BIGDATA: Small: DA: DCM: Low-memory Streaming Prefilters for Biological Sequencing Data

C. Titus Brown

July 2012

Summary

We will soon be able to exhaustively sequence the DNA and RNA of entire communities of bacteria, as well as every individual cell of a tumor. Both of these very different *applications* of sequencing share in the need to rapidly and efficiently sort through large amounts of noisy sequence data (dozens to 100s of terabases) to separate signal from noise and produce biological insight. However, current bioinformatic approaches for extracting information from this data cannot easily handle the vast amounts of data being acquired.

The primary challenges in processing this sequence data are twofold: the relatively high error rate of 0.1-1%, per base, and the volume of data we can now easily acquire with sequencers such as Illumina HiSeq. For years, sequencing capacity has been doubling every 6 months – significantly faster than compute capacity. Since almost all extant bioinformatic analysis approaches require multiple passes across the primary data, and many analysis algorithms have not been parallelized, **bioinformatic analysis capacity continues to lag ever further behind data generation capacity**. In addition, many of the existing software packages cannot easily be retooled to take advantage of manycore or GPU algorithms, and hence will not take advantage of expected advances in compute capacity and cyberinfrastructure

We propose to develop and implement novel streaming approaches for lossy compression and error correction in shotgun sequencing data. Our algorithms are few-pass (< 2), require no sample-specific information, and can be implemented in fixed or low memory; moreover, they are amenable to parallelization and can run efficiently in manycore environments. When implemented as a prefilter to existing analysis packages, our approaches will eliminate or correct the majority of errors in data sets, dramatically reducing the computational space and time requirements for downstream analysis using existing packages. Moreover, we will provide novel capability by extending error correction approaches to mRNAseq and metagenomic data sets.

Intellectual Merit: We will develop a range of algorithms for space- and time-efficient compression and error correction of short-read DNA and RNA sequence data. These strategies will substantially increase the scalability of many downstream analysis applications, ranging from community analysis of metagenomes to resequencing analysis of humans. We will provide analyses describing the tradeoffs between space and time efficiency and sensitivity, and deliver tested, documented reference implementations of our approaches that can be used by the community for practical evaluation and incorporation into analysis tools. Our approaches will significantly impact short-read sequence analysis by introducing efficient and effective streaming approaches to the two most common types of short-read analysis, mapping and assembly.

Broader Impacts: As with our previous work, we will maximize the utility and reusability of our approaches by: publishing in open-access journals using the ipython notebook "executable paper" format to maximize reproducibility; making our software available under a BSD-like open source license, on github.com, with automated tests and documentation; providing tutorials and accessible online discussions of our approach; and blogging regularly about our work. We will also engage with the authors of potential downstream analysis packages to ensure that our algorithms and output formats interface well with their software. The PI also periodically runs workshops on hands-on sequence analysis techniques for biology graduate students and summer undergraduates from biology majors, in which our approaches will be discussed in the context of major bioinformatics challenges.

Background and Significance

The vast increase in DNA sequencing capacity over the last decade is quickly turning biology into a data-intensive science. Areas as diverse as human medicine, microbial ecology, and evolutionary developmental biology are undergoing a rapid transformation as DNA and RNA sequencing becomes quick, easy, and inexpensive[Metzker, 2010].

This transformation is not without its costs and challenges. The general lack of computational culture in many areas of biology, including molecular and organismal biology, has left many biologists on the other side of a training gap – unable to make full use of this new data. A tandem challenge is that the size and cost of these sequencing data sets are dramatically outpacing computational capacity, making it ever more difficult to complete an analysis [Stein, 2010]. Faster and more efficient computational tools are critically important, not only to make better use of existing compute capacity but also to democratize sequence analysis by enabling individual labs, and biologists within those labs, to make use of the data they are generating. Several recent papers have shown that the primary data analysis now costs significantly more than simply generating the data in the first place [Angiuoli et al., 2011, Sboner et al., 2011].

These large data sets offer the potential for insight on an incredibly wide range of biological problems. Sequencing is increasingly serving studies in biomedical diagnostics, trait association studies, gene expression analysis, population ecology and speciation, experimental evolution, drug resistance, environmental change, complex microbial ecosystems, and developmental biology, as well as basic molecular biology[Metzker, 2010]. All of these fields make use of sequence in different ways, and all of them are drowning in sequence data.

Sequence analysis has now settled on a small number of primary workflows. For biomedical and agricultural research, genome resequencing analysis detects genetic markers that are associated with various phenotypes, including genetic diseases and cancer mutations [Meyerson et al., 2010], while digital RNA sequencing "counts" gene expression and can discover novel genes and gene fusions [Trapnell et al., 2010]. In ecology, longitudinal and latitudinal studies of population structure focus on targeted resequencing to find markers, as well as *de novo* assembly of entire genomes and transcriptomes [Ekblom and Galindo, 2011]. In microbial ecology, researchers are focused on targeted gene analysis and *de novo* assembly of genomes and transcriptomes [Wooley et al., 2010]. And in evolution, both resequencing analysis and *de novo* assembly-based approaches discover novel mutations and genome rearrangements [Barrick and Lenski, 2009]. While there are many additional uses for next-gen sequencing including ChIP-seq and others, in practice the majority of analyses rely on two underlying approaches: *mapping* sequence reads back to a reference sequence to identify variants or count reads, and constructing a new reference sequence based solely on reads using de novo assembly [Metzker, 2010, Miller et al., 2010]. By improving the efficiency of mapping and *de novo* assembly, we could improve essentially all sequence analysis.

The data set sizes involved are immense, and growing. As exemplars, we consider two extremes: single-cell tumor resequencing, and environmental microbial community analysis. For single-cell human tumor cell resequencing, our goal is to detect novel mutations and genome rearrangements in many individual cells [Navin et al., 2011]. This will eventually involve the resequencing of thousands of human genomes per sample, which must be sequenced with high sensitivity; in practice, this requires 10x to 100x coverage of a 6 Gbp diploid genome, or 60-600 Gbp per cell – at a minimum, then, 60 Tbp for 1000 cells. Microbial community analysis is in similar straits: to thoroughly sample the microbes present in a gram of soil, we estimate that 50 Tbp of sequencing must be generated in order to see potential keystone species at the observed frequency of 1 in a million [Brown and Tiedje, 2011]. These data sets are already coming – for a recently funded Amazon Rainforest Microbial Observatory Community Sequencing Project at JGI, we are obtaining a Tbp of sequence in exploratory sequencing for one transect with 10 samples.

The size of these data sets is, in some sense, surprising. Why are we generating so much data from human samples, for example, where we already know 99% or more of the source genome? And why do

soil samples require so much sequencing? The answer is rooted in the nature of the dominant sequencing approach, known as shotgun sequencing, in which sequence reads are sampled randomly from the population of DNA [Myers et al., 2000]. For example, an Illumina HiSeq machine can generate over 6 billion individual sequence reads of length 100 from a single DNA sample. However, the ability to target this sequencing at specific genomes or loci within the genomes is limited by throughput of experimental techniques: techniques such as array capture can enrich for known sequences, and a panoply of experimental isolation or enrichment approaches can be applied, but these are expensive, technically tricky, and limited to specific uses [Turner et al., 2009, Mamanova et al., 2010]. In practice it is faster and less complicated to simply sequence everything and let bioinformaticians sort it out (see e.g. [Cooper and Shendure, 2011]. Correspondingly, in shotgun sequencing the sensitivity of detection of a signal is limited by the dilution of that signal within the sample, so for example to detect one nucleotide variant in the human genome we must sequence the entire genome to 10-100x, or 30-300 Gbp, depending on the desired confidence!

The high error rates of sequencing are another complication. Illumina, perhaps the dominant sequencing technology, has per-nucleotide error rates of up to 1%, so on average each 100-bp read has one error in it. To discern true sequence variants from errors, typically the entire data set must be analyzed, e.g. to find statistically significant consensus signals at each location

[Cooper and Shendure, 2011]. Approaches for doing so are typically rather heavyweight, involving either large on-disk files or substantial working memory usage [Li et al., 2009, Iqbal et al., 2012] [Conway and Bromage, 2011, Kelley et al., 2010].

In the face of these growing Big Data sequence analysis challenges, many approaches have been developed to scale mapping and assembly techniques. An array of fast mapping and assembly techniques, largely relying on the Burrows-Wheeler transform and downstream indexing approaches, have made it possible to map 2 million reads in 6 minutes, and assemble human genomes in under 200 GB of RAM [Langmead and Salzberg, 2012, Gnerre et al., 2011, Simpson and Durbin, 2012]. Efficient error correction techniques have further improved assembly and mapping efficiency

[Pevzner et al., 2001, Kelley et al., 2010]. A wide variety of lossless approaches to compress raw sequencing data have been developed as part of the Pistoia Sequence Squeeze competition (unpublished). And even newer sparse graph techniques continue to be developed [Ye et al., 2012].

More recently, several approaches to scalable sequence compression and analysis have emerged that are based on efficient probabilistic data structures, online or streaming algorithms, or alternative compute architectures [Muthukrishnan, 2005, Shi et al., 2010]). For mapping and mRNAseq quantification, eXpress provides an online implementation of an EM algorithm for estimating abundances [Pachter, 2012]. The "quip" software uses d-left counting bloom filters to efficiently and losslessly compress raw sequence files and quality scores [Jones et al., 2012]. Our group has recently published a compressible graph representation using Bloom filters that enables efficient storage and partitioning of metagenome assembly graphs, and has been extended by others to enable human genome assembly in under 6 GB of RAM [Pell et al., 2012, Chikhi and Rizk, 2012]. One intriguing result is that when compared to maximally efficient entropic encoding, probabilistic data structures can yield significant improvements over exact representations; for example, storing an implicit de Bruijn graph can be done 10-fold more efficiently with Bloom filters than with any possible exact data structure [Pell et al., 2012, Conway and Bromage, 2011].

However, despite these advances, the central Big Data challenge remains in short-read sequence analysis. Almost all extant short-read resequencing analysis tools and assemblers depend on multiple passes of large short-read data sets. As data sizes continue to increase – Illumina is rumored to be planning a 1 Tbp/week upgrade within the next 6 months! – we cannot merely rely on the existing algorithmic strategies to scale. Yet, at the same time, a large body of effective software has already been developed to deal with short-read sequencing data. How to use these tools to process increasingly large amounts of sequencing data, in the service of addressing biological questions, is one of the most critical challenges facing biology today.

Research Plan

Preliminary Results

The goal of our bioinformatics research is to make *de novo* assembly and resequencing analysis fast, efficient, and available on a wide range of cyberinfrastructure and for a wide range of biological samples. We are focused on short-read data from Illumina machines because it offers by far the deepest sampling: many of our scientific problems, such as mRNAseq studies of non-model organisms, splice variant analysis of diseases, and shotgun metagenomics, depend on sensitive quantitative analysis of deep sampling [Metzker, 2010]. Moreover, even the current long-read technologies such as PacBio make use of Illumina sequence for error correction [Koren et al., 2012]. Illumina is also by far the biggest challenge in terms of scalability: the recent growth in sequence data generation has been largely driven by Illumina, and labs routinely generate 2-300 Gbp in a single experiment.

In this context, we have been investigating efficient data structures and algorithms for next-gen sequence data analysis for several years. In large part because of the youth of the field – extremely large volumes of sequence data only started to become readily available within the last 5 years – there are many opportunities for improved analysis strategies. Our work has focused particularly on **sketch data structures**, ultra-efficient data structures for representing some subset of information; on **streaming** and **online** algorithmic approaches that can efficiently process large volumes of data; on lossy compression and error detection within large data sets; and on **prefiltering approaches** that scale or improve downstream approaches by **compressing or correcting their input data**, without requiring reimplementing existing software. One particularly relevant aspect of our work has been our focus on scaling sequence assembly approaches into the cloud: much of our work has been motivated by our own perennial lack of access to sufficiently large dedicated compute infrastructure.

We have previously published a compression approach for de Bruijn assembly graphs with which we scaled down memory usage in metagenome assembly by an order of magnitude [Pell et al., 2012]. Below, we describe an orthogonal suite of prefiltering approaches that enable dramatic and substantial increases in analysis efficiency by taking advantage of the deep sampling and well-mixed reads from short-read shotgun sequencing.

The basic problems

Illumina machines currently produce millions to billions of DNA reads, 100-150 bp sequences of DNA sampled from a collection of DNA or RNA molecules; these reads can be individual, or "paired" – taken from either end of a single, longer sequence of a specified size [Metzker, 2010]. The reads also generally have some basic per-base quality information associated with them.

The dominant sample preparation strategy for generating these reads is known as "shotgun sequencing", in which the biological sample is fragmented randomly and then sampled at random [Myers et al., 2000]. With deep enough sampling, quantitative information about the abundances of molecules in the sample can be inferred, and entire genomes can be reconstructed computationally [Trapnell et al., 2010, Miller et al., 2010]; in fact, all of the genomes sequenced in the last 10 years have been sequenced using this approach. The advantages of shotgun sequencing are many: sample preparation is generally cheap and easy, organisms need only to be available in sufficient quantity to feed them into a sample preparation, and depth of sampling is limited only by the money available. The good news is that sequencing is growing cheaper and cheaper; the bad news is that this is resulting in more and more data analysis challenges!

The two primary analysis strategies applied to these reads are *mapping* and *de novo assembly* (see Figure 1). In mapping, reads are searched against a reference sequence to identify their likely source location; all of the reads can then be "piled up" against the reference sequence to identify potential sequence variants, or they can be counted to quantitate the presence of the reference sequence in the original sample, e.g. in studies of gene expression or allele frequency [Trapnell et al., 2010, Cooper and Shendure, 2011]. In terms

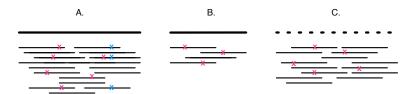


Figure 1: Mapping and assembly are the two most common sequence analysis challenges for short-read sequencing. In mapping (see (A) and (B) the reads (short black lines) are aligned against known reference sequence in the presence of errors (red Xs) and true sequence variants (blue Xs). One purpose of mapping is to distinguish true sequence variants from errors, and it can also be used to estimate the relative abundance of (A) and (B) in the population of molecules being sequenced. For *de novo* assembly (see (C)) the source sequences are unknown and must be reconstructed using overlaps between the sequencing reads; errors in sequencing complicate this process as well.

of scaling, mapping is generally amenable to scatter-gather scaling approaches like MapReduce, because the reference sequence is usually small enough to be distributed to multiple machines, and reads can be mapped individually and then gathered for downstream variant or quantitative analysis. One major computational challenge in mapping is in error correction: the ability of mappers to find the correct reference location is reduced 5-15% by even a fairly low error rate [Kelley et al., 2010], and error correction is not amenable to simple distributed processing; online error correction approaches do not yet exist in the literature, to our knowledge.

The second analysis strategy used is *de novo* assembly, which is used in non- or semi-model systems where no high quality reference sequence is available [Miller et al., 2010]. This includes many plants, animals, and microbes of ecological and evolutionary interest, as well as many host-associated microbial communities. The ultimate goal of assembly is to reconstruct the source reference sequence(s) in the sample, be they genomes or transcriptomes from individuals or populations; this is generally done by building "contigs", or contiguous DNA sequences, based on overlaps between shotgun sequencing reads. *De novo* assembly is a notoriously challenging offline problem that generally involves the storage and analysis of large graphs; it has, so far, been intrinsically viewed as an all-by-all analysis problem that is not amenable to distribution to the lack of locality in the graph. The current dominant assembly paradigm is to use de Bruijn graphs to build contigs by breaking reads down into easily hashed fixed-length sequences, or k-mers, that can then be connected by overlap and traversed as a graph [Compeau et al., 2011]. De Bruijn graphs have the clear advantage of scaling in memory usage with the number of k-mers present in the data as opposed to the data set size; unfortunately, de Bruijn graphs are sensitive to errors in the original reads, and often the majority of the k-mers present in de Bruijn graphs are due to errors in the original reads [Conway and Bromage, 2011].

Error detection, removal, and/or correction is thus central to both mapping and assembly approaches, and many packages exist for doing error trimming or correction (see references in [Kelley et al., 2010]). K-mer based approaches have been especially fruitful: most use an approach first introduced as the Spectral Alignment Problem, in which high abundance k-mers from deeply sequenced samples are considered "trusted", and low-abundance k-mers that likely overlap with errors are trimmed or corrected to match the trusted k-mers [Pevzner et al., 2001]. However, error detection and correction approaches are generally offline: they require multiple passes across the data, and hold large amounts of data in memory. This has become an increasing challenge for short-read data sets, which are outpacing compute capacity, and in particular are outgrowing readily available memory [Kelley et al., 2010].

Efficiently and effectively dealing with these vast sequencing data sets is an ongoing challenge. In addition to the basic cultural and technological challenges discussed in the introduction, many biologists cannot analyze even relatively small samples on the compute infrastructure available to them. This is partly because the problems are hard, and also partly because most existing bioinformatic analysis software has been designed and developed at centers with significant cyberinfrastructure, while most biologists do not have

ready access to large CI. (A reviewer of our digital normalization paper, discussed below, stated "this looks like a good approach for people who don't have the money to buy real computers", which highlights the problem perfectly...) Efficiency in computation, especially with respect to memory, has therefore been the focus of our bioinformatic research.

A memory-efficient data structure for counting k-mers.

We started by developing and implementing an efficient probabilistic data structure for counting k-mers. The counting method is based on our successful use of Bloom filters to represent de Bruijn graphs with high efficiency [Pell et al., 2012], and is essentially identical to a CountMin Sketch or Counting Bloom filter [Cormode and Muthukrishnan, 2005]. Briefly, to increment the count for a k-mer, we hash it into multiple hash tables, and increment the corresponding entry in each table; then, to retrieve the count for a given k-mer we select the minimum count across all of the hash entries. This results in a very memory efficient data structure, albeit one with slightly incorrect counts: if multiple k-mers hash to the same location, then an incorrect count may be retrieved for each of those k-mers. Nonetheless, because NGS data sets are dominated by low-count k-mers from sequencing errors, these miscount values are generally low [Melsted and Pritchard, 2011]; also (Zhang and Brown, in prep). Both theoretically and practically our implementation is substantially lower memory than other k-mer counting implementations.

Unlike exact k-mer counting packages, our CountMinSketch's memory usage is also independent of the k-mer size used; this is because it does not need to store k-mers explicitly, but rather uses only a hash. Our current implementation supports up to k=32, although we are working to extend this to arbitrary k.

Our k-mer hashing, counting, and graph analysis implementation is written in C++ and wrapped in Python, and available on github under a BSD license: github.com/ged-lab/khmer/. It has been used in [Pell et al., 2012] as well as digital normalization, discussed below, and is currently in use by several dozen groups. The khmer package comes with documentation, tutorials, and automated tests (Zhang and Brown, in prep), and has a community mailing list.

A streaming data reduction approach for prefiltering reads for de novo assembly

Driven by the need to assemble multiple very large genomic, metagenomic, and mRNAseq data sets, we next developed a single-pass streaming algorithm for lossy compression of Illumina data prior to assembly [Brown et al., 2012]. Our approach, which we term "digital normalization", relies on an online construction of a de Bruijn graph to select a subset of informative reads using locus-specific graph analysis. The key observation that lies behind digital normalization is that most short-read sequencing data sets are massively redundant, because they contain many reads that have been sampled from overlapping locations; by developing a simple estimator of sampling depth, we can detect and eliminate much of this redundancy online. Our current implementation relies on a simple fixed-memory estimator of coverage, median k-mer count per read; see [Brown et al., 2012] for details.

The core algorithm is a single-pass approach in which each read is decomposed into k-mers, which are then sorted by their abundance within the retained read set thus far; the median of this abundance is used as a proxy for locus coverage, and examined to see if it is above a specified abundance cutoff C. If it is above C, then the read is discarded as redundant, and otherwise the read is judged informative and added to the retained read set. This results in a single-pass streaming algorithm that retains a subset of reads that is informative for assembly.

Digital normalization ("diginorm") is an extraordinarily effective data reduction technique. On a typical 100x E. coli sequencing sample for which 100x coverage has been obtained (e.g. 5m Illumina reads of length 100), we can eliminate 95% of the reads prior to *de novo* assembly and achieve an essentially identical assembly based on the remaining 5x coverage. For high-coverage mRNAseq and single-cell sequencing data sets, data reduction rates approach 98% or more, due to the presence of many very high abundance components in the data set.

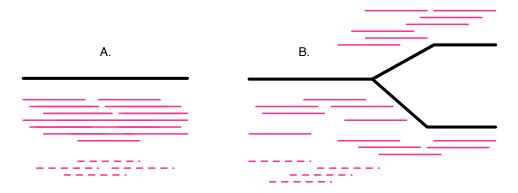


Figure 2: Digital normalization is an efficient approach for locus-specific downsampling of shotgun data sets. Given reads (short red and black lines) from multiple unknown source sequences, digital normalization chooses reads that provide coverage for independent loci up to a specified point (here, a coverage of 5; solid red lines) and discards reads past that coverage as redundant (dashed red lines). This allows less expensive reconstruction of the source sequences by using a only subset of the data. Both (A) simple and (B) more complex graph structures are retained.

Diginorm is also extraordinarily effective at removing errors. Because diginorm selects only a subset of reads to represent each region of the de Bruijn graph, only errors within those reads are retained; errors within rejected reads are discarded. This leads to a dramatic reduction in the number of errors in both simulated and real data.

Between the data reduction and error removal, digital normalization can scale genome assembly by a factor of 100x or more in both time and space, although this of course depends on the application and the data set; see Table 1 for effects on genomes, and [Brown et al., 2012] for details. Moreover, diginorm scales assembly without losing information: comparisons of assemblies from unnormalized data with assemblies from normalized data show near-identity (> 98%), and k-mer composition is nearly identical.

One significant point is that digital normalization changes the scaling behavior of assembly: the number of k-mers in the data, and hence the memory required to assemble it, no longer scales with the data set size; instead, it scales with the underlying complexity of the sample being sequenced, together with a dependency on some power of the error rate due to the accumulation of multiple errors in single reads. Another significant point is that digital normalization and derivative approaches are *always* lower memory than offline approaches, because they never collect the majority of errors - they are either discarded (by diginorm) or fixed (by error correction; below). Finally, digital normalization need only be done once, following which parameter exploration can be done with downstream analysis packages.

Table 1: Three-pass digital normalization reduces assembly time and memory. Digital normalization required a single pass in 2 GB of RAM.

Data set	Assembly time pre/post	Assembly memory pre/post
E. coli	1040s / 63s (16.5x)	11.2gb / 0.5 gb (22.4x)
S. aureus single-cell	5352s / 35s (153x)	54.4gb / 0.4gb (136x)
Deltaproteobacteria single-cell	4749s / 26s (182.7x)	52.7gb / 0.4gb (131.8x)

Critically, digital normalization is a *prefilter*: it does not perform any assembly itself, although it is trivial to implement within the context of an assembler such as Velvet [Zerbino and Birney, 2008]. It simply a subset of reads that can then be fed into any downstream application. Because digital normalization reduces data using basic principles of de Bruijn graphs, we have found that it performs well with the majority of commonly used assemblers. We are working to thoroughly evaluate digital normalization on ALLPATHS-

LG, SOAPdenovo, and a variety of other assemblers [Gnerre et al., 2011, Li et al., 2010].

We have used digital normalization successfully on microbial and mRNAseq data sets [Brown et al., 2012], small and large metagenomes, and several nematode genomes (Schwarz, Brown, and Sternberg, unpublished). Our crowning achievement thus far has been to reduce a 300 Gbp soil metagenome assembly from needing 3 TB of RAM to needing only 300 GB of RAM when using digital normalization and partitioning (Howe and Brown, in preparation); but our own experiences and those of collaborators and the community suggest that digital normalization can provide solution to the majority of assembly scaling problems.

We are currently working to improve the speed of digital normalization in khmer. Our current implementation is single-threaded and has not been optimized for multi-core CPUs. However, because it relies only on hashing, it can easily be parallelized: preliminary results from introducing spinlocks into the counting structure and using threads to load and insert the data suggest that on commodity Dell machines our implementation becomes I/O bound with 2-4 threads.

Despite the effectiveness of digital normalization, it does have a number of drawbacks. One significant challenge is that in normalizing the data to a standard coverage, it eliminates quantitative information that would allow the data to be used for e.g. transcript counting. While this information can be retrieved from the prefiltered data, this can be inconvenient given the large volumes of data involved. This change in quantitative information also can cause problems in running assemblers, which rely on the Lander-Waterman coverage statistics to determine heuristic cutoffs [Lander and Waterman, 1988]. Finally, the current implementation also eliminates repeats and truncates the very ends of contigs due to sampling bias, which we address below.

A streaming few-pass error detection method based on digital normalization.

We have further adapted the digital normalization algorithm to detect and trim likely sequencing errors within short-read data sets. Our method builds on the error elimination pass described in the diginorm paper, in which low-abundance k-mers are after normalization to a specific coverage. Essentially, we combine the online graph-based coverage estimation used by digital normalization with the k-mer abundance approach commonly used to detect errors [Pevzner et al., 2001, Kelley et al., 2010]. These approaches observe that because errors are random and uncorrelated with the true sequence, k-mers containing errors will generally be low abundance; thus, in high-coverage data sets, low-abundance k-mers can be used to identify errors (see Figure 3). This has proven to be extremely effective for error detection and correction in genomic sequence data sets, but has not been as useful in mRNAseq and metagenomics, where uniform coverage assumptions do not generally hold [Keegan et al., 2012, Grabherr et al., 2011]. Moreover, current approaches are multipass: all of the k-mers in sequencing reads must be counted first, and only then can the reads be analyzed in the context of those counts.

We have implemented a < 2-pass approach that uses the diginorm algorithm to determine when a particular region of the de Bruijn graph has accrued enough coverage to be used to do error detection (Figure 4). For example, once a region of the graph has 20x coverage, then reads from this region that contain k-mers with abundance of 1 are likely to be erroneous and can be trimmed or (potentially) corrected. Because we must observe some minimum number of reads in order to accumulate sufficient coverage to do error detection, at least some of the reads must be analyzed more than once, unlike diginorm; however, for deep data sets, the majority of reads will only need to be seen once, making it a < 2 pass approach. It is also streaming and online.

Interestingly, this algorithm provides solutions for a number of the problems with diginorm and existing error correction approaches: it preserves "correct" reads, so the filtered data can be used in quantification; it does not rely on uniform coverage to determine k-mer abundance cutoffs, and so can be used on genomic, metagenomic, and mRNAseq data sets; and it is few-pass. Moreover, because we have implemented the approach using khmer's CountMin Sketch data structure, it remains extremely efficient in memory use.

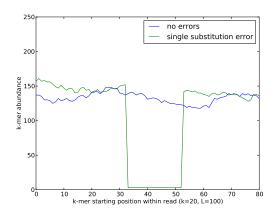


Figure 3: One common strategy for error detection and correction in short reads is to examine the abundances of k-mers chosen from the read within the entire data set. 20-mers taken from 100-base reads with no errors reflect the overall coverage of the data set (blue), while 20-mers that contain errors such as single nucleotide substitutions are novel within the data set and present as low-abundance (green) – here, an error is located at position 42. Our error detection strategy uses this signature to localize likely erroneous bases, while our proposed error correction strategy will seek base changes that result in the most consistent k-mer coverage across the read.

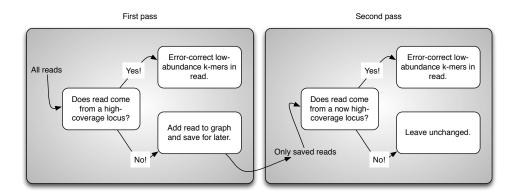


Figure 4: Few-pass error detection and correction. In a first pass, reads that add coverage to a locus below a specified threshold are loaded into the graph but not error-corrected, for lack of sufficient information; reads that belong to already high-coverage loci are immediately error-corrected. In the second pass, those reads that were used to build the graph are inspected to see if their source loci have sufficient coverage to error correct them. For a 100x coverage genomic shotgun sample with a read collection threshold of 20, this approach would be approximately 1.2-pass: about 20% of the reads would be collected on the first pass, and then examined and error-corrected on the second pass.

On both simulated and real genomic and metagenomic data, our current approach identifies and eliminates more than 90% of errors (and the majority of erroneous k-mers) using a collection threshold of 20 and a hardcoded error threshold of 1. We propose to improve this below.

Unfortunately, this is not a data reduction approach like digital normalization, because quantitative information needs to be retained for downstream mapping and variant analysis approaches. However error removal (and error correction) can dramatically decrease memory requirements for assembly and assembly-graph based algorithms (e.g. see [Simpson and Durbin, 2012] and [Iqbal et al., 2012]), as well as result in increased sensitivity of mapping [Kelley et al., 2010].

Summary of Preliminary Results

We have built an efficient k-mer counting implementation, based on the adaptation of a well-known probabilistic data structure, the CountMin Sketch. We have also demonstrated an effective compression approach that relies on locus-specific downsampling of genomic data using only a single approach. Moreover, we have extended this algorithm to implement k-mer based error detection within a streaming framework. The following Specific Aims will build on this framework to provide a range of useful Big Data reduction and error correction approaches.

Specific Aims:

Aim 1. Improve and extend digital normalization, an efficient streaming strategy for lossy compression of DNA sequence, to enable more scalable downstream analyses. Digital normalization provides an extraordinarily efficient and effective approach to data reduction prior to *de novo* assembly. It has been successfully applied to microbial genomes, to transcriptomes, and to metagenomes, as well as to some small eukaryotic genomes, and a number of groups are using it on a regular basis. However, there are several challenges that must be overcome in order for a diginorm-based approach to become more widely used. These include adding handling of repeats by using paired-end and mate-pair data types; developing an understanding of how coverage mean and variance is transformed by digital normalization; and adapting diginorm to sensitively retain data from diploid genomes and mixed populations.

Subaim 1a. Incorporate paired-end and mate-pair information into the digital normalization algorithm and implementation. Digital normalization currently ignores paired-end and mate-pair information: if one end of a pair has high coverage, it may be discarded as redundant. However, pairing information is used by assemblers to build scaffolds by linking contigs across repetitive regions, and it is an important step in the assembly of repeat-rich eukaryotic genomes.

We will provide an option in the digital normalization algorithm that considers pairs as either end of a longer read, such that if either end is below the coverage threshold for normalization, both read pairs are kept. This will retain the linkage information between the pairs for downstream assemblers.

To evaluate the effect of repeat retention on assembly of eukaryotic genomes, we will repeat the benchmarking done with the SGA algorithm on the *C. elegans* and human genomes

[Simpson and Durbin, 2012]. Specifically, we will compare ABySS, SGA, Velvet, and SOAPdenovo output and performance on unnormalized *C. elegans* data, data normalized to a coverage of 20 with and without paired end retention, and data normalized to a coverage of 5, with and without paired end retention. A k-mer parameter sweep for the normalized data sets will be conducted between 21 and 71 for each package, choosing the best N50 from each. We will use the same benchmarks as the SGA paper to compare the assemblies against the known reference, including substring coverage, assembly contiguity, and assembly completeness and accuracy. We will also compare k-mer content at k=20 and k=32, as in the diginorm paper, to assess whether any information is being irreversibly lost during the digital normalization procedure. Once we develop an empirical understanding of the best parameters for the smaller *C. elegans* data set, we will repeat the analysis for the far larger human genome data set.

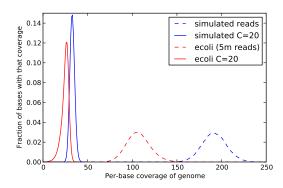


Figure 5: A graph showing the transformation in coverage caused by digital normalization for simulated (blue) and real *E. coli* (red) data. The X axis shows the number of reads covering a given base in the (known) reference genomes, and the Y axis shows the number of bases with that read coverage. Digital normalization shifts the mean and decreases the variance of the original coverage distribution (dashed lines) to the normalized distribution (solid lines). (From Brown et al.)

Subaim 1b. Develop a theoretical model for average and variance of coverage of digital normalized data. Although the basic theory behind digital normalization and our specific implementation using the median k-mer abundance is laid out in [Brown et al., 2012], much of our actual work with digital normalization has been empirical and driven by work with simulated and real data sets. In particular we lack an understanding of the relationship between the k-mer size used, the coverage desired, the read length, and the sensitivity with which low-coverage data is retained. This information is important for assemblers, which rely on theoretical models of coverage and variance around average coverage to determine which paths to traverse and how to identify repetitive sequence [Miller et al., 2010, Zerbino and Birney, 2008].

Empirical observation suggests that the variance of the normalized data is based on the variance of k-mer coverage within each read, which corresponds to the process by which we retain reads with digital normalization. However, we know of no theoretical models for how within-read k-mer coverage varies, although surely it will depend on both coverage and read length.

We will therefore develop a simple theoretical model based around Poisson-random sampling of reads of different lengths of a random genome without error, and compare with simulation results. We will then extend these results with a simple error model, and finally compare these results and performance on real data from known genomes. We successfully used this combined theory + simulation approach in [Pell et al., 2012].

Subaim 1c. Implement retention of diploid and lower-frequency alleles within digital normalization. Digital normalization currently works very well on several microbial genomes and RNAseq, but its effects on genomes and transcriptomes with significant internal variation (from diploid heterozygosity or strain variation) has not been systematically assessed. We would naively expect digital normalization to choose reads randomly from the distribution of true variants, which would lead to the retention of variants in proportion to their true frequency in the data set; this is one reason why digital normalization may eliminate low-frequency splice variants in mRNAseq [Brown et al., 2012]. However, digital normalization does retain low-frequency variants with a slight preference, because they are more likely to present as novel within the

The current implementation of digital normalization uses median k-mer count within a read to estimate its coverage. However, by choosing the coverage of a lower-ranked (and hence lower abundance) k-mer than the median to estimate coverage of a read, it is possible to retain more low-abundance variants while still eliminating many redundant and erroneous reads. For example, simulations of a population sample with a

de Bruijn graph.

1-in-10 minor allele frequency showed that we could easily retain all of the true k-mers while eliminating over 50% of erroneous k-mers and 80% of the reads.

We propose to extend the digital normalization algorithm and theory to sensitively retain diploid (50/50) and lower-frequency variants in shotgun data from genome resequencing and mixed population samples. Specifically, we will extend the theory from Subaim 1b to incorporate read abundance estimators more sensitive to variants than the median, such as the abundance of the k-th ranked k-mer, and then apply it to simulated and real data. We will further provide "tuned" parameter sets that relate retention of alleles at specified minor allele frequencies to digital normalization k-mer size and coverage parameters; this will allow better algorithmic performance and data reduction on e.g. diploid samples with 50/50 alelelic frequencies, while offering explicit tradeoffs on data from more complex samples.

We will evaluate this sensitivity with simulated data as well as with population data from the Cortex resequencing analysis paper [Iqbal et al., 2012]. Briefly, after tuning digital normalization to retain diploid and lower-frequency alleles, we will run our modified normalization procedure with various parameters on the simulated, individual and pooled data from [Iqbal et al., 2012], and then execute Cortex on the retained reads. Variant retention sensitivity will be compared to the known SNP-based calls also used in [Iqbal et al., 2012].

This modification will allow digital normalization error and data reduction to be combined with resequencing analysis packages such as Cortex, which use the de Bruijn graph formalism to identify novel genomic regions with great sensitivity. As with the standard median-count-based digital normalization, quantitative information about alleles will be lost with this approach due to the downsampling; Aim 2 will provide a way to retain this quantitative information.

We note in passing that reference-based subtraction is trivial to implement within this revised digital normalization framework: known reference sequence can simply be loaded in prior to streaming in the reads, and only reads containing alternative alleles will be kept. This also raises the interesting idea of doing online streaming variant calling in future work.

Aim 2. Develop efficient streaming strategies for error detection and error correction of DNA sequencing data. Perhaps the primary challenge in scaling assembly and resequencing analysis of large DNA data sets is efficiently distinguishing signal from noise; current approaches to error detection and correction use high quality but less-than-scalable offline and multipass techniques [Kelley et al., 2010]. In our preliminary results we show that a streaming, memory-efficient few-pass approach can be implemented to efficiently detect errors, and we propose to adapt existing error-correction approaches to use this streaming model. Note: we know of no other streaming or online approaches to error correction in short-read DNA sequencing data.

Subaim 2a. Develop streaming error correction for shotgun sequences using k-mer abundance models. In the Preliminary Results, we showed that it was straightforward to adapt digital normalization to detect errors based on k-mer abundance, providing a streaming few-pass approach to error trimming. We propose to combine this locus-specific error detection approach with the expectation-maximization (EM)-based algorithm used by the Quake error correction model. This will result in a time- and space-efficient error correction algorithm with high accuracy.

The Quake EM algorithm uses a complex model of k-mer coverage to determine likely locations for single-base errors as well as the most probable set of changes to remove the errors. At its core, however, it relies on the same information as most other k-mer based error correction algorithms: correcting errors based on "trusted" high abundance k-mers.

We will reimplement the Quake algorithm within the khmer package, based on its published algorithm. (Note that the Quake license does not permit us to include Quake source code within khmer.) In many ways our implementation will in fact be simpler, as the expected k-mer distribution of "trusted" k-mers can be determined according to the analytic model developed in Subaim 1(b), and the error models will be sharply 1-biased and largely static for digitally normalized data.

One interesting consequence of reimplementing Quake on top of our streaming error detection approach is that the Quake algorithm will now be directly usable on transcriptomic and metagenomic data: high coverage data will be corrected, and low coverage data will be untouched. This is a substantial benefit with significant implications for many data sets!

Another interesting consideration is that, with efficient online error correction, counting applications such as mRNAseq expression analysis could use the de Bruijn graph model to determine transcript abundance. This is a potential future direction for our research.

We will evaluate our approaches on the same data sets used by Quake as well as on the microbial data sets used in [Brown et al., 2012] and the human data sets from Cortex [Iqbal et al., 2012]. This will provide an array of short-read data sets from microbes and humans for which we have solid reference sequences, on which to evaluate sensitivity (% reads mapped) and specificity (% errors introduced).

To evaluate metagenomics and RNAseq error correction that is only possible with our approach, we will use the Human Microbiome Project "mock community" data sets (for metagenomics) and the Trinity data sets for mouse and yeast [HMP, 2012, Grabherr et al., 2011]. In both cases we have excellent reference data sets with which to assess sensitivity and specificity of error corrected reads by using mapping.

Subaim 2b. Implement a variant of q-mer counting within digital normalization. Quake uses a q-mer counting model that takes into account quality values for each base, i.e. metadata about sequence quality. This works by weighting k-mers based on their constituent bases' quality scores.

We will implement q-mer counting by quantizing floating point values within k-mer counting bins. For example, within an 8-bit bin, we can provide quantization down to a decimal point for a coverage cutoff of 20. This will downgrade the effect that k-mers from poor-quality reads have on error correction. Note that this could also increase the quality of digital normalization by explicitly retaining more copies of graph regions represented only by low-quality reads. Evaluation of improvements will be done in comparison with the results achieved in Subaim 2(a).

Subaim 2c. Adapt error detection approaches to preferentially retain minor alleles. While the Quake algorithm performs well on diploid genomes, by design it cannot properly handle lower-frequency variants from mixed population sequencing. This is important for samples where we are DNA limited and must sequence multiple individuals, e.g. in insect studies; and for strain variation analysis in metagenomic data sets, where multiple strains of very similar bacteria may be present in the same sample, with some low-frequency strains possessing novel traits (e.g. drug resistance) due to a few nucleotide changes.

We propose to incorporate the theoretical framework for digital normalization developed in Subaim 1(c) to provide an understanding of how to tune parameters to robustly distinguish sequencing errors from low-frequency alleles above a certain abundance. While there is clearly no way to completely separate errors from low-frequency alleles, even a 50% decrease in errors can yield a significant increase in sensitivity of mapping and resequencing analysis.

Evaluation of sensitivity and specificity will be performed using Cortex and GATK on the Cortex data sets as well as additional data sets taken from the 1000 Human Genomes project [Iqbal et al., 2012, McKenna et al., 2010]. We expect Cortex memory usage to decrease significantly with no degradation in sensitivity or specificity, and we expect no change in sensitivity or specificity with GATK.

Aim 3. Explore algorithm and data structure improvements for HPC and cloud environments. Our current algorithmic approaches use the CountMin Sketch data structure in khmer, which is implemented as a set of large hash tables. Updates and access can be efficiently parallelized for multicore shared-memory machines; our next release of khmer is multithreaded and I/O bound for > 4 threads on commodity Dell hardware. However, for large data sets, and especially for data from samples with high real k-mer diversity such as metagenomes, we are increasingly memory limited. Moreover, existing commodity architectures such as Non-Uniform Memory Access (NUMA) limit the increases in efficiency we can get with parallelism.

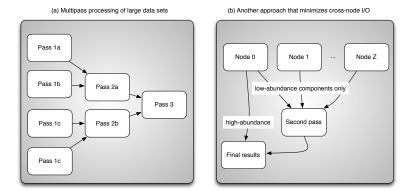


Figure 6: Strategies for distributing diginorm-like approaches. Both strategies rely on the assumption that the data is well mixed. The first strategy, left, divides the data up across multiple machines and performs digital normalization independently in several different passes. This only incrementally reduces the data set sizes and requires substantial communication between nodes at the end of each pass, but is straightforward to implement. The second strategy, right, uses Z independent workers, of which all but the first node discards high-abundance reads; in the second pass, all of the low-abundance reads are combined and normalized, and then combined with the high-abundance reads from the first node.

Within our current framework of the CountMin Sketch, we have reached the limits of memory efficiency [Cormode and Muthukrishnan, 2005]. We propose to exploit potential improvements to the underlying data structures as well as benchmark several distributed algorithms to provide options appropriate to HPC environments, as well as low-memory or low-bandwidth distributed environments such as the Amazon cloud.

Subaim 3a. Incorporating d-left Counting Bloom Filters Recently, d-left Counting Bloom Filters (dl-CBFs), improved constructions of CBFs, have emerged that rely on near-perfect "d-left" hashing [Bonomi et al., 2006b]. We propose to implement this as an alternative counting store within khmer, which would simplify the use of khmer on small memory machines. We will implement dlCBFs within khmer as an alternate storage mechanism and evaluate it in the context of the above applications. We are also interested in trying out dlCBFs with dynamic bit reassignment, or ddlCBFs, which allow variable use of memory but are not as efficient under high load [Bonomi et al., 2006a]. We expect that ddlCBFs will be an excellent replacement for our current implementation in lightly loaded counting situations, e.g. transcriptomes and microbial genomes.

Subaim 3b. Progressive filtering algorithms While our goal is to produce efficient few-pass algorithms for data reduction and error correction, in memory-limited situations there should be straightforward multipass approaches to digital normalization-based algorithms (see Figure 6 (a)). The essential idea is to monitor the CBF data structure until it reaches an unacceptably high false positive rate, and then stop loading in new data. The data structure can still be used to filter data for loci with sufficient coverage, however, and for data sets with many high-abundance source components this would still result in substantial data reduction or error-correction. This partly-predigested data set could then be passed on to a larger memory machine for additional processing. We will implement and benchmark this approach in the context of HPC and cloud environments.

Subaim 3c. Distributed digital normalization By construction, large shotgun data sets are well-mixed: reads are in essentially random order. Distributed digital normalization for extremely large data sets could potentially be done efficiently by distributing the data across Z machines, collecting the low and high abundance reads on one designated node, and then collecting only the low-abundance reads on the other Z-1 nodes; see Figure 6 (b). This minimizes the collation of common components across workers by assuming that common components will be collected by the designated node.

We will build a trial implementation and explore the parameter space for Z, per-node memory, and abundance threshold in both HPC and Amazon cloud computing environments.

Summary of Proposed Research Aims

In summary, we propose to extend, improve, and develop several conceptually simple streaming algorithms for prefiltering of sequence data, including an existing lossy compression approach, digital normalization, and a novel error detection and correction approach. We will base data structure and algorithm extensions on simple, well-understood approaches from bioinformatics and other fields. The net output of our work will be a suite of efficient and effective prefiltering implementations that will scale and improve the majority of downstream short-read analysis tools. The effect of these tools will be to dramatically reduce the time and memory required for first-pass error correction and data analysis and reduction in the biological sciences, without affecting sensitivity or specificity of the results.

Broader Impacts and PI's Research Plan

The PI's group is both generating their own data and collaborating with a number of other researchers on data analysis. The research plan described above has emerged from our larger interest in making sequence analysis as broadly available and effective as possible for hypothesis generation in science; accordingly, in addition to our actual research, we engage in many activities, including: doing and teaching effective software development; enhancing reproducibility in research by example; training computationally naive biology students in bioinformatics; generating tutorials and online materials to support training; recruiting and training underrepresented minorities; and contributing to MSU-local and online communities through workshops, courses, and blogging.

Software development, "executable papers", and reproducible research

Dr. Brown has been developing software for over 25 years in commercial, academic, and open source environments; he is the author of a number of open source packages, including several utilities for automated testing in Python. Our group uses version control, automated tests, code coverage analysis of the tests, and continuous integration practices as a daily part of software development, in an attempt to ensure code accuracy. We also teach this mode of development in regularly held workshops at MSU and elsewhere.

In the interests of reproducible science, we have also been making our complete source code and data sets available via github.com and Amazon Web Services such that anyone can replicate our results with their own virtual machine. We have also begun to distribute IPython notebooks [Pérez and Granger, 2007] for data analysis and figure generation, which makes our papers "executable", in the sense that all of the primary computational analyses can be rerun with only a few commands (see e.g. [Brown et al., 2012] for an example).

Outreach, education, and training

Our lab engages in many outreach, education, and training activities. Specifically,

• Dr. Brown runs the yearly MSU Analyzing Next-Generation Sequencing Data workshop. This course is run for grad students, postdocs, and faculty, and has attracted over 300 applicants in 3 years. It is currently funded by an NIH R25 grant (through 2013). Our goal in this workshop is to provide biologists with an opportunity to learn basic bioinformatics skills.

The course materials, at http://ged.msu.edu/angus/, are available under a Creative Commons license permitting remixing and adaptation, and we provide instructions on how to reuse them. Approximately 60,000 unique visitors per year visit the primary site, according to Google Analytics.

This site contains detailed tutorials on how to do *de novo* assembly, mRNAseq differential expression analysis, and variant discovery on the Amazon cloud. Over the past year we have added tutorials

on using digital normalization and partitioning, and we will continue to do so as we develop new techniques.

- For the past two summers we have participated in the Summer Research Opportunity Program for underrepresented minorities. In 2011 we trained two UMs in bioinformatics, and in 2012 we trained three. In 2013 we are proposing a more general model in which we will train a large group of SROP undergrads in general computation, supplanting an ineffective statistics workshop; the students will then go on to their individual research labs, but with regular touchback meetings with our group. This grant will support several such undergrads (see Budget Justification and Work Plan).
- Dr. Brown runs an introductory graduate course on computational science for biologists under the
 auspices of the MSU BEACON NSF STC on "Evolution in Action". The course is teleconferenced
 across U. Idaho, UW Seattle, and UT Austin. This course includes a module focusing on sequence
 analysis considerations for evolutionary biologists and ecologists, where software including our tools
 and approaches are discussed, demonstrated, and then executed by the students.
- Members of the lab, including the PI, regularly participate in "Software Carpentry" workshops focusing on computational science skills (http://software-carpentry.org/). These workshops are held at MSU and elsewhere.
- All publications in the lab are posted to github.com/ged-lab/ and arXiv.org upon submission, and we intend to publish them all as Open Access.
- All source code in the lab is made available via the github.com/ged-lab/ version control archive, under the Open Source BSD license that permits maximal use and reuse of the code.

Capacity Building

We propose to integrate our work on prefiltering data with education and training efforts, with specific respect to using cloud-enabled computing for small-lab/Big Data analysis. This is a natural extension of our existing activities in training and outreach on the cloud and is organic to our proposal above.

In addition to teaching, training, and writing tutorials, we will track expected and actual costs for Amazon cloud usage and include it in our publications.

Evaluation Plan

Details of technology evaluation are included in each individual section. Conveniently, because we are building prefiltering approaches that apply to published analysis packages, for sensitivity and accuracy we will compare our prefiltering approaches to the published results. This will also be combined with cost tracking of the analysis of unfiltered and prefiltered data on HPC and Amazon resources as mentioned above.

Results of Prior NSF Support

Project Proposal Title: Symbiont Separation and Investigation of the Novel Heterotrophic Osedax Symbiosis using Comparative Genomics; NSF 09-23812 (PI Brown); Project Location: Michigan State University; Total Award Amount: \$50k; Starting Date: 01/01/10; Ending Date: 12/31/12.

This project is a collaborative project with Dr. Shana Goffredi at Occidental College, in which we are participating in bioinformatic analysis of MDA-amplified metagenomic samples. These samples originated from a bead-based enrichment of *Oceanospirallales sp.* symbionts taken from an *Osedax* bone eating worm. We received our first Illumina samples in May, and applied the digital normalization technique described above to the data sets. We obtained an est. 85% complete genome assembly of the desired microbe, a better than 2-fold increase over a pre-normalized result. A manuscript on the metabolic analysis of these microbes based on their genome content is in preparation.

Facilities, Equipment, and Other Resources

Facilities

Dr. Brown has both computational and wet lab space.

The wet lab space consists of 1000 sq ft with a hood, benches for 6 students, and several freezers. The lab is equipped for chick embryology work with a fluorescent dissecting scope, two egg benches, and two incubators. The lab is also fully equipped with microcentrifuges, PCR machines, water baths, and a microwave.

The computational space consists of 2500 sq ft shared with another computational biology lab. There are offices for 10 students and postdocs, two faculty, and a large central collaboration space.

Equipment

The lab has three Dell/Intel servers, each with 16 GB of RAM, on which primary development work and testing is done. We also have access to the High Performance Compute facility at MSU, which possesses over 4,000 CPUs in chassis each with up to 24 GB of RAM, and several high memory computers available on a scheduled basis.

References

- [Angiuoli et al., 2011] Angiuoli, S., White, J., Matalka, M., White, O. and Fricke, W. (2011). Resources and costs for microbial sequence analysis evaluated using virtual machines and cloud computing. PLoS One 6, e26624.
- [Barrick and Lenski, 2009] Barrick, J. and Lenski, R. (2009). Genome-wide mutational diversity in an evolving population of Escherichia coli. Cold Spring Harb Symp Quant Biol *74*, 119–29.
- [Bonomi et al., 2006a] Bonomi, F., Mitzenmacher, M., Panigrahy, R., Singh, S. and Varghese, G. (2006a). Bloom Filters via d-left Hashing and Dynamic Bit Reassignment. In Proceedings of the Allerton Conference on Communication, Control and Computing.
- [Bonomi et al., 2006b] Bonomi, F., Mitzenmacher, M., Panigrahy, R., Singh, S. and Varghese, G. (2006b). An Improved Construction for Counting Bloom Filters. In 14th Annual European Symposium on Algorithms, LNCS 4168 pp. 684–695,.
- [Brown et al., 2012] Brown, C., Howe, A., Zhang, Q., Pyrkosz, A. and Brom, T. (2012). A reference-free algorithm for computational normalization of shotgun sequencing data. In review at PLoS One, July 2012; Preprint at http://arxiv.org/abs/1203.4802.
- [Brown and Tiedje, 2011] Brown, C. and Tiedje, J. (2011). Metagenomics: the paths forward, vol. Handbook of Molecular Microbial Ecology II: Metagenomics in Different Habitats,. Wiley.
- [Chikhi and Rizk, 2012] Chikhi, R. and Rizk, G. (2012). Space-efficient and exact de Bruijn graph representation based on a Bloom filter. In WABI p. to appear,.
- [Compeau et al., 2011] Compeau, P., Pevzner, P. and Tesler, G. (2011). How to apply de Bruijn graphs to genome assembly. Nat Biotechnol 29, 987–91.
- [Conway and Bromage, 2011] Conway, T. and Bromage, A. (2011). Succinct data structures for assembling large genomes. Bioinformatics *27*, 479–86.
- [Cooper and Shendure, 2011] Cooper, G. and Shendure, J. (2011). Needles in stacks of needles: finding disease-causal variants in a wealth of genomic data. Nat Rev Genet 12, 628–40.
- [Cormode and Muthukrishnan, 2005] Cormode, G. and Muthukrishnan, S. (2005). An improved data stream summary: the count-min sketch, and its applications. Journal of Algorithms 55.
- [Ekblom and Galindo, 2011] Ekblom, R. and Galindo, J. (2011). Applications of next generation sequencing in molecular ecology of non-model organisms. Heredity (Edinb) *107*, 1–15.
- [Gnerre et al., 2011] Gnerre, S., Maccallum, I., Przybylski, D., Ribeiro, F., Burton, J., Walker, B., Sharpe, T., Hall, G., Shea, T., Sykes, S., Berlin, A., Aird, D., Costello, M., Daza, R., Williams, L., Nicol, R., Gnirke, A., Nusbaum, C., Lander, E. and Jaffe, D. (2011). High-quality draft assemblies of mammalian genomes from massively parallel sequence data. Proc Natl Acad Sci U S A 108, 1513–8.
- [Grabherr et al., 2011] Grabherr, M., Haas, B., Yassour, M., Levin, J., Thompson, D., Amit, I., Adiconis, X., Fan, L., Raychowdhury, R., Zeng, Q., Chen, Z., Mauceli, E., Hacohen, N., Gnirke, A., Rhind, N., di Palma, F., Birren, B., Nusbaum, C., Lindblad-Toh, K., Friedman, N. and Regev, A. (2011). Full-length transcriptome assembly from RNA-Seq data without a reference genome. Nat Biotechnol *29*, 644–52.
- [HMP, 2012] HMP (2012). A framework for human microbiome research. Nature 486, 215–21.

- [Iqbal et al., 2012] Iqbal, Z., Caccamo, M., Turner, I., Flicek, P. and McVean, G. (2012). De novo assembly and genotyping of variants using colored de Bruijn graphs. Nat Genet 44, 226–32.
- [Jones et al., 2012] Jones, D., Ruzzo, W., Peng, X. and Katze, M. (2012). Compression of next-generation sequencing reads aided by highly efficient de novo assembly. Preprint at http://arxiv.org/abs/1207.2424.
- [Keegan et al., 2012] Keegan, K., Trimble, W., Wilkening, J., Wilke, A., Harrison, T., D'Souza, M. and Meyer, F. (2012). A Platform-Independent Method for Detecting Errors in Metagenomic Sequencing Data: DRISEE. PLoS Comput Biol *8*, e1002541.
- [Kelley et al., 2010] Kelley, D., Schatz, M. and Salzberg, S. (2010). Quake: quality-aware detection and correction of sequencing errors. Genome Biol 11, R116.
- [Koren et al., 2012] Koren, S., Schatz, M., Walenz, B., Martin, J., Howard, J., Ganapathy, G., Wang, Z., Rasko, D., McCombie, W., Jarvis, E. and Phillippy, A. (2012). Hybrid error correction and de novo assembly of single-molecule sequencing reads. Nat Biotechnol.
- [Lander and Waterman, 1988] Lander, E. and Waterman, M. (1988). Genomic mapping by fingerprinting random clones: a mathematical analysis. Genomics 2, 231–9.
- [Langmead and Salzberg, 2012] Langmead, B. and Salzberg, S. (2012). Fast gapped-read alignment with Bowtie 2. Nat Methods 9, 357–9.
- [Li et al., 2009] Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G. and Durbin, R. (2009). The Sequence Alignment/Map format and SAMtools. Bioinformatics 25, 2078–9.
- [Li et al., 2010] Li, Y., Hu, Y., Bolund, L. and Wang, J. (2010). State of the art de novo assembly of human genomes from massively parallel sequencing data. Hum Genomics 4, 271–7.
- [Mamanova et al., 2010] Mamanova, L., Coffey, A., Scott, C., Kozarewa, I., Turner, E., Kumar, A., Howard, E., Shendure, J. and Turner, D. (2010). Target-enrichment strategies for next-generation sequencing. Nat Methods *7*, 111–8.
- [McKenna et al., 2010] McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M. and DePristo, M. (2010). The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res 20, 1297–303.
- [Melsted and Pritchard, 2011] Melsted, P. and Pritchard, J. (2011). Efficient counting of k-mers in DNA sequences using a bloom filter. BMC bioinformatics 12, 333.
- [Metzker, 2010] Metzker, M. (2010). Sequencing technologies the next generation. Nat Rev Genet 11, 31–46.
- [Meyerson et al., 2010] Meyerson, M., Gabriel, S. and Getz, G. (2010). Advances in understanding cancer genomes through second-generation sequencing. Nat Rev Genet 11, 685–96.
- [Miller et al., 2010] Miller, J., Koren, S. and Sutton, G. (2010). Assembly algorithms for next-generation sequencing data. Genomics *95*, 315–27.
- [Muthukrishnan, 2005] Muthukrishnan, S. (2005). Data streams: algorithms and applications. Foundations and trends in theoretical computer science, Now Publishers.

- [Myers et al., 2000] Myers, E., Sutton, G., Delcher, A., Dew, I., Fasulo, D., Flanigan, M., Kravitz, S., Mobarry, C., Reinert, K., Remington, K., Anson, E., Bolanos, R., Chou, H., Jordan, C., Halpern, A., Lonardi, S., Beasley, E., Brandon, R., Chen, L., Dunn, P., Lai, Z., Liang, Y., Nusskern, D., Zhan, M., Zhang, Q., Zheng, X., Rubin, G., Adams, M. and Venter, J. (2000). A whole-genome assembly of Drosophila. Science 287, 2196–204.
- [Navin et al., 2011] Navin, N., Kendall, J., Troge, J., Andrews, P., Rodgers, L., McIndoo, J., Cook, K., Stepansky, A., Levy, D., Esposito, D., Muthuswamy, L., Krasnitz, A., McCombie, W., Hicks, J. and Wigler, M. (2011). Tumour evolution inferred by single-cell sequencing. Nature 472, 90–4.
- [Pachter, 2012] Pachter, L. (2012). Models for transcript quantification from rna-seq. Preprint at http://arxiv.org/abs/1104.3889v2.
- [Pell et al., 2012] Pell, J., Hintze, A., Canino-Koning, R., Howe, A., Tiedje, J. and Brown, C. (2012). Scaling metagenome sequence assembly with probabilistic de bruijn graphs. Accepted at PNAS, July 2012; Preprint at http://arxiv.org/abs/1112.4193.
- [Pérez and Granger, 2007] Pérez, F. and Granger, B. E. (2007). IPython: a System for Interactive Scientific Computing. Comput. Sci. Eng. *9*, 21–29.
- [Pevzner et al., 2001] Pevzner, P., Tang, H. and Waterman, M. (2001). An Eulerian path approach to DNA fragment assembly. Proc Natl Acad Sci U S A 98, 9748–53.
- [Sboner et al., 2011] Sboner, A., Mu, X., Greenbaum, D., Auerbach, R. and Gerstein, M. (2011). The real cost of sequencing: higher than you think! Genome Biol 12, 125.
- [Shi et al., 2010] Shi, H., Schmidt, B., Liu, W. and Muller-Wittig, W. (2010). A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware. J Comput Biol 17, 603–15.
- [Simpson and Durbin, 2012] Simpson, J. and Durbin, R. (2012). Efficient de novo assembly of large genomes using compressed data structures. Genome Res 22, 549–56.
- [Stein, 2010] Stein, L. (2010). The case for cloud computing in genome informatics. Genome Biol 11, 207.
- [Trapnell et al., 2010] Trapnell, C., Williams, B., Pertea, G., Mortazavi, A., Kwan, G., van Baren, M., Salzberg, S., Wold, B. and Pachter, L. (2010). Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nat Biotechnol 28, 511–5.
- [Turner et al., 2009] Turner, E., Ng, S., Nickerson, D. and Shendure, J. (2009). Methods for genomic partitioning. Annu Rev Genomics Hum Genet 10, 263–84.
- [Wooley et al., 2010] Wooley, J., Godzik, A. and Friedberg, I. (2010). A primer on metagenomics. PLoS Comput Biol 6, e1000667.
- [Ye et al., 2012] Ye, C., Ma, Z., Cannon, C., Pop, M. and Yu, D. (2012). Exploiting sparseness in de novo genome assembly. BMC Bioinformatics *13 Suppl 6*, S1.
- [Zerbino and Birney, 2008] Zerbino, D. R. and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res. 18, 821–829.