WGSQUIKR: FAST WHOLE-GENOME SHOTGUN METAGENOMIC CLASSIFICATION

DAVID KOSLICKI[1]*, SIMON FOUCART[2], GAIL ROSEN[3]

[1]*Mathematics Department, Oregon State University, Corvallis, OR 97330*

[2]*Department of Mathematics, University of Georgia, Athens, GA 30602*

[3]*Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104*

Abstract. With the decrease in cost and increase in output of whole-genome shotgun technologies, many metagenomic studies are utilizing this approach in lieu of the more traditional 16S rRNA amplicon technique. Due to the large number of relatively short reads output from whole-genome shotgun technologies, there is a need for fast and accurate short-read OTU classifiers. While there are relatively fast and accurate algorithms available, such as MetaPhlAn, MetaPhyler, PhyloPythiaS, and PhymmBL, these algorithms still classify samples in a read-by-read fashion and so execution times can range from hours to days on large datasets.

We introduce WGSQuikr, a reconstruction method which can compute a vector of taxonomic assignments and their proportions in the sample with remarkable speed and accuracy. We demonstrate on simulated data that WGSQuikr is typically more accurate and up to an order of magnitude faster than the aforementioned classification algorithms. We also verify the utility of WGSQuikr on real biological data in the form of a mock community.

WGSQuikr is a Whole-Genome Shotgun QUadratic, Iterative, K-mer based Reconstruction method which extends the previously introduced 16S rRNA-based algorithm Quikr. A MATLAB implementation of WGSQuikr is available at:

`http://sourceforge.net/projects/wgsquikr`.

1. Introduction

While 16S rRNA amplicon sequencing is a popular approach to reconstructing the taxonomic composition of a bacterial community, there are some limitations to this approach. For example, multiple copies of 16S rRNA genes in a single organism and nearly identical 16S rRNA genes in other species can both lead to mis-estimates of bacterial compositions [6]. These and other considerations have contributed to an increased usage of whole-genome shotgun (WGS) sequencing to analyze microbial communities. However, the large amount of short reads resulting from WGS methods (ranging from 70 million 200bp-length reads for Ion Torrent's Proton Torrent sequencer, to 3 billion 100bp-length reads for Illumina's HiSeq, to 15 million 36bp-length reads for Illumina's MiSeq) necessitates fast and accurate algorithms to process these large amounts of data. Current methods, while relatively accurate, can still take from 8 hours (MetaPhyler [17]) to 4 days (PhymmBL [4]) to analyze a relatively small dataset of 70 thousand 300bp reads [17].

We introduce a method that extends the previously introduced 16SrRNA-based algorithm Quikr [15], allowing for the accurate analysis of very large whole-genome shotgun datasets (billions of

Date: February 19, 2014.

*Corresponding Author: `david.koslicki@math.oregonstate.edu`.

S.F and G.R. supported by NSF grant DMS-1120622.

reads) on a laptop computer in under an hour. This is facilitated by leveraging ideas from compressive sensing to reconstruct all taxonomic relative abundances of a bacterial community simultaneously (as opposed to read-by-read classification). Beyond significant speed improvements, we demonstrate on simulated data that this method has, on average, better reconstruction fidelity than any other technique to date, even down to the genus level.

Briefly, our method first measures the frequency of k-mers (for a fixed $k \sim 7$) in a database of known bacterial genomes, calculates the frequency of k-mers in a given sample, and then reconstructs the concentrations of the bacteria in the sample by solving a system of linear equations under a sparsity assumption. To solve this system, we employ MATLAB's [1] iterative implementation of nonnegative least squares and hence we refer to this method as *WGSQuikr*: Whole-Genome Shotgun QUadratic, Iterative, K-mer based Reconstruction. We point out that WGSQuikr has not yet been optimized for performance but still demonstrates a significant speed improvement over existing methods.

2. Methods

2.1. **k-mer Training Matrix.** The training step consists of converting an input database of whole bacterial genomic sequences (with their associated plasmid sequences) into a *k-mer training matrix*. For a fixed k-mer size, we calculate the frequency of each k-mer in each database sequence. Hence, given a database of genome sequences $D = \{d_1, \ldots, d_M\}$, the $(i, j)^{\text{th}}$ entry of the k-mer training matrix $A^{(k)}$ is the frequency of the i^{th} k-mer (in lexicographic order) in the j^{th} sequence d_j.

Herein, we consider a single, manually curated database D consisting of 1,401 bacterial genomes and 1,082 plasmids, resulting in 2,483 unique sequences, which along with their taxonomic information, were retrieved from NCBI [23] in October, 2012. The bacterial sequences in this database cover 1,109 species and 614 genera.

2.2. **Sample k-mer Frequencies.** Given a sample dataset of WGS reads, we orient all the reads in the forward direction, and then calculate the frequency of all k-mers in the entire sample. We refer to this vector $s^{(k)}$ as the *sample k-mer frequency vector*.

2.3. **Sparsity Promoting Quadratic Optimization.** We assume that the given environmental sample only contains bacteria that exist in the database $D = \{d_1, \ldots, d_M\}$ being utilized. Hence we can represent the composition of the sample as a vector $x \in \mathbb{R}^M$ with nonnegative entries summing to one (i.e. a probability vector) where x_i is the concentration of the organism with genome d_i. However, as will be demonstrated in subsection 3.4, WGSQuikr still performs adequately when the sample *does* contain novel bacteria not in the database being utilized.

The problem at hand is then to reconstruct the bacterial concentrations x by solving the linear system

$$A^{(k)} x = s^{(k)}. \tag{2.1}$$

Equation (2.1) is solved by using a sparsity-promoting optimization procedure motivated by techniques used in the compressive sensing literature. Sparsity is emphasized since it is reasonable to assume that relatively few bacteria from the database D are actually present in the given sample. We use a variant of nonnegative basis pursuit denoising [10, 7] which reduces to a nonnegative least squares problem. Unlike the 16S rRNA version of Quikr [15], WGSQuikr experiences no convergence issues thanks to the inclusion of an adaptive choice of a regularization parameter which is calculated individually for each dataset. The details regarding this procedure are contained in Appendix B.4.

2.4. **Reconstruction Metrics.** We denote the *actual* and *predicted* concentrations of the bacteria as probability vectors x and x^* respectively. The reconstruction metric primarily employed herein is the ℓ_1 distance between x and x^*: $||x - x^*||_{\ell_1}$. This quantity takes values between 0 and 2 (with perfect reconstruction being $||x - x^*||_{\ell_1} = 0$) and is commonly referred to as "total error" (as it is the total of the absolute errors). The term *reconstruction fidelity* will be used to communicate generically how well x^* approximates x. We will mainly be concerned with reconstruction fidelity down to the genus level since the assumption given in subsection 2.3 indicates that WGSQuikr is applicable in situations where the given metagenomic sample does not contain (too distantly related) novel taxonomic units absent from the training database. This is more likely to be the case at the genus level than at the species or strain level.

2.5. **Simulated Data.** To test the performance of the WGSQuikr method, the shotgun read simulator Grinder [2] was used to generate 720 simulated WGS datasets totaling over 1 billion reads. These datasets have a wide range of differing characteristics designed to replicate a range of technologies in a variety of conditions (for example: differing species abundances, read coverages, read lengths, error models, abundance models, etc.). The particular parameter values can be found in Appendix A. We verified that our results do not depend on the randomly chosen bacterial species in each dataset by re-running each simulation 5 times and observing that the results in section 3 do not change.

2.6. **Mock Communities.** To benchmark the Quikr method on real biological data, we examined the "even" mock microbial community (NCBI SRR172902) developed by the Human Microbiome Project [13]. This community contains known concentrations of bacteria from 21 different organisms that span a diverse range of properties (GC content, genome size, etc.).

3. Results

There are many whole-genome shotgun metagenomic classifiers that WGSQuikr can be compared to. A selection includes NBC [27, 22], Phymm [4], PhymmBl [4, 5], MetaPhyler [17], RITA [18], PhyloPythiaS [20], MetaPhlAn [24], Genometa [9] and MetaID [25].Typically, these algorithms classify a sample in a read-by-read fashion against a known database. Briefly, NBC accomplishes this in a Bayesian framework utilizing k-mer counts. Phymm and PhymmBL use interpolated Markov models to characterize variable-length oligonucleotides. MetaPhyler and MetaPhlAn use clade-identifying marker genes. Genometa and RITA are BLAST-based techniques, and MetaID uses large common and unique k-mers to classify reads. It has been shown [24, 9, 17] that the methods roughly rank in terms of increasing execution time as: MetaPhlAn, MetaPhyler, PhyloPythiaS, Phymm, PhymmBL, NBC, Genometa, and RITA (MetaID [25] details no run-time data).

WGSQuikr differs from all of these methods as it classifies an entire dataset simultaneously rather than in a read-by-read fashion. Furthermore, the other k-mer based techniques typically use k-mers for $k \geq 12$, whereas WGSQuikr uses $k \sim 7$. As WGSQuikr is intended to be used as a fast classification method at a taxonomic level in which few novel taxa appear, we choose to compare to the two fastest methods available: MetaPhlAn and MetaPhyler. WGSQuikr and these two algorithms will be evaluated on all simulated data and the mock community using the default parameters.

3.1. **Speed Comparison.** Throughout the following, we fixed the k-mer size at $k = 7$. We observed the general trend that the algorithm execution time increased exponentially as a function of k, while the ℓ_1-error decreased roughly linearly. We chose $k = 7$ as this provided a reasonable tradeoff between fast execution time and low reconstruction error. Figure 1 shows a log-log plot of the execution time for WGSQuikr, MetaPhyler, and MetaPhlAn on datasets ranging from 100 reads to 10 million reads of 75bp in length. Figure 1 includes the time required to form the sample k-mer frequency vector for the WGSQuikr algorithm. As $k = 7$ is relatively small, the time required to

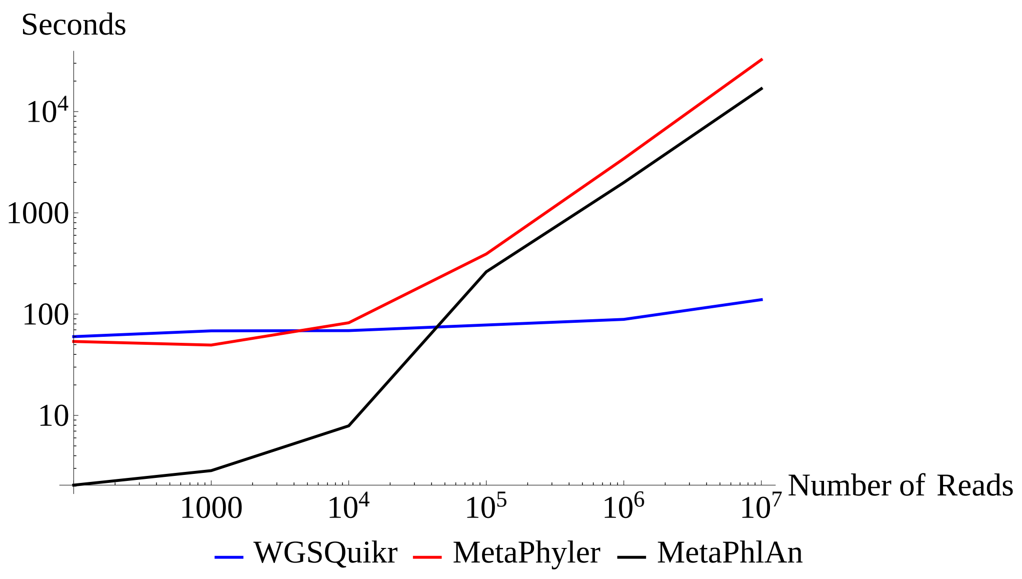

FIGURE 1. Log-log plot of number of reads versus execution time (seconds) for WGSQuikr, MetaPhyler and MetaPhlAn.

form this vector is negligible (e.g. for a sample with 1M 75bp reads, it takes less than 5 seconds to form the sample 7-mer frequency vector). The execution time is nearly constant for WGSQuikr to solve (2.1) via the algorithm detailed in B.2 and B.4. This is due to the algorithm taking as input the k-mer frequency vector, whose size depends only on k, not the size of the given dataset. This also explains the reason for the significant speed improvement of WGSQuikr: the entire sample is classified simultaneously, as opposed to in a read-by-read fashion such as with MetaPhyler or MetaPhlAn.

Figure 2 shows a box-and-whisker plot of the execution time for WGSQuikr, MetaPhyler, and MetaPhlAn on the simulated datasets described in subsection 2.5. Note the significant improvement in speed: the average execution time of WGSQuikr is over 6 times faster than the average MetaPhyler execution time. For the larger datasets (5M reads), WGSQuikr is on average 27 times faster than MetaPhyler and 5 times faster than MetaPhlAn.

3.2. **Simulated Data Results.**

3.2.1. *Reconstruction Error.* We evaluated the ℓ_1-error at the genus level on the simulated datasets and summarize the mean ℓ_1-error at the genus level in table 1. The histogram in figure 3 shows the ℓ_1-error versus fraction of the simulated datasets for WGSQuikr, MetaPhyler, and MetaPhlAn. Also included is a smooth kernel distribution approximation of each of the histograms (shown as lines in figure 3) to emphasize how WGSQuikr typically has less error than MetaPhyler and MetaPhlAn.

We hypothesize that the reason WGSQuikr demonstrates such an improvement in ℓ_1-error over MetaPhyler and MetaPhlan is WGSQuikr's ability to very accurately reconstruct the frequency of the most abundant organisms in a sample. Indeed, at the genus level, the mean ℓ_1-error decreased by 31% when focusing on only the top 10 most abundant genera. See subsection 3.3 for further supporting evidence.

3.2.2. *Reconstruction Fidelity vs Simulation Parameters.* In order to investigate what properties of a given dataset influence the reconstruction error of WGSQuikr, we grouped the simulated datasets by each simulation parameter (number of reads, read length, abundance model, or diversity).

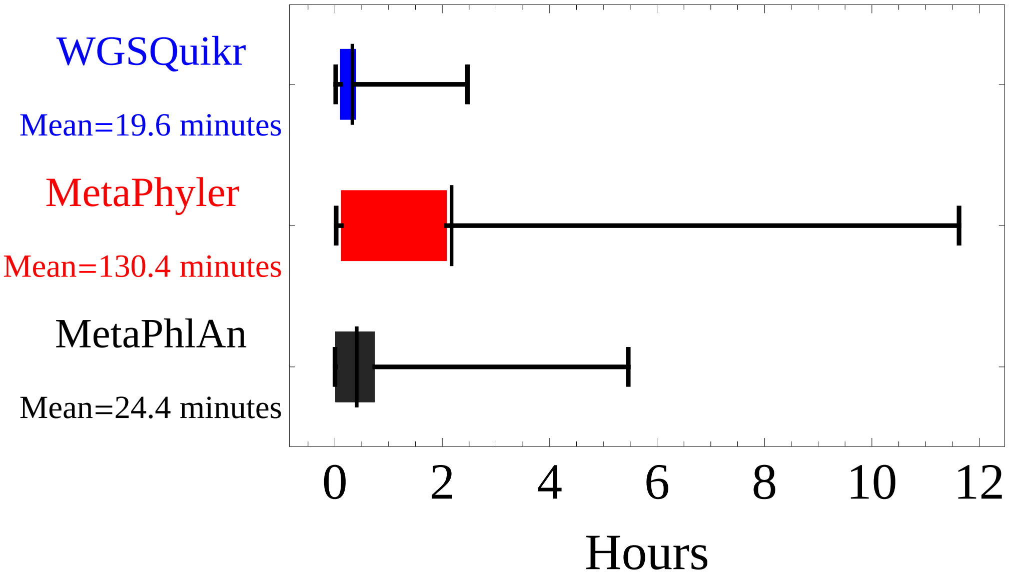

FIGURE 2. Box-and-whisker plot of execution time (in minutes) on the simulated experiments for WGSQuikr, MetaPhyler, and MetaPhlAn. The boxes demarcate 75% quantiles, whiskers demarcate range, and the vertical black bars are drawn at the mean.

TABLE 1. Comparison of mean ℓ_1-errors at the genus level (smaller values are better).

Method	Mean ℓ_1-error
WGSQuikr	0.644
MetaPhyler	1.006
MetaPhlAn	0.984

Figure 4 summarizes the mean error of WGSQuikr as a function of each one of these parameters, and includes the results for MetaPhyler and MetaPhlAn for comparison.

It is interesting to note that WGSQuikr runs particularly well on short read data. Indeed, WGSQuikr gave reasonable results when the read length was as short as 35bp or 50bp long, whereas MetaPhyler and MetaPhlAn both failed to return results in such cases. Furthermore, WGSQuikr exhibits roughly half as much ℓ_1-error (0.52) as MetaPhyler (0.97) and MetaPhlAn (0.99) for datasets consisting of reads normally distributed around 150bp.

Given a larger number of reads, a lower diversity, and an abundance model closer to exponential, all three methods experienced improvement in reconstruction fidelity. Interestingly, longer read lengths seemed to negatively impact all three methods.

3.3. **Mock Community Results.**

To show that WGSQuikr can allow for fast, high-level analysis of large datasets on a laptop computer, we analyzed the mock community described in subsection 2.6 using a 2013 Macbook Air. The dataset consists of over 6M reads of 75bp in length and is over 900MB in size. Using this laptop, which was equipped with a dual-core 1.3GHz Intel i5 processor, WGSQuikr completed analyzing the mock community in less than 8 minutes and used no more than 2GB of RAM. In contrast, using a much more powerful hexa-core 2.66GHz Intel Xeon X5650, MetaPhyler took 5.5 hours and MetaPhlAn took 2.9 hours.

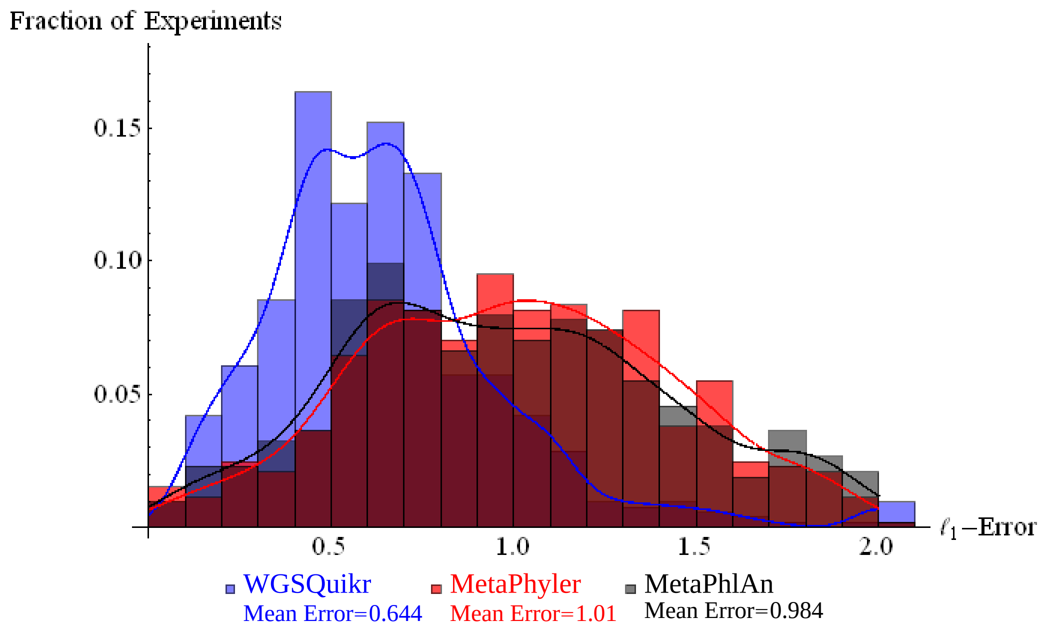

FIGURE 3. Histogram of ℓ_1-error versus fraction of simulated experiments at the genus level for WGSQuikr, MetaPhyler, and MetaPhlAn.

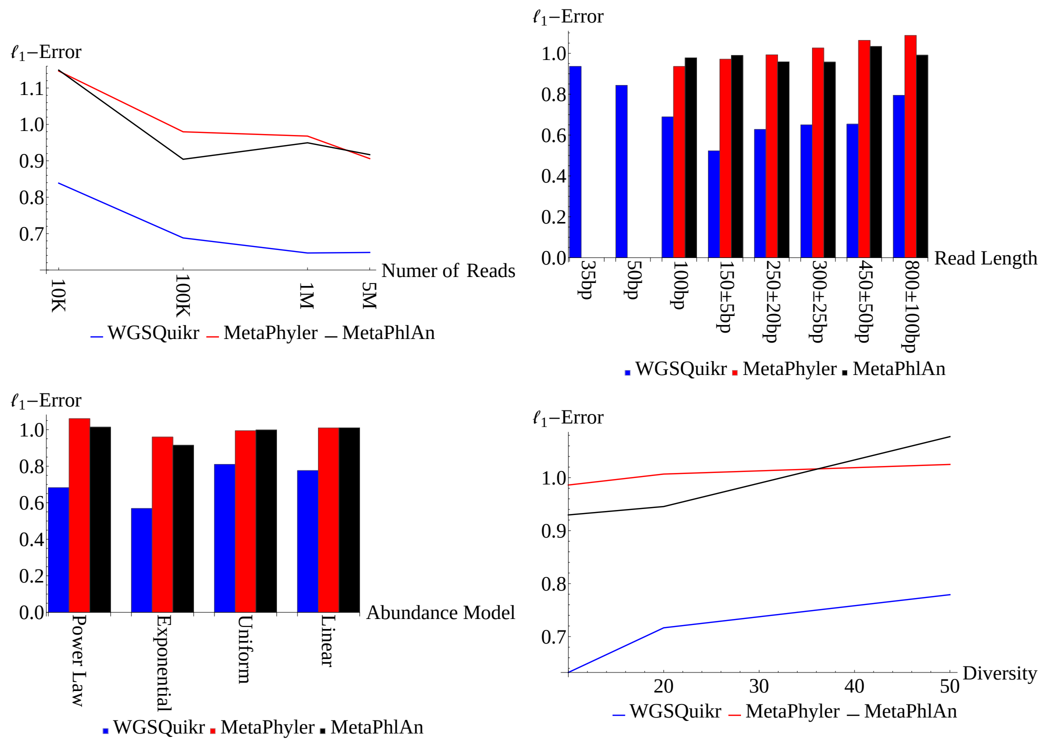

FIGURE 4. Mean ℓ_1-error at the genus level as a function of simulated dataset parameters for each method. MetaPhyler and MetaPhlAn failed to run on the datasets where reads were 35bp or 50bp in length.

The relative abundances of the organisms in the mock community are shown in figure 5 along with their predicted abundance for all three methods. Eukaryota where not included in the training databases of any of the methods, hence its absence in figure 5.

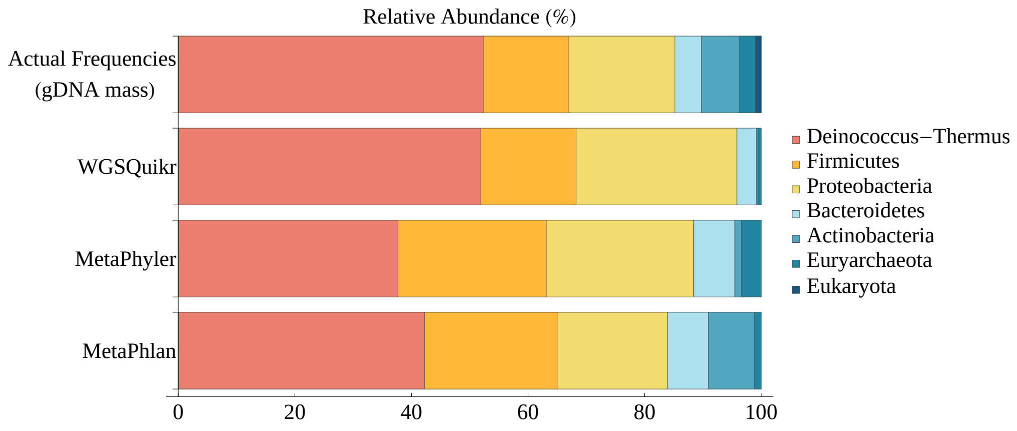

FIGURE 5. Relative abundances at the phylum level for reconstructions of organisms in the mock community.

As figure 5 indicates, out of all three methods, WGSQuikr recreates the relative abundance of the most frequently occurring phyla most accurately, at the expense of less accurate abundance estimation of the more rare phyla. This behavior was also observed at the genus level. This indicates that WGSQuikr is an effective tool for rapidly determining the predominant structure of a given metagenomic sample, at the expense of less accurate reconstruction of rare taxa.

3.4. **Cross Validation.** To gauge how well the WGSQuikr method will perform when the given sample contains bacteria not in the database (simulating novelty), we performed a 10-fold cross-validation. Throughout the cross-validation, the k-mer size was fixed at $k = 7$. The database D was partitioned into 10 disjoint sets and $1/10^{\text{th}}$ was set aside as testing data with the remaining $9/10^{\text{ths}}$ used to form a new k-mer matrix. Grinder [2] parameters were then chosen to generate a test sample from the testing data. In particular, these parameters were chosen as follows: read lengths normally distributed with a mean of 150bp and a standard deviation of 5bp, 1M total reads, a power law abundance model, a diversity of 10 species, and the homopolymer error model as in [21]. The mean ℓ_1-error was then taken over the choice of which $1/10^{\text{th}}$ was the testing data. Lastly, an average was taken over 100 iterates of this procedure.

TABLE 2. Results of 100 iterates of the 10-fold cross-validation procedure for WGSQuikr at the phylum and genus levels.

Taxonomic Rank	Mean ℓ_1-error $\pm$ variance
Phylum	0.342 ± 0.0574
Genus	1.22 ± 0.013

Table 2 summarizes the results of this procedure. The small mean and variance indicates that WGSQuikr performs well at the phylum level, even if a significant portion (10%) of the sample contains sequences not present in the database. At the genus level, the reconstruction was less accurate (compare to figure 3), indicating that WGSQuikr will benefit from the inclusion of as many bacterial genomes as possible. Hence, WGSQuikr performs best at a taxonomic rank that minimizes the number of novel taxa.

4. Conclusion

WGSQuikr represents a new class of metagenomics algorithms, one in which the taxonomic assignments of an entire WGS metagenome are computed, instead of performing the assignment in a read-by-read fashion. This allows for nearly constant execution time and low memory usage, and so is particularly well suited for analyzing very large datasets on a standard laptop computer. In contrast to current methods such as MetaPhyler and MetaPhlAn, WGSQuikr can be used to analyze metagenomes consisting of very short reads (such as the 35-50bp datasets generated by Illumina's MiSeq) in less than a few hours. As Illumina's advertised quality scores for such short read datasets are typically much higher than for longer read datasets, this may allow for more accurate analysis of metagenomes in unprecedentedly short time frames.

5. Acknowledgments

Funding: NSF (DMS-1120622)

Computations were generally performed using resources provided by the Ohio Supercomputer Center and funded by the Mathematical Biosciences Institute at The Ohio State University.

Appendix A. Design of Grinder Experiments

We detail here the production of the simulated data produced by the shotgun/amplicon read simulator Grinder [2]. These datasets were designed to mimic reads generated by a variety of Illumina platforms, and hence we set the read-length distributions to be normally distributed with a variety of means which are summarized in table 3. An equal number of datasets were generated

Mean (bp)	Standard Deviation (bp)	Number of Experiments
35	0	96
50	0	96
100	0	96
150	5	96
250	20	96
300	25	96
450	50	96
800	100	96

Table 3. Grinder experiment read lengths.

consisting of the following number of reads: 10K, 100K, 1M, 5M. Three different diversity values were chosen to be 10, 20, and 50 with abundances modeled by using the following four distributions: linear, uniform, power-law with parameter 0.750 and exponential with parameter 1. Homopolymers of length n were generated by a normal distribution with mean n and variance $0.15 * \sqrt{n}$ as in [21]. Sequencing errors were designed to model Illumina errors and used the 4^{th} degree polynomial in [14] with 80% of these errors set to be substitutions, while the remaining 20% set to be indels. Reference sequences were sampled proportionally to their length to mimic the length bias seen in WGS datasets.

Appendix B. Quikr Method Technical Details

B.1. **Mathematical Formulation.** Given the alphabet $\mathcal{A} = \{A, C, T, G\}$, let $\mathcal{A}^n$ denote the set of all words v of length $|v| = n$ on $\mathcal{A}$, and let $\mathcal{A}^* = \bigcup_{n\geq 0} \mathcal{A}^n$ be the set of all finite words on $\mathcal{A}$. Hence words containing non-$ACTG$ characters are ignored. Let $D = \{d_1, \dots, d_M\}$ be a database of genomic sequences $d_j \in \mathcal{A}^*$ and let $S = \{s_1, \dots, s_t\}$ be a set of sample sequences (the reads to be classified). Fix a k-mer size and endow $\mathcal{A}^k = \{v_1, \ldots, v_{4^k}\}$ with the lexicographic order. Let $\mathrm{occ}_v(w)$ represent the number of occurrences (with overlap) of the subword v in the word w. That is, for $w, v \in \mathcal{A}^n$, let

$$\mathrm{occ}_v(w) = |\{j : w_j w_{j+1} \cdots w_{j+|v|-1} = v\}|. \tag{B.1}$$

For $j = 1, \ldots, M$ and $i = 1, \ldots, 4^k$, define the *k-mer training matrix* entrywise as

$$A^{(k)}_{i,j} = \frac{occ_{v_i}(d_j)}{|d_j| - k + 1}. \tag{B.2}$$

The matrix $A^{(k)}$ satisfies $A^{(k)}_{i,j} \geq 0$ and is column-normalized, i.e.

$$\sum_{i=1}^{4^k} A^{(k)}_{i,j} = 1 \quad \text{for all } j = 1, \ldots, M. \tag{B.3}$$

Define the *sample k-mer frequency vector* entrywise for $i = 1, \ldots, 4^k$ as

$$s^{(k)}_i = \frac{\sum_{j=1}^{t} occ_{v_i}(s_j)}{\sum_{l=1}^{4^k} \sum_{j=1}^{t} occ_{v_l}(s_j)}. \tag{B.4}$$

We assume even coverage of each genome. That is, we assume that the composition of the bacterial community is represented by a probability vector $x \in \mathbb{R}^M$ satisfying the following: given a database sequence $d \in D$, the set of reads $\{s^d_1, \ldots, s^d_{t_d}\} \subset S$ coming from this sequence, and x_d the concentration in the sample of the bacteria corresponding to sequence d, for each i the following holds:

$$\frac{\sum_{j=1}^{t_d} occ_{v_i}(s^d_j)}{\sum_{l=1}^{4^k} \sum_{j=1}^{t_d} occ_{v_l}(s^d_j)} = x_d \times \frac{occ_{v_i}(d)}{\sum_{l=1}^{4^k} occ_{v_l}(d)}. \tag{B.5}$$

This means that the total k-mer count of all the read fragments corresponding to the sequence d is proportional to the k-mer count of the sequence d itself, with the proportionality constant being equal to the concentration of the sequence d in the sample S. Our assumptions imply that

$$A^{(k)} x = s^{(k)}. \tag{B.6}$$

We will try to recover the probability vector x satisfying $x_j \geq 0$ for all $j = 1, \ldots, M$ and $\sum_{j=1}^{M} x_j = 1$ from information in the form of equation (B.6).

B.2. **Nonnegative Basis Pursuit Denoising.** Given that a bacterial community is typically distributed as a sparse vector x (a small percentage of all extant bacteria are actually present in a given sample), we pursue sparsity-promoting minimizations involving the ℓ_1-norm. Basis Pursuit [10], called (BP) below, is one of the most popular methods. In our situation, it is natural to include the nonnegativity constraint, leading to (BP$_{\geq 0}$). We further modify the optimization by relaxing the equality constraint to arrive at the regularized problem (REG$^2_1$). Thus, the three optimization problems considered are:

$$\text{(BP)} \qquad \underset{z \in \mathbb{R}^M}{\text{minimize}} \; ||z||_1 \qquad \text{subject to } A^{(k)} z = s^{(k)},$$

$$(\text{BP}_{\geq 0}) \qquad \underset{z \in \mathbb{R}^M}{\text{minimize}} \; ||z||_1 \qquad \text{subject to } A^{(k)} z = s^{(k)} \text{ and } z \geq 0,$$

$$(\text{REG}^2_1) \qquad \underset{z \in \mathbb{R}^M}{\text{minimize}} \; ||z||_1^2 + \lambda^2 ||A^{(k)} z - s^{(k)}||_2^2 \qquad \text{subject to } z \geq 0,$$

It can be demonstrated, thanks to (B.3), that (BP) and (BP$_{\geq 0}$) are equivalent in the sense that x is a solution of (BP) if and only if it is a solution of (BP$_{\geq 0}$), and that the latter is approached by solutions of (REG$^2_1$) when $\lambda \to \infty$, see [10].

We shall solve (REG_1^2) since it has the notable advantage of being transformed into a nonnegative least squares problem. Indeed, with

$$\tilde{A}^{(k)} := \left[\frac{1\cdots 1}{\lambda A^{(k)}}\right], \qquad \tilde{s}^{(k)} := \left[\frac{0}{\lambda s^{(k)}}\right], \tag{B.7}$$

the minimization (REG_1^2) is equivalent to

$$\underset{z\in\mathbb{R}^M}{\text{minimize}}\ ||\tilde{A}^{(k)}z - \tilde{s}^{(k)}||_2^2 \qquad \text{subject to } z \geq 0. \tag{NNLSQ}$$

B.3. **Algorithmic Implementation.** To solve (NNLSQ) we utilized MATLAB's [1] implementation of `lsqnonneg()` which in turn is an implementation of the iterative Lawson-Hanson algorithm described in [16]. To calculate the matrices $A^{(k)}$ and the vector $s^{(k)}$ we used a custom SML [19] subword counting program written by Christopher Cramer and compiled for Linux using MLton [28].

B.4. **Selection of λ.** Parameter tuning is a common issue to be addressed when using regularized optimization procedures to solve linear inverse problems [26, 3, 8]. Two common methods include Generalized Cross Validation [11] and the L-Curve method [12]. The adaptive method by which we select the λ used in (B.7) is similar in spirit to the L-Curve method. In every case, it was observed that as a function of λ, the number of iterates necessary to solve (NNLSQ) via the Lawson-Hanson algorithm was linear, then exponential, then non-increasing. Figure 6 demonstrates this phenomenon for a sample dataset by plotting the number of iterates as a function of λ. Let $its(\lambda)$ be the number of iterates needed to solve (NNLSQ) via the Lawson-Hanson algorithm. It was also observed that x was most accurately reconstructed at the λ for which the number of iterations experienced its greatest increase. Figure 7 demonstrates this fact for an example dataset by plotting the ℓ_1-error as well as $\frac{d}{d\lambda}its(\lambda)$ as a function of λ, where $\frac{d}{d\lambda}its(\lambda)$ denotes the first differences of $its(\lambda)$.

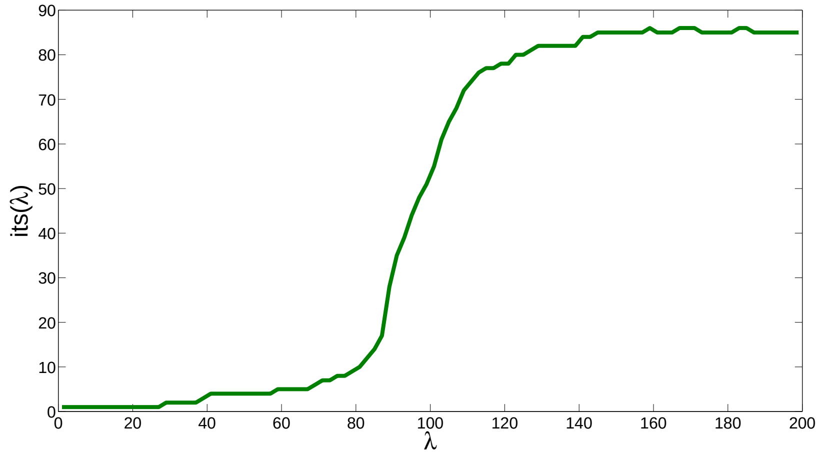

FIGURE 6. Number of iterates to solve (NNLSQ) as a function of λ.

With this information in hand, the following adaptive approach was used to choose λ. First, let $\frac{d}{d\lambda}its(\lambda_{t_0:I:t_1}))$ designate the first differences of $its(\lambda)$ when λ ranges from t_0 to t_1 in increments of I. We allowed λ to increase in increments of 100 from $\lambda = 1$ until a smoothing spline approximation of $\frac{d}{d\lambda}its(\lambda_{1:100:t_1})$ was shown to be negative for some t_1. A smoothing spline approximation was utilized because $\frac{d}{d\lambda}its(\lambda_{t_0:I:t_1})$ is noisy when increasing λ in such large increments. Next, the maximum of $\frac{d}{d\lambda}its(\lambda_{1:100:t_1})$ with respect to λ was identified, call it $\lambda = t_M$. Then the maximizer of $\frac{d}{d\lambda}its(\lambda_{t_M-100:10:t_M+100})$ was used as the value of λ to solve (NNLSQ). Clearly this method can

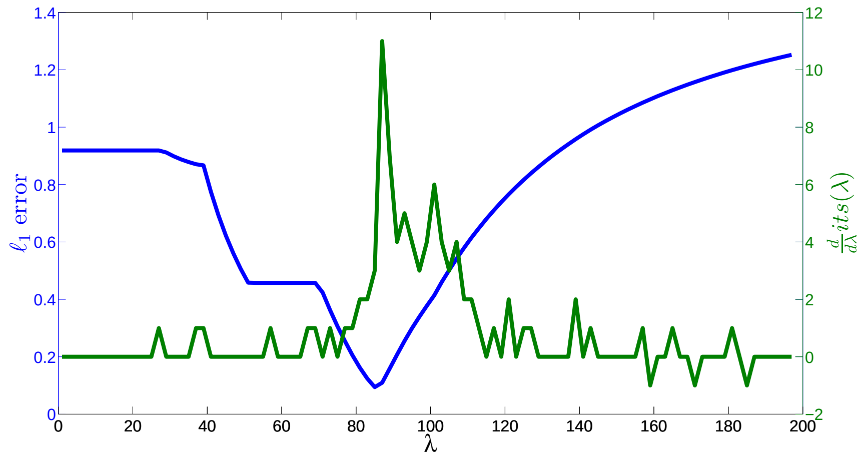

FIGURE 7. ℓ_1-error and first difference of number of iterates needed to solve (NNLSQ) as functions of λ.

be refined further by taking smaller and smaller increments surrounding the maximizer of $\frac{d}{d\lambda}its(\lambda)$, but we found this two-step approach to provide sufficient speed and accuracy improvements over current methods.

REFERENCES

[1] MATLAB 2012b, The MathWorks, Inc., Natick, MA, USA.
[2] F. E. Angly, D. Willner, F. Rohwer, P. Hugenholtz, and G. W. Tyson. Grinder: a versatile amplicon and shotgun sequence simulator. *Nucleic acids research*, 61(0):1–8, Mar. 2012.
[3] M. Bertero and P. Boccacci. *Introduction to Inverse Problems in Imaging*. Institute of Physics Publishing, London, 1998.
[4] A. Brady and S. Salzberg. Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models. *Nature Methods*, 6(9):673–676, 2009.
[5] A. Brady and S. Salzberg. PhymmBL expanded: confidence scores, custom databases, parallelization and more. *Nature Methods*, 8(5):367, 2011.
[6] N. Carlos, Y.-W. Tang, and Z. Pei. Pearls and pitfalls of genomics-based microbiome analysis. *Emerging Microbes & Infections*, 1(12):e45, Dec. 2012.
[7] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, Jan. 1998.
[8] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm Pure Appl. Math*, 57(11):1413–1457, 2004.
[9] C. F. Davenport, J. Neugebauer, N. Beckmann, B. Friedrich, B. Kameri, S. Kokott, M. Paetow, B. Siekmann, M. Wieding-Drewes, M. Wienhöfer, S. Wolf, B. Tümmler, V. Ahlers, and F. Sprengel. Genometa - a fast and accurate classifier for short metagenomic shotgun reads. *PLoS ONE*, 7(8):e41224, 2012.
[10] S. Foucart and D. Koslicki. Sparse Recovery by means of Nonnegative Least Squares. *IEEE Signal Processing Letters*, under review, 2013.
[11] G. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.
[12] P. Hansen and D. O'Leary. The use of the L-curve in the regularization of discrete ill-posed problems. *SIAM J. Sci. Comput.*, 14(6):1487–1503, 1993.
[13] Jumpstart Consortium HMP Data Generation Working Group. Evaluation of 16S rDNA-Based Community Profiling for Human Microbiome Research. *PLoS ONE*, 7(6):e39315, 2012.
[14] J. O. Korbel, A. Abyzov, X. J. Mu, N. Carriero, P. Cayting, Z. Zhang, M. Snyder, and M. B. Gerstein. PEMer: a computational framework with simulation-based error models for inferring genomic structural variants from massive paired-end sequencing data. *Genome biology*, 10(2):R23, Jan. 2009.
[15] D. Koslicki, S. Foucart, and G. Rosen. Quikr: a Method for Rapid Reconstruction of Bacterial Communities via Compressive Sensing. *Bioinformatics (Oxford, England)*, 29(17):2096–2102, 2013.

[16] C. Lawson and R. Hanson. *Solving Least Squares Problems*. Prentice-Hall, 1974.

[17] B. Liu, T. Gibbons, M. Ghodsi, T. Treangen, and M. Pop. Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences. *BMC genomics*, 12(Suppl 2):S4, Jan. 2011.

[18] N. J. MacDonald, D. H. Parks, and R. G. Beiko. Rapid identification of high-confidence taxonomic assignments for metagenomic data. *Nucleic Acids Research*, 40(14):e111, 2012.

[19] R. Milner, M. Tofte, and R. Harper. *The Definition of Standard ML*. MIT press, Cambridge, MA, 1997.

[20] 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.

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

[22] G. Rosen, E. Garbarine, D. Caseiro, R. Polikar, and B. Sokhansanj. Metagenome fragment classification using N-mer frequency profiles. *Advances in bioinformatics*, 2008:205969, Jan. 2008.

[23] E. W. Sayers, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. DiCuccio, R. Edgar, S. Federhen, M. Feolo, L. Y. Geer, W. Helmberg, Y. Kapustin, D. Landsman, D. J. Lipman, T. L. Madden, D. R. Maglott, V. Miller, I. Mizrachi, J. Ostell, K. D. Pruitt, G. D. Schuler, E. Sequeira, S. T. Sherry, M. Shumway, K. Sirotkin, A. Souvorov, G. Starchenko, T. A. Tatusova, L. Wagner, E. Yaschenko, and J. Ye. Database resources of the National Center for Biotechnology Information. *Nucleic acids research*, 37(Database issue):D5–15, Jan. 2009.

[24] N. Segata, L. Waldron, A. Ballarini, V. Narasimhan, O. Jousson, and C. Huttenhower. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature methods*, (9):811–8147, June 2012.

[25] S. Srinivasan and C. Guda. MetaID: A novel method for identification and quantification of metagnomic samples. *BMC Genomics*, 14(Suppl 8):S4, 2013.

[26] C. Vogel. *Computational Methods for Inverse Problems*. SIAM, Philadelphia, PA, 2002.

[27] 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. *Applied and environmental microbiology*, 73(16):5261–7, Aug. 2007.

[28] S. Weeks. Whole-program compilation in MLton. In *Proceedings of the 2006 workshop on ML*, page 1, New York, NY., 2006. ACM.