Metagenomic Taxonomic Inference (MTI)

Felix A. Sosa*
Electrical Engineering and Computer Science, University of Central Florida

Trevor C. Ballard*
Electrical Engineering and Computer Science, University of Central Florida

Harsh Patel*
Electrical Engineering and Computer Science, University of Central Florida

Austin Vo*
Electrical Engineering and Computer Science, University of Central Florida

Shibu Yooseph
Electrical Engineering and Computer Science, University of Central Florida

***Abstract*—As biological data becomes more readily available through the advancement of DNA sequencing technologies, the need to efficiently extract non-trivial knowledge from this data is becoming more urgent. In this paper, we propose an end-to-end Metagenomic Taxonomic Inference (MTI) pipeline that allows users to infer the taxonomic makeup and relative abundance of a given sample of sequenced DNA data. The pipelines core components are based on the GRAMMy [2] and BWA [3] systems described herein but compose them in a novel fashion on the newly installed COOMBS cluster at the University of Central Florida, making MTI UCF's first in-house end-to-end bioinformatics tool.**

***Index Terms*—Genomics, bioinformatics, expectation-maximization algorithms, transforms, genetics, mixture models, probabilistic computing**

I. Introduction

Metagenomics is the direct genetic analysis of genomes within an environmental sample. It applies a suite of technologies and bioinformatics tools to access the genetic content of entire communities of organisms. The field of metagenomics has been posited to be responsible for significant advances in many subfields of biology including microbial ecology, evolution, and diversity over the past 5 to 10 years [1]. As the promise of further study into this field and technologies grows, many research groups are actively engaging in it now and there is a growing need for cohesive end-to-end tools for researchers to choose from.

This paper proposes a cohesive end-to-end pipeline for bioinformatics researchers to characterize communities of organisms under investigation. Using Mixture Model theory, MTI is designed to make inferences of the taxonomic makeup and relative abundances of any given metagenomic sample. MTI composes Genome Relative Abundance using Mixture Model theory (GRAMMy) by Xia et. al. [2] and the Burrows-Wheeler Aligner (BWA) by Li and Durner [3] into a forward pipeline to make accurate inferences on the relative abundances of species within a metagenomic sample.

Composing GRAMMy and BWA together with custom visualizations into an end-to-end pipeline makes MTI a powerful inference tool suitable for the technical needs of bioinformatics researchers looking to characterize the taxonomies of their metagenomic samples.

* Authors contributed equally

II. The MTI Pipeline

The MTI pipeline consists of four major pieces: Main File, BWA, GRAMMy, and Visualization. See Fig. 1. The Main File and Visualization are both written in Python 3.6, GRAMMy is written in C++, and BWA is written in C. You can find all of the code and documentation at the authors GitHub: https://github.com/ballardt/mti. The pipeline is explained below.

A. Main File

The Main File is the main entry point into the MTI pipeline. It handles user input and input to and output from both BWA and GRAMMy. Proper use of the Main File has a user supply sample metagenomic data to be analyzed in the form of a FASTA/Q file, a file format for representing nucleotide or peptide sequences, and the reference genome data in the form of an FNA file, an extension of the FASTA/Q format to specify nucleic acid sequences, to be compared with the sample for analysis. The output of the Main File a Genome Relative Abundance (GRA) file containing the final output of the pipeline.

The Main File passes the FASTA/Q input from the user into BWA which then, after mapping the sample to the reference, outputs a Sequence Alignment/Map (SAM) [4] file containing the mappings. The SAM file is then input to a parser we wrote in python 3.6 that parses the SAM file into a more readable CSV file that is then input to GRAMMy. GRAMMy then outputs a GRA file which is stored in local storage by the Main File. At this point the analysis is complete and the GRA file is used by Visualization to present to the user rich visualizations of the data.

B. BWA

The Burrows-Wheeler Aligner BWA is a software package developed by Li and Durner that consists of three different algorithms based on the Burrows Wheeler Transform for mapping low-divergent sequences against a large reference genome [3]. The three algorithms are BWA-backtrack, BWA-SW and BWA-MEM. Each are suited for samples with differing properties.

BWA-backtrack is primarily designed for Illumina sequence reads up to 100bp [3]. BWA-SW and BWA-MEM are suited for longer sequence reads - usually from 70bp to 1Mbp [3].

BWA-MEM is the newest of the three and is recommended due to its being faster, more accurate, and having overall better performance than BWA-SW [3]. The MTI pipeline uses BWA-MEM.

We chose BWA as our mapping tool for MTI because of its greater read mapping speed [3] and ability to output the standard SAM file format that can be used with the SAMtools software package by Li et. al. [4]

C. GRAMMy

The Genome Relative Abundance using Mixture Model theory was developed by Li and Durner as a computational framework to make accurate inferences on the compositions of microbial communities. GRAMMy utilizes Mixture Model theory, a probabilistic model that allows for the robust representation of the presence of subpopulations within an overall population or, in other words, estimating the different populations within a metagenomic sample. GRAMMy is based on an Expectation Maximization (EM) algorithm [2]. GRAMMy accepts a CSV file from the parser and subsequently performs the Maximum Likelihood Estimation (MLE) of the genomic relative abundance levels within the sample [2].

Genomic relative abundance (GRA) is defined as the relative abundance measure of mostly unicellular microbial organisms. The sampling and sequencing procedure will go as follows: Randomly choose a reference genome gj with probability j that is proportional to ajlj, where aj is the abundance and lj is the genome length. Randomly generate a read rj from gj.

With a reasonable assumption of independence between steps 1 and 2, the procedure should be equivalent to sampling from the mixture distribution 1.

$$M : M = \sum_{j=1}^{m} \pi_j f_{g_j} \tag{1}$$

Where f_g is a probability distribution such that the probability of generating a read r_j from g_j is $f_g(r_j)$.

The GRA for known genomes $a = (a_1, a_2, ..., a_{m1})$ is the normalized abundance, where the relative abundance for the j-th known genome is described in 2.

$$a_j = \frac{\pi_j}{l_j \sum_{k=1}^{m-1} \frac{\pi_k}{l_k}} \tag{2}$$

where $j \in \{1, 2, ..., m-1\}$.

To estimate the mixing parameters π_j for j genomes, the following Expectation-Maximization (EM) algorithm will be used to calculate the MLE of each read to each reference genome.

We assume there is a responsibility matrix Z such that each entry z_{ij} is the probability that read r_i is from reference genome g_j. The following E and M-step procedures will list out the EM algorithm implemented in GRAMMy. Note that a variable with superscript t stands for its value at the t-th iteration, i.e. $p(t)$ is the estimate of at the t-th step.

Assuming that $\pi(t)$ is known, $Z(t)$ can be updated by the corresponding posterior probabilities in 2 in the E-step where U is the maximum number of mismatches allowed in the read alignment [5].

$$Z_{ij}^t = \frac{\begin{cases} \sigma^2(1-\sigma)^{1-z_{\pi_j}} & \text{if } z \le U \\ 0 & \text{if } z > U \end{cases}}{\sum_{k=1}^{m} \begin{cases} \sigma^2(1-\sigma)^{1-z_{\pi_j}} & \text{if } z \le U \\ 0 & \text{if } z > U \end{cases}} \tag{3}$$

Assuming that $Z(t)$ is known, the new mixing parameter $\pi(t+1)$ is updated in the M-step described in 3.

$$\pi_j^{t+1} = \frac{\sum_{i=1}^{n} z_{ij}^t}{n} \tag{4}$$

When the MLE of π is found, the MLE of a can be calculated using 2, thereby inferring the GRAs.

D. Visualizations

The information within a GRA file is important when attempting to understand a metagenomic sample, but it is difficult to reason about in the standard GRA format. A single sample could yield a GRA file with thousands of columns, which are nearly impossible to contextualize when presented as raw numeric values and taxon IDs. Furthermore, the user may want to consider the results per some metadata or query; e.g., the relative abundance of bacteria in samples taken from individuals over the age of 40 versus those under the age of 40. We identified five visualization targets that will suffice for the majority of use cases: heatmap, taxonomic tree, violin plots, and bar chart.

A heatmap, as shown in Fig. 2, is a graph in which values are represented as colors. There are many different forms of heatmaps, but for our purposes the axes represent the samples provided by the user and the microbial species found within those samples, or some filtering or grouping of either. The color of each cell in our heatmap represents the relative abundance of a microbe in a sample. Taxonomic trees, like phylogenetic trees or evolutionary trees, can take on wildly different forms according to their purpose. In our users case, a taxonomic tree offers a quick look at the relationship between the organisms in a sample and how similar or dissimilar they are to each other. While there are some critical differences between taxonomic and phylogenetic trees concerning the arrangement of organisms, we have opted to visualize taxonomy because it is often more useful to examine the similarity of organisms in the sample than it is to chart their raw evolutionary history.

In our visualization, each node includes the name of the organism or taxonomic classification, the relative abundance of that node in the sample, the error bound for the relative abundance, and any arbitrary data specified by the user, such as the presence of tagged genes. Certain values like relative abundance are combined as one traverses towards the root of the tree; e.g. if node A has children B and C, and B and C each have a relative abundance of 0.25, node A will show a relative abundance of 0.5. Users may filter and group

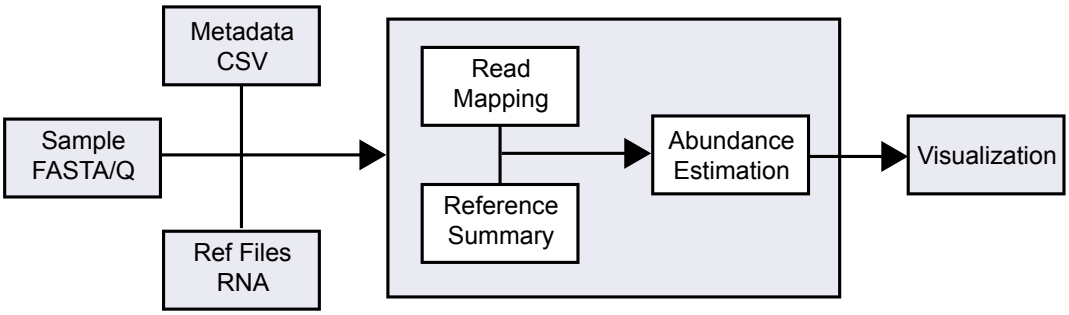

Fig. 1. The pipeline for MTI. Users provide the genomic sample, in the form of a FASTA/Q file, optional metadata, in the form of a CSV, and a reference database. MTI then performs a read mapping using the Burrows-Wheeler Aligner and genomic inference with GRAMMy. Statistical summaries of the sample are then visualized via multiple options including heatmaps, violin plots, bar graphs, and scatter plots.

samples or choose taxonomic classifications when displaying the taxonomic tree similarly to heatmaps. Samples may be excluded from the visualization, and users may choose to only visualize one type of organism in the tree. For example, a user may choose to visualize the taxonomic relationship between all bacteria in samples 1 and 4, or they may wish to only see all members of Enterobacterales present in sample 3.

Scatterplots, as shown in Fig. 3 are a very common way to visualize the relationship between two variables. In two-dimensional scatterplots, each point on the graph represents a single instance of some data point with the value of the horizontal and vertical axes at those values. The data points may also be colored to represent a third variable. A line of best fit is often plotted as well to specify and generalize the correlation between the variables on the axes.

In our case, the scatterplot will most frequently be used to visualize the relationship between the relative abundance of some organism against some sample metadata provided by the user, and points may be colored according to categorization. For example, a user may want to know how the relative abundance of pathogenic and non-pathogenic strains of some bacteria. In this case, the relative abundance may represent the horizontal axis, the age of the person each sample was taken from may represent the vertical axis, and points may be blue if they are non-pathogenic and red if they pathogenic.

This sort of use case is frequently one of the best ways to better understand a users samples per some metric because it offers a simple way to graph the relationship between two or three arbitrary variables.

Bar charts, as shown in Fig. 5, are used in cases like those for scatter plots, but are more suited to tracking changes over time or comparing values between two or more categories. One example of a situation in which bar charts could be helpful is if the user wanted to compare the relative abundance of some bacteria in a few different metagenomes over time, such as three different people over the course of a year. The months of the year may be placed on the horizontal axis, relative abundance of the bacteria may be placed on the vertical axis, and three bars may be placed at each month representing the bacteria in each of the three individuals.

We include violin plots, as shown in Fig. 4, as well because they are a simple way of visualizing and comparing variable distributions. For example, a user may want to compare the distribution of a pathogenic strain of bacteria versus a non-pathogenic strain of the same bacteria across multiple samples. This would produce a violin plot with the relative abundance on the horizontal axis and two variables representing the bacterial strains on the vertical axis.

Each of the five visualizations has been sufficient to extract significant knowledge about our simulated datasets during testing.

III. Testing And Benchmarking

To test our pipeline, we needed simulated datasets with known or expected GRA values that could be compared with the MTI pipelines values given the dataset as input. The datasets were manually created with the help of open source software known as wgsim [4]. These simulated datasets served as our benchmarks for testing and debugging throughout the latter half of the development of the MTI pipeline. This process of creating simulated data is referred to as pre-processing.

Pre-processing included approximately eight steps. The first step was to create a CSV file containing compiled metadata of complete genomes of bacteria. We accomplished this by creating a bash script that searched online over NCBIs GenBank for the necessary metadata including the GenBank Account Number, Molecule Type, Molecule Length, Taxon ID, Organism Name, and Taxonomy (ascending from direct most parent to base cellular organism). After the collection of metadata in a CSV, we would pick a subset of organisms from the CSV.

After the selection of a subset of organisms, we would ensure that the chosen organisms are associated to FASTA files that correspond to the original CSV file. This is to ensure we do not have data that does not correspond to any metadata during analysis and lead to bad estimates from GRAMMy. Specifically, we would check that all GenBank Account Numbers in the chosen dataset are in each of the associated FASTA files and that each of the GenBank Account

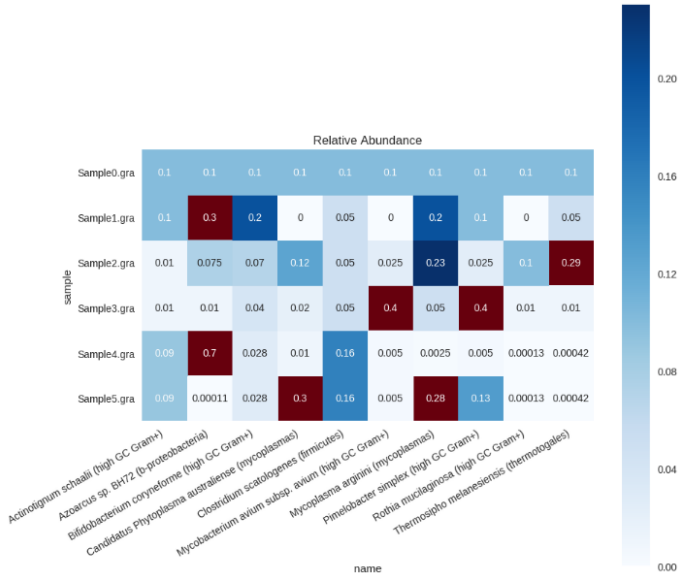

Fig. 2. Example of heatmap provided by MTI. The x-axis correlates to the samples provided by the user and the y-axis is the inferred species. The colors correlate with the relative abundance of the species within the sample with red being denoted as an outlier value.

Numbers in the associated FASTA files are in the chosen dataset. After ensuring the data is good, we would estimate the GRA of the dataset with a given distribution using simple spreadsheet formulas.

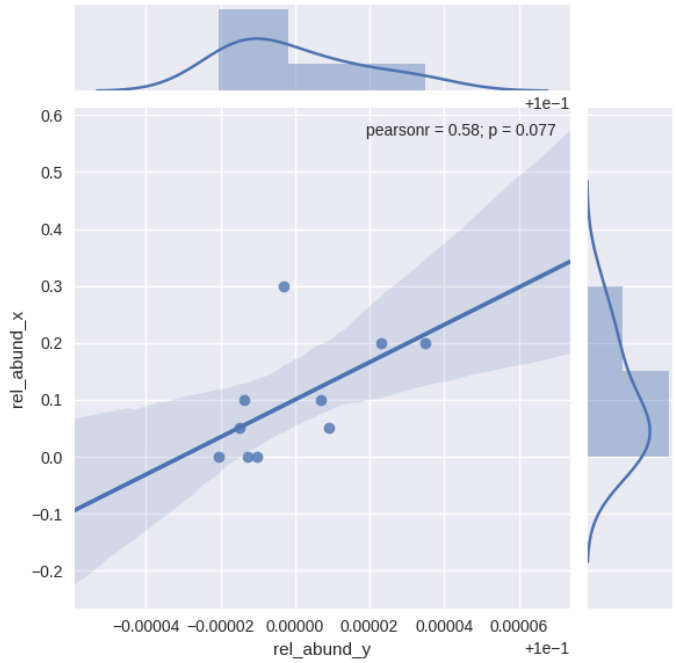

Fig. 3. Example of scatter plot provided by MTI. The axes both represent the relative abundances of two samples, respectively. Users also have the option to view pearson correlation.

After estimating the GRAs manually, we would generate a bash script containing the reference genome database. This is basically concatenating all of the FASTA files associated to the dataset and is easily done using spreadsheet formulas.

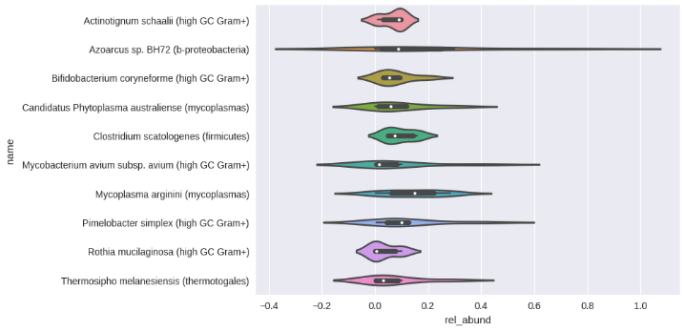

Fig. 4. Example of violin plot provided by MTI.

After creating the bash script containing the FASTA files, we generate a bash script that will utilize wgsim to create the simulated data.

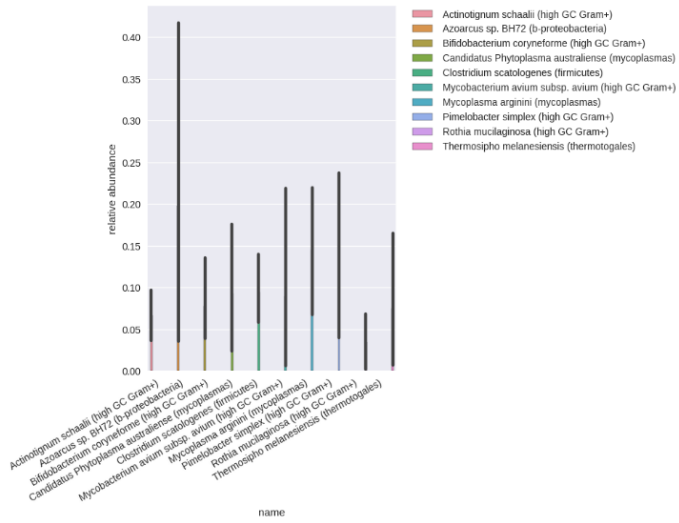

Fig. 5. Example of bar plot provided by MTI.

We then run both bash scripts to generate reference genomes and simulated sample inputs in the Main File directory. Finally, we input these simulated genome references and sample inputs into the Main File and compare the GRA output by GRAMMy with the estimated GRA from the fourth step. If the error is acceptable (¡ 5% error) we conclude that GRAMMy is successful in inferring the GRA of the simulated data and complexify the data by changing parameters such as the distribution and composition of the simulated sample until we are confident in using real datasets. Otherwise, we debug the pipeline until we get acceptable error rates with the simulated data.

IV. Discussion and Future Directions

The MTI pipeline so far has proven to successfully pass benchmarks and reach low error rates. Though this makes the pipeline a success there are a few modifications and edits that

could be made to the pipeline to give it more power and rigor. Three that are low-hanging fruit in that their implementation would be simple and their benefit significant are automating the pre-processing so as to decrease testing time and act as quality assurance for any modifications, implement a parallel GRAMMy system via general-purpose computing on graphics processing units (GPGPU) with CUDA, and expand the pipeline to incorporate genomes of organisms such as viruses.

Pre-processing for the pipeline proved to be difficult in that preparing simulated data manually was time-intensive and prone to human error. In the latter portion of our project, when running benchmarks using larger genomes than usual for testing scalability, we consistently had error rates in output GRAs from expected GRAs upwards of 20%. We traced the issue down to incorrect data prepared during pre-processing. Though we fixed the issue and ensured the pipeline was functional, the process took significant time from the pipeline. Automating this process would provide a less time-intensive and more consistent alternative to current manual methods employed.

Another means of saving time would be to implement a GRAMMy system using GPGPU with CUDA. Having access to graphics processing would allow users to benefit from potential significant speed increases and thus faster analyses. This speed up could lead to quicker or more data that could help intensive and important projects relying on this pipeline. Importantly, an implementation of GRAMMy with CUDA would not prove to be difficult relative to other possible modifications as GRAMMy is written in C++ and benefits from helpful libraries and online support for being translated into CUDA.

Finally, an expansion or testing of the pipelines capacity to effectively infer the GRA of other organisms has obvious benefits on expanding the utility of the pipeline for other researchers. Though, we are unsure if the implementation of this capacity would be a simple task, testing the capacity of GRAMMy its current state is sure to be. Thus, we believe it as well as the other two suggested modifications could prove fruitful for future efforts with the MTI pipeline.

V. Conclusion

We describe an end-to-end Metagenomic Taxonomic Inference pipeline that can be used to infer the taxonomic makeup of metagenomic data. The pipeline features read mappings performed by the Burrows-Wheeler Aligner, relative abundance estimation through Genome Relative Abundance using Mixture Model theory, and rich visualizations including heatmaps, bar charts, scatter plots, taxonomic trees, and violin plots.

Together, these components make MTI a valuable tool for bioinformatics researchers looking to analyze the taxonomy of any metagenomic sample. Though, the pipeline is finished and functional, we also detailed three possible modifications that would greatly increase the utility of the pipeline. These were to implement parallel computations in GRAMMy through GPGPU via CUDA, automating the testing and debugging of the pipeline with simulated data, and testing the pipelines capabilities with other genomic data to see if it can infer the taxonomic makeup of organisms other than bacteria such as viruses.

Acknowledgment

The authors wish to acknowledge the assistance and support of Dr. Shibu Yooseph and Dr. Mark Heinrich of the Electrical Engineering and Computer Science Dept. at the University of Central Florida.

References

[1] Thomas, Torsten, Jack Gilbert, and Folker Meyer. Metagenomics - a Guide from Sampling to Data Analysis. Microbial Informatics and Experimentation 2 (2012): 3. PMC. Web. 18 Apr. 2017.E power amplifier, IEEE Trans. Microwave Theory Tech., vol. 48, no. 12, pp. 2397-2402, December 2000.

[2] Xia LC, Cram JA, Chen T, Fuhrman JA, Sun F (2011) Accurate Genome Relative Abundance Estimation Based on Shotgun Metagenomic Reads. PLoS ONE 6(12): e27992. https://doi.org/10.1371/journal.pone.0027992.

[3] Li H. and Durbin R. (2010) Fast and accurate long-read alignment with Burrows-Wheeler Transform. Bioinformatics, Epub. [PMID: 20080505]

[4] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, 1000 Genome Project Data Processing Subgroup; The Sequence Alignment/Map format and SAMtools. Bioinformatics 2009; 25 (16): 2078-2079. doi: 10.1093/bioinformatics/btp352

[5] Tae-Hyuk Ahn, Juanjuan Chai, Chongle Pan; Sigma: Strain-level inference of genomes from metagenomic analysis for biosurveillance. Bioinformatics 2015; 31 (2): 170-177. doi: 10.1093/bioinformatics/btu641