NOVEL COMPUTATIONAL APPROACHES TO INVESTIGATE MICROBIAL DIVERSITY

By

Qingpeng Zhang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science — Doctor of Philosophy

2015

ABSTRACT

NOVEL COMPUTATIONAL APPROACHES TO INVESTIGATE MICROBIAL DIVERSITY

By

Qingpeng Zhang

Species diversity is an important measurement of ecological communities. Scientists believe that there is a strong relationship between species diversity and ecosystem processes. However efforts to investigate microbial diversity using whole genome shotgun reads data are still scarce. With novel applications of data structures and the development of novel algorithms, firstly we developed an efficient k-mer counting approach and approaches to enable scalable streaming analysis of large and error-prone short-read shotgun data sets. Then based on these efforts, we developed a statistical framework allowing for scalable diversity analysis of large, complex metagenomes without the need for assembly or reference sequences. This method is evaluated on multiple large metagenomes from a variety of environments, such as seawater, human microbiome, soil. Given the velocity in growth of sequencing data, this method is promising for analyzing highly diverse samples with relatively low computational requirements. Further, as the method does not depend on reference genomes, it also provides opportunities to tackle the large amounts of unknowns we find in metagenomic datasets.

To my parents, Mr. Hui Zhang and Mrs. Xiuxiang Zhao.

ACKNOWLEDGMENTS

First and foremost, I would like to sincerely thank my advisor, Dr. C. Titus Brown, for his insightful guidance, his generous support, his confidence in me, and especially his patience and encouragement when progress was slow. It's been an honor and a privilege working with you, Titus!

Next I would like to thank my committee members, Dr. James Cole, Dr. Richard Enbody, Dr. Yanni Sun and Dr. Eric Torng, for their valuable input to my research and their review of this dissertation. I am also indebted to my collaborators, including Dr. Shana Goffredi, Dr. Joan Rose, Dr. Tom Schmidt, Dr. Tracy Teal, Dr. James Tiedje, and Dr. Susannah Tringe. I learned from their insights a lot.

I would also like to thank the GED lab members, who are also my best friends. I thank Dr. Sherine Awad, Michael Crusoe, Jiarong Guo, Luiz Irber and Camille Scott for their excellent collaborative ideas and their generous help with my preparation of this dissertation and the defense. I should not forget several former members of the GED lab, including Rose Canino-Koning, Dr. Elijah Lowe, Eric McDonald, Dr. Kanchan Pavangadkar, Dr. Jason Pell, Dr. Likit Preeyanon, and Neem Serra. I really miss the time we worked together. Your work is very important to the success of my project. Thanks you!

To all my friends, thank you for the support and encouragement. A big thank to Dr. Adina Howe, a colleague, mentor and friend, for everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Overview

Species diversity is an important measurement of ecological communities. Scientists believe that there is a relationship between species diversity and ecosystem processes [70]. Evaluating the species diversity in a community is a central research topic in macroorganism ecology. Many methods have been developed over the last few decades, aimed at answering questions such as "how many species of birds are in this habitat". Nevertheless, until recently scientists had not started to think seriously about larger-scale questions such as "How many species are there on earth?" [81] or "How many species are there in the ocean?" [89] until recently. Why? The answer is straightforward: Microorganisms represent the vast majority of the Earth's biodiversity and the assessment of microbial diversity is quite difficult.

It is believed that microbial diversity is the outermost frontier of the exploration of diversity [74]. Microorganisms are ubiquitous. They were the first forms of life on the Earth. There are more bacterial cells in our body than human cells [113]. There are several reasons why assessment of microbial diversity is such a challenge. First, the concept of species is ambiguous. Morphological examination is impossible: fewer than 1% of microorganisms in the biosphere cannot be cultivated by traditional cultivation techniques[29]. To overcome this obstacle, metagenomics has emerged, driven by the progress of next-generation sequencing (NGS) technology. Lots of metagenomics projects have been performed on samples ranging

from acid mine drainage channels to human gut. For complex environmental samples such as soil, the resulting data sets can be huge. There are approximately a billion microbial cells, with about 4 petabase pairs of DNA($4 * 10^{12}$ bp) [144]. Since we have limited sequencing power, the resulting metagenomics data sets from highly complex samples (e.g. soil) only correspond to a tiny fraction of the actual genomic content in the sample. The large size of data sets and the low sequencing coverage make the assessment of microbial diversity of high diversity sample even harder. Novel methods are needed.

1.2 Next-generation sequencing

Sequencing technology is changing quickly. Over the past decade, next-generation sequencing (NGS) has become the dominant technology and almost replaced classic Sanger sequencing technology. Illumina and Roche 454 are the two most popular platforms. Illumina can generate reads of shorter lengths, typically up to 150 base pairs(bp) for HiSeq and 250 bp for MiSeq platform[102, 80]. However, at a much lower cost compared to the Roche 454 sequencing technology, which generates reads with a length of 500 to 1K bp. In fact a recent study comparing Illumina versus Roche 454 for metagenomics shows that both platforms agreed on over 90% of the assembled contiguous reads (contigs) and 89% of the unassembled reads[72]. Because of the advantage of the low cost, there is a trend towards Illumina is dominating the sequencing market, which means that while designing any tool for metagenomics, a developer should take the relatively short length of Illumina reads into account.

1.3 Metagenomics

It is believed that the word "metagenomics" was coined in 1998 [47]; it can be translated as 'beyond the genome' [36]. At that time, it was based on the technique of cloning environmental DNA randomly and screening for genes of interest, especially 16S ribosomal RNA (rRNA) genes. This technique was firstly applied in practice by Schmidt et al. in 1991 [117]. It was a crucial step in expanding sequence-based investigation to the microbial world. Before that it was standard protocol to culture and isolate microbes and do analysis. It resulted in a much narrower picture of the diversity of an ecosystem as only a small portion of the microbial species (5% or less) in the biosphere can be cultured with traditional cultivation techniques [128]. Metagenomics, with the concept of cloning DNA directly from sample without cultivation, brought researchers the ability to explore the entire spectrum of organisms in an environment.

The number of microbial species in some ecological communities is huge. In soil, it is estimated that there exist millions of species with most of them in low abundance [35]. The improvement of NGS technology with ever higher throughput and ever lower costs has been accelerating metagenomics research recently, since only high throughput NGS strategy can sample the contents of those populations deeply enough to examine rare species.

Currently, there are two approaches in metagenomics. One is amplicon metagenomics, in which genes of interest, such as 16S rRNA genes, are amplified and sequenced [128]. This is the traditional way dating back to the 1991 work by Schmidt et al. Many microbial diversity studies have relied on this approach. The other approach is whole genome shotgun metagenomics, which sequences randomly isolated DNA fragments without targeting specific genes. Since the whole genomes of organisms in a sample are available, and not just the

limited genes of interest like 16S/18S rRNA, this whole genome shotgun sequencing approach can in theory provide better taxonomic resolution and more information benefiting other investigation [135] [102]. Now there are thousands of metagenomic samples available in online database, such as MG-RAST [38].

There have been many metagenomics projects focusing on the microbial samples of different kinds of habitat, from extreme environment such as acid mine drainage channels with low complexity [135], and medium complexity samples like human gut [102] and cow rumen [48], to high complexity samples like seawater [137] and soil [37].

Metagenomics studies have revealed lots of knowledge of the microbial community in different habitats. Some of them shed light on the explanation of some serious human diseases. Studies have shown associations between human gut metagenomes and type II diabetes [104], obesity [134, 60] or Crohn's disease [90].

In almost all of these metagenomics projects, diversity analysis plays an important role in supplying information about the richness of species, the species abundance distribution in a sample or the similarity and difference between samples, all of which are crucial to draw insightful and reliable conclusions.

1.4 Concept of diversity

When we characterize an ecological community, diversity measurements are often the first step. It is always desirable to know how many species there are in a sample – its "richness" – and how abundant each species is relative to others in the same sample – its "evenness". They are straightforward conceptually. However, in practice, there are a large number of quantities that are used to measure species diversity, for the many different approaches to

sampling individuals.

At a high level, three diversity indices are well established and used in ecology; these are α-diversity, β-diversity, and γ-diversity. The α-diversity is the diversity in one defined habitat or sample. The β-diversity compares species diversity between habitats or samples. The γ-diversity is the total diversity over a large region containing multiple ecosystems[74].

The concept of diversity has two aspects, richness and evenness. Richness is the total number of species identified in a sample, which is the simplest descriptor of a community structure. Evenness is a measure of how different the abundance of a species is compared to other species in a community. If all the species in a community has the same abundance, the community has a higher evenness diversity. However, almost all natural communities are highly uneven, which means the community is dominated by relatively few species and there are a large number of species with low abundance. It raises a question about the effectiveness of using the measurement of richness to represent species diversity. Is a community with 1 dominant species and 10 rare species more diverse than a community with 3 dominant species and 2 rare species? Thus, new metrics taking both richness and evenness into account have been suggested. The two most popular diversity indices are Shannon diversity [121], which is based on information theory and shows the information in a community as an estimate of diversity, and Simpson diversity [122], which basically shows the probability that two individuals picked randomly from a community belong to the same species.

Besides these two, Hill [49] proposed a new diversity index based on the species abundance distribution, which uses a weighted count of species to measure diversity. It can be considered as a generalized diversity index, since both Shannon and Simpson index and richness can be seen as special cases of the Hill diversity index. It is necessary to note that we cannot tell if any index is generically better than the others. It all depends on the characteristics of a

community and the process of sampling, as well as other factors. How to choose the diversity indices to better represent the diversity information of a community has been investigated and discussed extensively [5, 45, 91].

In microbial ecology, richness is simply the most popular index to measure microbial diversity, partially because of the challenge raised by the different characteristics of a microbial community. Most of the microbial diversity studies concentrate on species richness comparison[5]. Lots of methods to estimate richness in classic ecology were borrowed to tackle the problem of estimating microbial diversity, which will be discussed in the next section.

1.5 Problem statement

In almost all metagenomics projects, diversity analysis plays an important role in supplying information about the richness of species, the species abundance distribution in a sample, and the similarity and difference between different samples. The topic of microbial diversity measurement has been investigated for a long time with many methods and software packages developed. However, there still remains lots of room for more work.

Traditionally used for amplicon metagenomics data set, OTUs(Operational Taxonomic Units) based on 16S rRNA genes are used as the basic units for diversity analysis on shotgun metagenomic data. OTUs can be good replacements of the concept of "species" in metagenomics. Basically contigs are assembled from reads and are "binned" into OTUs using composition-based or similarity-based approaches. Then the diversity can be estimated by using the abundance information of the OTUs. The mainstream methods to measure microbial diversity are still focusing on the use of 16S rRNA amplicon metagenomics data.

Many of the popular microbial diversity analysis software packages generally accept 16S rRNA data as input. This is understandable because the concept of OTU is from the similarity of 16S rRNA sequences. Using 16S rRNA data to measure diversity is popular but is not without problems. The 16S rRNAs may not be that reliable to be OTU markers. The reliability is sensitive to potential horizontal gene transfer and the variance of gene copy in bacteria. There have been suggestions that alternative marker genes should be used, such as single copy housekeeping genes. Thus, measuring diversity beyond using 16S rRNA data is worth investigating. Recently there are many more projects generating whole genome shotgun metagenomics data sets. However, they are mainly used for assembly and annotation purpose. Less attention was paid to diversity measurement using these whole genome metagenomics data sets. One possible reason is that the whole genome metagenomics data sets are often with low depth given the high diversity of metagenomics samples compared to 16S rRNA ampicon metagenomics data set. Assembly and annotation are always challenging with the low depth and lack of reference sequences. It is also true for diversity measurement. On the other hand, although with low depth, some whole genome metagenomics data sets are of large size because of the high diversity. For instance, there may be 4 petabase pairs of DNA in a gram of soil [144]. Many of those methods for sequence binning or diversity estimation do not scale well and will not work for large metagenomics data sets. For instance, many composition-based binning approaches involve k-mer/signature frequency distribution calculation, which is rather computationally expensive. Even basic sequence alignment will be impossible for large metagenomics data sets. Many of those statistical software packages to estimate diversity using various estimators are not prepared for the large scale of whole genome metagenomics data.

With the development of NGS technology, the cost of sequencing is dropping rapidly.

Whole genome metagenomics sequencing is more popular and a large amount of metagenomics data is being generated with increasing speed, which cannot be even met by the increase of computational capacity. Novel methods that can scale well are extremely needed to deal with the increasingly large metagenomics data set.

1.6 Significance of research

We established a series of approaches to enable scalable and effective investigation of microbial diversity using whole-genome shotgun metagenomic data. Firstly a k-mer counting package - khmer was developed to enable fast and memory efficient k-mer-based analysis of sequencing data sets[148, 28]. Khmer relies on Count-Min Sketch, a probabilistic data structure used to store the frequency of distinct elements efficiently. Unlike other data structures used for k-mer counting, such as hash tables, suffix arrays, and trie structure, the Count-Min Sketch has significantly low memory usage for sparse data sets with trade-off with counting false positive. We conducted extensive analysis on the performance of the counting algorithm and benchmark to compare the performance of the khmer to other k-mer counting packages. The initial motivation of developing khmer was to count the k-mers in metagenomes for diversity analysis. Now khmer has been widely used for many other purposes, from enabling large scale de novo metagenome assembly to sequencing error detection and correction.

Based on the efficient k-mer counting package khmer, especially with the ability to do online counting and retrieval entirely in memory, we developed digital normalization[10], "a single-pass computational algorithm that systematizes coverage in shotgun sequencing data sets, thereby decreasing sampling variation, discarding redundant data, and removing the majority of errors." Digital normalization can reduce the computational expense of down-

stream analysis such as assembly dramatically because after the normalization of sampling variance, redundant reads are discarded as well as the errors in them. The algorithm of digital normalization has been used by many research groups to facilitate their analysis and has been implemented in different tools like Trinity and Illumina's TruSeq pipeline. Like digital normalization, based on the same approach to estimate sequencing depth without a reference assembly, a streaming approach to analyze and trim sequencing errors in short reads datasets was developed[147]. The approach offers a general framework for streaming sequence analysis and could be used for error correction and variant calling. Moreover, the approach can be applied generically to data sets with variable sequencing coverage, such as metagenomes especially.

Further more, by integrating efficient k-mer counting and a novel de Bruijn graph mapping method based on digital normalization we developed a novel approach to allow for scalable diversity analysis of large, complex metagenomes. A novel concept - IGS (informative genomic segment) is proposed to represent the unique information in a metagenomics data set. The IGSs can be used as a complement of OTUs to be the cornerstone for diversity analysis of whole shotgun metagenomics data sets. The abundance of IGSs in different samples can be retrieved by mapping the reads to de Bruijn graphs. In this procedure, not like many other microbial diversity analysis methods, assembly or binning is not required any more. This method was evaluated on multiple metagenomes from a variety of environments (e.g., human body part, seawater, soil). Given the velocity in growth of sequencing data, this method is promising for analyzing highly diverse samples with relatively low computational requirements. Further, as the method does not depend on reference genomes, it also provides opportunities to tackle the large amounts of unknown "dark matter" we find in metagenomic datasets.

1.7 Outline of dissertation

In this dissertation I will discuss in detail a series of approaches enabling scalable and effective investigation of microbial diversity using whole-genome shotgun metagenomic data. In chapter 2, I will do a brief review of relevant literature about the challenges I face to enable diversity analysis of metagenomic data. In chapter 3, I will describe a novel approach to count k-mers efficiently and a scalable approach to retrieve the coverage of a read in a data set based on efficient and online k-mer counting. In chapter 4, I will introduce the two applications of this approach, digital normalization to reduce the redundancy of metagenomic reads dataset and a streaming method to analyze sequencing error. Both are critically important to the improvement of other metagenomic data analysis approaches, like assembly, error trimming or contigs/reads binning. In chapter 5, I will discuss how I developed the concept of IGS based on efficient k-mer counting and digital normalization. The effort to increase the accuracy of IGS based method will be discussed and the performance of the IGS method on simulated data sets and real data sets will be demonstrated. I will give a summary about how the novel statistical framework based on IGS makes a difference to the diversity analysis in current microbial ecology research and some directions of future work will be discussed in the last chapter.

Chapter 2

Review of relevant literature

2.1 Challenges in counting k-mers accurately and efficiently

A k-mer is a substring with length k in a DNA sequence. K-mer counting is the problem to determine the occurrences of such k-mers in a DNA dataset [78]. Efficient k-mer counting plays an important role in solving many bioinformatics problems.

One important problem is *de novo* assembly of very large number of short reads. With the development of next generation sequencing(NGS) technology, many research groups can afford the sequencing of the sample of specific species or even metagenomic samples with numerous different species[86]. Large amount of NGS short reads are generated and de novo assembly is required for these sequence data sets[87]. Currently, de Bruijn graph method is popular in the attempts to do *de novo* assembly because of its advantage in assembling next generation sequencing short reads[99]. Several popular assemblers have been developed based on de Bruijn graph, such as Velvet[146], ALLPATHS[12], ABySS[126] and SOAPdenovo[67]. All the k-mers in a sequence data set are represented as nodes in the de Bruijn graph. If two k-mers have an overlap of (k-1)-mer, the two k-mers can be connected. Since k-mer is such a basic unit in *de Bruijn* graph *de novo* assembly, it is of great importance to determine the occurrence of k-mers. One example is that sequencing errors can generate many erroneous

unique k-mers and we can filter out the reads with too many unique k-mers before doing the assembly. Similarly, we can also filter out the reads with k-mers that occur too many times for smoothing MDA-abundance reads dataset. Pre-filtering reads to reduce the size of reads data set to assemble is important to reduce the time and memory usage. Another application of k-mer counting is the evaluation of microbial diversity in metagenomic samples. Counting the number of k-mers and geting the k-mer abundance distribution can give us some hints about the richness and evenness of a metagenomic sample, although the erroneous k-mers from sequencing error will cause some problems. Also, one of the popular approaches to do metagenomic contigs binning is based on analyzing the k-mer abundance profile [95] [7][110].Actually this is one of our motivations of the development of khmer, which will be discussed in next chapter. One more example of applications of k-mer counting is the de novo detection of repetitive elements. K-mer frequency can give important information for predicting regions with such repetitive elements as transposons with important biological function.[63]

There are two specific characteristics of NGS shot reads that make k-mer counting complicated. One is that large size of NGS reads data means there are large number of k-mers to count. However the large number of k-mers in a NGS reads data set still only account for a small proportion of the total number of possible k-mers. For a typical value of k as 20, there are 4^{20} possible k-mers, which are far more than the actual number of k-mers present in any genome reads data set. Delicate choice of data structure to use is desired to accommodate to such sparseness of actual k-mers to enable efficient counting. Also, out of the large number of k-mers in a NGS reads data set, many of them, especially unique ones, are erroneous because of sequencing errors. Generally one sequencing error will introduces k erroneous k-mers. With a relatively high error rate as 0.1-1% in Illumina reads data [86]),

as we do more sequencing and have more reads, there will be actually more erroneous k-mers than true k-mers in the data set. Effective and efficient methods to count the large number of k-mers with most of them as erroneous is highly required in NGS reads data analysis [88].

Current methods to do k-mer counting involve the data structures like hash tables, suffix arrays, binary trees or tries structures. If the size of sequence dataset to count is modest, a simple hash table will suffice, where the key is the k-mer and the value is the corresponding count. However there is an obvious obstacle. If the size of sequence dataset is larger, the efficiency of the counting using a simple hash table drops dramatically. Instead another k-mer counting tool - Tallymer, uses suffix array data structure[63]. Admittedly it is more efficient than simple hash table in general. However, the memory requirement is still linear to the number of unique k-mers. Thus this method is not very scalable. For example, the size of a soil metagenomic data set for one sample only have already exceeded 400G bytes, with only a limited sequencing depth[51]. Reads dataset with size like this is difficult for Tallymer to handle. Jellyfish, [78] another popular k-mer counting tool, uses updated hash table data structure, which can reduce the memory usage to store k-mers and the "lock-free" feature of the hash table also enables the parallelism of Jellyfish to make it more efficient. But it still has the same problem as Tallymer linear increase of memory usage with respect to the number of unique k-mers in the data set to count.

Many more k-mer counting software packages based on different data structure and algorithms have been developed in recent years, including BFCounter, DSK, KMC, Turtle and KAnalyze [85, 106, 30, 111, 4]. These software packages differ with each other in algorithmic trade-offs and functionality, especially on how to deal with the trade-off between disk and memory usage, enabling online counting and retrieval or not, exact count or not, and others. Table 2.1 shows a summary of most current k-mer counting packages with the function and

limitation for each.

software	**algorithm**	**(built in) get k-mer occurrence histogram**	**(built in)specific k-mer count retrieval?**	**API?**	**multithreaded?**	**online counting?**
BFCounter	bloom filter,filter out low abundance kmer	N	N	N	Y	Y
DSK	fixed-memory and fixed-disk space streaming algorithm	Y	N	N	N	N
Jellyfish	lock-free hash table	Y	Y	N	Y	Y
KAnalyze	split to disk and merge	N	N	Y	Y	N
Khmer	count-min sketch	Y	Y	Y	Y	Y
KMC	parallel disk-based,similar to DSK	N	N	N	Y	N
MSPKmerCounter	Minimum Substring Partitioning	Y	Y	N	Y	N
Tallymer	enhanced suffix arrays	Y	Y	N	N	N
Turtle	pattern-blocked Bloom filter, filter out low abundance kmer	N	N	N	Y	Y

Table 2.1: **Description of k-mer counting packages.**

2.2 Tackling large and error-prone short-read shotgun data sets

With the dramatic improvements of next generation sequencing(NGS) technologies and the dropping sequencing cost, more research groups can afford sequencing the sample of specific species or even metagenomic samples with numerous different species in large scale[86]. This leads to the explosive growth of sequencing data to analyze. Important biological insights will be drawn from mining the large genomic data. However serious challenges need to be overcome to enable efficient and effective analysis.

Similar to the k-mer counting problem, the first obstacle to analyze large NGS data is the formidable size of the NGS data from high sequencing depth. Higher sequencing depth is required to assemble a genome successfully using NGS short reads than using longer reads from traditional Sanger sequencing technology.For example, to get a decent coverage of human genome with by NGS short reads to get satisfying assembly, 100x sequencing depth is generally required, which leads to a 300 GB NGS reads data set[39]. For samples with variable abundance of genomic contents such as transcriptome or metagenome, to get enough sequencing depth for the rarer genomic content, the overall sequencing depth will greatly increase, which leads to dramatically large reads data sets. However, with uneven abundance of genomic contents meanwhile.

Still like the k-mer counting problem, sequencing errors bring significant troubles to effective and efficient analysis of large NGS data. With a relatively high error rate as 0.1-1% in Illumina reads data [86]), the more sequencing we perform, the more sequencing errors we will have in the data, no matter if the sequencing depth is high or low. So for deep sequencing, most of the novelty will be dominated by sequencing errors [24]. As discussed previously,

generally one sequencing error will introduce k erroneous k-mers. So many sequencing errors will undermine the effectiveness of many down-streaming analysis of such short reads data.

With the two characteristics of NGS short reads data - large size and error-prone, combined with the dropped cost of sequencing, we are facing the third challenge, that the increase of our capacity to analyze data cannot catch up with the capacity of generating data [132]. It is difficult to easily manipulate and analyze the large genomic data without significant improvement of computational approaches.

The approaches to overcome these problems, are straightforward, in a way. To deal with the large size of data, we try to decrease the size. To deal with the errors in data, we try to remove or correct them.

Assembly can be seen as a solution to reduce the size of large data, with the sacrifice of losing abundance information of genomic contents. There have been significant theoretical progress to store and analyze the big sequencing data efficiently [23, 123]. Based on the progress, a number of new assemblers have been developed, such as ABySS, Velvet, SOAPdenovo, ALLPATHS, SGA, and Cortex [125, 145, 69, 39, 124, 58]. For sequencing data with uneven abundance of genomic contents like metagenome or transcriptome, novel assembler or 3rd party add-ons are developed specifically, such as Trinity, Oases, MetaVelvet, MetaVelvet-SL, Meta-IDBA, Velvet-SC, DIME, MEGAHIT, Omega [41, 118, 93, 1, 97, 19, 43, 66, 46].

Assembly can be seen as a solution to decrease error rate too. There may still be errors in assembled contigs or genomes. However, large amount of errors located in numerous reads have been discarded in the procedure of assembly. To remove or correct errors directly from reads, k-mer spectral analysis is a popular approach[99]. Basically low-abundance k-mers will be found and treated as likely errors. Those likely errors can be removed, trimmed with other k-mers, or corrected using statistical learning method [62]. More error removal and

correction software packages were developed[83, 14, 62]. Such error removal or correction approaches can be applied to preprocess reads data before assembly.

As necessary components in NGS data analysis pipeline, assembly and error removal/correction have proved to be effective to enable more accurate interrogation to the large, error-prone short-read data. However both assembly and error removal/correction are very compute intensive. The memory usage of assembly does not scale well to the size of data to assemble. The error removal/correction based on k-mer spectral analysis normally involves two iterations of examining the data, which is also memory intensive and time consuming. Streaming and semi-streaming algorithms, which examine the data only once or less than twice and scale well to the size of input data in memory/time requirements, are promising to be integrated with the pre-assembly preprocessing and error removal/correction to achieve higher efficiency of NGS data analysis finally. In this dissertation, we will discuss our efforts in this direction.

2.3 Challenges in measuring diversity of metagenomics

2.3.1 Diversity measurement in microbial ecology

There have been numerous mature methods and tools to measure diversity of macroorganisms in decades of development of classic ecology. One would think that we just need to borrow those methods to use in microbial field. Unfortunately in reality this is not the case. The microbial communities are so different from macroorganisms like plant or animal communities, with the number of species many order of magnitude larger [139]. This fact raises serious sampling problems. It is extremely difficult to cover enough fraction of the microbial community even with impressively large sample size thanks to modern metagenomic

approaches [109]. In a word, diversity measurement is a rather big challenge for microbial communities and novel and effective methods are highly demanded [114].

2.3.1.1 OTU Identification using sequence markers

To borrow the methods of diversity measurement from classic ecology on the use of evaluating microbial diversity, the first problem is that in microbial world, there is no unambiguous way to define "species" [131]. It is impossible to identify a microbial individual as a specific species morphologically. In fact in metagenomics the concept of "species" has been replaced by OTUs(Operational Taxonomic Units). An OTUs are those microbial individuals within a certain evolutionary distance. Practically we mainly use 16S rRNA genes as the evolutionary marker genes, because 16S rRNA genes exist universally among different microbial species and their sequences change at a rate corresponding with the evolutionary distance. So we can describe microbial individuals with higher than a certain percent(e.g. 97%) 16S rRNA sequence similarity as one OTU, or belonging to one species [114].

2.3.1.2 Binning of metagenomic reads into OTUs

In classic ecology dealing with samples from macroorganisms communities, before we can use any statistical method to measure diversity, it is standard procedure to identify the species of each individual in a sample. It is the same for diversity measurement of microbial communities. Difference is that here we need to place the sequences(individuals) into respective "bin" or OTUs(species). There are two strategies to do such binning - Composition-based or intrinsic binning approach and similarity-based or extrinsic binning approach.

2.3.1.2.1 Composition-based approach Lots of efforts have been put to get a comprehensive category of reference microbial genome sequences [54, 141]. Currently there are a large number of finished or high-quality reference sequences of thousands of microbial species available in different databases and this number is still increasing quickly [79, 38, 138]. So the first intrinsic composition-based approach is to use those reference genomes to train a taxonomic classifier and use that classifier to classify the metagenomics reads into bins. Different statistical approaches like Support Vector Machines [95], interpolated Markov models[7],naive Bayesian classifiers, and Growing Self Organizing Maps [110] were used to train the classifier. Without using any reference sequences for the training, it is possible to use signatures like k-mers or codon-usage to develop reference-independent approach. The assumption is that the frequencies distribution of the signatures are similar of the sequences from the same species. TETRA is such a reference-independent tools using Markov models based on k-mer frequencies [133]. There is another tool using both TETRA and codon usage statistics to classify reads [136].

2.3.1.2.2 Similarity-based approach The similarity-based extrinsic approach is to find similarity between the reads sequences and reference sequences and a tree can be built using the similarity distance information. MEGAN [56] is a typical tool using this method, which reads a BLAST file output. Other sequence alignment tools can also be used here like BowTie2 or BWA. Recently, an alternative strategy was developed, which only uses the reference sequences with the most information rather than all the reference sequences to do alignment. Those reference sequences include 16S rRNA genes or some other specific marker genes. The benefit is obvious, it is more time-efficient since there are fewer reference sequences to align to. Also, it can provide better resolution and binning accuracy since the

marker genes can be selected carefully with the best distinguishing power. AMPHORA2 [142] and MetaPhlAn [119] are two typical tools using this strategy.

2.3.1.3 Statistics for diversity estimation

After the binning of sequences into OTU, we need statistical analysis to help us estimate the diversity. Many statistical methods have been developed and widely used in classical ecology of macroorganisms. However the first difference between diversity measurement of macroorganisms and microbial community is that generally the microbial community diversity is much larger than observed sample diversity, thanks to the high diversity characteristics of microbial community and the limit of metagenomics sampling and sequencing. The first approach which is also considered as classic is rarefaction. Rarefaction curve can be used to compare observed richness among different samples that have been sampled unequally, which is basically plotting of the number of observed species as a function of the sampled individuals. It is worth noting that rarefaction curve shows the observed diversity, not the total diversity. We should not disregard those unseen microbial species, which is pretty common for microbial community sampling.

To estimate the total diversity from observed diversity, different estimators are required.

The first one is extrapolation from accumulation curve. The asymptote of this curve is the total diversity, which means the number of species will not increase any more with sampling more individuals. To get the value of that asymptote point, from observed accumulation curve, a function needs to be assumed to fit the curve. Several proposals have been made to use this extrapolation method [22, 40]. The problem is that if the sampling effort only covers a small fraction of the total sample, which means the accumulation curve just starts, it is difficult to find an optimal function to fit the curve. Different functions can fit the

curve equally well but will deduct dramatically different asymptote value. So this curve extrapolation method should be used cautiously.

Another one is parametric estimator, which assumes that the relative abundance follows a particular distribution. Then the number of unobserved species in the community can be estimated by fitting observed sample data to such abundance distribution then the total number of species in the community can be estimated. Lognormal abundance distribution is mostly used in different project since most communities of macroorganisms has a lognormal abundance distribution and it is believed that it is also typical for some microbial communities [29, 115, 105]. It is understandable that there is always controversy as to which models fit the communities best since in an ideal world the abundance distribution should be inferred from the data,not be assumed unverifiably. The problem is that we can only infer the abundance distribution accurately when the sample size is large enough. There have been some attempts on this direction recently [35] and more robust methods are still needed.

If the species abundance distribution cannot be inferred, we can still use nonparametric estimators to estimate the total diversity without assuming that abundance distribution arbitrarily. These estimators are related to MRR(mark-release-recapture) statistics, which compare the number of species observed more than once and the number of species observed only once. If current sampling only covers a small fraction of a diverse community, the probability that a species is observed more than once will be low and most species will be observed only once. If current sampling is enough to cover most species in the community, the opposite will be the case. A series of estimators invented by Chao are the representative estimators in this category, including Chao1 [15], Chao2 [16], ACE [17] and ICE [65]. For example, Chao1 formular is:

$$S_{Chao1} = S_{obs} + \frac{{n_1}^2}{2n_2}$$

where S_{obs} is the number of species observed, n_1 the number of species observed once(singletons, with only one individule), and n_2 the number of species observed twice(doubletons, with exactly two individuals) in the sample. The ACE uses data from all species rather than just singletons and doubletons. Its formular is:

$$S_{ACE} = S_{abund} + \frac{S_{rare}}{C_{ACE}} + \frac{F_1}{C_{ACE}}{\gamma_{ACE}}^2$$

where S_{rare} is the number of rare species (with few than 10 individuals observed) and S_{abund} is the number of abundant species (with more than 10 individuals).

In past years there are several software packages that have been developed for biodiversity analysis. Out of them, EstimateS [21] is a software that can be used for general purpose diversity analysis, which implement a rich set of diversity analysis algorithms. However it is not designed specifically for microbial diversity analysis. So microbial diversity data should be preprocessed to general population data to be fed into EstimateS. Two other softwares - MOTHUR [116] and QIIME [13] are designed for microbial diversity. So they are more popular in microbial diversity analysis. CatchAll [11] is a relatively newer package, which can estimate the diversity using both nonparametric and parametric estimators including many variants and return the results using different estimators and the respective credibility of the results.

Chapter 3

Efficient online k-mer counting using a probabilistic data structure

3.1 Introduction

Our motivation for exploring efficient k-mer counting comes from our work with metagenomic data, where we routinely encounter data sets that contain 300×10^9 bases of DNA and over 50 billion distinct k-mers [50]. K-mer counting plays a key role in our initial investigation on using distinct k-mers to measure microbial diversity. We needed to count how many distinct k-mers in different metagenomics data sets and get the abundance distribution. In the beginning we used an existing k-mer counting tool - Tallymer [63] . However as we started to deal with larger metagenomic data, where we routinely encounter data sets that contain 300×10^9 bases of DNA and over 50 billion distinct k-mers [50], it was not efficient enough and for some data set it cannot handle at all. Also, to efficiently filter, partition, and assemble these data, we need to store counts for each of these k-mers in main memory, and query and update them in realtime — a set of functionality not readily offered by current packages. Moreover, we wish to enable the use of cloud and desktop computers, which may have poor I/O performance or limited memory. These needs have dictated our exploration of efficient in-memory k-mer counting techniques.

In this chapter we present the khmer software package for fast and memory efficient

online counting of k-mers in sequencing data sets. Unlike previous methods based on data structures such as hash tables, suffix arrays, and trie structures, khmer relies entirely on a simple probabilistic data structure, a Count-Min Sketch. The Count-Min Sketch permits online updating and retrieval of k-mer counts in memory which is necessary to support online k-mer analysis algorithms. On sparse data sets this data structure is considerably more memory efficient than any exact data structure. In exchange, the use of a Count-Min Sketch introduces a systematic overcount for k-mers; moreover, only the counts, and not the k-mers, are stored.

We use the Amazon cloud to compare time, memory, and disk usage of our k-mer counting implementation with that of other k-mer counting software packages, for two problems. First, we generate a k-mer abundance distribution for large data sets; and second, we query many individual k-mer counts at random from a previously constructed k-mer count database. We show that khmer is competitive in speed, memory, and disk usage for these problems. We also analyze the effects of counting error on calculations of the k-mer count in sequencing data sets, and in particular on metagenomic data sets.

This chapter contains published materials from [148].

3.2 Count-Min Sketch and its application in k-mer counting

Below, we describe an implementation of a simple probabilistic data structure for k-mer counting. This data structure is based on a Count-Min Sketch [26], a generalized probabilistic data structure for storing the frequency distributions of distinct elements.

Probabilistic approaches can be particularly memory efficient for certain problems, with

memory usage significantly lower than any exact data structure [96]. However, their use introduces set membership or counting false positives, which have effects that must be analyzed in the context of specific problems. Moreover, unlike existing techniques, the Count-Min Sketch stores only counts; k-mers must be retrieved from the original data set. In exchange, the low memory footprint enabled by this probabilistic approach enables online updating and retrieval of k-mer counts entirely in memory, which in turn supports streaming applications such as digital normalization, which will be discussed in next chapter.

Because of the probabilistic characteristics of this approach, We will also discuss the choice of optimal parameters to balance the efficiency and accuracy of k-mer counting.

3.2.1 Implementing a Count-Min Sketch for k-mers

The two basic operations supported by khmer are `increment_count(kmer)` and `c = get_count(kmer)`. Both operate on the data structure in memory, such that neither incrementing a count nor retrieving a count involves disk access.

The implementation details are similar to those of the Bloom filter in [96], but with the use of 8 bit counters instead of 1 bit counters. Briefly, Z hash tables are allocated, each with a different size of approximately H bytes $(H_1, H_2, ..., H_Z)$; the sum of these hash table sizes must fit within available main memory. To increment the count for a particular k-mer, a single hash is computed for the k-mer, and the modulus of that hash with each hash table's size H gives the location for each hash table; the associated count in each hash table is then incremented by 1. We use different sizes for each hash table so as to vary the hash function. Even if two k-mers have the same modulus in one hash table (a collision), they are unlikely to collide in the other hash tables. To retrieve the count for a k-mer, the same hash is computed and the minimum count across all hash tables is computed. While different in

implementation detail from the standard Bloom filter, which uses a single hash table with many hash functions, the performance details are identical [96]. One particularly important feature of the Count-Min Sketch is that the counting error is *one-sided* [26]. Because counts are only incremented, collisions result in inflated miscounts; if there is no collision for a particular k-mer, the count is correct.

An additional benefit of the Count-Min Sketch is that it is extremely easy to implement correctly, needing only about 3 dozen lines of C++ code for a simple threadsafe implementation. (We have described how khmer scales with multiple threads in [82].)

To determine the expected false positive rate — the average frequency with which a given k-mer count will be incorrect when retrieved — we can look at the hash table load. Suppose N distinct k-mers have been counted using Z hash tables, each with size H. The probability that no collisions happened in a specific entry in one hash table is $(1-1/H)^N$, or approximately $e^{-N/H}$. The individual collision rate in one hash table is then $\approx 1-e^{-N/H}$. The total collision rate, which is the probability that a collision occurred in each entry where a k-mer maps across all Z hash tables, is $\approx (1-e^{-N/H})^Z$, which is also the expected false positive rate.

While the false positive rate can easily be calculated from the hash table load, the average *miscount* — the degree to which the measured count differs from the true count — depends on the k-mer frequency distribution, which must be determined empirically. We analyze the effects of this below.

3.2.2 Choosing number and size of hash tables used for k-mer counting

The false positive rate depends on the number of distinct k-mers N, the number of hash tables Z, and the size of the hash tables H: $f \approx (1 - e^{-N/H})^Z$, with an associated memory usage of $M = HZ$. We face two common scenarios: one in which we have a fixed number of k-mers N and fixed memory M and we want to calculate the optimal number of hash tables Z; and one in which we have a desired maximum false positive rate f and a fixed number of k-mers N, and we want to calculate the minimum memory usage required to achieve f.

For fixed memory M and number of distinct k-mers N, the optimal number of hash tables can be found by minimizing f; taking the derivative, df/dZ, with $f \approx \exp(Z \log(1 - e^{-ZN/M}))$ and solving for 0, we find that f is minimized when $Z = \log(2) * (M/N)$ (see [8] for details).

Given a desired false positive rate f and a fixed number of k-mers N, the optimal memory usage can be calculated as follows. First, the optimal number of hash tables is determined by the expected false positive rate alone: $Z = \log_{0.5} f$. Using this Z, the minimum average hash table size H necessary to achieve f can be calculated as $H = (\log_{0.6185}(f) \times N)/Z$ (see [8] for details).

A remaining problem is that the number of distinct k-mers N is typically not known. However, memory- and time-efficient algorithms for calculating N do exist and we has implemented this in khmer[34].

3.3 khmer can count k-mers efficiently

3.3.1 khmer is a generally useful k-mer counting approach

Data set	size of file (GB)	number of reads	number of distinct k-mers	total number of k-mers
subset 1	1.90	9,744,399	561,178,082	630,207,985
subset 2	2.17	19,488,798	1,060,354,144	1,259,079,821
subset 3	3.14	29,233,197	1,445,923,389	1,771,614,378
subset 4	4.05	38,977,596	1,770,589,216	2,227,756,662
entire data set	5.00	48,721,995	2,121,474,237	2,743,130,683

Table 3.1: **Benchmark soil metagenome data sets for k-mer counting performance, taken from [50].**

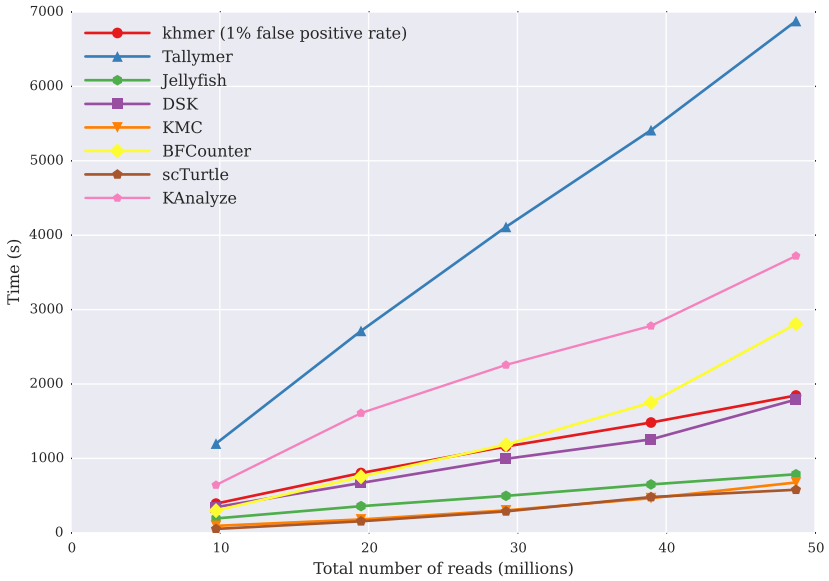

Figure 3.1: **Comparison of the time it takes for k-mer counting tools to calculate k-mer abundance histograms, with time (y axis, in seconds) against data set size (in number of reads, x axis). All programs executed in time approximately linear with the number of input reads.**

We measured time and memory required to calculate k-mer abundance histograms in five soil metagenomic read data sets using khmer, Tallymer, Jellyfish, DSK, KMC, Turtle, and KAnalyze (Table 3.1; Figures 3.1 and 3.2). We chose to benchmark abundance histograms

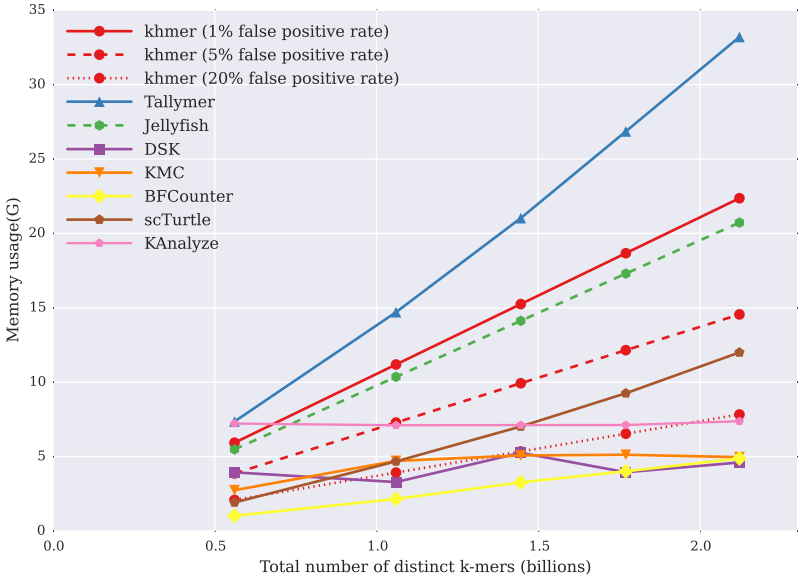

Figure 3.2: **Memory usage of k-mer counting tools when calculating k-mer abundance histograms, with maximum resident program size (y axis, in GB) plotted against the total number of distinct k-mers in the data set (x axis, billions of k-mers).**

because this functionality is common to all the software packages, and is a common analysis approach for determining assembly parameters [18]. We applied each package to increasingly large subsets of a 50m read soil metagenome data set [50]. For the BFCounter, KMC, Turtle and KAnalyze packages, which do not generate k-mer abundance distribution directly, we output the frequency of each k-mer to a file but do no further analysis.

khmer offers a general range of useful performance tradeoffs for disk I/O, time and memory. From the performance comparison between khmer and other k-mer counting packages in calculating k-mer abundance distributions, khmer is comparable with existing packages. Figure 3.1 shows that the time usage of the khmer approach is comparable to DSK and BFCounter, and, as expected, increases linearly with data set size. Tallymer is the slowest of the four tools in this testing, while KMC, Turtle, and Jellyfish are the fastest. From

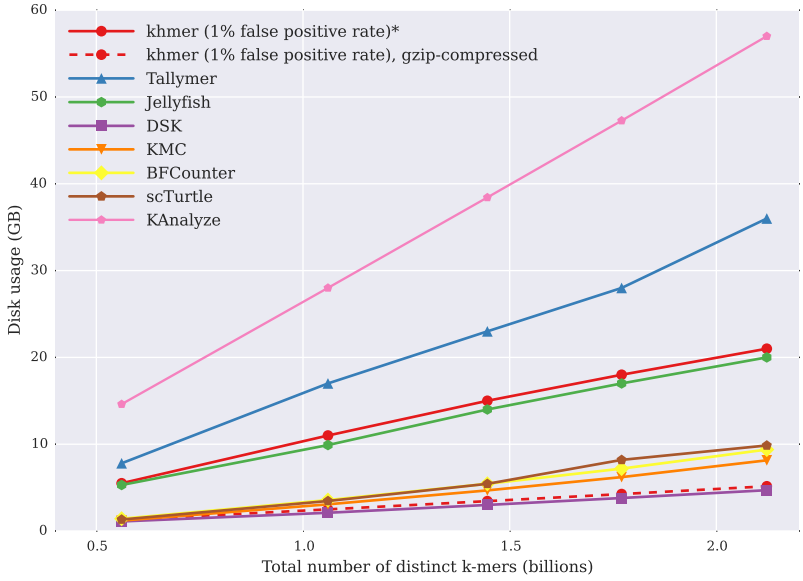

Figure 3.3: **Disk storage usage of different k-mer counting tools to calculate k-mer abundance histograms in GB (y axis), plotted against the number of distinct k-mers in the data set (x axis). *Note that khmer does not use the disk during counting or retrieval, although its hash tables can be saved for reuse.**

Figure 3.2, we see that the memory usage of Jellyfish, Tallymer, BFCounter, and Turtle increases linearly with data set size. Tallymer uses more memory than Jellyfish generally, while BFCounter and Turtle have considerably lower memory usage. DSK, KMC, and KAnalyze use constant memory across the data sets, but at the cost of more limited functionality (discussed below).

We also measured disk usage during counting. Figure 3.3 shows that the disk usage also increases linearly with the number of k-mers in the data set. For a high-diversity metagenomic data set of 5 GB, the disk usage of both Jellyfish and Tallymer is around 30 GB. khmer counts k-mers entirely in working memory and does not rely on any on-disk storage to store or retrieve k-mer counts, although for practicality the hash tables can be saved for later reuse; the uncompressed disk usage for khmer in Figure 3.3 is the same as its

memory. At the expense of more time, khmer supports saving and loading gzip-compressed hash tables, which are competitive in size to DSK's on-disk database (Figure 3, dashed line).

3.3.2 khmer memory usage is fixed and low

The memory usage of the basic Count-Min Sketch approach is fixed: khmer's memory usage does not increase as data is loaded. While this means that khmer will never crash due to memory limitations, and all operations can be performed in main memory without recourse to disk storage, the false positive rate may grow too high. Therefore the memory size must be chosen in light of the false positive rate and miscount acceptable for a given application. In practice, we recommend choosing the maximum available memory, because the false positive rate decreases with increasing memory and there are no negative effects to minimizing the false positive rate.

For any given data set, the size and number of hash tables will determine the accuracy of k-mer counting with khmer. Thus, the user can control the memory usage based on the desired level of accuracy (Figure 3.2). The time usage for the first step of k-mer counting, consuming the reads, depends on the total amount of data, since we must traverse every k-mer in every read. The second step, k-mer retrieval, is algorithmically constant for fixed k; however, for practicality, the hash tables are usually saved to and loaded from disk, meaning that k-mer retrieval time depends directly on the size of the database being queried.

The memory usage of khmer is particularly low for sparse data sets, especially since only main memory is used and no disk space is necessary beyond that required for the read data sets. This is no surprise: the information theoretic comparison in [96] shows that, for sparse sequencing data sets, Bloom filters require considerably less memory than any possible exact information storage for a wide range of false positive rates and data set sparseness.

In our implementation we use 1 byte to store the count of each k-mer in the data structure. Thus the maximum count for a k-mer will be 255. In cases where tracking bigger counts is required, khmer also provides an option to use an STL map data structure to store counts above 255, with the trade-off of significantly higher memory usage. In the future, we may extend khmer to counters of arbitrary bit sizes.

The memory usage of khmer also increases linearly with data set size as long as we hold the false positive rate constant. However, the memory usage of khmer varies substantially with the desired false positive rate: we can decrease the memory usage by increasing the false positive rate as shown in Figure 3.2. We also see that with a low false positive of 1%, the memory usage is competitive with Tallymer and Jellyfish; with a higher 5% false positive rate, the memory usage is lower than all but the disk-based DSK; with an false positive rate as high as 20%, the memory usage is further lower, close to DSK, KAnalyze, and KMC.

3.3.3 khmer accesses k-mer counts efficiently

We measured the time it took to access 9.7m 22-mers across five different data sets after the initial databases had been built (Figure 3.4). Note that Tallymer, Jellyfish, and khmer all support random access to k-mer counts, while BFCounter, DSK, KMC, Turtle and KAnalyze do not. Here, khmer performed well, dramatically outperforming Jellyfish and Tallymer. In all three cases, system time dominated the overall time required to retrieve k-mers, suggesting that the primary reason for the increase in retrieval time was due to the increased size of the database on the disk (data not shown). In particular, khmer is independent of the size of the database in retrieval time once the hash tables are loaded into memory.

This highly memory- and time-efficient online counting is particularly important for the streaming approaches to data analysis needed as data set sizes increase, like digital nor-

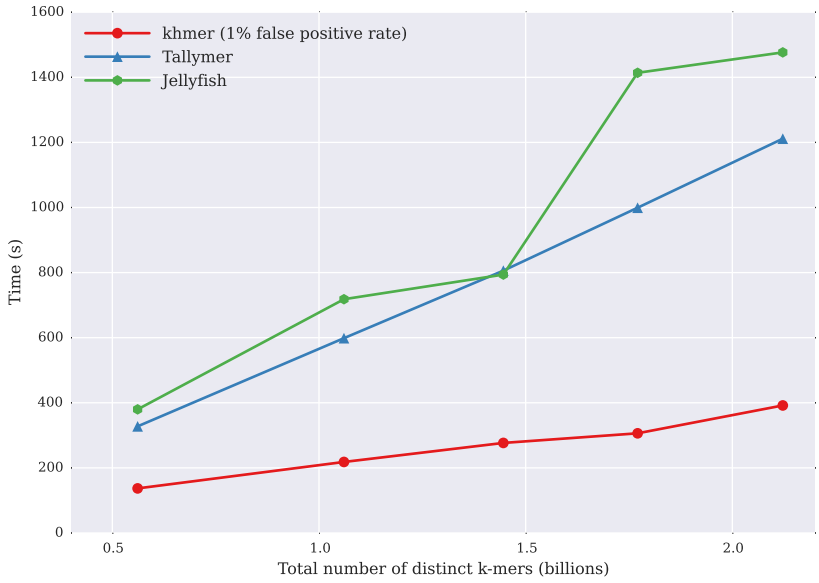

Figure 3.4: **Time for several k-mer counting tools to retrieve the counts of 9.7m randomly chosen k-mers (y axis), plotted against the number of distinct k-mers in the data set being queried (x axis). BFCounter, DSK, Turtle, KAnalyze, and KMC do not support this functionality.**

malization which will be discussed in next chapter and the IGS based method to analyze microbial diversity which will be discussed in later chapters. Because query and updating of k-mer counts can be done directly as data is being loaded, with no need for disk access or an indexing step, khmer can also perform well in situations with poor disk I/O performance. (Note that BFCounter also supports online k-mer counting [85].)

3.4 False positive rates in k-mer counting are low and predictable

The Count-Min Sketch is a probabilistic data structure with a one-sided error that results in random overestimates of k-mer frequency, but does not generate underestimates. Next

we will discuss the characteristics of such counting inaccuracy and the influence of such inaccuracy to the real-world applications of khmer for k-mer counting.

3.4.1 The measured counting error is low on short-read data

Due to the use of Count-Min Sketch and its lack of collision tracking, khmer will report some incorrect counts for k-mers; these counts are always higher than the true counts, up to the bound of 255 (a limit imposed by our use of 8-bit counters).

In the Count-Min Sketch, the total memory usage is fixed; the memory usage, the hash functions, and the total number of distinct objects counted all influence the accuracy of the count. While the probability of an inaccurate count can easily be estimated based on the hash table load, the miscount size is dependent on details of the frequency distribution of k-mers [26].

More specifically, in the analysis of the Count-Min Sketch, the difference between the incorrect count and actual count is related to the total number of k-mers in a data set and the size of each hash table [26]. Further study has shown that the behavior of Count-Min Sketch depends on specific characteristics of the data set under consideration, like the distribution of k-mer abundances [112, 27]. In general, the average miscount will be small if the data is left-skewed. As noted by Melsted and Pritchard, a large number of k-mers in short-read data are low-abundance, leading to precisely the skew that would yield low miscounts [85]. Here we use both real and simulated data sets (Table 3.2) to evaluate the counting performance in practice.

Figure 3.5 shows the relationship between average miscount and counting false positive rate for five different test data sets with similar numbers of distinct k-mers: one metagenome data set; a simulated set of random k-mers; a simulated set of reads, chosen with 3x coverage

data set	size of data set file	number of total k-mers	number of distinct k-mers
Real metagenomics reads	7.01M	2,917,200	1,944,996
Totally random reads with randomly generated k-mers	3.53M	2,250,006	1,973,059
Simulated reads from simulated genome with error	5.92M	3,757,479	2,133,592
Simulated reads from simulated genome without error	9.07M	5,714,973	1,989,644
Real *E. coli* reads	4.85M	4,004,911	2,079,302

Table 3.2: **Data sets used for analyzing miscounts.**

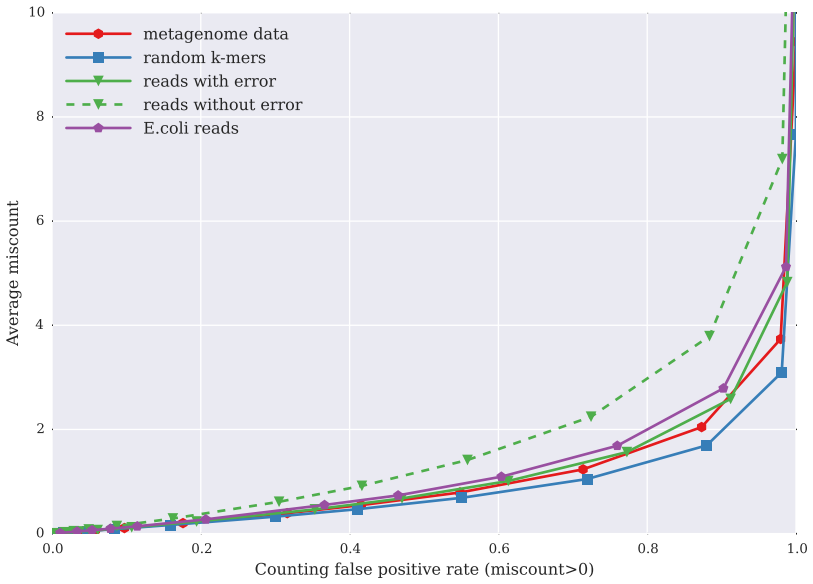

Figure 3.5: **Relation between average miscount — amount by which the count for k-mers is incorrect — on the y axis, plotted against false positive rate (x axis), for five data sets. The five data sets were chosen to have the same total number of distinct k-mers: one metagenome data set; a set of randomly generated k-mers; a set of reads, chosen with 3x coverage and 1% error, from a randomly generated genome; a simulated set of error-free reads (3x) chosen from a randomly generated genome and a set of *E. coli* reads.**

and 1% error; a simulated set of reads (3x) with no error; and a set of *E. coli* reads (Table 3.2). Even when the counting false positive rate is as high as 0.9 — where 90% of k-mers have an incorrect count — the average miscount is still below 4.

We separately analyzed the average *percentage* miscount between true and false k-mers;

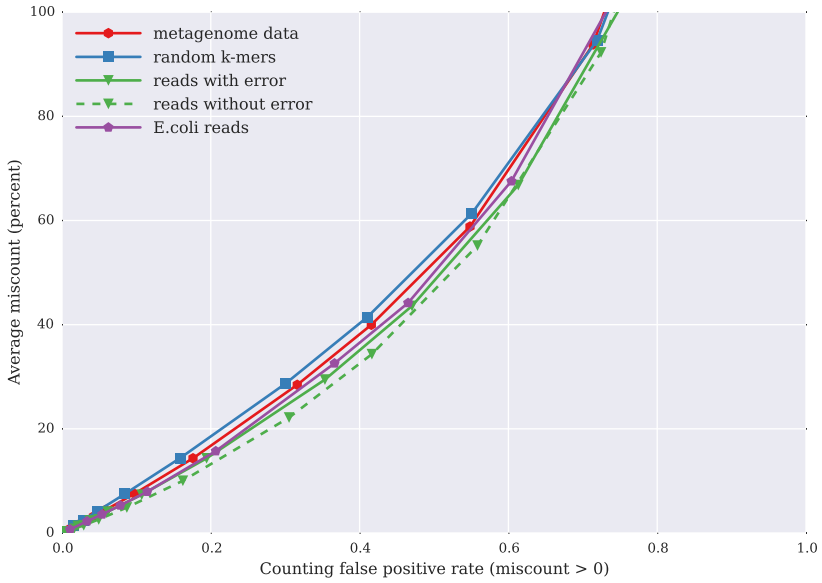

Figure 3.6: **Relation between percent miscount — amount by which the count for k-mers is incorrect relative to its true count — on the y axis, plotted against false positive rate (x axis), for five data sets. The five data sets are the same as in Figure 3.5.**

e.g. an miscount of 4 for a k-mer whose true count is 1 would be 400%. Figure 3.6 shows the relationship between average miscount and counting false positive rate for the same five data sets as in Figure 3.5. For a false positive rate of 0.1 (10% of k-mer counts are incorrect), the average percentage miscount is less than 10% for all five data sets; this will of course generally be true, because the average miscount is bounded by the product of the false positive rate with k-mer abundance.

We see here that for a fixed false positive rate, the simulated reads without error have the highest average miscount, and the randomly generated k-mers have the lowest average miscount. This is because these two abundance distributions have the least and most left-skew, respectively: the simulated reads without error have no abundance-1 k-mers, while the randomly generated k-mers are entirely low abundance. Thus, this counting approach is

especially suitable for high diversity data sets, such as metagenomic data, in which a larger proportion of k-mers are low abundance or unique due to sequencing errors.

For many applications, an approximate k-mer count is sufficient. For example, when eliminating reads with low abundance k-mers, we can tolerate a certain number of low-frequency k-mers remaining in the resulting data set falsely. If RAM-limited we can do the filtering iteratively so that at each step we are making more effective use of the available memory.

In practice, we have found that a false positive rate of between 1% and 10% offers acceptable miscount performance for a wide range of tasks, including error profiling, digital normalization and low-abundance read-trimming. Somewhat surprisingly, false positive rates of up to 80% can still be used for both read trimming and digital normalization in memory-limited circumstances, although multiple passes across the data may be needed.

For many applications, the fact that khmer does not break an imposed memory bound is extremely useful, since for many data sets — especially metagenomic data sets — high memory demands constrain analysis [50, 73]. Moreover, because the false positive rate is straightforward to measure, the user can be warned that the results should be invalidated when too little memory is used. When combined with the graceful degradation of performance for both error trimming and digital normalization, khmer readily enables analysis of extremely large and diverse data sets [52]. In an experiment to assemble the reads of a soil metagenomic sample collected from Iowa prairie, the number of reads to assemble drops from 3.3 million to 2.2 million and the size of the data set drops from 245GB to 145GB accordingly after digital normalization [50]. 240GB memory was used in the process. This also shows that khmer works well to analyze large, real-world metagenomic data sets.

3.4.2 Real-world applications of khmer

Khmer has been widely used by many research groups for solving different bioinformatics problems. It is the foundation of all the work that will be discussed in this thesis later. We will show the real-worl applications of khmer extensively in the chapters next.

For many applications, an approximate k-mer count is sufficient. For example, when eliminating reads with low abundance k-mers, we can tolerate a certain number of low-frequency k-mers remaining in the resulting data set falsely. If RAM-limited we can do the filtering iteratively so that at each step we are making more effective use of the available memory.

In practice, we have found that a false positive rate of between 1% and 10% offers acceptable miscount performance for a wide range of tasks, including error profiling, digital normalization and low-abundance read-trimming. Somewhat surprisingly, false positive rates of up to 80% can still be used for both read trimming and digital normalization in memory-limited circumstances, although multiple passes across the data may be needed.

For many applications, the fact that khmer does not break an imposed memory bound is extremely useful, since for many data sets — especially metagenomic data sets — high memory demands constrain analysis [50, 73]. Moreover, because the false positive rate is straightforward to measure, the user can be warned that the results should be invalidated when too little memory is used. When combined with the graceful degradation of performance for both error trimming and digital normalization, khmer readily enables analysis of extremely large and diverse data sets [52]. In an experiment to assemble the reads of a soil metagenomic sample collected from Iowa prairie, the number of reads to assemble drops from 3.3 million to 2.2 million and the size of the data set drops from 245GB to 145GB

accordingly after digital normalization [50]. 240GB memory was used in the process. This also shows that khmer works well to analyze large, real-world metagenomic data sets.

3.5 Conclusion

K-mer counting is widely used in bioinformatics, and as sequencing data set sizes increase, graceful degradation of data structures in the face of large amounts of data has become important. This is especially true when the theoretical and practical effects of the degradation can be predicted (see e.g. [85, 96, 111]). This is a key property of the Count-Min Sketch approach, and its implementation in khmer.

The khmer software implementation offers good performance, a robust and well-tested Python API, and a number of useful and well-documented scripts. While Jellyfish, DSK, KMC, and Turtle also offer good performance, khmer is competitive, and, because it provides a Python API for online counting, is flexible. In memory-limited situations with poor I/O performance, khmer is particularly useful, because it will not break an imposed memory bound and does not require disk access to store or retrieve k-mer counts. However, in exchange for this memory guarantee, counting becomes increasingly incorrect as less memory is used or as the data set size grows large; in many situations this may be an acceptable tradeoff.

3.6 Data

3.6.1 Code and data set availability

The khmer software [28] is implemented in C++ in a Python wrapper, enabling flexible use and reuse by users with a wide range of computational expertise. The software package is freely available for academic and commercial use and redistribution under the BSD license at github.com/ged-lab/khmer/. khmer comes with substantial documentation and many tutorials, and contains extensive unit tests. Moreover, we have built several applications on top of khmer, including memory-efficient de Bruijn graph partitioning [96] and lossy compression of short-read data sets for assembly [10].

The version of khmer used to generate the results in this chapter is available at http://github.com/ged-lab/khmer.git, tag '2013-khmer-counting'. Scripts specific to this paper are available in the paper repository at https://github.com/ged-lab/2013-khmer-counting. The IPython[98] notebook file and data analysis to generate the figures are also available in that github repository. Complete instructions to reproduce all of the results in this paper are available in the khmer-counting repository; see README.rst.

3.6.2 Sequence data

One human gut metagenome reads data set (MH0001) from the MetaHIT (Metagenomics of the Human Intestinal Tract) project [103] was used. It contains approximately 59 million reads, each 44bp long; it was trimmed to remove low quality sequences.

Five soil metagenomics reads data sets with different size were taken from the GPGC project for benchmark purpose (see Table 3.1). These reads are from soil in Iowa region and

they are filtered to make sure there are less than 30% Ns in the read and each read is longer than 30 bp. The exact data sets used for the paper are available on Amazon S3 and the instructions to acquire these data sets are available in the paper repository on github.com.

We also generated four short-read data sets to assess the false positive rate and miscount distribution. One is a subset of a real metagenomics data set from the MH0001 data set, above. The second consists of randomly generated reads. The third and fourth contain reads simulated from a random, 1 Mbp long genome. The third has a substitution error rate of 3%, and the fourth contains no errors. The four data sets were chosen to contain identical numbers of distinct 22-mers. The scripts necessary to regenerate these data are available in the paper repository on github.com.

Chapter 4

A framework for streaming analysis of short DNA sequencing reads based on k-mer counting

4.1 Introduction

In the previous chapter, we introduced an efficient k-mer counting approach based on a probabilistic data structure. In this chapter, we will discuss the application of this approach to enable streaming analysis of short DNA sequencing reads. First, we will show a novel approach to use median k-mer count in a read to estimate sequencing depth without a reference assembly. Next, based on this approach, two streaming methods that are critically important in next generation sequencing data analysis will be discussed. One is the single-pass method to eliminate redundant reads in data sets to reduce computational cost in down-streaming analysis like assembly, which is termed as "digital normalization". The other one is the method to analyze and trim sequencing errors in short reads data sets, in a semi-streaming or streaming fashion.

The approach to use median k-mer count to estimate sequencing depth of a read is also the foundation of the IGS based diversity analysis approach we will discuss in the next chap-

ter. The streaming methods to remove redundant reads and sequencing error analysis and trimming are not directly related to the IGS based diversity analysis method, nevertheless the applications of the two streaming methods to facilitate the assembly of metagenomic reads and remove sequencing errors in reads data sets benefit many bioinformatics approaches, including the microbial diversity analysis.

This chapter contains published materials from [148] [10] and [147].

4.2 Estimating sequencing depth without a reference assembly

Short-read assembly requires deep sequencing to systematically sample the source genome, because shotgun sequencing is subject to both random sampling variation and systematic sequencing biases. For example, 100x sampling of a human genome is required for recovery of 90% or more of the genome in contigs > 1kb [39]. In principle, much of this high-coverage data is redundant and could be eliminated without consequence to the final assembly. However, determining which reads to eliminate requires a per-read estimate of coverage. Traditional approaches estimate coverage by mapping reads to an assembly. This presents a chicken-and-egg problem: to determine which regions are oversampled, we must already have an assembly!

We may calculate a *reference-free* estimate of genome coverage by looking at the k-mer abundance distribution within individual reads. First, observe that k-mers, DNA words of a fixed length k, tend to have similar abundances within a read: this is a well-known property of k-mers that stems from each read originating from a single source molecule of DNA. The more times a region is sequenced, the higher the abundance of k-mers from that region

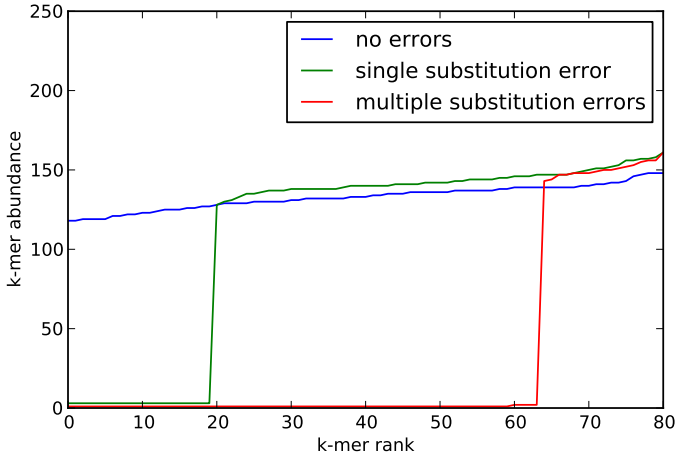

Figure 4.1: **Representative rank-abundance distributions for 20-mers from 100-base reads with no errors, a read with a single substitution error, and a read with multiple substitution errors.**

would be. In the absence of errors, average k-mer abundance could be used as an estimate of the depth of coverage for a particular read (Figure 4.1, "no errors" line). However, when reads contain random substitution or indel errors from sequencing, the k-mers overlapping these errors will be of lower abundance; this feature is often used in k-mer based error correction approaches [62]. For example, a single substitution introduces k low-abundance k-mers within a read. (Figure 4.1, "single substitution error" line). However, for small k and reads of length L where $L > 3k - 1$, a single substitution error will not skew the *median* k-mer abundance. Only when multiple substitution errors are found in a single read will the median k-mer abundance be affected (Figure 4.1, "multiple substitution errors"). The effect of multiple errors to median k-mer abundance will be discussed in details in the next chapter.

Using a fixed-memory CountMin Sketch data structure to count k-mers (see [25]), we find that median k-mer abundance correlates well with mapping-based coverage for artificial and real genomic data sets. There is a strong correlation between median k-mer abundance

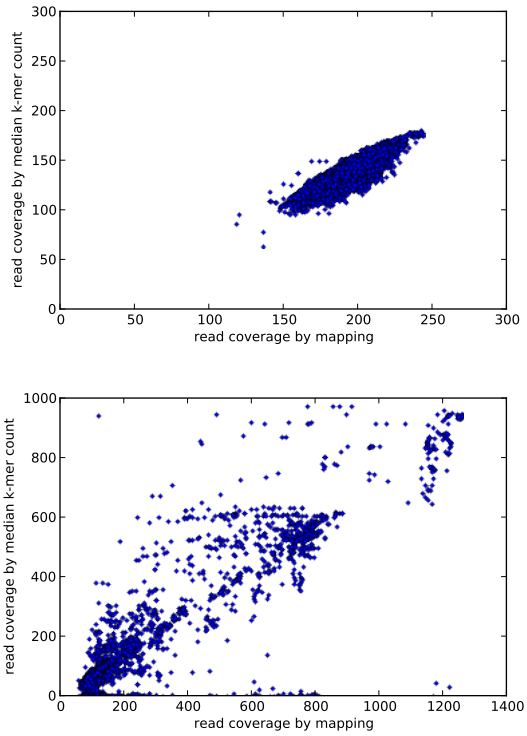

Figure 4.2: **Mapping and k-mer coverage measures correlate for simulated genome data and a real *E. coli* data set (5m reads). Simulated data $r^2 = 0.79$; *E. coli* $r^2 = 0.80$.**

and mapping-based coverage both for simulated 100-base reads generated with 1% error from a 400kb artificial genome sequence ($r^2 = 0.79$; also see Figure 4.2a), as well as for real short-read data from *E. coli* ($r^2 = 0.80$, also see Figure 4.2b). This correlation also holds for simulated and real mRNAseq data: for simulated transcriptome data, $r^2 = 0.93$ (Figure 4.3a), while for real mouse transcriptome data, $r^2 = 0.90$ (Figure 4.3b). Thus the median k-mer abundance of a read correlates well with mapping-based estimates of read coverage.

The coverage on read level estimated from median k-mer count of a read is always smaller than the mapping-based estimates of read coverage, which is essentially the coverage on nucleotide level. There is a way to convert the coverage on read level into real sequencing

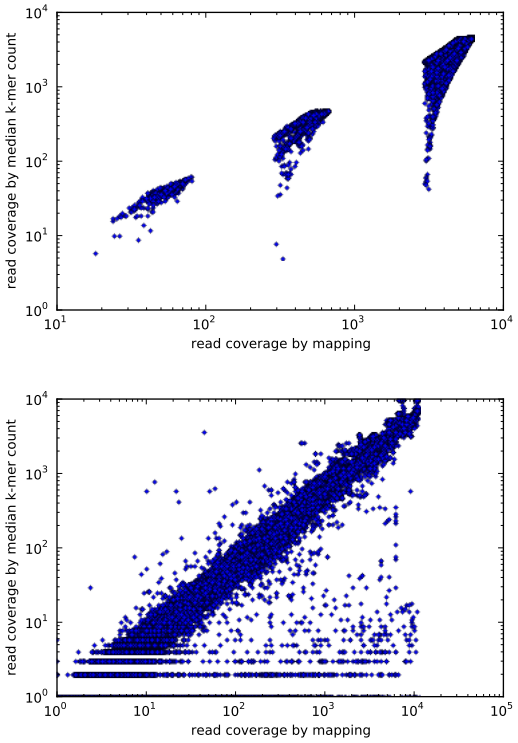

Figure 4.3: **Mapping and k-mer coverage measures correlate for simulated transcriptome data as well as real mouse transcriptome data. Simulated data $r^2 = 0.93$; mouse transcriptome $r^2 = 0.90$.**

depth (coverage on nucleotide level). To cover a k-mer by a read, all the nucleotide in this k-mer must be covered by the read. If the coverage in nucleotide level is C_N, the coverage of a k-mer in a genome will be C_N * (L-k+1)/L, with length of reads as L. [62] Figure 4.2 and Figure 4.3 shows such relationship obviously.

Such difference between coverage on read level and on nucleotide level is important in the IGS based diversity analysis method, which will be discussed in more details in the next chapter. For the streaming methods to analyze short reads data discussed in this chapter, the coverage means the coverage on read level estimated from median k-mer abundance in a read.

4.3 A streaming algorithm to digitally normalize the coverage distribution of data sets

Below, we introduce "digital normalization", a single-pass lossy compression algorithm for elimination of redundant reads in data sets based on saturating coverage of a de Bruijn graph. While several non-streaming implementations exist, including Trinity's *in silico* normalization [44, 9], digital normalization can be efficiently implemented as a *streaming* algorithm. Critically, no reference sequence is needed to apply digital normalization. Digital normalization is inspired by experimental normalization techniques developed for cDNA library preparation, in which hybridization kinetics are exploited to reduce the copy number of abundant transcripts prior to sequencing [6, 127]. *Digital* normalization works after sequencing data has been generated, progressively removing high-coverage reads from shotgun data sets. This normalizes average coverage to a specified value, reducing sampling variation while removing reads, and also removing the many errors contained *within* those reads. This data and error reduction results in dramatically decreased computational requirements for *de novo* assembly. This has the advantage of enabling low-memory preprocessing of both high-coverage genomic data sets, as well as mRNAseq or metagenomic data sets with high-coverage components [10, 50]. Moreover, unlike experimental normalization where abundance information is removed prior to sequencing, in digital normalization this information can be recovered from the unnormalized reads.

We present here a fixed-memory implementation of digital normalization that operates in time linear with the size of the input data. We then demonstrate its effectiveness for reducing computational requirements for *de novo* assembly on several real data sets. These data sets include *E. coli* genomic data, data from two single-cell MD-amplified microbial

genomes, and yeast and mouse mRNAseq.

4.3.1 Eliminating redundant reads reduces variation in sequencing depth

Deeply sequenced genomes contain many highly covered loci. For example, in a human genome sequenced to 100x average coverage, we would expect 50% or more of the reads to have a coverage greater than 100. In practice, we need many fewer of these reads to assemble the source locus.

Using the median k-mer abundance estimator discussed above, we can examine each read in the data set progressively to determine if it is high coverage. At the beginning of a shotgun data set, we would expect many reads to be entirely novel and have a low estimated coverage. As we proceed through the data set, however, average coverage will increase and many reads will be from loci that we have already sampled sufficiently.

Suppose we choose a coverage threshold C past which we no longer wish to collect reads. If we only keep reads whose estimated coverage is less than C, and discard the rest, we will reduce the average coverage of the data set to C. This procedure is algorithmically straightforward to execute: we examine each read's estimated coverage, and retain only those whose coverage is less than C. The following pseudocode provides one approach:

```
for read in dataset:
   if estimated_coverage(read) < C:
      accept(read)
   else:
      discard(read)
```

where accepted reads contribute to the `estimated_coverage` function. Note that for any data set with an average coverage $> 2C$, this has the effect of discarding the majority of reads. Critically, low-coverage reads, especially reads from undersampled regions, will always be retained.

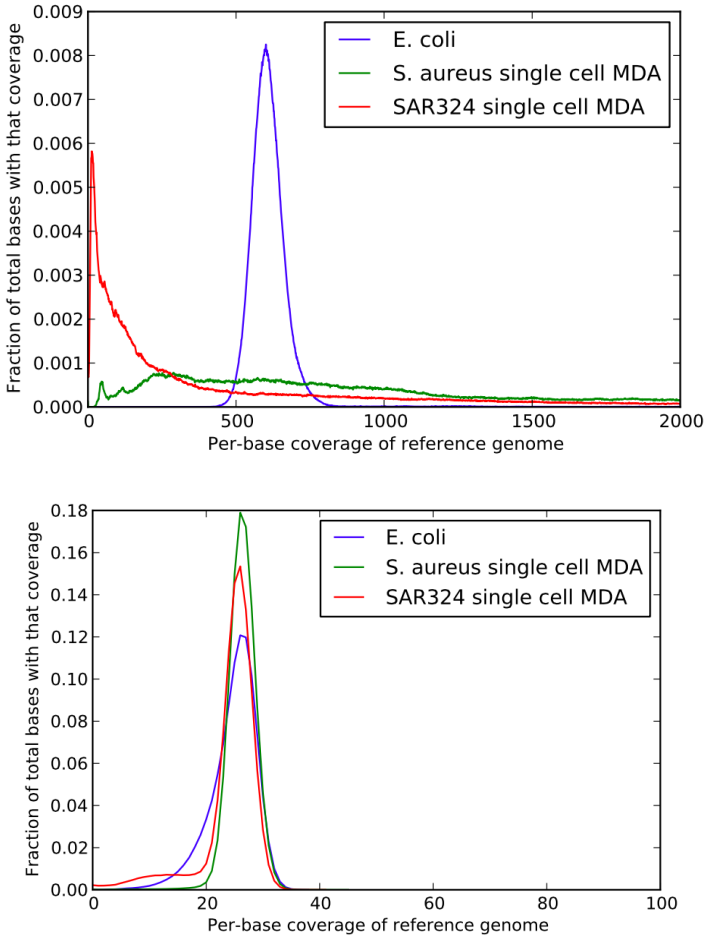

Figure 4.4: **Coverage distribution of three microbial genome samples, calculated from mapped reads (a) before and (b) after digital normalization (k=20, C=20).**

The net effect of this procedure, which we call digital normalization, is to normalize the coverage distribution of data sets. In Figure 4.4a, we display the estimated coverage of an *E. coli* genomic data set, a *S. aureus* single-cell MD-amplified data set, and an MD-amplified data set from an uncultured *Deltaproteobacteria*, calculated by mapping reads to the known

or assembled reference genomes (see [19] for the data source). The wide variation in coverage for the two MDA data sets is due to the amplification procedure [130]. After normalizing to a k-mer coverage of 20, the high coverage loci are systematically shifted to an average mapping coverage of 26, while lower-coverage loci remain at their previous coverage. This smooths out coverage of the overall data set.

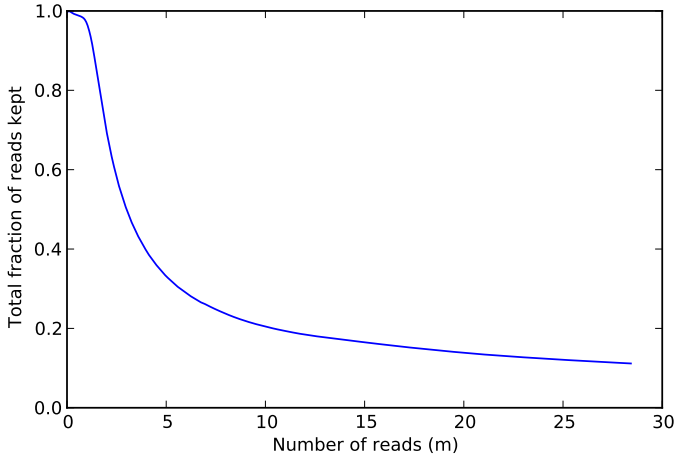

Figure 4.5: **Fraction of reads kept when normalizing the *E. coli* dataset to C=20 at k=20.**

At what rate are sequences retained? For the *E. coli* data set, Figure 4.5 shows the fraction of sequences retained by digital normalization as a function of the total number of reads examined when normalizing to C=20 at k=20. There is a clear saturation effect showing that as more reads are examined, a smaller fraction of reads is retained; by 5m reads, approximately 50-100x coverage of *E. coli*, under 30% of new reads are kept. This demonstrates that as expected, only a small amount of novelty (in the form of either new information, or the systematic accumulation of errors) is being observed with increasing sequencing depth.

Here we show that digital normalization provides a general strategy for applying online or streaming approaches to analysis of *de novo* sequencing data. The basic algorithm pre-

sented here is explicitly a single-pass or streaming algorithm, in which the entire data set is never considered as a whole; rather, a partial "sketch" of the data set is retained and used for progressive filtering. Online algorithms and sketch data structures offer significant opportunities in situations where data sets are too large to be conveniently stored, transmitted, or analyzed [92]. This can enable increasingly efficient downstream analyses. Digital normalization can be applied in any situation where the abundance of particular sequence elements is either unimportant or can be recovered more efficiently after other processing, as in assembly, which we will discuss next.

4.3.2 Digital normalization scales assembly of microbial genomes

We applied the digital normalization and error trimming protocol to three real data sets from Chitsaz et al (2011) [19]. For all three samples, the number of reads remaining after digital normalization was reduced by at least 30-fold, while the memory and time requirements were reduced 10-100x.

Despite this dramatic reduction in data set size and computational requirements for assembly, both the *E. coli* and *S. aureus* assemblies overlapped with the known reference sequence by more than 98%. This confirms that little or no information was lost during the process of digital normalization; moreover, it appears that digital normalization does not significantly affect the assembly results. (Note that we did not perform scaffolding, since the digital normalization algorithm does not take into account paired-end sequences, and could mislead scaffolding approaches. Therefore, these results cannot directly be compared to those in Chitsaz et al. (2011) [19].)

The *Deltaproteobacteria* sequence also assembled well, with 98.8% sequence overlap with the results from Chitsaz et al. Interestingly, only 30kb of the sequence assembled with

Velvet-SC in Chitsaz et al. (2011) was missing, while an additional 360kb of sequence was assembled only in the normalized samples. Of the 30kb of missing sequence, only 10% matched via TBLASTX to a nearby *Deltaproteobacteria* assembly, while more than 40% of the additional 360kb matched to the same *Deltaproteobacteria* sample. Therefore these additional contigs likely represent real sequence, suggesting that digital normalization is competitive with Velvet-SC in terms of sensitivity.

Data set	N reads pre/post	Assembly time pre/post	Assembly memory pre/post
Yeast (Oases)	100m / 9.3m	181 min / 12 min (15.1x)	45.2gb / 8.9gb (5.1x)
Yeast (Trinity)	100m / 9.3m	887 min / 145 min (6.1x)	31.8gb / 10.4gb (3.1x)
Mouse (Oases)	100m / 26.4m	761 min/ 73 min (10.4x)	116.0gb / 34.6gb (3.4x)
Mouse (Trinity)	100m / 26.4m	2297 min / 634 min (3.6x)	42.1gb / 36.4gb (1.2x)

Table 4.1: **Single-pass digital normalization to C=20 reduces computational requirements for transcriptome assembly.**

4.3.3 Digital normalization scales assembly of transcriptomes

We next applied single-pass digital normalization to published yeast and mouse mRNAseq data sets, reducing them to 20x coverage at k=20 [41]. Digital normalization on these samples used 8gb of memory and took about 1 min per million reads. We then assembled both the original and normalized sequence reads with Oases and Trinity, two *de novo* transcriptome assemblers (Table 4.1) [118, 41]. For both assemblers the computational resources necessary to complete an assembly were reduced (Table 4.1). However, normalization had different effects on performance for the different samples. On the yeast data set, time and memory requirements were reduced significantly, as for Oases running on mouse. However, while Trinity's runtime decreased by a factor of three on the normalized mouse data set, the memory requirements did not decrease significantly. This may be because the mouse transcriptome is 5-6 times larger than the yeast transcriptome, and so the mouse mRNAseq

is lower coverage overall; in this case we would expect fewer errors to be removed by digital normalization.

Data set	Contigs > 300	Total bp > 300	Contigs > 1000	Total bp > 1000
Yeast (Oases)	12,654 / 9,547	33.2mb / 27.7mb	9,156 / 7,345	31.2mb / 26.4mb
Yeast (Trinity)	10,344 / 12,092	16.2mb / 16.5mb	5,765 / 6,053	13.6 mb / 13.1mb
Mouse (Oases)	57,066 / 49,356	98.1mb / 84.9mb	31,858 / 27,318	83.7mb / 72.4mb
Mouse (Trinity)	50,801 / 61,242	79.6 mb / 78.8mb	23,760 / 24,994	65.7mb / 59.4mb

Table 4.2: **Digital normalization has assembler-specific effects on transcriptome assembly.**

The resulting assemblies differed in summary statistics (Table 4.2). For both yeast and mouse, Oases lost 5-10% of total transcripts and total bases when assembling the normalized data. However, Trinity *gained* transcripts when assembling the normalized yeast and mouse data, gaining about 1% of total bases on yeast and losing about 1% of total bases in mouse. Using a local-alignment-based overlap analysis, we found little difference in sequence content between the pre- and post- normalization assemblies: for example, the normalized Oases assembly had a 98.5% overlap with the unnormalized Oases assembly, while the normalized Trinity assembly had a 97% overlap with the unnormalized Trinity assembly.

To further investigate the differences between transcriptome assemblies caused by digital normalization, we looked at the sensitivity with which long transcripts were recovered post-normalization. When comparing the normalized assembly to the unnormalized assembly in yeast, Trinity lost only 3% of the sequence content in transcripts greater than 300 bases. However, 10% of the sequence content in transcripts greater than 1000 bases. However, Oases lost less than 0.7% of sequence content at 300 and 1000 bases. In mouse, we see the same pattern. This suggests that the change in summary statistics for Trinity is caused by fragmentation of long transcripts into shorter transcripts, while the difference for Oases is caused by loss of splice variants. Indeed, this loss of splice variants should be expected, as

there are many low-prevalence splice variants present in deep sequencing data [100]. Interestingly, in yeast we recover *more* transcripts after digital normalization; these transcripts appear to be additional splice variants.

Data set	True 20-mers	20-mers in reads	20-mers at C=20	% reads kept
Simulated genome	399,981	8,162,813	3,052,007 (-2)	19%
Simulated mRNAseq	48,100	2,466,638 (-88)	1,087,916 (-9)	4.1%
E. coli genome	4,542,150	175,627,381 (-152)	90,844,428 (-5)	11%
Yeast mRNAseq	10,631,882	224,847,659 (-683)	10,625,416 (-6,469)	9.3%
Mouse mRNAseq	43,830,642	709,662,624 (-23,196)	43,820,319 (-13,400)	26.4%

Table 4.3: **Digital normalization to C=20 removes many erroneous k-mers from sequencing data sets. Numbers in parentheses indicate number of true k-mers lost at each step, based on reference.**

Data set	True 20-mers	20-mers in reads	20-mers remaining	% reads kept
Simulated genome	399,981	8,162,813	453,588 (-4)	5%
Simulated mRNAseq	48,100	2,466,638 (-88)	182,855 (-351)	1.2%
E. coli genome	4,542,150	175,627,381 (-152)	7,638,175 (-23)	2.1%
Yeast mRNAseq	10,631,882	224,847,659 (-683)	10,532,451 (-99,436)	2.1%
Mouse mRNAseq	43,830,642	709,662,624 (-23,196)	42,350,127 (-1,488,380)	7.1%

Table 4.4: **Three-pass digital normalization removes most erroneous k-mers. Numbers in parentheses indicate number of true k-mers lost at each step, based on known reference.**

The difference between Oases and Trinity results show that Trinity is more sensitive to digital normalization than Oases: digital normalization seems to cause Trinity to fragment long transcripts. One potential issue is that Trinity only permits k=26 for assembly, while normalization was performed at k=20; digital normalization may be removing 26-mers that are important for Trinity's path finding algorithm. Alternatively, Trinity may be more sensitive than Oases to the change in coverage caused by digital normalization. Regardless, the strong performance of Oases on digitally normalized samples, as well as the high retention of k-mers (Table 4.3) suggests that the primary sequence content for the transcriptome remains present in the normalized reads, although it is recovered with different effectiveness by the two assemblers.

4.3.4 lower bound on memory usage for effective digital normalization

In this section, we will discuss the lower bound on memory usage for effective digital normalization and the effects of high false positive rates particularly.

memory	FP rate	retained reads	retained reads %	true k-mers missing	total k-mers
before diginorm	-	5,000,000	100.0%	170	41.6m
2400 MB	0.0%	1,656,518	33.0%	172	28.1m
240 MB	2.8%	1,655,988	33.0%	172	28.1m
120 MB	18.0%	1,652,273	33.0%	172	28.1m
60 MB	59.1%	1,633,182	32.0%	172	27.9m
40 MB	83.2%	1,602,437	32.0%	172	27.6m
20 MB	98.8%	1,460,936	29.0%	172	25.7m
10 MB	100.0%	1,076,958	21.0%	185	20.9m

Table 4.5: **Low-memory digital normalization. The results of digitally normalizing a 5m read *E. coli* data set (1.4 GB) to C=20 with k=20 under several memory usage/false positive rates. The false positive rate (column 1) is empirically determined. We measured reads remaining, number of "true" k-mers missing from the data at each step, and the number of total k-mers remaining. Note: at high false positive rates, reads are erroneously removed due to inflation of k-mer counts.**

memory	FP rate	N contigs	total length(bases)	% of true genome covered
before diginorm	-	106	4,546,051	97.84%
2400 MB	0.0%	617	4,549,235	98.05%
240 MB	2.8%	87	4,549,253	98.04%
120 MB	18.0%	86	4,549,335	98.04%
60 MB	59.1%	90	4,548,619	98.03%
40 MB	83.2%	89	4,550,599	98.11%
20 MB	98.8%	85	4,550,014	98.04%
10 MB	100.0%	97	4,545,871	97.97%

Table 4.6: ***E. coli* genome assembly after low-memory digital normalization. A comparison of assembling reads digitally normalized with low memory/high false positive rates. The reads were digitally normalized to C=20 (see [10] for more information) and were assembled using Velvet. We measured total length of assembly, as well as percent of true MG1655 genome covered by the assembly using QUAST.**

We applied digital normalization to the *E. coli* data set used above, and chose seven

different Count-Min Sketch sizes to yield seven different false positive rates 4.5. The data set was normalized to a k-mer coverage of 20 and the resulting data were evaluated for retention of true and erroneous k-mers, as in [10] (Table 4.5). The results show that digital normalization retains the same set of underlying "true" k-mers until the highest false positive rate of 100% (Table 4.5, column 5), while discarding only about 2% additional reads (Table 4.5, column 6).

To evaluate the effect of digital normalization with high false positive rates on actual genome assembly, we next performed normalization to a coverage of 20 with the same range of false positive rates as above. We then assembled this data with Velvet [146] and compared the resulting assemblies to the known *E. coli* MG1655 genome using QUAST [**?**] (Table 4.6). To our surprise, we found that even after executing digital normalization with a false positive rate of 83.2%, a nearly complete assembly was generated. No progressive increase in misassemblies (measured against the real genome with QUAST) was seen across the different false positive rates (data not shown). This suggests that below 83.2% FP rate, the false positive rate of digital normalization has little to no effect on assembly quality with Velvet. (Note that the Velvet assembler itself used considerably more memory than digital normalization.)

While these results are specific to Velvet and the coverage parameters used in digital normalization, they do suggest that no significant information loss occurs due to false positive rates below 80%. Further evaluation of assembly quality in response to different normalization parameters and assemblers is beyond the scope of this dissertation.

4.3.5 Digital normalization dramatically scales *de novo* assembly

The results from applying digital normalization to read data sets prior to *de novo* assembly are extremely good: digital normalization reduces the computational requirements (time and memory) for assembly considerably, without substantially affecting the assembly results. It does this in two ways: first, by removing the majority of reads without significantly affecting the true k-mer content of the data set. Second, by eliminating these reads, digital normalization also eliminates sequencing errors contained within those reads, which otherwise would add significantly to memory usage in assembly [24].

Digital normalization also lowers computational requirements by eliminating most repetitive sequence in the data set. Compression-based approaches to graph storage have demonstrated that compressing repetitive sequence also effectively reduces memory and compute requirements [101, 124]. Note however that *eliminating* many repeats may also have its negatives (discussed below).

Digital normalization should be an effective preprocessing approach for most assemblers. In particular, the de Bruijn graph approach used in many modern assemblers relies on k-mer content, which is almost entirely preserved by digital normalization (see Tables 4.3 and 4.4) [87].

4.4 A streaming algorithm to analyze and trim errors in short reads .

K-mer spectral analysis is a powerful approach to error detection and correction in shotgun sequencing data that uses k-mer abundances to find likely errors in the data [99, 68]. The

essential idea is that low-abundance k-mers contained in a high-coverage data set typically represent random sequencing errors. A variety of read trimming and error correcting tools use k-mer counting to reduce the error content of the read data set, independently of quality scores or reference genomes [62].

Approaches derived from spectral analysis can be very effective: spectral error correction achieves high accuracy, and later we will show that spectral k-mer trimming is considerably more effective at removing errors than quality score-based approaches. However, spectral analysis is also very computational intensive: most implementations count all the k-mers in sequencing data sets, which can be memory- or I/O-intensive for large data sets.

In the section above, we discussed a streaming algorithm for downsampling read data sets to normalize read coverage spectra, termed "digital normalization". This procedure estimates the k-mer coverage of each read in a stream using an online algorithm. Reads above a certain estimated coverage are set aside and their k-mers are not tracked. The diginorm algorithm only examines the data once, and counts only the k-mers in retained reads, leading to sublinear memory usage for high-coverage data sets.

Here we introduce a semi-streaming algorithm for k-mer spectral analysis, based on digital normalization, that can detect and remove errors in sequencing reads. This algorithm operates in sublinear memory with respect to the input data, and examines the data at most twice. The approach offers a general framework for streaming sequence analysis and could be used for error correction and variant calling. Moreover, the approach can be applied generically to data sets with variable sequencing coverage such as transcriptomes, metagenomes, and amplified genomic DNA. We also provide a fully streaming approach for estimating per-position sequencing error rates in reads that operates in fixed memory and only examines part of the input data.

4.4.1 Two-pass non-streaming method to enable read error analysis

Firstly, we implemented a two-pass non-streaming method to trim read based on the new efficient k-mer counting approach we introduced in last chapter. Basically in the first pass all the reads in a data set is loaded and the counts of each k-mer are stored in the Count-Min Sketch. Then during the second pass, for each read, the count of every k-mer in it will be examined, if a k-mer with a count as 1 in the whole data set is detected, an sequencing error is detected and the read will be truncated from this k-mer to the end. In this experiment, we especially evaluated the effect of false-positive induced miscounts on read trimming. Because the Count-Min Sketch never undercounts k-mers, reads will never be erroneously trimmed at truly high-abundance k-mers; however, reads may not be trimmed correctly when miscounts inflate the count of low-abundance k-mers. In cases where many errors remain, read trimming can potentially be applied multiple times, with each round reducing the total number of k-mers and hence resulting in lower false positive rates for the same memory usage.

	FP rate	bases trimmed	distinct k-mers	unique k-mers	unique k-mers at 3' end
untrimmed	-	-	41.6m	34.1m	30.4%
khmer iteration 1	80.0%	13.5%	13.3m	6.5m	29.8%
khmer iteration 2	40.2%	1.7%	7.6m	909.9k	12.3%
khmer iteration 3	25.4%	0.3%	6.8m	168.1k	3.1%
khmer iteration 4	23.2%	0.1%	6.7m	35.8k	0.7%
khmer iteration 5	22.8%	0.0%	6.6m	7.9k	0.2%
khmer iteration 6	22.7%	0.0%	6.6m	1.9k	0.0%
filter by FASTX	-	9.1%	26.6m	20.3m	26.3%
filter by seqtk(default)	-	8.9%	17.7m	12.1m	12.3%
filter by seqtk(-q 0.01)	-	15.4%	9.9m	5.1m	5.2%
filter by seqtk(-b 3 -e 5)	-	8.0%	34.5m	27.7m	25.3%

Table 4.7: **Iterative low-memory k-mer trimming. The results of trimming reads at unique (erroneous) k-mers from a 5m read *E. coli* data set (1.4 GB) in under 30 MB of RAM. After each iteration, we measured the total number of distinct k-mers in the data set, the total number of unique (and likely erroneous) k-mers remaining, and the number of unique k-mers present at the 3' end of reads.**

We performed six iterations of unique k-mer trimming on 5 million Illumina reads from sequencing of *E. coli*, with memory usage less than 30 MB. For each iteration we measured empirical false positive rate compared with number of bases trimmed as well as the total number of k-mers (Table 4.7). In the first round, the estimated false positive rate was 80.0%, and 13.5% of the total bases were removed by trimming reads at low-abundance k-mers; the second iteration had a false positive rate of 37.7%, and removed only 1.5% additional data; and by the fourth iteration the false positive rate was down to 23.2% with 0.0% of the data removed.

The elimination of so many unique k-mers (column 5) in the first pass was unexpected: the high false positive rate should have resulted in fewer k-mers being identified as unique, were the erroneous k-mers independent of each other. Upon examination, we realized that in Illumina data erroneous k-mers typically come from substitution errors that yield runs of up to k erroneous k-mers in a row [62]. When trimming reads with high false positive rates, these runs are typically trimmed after the first few unique k-mers, leaving unique k-mers at the 3' end. Because of this we hypothesized that high-FP rate trimming would result in the retention of many unique k-mers at the 3' end of the read, and this was confirmed upon measurement (Table 4.7, column 6, pass 1 vs pass 2).

In comparison to quality-based trimming software such as seqtk and FASTX, trimming at unique k-mers performed very well: in this data set, all unique k-mers represent errors, and even with an initial false positive rate of 80%, khmer outperformed all but the most stringent seqtk run (Table 4.7). With a lower false positive rate or multiple passes, khmer eliminates more erroneous k-mers than seqtk or FASTX. The tradeoff here is in memory usage: for larger data sets, seqtk and FASTX will consume the same amount of memory as on smaller data sets, while khmer's memory usage will need to grow with the data set size.

With Illumina sequencing, average and per-position error rates may vary between sequencing runs, but are typically systematic within a run [61]. Melsted and Halldorson (2014) introduced an efficient streaming approach to estimating per-run sequencing error, but this approach does not apply to error rates by position within reads [84]. Here, k-mer spectral error analysis can be used to calculate per-position relative sequencing error for entire data sets.

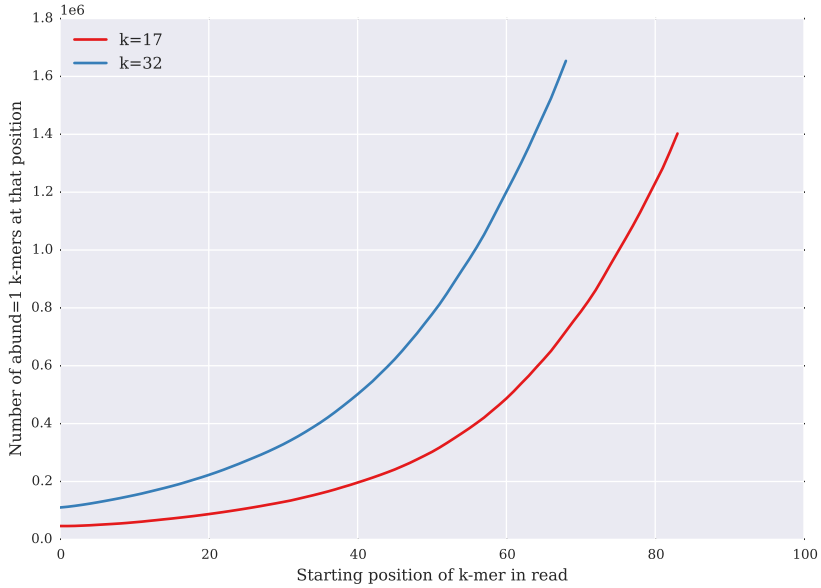

Figure 4.6: **Number of unique k-mers (y axis) by starting position within read (x axis) in an untrimmed *E. coli* 100-bp Illumina shotgun data set, for k=17 and k=32. The increasing numbers of unique k-mers are a sign of the increasing sequencing error towards the 3' end of reads. Note that there are only 69 starting positions for 32-mers in a 100 base read.**

In Figure 4.6, we use khmer to examine the sequencing error pattern of a 5m-read subset of an Illumina reads data set from single-colony sequencing of *E. coli* [19]. The high rate of occurrence of unique k-mers close to the 3' end of reads is due to the increased sequencing error rate at the 3' end of reads.

The results above demonstrated that the newly developed k-mer counting approach can

be integrated successfully to do effective error analysis. This is an application where the counting error of the Count-Min Sketch approach used by khmer may be particularly tolerable: it will never falsely call a high-abundance k-mer as low-abundance because khmer never underestimates counts.

4.4.2 A semi-streaming algorithm can be used for error analysis

As shown above, k-mer spectral error detection, trimming, and correction approaches are typically implemented as a two-pass offline algorithm, in which k-mer counts are collected in a first pass and then reads are analyzed in a second pass. While several algorithms that run in sublinear memory do exist (e.g., Lighter [129]), these are still offline algorithms that require two or more passes across the data.

In high coverage data sets it is possible to implement a more algorithmically efficient approach, by detecting reads that are high coverage in the context of reads previously encountered in the same pass of the data. Shotgun sequencing oversamples most regions – for example, for a 100x coverage genomic data set, we would expect 50% or more of the genome to be represented by more than 100 reads. This is a consequence of the Poisson-random sampling that underlies shotgun sequencing [64]. This oversampling provides an opportunity, however: if we regard the read data set as a stream of incoming data randomly sampled from a pool of molecules, high-abundance species or subsequences within the pool will be more highly sampled in the stream than others, and will thus generally appear earlier in the stream. For example, in mRNAseq, highly expressed transcripts should almost always be sampled much more frequently than low-expressed transcripts, and so more reads from highly expressed transcripts will be seen in any given subset.With this in mind, we can develop an approach to do *semi-streaming* error analysis by detecting and analyzing high-coverage reads

during the first pass.

We implemented this by integrating k-mer spectral error analysis directly into the digital normalization algorithm. As digital normalization, here we still use the median k-mer abundance of the k-mers in a read to estimate that read's abundance [10]; crucially, this can be done at any point in a stream, by using the online k-mer counting functionality of khmer to determine the abundance of k-mers seen thus far in the stream [148].

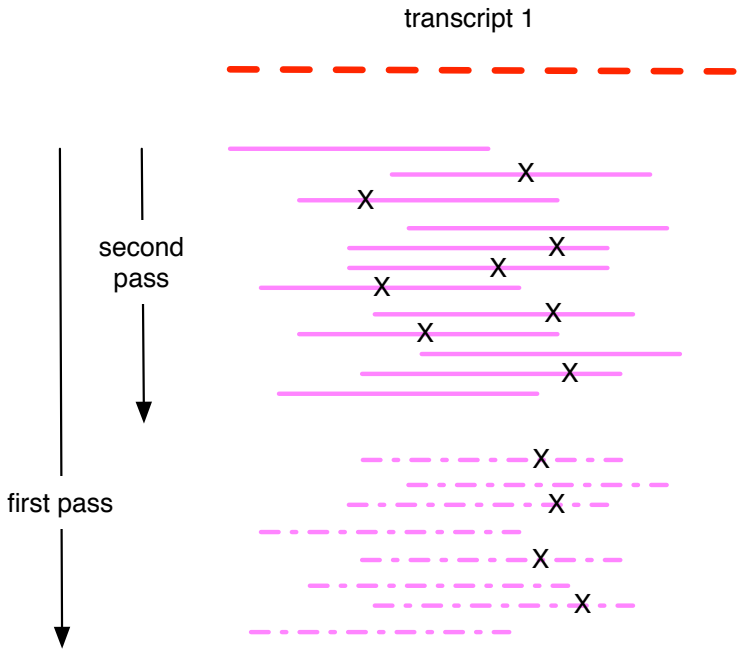

Figure 4.7: **Diagram of semi-streaming error detection. In a first pass over the read data, reads are loaded in until the graph locus to which they belong is saturated. From that point on, reads are examined for errors and not loaded into the graph. In a second pass, only the subset of reads loaded into the graph are examined for errors.**

The conceptual idea is presented in Figure 4.7. On the first pass, low-coverage reads would be incorporated into the k-mer database with all the k-mers in them loaded into memory, and set aside for later analysis, because we cannot reliably detect error, which is a low-abundance k-mer in a low-coverage read. Meanwhile, the high-coverage reads would be analyzed for errors but would not be incorporated into the k-mer database. This step is

similar to digital normalization where the high-coverage reads are discarded. Not loading the k-mers in those high-coverage reads decreases the counts of those high abundance k-mers in the k-mer database a bit but this does not affect the counts of those low abundance k-mers. So this process will not influence the detection of errors. Actually the special treatment to high coverage reads also dismisses many errors in those high coverage reads and this makes the detection of low abundance k-mers more accurate.

On the second pass, the set aside reads which were considered as low-coverage reads would be checked for coverage again, and either ignored or analyzed for errors. Crucially, this second pass involves *at most* another full pass across the data, but only when the entire data set is below the coverage threshold; the larger the high coverage component of the data, the smaller the fraction of the data that is examined twice.

In Figure 4.8, we show diginorm-generated coverage saturation curves for both real and error-free simulated reads from *E. coli* MG1655. In both cases, after the first 1m reads, the majority of reads have an estimated coverage of 20 or higher, and hence can be used for error analysis on the remainder of the data encountered in the first pass.

The algorithm for the *semi-streaming* analysis of reads can be described as follows:

```
for read in data:  # first pass
   if estimated_coverage(read) < C:
      count_kmers(read, k-mer_database) #
      save(read)
   else:
      analyze(read)

for read in saved_reads:   # second pass
```

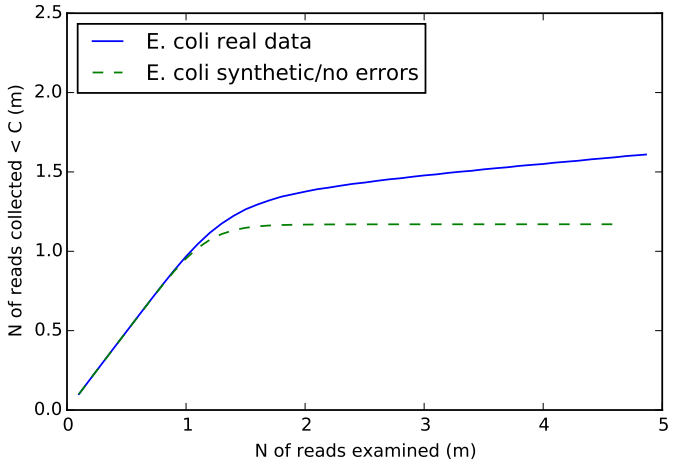

Figure 4.8: **Saturation curve of a real and a simulated *E. coli* read data set. Reads are collected when they have an estimated coverage of less than 20; in the early phase (< 1m reads), almost all reads are collected, but by 2m reads into the data set, the majority of reads come from loci with an estimated sequencing depth of > 20 and are rejected.**

```
if estimated_coverage(read) >= C:

   analyze(read)
```

As with digital normalization, a basic semi-streaming approach is very simple to implement: with an online way to count k-mers, the algorithm is approximately 10 lines of Python code. The approach also requires very few parameter choices: the only two parameters are k-mer size and target coverage C. However, we do not yet know how these parameters interact with read length, error rate, or data set coverage; systematic evaluation of parameters and the development of underlying theory is left for future work. In practice, we expect that additional work will need to be done to adapt existing error correction approaches to use the semi-streaming approach.

4.4.3 Semi-streaming error trimming on synthetic and real data:

We next adapted the error detection algorithm to do semi-streaming error trimming on synthetic or real genomic, metagenomic, and transcriptomic data.

On the synthetic "simple genome" this trimming approach eliminates 149 reads entirely and truncates another 392 reads. Of the 100,000 bp in the simulated reads, 31,910 (31.9%) were removed by the trimming process. In exchange, trimming eliminated *all* of the errors, bringing the overall error rate from 0.63% to 0.00%.

For the synthetic "simple metagenome" we only trimmed reads with estimated coverage of 20 or higher. Here, of 2347 reads containing 234,700 bp, 314 reads (13.4%) were removed and 851 reads (36.3%) were trimmed, discarding a total of 74,321 bases (31.7%). Of 1451 errors total, all but 61 were eliminated, bringing the overall per-base error rate from 0.62% to 0.04%. The simple mRNAseq data set showed similar improvement: 83 of 568 reads were removed, and 208 were trimmed, removing 19,507 of 56,800 bases (34.34%). The initial error rate was 0.65% and the final error rate was 0.07%.

Applying the semi-streaming error trimming to the *E. coli* MG1655 data set, we trimmed 2.0m reads and removed 50,281 reads entirely. Of 8.0m errors, all but 203,345 were removed, bringing the error rate from 1.49% to 0.07%. Trimming discarded 53 Mbp of the original 486 Mbp (11.1%).

On the mouse mRNAseq data set, semi-streaming error trimming removed 919,327 reads and trimmed 648,322 reads, removing 19.8% of the total bases, bringing the overall error rate from 1.59% to 1.21%. When we measured only the error rate in the high-coverage reads, trimming brought the error rate from 1.20% to 0.42%. On the mock metagenome data set, 27,554 reads were removed and 171,705 reads were trimmed, removing 0.36% of bases; this

Data set	pre-trim error	% bp trim	% reads trim	post-trim error
E. coli	1.49%	11.05%	41.9%	0.07%
mouse mRNAseq	1.59%	13.9%	19.8%	1.21%
(high coverage only)	1.20%	20.4%	29.0%	0.42%
Mock metagenome	0.31%	0.4%	1.1%	0.28%
(high coverage only)	0.16%	1.4%	3.5%	0.07%

Table 4.8: **A summary of trimming statistics for semi-streaming error trimming. Error rates before and after trimming were estimated by mapping. "High coverage" numbers refer to the subset of reads with $C \geq 20$ that were subject to analysis.**

Data set	mouse mRNAseq	mock metagenome
Total reads	81.3m	103.2m
Total bp	6.18 Gbp	10.4 Gbp
High-coverage reads	74.6m	91.9m
Number of passes	1.18	1.43
% reads trim	25.0%	11.75%
% bp trim	13.74%	4.03%
Pre-trim error rate	1.89%	0.27%
Post-trim error rate	1.30%	0.15%

Table 4.9: **Results of streaming error trimming on complete data sets. Error rates before and after trimming were estimated by mapping.**

low percentage is because of the very low coverage of most of the reads in this data set.

In practice, the space and time performance of both digital normalization and the generalized streaming approach presented here depend on specific details of the data set under analysis and the precise implementation of the coverage estimator. While our intention in this paper is to demonstrate the general streaming approach, we note that even our naive implementation for e.g. streaming trimming is useful and can be applied to very large data sets. For high coverage data, we can efficiently error-trim 10s of millions of reads in both sublinear memory and fewer than two passes across the data. In Table 4.9, we show the summary statistics for streaming error trimming of the full mouse mRNAseq and mock metagenome data; in contrast to the smaller subsets used previously (see Table 4.8), when we consider the full data sets the majority of reads are examined only once (see "Number of

passes", Table 4.9).

The implementation of semi-streaming error trimming used here is somewhat inefficient, and relies on redundantly storing all of the reads needed for the second pass on disk during the first pass. In the worst case, where all reads are low coverage, a complete copy of the data set may need to be stored on disk! This is an area for future improvement. However, when we look at full data sets, fewer than half the reads are examined twice (see Number of passes, Table 4.9).

4.4.4 Semi-streaming Illumina error rates and error profiles analysis

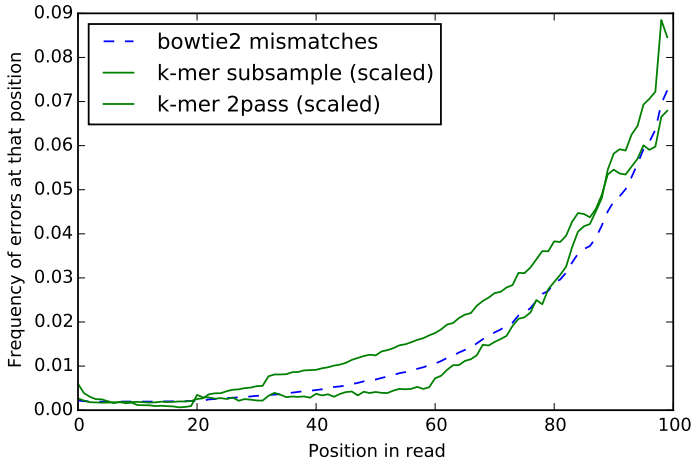

Figure 4.9: **Error spectrum of reads in the *E. coli* data set. The sublinear k-mer spectrum analysis is calculated based on saturation of a fraction of the data set, while the two-pass spectral analysis uses all of the data. bowtie2 mismatches are based on all mapped reads. The y values for the k-mer spectral analyses are scaled by a factor of four for ease of comparison.**

We can adapt the streaming approaches in the previous section to efficiently provide estimates for *subsets* of the data. The basic idea is to consume reads until some reads have saturated, and then to calculate error rates for new reads from the saturated loci in the

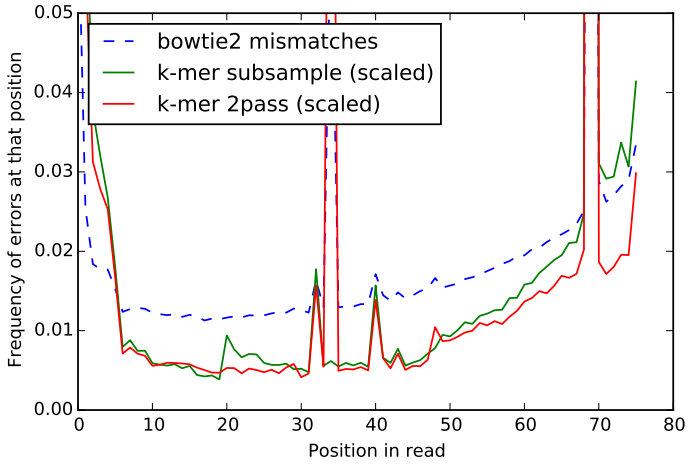

Figure 4.10: **Error spectrum of reads in the mouse RNAseq data set. The sublinear k-mer spectrum analysis is calculated based on saturation of a fraction of the data set, while the two-pass spectral analysis uses all of the data, and bowtie2 mismatches are based on all mapped reads. The peak of errors at position 34 in the bowtie2 mapping reflects errors that in the first part of the data set are called as Ns, and hence are ignored by the sublinear error analysis; see text for details. Note, the bowtie2 mismatch rates are larger than the spectral rates, so for ease of comparison the y values for the k-mer spectral analyses are scaled by a factor of four.**

graph. This can be done in one pass for data sets with sufficiently high coverage data: as shown above (Figure 4.8), in some data sets, most of the reads will have sufficient coverage to call errors by the time 20% of the data set has been consumed.

Using the same error detection code as above, we implemented a sublinear memory/sublinear time algorithm that collects reads until some regions have reached 20x coverage, or 200,000 reads have surpassed a coverage of 10x. In either case, all reads at or above a coverage of 10 are analyzed for errors, with a trusted k-mer cutoff of 3. In Figure 4.9 and Figure 4.10 we show the resulting error profiles for the *E. coli* and mouse RNAseq data sets, compared with the profile obtained by examining the locations of mismatches to the references. We also show the error profile obtained with the full two-pass approach (using digital normalization and then error detection as above) for comparison.

In the *E. coli* data set (Figure 4.9), we see the increase in error rate towards the 3' end of the gene that is characteristic of Illumina sequencing [31]. All three error profiles agree in shape (Pearson's correlation of 0.99 between each pair) although they are offset considerably in absolute magnitude. The k-mer error profile was calculated from the first 850,000 reads, but is consistent across five other subsets of the data chosen randomly with reservoir sampling (data not shown); all five subsets had Pearson's correlation coefficients greater than 0.99 with the bowtie2 mapping profile and the two-pass spectral approach.

The RNAseq error profile exhibits two large spikes, one at position 34 and one at position 69. Both spikes appear to be genuine and correlate with large numbers of Ns in those positions in the original data set. The spikes are present in the profiles derived from two-pass spectral analysis as well as the bowtie2 mismatch calculation. However, the sublinear approach does not detect them when using the first 675,000 reads. This is because of the choice of subsample: five other subsamples, chosen randomly from the entire data set with reservoir sampling, match the match the two-pass spectral analysis (data not shown). The error profiles calculated from all six subsamples with the sublinear algorithm have a Pearson's correlation coefficient greater than 0.96 with the error profiles from the full two-pass spectral approach and the bowtie2 mismatches.

The ability to analyze high-coverage reads without examining the entire data set offers some intriguing possibilities. One concrete application that we demonstrate here is the use of high coverage reads to infer data-set wide error characteristics for shotgun data, in a way that is robust to the sample type [61]. This approach could also be integrated directly into sequencers to assess whether the target coverage has been obtained, and perhaps stop sequencing. More generally, the approach of using saturating coverage to truncate computational analysis may have application to streaming sequencing technologies such as

SMRT and Nanopore sequencing, where realtime feedback between sequencing and sequence analysis could be useful [32, 53].

4.5 Time and space usage of the streaming algorithm for analyzing short DNA sequencing reads

As shown above, the essential idea of error analysis generally is that low-abundance k-mers contained in a high-coverage data set typically represent random sequencing errors. We address this problem by making use of k-mer spectra, a common approach in which reads are treated as subpaths through a De Bruijn graph, and errors in the reads are identified by finding low-frequency subpaths [99].

We generalize this approach by building the graph with an online algorithm and detecting regions of the graph saturated by observations. These regions can then be used for per-read analysis without necessarily examining the entire data set.

Detecting graph saturation: We detect graph saturation with digital normalization. The digital normalization algorithm is, in Python pseudocode:

```
for read in data:
   if coverage(read, table) < DESIRED:
      add_read_to_graph(read, graph)
      analyze(read)
```

This is a single-pass algorithm that can be implemented in fixed space using a Count-Min Sketch to store the De Bruijn graph necessary for coverage estimation [96, 148]. For any error-containing data set with coverage greater than `DESIRED`, the graph requires space less

than the size of the input - typically space sublinear in the data size, for any fixed-size source text (see Figure 4.8 and [148]).

The digital normalization algorithm was developed as a *filter*, in which the reads are passed on to another program (such as a *de novo* assembler) for further analysis – these later analyses are typically based on multi-pass, heavyweight algorithms. Here, digital normalization is performing lossy compression, reducing the number of error-containing sentences while attempting to retain the structure of the De Bruijn graph [10, 148, 71]. This reliance on a post-normalization heavyweight analysis step limits the applicability of digital normalization and presents challenges in the analysis of extremely large data sets, which motivated this work.

Semi-streaming analysis: The algorithm for *semi-streaming* analysis of reads is as follows:

```
for read in data:  # first pass
   if coverage(read, graph) < DESIRED:
      add_read_to_graph(read, graph)
      save(read)
   else:
      analyze(read)


for read in saved_reads:   # second pass
   if coverage(read, graph) >= DESIRED:
      analyze(read)
```

Here, the space used for the graph remains identical to the digital normalization algorithm and is typically sublinear in space for high coverage data sets, but the algorithm is no longer single-pass, and requires re-examining some subset of the input data in a second pass. In the worst case scenario, with an undersampled source text (or randomly generated sentences), this is a fully offline two-pass approach that requires re-examining *all* of the input data for the second pass. In practice, most real data sets will require fewer than two passes: graphically, any deviation from the identity line in a saturation analysis as in Figure 4.8 yields a few-pass algorithm.

4.6 Conclusion

Shotgun DNA sequencing generates data as a stream of items representing sentences ("reads") randomly sampled from a larger text, with replacement. There are several distinct features of this kind of stream-like data. The first is that important details of the source text, such as its size and statistical composition, may be completely unknown; that is, often the reads themselves are the most specific information we have about the source text. Second, the source text may be incompletely sampled by the reads, and whether or not it is completely sampled may not be known in advance. And third, read data sets are typically stored on disk, at least in current implementations; our goal is to identify more efficient approaches to examining these data sets without necessarily moving to a pure streaming model, which allows us to make use of the *semi-streaming* paradigm introduced by Feigenbaum et al. [33].

In this chapter, we discussed our solutions to two primary problems. One is to efficiently distill the non-redundant informaiton from the streaming to reduce the size of data finally without losing much important information. The other one is to efficiently identify the

locations of errors in these reads by finding differences with respect to the (unknown) source text. Both solutions are based on a novel approach to use median k-mer count in a read to estimate sequencing depth without a reference assembly, which will also be the foundation of the IGS based diversity analysis method discussed in the next chapter.

Streaming represents the future of big data. This kind of problems we are dealing with is not only critically important to a better understanding of the exploding big biological data, but also a gateway to a larger set of interesting domain problems dealing with big data, like estimating the true abundance of the sentences in the larger text or detecting evil traffic in the Internet data stream.

4.7 Data

4.7.1 Code availability

The algorithms of digital normalization and error analysis are implemented in the khmer software package, written in C++ and Python, available at github.com/ged-lab/khmer/. khmer also relies on the screed package for loading sequences, available at github.com/ged-lab/screed/. khmer and screed are Copyright (c) 2010 Michigan State University, and are free software available for distribution, modification, and redistribution under the BSD license.

The code and detailed instruction used to generate all of the results in this chapter is available at github.com/ged-lab/2012-paper-diginorm/ and http://github.com/ged-lab/2014-streaming/.

4.7.2 Data sets used for digital normalization

The *E. coli*, *S. aureus*, and *Deltaproteobacteria* data sets were taken from Chitsaz et al. [19], and downloaded from bix.ucsd.edu/projects/singlecell/. The mouse data set was published by Grabherr et al. [41] and downloaded from trinityrnaseq.sf.net/. All data sets were used without modification. The complete assemblies, both pre- and post-normalization, for the *E. coli*, *S. aureus*, the uncultured *Deltaproteobacteria*, mouse, and yeast data sets are available from ged.msu.edu/papers/2012-diginorm/.

The simulated genome and transcriptome were generated from a uniform AT/CG distribution. The genome consisted of a single chromosome 400,000 bases in length, while the transcriptome consisted of 100 transcripts of length 500. 100-base reads were generated uniformly from the genome to an estimated coverage of 200x, with a random 1% per-base error. For the transcriptome, 1 million reads of length 100 were generated from the transcriptome at relative expression levels of 10, 100, and 1000, with transcripts assigned randomly with equal probability to each expression group; these reads also had a 1% per-base error.

4.7.3 Synthetic data sets used for error analysis

We computationally constructed three small short-read DNA data sets for initial exploration of ideas. All synthetic sequences have equiprobable A/C/G/T. All synthetic reads are 100bp long and were sampled with 1% error. The "simple genome" data set consists of 1000 reads chosen uniformly from a 1 kb randomly constructed genome. The "simple transcriptome" data set consists of 568 reads chosen uniformly from synthetic transcripts containing different subsets of four 250-base exons, with expression levels varying by a factor of 30 from minimum to maximum. The "simple metagenome" data set consists of reads sampled from three

Name	Number of reads	Description
simple genome	1000	1kb genome; no repeats
E. coli MG1655	4,863,836	Subset of ERA000206 ([20])
simple transcriptome	568	300:1 high:low abundance; shared exons
mouse mRNAseq	7,915,339	Subset of GSE29209 ([42])
simple metagenome	2,347	316:1 high:low abundance species
mock metagenome	18,805,251	Subset of SRR606249 ([120])

Table 4.10: **Data sets used for evaluation.**

different 500 bp sequences, across 30 fold variation in abundance. In all three cases, the errors during read sampling were recorded for comparison with predictions.

4.7.4 Real data sets used for error analysis

We used three shotgun Illumina data sets: a genomic data set from *E. coli*, a mRNAseq data set from *Mus musculus*, and a mock community metagenome. For *E. coli*, we took a 5m read subset of ERA000206 from [20]. For mRNAseq, we used a 10m read subset of GSE29209 from [42]. For the mock metagenome, we used a 20m read subset of SRR606249 from [120]. Prior to analysis, we eliminated any read with an 'N' in it and filtered the reads by mapping to the known references, yielding the read numbers in Table 4.10.

Chapter 5

A framework for diversity analysis of whole shotgun metagenomic reads data

5.1 Introduction

Here we propose a novel concept, "informative genomic segment" or IGS, and use IGS as a replacement of OTUs as the basic unit for diversity analysis of whole shotgun metagenome data sets. IGSs represent the unique information in a metagenomics data set and the abundance of IGSs in different samples can be retrieved by analyzing read coverage with an efficient k-mer counting method (discussed in the previous two chapters). The samples-by-IGS abundance data matrix is a promising replacement of samples-by-OTU data matrix used in 16S rRNA based analysis and many existing statistical methods can be applied to work on the samples-by-IGS data matrix to investigate diversity. We applied this method to several simulated data sets and several real metagenomic data sets, including human microbiome, sea water and soil data. The results of beta diversity analysis showed that the samples were clustered with comparable or better accuracy than existing alignment-based method. The results of alpha diversity analysis showed this was a promising new approach to estimate

metagenome sizes. Since this method is totally binning-free, assembly-free, annotation-free, and reference-free, it is promising for dealing with highly diverse environmental samples, where we are facing many unknowns.

5.2 The concept of IGS(informational genomic segment)

In classic ecology dealing with macroorganisms, diversity measurement is based on the concept of species. For 16S rRNA amplicon metagenomics data set, it is based on the concept of OTUs. While the concept of OTUs can be used to analyze large shotgun metagenomics data set, normally assembly, binning and annotation are required before doing diversity analysis. However for many metagenomics projects these are difficult tasks, lacking necessary reference genome or requiring expensive computation. So we are interested in finding an approach to bypass difficult tasks like assembly, binning, annotation and use the raw reads to make the diversity analysis of large shotgun whole genome metagenomic data possible.

We began such efforts by proposing that the concept of k-mer (a DNA segment with the length of k) could be used as the basic unit to measure the diversity. K-mers can be considered as the atom of information in DNA sequences. One of the composition-based approaches to binning is to use the k-mers as the source of signatures[3, 57]. Suppose the sizes of microbial genomes are similar and the difference between genomic content of microbial genomes is similar, the number of distinct k-mers in the sequence data set correlates to the number of species in a sample. However, because of sequencing error, which is unavoidable due to the limit of sequencing technology, this k-mer based analysis does not work well on unassembled data. One sequencing error on a read will generate up to k erroneous k-mers. In metagenomics data sets, especially with high coverage, most of the distinct observed k-mers

are from sequencing errors.

Next we shifted the focus from k-mers onto a higher level - reads. In previous chapter, we have discussed a novel approach to use median k-mer count in a read to estimate sequencing depth without a reference assembly, based on which the framework for streaming analysis of short DNA sequencing reads was developed. It also offers a novel way to distill information from reads by reducing the bad effect of sequencing errors so that we can use those informative reads to measure the microbial diversity. We term those informative reads as IGSs (informative genomic segments), which can be considered segments of DNA on a microbial genome. Those IGSs should be different enough to represent the abstract information a genome contains. Suppose microbial genomes contain similar number of those IGSs, as they contain similar number of distinct k-mers; then the number of IGSs will correlate with the species richness in a sample, and the abundance distribution of IGSs will be related to species evenness in a sample. Furthermore, we can get the abundance of the IGSs across different samples. Many classic diversity estimation methods based on OTUs described in the literature review chapter can be applied to estimate the diversity of IGSs and the diversity of actual species subsequently.

For alpha diversity, we can generate a list of IGSs and the respective abundance in a sample. Then existing estimators like Chao's can be applied to estimate the total number of IGSs in the sample. Rarefaction curves based on the number of IGSs can also be generated.

For beta diversity, we can generate a samples-by-IGS data matrix from the abundance of IGSs across samples, as a replacement of samples-by-OTU data matrix in OTU-based analysis and samples-by-species data matrix in traditional ecology. From that samples-by-IGS data matrix, we can use existing methods to calculate similarity/dissimilarity/distance between samples and do further analysis like clustering and ordination.

5.2.1 IGS(informative genomic segment) can represent the novel information of a genome

Median k-mer abundance can represent sequencing depth of a read, as discussed in last chapter (published as [10]). For a sequencing reads data set with multiple species, the sequencing depth of a read is related to the abundance of species where the read originates from.

The upper plot in Figure 5.1 shows the abundance distribution of reads from 4 simulated sequencing data sets with different sequencing depths - 3 sequencing data sets generated with different sequencing coverage(1x, 10x, 40x) from 3 simulated random genomes respectively and 1 combined data set with all the aforementioned data sets. No error is introduced in these simulated data sets. Obviously the reads from the three data sets can be separated by estimated sequencing depth. The combined data set can be considered as a sequencing data set with three species with different abundance.

Each point on the curve shows that there are Y reads with a sequencing depth of X. In other word, for each of those Y reads, there are $X-1$ other reads that cover the same DNA segment in a genome that single read originates. So we can estimate that there are Y/X distinct DNA segments with reads coverage as X. We term these distinct DNA segments in species genome as IGSs (informative genomic segments). We can transform the upper plot in Figure 5.1 to show the number of IGSs and their respective reads coverage, as shown in lower plot. We sum up the numbers of IGSs with different reads coverage for each data set and get the result as shown in Table 5.1. The sum numbers of IGSs here essentially are the areas below each curve in the figure.

Even though the datasets have different sequencing depth like 10X and 40X, they have

similar numbers of IGSs. Dataset with 1X sequencing depth has fewer IGSs because the depth is not enough to cover all the content of the genome(63.2%). The IGSs can be seen as the genomic segments on a genome with the length of reads.(Figure 5.2 Assume the composition of species genome is totally random, which is the case in the simulated data sets, the number of IGSs (N) in a species genome is related to the size of genome(G), read length(L) and k size(k), which can be denoted as

$$N = \frac{G}{L}$$

which is the number of reads that can have a 1X coverage of the genome. For the simulated genome with size of 1M bps, read length as 80bps, expected number of IGSs is

$$1000000/80 = 12500$$

,

which is close to the observed value (Table 5.1).

Data set	total number of IGSs
1X depth	6419
10X depth	12022
40X depth	12371
1X,10X,40X combined	30748

Table 5.1: **Total number of IGSs in different simulated reads data sets.**

5.2.2 Using IGS to analyze alpha diversity

Basically the abundance distribution of IGSs with different coverage in a sample data set can be acquired using the method shown above.

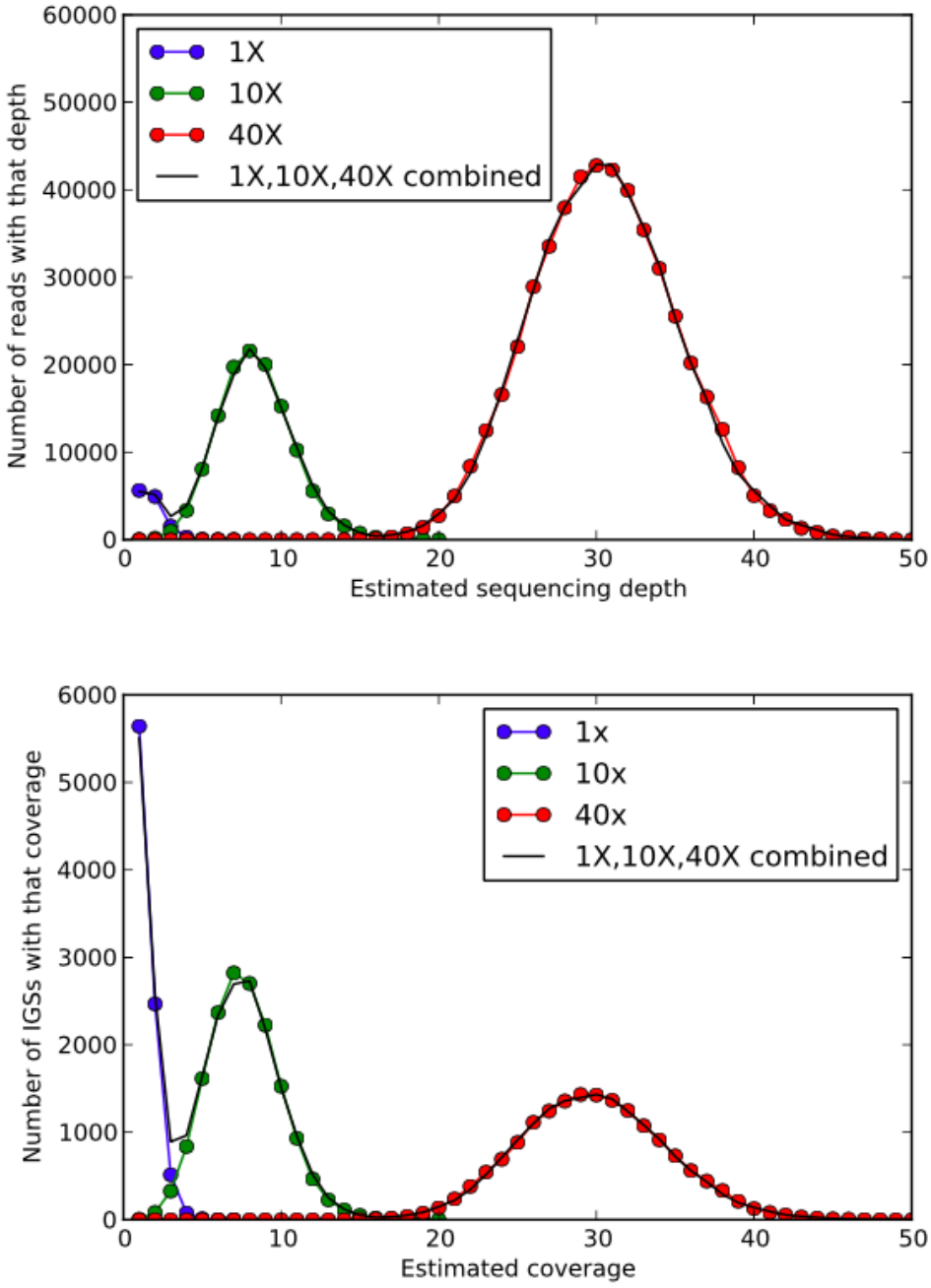

Figure 5.1: **Transforming reads into IGSs.**

Suppose from a reads data set, the coverage distribution of reads is as shown in this table:

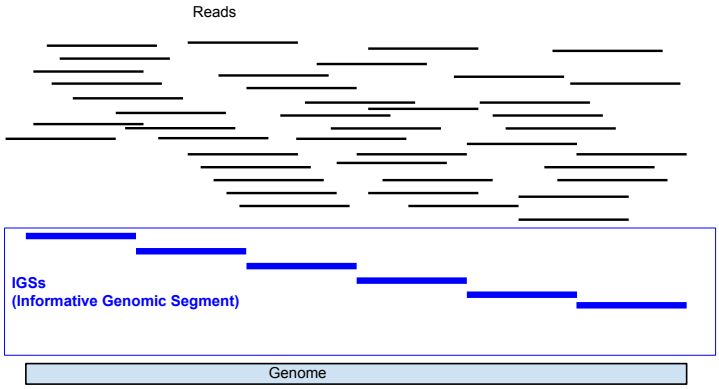

Figure 5.2: **IGS(informative genomic segment) can represent the novel information of a genome.**

coverage	number of reads
3	69
4	96
5	125
6	150
...	...

There are 69 reads with coverage as 3, 96 reads with coverage as 4, and so on. We transform this coverage distribution of reads into abundance distribution of IGSs, as shown in this table:

abundance	number of IGS
3	23
4	24
5	25
6	25
...	...

For example, there are 23 IGSs with abundance 3. This is calculated by dividing total

number of reads with coverage as 3, which is 69, by the coverage 3. Similarly there are 24 IGSs with abundance 4. If we draw an analogy between IGSs and OTUs, this is like saying there are 23 different OTUs with 3 reads mapped to each OTU, and 24 different OTUs with 4 reads mapped to each OTU, and so on.

Next all the different IGSs and the corresponding abundances can be listed, as shown here:

IGS ID	abundance
1	3
1	3
1	3
...	...
23	3
24	4
25	4
...	...
47	4
48	5
...	...

This list is the counterpart of an OTU table in OTU based diversity analysis. With such table at hand, existing statistical methods and software packages can be directly used to investigate the alpha diversity.

5.2.3 Using IGS to analyze beta diversity

As with alpha diversity analysis, the OTU table is also a foundation for beta diversity analysis. As long as we get a reliable OTU table, there are existing pipelines to perform the beta diversity analysis.

A typical OTU table across different samples is like this, which is also called a samples-by-OTU data matrix:

OTU ID	Sample A	Sample B	Sample C
OTU1	3	4	2
OTU2	2	5	0
OUT3	3	1	4
...	...	...	...

Like a OTU table, we hope to have the IGS table for the IGSs:

IGS ID	Sample A	Sample B	Sample C
IGS1	2	3	2
IGS2	2	4	0
IGS3	3	3	4
...	...	...	...

So now the problem is how we can generate a samples-by-IGS matrix as the counterpart of samples-by-OTU matrix so many of the existing tools/methods used for OTU-based diversity analysis can be borrowed for this kind of IGS-based analysis, just as what is shown above for alpha diversity analysis.

Firstly, using the same approach to get the coverage of a read in the sample data set where it is from (Figure 5.3-A), we can get the coverage of a read from sample A dataset

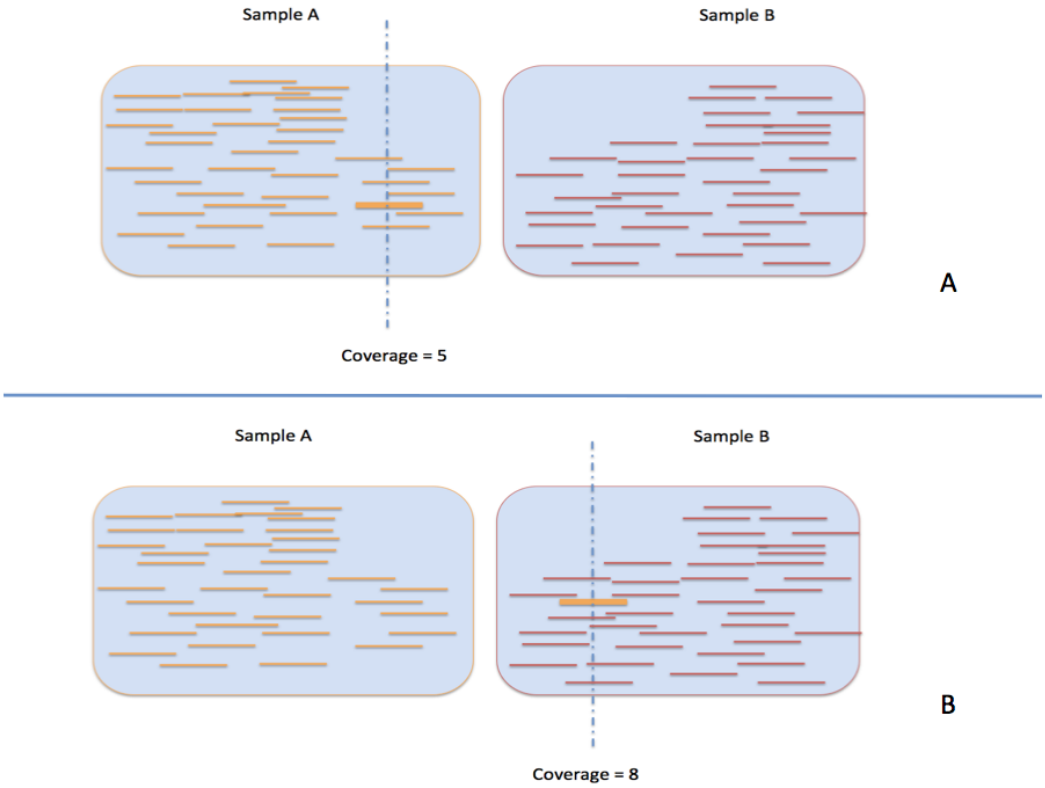

Figure 5.3: **Get the coverage of a read in samples. A read in sample A has the coverage of 5 in sample A, has the coverage of 8 in sample B.**

in sample B dataset (Figure 5.3-B). We still use the median k-mer count to represent the coverage of a read.

For a data set with several samples to analyze, firstly we can get the coverage of a read across different samples and get a read coverage profile like "4:6:2", as shown in Figure 5.4(A). For all the reads in the samples we can get such read coverage profiles, as shown in Figure 5.4(B). We have already known that "contigs with similar coverage profiles are likely to have originated from the same microbial population"[57]. Several new binning methods based on coverage profiles have been developed based on such assumption[2, 59, 3, 94, 57]. Thus, here we can assume that reads with similar coverage profiles are likely to have originated from the same genomic region. Actually it is safer to say reads with different coverage profiles

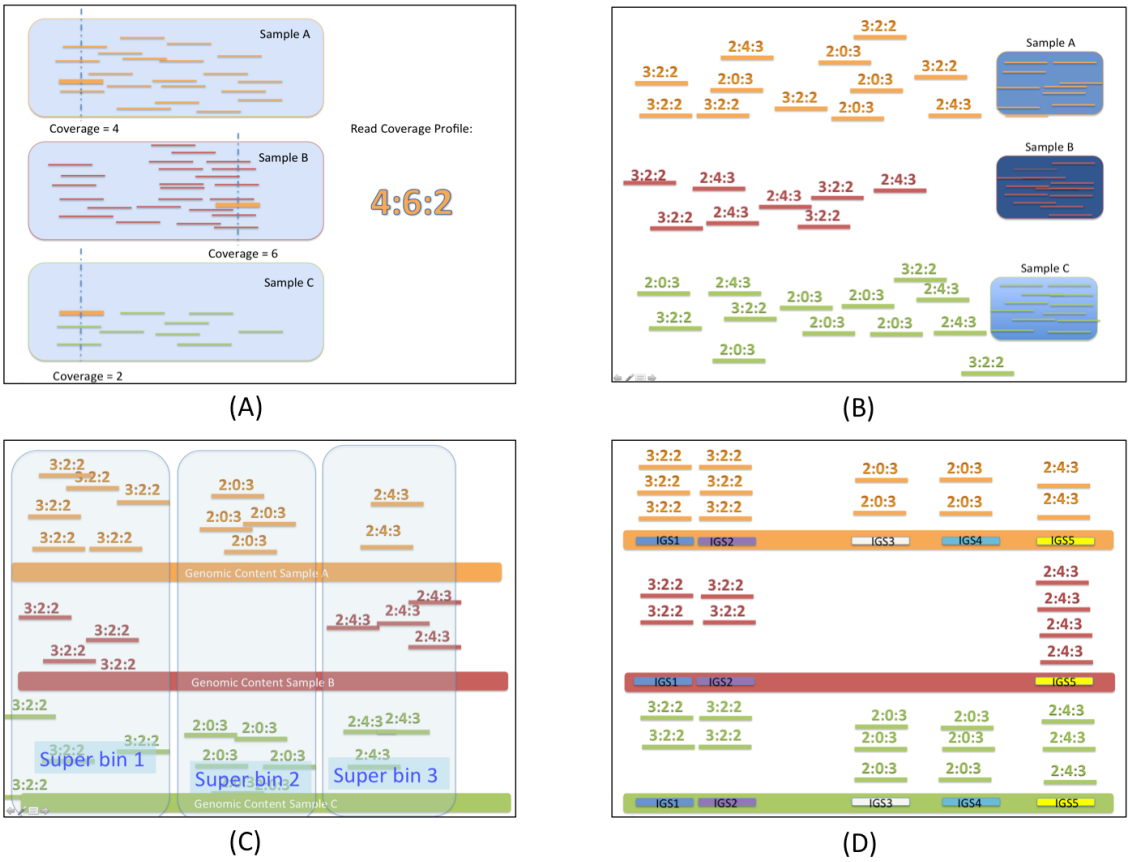

Figure 5.4: **From read coverage profile to IGS. (A): Get the coverage profile of one read. (B): Get the coverage profiles of all the reads in 3 samples. (C): Group the reads with same coverage profiles into "super bin". (D): Calculate the number of IGSs in each "super bin".**

are not likely to have originated from the same genomic region. The next step is to group those reads with same coverage profiles together into "super bin" in different samples, as shown in Figure 5.4(C). The reads in each "super bin" may not be from the same species, however, they should be from the species that have same abundance profile across samples. In the example shown in the figure, the 6 reads from sample A, the 4 reads from sample B and the 4 reads from sample C all have the same coverage profile as "3:2:2". Actually, the numbers of reads from different samples with same coverage profile have similar ratio to the numbers in the profile, like "6:4:4" versus "3:2:2" in the example above. (Experiments using simulated data are not shown here.)

Next we can use the approach similar to the one used for alpha diversity analysis to estimate the size of the genomic region each "super bin" covers, represented by the number of IGSs. Still for the example in Figure 5.4(C), 6 reads from sample A grouped into the first "super bin" have a coverage of 3 in sample A, where they originate from. The number of IGSs those 6 reads cover can be calculated as 6/3, which is 2. Thus 2 IGSs have an abundance profile as "3:2:2" across the samples. Similarly there are 2 IGSs with abundance profile as "2:0:3" and 1 IGS with abundance profile as "2:4:3", as shown in Figure 5.4(D).

List those IGSs and the corresponding abundance profiles across samples, we can have the samples-by-IGS matrix like this:

IGS	Sample A	Sample B	Sample C
IGS1	3	2	2
IGS2	3	2	2
IGS3	2	0	3
IGS4	2	0	3
IGS5	2	4	3

With such samples-by-IGS matrix, similarity matrix between samples can be calculated using different similarity indices, like Bray-Curtis. Next clustering and ordination methods can be applied to better interpret the relationship between samples.

5.3 Evaluating IGS method using simulated data sets

5.3.1 Using a simple simulated data set to evaluate the IGS method

For this experiment, firstly we create 6 synthetic samples (Sample 1-6) based on 9 synthetic 100K genomes (genome A-I), with different composition of species and diversity (Table 5.2).

For sample1, there are two species - A and B, with abundance distribution as 3:1. The sequencing depth of all the synthetic data sets is 10X. As a simple experiment to demonstrate the effectiveness of the IGS based method, there is no sequencing error introduced in the synthetic reads data sets.

sample ID	species composition	sequencing depth	abundance of species	size of metagenome (bp)
sample1	AAAB	10	A:30 B:10	200K
sample2	AABC	10	A:20 B:10 C:10	300K
sample3	ABCD	10	A:10 B:10 C:10 D:10	400K
sample4	ABCE	10	A:10 B:10 C:10 E:10	400K
sample5	AFGH	10	A:10 F:10 G:10 H:10	400K
sample6	IFGH	10	I:10 F:10 G:10 H:10	400K

Table 5.2: **6 synthetic simple metagenomes**

To evaluate the effectiveness of alpha diversity analysis using IGS based method, we can use a metric to estimate the total number of IGSs in a sample, which can be used to calculate the estimated genome size of a sample using the formula below: size of genome = number of IGS $*$ reads_length

In this experiment, we use ACE metric since we find it is more accurate than Chao1, since it uses more abundance information.

	observed IGS	ACE	simpson evenness	estimated genome size (Kbp)	real genome size (Kbp)
sample1	2002	2002.0	0.76	200.2	200
sample2	3038	3038.0	0.83	303.8	300
sample3	4076	4076.0	0.91	407.6	400
sample4	4078	4078.0	0.91	407.8	400
sample5	4069	4069.0	0.91	406.9	400
sample6	4087	4087.0	0.91	408.7	400

Table 5.3: **Alpha diversity analysis result of the simple simulated data using IGS method.**

Table 5.3 shows the alpha diversity analysis result of the simple simulated data using IGS method. The estimated genome sizes of the samples are close to real size. This is not

surprising since for this simple experiment, there is no error introduced and the coverage is high (10x) to cover most of the genetic materials in the samples. Also the Simpson evenness shows the relative evenness of the samples correctly. Sample 1 is the least even with composed of two species with abundance ratio as 1:3. This shows that in this simple example, the IGS method cannot only analyze the richness of samples but also the evenness.

To evaluate the effectiveness of beta diversity analysis using IGS based method, we compared the dissimilarity matrix generated by IGS based method with the true matrix, since we know exactly the species composition of the simulated data set.

	sample 1	sample2	sample 3	sample 4	sample 5	sample 6
sample 1	0.00	0.25	0.50	0.50	0.75	1.00
sample 2	0.25	0.00	0.25	0.25	0.75	1.00
sample 3	0.50	0.25	0.00	0.25	0.75	1.00
sample 4	0.50	0.25	0.25	0.00	0.75	1.00
sample 5	0.75	0.75	0.75	0.75	0.00	0.25
sample 6	1.00	1.00	1.00	1.00	0.25	0.00

Table 5.4: **Dissimilarity matrix between synthetic samples using Bray-curtis from species composition directly.**

	sample 1	sample2	sample 3	sample 4	sample 5	sample 6
sample 1	0.00	0.35	0.60	0.66	0.80	1.00
sample 2	0.35	0.00	0.42	0.51	0.84	1.00
sample 3	0.60	0.42	0.00	0.56	0.89	1.00
sample 4	0.66	0.51	0.56	0.00	0.89	1.00
sample 5	0.80	0.84	0.89	0.89	0.00	0.42
sample 6	1.00	1.00	1.00	1.00	0.25	0.00

Table 5.5: **Dissimilarity matrix between synthetic samples using Bray-Curtis from sequencing reads using IGS method.**

The true dissimilarity matrix of the 6 simulated samples using Bray-Curtis metric from species composition directly is shown in Table 5.4. For a simulated data set with 10x coverage and no error introduced (which again will tell us the optimal performance of IGS method), the dissimilarity matrix can be calculated by using the IGS method, as shown in Table

5.5. We can see the absolute values in the matrix are not very close to those in the real matrix. However, the relative values correspond to those in the real matrix well enough to show the relative distance between each pair of samples. To get a objective metric, we use the Mantel [77] test to calculate the correlation value between the two matrixes. The correlation is 0.9714, which means a very strong correlation between the two matrices. Thus the dissimilarity matrix from the IGS method reflects the true relationship between samples effectively.

If the matrix can reflect the real relationship between samples reliably, the clustering and ordination will only be routine tasks.

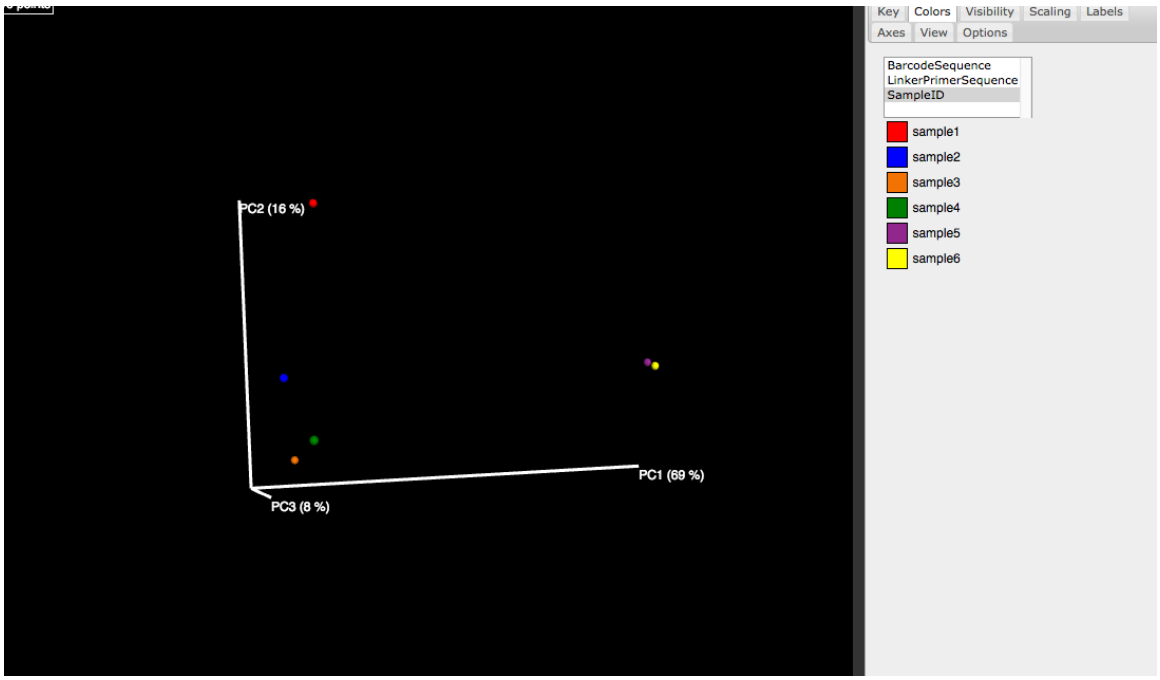

Figure 5.5: **Ordination of the 6 synthetic samples using IGS method.**

Figure 5.5 and Figure 5.6 show that IGS method can yield similarity between samples correctly. Sample 5 and sample 6 are very close to each other on the figure, which matches their species composition.

The clustering and ordination are all from the dissimilarity matrix. We think comparing

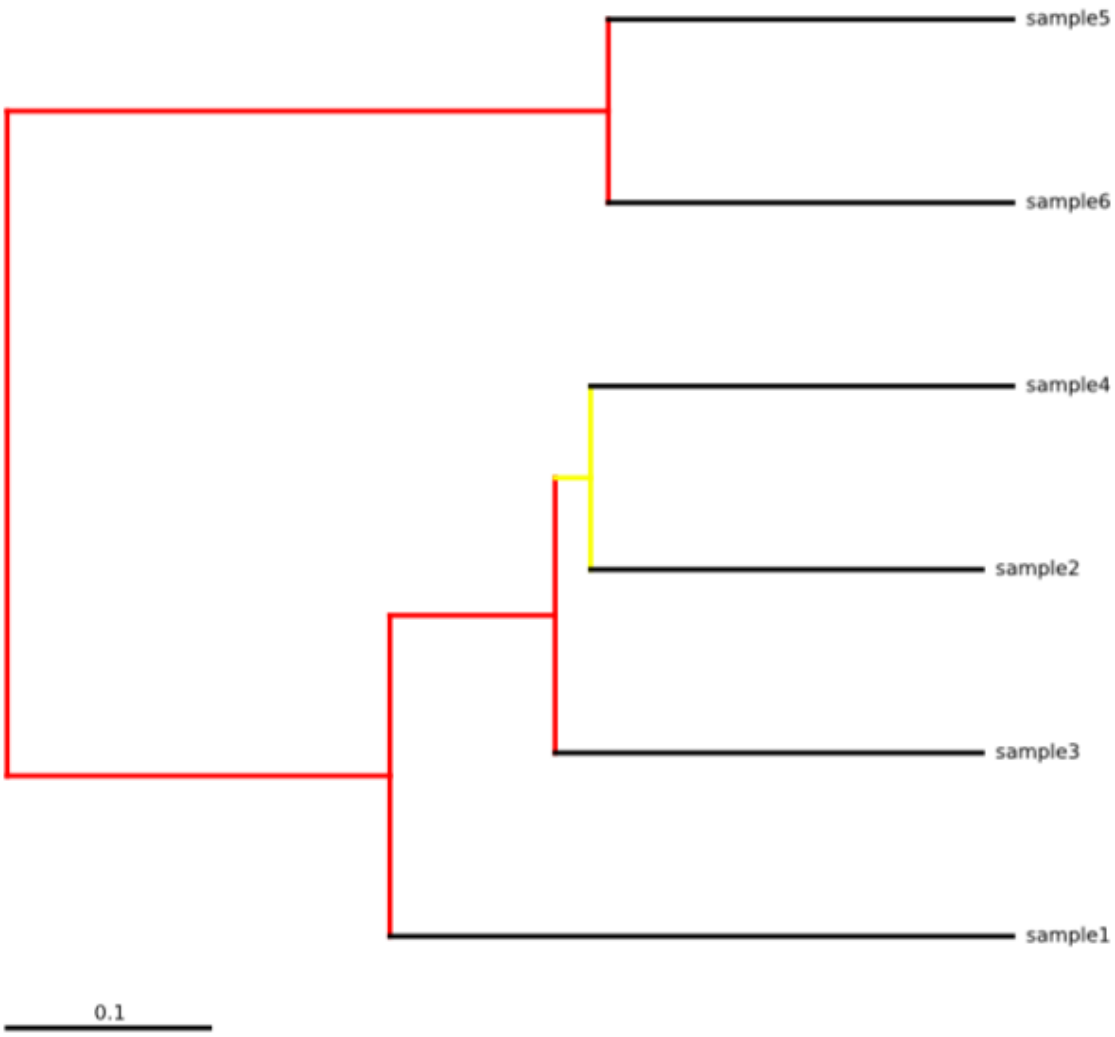

Figure 5.6: **Clustering of the 6 synthetic samples using IGS method**

matrices directly makes more sense than comparing the clustering and ordination plots. So we will not show the clustering and ordination figure in other evaluations in this section. Mantel correlation will be used to measure the accuracy of beta diversity analysis.

These results show that the IGS method can work well on a simple scenario, with high sequencing depth (10X) and no sequencing error. Next we will check the influence on the analysis accuracy of variable sequencing depth and sequencing error, and introduce new ways

to preprocess the data to decrease the influence of sequencing error.

5.3.2 Improving the accuracy of this method in real world analysis

Previously we have shown the IGS method generally works on a simple simulated data set, with high sequencing depth and no sequencing error. In the real world, in many situations we have to deal with metagenomic data sets with relatively low sequencing depth, like soil or sea water samples. Also it is a fact that all sequencing technology generates some errors. As discussed in the introduction chapter, one of the reasons we developed the IGS method is that we expect the IGS method to be less prone to sequencing error based on the abundance counting of reads rather than k-mers. However the effect of those factors on the accuracy should still be observable.

In this section, we will analyze the effect of these factors on the accuracy of the IGS method and investigate ways we can reduce the effect in order to increase the accuracy of analysis.

As in last section, six synthetic samples were generated with the same species composition with same coverage as 10X but with different sequencing error rate (0.5%, 1.0%, 1.5%, and 0% - no error at all).

To show the influence of sequencing error on accuracy of the analysis, we compared the richness estimation using reads with different sequencing error rate, as shown in Figure 5.7. For data set without error (error rate = 0), the estimated size of the metagenome matches the real size perfectly. With increasing error rate, the size of metagenome is increasingly over-estimated. This is due to several factors,which will be discussed below.

We also check the beta diversity analysis with different error rate and notice that the beta diversity is less prone to increasing sequencing error rate 5.8. We will therefore focus

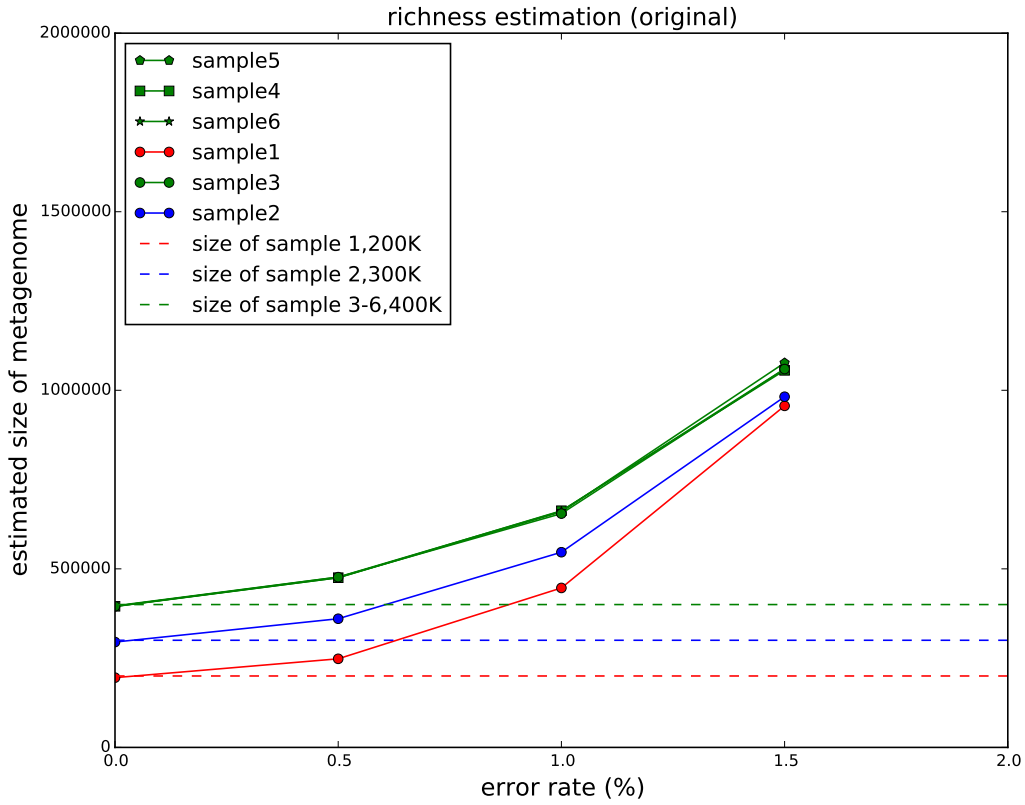

Figure 5.7: **Richness estimation using IGS method without adjustment.**

on alpha diversity in the discussion below.

5.3.2.1 the effect of sequencing error to the accuracy of analysis

The first factor to take into account is sequencing error. One sequencing error will generate up to k erroneous k-mers. This is the reason why it is difficult to use k-mer counting only to do diversity analysis, as a large proportion of k-mers in a reads data set are erroneous, especially for low coverage reads data. As discussed in the section about digital normalization, using median k-mer count to retrieve the coverage of a read is less prone to sequencing error, because this will not always affect the median k-mer count.

Take the experiment we did previously as an example, for read length of 100bp and k

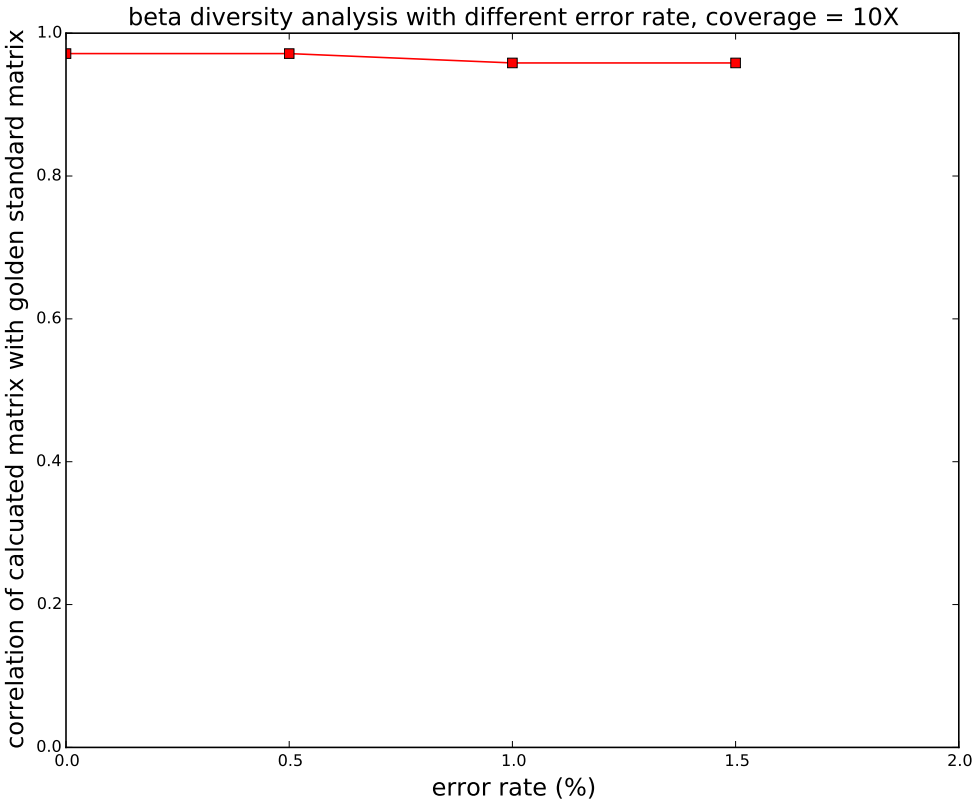

Figure 5.8: **Beta diversity analysis using IGS method without adjustment.**

as 19, one sequencing error will affect the count of 19 k-mers at most, and two sequencing errors will affect the count of at most 38 k-mers. The count of these k-mers will generally be much lower than the true count. So out of the 82 k-mers in the 100bp read, at most 38 k-mers will have incorrectly low counts. However, this will not affect the median k-mer count, which is the count of the 41th k-mer if ranked by count. However, if there are three or more errors in the read, the situation is more complicated. For 3 errors in a read, 3 to 57 k-mers will be affected by the errors to have an incorrect count as 1. The distribution of the probability about the number of affected k-mers can be acquired by a model similar to Lander-Waterman model used in genome sequencing theory. Here we got the distribution

using simulation, as shown in Figure 5.9. From this probability distribution, we can get the probability that 3 errors will affect more than 40 k-mers is 0.43. In this case, 3 errors will affect the median k-mer count of a read. We can also get such probability for 4 errors or more. Combining to the probability that a certain number of errors occur in a read with a specific sequencing error rate, which is easy to derive from binomial distribution, we can get the probability that the coverage of a read is incorrectly assessed as 1. Still for the example here, this probability is the probability that 3 errors occur in a read multiplied by the probability that 3 errors will affect median k-mer count, plus the probability that 4 errors occur in a read multiplied by the probability that 4 errors will affect median k-mer count,and so on.

Generally, let $P\_error(n, e, L)$ is the probability that n errors occur in a read with length as L, with error rate as e and $P\_effect(n, k, L)$ is the probability that n errors in a read with length of L affect median k-mer count. The probability that the coverage of a read is incorrectly assessed as 1 is

$$\sum_{n=3}^{\infty} P\_error(n, e, L) \times P\_effect(n, k, L)$$

, and by binomial distribution,

$$P\_error(n, e, L) = f(n; L, e) = Pr(X = n) = \binom{L}{n} e^n (1 - e)^{L-n}$$

Practically, when $n > 5$ and $e < 0.015$, $P\_error(n, e, L)$ is very small, we only consider number of errors in a read as 3, 4 and 5.

From the discussion above, the sequencing errors reduces the estimated coverage of some reads incorrectly to 1 and the probability this occurs to a read can be estimated. So to

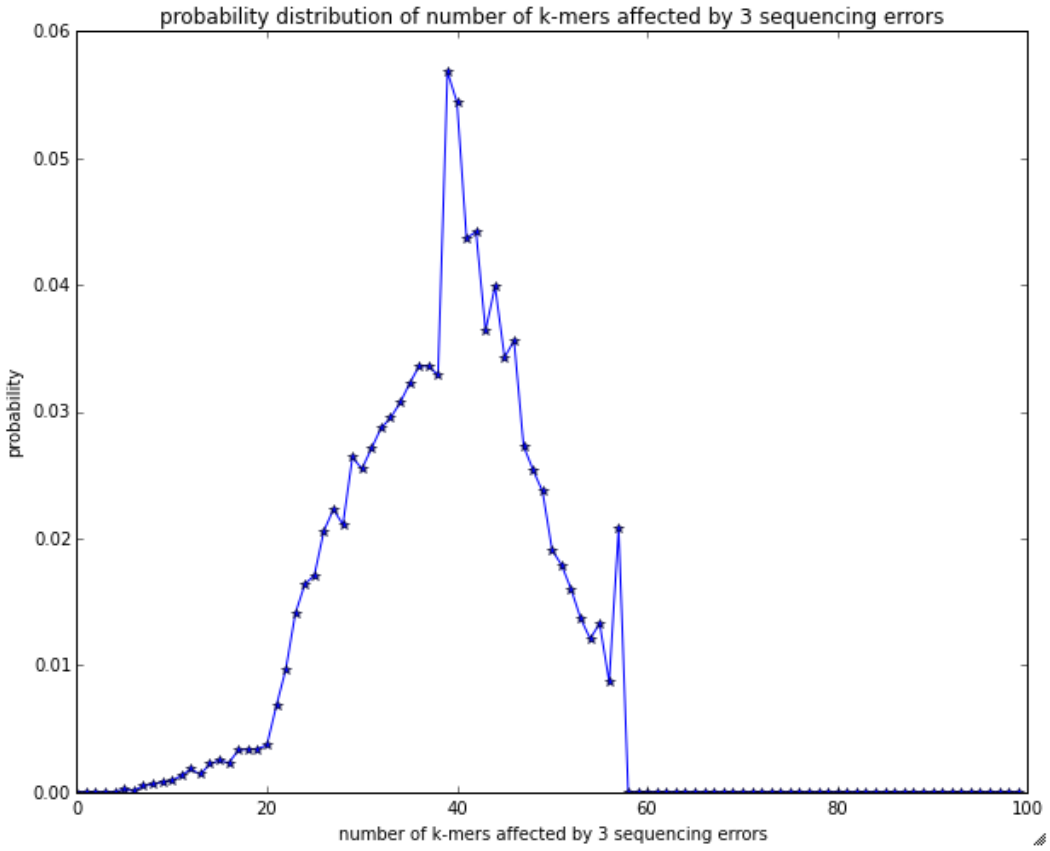

Figure 5.9: **Richness estimation using IGS method without adjustment.**

reduce the effect of sequencing error on this aspect, we can calculate the expected number of reads that are affected and remove those reads from the set of reads with coverage of 1 before generating list of IGS from the reads abundance distribution.

Also, we want to make sure 2 errors in a read will not affect median k-mer count, since it is more common to have 2 errors in a read practically. In this case,

$$2 \times k < \lfloor \frac{L-k+1}{2} \rfloor,$$

we can get $k < L/5$,basically. For L as 100, k will be 19, which is what we choose in the testing. However, the k should not be too small, or the k-mers cannot handle the diversity of information of a large data set.

Taking the sequencing error into account, we used the methods introduced above to adjust the estimation of metagenome size of the 6 synthetic samples. The estimation after adjustment is closer to real number, as shown in Figure 5.10..

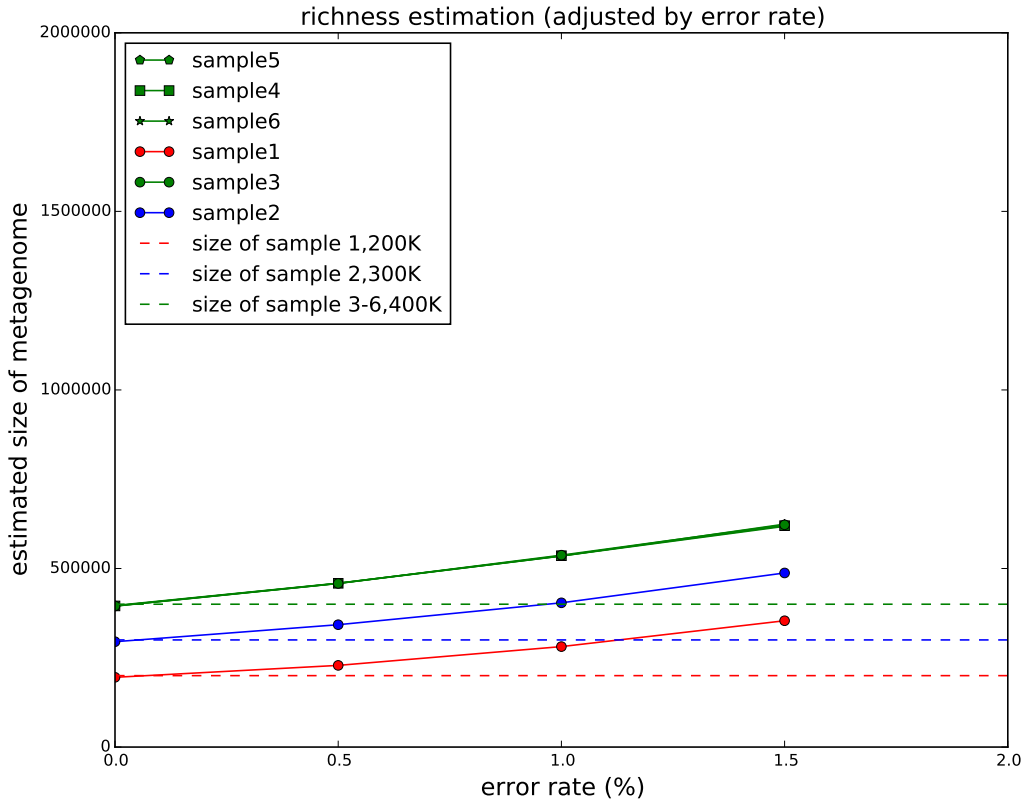

Figure 5.10: **Richness estimation using IGS method adjusted by sequencing error rate.**

5.3.2.2 the effect of Bloom filter size on the accuracy of analysis

As discussed in the chapter about k-mer counting, the collision in bloom filter which we use for efficient k-mer counting will result in counting error. If the false positive rate for a specific bloom filter we use for k-mer counting is 0.1, 10% of the k-mers will have incorrect counts. When we use median k-mer count to get read coverage, such incorrect count has

the effect on two aspects. On one hand, some k-mers in a read will have incorrect higher count. However, if the false positive rate is low, this will not affect median k-mer count. This shows the method of using median k-mer count to get read coverage is not only less prone to sequencing error, but also less prone to the inaccuracy characteristics of underlying data structure. One the other hand, this inaccurate count also affects the counts of those erroneous k-mers generated by sequencing error. For example, 3 errors in a read affect the count of 43 k-mers, the counts for these 43 k-mers are supposed to be 1. However, because of the collision in the Bloom filter and the resulting incorrect k-mer counting, if the false positive rate is 0.1, about 4 out of the 43 k-mers will have inflated count, mostly as 2. So the combined effect of sequencing error and collision in bloom filter is that some reads will have incorrect coverage as 1 and some reads will have incorrect as 2. We can get the percentage of total reads that will have such incorrect coverage, using statistical model similar to that discussed in last section. Using same example, 3 errors occur in a read, if the 3 errors affect 41-45 k-mers(with a chance of 0.20), the median k-mer count will be 2, due to the collision in bloom filter, while if he 3 errors affect more than 45 k-mers(with a chance of 0.24), the median k-mer count will be 1, purely due to sequencing errors.

We did the same experiment but also adjusted the estimation according to the false positive rate of bloom filter and got better estimation, as shown in Figure 5.11.

With adjustment to estimation taking sequencing error and collision in bloom filter into account, as shown in Figure 5.11, the estimated genome size is closer to real number. With an error rate of 1%, a false positive rate of 0.1, and with 10X coverage data, the estimated genome size is about 20-25% more than real number. However the estimation is still increasing with higher error rate. This means there are still other factors influencing this accuracy of the estimation that we failed to take into account.

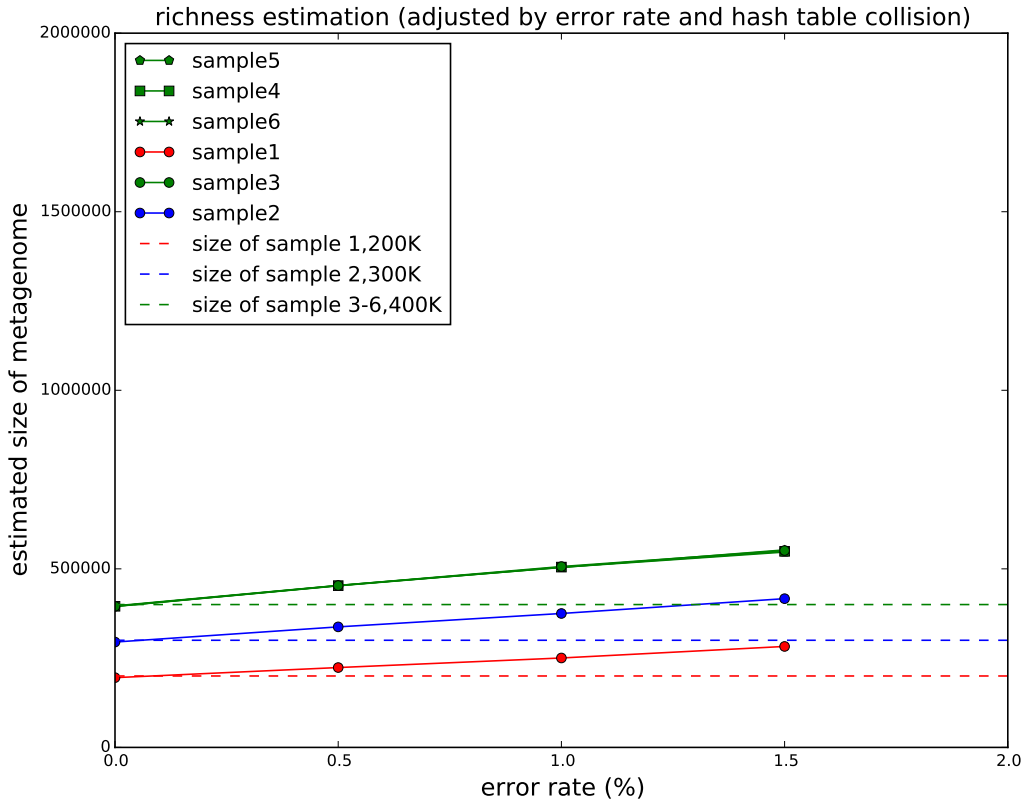

Figure 5.11: **Richness estimation using IGS method adjusted by sequencing error rate and false positive rate of bloom filter.**

5.3.3 the effect of sequencing depth to the accuracy of IGS method

We have shown that the IGS method can generate good result from relatively high coverage data (like 10X). It is expected that the higher the coverage of data is, the more accurate analysis we can conduct. However for many metagenomics project,especially environmental samples, it is difficult to yield high enough sequencing depth. We investigated the effect of sequencing depth on the accuracy of the IGS method.

Figure 5.12 shows how well the matrix calculated from a data set with variable coverage reflects the real relationship between samples. It is as expected that higher coverage data will yield a more accurate distance matrix. Note even with a coverage as low as 0.1, the

correlation is 0.89. This can give us the hint about how reliable the result will be if we only use a small proportion of data from a large metagenomic data set. So the beta diversity analysis using IGS method not only is less prone to sequencing error, but also less prone to sequencing depth.

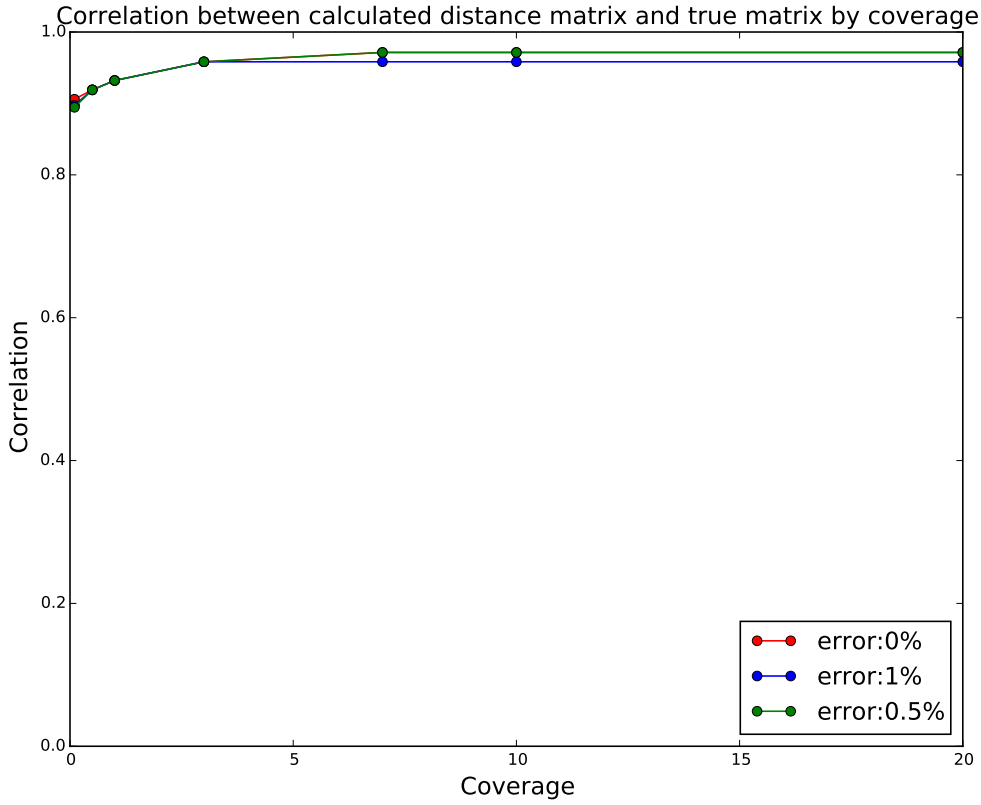

Figure 5.12: **Correlation between calculated distance matrix and true matrix from different data sets with different sequencing depth.**

Figure 5.13 5.14 5.15 shows the estimated genome size from data sets with variable coverage with different error rates. It's interesting that the estimated genome size is very high with extremely low coverage. This is probably due to the limits of the statistical model in estimating the total size of information with limited observed information. After all, only a small proportion of the genomes in the sample are covered by reads.

We can see the pattern again here that higher error rate will influence the accuracy of genome size estimation, especially when the coverage is low. However for error rates from 0% to 1%, as long as the coverage is higher than 1X, the estimation of genome size starts to be stable. It is important to point that even though the absolute value of estimated genome size may be overestimated, the relationship between samples is reliable, as shown in the figures. Sample 3,4,5,6 all have 4 species, while sample 2 has 3 species, and sample 1 has 2 species. They can be separately effectively.

The estimation of genome size does not increase much with increasing coverage, even for the data set with error rate as 1%. This proves that the adjustment method discussed previously does eliminate most of the bad effect of sequencing errors. That being said, it is still beneficial to do some preprocessing to the data to reduce the error rate. If the error rate can be reduced from 1% to 0.5%, the estimated size of genome will be more accurate. This again demonstrates the importance of the streaming method doing error profile analysis discussed in chapter 4 above.

5.3.4 Compare IGS method to COMMET in beta diversity analysis

Next we test how well the matrix calculated by various methods can reflect the real relationship between samples. COMMET[75], the successor of Compareads [76], is one of few software packages for comparing metagenomes. It is based on the method of count shared reads between metagenomes. The higher percentage of reads shared by two metagenomes, the more similar the two metagenomes are inferred to be. So basically this is a straightforward abundance-based method to evaluate the similarity.

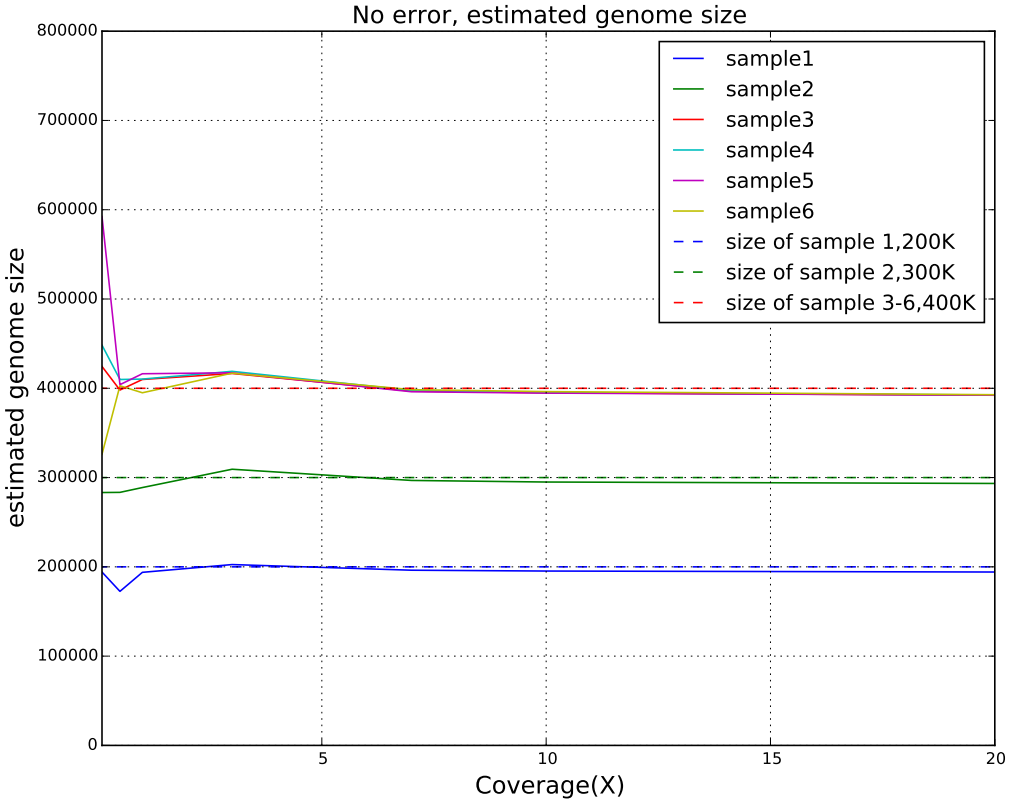

Figure 5.13: **Estimated genome size from data sets with variable coverage, without error.**

We have the simulated data set with sequencing depth as 0.1X and 10X, with sequencing error as 1% and without sequencing error was used in this experiment. This data set has the same species composition as that used in other experiment previously.

As shown in Figure 5.16, firstly, for all data sets, the matrix from IGS method has a higher correlation to golden standard than that from COMMET. As expected, the matrix from data sets with sequencing error has a lower correlation than that from error-free data sets. COMMET is more prone to sequencing error rate, compared to IGS method, for high coverage data or low coverage data. Also higher coverage will yield more accurate matrix, which is not surprising.

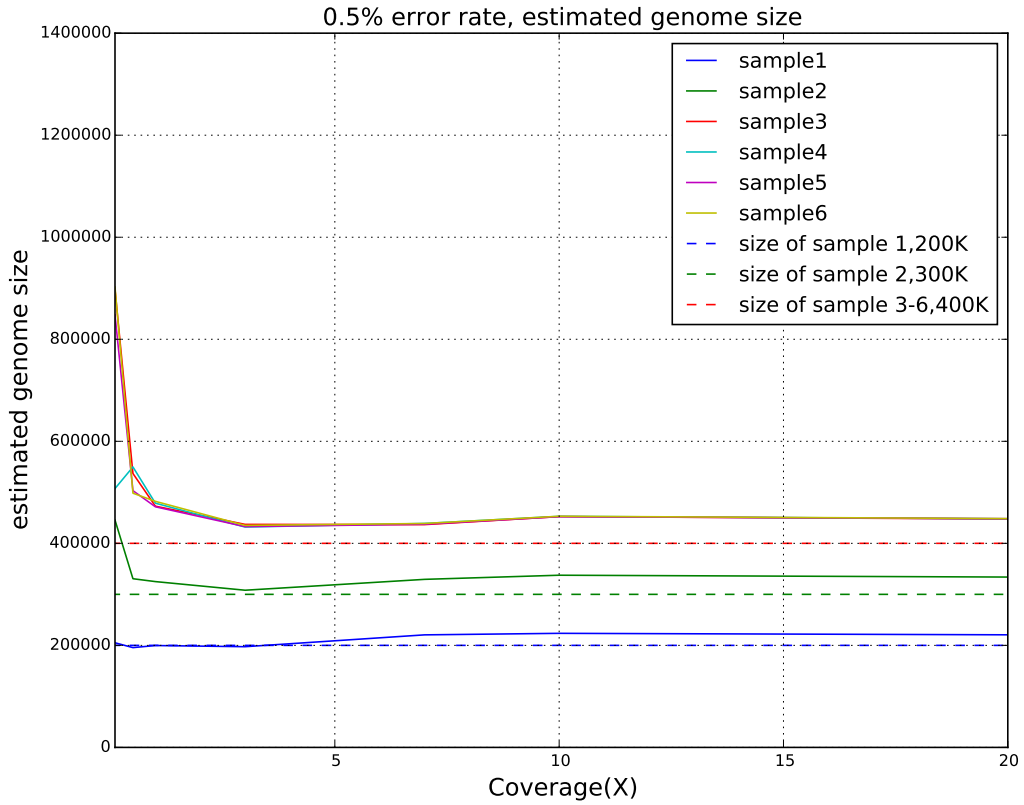

Figure 5.14: **Estimated genome size from data sets with variable coverage, with error rate as 0.5%.**

In the experiment below with real metagenomic dataset, we will see more evidence that the IGS method has better performance than some other metagenome comparison methods.

5.3.5 The IGS method can provide a whole framework to do alpha or Tbeta diversity, with good versatility.

From the testing using simulated data sets shown here, we are confident that our IGS method works well and can give reliable results from data sets with error and low sequencing depth.

The IGS method can provide a whole framework to do alpha or beta diversity. Here we tested beta diversity using only Bray-Curtis metric and alpha diversity on richness only. In

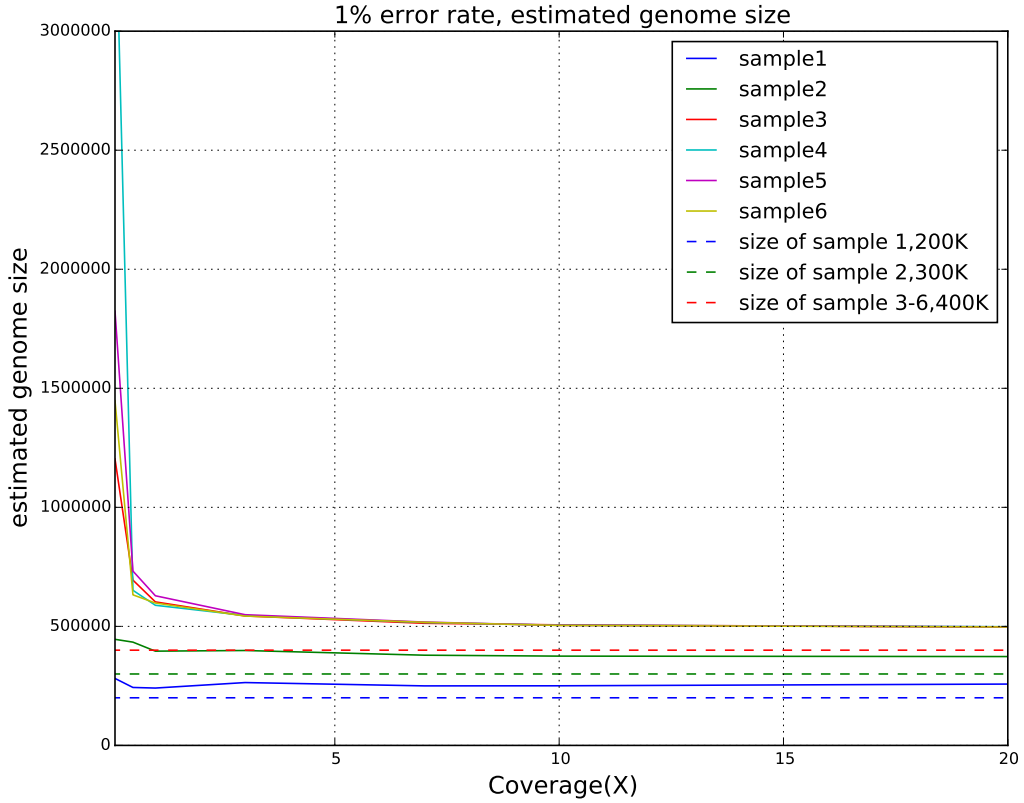

Figure 5.15: **Estimated genome size from data sets with variable coverage, with error rate as 1.0%.**

fact, any standard metric can be applied to the IGS-by-samples table.

The other software package to do metagenome comparison - Compareads/COMMET - based on reads overlap between samples can get a matrix reflecting the real relationship between samples. However, it is stuck with one metric, which is based on the percentage of overlapping reads between samples. This metric is like Bray-Curtis, but not exactly the same.

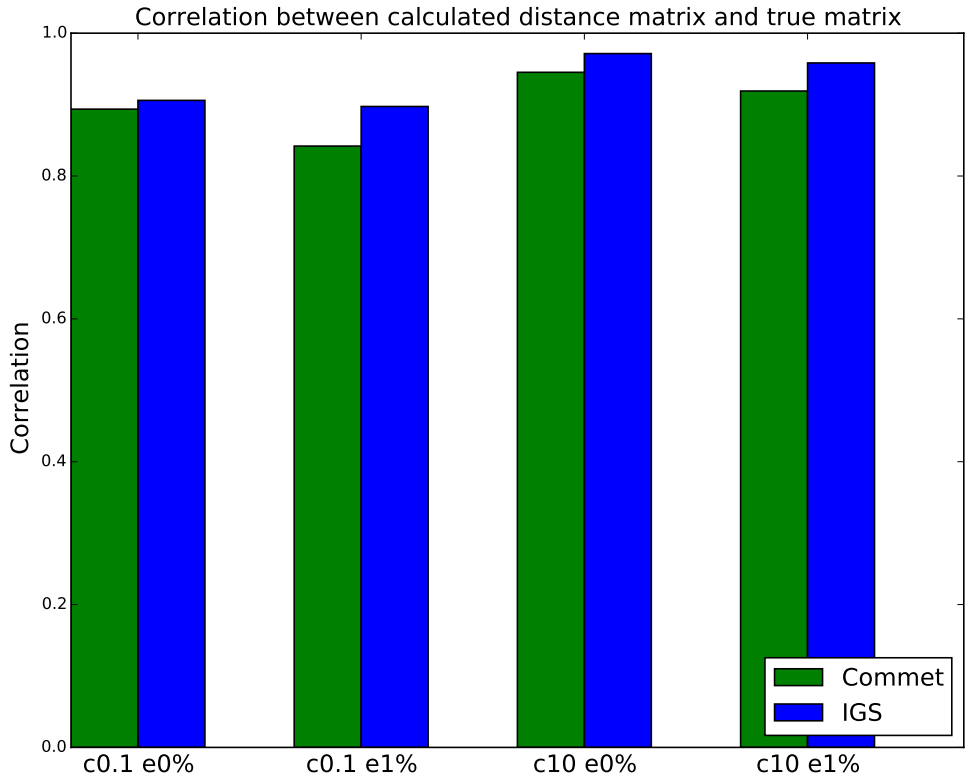

Figure 5.16: **Correlation between calculated distance matrix and true distance matrix from different data sets and using different methods.**

5.4 Applying IGS method to real metagenome data sets

Having shown that the IGS method delivered good results about microbial diversity from simulated synthetic data sets, we will now evaluate the novel method on several published metagenomic datasets, including samples from ocean, human microbiome and soil. For the ocean sample and human microbiome data sets, we will compare the result from IGS method with that from the original publication. For soil sample, since there is no other diversity analysis that has been conducted to these data sets, we will show the result we got from IGS

and try to interpret the ecological meaning.

5.4.1 GOS data sets: Sorcerer II Global Ocean Sampling Expedition

We tested the IGS method on a well known public dataset from the Sorcerer II Global Ocean Sampling expedition. During the expedition, 44 water samples were collected from different locations across the Atlantic and Pacific Oceans and were sequenced using Sanger technology. The whole dataset is composed of 7m reads from of 44 samples. A whole metagenomic comparison of the samples was done using a sequence alignment method in the original research.

The IGS method took only several hours on MSU HPC to generate the dissimilarity matrix of the samples thanks to the scalability and distributability of the IGS based method. After clustering, Figure 5.17 shows that, consistently with the original study, the samples are clustered according to their geographical origin. The group with yellow color contains samples from Tropical- Galapogas. The group with light purple color contains samples from Tropical -Open Ocean. The group with dark purple color contains samples from Sargasso. The group with green color contains samples from Temperate.

If we compare the cluster we got from IGS method with the cluster in the original study, we can see the IGS method yields a cluster more consonant with the sample origin than the method used in the original study. For example, in the original study, sample 14,21 and 22 from Tropical - Galapogas are separated from other Tropical- Galapagos samples, while in Figure 5.17 they are grouped together. Also, samples 00a,00b,00c,00d, all from the same location, are grouped together in our result,while in original research, sample 00a is

separated from the other three samples.

Compared with the clustering generated using Compareads, our method is comparable, with some distinct differences. For example, sample 16 is clustered together with 15,17, 18, and 19 in our result. However, in the result by Compareads, sample 16 is clustered with 23 and 26, in contradiction to the geographical origin of the samples.

Next we used IGS method to analyze the alpha diversity. Figure 5.18 shows the rarefaction curve of IGSs of the samples. As expected, we cannot see the saturation,which means the sequencing data set is still far from deeply covered. Because the data sets for different samples have dramatically different sizes, we estimated the total number of IGSs using the Chao1 estimator with a limited number of reads in each sample (50000) to make sure the smallest data set has enough reads for comparison, as shown in Figure 5.19 .

We see that the richness of samples is related to the geographical origin. The samples from tropical areas have a higher richness than the samples from more northern areas. The relationship between samples is consistent with the clustering in beta diversity analysis shown above. As discussed in the section above about alpha diversity analysis to synthetic data, such number of total IGSs may be over-estimated but the relative relationship between samples on richness should be reliable. (This is not discussed in original research work on the GOS samples.)

5.4.2 Human Microbiome Project(HMP) metagenomics data set

We tested the IGS method on 12 HMP (Human Microbiome Project) samples from different body parts including skin, oral and vaginal. Principal component analysis (Figure 5.20) shows the samples are separated well by the body parts where they are collected.

Rarefaction curve and estimated number of IGSs show that the richness of samples is

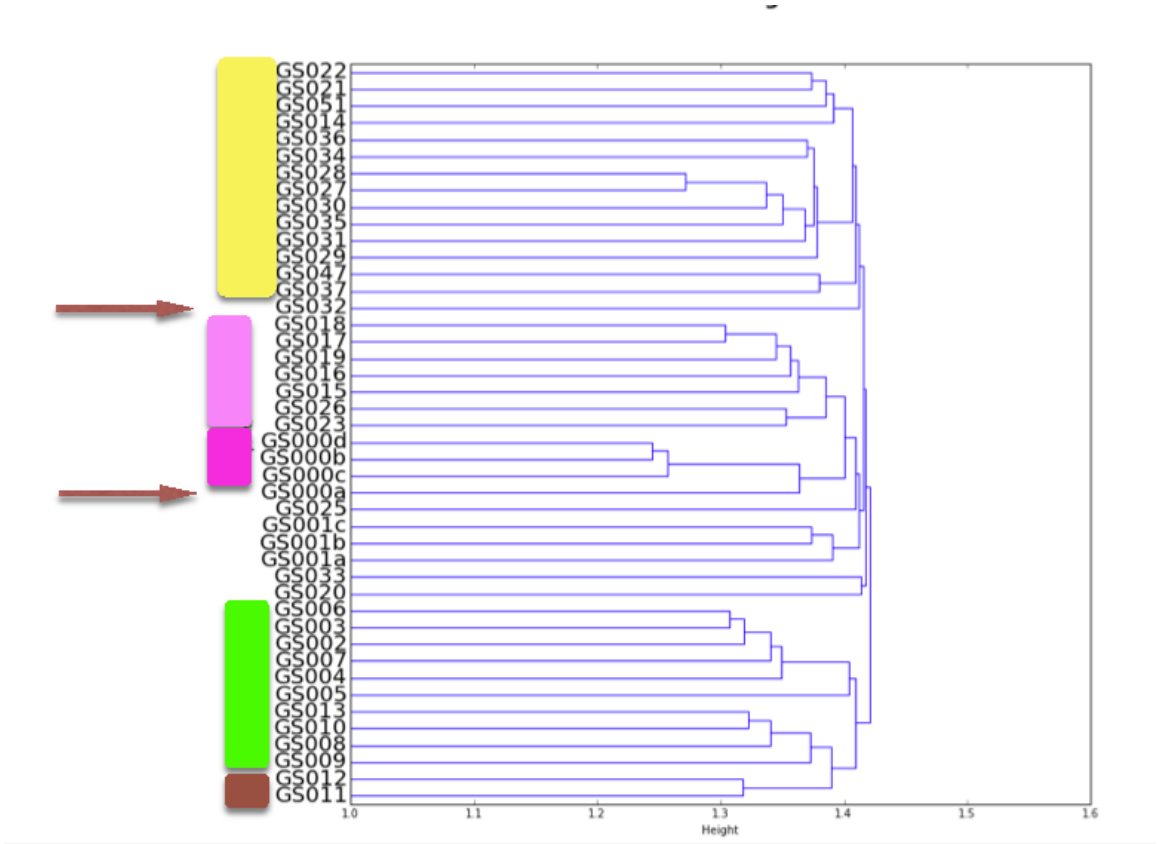

Figure 5.17: **Clustering of Global Ocean Sampling Expedition samples using IGS method.**

related to the body part where they are collected. The oral samples have higher richness than skin or vaginal samples, which is consistent with other research. [55]

5.4.3 GPGC - Great Prairie Soil Metagenome Grand Challenge

Having tested the IGS method on two relatively smaller metagenomic data sets, we will now use it to analyze a larger data set from 8 soil samples collected from fields with different treatments and different locations across the great prairie region in the US. (Table 5.6).

As discussed above using simulated data sets, read data sets with lower sequencing coverage will reduce the accuracy of the analysis. However, as shown in Figure 5.12, with sequencing depth as 0.1x, the calculated distance matrix using IGS method still has a rea-

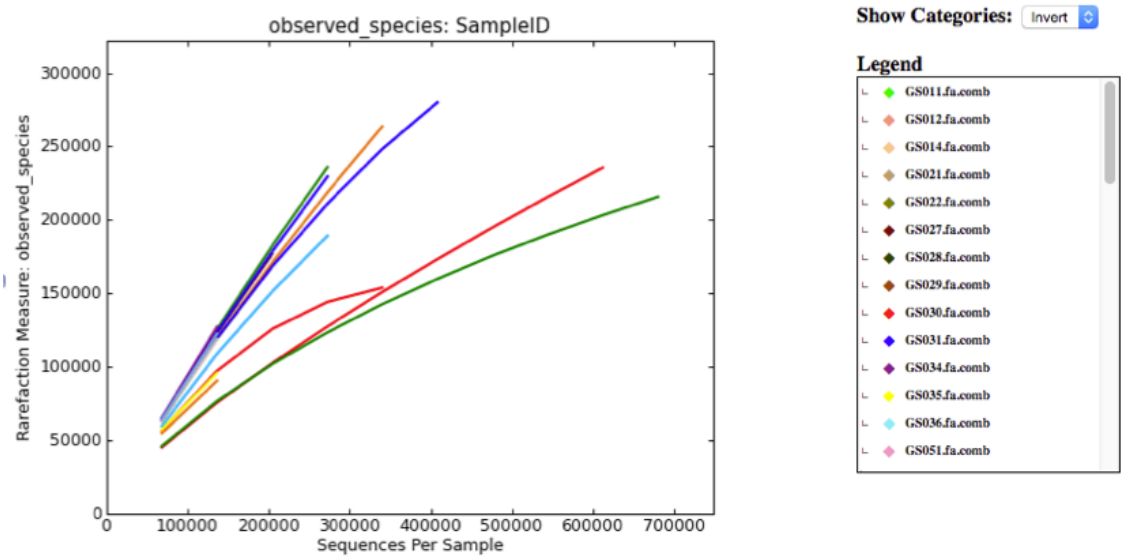

Figure 5.18: **Rarefaction curve of IGSs of Global Ocean Sampling Expedition samples.**

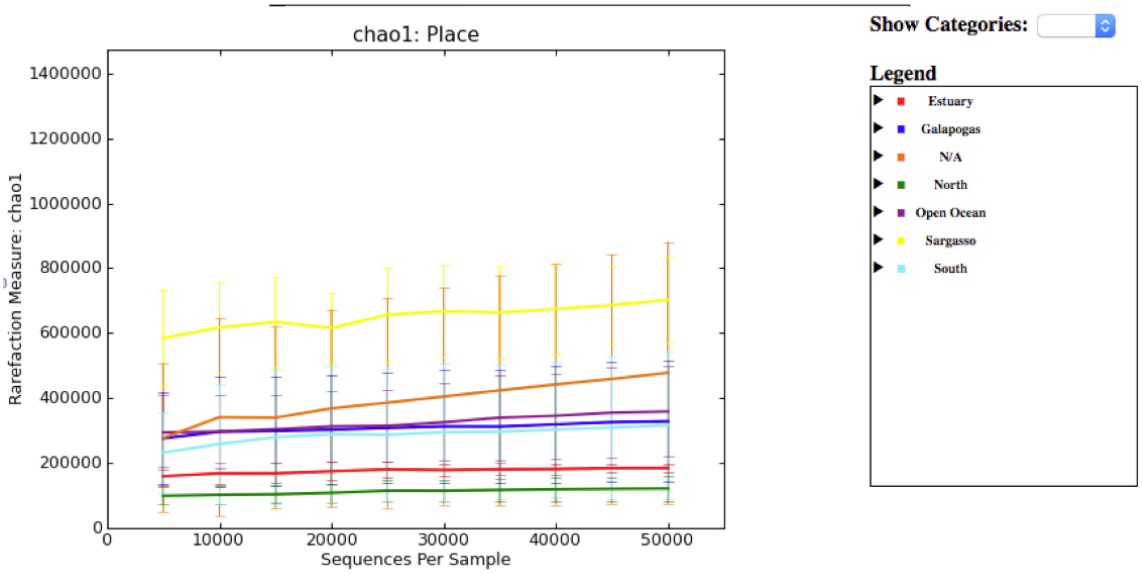

Figure 5.19: **Estimated number of IGSs of Global Ocean Sampling Expedition samples.**

sonably high correlation with golden standard distance matrix. So we can use subset of a large data set to acquire the diversity information, with the trade-off of lower accuracy.

For the GPGC datasets, we made a subset with 2 million reads from each sample and

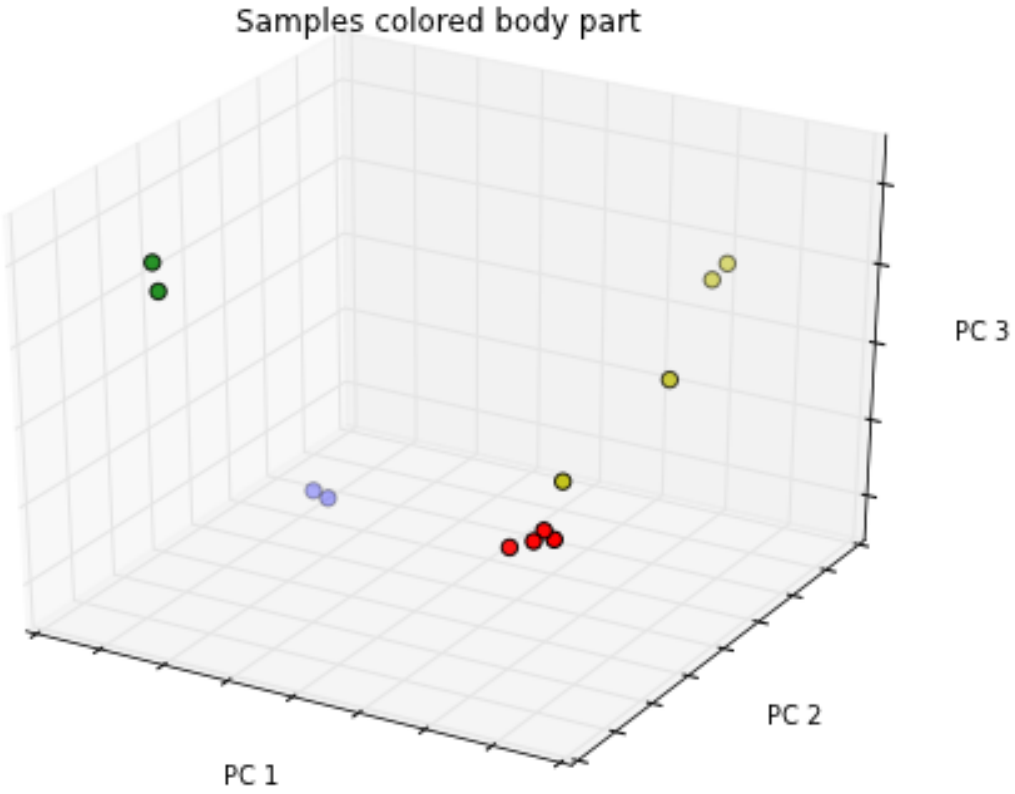

Figure 5.20: **Principal coordinates analysis of 12 Human Microbiome Project samples, red: anterior nares- skin, green: throat -oral, blue: buccal mucosa -oral, orange: posterior fornix -vaginal.**

conducted the diversity analysis using the IGS method.

Principal coordinates analysis (Figure 5.22) shows the samples are separated well by location where they are collected.This proves that the geographical origin plays a more important part in determining the similarity of genomic composition of samples, compared to different treatments.

Figure 5.23 shows the rarefaction curve and estimated number of IGSs of the samples. Basically the "corn" and "switchgrass" samples have higher richness than "restored" and "prairie" samples. This observation that cultivation increases the richness of soil is consistent with the intermediate disturbance hypothesis [140]. The disturbance from treatment

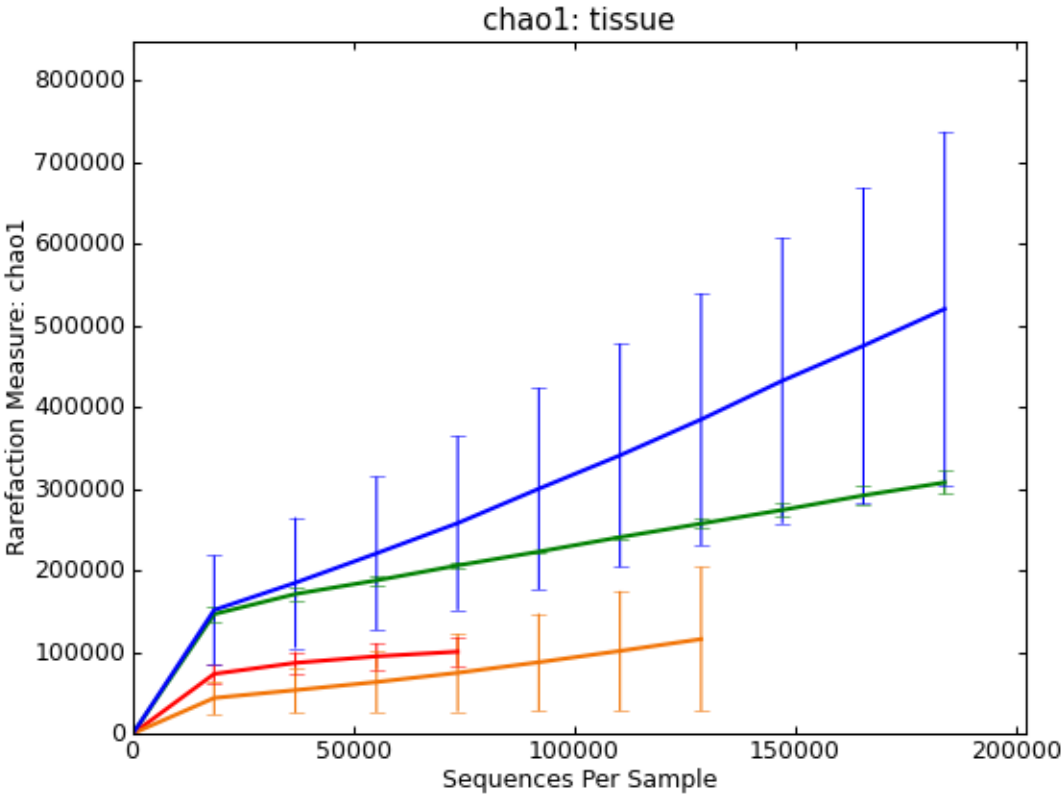

Figure 5.21: **Alpha diversity of 12 Human Microbiome Project samples: estimation of metagenome size of HMP samples, red: anterior nares- skin, green: throat -oral, blue: buccal mucosa -oral, orange: posterior fornix -vaginal.**

like cultivation opens more niches and the stable communities in prairie eliminate some populations by the principle of competitive exclusion.

Its harder to explain the rank by state. The Kansas site experiences more drought stress and higher tempseratures. The Iowa and Wisconsin sites experience more cold, especially freezing conditions arresting their biology for 3-4 months. However, the freeze-thaw cycles also kill off some each cycle, which is similar to intermediate disturbance. With new growth each spring, this new growth would be the fast growers with less diverse. Why Iowa is the least diverse is still difficult to explain for now.

From the alpha diversity, we also have a rough estimation of the total size of the

sample	# of reads	size of .gz file	# of bps	ave. length
iowa corn	1,514,290,825	46G	144,202,427,079	95.2
iowa prairie	2,597,093,273	74G	226,815,059,143	87.3
kansas_corn	2,029,883,371	66G	206,933,829,048	101.9
kansas_prairie	4,987,358,734	145G	499,387,223,498	100.3
wisconsin_corn	1,616,440,116	51G	162,257,698,471	100.4
wisconsin_prairie	1,653,557,590	53G	166,467,901,724	100.7
wisconsin_restored	226,830,595	11G	34,241,520,930	151.0
wisconsin_switchgrass	310,966,735	13G	40,259,619,921	129.5

Table 5.6: **GPGC data sets**

metagenome in Iowa soil, which is about 540G base pairs. This proves the high complexity of soil sample and we still need considerably more sequencing effort to achieve a reasonable high coverage.

5.4.4 More soil metagenomic samples

Additionally we test the IGS method on two other unpublished data sets. One is a series of soil samples collected from KBS with different treatment. Figure 5.24 shows the IGS method can separate the samples by treatment well.

The other data set is a series of soil samples from Amazon rainforest. The samples are separated well by the treatment. (Figure 5.25) It is also obvious that samples from forest have lower richness than prairie. (Figure 5.26)

5.5 Data

5.5.1 Code availability

The algorithms of the IGS based diversity analysis are implemented in the khmer software package, written in C++ and Python, available at github.com/ged-lab/khmer/. khmer also

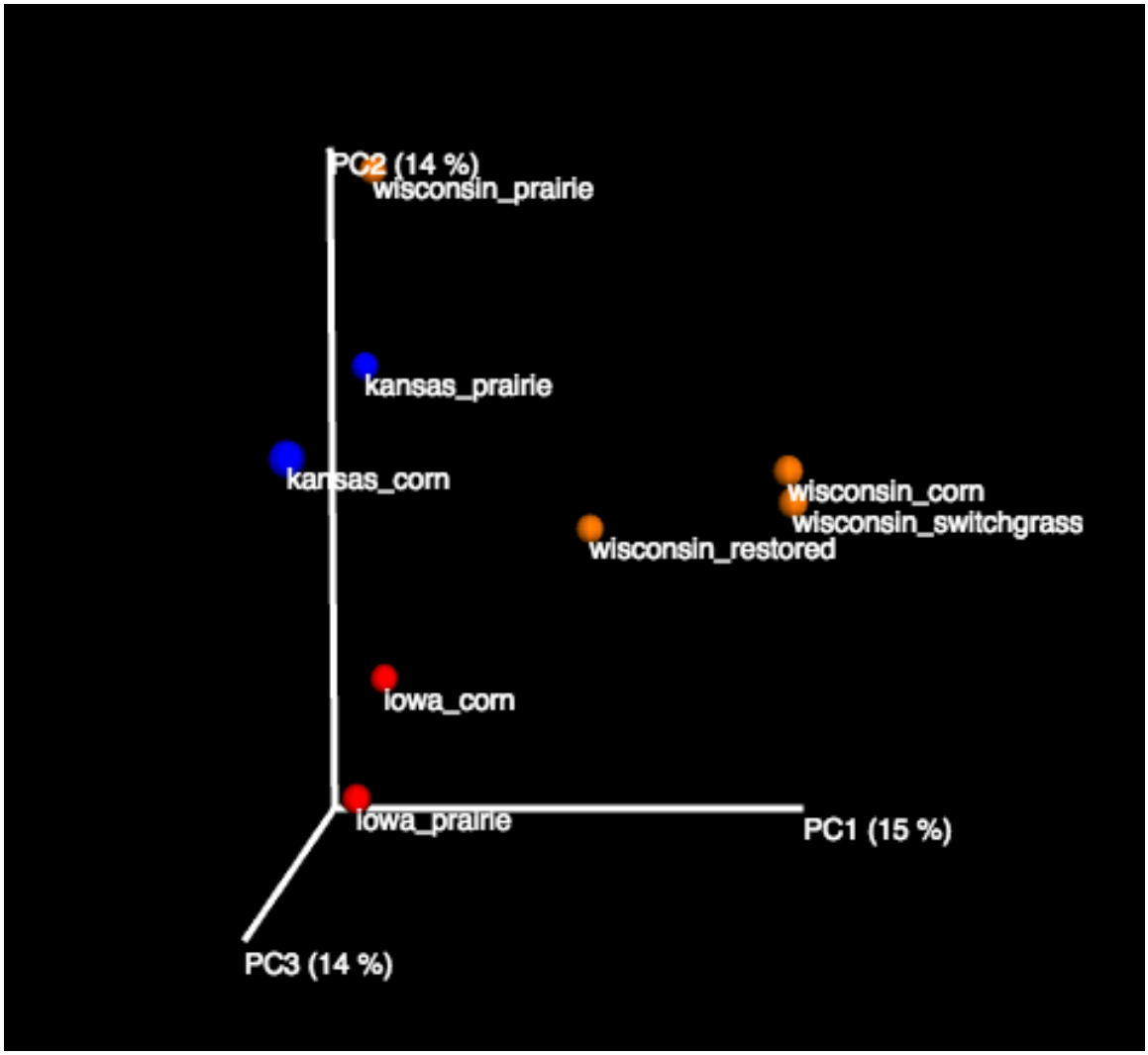

Figure 5.22: **Principal coordinates analysis of 8 Great Prairie Soil Metagenome Grand Challenge (GPGC) samples.**

relies on the screed package for loading sequences, available at github.com/ged-lab/screed/. khmer and screed are Copyright (c) 2010 Michigan State University, and are free software available for distribution, modification, and redistribution under the BSD license.

The code and detailed instruction used to generate all the results in this chapter is available at http://github.com/ged-lab/2013-diversity/.

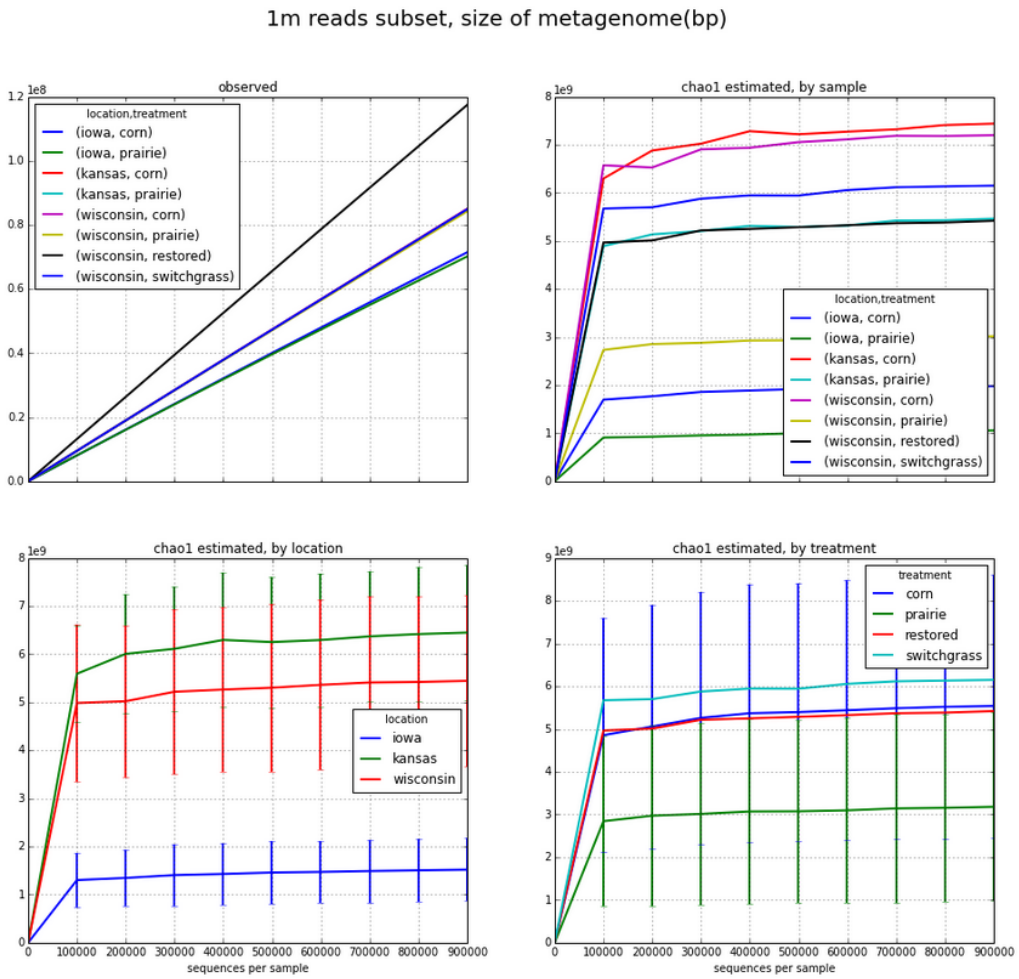

Figure 5.23: **Alpha diversity analysis of 8 GPGC samples. Upper left, rarefaction curve of IGSs. Upper right, estimated number of IGSs in different samples. Lower left, estimated number of IGSs in samples grouped by location (Iowa, Kansas and wisconsin). Lower right, estimated number of IGSs in samples grouped by treatment (corn, prairie, restored, switchgrass).**

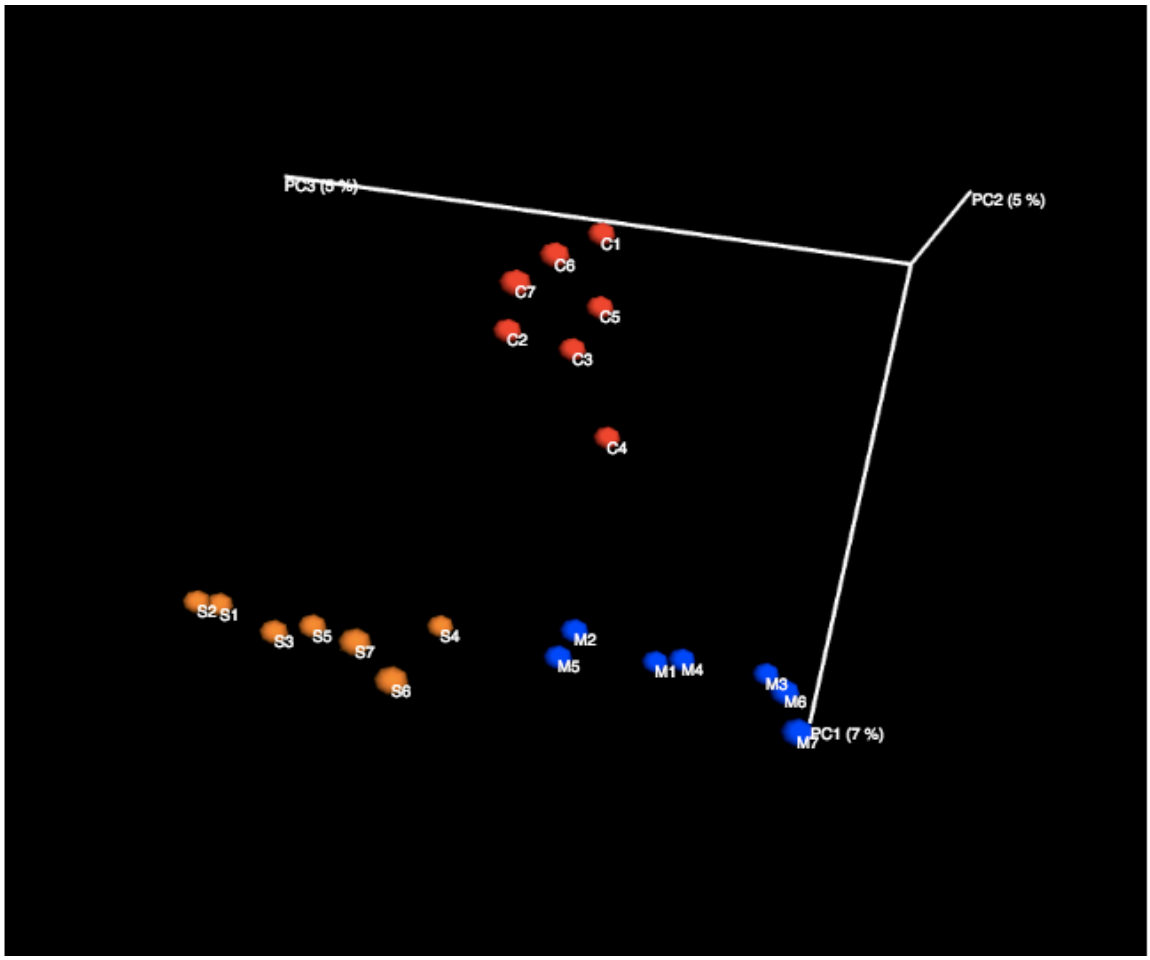

Figure 5.24: **Principal coordinates analysis of soil samples with different treatments collected from Kellogg Biological Station(KBS). Red, corn. Blue, miscanthus. Brown, switchgrass.**

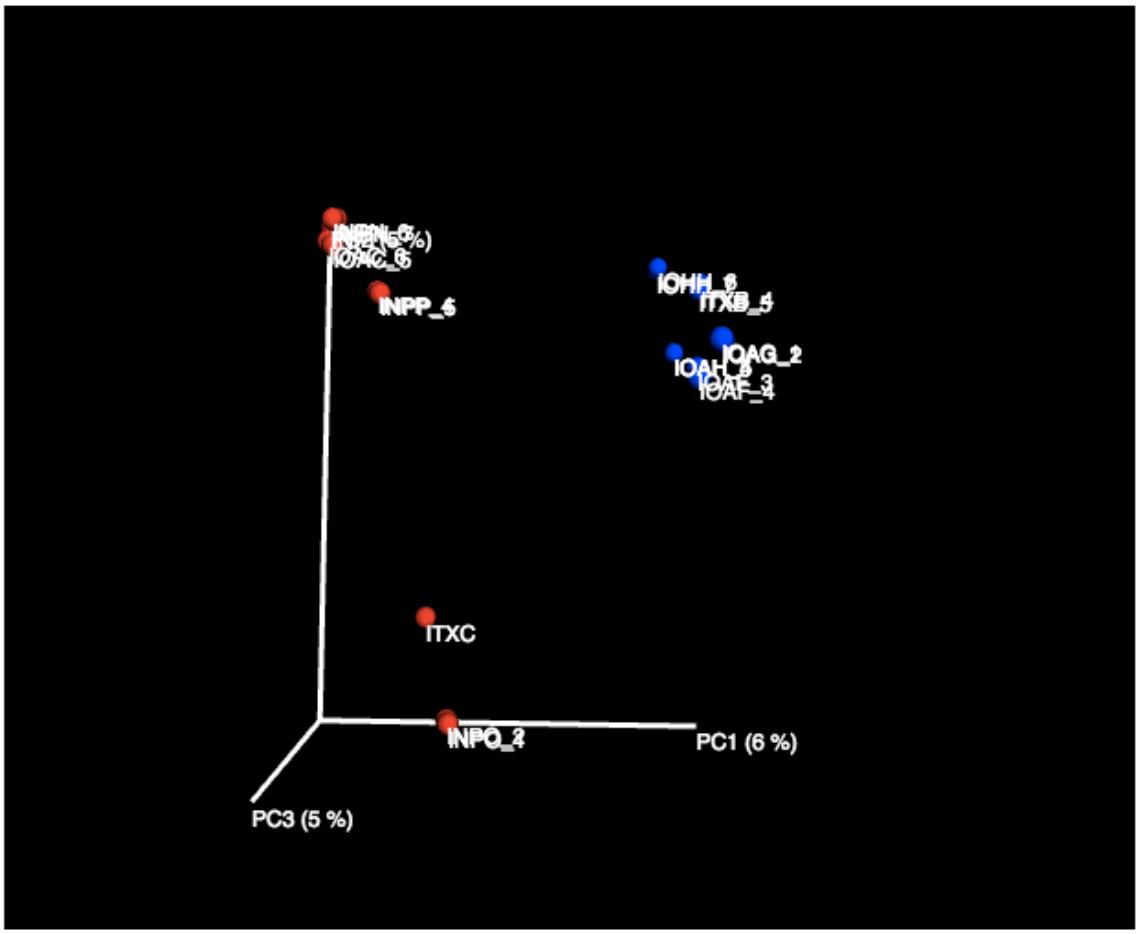

Figure 5.25: **Principal coordinates analysis of soil samples collected from Amazon rainforest. Red, forest samples. Blue, prairie samples.**

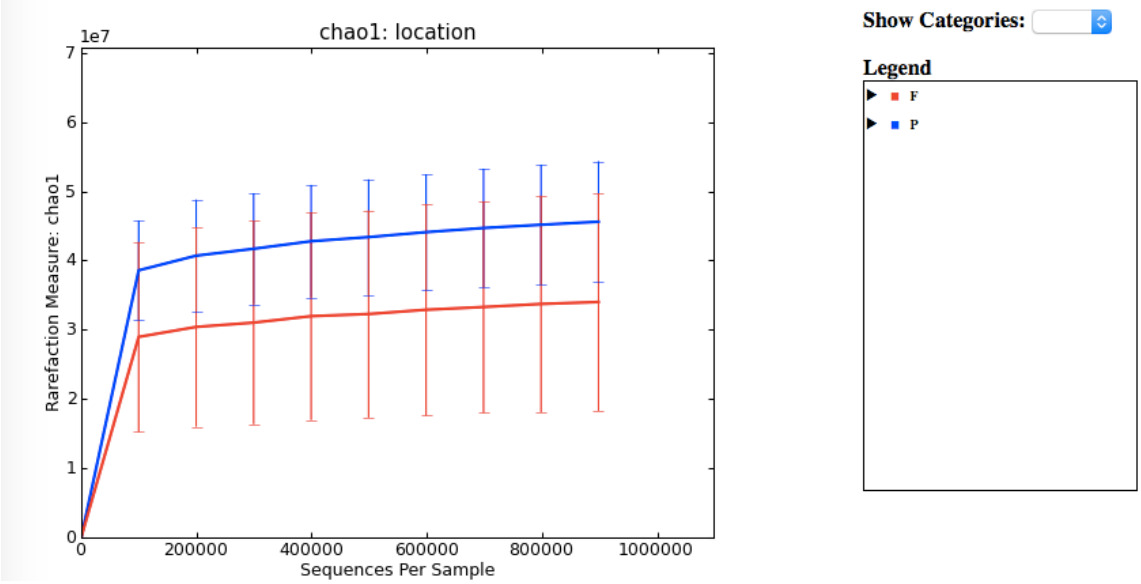

Figure 5.26: **Estimated number of IGSs in metagenomic data from soil samples collected from Amazon rainforest. Grouped by treatment. Red, forest samples. Blue, prairie samples.**

Chapter 6

Conclusion

We have developed a novel statistical framework to enable microbial diversity analysis using whole genome shotgun metagenomic reads data without the requirement of assembly, binning, reference or annotation. This dissertation covers an overview of existing approaches of microbial diversity analysis of metagenomic samples, especially based on the concept of OTU, including the steps in the procedure, such as contigs binning, statistical analysis of OTU abundance information to estimate the microbial diversity. Next the statistical framework based on the novel concept of IGS was discussed. As the foundation of the framework, we described a novel method to count k-mers efficiently and a scalable approach to retrieve the coverage of a read in a data set based on efficient and online k-mer counting. We also introduced the applications of this approach in reducing the redundancy of metagenomic reads dataset and analyzing sequencing error, which is beneficial to other tasks in metagenomic data analysis, like assembly or error trimming. Finally, we discussed how we developed the concept of IGS based on the methods of efficient k-mer counting and digital normalization discussed before. The application of IGS to analyze microbial diversity of metagenomic data sets was discussed and the performance of the IGS method on simulated data sets and real data sets were demonstrated in the final chapter. In this chapter, we summarize how the novel statistical framework based on IGS makes a difference to the diversity analysis in current microbial ecology research. Finally some directions of future work will be discussed.

6.1 Summary

Diversity analysis is a key part of the microbial ecology research, like of macroorganism ecology. However due to the obscure definition of the term "species" in microbial ecology, we can virtually never measure the diversity of species directly, rather we use other taxonomic concepts like operational taxonomic unit (OTU) to evaluate the diversity of microbial community, instead of species. 16S rRNA sequencing reads may be classified into different OTUs. Shotgun whole genome sequencing reads can also be classified into OTUs. But most, if not all the existing methods based on the concept of OTU rely heavily on preprocessing of original reads data in some way like assembly or external information like reference sequences for annotation. Both of the prerequisites are not satisfied for many metagenomic projects. For metagenomic data set with low sequencing coverage, the assembly process skews the analysis by including primarily the most abundant organisms. Sequences that are rare are not assembled into contigs and are therefore not included in the contig analysis It is common that only a small proportion of reads can be used in assembly especially in a complex environmental sample. [50] The reference sequence database is far from completion especially for microbes in environmental samples from soil or sea water. The paucity of reference databases affects the ability to identify functional capacity of microbes, when traits cannot be identified. Thus, applying these methods can only obtain an incomplete diversity of the microbial community in the metagenomic sequencing data set.

The IGS based method discussed in this dissertation offers a novel framework that overcomes the limitations of assembly, binning, or annotation without the requirement of reference sequence database. It can take advantage of all the information in the metagenomic reads data and gain a full picture of the diversity of the microbial community. Importantly,

this is a new framework with the concept of IGS instead of OTU as the taxonomic unit to analyze microbial diversity. Thus, this framework can be used to perform all possible diversity analysis that OTU-based framework can do. Moreover, this is a more thorough approach than many other methods developed to solve only specific problems in the field of diversity analysis. For example, there are several methods developed to estimate the species richness in a metagenomic sample [108]. Our IGS based framework cannot only estimate the species richness or size of metagenome as shown in the section above, but also it can estimate the evenness or species abundance distribution of a metagenomic sample, which is also an important aspect of alpha diversity analysis. For beta diversity or compositional similarity analysis between metagenomic samples, there are several methods developed to compare metagenomic samples based on reads mapping or counting shared reads [107]. However they only estimate abundance-based similarity, similar to the Bray-Curtis indices used in the experiment discussed in the section above. It should be noted that the IGS-based framework can also be used to estimate incidence-based similarity, which cannot be estimated using other existing approaches.

Besides the potential for a broad application of the IGS based framework, it is also efficient and highly scalable to handle extremely large metagenomic sequencing data sets. We have discussed the efficiency of the novel k-mer counting method and the following method of digital normalization, with the ability to retrieve the coverage of a read accurately and efficiently. We also performed a thorough analysis to examine the effect of the size of used data structure to the accuracy. We can take advantage of the probabilistic characteristics of the data structure to make a trade-off between expected analysis accuracy and expected usage of computational power. In this way, we make the analysis highly scalable to keep pace with the increasing size of metagenomic sequencing data.

In addition, we examined the effect of sequencing depth to the accuracy of estimating microbial diversity. It was expected that using more number of reads, that is, a data set with higher sequencing depth increases the accuracy of diversity estimation. For similarity analysis between samples (beta diversity), a data set with relatively low sequencing depth can still get decent results, as shown in the experiment with synthetic data and real soil data sets. However, for alpha diversity such as richness estimation, use of a data set with lower sequencing depth results in the diversity estimation more distant from the real number. Although the absolute value of such species richness of a sample is not accurate, the relative comparison of species richness between samples is less prone to smaller reads data with lower sequencing depth. These results suggested that for a specific purpose, only a subset of the large metagenomic reads data can be enough to achieve reasonably satisfying result. Under certain circumstance, this feature is quite helpful and can reduce the computational expense dramatically.

6.2 Future work

Though this dissertation demonstrated the performance of the new approach to analyze microbial diversity using whole genome shotgun sequencing data without the requirement of assembly, binning, or annotation, there is still plenty of room for improvement.

Primary questions in the process of developing the IGS based method are how many species there are in a sample or how similar the samples are with each other, mostly focusing on the quantitative aspect. Admittedly these are important questions to the microbial ecologists, but they are also curious about the qualitative aspect, such as what drives differences between samples and eventually its functional potential[143]. Thus, a natural expansion of

the IGS based framework will focus on answering questions mentioned above.

Now we have an efficient and scalable approach to obtain the coverage of a read in a sample, it is straightforward to extract the reads according to its coverage profile across samples so we can get a subset of reads that have specific properties, like the reads that are common in all samples. In this way we may collect these "common" reads across the samples and try to co-assemble them since now they should have higher coverage. Or we can get a subset of reads that are common in a group of samples but do not exist in another group of samples, like the samples from patients and healthy persons. These "signature" reads may offer important insights to understand what happens to the microbial community while the environment changes. Admittedly these kinds of "extraction" can be implemented using other methods like reads alignment method. However, they may not be as efficient and scalable as the IGS based method, especially for extremely large metagenomic data.

One advantage of the IGS based method is that binning is not required in this procedure. Firstly, traditionally binning is used to classify contigs after read assembly effort. The similarity based binning method relies on sequence alignment, which is inefficient, even infeasible for large metagenomic data. Secondly, reference sequences are normally required for similarity based binning approach. The composition-based approach relies on the frequency profile of sequence signatures and machine learning approach on that profile, which is computationally expensive. The third approach based on coverage profile across samples was developed recently[2, 59, 3, 94, 57].Mostly, the coverage profile is used with the companion of composition frequency profile to classify contigs. The assumption on which the coverage profile based binning approaches are based on, that contigs with similar coverage profile across samples are more likely to be from the same microbial species, is actually similar to the assumption on which using IGS to do beta diversity is based, that the IGSs with similar

coverage profile across samples are likely to be from the same microbial organism. Thus, it is promising to classify the IGSs by the coverage profile across samples. We have already overcome the challenge of retrieving the coverage profile efficiently based on the probabilistic data structure, while in those coverage profile based binning approaches the coverage profile is normally retrieved by assembly of contigs and mapping reads back to contigs, which both require higher coverage reads to do assembly and are computationally expensive.

There are two obstacles to overcome in this coverage profile based IGS binning approach. First, with relatively small number of samples, the resolution will be limited, since the total number of different coverage profiles will be limited. This is probably the reason why most of those coverage profile based contig binning methods have to integrate composition profile information also. Second, on the other hand, if there are a large number of samples, there will be too many different coverage profiles. We can use more sophisticated approach to classify the coverage profiles to reduce the number of bins, as in those coverage profile based contigs binning methods. Another approach worthy of note is that there is a method termed partitioning developed in our group as a divide and conquer approach to scale metagenome assembly. It can be considered as a binning approach also, where the reads in the same partition are more likely to originate from the same microbial organism. We can try to integrate the partitioning and IGS coverage profile to improve the accuracy of the binning. In summary, this will be one of the first attempts to do reads binning. After the IGS/reads binning, we expect to do better assembly and annotation and gain more knowledge about the function and phylogenetic information.

We have shown that after adjustment according to sequencing error and collision rate of the bloom filter, the estimated size of metagenome is close to real number for synthetic data sets. Howevere, the difference between the estimation and real number is still increasing

with higher error rate, which means there are other factors that affect the accuracy of estimation. This is worthy of further investigation. The size estimation of metagenome is extremely important in metagenomic data analysis and it is closely related to the estimation of sequencing depth or how much more effort is required to gain enough sequencing depth. As shown in the results, we are confident that the relative relationship between the richness of different samples is reliable from the IGS based alpha diversity analysis. How accurate the absolute value of the richness or the size of metagenome in a real data is requires further investigation and new statistical model may be needed to adapt to the abundance distribution of IGSs. Furthermore, any information about the richness of a sample is beneficial to the optimal choice of parameter for digital normalization.

Our efforts to examine the effect of sequencing depth on the accuracy of beta diversity reveals that using a relatively small subset of the whole data set may get reasonably good result showing the separation of samples after clustering or ordination. However, how good the separation is seems to be related to the characteristics of samples and cannot be determined easily before starting the analysis. Thus, a potential approach should estimate beta analysis in an iterative way. We already know that using more data will benefit more accurate analysis or better separation for the purpose of comparing metagenomic samples. In such iteration procedure, pattern of separation can be monitored as more reads are loaded into the analysis and the procedure can be stopped as long as the pattern of the separation of samples is significant enough. This way, we may save lots of computational cost and still have enough information about the relationship between samples.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Afiahayati, K. Sato, and Y. Sakakibara. Metavelvet-sl: an extension of the velvet assembler to a de novo metagenomic assembler utilizing supervised learning. *DNA Res*, 22(1):69–77, Feb 2015.

[2] M. Albertsen, P. Hugenholtz, A. Skarshewski, K. L. Nielsen, G. W. Tyson, and P. H. Nielsen. Genome sequences of rare, uncultured bacteria obtained by differential coverage binning of multiple metagenomes. *Nat Biotechnol*, 31(6):533–8, Jun 2013.

[3] J. Alneberg, B. S. Bjarnason, I. de Bruijn, M. Schirmer, J. Quick, U. Z. Ijaz, L. Lahti, N. J. Loman, A. F. Andersson, and C. Quince. Binning metagenomic contigs by coverage and composition. *Nat Methods*, 11(11):1144–6, Nov 2014.

[4] P. Audano and F. Vannberg. Kanalyze: A fast versatile pipelined k-mer toolkit. *Bioinformatics: Advance Access published March 18, 2014*, page doi: 10.1093/bioinformatics/btu152, Mar 2014.

[5] B. J. M. Bohannan and J. Hughes. New approaches to analyzing microbial biodiversity data. *Curr Opin Microbiol*, 6(3):282–7, Jun 2003.

[6] M. Bonaldo, G. Lennon, and M. Soares. Normalization and subtraction: two approaches to facilitate gene discovery. *Genome Res*, 6(9):791–806, 1996.

[7] A. Brady and S. Salzberg. Phymmbl expanded: confidence scores, custom databases, parallelization and more. *Nat Methods*, 8(5):367, May 2011.

[8] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. *Internet mathematics*, 1(4):485–509, 2004.

[9] C. T. Brown. What does trinity's in silico normalization do? 2012.

[10] C. T. Brown, A. Howe, Q. Zhang, A. B. Pyrkosz, and T. H. Brom. A reference-free algorithm for computational normalization of shotgun sequencing data. *arXiv preprint*, 03 2012.

[11] J. Bunge. Estimating the number of species with catchall. *Pac Symp Biocomput*, pages 121–30, 2011.

[12] J. Butler, I. MacCallum, M. Kleber, I. A. Shlyakhter, M. K. Belmonte, E. S. Lander, C. Nusbaum, and D. B. Jaffe. Allpaths: de novo assembly of whole-genome shotgun microreads. *Genome Res*, 18(5):810–20, May 2008.

[13] J. G. Caporaso, J. Kuczynski, J. Stombaugh, K. Bittinger, F. D. Bushman, E. K. Costello, N. Fierer, A. G. Peña, J. K. Goodrich, J. I. Gordon, G. A. Huttley, S. T. Kelley, D. Knights, J. E. Koenig, R. E. Ley, C. A. Lozupone, D. McDonald, B. D. Muegge, M. Pirrung, J. Reeder, J. R. Sevinsky, P. J. Turnbaugh, W. A. Walters, J. Widmann, T. Yatsunenko, J. Zaneveld, and R. Knight. Qiime allows analysis of high-throughput community sequencing data. *Nat Methods*, 7(5):335–6, May 2010.

[14] M. Chaisson, P. Pevzner, and H. Tang. Fragment assembly with short reads. *Bioinformatics*, 20(13):2067–74, 2004.

[15] A. Chao. Nonparametric estimation of the number of classes in a population. *Scandinavian Journal of statistics*, pages 265–270, 1984.

[16] A. Chao. Estimating the population size for capture-recapture data with unequal catchability. *Biometrics*, 43(4):783–91, Dec 1987.

[17] A. Chao and M. C. Yang. Stopping rules and estimation for recapture debugging with unequal failure rates. *Biometrika*, 80(1):193–201, 1993.

[18] R. Chikhi and P. Medvedev. Informed and automated k-mer size selection for genome assembly. *Bioinformatics*, 30(1):31–7, Jan 2014.

[19] H. Chitsaz, J. Yee-Greenbaum, G. Tesler, M. Lombardo, C. Dupont, J. Badger, M. Novotny, D. Rusch, L. Fraser, N. Gormley, O. Schulz-Trieglaff, G. Smith, D. Evers, P. Pevzner, and R. Lasken. Efficient de novo assembly of single-cell bacterial genomes from short-read data sets. *Nat Biotechnol*, 29(10):915–21, 2011.

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

[21] R. Colwell. Estimates: statistical estimation of species richness and shared species from samples. 2004. *Consultado en: http://viceroy. eeb. uconn. edu/estimates*.

[22] R. K. Colwell, C. X. Mao, and J. Chang. Interpolating, extrapolating, and comparing incidence-based species accumulation curves. *Ecology*, 85(10):2717–2727, 2004.

[23] P. Compeau, P. Pevzner, and G. Tesler. How to apply de bruijn graphs to genome assembly. *Nat Biotechnol*, 29(11):987–91, 2011.

[24] T. Conway and A. Bromage. Succinct data structures for assembling large genomes. *Bioinformatics*, 27(4):479–86, 2011.

[25] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch, and its applications. *Journal of Algorithms*, 2004.

[26] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, Apr. 2005.

[27] G. Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *SDM*, pages 44–55, 2005.

[28] M. Crusoe, G. Edvenson, J. Fish, A. Howe, E. McDonald, J. Nahum, K. Nanlohy, H. Ortiz-Zuazaga, J. Pell, J. Simpson, C. Scott, R. Srinivasan, Q. Zhang, and C. T. Brown. *The khmer software package: enabling efficient sequence analysis*, 2014.

[29] T. P. Curtis, W. T. Sloan, and J. W. Scannell. Estimating prokaryotic diversity and its limits. *Proc Natl Acad Sci U S A*, 99(16):10494–9, Aug 2002.

[30] S. Deorowicz, A. Debudaj-Grabysz, and S. Grabowski. Disk-based k-mer counting on a pc. *BMC Bioinformatics*, 14(1):160, May 2013.

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

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

[33] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2):207–216, Dec. 2005.

[34] P. Flajolet, É. Fusy, O. Gandouet, and F. Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *DMTCS Proceedings*, (1), 2008.

[35] J. Gans, M. Wolinsky, and J. Dunbar. Computational improvements reveal great bacterial diversity and high metal toxicity in soil. *Science*, 309(5739):1387–90, Aug 2005.

[36] J. A. Gilbert and C. L. Dupont. Microbial metagenomics: beyond the genome. *Ann Rev Mar Sci*, 3:347–71, 2011.

[37] J. A. Gilbert, F. Meyer, D. Antonopoulos, P. Balaji, C. T. Brown, C. T. Brown, N. Desai, J. A. Eisen, D. Evers, D. Field, W. Feng, D. Huson, J. Jansson, R. Knight, J. Knight, E. Kolker, K. Konstantindis, J. Kostka, N. Kyrpides, R. Mackelprang, A. McHardy, C. Quince, J. Raes, A. Sczyrba, A. Shade, and R. Stevens. Meeting report: the terabase metagenomics workshop and the vision of an earth microbiome project. *Stand Genomic Sci*, 3(3):243–8, 2010.

[38] E. M. Glass, J. Wilkening, A. Wilke, D. Antonopoulos, and F. Meyer. Using the metagenomics rast server (mg-rast) for analyzing shotgun metagenomes. *Cold Spring Harb Protoc*, 2010(1):pdb.prot5368, Jan 2010.

[39] S. Gnerre, I. Maccallum, D. Przybylski, F. Ribeiro, J. Burton, B. Walker, T. Sharpe, G. Hall, T. Shea, S. Sykes, A. Berlin, D. Aird, M. Costello, R. Daza, L. Williams, R. Nicol, A. Gnirke, C. Nusbaum, E. Lander, and D. Jaffe. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc Natl Acad Sci U S A*, 108(4):1513–8, 2011.

[40] N. J. Gotelli and R. K. Colwell. Quantifying biodiversity: procedures and pitfalls in the measurement and comparison of species richness. *Ecology letters*, 4(4):379–391, 2001.

[41] M. Grabherr, B. Haas, M. Yassour, J. Levin, D. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, Z. Chen, E. Mauceli, N. Hacohen, A. Gnirke, N. Rhind, F. di Palma, B. Birren, C. Nusbaum, K. Lindblad-Toh, N. Friedman, and A. Regev. Full-length transcriptome assembly from rna-seq data without a reference genome. *Nat Biotechnol*, 29(7):644–52, 2011.

[42] M. G. Grabherr, B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng, Z. Chen, E. Mauceli, N. Hacohen, A. Gnirke, N. Rhind, F. di Palma, B. W. Birren, C. Nusbaum, K. Lindblad-Toh, N. Friedman, and A. Regev. Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat. Biotechnol.*, 29(7):644–652, Jul 2011.

[43] X. Guo, N. Yu, X. Ding, J. Wang, and Y. Pan. Dime: a novel framework for de novo metagenomic sequence assembly. *J Comput Biol*, 22(2):159–77, Feb 2015.

[44] B. J. Haas, A. Papanicolaou, M. Yassour, M. Grabherr, P. D. Blood, J. Bowden, M. B. Couger, D. Eccles, B. Li, M. Lieber, M. D. Macmanes, M. Ott, J. Orvis, N. Pochet, F. Strozzi, N. Weeks, R. Westerman, T. William, C. N. Dewey, R. Henschel, R. D. Leduc, N. Friedman, and A. Regev. De novo transcript sequence reconstruction from rna-seq using the trinity platform for reference generation and analysis. *Nat Protoc*, 8(8):1494–512, Aug 2013.

[45] B. Haegeman, J. Hamelin, J. Moriarty, P. Neal, J. Dushoff, and J. S. Weitz. Robust estimation of microbial diversity in theory and in practice. *ISME J*, 7(6):1092–101, Jun 2013.

[46] B. Haider, T.-H. Ahn, B. Bushnell, J. Chai, A. Copeland, and C. Pan. Omega: an overlap-graph de novo assembler for metagenomics. *Bioinformatics*, 30(19):2717–22, Oct 2014.

[47] J. Handelsman, M. R. Rondon, S. F. Brady, J. Clardy, and R. M. Goodman. Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chem Biol*, 5(10):R245–9, Oct 1998.

[48] M. Hess, A. Sczyrba, R. Egan, T.-W. Kim, H. Chokhawala, G. Schroth, S. Luo, D. S. Clark, F. Chen, T. Zhang, R. I. Mackie, L. A. Pennacchio, S. G. Tringe, A. Visel, T. Woyke, Z. Wang, and E. M. Rubin. Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science*, 331(6016):463–7, Jan 2011.

[49] M. O. Hill. Diversity and evenness: a unifying notation and its consequences. *Ecology*, 54(2):427–432, 1973.

[50] A. C. Howe, J. K. Jansson, S. A. Malfatti, S. G. Tringe, J. M. Tiedje, and C. T. Brown. Tackling soil diversity with the assembly of large, complex metagenomes. *Proc Natl Acad Sci U S A*, 111(13):4904–9, Apr 2014.

[51] A. C. Howe, J. K. Jansson, S. A. Malfatti, S. G. Tringe, J. M. Tiedje, and C. T. Brown. Tackling soil diversity with the assembly of large, complex metagenomes. *Proc Natl Acad Sci U S A*, 111(13):4904–9, Apr 2014.

[52] A. C. Howe, J. Pell, R. Canino-Koning, R. Mackelprang, S. Tringe, J. Jansson, J. M. Tiedje, and C. T. Brown. Illumina sequencing artifacts revealed by connectivity analysis of metagenomic datasets. *PLoS ONE*, -.

[53] S. Howorka, S. Cheley, and H. Bayley. Sequence-specific detection of individual DNA strands using engineered nanopores. *Nat. Biotechnol.*, 19(7):636–639, Jul 2001.

[54] Human Microbiome Jumpstart Reference Strains Consortium, K. E. Nelson, G. M. Weinstock, S. K. Highlander, K. C. Worley, H. H. Creasy, J. R. Wortman, D. B. Rusch, M. Mitreva, E. Sodergren, A. T. Chinwalla, M. Feldgarden, D. Gevers, B. J. Haas, R. Madupu, D. V. Ward, B. W. Birren, R. A. Gibbs, B. Methe, J. F. Petrosino, R. L. Strausberg, G. G. Sutton, O. R. White, R. K. Wilson, S. Durkin, M. G. Giglio, S. Gujja, C. Howarth, C. D. Kodira, N. Kyrpides, T. Mehta, D. M. Muzny, M. Pearson, K. Pepin, A. Pati, X. Qin, C. Yandava, Q. Zeng, L. Zhang, A. M. Berlin, L. Chen, T. A. Hepburn, J. Johnson, J. McCorrison, J. Miller, P. Minx, C. Nusbaum, C. Russ, S. M. Sykes, C. M. Tomlinson, S. Young, W. C. Warren, J. Badger, J. Crabtree, V. M. Markowitz, J. Orvis, A. Cree, S. Ferriera, L. L. Fulton, R. S. Fulton, M. Gillis, L. D. Hemphill, V. Joshi, C. Kovar, M. Torralba, K. A. Wetterstrand, A. Abouellleil, A. M. Wollam, C. J. Buhay, Y. Ding, S. Dugan, M. G. FitzGerald, M. Holder, J. Hostetler, S. W. Clifton, E. Allen-Vercoe, A. M. Earl, C. N. Farmer, K. Liolios, M. G. Surette, Q. Xu, C. Pohl, K. Wilczek-Boney, and D. Zhu. A catalog of reference genomes from the human microbiome. *Science*, 328(5981):994–9, May 2010.

[55] Human Microbiome Project Consortium. Structure, function and diversity of the healthy human microbiome. *Nature*, 486(7402):207–14, Jun 2012.

[56] D. H. Huson, A. F. Auch, J. Qi, and S. C. Schuster. Megan analysis of metagenomic data. *Genome Res*, 17(3):377–86, Mar 2007.

[57] M. Imelfort, D. Parks, B. J. Woodcroft, P. Dennis, P. Hugenholtz, and G. W. Tyson. Groopm: an automated tool for the recovery of population genomes from related metagenomes. *PeerJ*, 2:e603, 2014.

[58] Z. Iqbal, M. Caccamo, I. Turner, P. Flicek, and G. McVean. De novo assembly and genotyping of variants using colored de bruijn graphs. *Nat Genet*, 44(2):226–32, Feb 2012.

[59] F. H. Karlsson, V. Tremaroli, I. Nookaew, G. Bergström, C. J. Behre, B. Fagerberg, J. Nielsen, and F. Bäckhed. Gut metagenome in european women with normal, impaired and diabetic glucose control. *Nature*, 498(7452):99–103, Jun 2013.

[60] A. L. Kau, P. P. Ahern, N. W. Griffin, A. L. Goodman, and J. I. Gordon. Human nutrition, the gut microbiome and the immune system. *Nature*, 474(7351):327–36, Jun 2011.

[61] K. P. Keegan, W. L. Trimble, J. Wilkening, A. Wilke, T. Harrison, M. D'Souza, and F. Meyer. A platform-independent method for detecting errors in metagenomic sequencing data: DRISEE. *PLoS Comput. Biol.*, 8(6):e1002541, 2012.

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

[63] S. Kurtz, A. Narechania, J. C. Stein, and D. Ware. A new method to compute k-mer frequencies and its application to annotate large repetitive plant genomes. *BMC Genomics*, 9:517, 2008.

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

[65] S.-M. Lee and A. Chao. Estimating population size via sample coverage for closed capture-recapture models. *Biometrics*, pages 88–97, 1994.

[66] D. Li, C.-M. Liu, R. Luo, K. Sadakane, and T.-W. Lam. Megahit: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*, Jan 2015.

[67] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, and J. Wang. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res*, 20(2):265–72, Feb 2010.

[68] X. Li and M. S. Waterman. Estimating the repeat structure and length of dna sequences using l-tuples. *Genome Res*, 13(8):1916–22, Aug 2003.

[69] Y. Li, Y. Hu, L. Bolund, and J. Wang. State of the art de novo assembly of human genomes from massively parallel sequencing data. *Hum Genomics*, 4(4):271–7, 2010.

[70] M. Loreau, S. Naeem, P. Inchausti, J. Bengtsson, J. P. Grime, A. Hector, D. U. Hooper, M. A. Huston, D. Raffaelli, B. Schmid, D. Tilman, and D. A. Wardle. Biodiversity and ecosystem functioning: current knowledge and future challenges. *Science*, 294(5543):804–8, Oct 2001.

[71] E. K. Lowe, B. Swalla, and C. Brown. Evaluating a lightweight transcriptome assembly pipeline on two closely related ascidian species. *PeerJ Preprints*, 2, 2014.

[72] C. Luo, D. Tsementzi, N. Kyrpides, T. Read, and K. T. Konstantinidis. Direct comparisons of illumina vs. roche 454 sequencing technologies on the same microbial community dna sample. *PLoS One*, 7(2):e30087, 2012.

[73] W. Luo, M. S. Friedman, K. Shedden, K. D. Hankenson, and P. J. Woolf. Gage: generally applicable gene set enrichment for pathway analysis. *BMC Bioinformatics*, 10:161, 2009.

[74] A. E. Magurran and B. J. McGill. *Biological diversity: frontiers in measurement and assessment*, volume 12. Oxford University Press Oxford, 2011.

[75] N. Maillet, G. Collet, T. Vannier, D. Lavenier, and P. Peterlongo. Commet: Comparing and combining multiple metagenomic datasets. In H. J. Zheng, W. Dubitzky, X. Hu, J. Hao, D. P. Berrar, K. Cho, Y. Wang, and D. R. Gilbert, editors, *2014 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2014, Belfast, United Kingdom, November 2-5, 2014*, pages 94–98. IEEE, 2014.

[76] N. Maillet, C. Lemaitre, R. Chikhi, D. Lavenier, and P. Peterlongo. Compareads: comparing huge metagenomic experiments. *BMC Bioinformatics*, 13 Suppl 19:S10, 2012.

[77] N. Mantel. The detection of disease clustering and a generalized regression approach. *Cancer Res*, 27(2):209–20, Feb 1967.

[78] G. Marçais and C. Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, 2011.

[79] V. M. Markowitz, I.-M. A. Chen, K. Chu, E. Szeto, K. Palaniappan, Y. Grechkin, A. Ratner, B. Jacob, A. Pati, M. Huntemann, K. Liolios, I. Pagani, I. Anderson, K. Mavromatis, N. N. Ivanova, and N. C. Kyrpides. Img/m: the integrated metagenome data management and comparative analysis system. *Nucleic Acids Res*, 40(Database issue):D123–9, Jan 2012.

[80] O. U. Mason, T. C. Hazen, S. Borglin, P. S. G. Chain, E. A. Dubinsky, J. L. Fortney, J. Han, H.-Y. N. Holman, J. Hultman, R. Lamendella, R. Mackelprang, S. Malfatti, L. M. Tom, S. G. Tringe, T. Woyke, J. Zhou, E. M. Rubin, and J. K. Jansson. Metagenome, metatranscriptome and single-cell sequencing reveal microbial response to deepwater horizon oil spill. *ISME J*, 6(9):1715–27, Sep 2012.

[81] R. M. May. How many species are there on earth? *Science*, 241(4872):1441–9, Sep 1988.

[82] E. McDonald and C. T. Brown. Working with big data in bioinformatics. In T. Armstrong, editor, *The Performance of Open Source Applications*, chapter 12, page 151. lulu.com, 2013.

[83] P. Medvedev, E. Scott, B. Kakaradov, and P. Pevzner. Error correction of high-throughput sequencing datasets with non-uniform coverage. *Bioinformatics*, 27(13):i137–41, Jul 2011.

[84] P. Melsted and B. V. Halldorsson. KmerStream: streaming algorithms for k-mer abundance estimation. *Bioinformatics*, 30(24):3541–3547, Dec 2014.

[85] P. Melsted and J. K. Pritchard. Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC bioinformatics*, 12:333, Jan. 2011.

[86] M. Metzker. Sequencing technologies - the next generation. *Nat Rev Genet*, 11(1):31–46, 2010.

[87] J. Miller, S. Koren, and G. Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–27, 2010.

[88] A. E. Minoche, J. C. Dohm, and H. Himmelbauer. Evaluation of genomic high-throughput sequencing data generated on illumina hiseq and genome analyzer systems. *Genome Biol*, 12(11):R112, 2011.

[89] C. Mora, D. P. Tittensor, S. Adl, A. G. B. Simpson, and B. Worm. How many species are there on earth and in the ocean? *PLoS Biol*, 9(8):e1001127, Aug 2011.

[90] X. C. Morgan, T. L. Tickle, H. Sokol, D. Gevers, K. L. Devaney, D. V. Ward, J. A. Reyes, S. A. Shah, N. LeLeiko, S. B. Snapper, A. Bousvaros, J. Korzenik, B. E. Sands, R. J. Xavier, and C. Huttenhower. Dysfunction of the intestinal microbiome in inflammatory bowel disease and treatment. *Genome Biol*, 13(9):R79, 2012.

[91] E. K. Morris, T. Caruso, F. Buscot, M. Fischer, C. Hancock, T. S. Maier, T. Meiners, C. Müller, E. Obermaier, D. Prati, S. A. Socher, I. Sonnemann, N. Wäschke, T. Wubet, S. Wurst, and M. C. Rillig. Choosing and using diversity indices: insights for ecological applications from the german biodiversity exploratories. *Ecol Evol*, 4(18):3514–24, Sep 2014.

[92] S. Muthukrishnan. *Data streams: algorithms and applications*. Foundations and trends in theoretical computer science. Now Publishers, 2005.

[93] T. Namiki, T. Hachiya, H. Tanaka, and Y. Sakakibara. Metavelvet: an extension of velvet assembler to de novo metagenome assembly from short sequence reads. *Nucleic Acids Res*, 40(20):e155, Nov 2012.

[94] H. B. Nielsen, M. Almeida, A. S. Juncker, S. Rasmussen, J. Li, S. Sunagawa, D. R. Plichta, L. Gautier, A. G. Pedersen, E. Le Chatelier, E. Pelletier, I. Bonde, T. Nielsen, C. Manichanh, M. Arumugam, J.-M. Batto, M. B. Quintanilha Dos Santos, N. Blom, N. Borruel, K. S. Burgdorf, F. Boumezbeur, F. Casellas, J. Doré, P. Dworzynski, F. Guarner, T. Hansen, F. Hildebrand, R. S. Kaas, S. Kennedy, K. Kristiansen, J. R. Kultima, P. Léonard, F. Levenez, O. Lund, B. Moumen, D. Le Paslier, N. Pons, O. Pedersen, E. Prifti, J. Qin, J. Raes, S. Sørensen, J. Tap, S. Tims, D. W. Ussery, T. Yamada, MetaHIT Consortium, P. Renault, T. Sicheritz-Ponten, P. Bork, J. Wang, S. Brunak, S. D. Ehrlich, and MetaHIT Consortium. Identification and assembly of genomes and genetic elements in complex metagenomic samples without using reference genomes. *Nat Biotechnol*, 32(8):822–8, Aug 2014.

[95] K. R. Patil, L. Roune, and A. C. McHardy. The phylopythias web server for taxonomic assignment of metagenome sequences. *PLoS One*, 7(6):e38581, 2012.

[96] J. Pell, A. Hintze, R. Canino-Koning, A. Howe, J. M. Tiedje, and C. T. Brown. Scaling metagenome sequence assembly with probabilistic de bruijn graphs. *Proc Natl Acad Sci U S A*, 109(33):13272–7, Aug 2012.

[97] Y. Peng, H. Leung, S. Yiu, and F. Chin. Meta-idba: a de novo assembler for metagenomic data. *Bioinformatics*, 27(13):i94–101, 2011.

[98] F. Pérez and B. Granger. Ipython: A system for interactive scientific computing. *Computing in Science Engineering*, 9(3):21–29, 2007.

[99] P. A. Pevzner, H. Tang, and M. S. Waterman. An eulerian path approach to dna fragment assembly. *Proc Natl Acad Sci U S A*, 98(17):9748–53, Aug 2001.

[100] J. Pickrell, A. Pai, Y. Gilad, and J. Pritchard. Noisy splicing drives mrna isoform diversity in human cells. *PLoS Genet*, 6(12):e1001236, 2010.

[101] A. Pinho, D. Pratas, and S. Garcia. Green: a tool for efficient compression of genome resequencing data. *Nucleic Acids Res*, 40(4):e27, 2012.

[102] J. Qin, R. Li, J. Raes, M. Arumugam, K. S. Burgdorf, C. Manichanh, T. Nielsen, N. Pons, F. Levenez, T. Yamada, D. R. Mende, J. Li, J. Xu, S. Li, D. Li, J. Cao, B. Wang, H. Liang, H. Zheng, Y. Xie, J. Tap, P. Lepage, M. Bertalan, J.-M. Batto, T. Hansen, D. Le Paslier, A. Linneberg, H. B. Nielsen, E. Pelletier, P. Renault, T. Sicheritz-Ponten, K. Turner, H. Zhu, C. Yu, S. Li, M. Jian, Y. Zhou, Y. Li, X. Zhang, S. Li, N. Qin, H. Yang, J. Wang, S. Brunak, J. Doré, F. Guarner, K. Kristiansen, O. Pedersen, J. Parkhill, J. Weissenbach, MetaHIT Consortium, P. Bork,

S. D. Ehrlich, and J. Wang. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285):59–65, Mar 2010.

[103] J. Qin, R. Li, J. Raes, M. Arumugam, K. S. Burgdorf, C. Manichanh, T. Nielsen, N. Pons, F. Levenez, T. Yamada, D. R. Mende, J. Li, J. Xu, S. S. S. Li, D. Li, J. Cao, B. Wang, H. Liang, H. Zheng, Y. Xie, J. Tap, P. Lepage, M. Bertalan, J.-M. Batto, T. Hansen, D. Le Paslier, A. Linneberg, H. B. r. Nielsen, E. Pelletier, P. Renault, T. Sicheritz-Ponten, K. Turner, H. Zhu, C. Yu, M. Jian, Y. Zhou, Y. Li, X. Zhang, N. Qin, H. Yang, J. J. Wang, S. r. Brunak, J. Doré, F. Guarner, K. Kristiansen, O. Pedersen, J. Parkhill, J. Weissenbach, P. Bork, and S. D. Ehrlich. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285):59–65, 2010.

[104] J. Qin, Y. Li, Z. Cai, S. Li, J. Zhu, F. Zhang, S. Liang, W. Zhang, Y. Guan, D. Shen, Y. Peng, D. Zhang, Z. Jie, W. Wu, Y. Qin, W. Xue, J. Li, L. Han, D. Lu, P. Wu, Y. Dai, X. Sun, Z. Li, A. Tang, S. Zhong, X. Li, W. Chen, R. Xu, M. Wang, Q. Feng, M. Gong, J. Yu, Y. Zhang, M. Zhang, T. Hansen, G. Sanchez, J. Raes, G. Falony, S. Okuda, M. Almeida, E. LeChatelier, P. Renault, N. Pons, J.-M. Batto, Z. Zhang, H. Chen, R. Yang, W. Zheng, S. Li, H. Yang, J. Wang, S. D. Ehrlich, R. Nielsen, O. Pedersen, K. Kristiansen, and J. Wang. A metagenome-wide association study of gut microbiota in type 2 diabetes. *Nature*, 490(7418):55–60, Oct 2012.

[105] C. Quince, T. P. Curtis, and W. T. Sloan. The rational exploration of microbial diversity. *ISME J*, 2(10):997–1006, Oct 2008.

[106] G. Rizk, D. Lavenier, and R. Chikhi. Dsk: k-mer counting with very low memory usage. *Bioinformatics*, 29(5):652–3, Mar 2013.

[107] L. M. Rodriguez-R and K. T. Konstantinidis. Nonpareil: A redundancy-based approach to assess the level of coverage in metagenomic datasets. *Bioinformatics*, Oct 2013.

[108] L. M. Rodriguez-R and K. T. Konstantinidis. Nonpareil: a redundancy-based approach to assess the level of coverage in metagenomic datasets. *Bioinformatics*, 30(5):629–35, Mar 2014.

[109] L. F. W. Roesch, R. R. Fulthorpe, A. Riva, G. Casella, A. K. M. Hadwin, A. D. Kent, S. H. Daroub, F. A. O. Camargo, W. G. Farmerie, and E. W. Triplett. Pyrosequencing enumerates and contrasts soil microbial diversity. *ISME J*, 1(4):283–90, Aug 2007.

[110] G. L. Rosen, E. R. Reichenberger, and A. M. Rosenfeld. Nbc: the naive bayes classification tool webserver for taxonomic classification of metagenomic reads. *Bioinformatics*, 27(1):127–9, Jan 2011.

[111] R. S. Roy, D. Bhattacharya, and A. Schliep. Turtle: Identifying frequent k-mers with cache-efficient algorithms. *Bioinformatics: Advance Access published March 10, 2014*, page doi: 10.1093/bioinformatics/btu132, Apr 2014.

[112] F. Rusu and A. Dobra. Sketches for size of join estimation. *ACM Transactions on Database Systems*, 33(3):1–46, Aug. 2008.

[113] D. C. Savage. Microbial ecology of the gastrointestinal tract. *Annu Rev Microbiol*, 31:107–33, 1977.

[114] P. D. Schloss and J. Handelsman. Introducing dotur, a computer program for defining operational taxonomic units and estimating species richness. *Appl Environ Microbiol*, 71(3):1501–6, Mar 2005.

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

[116] P. D. Schloss, S. L. Westcott, T. Ryabin, J. R. Hall, M. Hartmann, E. B. Hollister, R. A. Lesniewski, B. B. Oakley, D. H. Parks, C. J. Robinson, J. W. Sahl, B. Stres, G. G. Thallinger, D. J. Van Horn, and C. F. Weber. Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl Environ Microbiol*, 75(23):7537–41, Dec 2009.

[117] T. M. Schmidt, E. F. DeLong, and N. R. Pace. Analysis of a marine picoplankton community by 16s rrna gene cloning and sequencing. *J Bacteriol*, 173(14):4371–8, Jul 1991.

[118] M. Schulz, D. Zerbino, M. Vingron, and E. Birney. Oases: robust de novo rna-seq assembly across the dynamic range of expression levels. *Bioinformatics*, 28(8):1086–92, 2012.

[119] N. Segata, L. Waldron, A. Ballarini, V. Narasimhan, O. Jousson, and C. Huttenhower. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nat Methods*, 9(8):811–4, Aug 2012.

[120] M. Shakya, C. Quince, J. H. Campbell, Z. K. Yang, C. W. Schadt, and M. Podar. Comparative metagenomic and rRNA microbial diversity characterization using archaeal and bacterial synthetic communities. *Environ. Microbiol.*, 15(6):1882–1899, Jun 2013.

[121] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[122] E. H. Simpson. Measurement of diversity. *Nature*, 1949.

[123] J. Simpson and R. Durbin. Efficient construction of an assembly string graph using the fm-index. *Bioinformatics*, 26(12):i367–73, 2010.

[124] J. Simpson and R. Durbin. Efficient de novo assembly of large genomes using compressed data structures. *Genome Res*, 22(3):549–56, 2012.

[125] J. Simpson, K. Wong, S. Jackman, J. Schein, S. Jones, and I. Birol. Abyss: a parallel assembler for short read sequence data. *Genome Res*, 19(6):1117–23, 2009.

[126] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. M. Jones, and I. Birol. Abyss: a parallel assembler for short read sequence data. *Genome Res*, 19(6):1117–23, Jun 2009.

[127] M. Soares, M. Bonaldo, P. Jelene, L. Su, L. Lawton, and A. Efstratiadis. Construction and characterization of a normalized cdna library. *Proc Natl Acad Sci U S A*, 91(20):9228–32, 1994.

[128] M. L. Sogin, H. G. Morrison, J. A. Huber, D. Mark Welch, S. M. Huse, P. R. Neal, J. M. Arrieta, and G. J. Herndl. Microbial diversity in the deep sea and the underexplored "rare biosphere". *Proc Natl Acad Sci U S A*, 103(32):12115–20, Aug 2006.

[129] L. Song, L. Florea, and B. Langmead. Lighter: fast and memory-efficient sequencing error correction without counting. *Genome Biol.*, 15(11):509, 2014.

[130] C. Spits, C. L. Caignec, M. D. Rycke, L. V. Haute, A. V. Steirteghem, I. Liebaers, and K. Sermon. Whole-genome multiple displacement amplification from single cells. *Nat Protoc*, 1(4):1965–70, 2006.

[131] E. Stackebrandt, W. Frederiksen, G. M. Garrity, P. A. D. Grimont, P. Kämpfer, M. C. J. Maiden, X. Nesme, R. Rosselló-Mora, J. Swings, H. G. Trüper, L. Vauterin, A. C. Ward, and W. B. Whitman. Report of the ad hoc committee for the re-evaluation of the species definition in bacteriology. *Int J Syst Evol Microbiol*, 52(Pt 3):1043–7, May 2002.

[132] L. Stein. The case for cloud computing in genome informatics. *Genome Biol*, 11(5):207, 2010.

[133] H. Teeling, J. Waldmann, T. Lombardot, M. Bauer, and F. O. Glöckner. Tetra: a web-service and a stand-alone program for the analysis and comparison of tetranucleotide usage patterns in dna sequences. *BMC Bioinformatics*, 5:163, Oct 2004.

[134] P. J. Turnbaugh, M. Hamady, T. Yatsunenko, B. L. Cantarel, A. Duncan, R. E. Ley, M. L. Sogin, W. J. Jones, B. A. Roe, J. P. Affourtit, M. Egholm, B. Henrissat, A. C. Heath, R. Knight, and J. I. Gordon. A core gut microbiome in obese and lean twins. *Nature*, 457(7228):480–4, Jan 2009.

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

[136] S. Tzahor, D. Man-Aharonovich, B. C. Kirkup, T. Yogev, I. Berman-Frank, M. F. Polz, O. Béjà, and Y. Mandel-Gutfreund. A supervised learning approach for taxonomic classification of core-photosystem-ii genes and transcripts in the marine environment. *BMC Genomics*, 10:229, 2009.

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

[138] Q. Wang, G. M. Garrity, J. M. Tiedje, and J. R. Cole. Naive bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy. *Appl Environ Microbiol*, 73(16):5261–7, Aug 2007.

[139] W. B. Whitman, D. C. Coleman, and W. J. Wiebe. Prokaryotes: the unseen majority. *Proc Natl Acad Sci U S A*, 95(12):6578–83, Jun 1998.

[140] D. M. Wilkinson. The disturbing history of intermediate disturbance. *Oikos*, pages 145–147, 1999.

[141] D. Wu, P. Hugenholtz, K. Mavromatis, R. Pukall, E. Dalin, N. N. Ivanova, V. Kunin, L. Goodwin, M. Wu, B. J. Tindall, S. D. Hooper, A. Pati, A. Lykidis, S. Spring, I. J. Anderson, P. D'haeseleer, A. Zemla, M. Singer, A. Lapidus, M. Nolan, A. Copeland, C. Han, F. Chen, J.-F. Cheng, S. Lucas, C. Kerfeld, E. Lang, S. Gronow, P. Chain, D. Bruce, E. M. Rubin, N. C. Kyrpides, H.-P. Klenk, and J. A. Eisen. A phylogeny-driven genomic encyclopaedia of bacteria and archaea. *Nature*, 462(7276):1056–60, Dec 2009.

[142] M. Wu and A. J. Scott. Phylogenomic analysis of bacterial and archaeal sequences with amphora2. *Bioinformatics*, 28(7):1033–4, Apr 2012.

[143] Z. Xu, D. Malmer, M. G. I. Langille, S. F. Way, and R. Knight. Which is more important for classifying microbial communities: who's there or what they can do? *ISME J*, 8(12):2357–9, Dec 2014.

[144] I. Zarraonaindia, D. P. Smith, and J. A. Gilbert. Beyond the genome: community-level analysis of the microbial world. *Biol Philos*, 28(2):261–282, Mar 2013.

[145] D. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Res*, 18(5):821–9, 2008.

[146] D. R. Zerbino and E. Birney. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Res*, 18(5):821–9, May 2008.

[147] Q. Zhang, S. Awad, and C. T. Brown. Crossing the streams: a framework for streaming. https://peerj.com/preprints/890/, 2015.

[148] Q. Zhang, J. Pell, R. Canino-Koning, A. C. Howe, and C. T. Brown. These are not the k-mers you are looking for: efficient online k-mer counting using a probabilistic data structure. *PLoS ONE*, 7(9), 2014.