constructing shortest common superstrings. *Theor. Comput. Sci.* **57**, 131–145 (1988).
41. Sutton, G. G., White, O., Adams, M. D. & Kerlavage, A. R. TIGR Assembler: A New Tool for Assembling Large Shotgun Sequencing Projects. *Genome Sci. Technol.* **1**, 9–19 (1995).
42. Green, P. phrap documentation. Available at: http://www.phrap.org/phredphrap/phrap.html. (Accessed: 11th April 2011)
43. Myers, E. W. A Whole-Genome Assembly of Drosophila. *Science* **287**, 2196–2204 (2000).
44. Hernandez, D., Francois, P., Farinelli, L., Osteras, M. & Schrenzel, J. De novo bacterial genome sequencing: Millions of very short reads assembled on a desktop computer. *Genome Res.* **18**, 802–809 (2008).
45. Simpson, J. T. & Durbin, R. Efficient de novo assembly of large genomes using compressed data structures. *Genome Res.* **22**, 549–556 (2012).
46. Pevzner, P. A. 1-Tuple DNA Sequencing: Computer Analysis. *J. Biomol. Struct. Dyn.* **7**, 63–73 (1989).
47. Idury, R. M. & Waterman, M. S. A New Algorithm for DNA Sequence Assembly. *J. Comput. Biol.* **2**, 291–306 (1995).
48. Pevzner, P. A., Tang, H. & Waterman, M. S. An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci.* **98**, 9748–9753 (2001).
49. Zerbino, D. R. & Birney, E. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.* **18**, 821–829 (2008).
50. Zerbino, D. R., McEwen, G. K., Margulies, E. H. & Birney, E. Pebble and Rock Band: Heuristic Resolution of Repeats and Scaffolding in the Velvet Short-Read de Novo Assembler. *PLoS One* **4**, e8407 (2009).
51. Butler, J. *et al.* ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome*

Res. **18**, 810–820 (2008).
52. Gnerre, S. *et al.* High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc. Natl. Acad. Sci.* **108**, 1513–1518 (2011).
53. Simpson, J. T. *et al.* ABySS: A parallel assembler for short read sequence data. *Genome Res.* **19**, 1117–1123 (2009).
54. Li, R. *et al.* De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res.* **20**, 265–272 (2010).
55. Luo, R. *et al.* SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *Gigascience* **1**, 18 (2012).
56. Bankevich, A. *et al.* SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *J. Comput. Biol.* **19**, 455–477 (2012).
57. Weisenfeld, N. I. *et al.* Comprehensive variation discovery in single human genomes. *Nat. Genet.* **46**, 1350–1355 (2014).
58. Garg, S. *et al.* A graph-based approach to diploid genome assembly. *Bioinformatics* **34**, i105–i114 (2018).
59. Peltola, H., Söderlund, H. & Ukkonen, E. SEQAID: a DNA sequence assembling program based on a mathematical model. *Nucleic Acids Res.* **12**, 307–321 (1984).
60. Nagarajan, N. & Pop, M. Parametric Complexity of Sequence Assembly: Theory and Applications to Next Generation Sequencing. *J. Comput. Biol.* **16**, 897–908 (2009).
61. Chaisson, M., Pevzner, P. & Tang, H. Fragment assembly with short reads. *Bioinformatics* **20**, 2067–2074 (2004).
62. Poptsova, M. S. *et al.* Non-random DNA fragmentation in next-generation sequencing. *Sci. Rep.* **4**, 4532 (2015).
63. Benjamini, Y. & Speed, T. P. Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Res.* **40**, e72–e72 (2012).
64. Miller, C. A., Hampton, O., Coarfa, C. & Milosavljevic, A. ReadDepth: A Parallel R Package for Detecting Copy Number Alterations from Short Sequencing Reads. *PLoS One* **6**, e16327 (2011).
65. Pevzner, P. A. De Novo Repeat Classification and Fragment Assembly. *Genome Res.* **14**, 1786–1796 (2004).
66. Pevzner, P. A. & Tang, H. Fragment assembly with double-barreled data. *Bioinformatics* **17**, S225–S233 (2001).
67. Vurture, G. W. *et al.* GenomeScope: fast reference-free genome profiling from short reads. *Bioinformatics* **33**, 2202–2204 (2017).
68. Lander, E. S. & Waterman, M. S. Genomic mapping by fingerprinting random clones: A mathematical analysis. *Genomics* **2**, 231–239 (1988).
69. Nijkamp, J. F. *et al.* De novo detection of copy number variation by co-assembly. *Bioinformatics* **28**, 3195–3202 (2012).
70. Ferguson, T. S. *A Course in Large Sample Theory.* (Chapman and Hall/CRC, 1996).
71. Kelley, D. R., Schatz, M. C. & Salzberg, S. L. Quake: quality-aware detection and correction of sequencing errors. *Genome Biol.* **11**, R116 (2010).
72. Dohm, J. C., Lottaz, C., Borodina, T. & Himmelbauer, H. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Res.* **36**, e105–e105 (2008).
73. Kundu, D. & Manglick, A. Discriminating Between The Log-normal and Gamma Distributions. *J. Appl. Stat. Sci.* **14**, 175–187 (2005).
74. Newville, M. *et al.* lmfit/lmfit-py 0.9.11. (2018). doi:10.5281/zenodo.1301254
75. Homer, Ni. DWGSIM. (2014).
76. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv Prepr. arXiv* (2013).
77. Mohamadi, H., Khan, H. & Birol, I. ntCard: a streaming algorithm for cardinality estimation in genomics data. *Bioinformatics* **33**, 1324–1330 (2017).

Appendices

Appendix A

A.1 Reference genome sources

C. elegans, strain Bristol N2 (GCA_000002985.3):

https://www.ncbi.nlm.nih.gov/assembly/GCF_000002985.6

E. coli, strain BL21-Gold(DE3)pLysS AG (GCA_000023665.1), for simulated data experiments:

ftp://ftp.ensemblgenomes.org/pub/bacteria/release-48/fasta/bacteria_22_collection/escherichia_coli_bl21_gold_de3_plyss_ag_/dna/

E. coli, strain K-12 substrain MG1655 (GCA_000005845), for real data experiments:

ftp://ftp.ensemblgenomes.org/pub/bacteria/release-48/fasta/bacteria_0_collection/escherichia_coli_str_k_12_substr_mg1655/dna/

H. sapiens (GRCh38.p13/GCA_000001405.28):

ftp://ftp.ensembl.org/pub/release-98/fasta/homo_sapiens/dna/

P. trichocarpa (GCA_000002775.3):

ftp://ftp.ensemblgenomes.org/pub/plants/release-48/fasta/populus_trichocarpa/dna/

A.2 Real whole-genome sequencing read dataset source

E. coli, strain K-12 substrain MG1655: https://www.ebi.ac.uk/ena/browser/view/ERR022075

A.3 Read simulation

Simulation was done with specified per-base error rates increasing over each read and higher in the second read[72], and other non-default parameter values used for all datasets as follows:

Length of the first read, `1 = 150`

Length of the second read, `2 = 150`

First read per-base error rate, `e = 0.001-0.005`

Second read per-base error rate, `E = 0.002-0.01`

Maximum number of Ns allowed in a given read, `n = 150`

Standard deviation of base quality scores, `Q = 4`

Mean per-base read coverage was specified as applicable with `-C`, and haploid mode was set (to replace the diploid default) using `-H` for *E. coli* dataset simulations. (While not technically haploid, *E. coli* genomes are effectively so for present purposes.) Mutation rate `-r` was varied to attain different target per-base heterozygosity rates.

A.4 Read assembly

Standard parameter values were used except for *k* and the number of MPI processes used during assembly. An example invocation follows:

```
abyss-pe name=celegans k=80 j=16 np=16 v=-v in=”read1.fastq read2.fastq” celegans-2.dot
ABYSS_OPTIONS=”-b0”
```

A.5 Copy number estimation

Optional arguments relevant here, in switch form, are `--haploid` to specify that the input data comes from a haploid rather than diploid genome, and `–half` to specify (for diploid data) that program output should distinguish between copy numbers 0.5 and 1 (which would otherwise be aggregated as copy number 1). Peak density copy number, if known, can also optionally be specified as the final argument. Additionally, a FASTA file listing input contigs, k-mer length, and output folder path are required arguments.

Example command-line invocations:

```
$ python est.py –haploid abyss-out/ecoli-2.fa 75 results/
$ python est.py –half abyss-out/celegans-2.fa 75 results/ 1
```

Appendix B

B.1 Heterozygosity estimation

To estimate the heterozygosity of a sequencing read set, we used the following steps:

1. Performed variant calling based on reference alignment of sequencing reads (using BWA-MEM for alignment, Samtools' view and sort for alignment output formatting, then BCFtools' mpileup to generate genotype likelihoods and call for the variant calling itself).
2. Filtered out monoallelic variants, and likely low-quality calls using read depth, mapping quality, and call quality criteria.
3. Used calls remaining after (2), together with haploid genome size, to compute an approximate per-base heterozygosity rate.

B.2 Contig reference alignment

For contig alignment using BWA-MEM[76], the option to output all alignments, -a, was set, while minimum seed length -k was set in simulated data experiments to the *k* used for assembly, and in E. coli real read set experiments to 24 (i.e. almost 25% of read length). Default values were used for other parameters.

B.3 Alignment counting

Since BWA-MEM outputs alignments varying widely in quality by default, some work is needed to identify only the ones corresponding to the locus or loci that generated any given contig.

For contigs assembled from a genome known to have a small number and degree of differences from the reference (e.g. a simulated dataset or closely related specimen), we know that at any given reference locus, the only possible sources of discrepancy between its sequence and that of a matching contig are assembly error, sequencing error, and a small bona fide (simulated or natural) difference, each of which has a low probability that is effectively equal across all loci. Therefore, the best alignments for each contig, i.e. those with the single highest sequence identity as indicated by CIGAR, MD, and NM field values, represent the correct match loci. There may be one or more such alignments, each corresponding to a distinct locus, but they (are required to) share identical CIGAR, MD, and NM values—i.e. have identical sequences—to be counted as matches. To exclude major misassemblies, e.g. chimeric contigs, we also required that the total edit distance (counting clips) of a best alignment be at most one standard deviation from the mean edit distance of all best alignments in the relevant contig's length stratum.

B.4 GenomeScope

Example invocation, with *k*=90, `k90.hist` as input k-mer profile, and `out/k90/` as output folder:

```
$ genomescope.R -i k90.hist -o out/k90/ -k 90
```

B.5 Unicycler

As stated in online documentation, Unicycler's contig depth normalisation typically results in the (bacterial) chromosome having a depth near 1x. The `–depth_filter` argument specifies a fraction of the chromosomal depth for exclusion of contigs from copy number assignment and scaffolding. The following is an example invocation, with `-1` and `-2` specifying input read files, `-o` for output folder, `--kmers` for assembly *k* value, `--min_fasta_length` specifying the minimum contig length for

Unicycler's output FASTA file (which we always set equal to *k*), --no_correct to turn off SPAdes' built-in pre-assembly read correction, and --verbosity, --keep, and –depth_filter values constant:

```
$ unicycler -1 reads/read1.fastq -2 reads/read2.fastq -o unicycler/ --verbosity 2 –min_fasta_length 75
--keep 3 --spades_path /gsc/btl/linuxbrew/bin/spades.py --no_correct --kmers 75 –depth_filter 0.1
```

For each case, a separate evaluation with and without SPAdes' read correction was run.