3-20-2020

Scalable Profiling and Visualization for Characterizing Microbiomes

Camilo Valdes
Florida International University, cvalde03@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

SCALABLE PROFILING AND VISUALIZATION FOR CHARACTERIZING MICROBIOMES

A dissertation submitted in partial fulfillment of

the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Camilo Valdes

2020

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Camilo Valdes and entitled Scalable Profiling and Visualization for Characterizing Microbiomes, having been approved in respect to style and intellectual content, is referred to you for judgement.

We have read this dissertation and recommend that it be approved.

Leonardo Bobadilla

Kalai Mathee

Jennifer Clarke

Ruogu Fang

Giri Narasimhan, Major Professor

Date of Defense: March 20, 2020

The dissertation of Camilo Valdes is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
And Dean of the University Graduate School

Florida International University, 2020

DEDICATION

For Kobie and my Family. Thank you for the love and support.

ACKNOWLEDGMENTS

The work in this dissertation would not have been possible without the help and support of many people that have helped me throughout many years.

First and foremost I would like to thank my advisor, Dr. Giri Narasimhan, for his patience, advise, mentorship, and drive for excellence. Throughout the many years that the work in this dissertation has spanned, Dr. Giri has taught me how to improve ideas and not settle for good enough. Without his direction and leadership, the work here would not have been possible.

My committee members, Dr. Ruogu Fang, Dr. Jennifer Clarke, Dr. Kalai Mathee, and Dr. Leonardo Bobadilla have been an instrumental source of growth, instruction, counsel and advice. Without their support, suggestions, and advice I would have never been exposed to a lot of the topics that have driven my curiosity over the past couple of years.

I would also like to thank my lab-mates, and the members of the Bioinformatics Research Group (BioRG), who have been a constant source of discussion, criticism, and support. With their help the work here has been made all the better.

ABSTRACT OF THE DISSERTATION

SCALABLE PROFILING AND VISUALIZATION FOR CHARACTERIZING MICROBIOMES

by

Camilo Valdes

Florida International University, 2020

Miami, Florida

Professor Giri Narasimhan, Major Professor

Metagenomics is the study of the combined genetic material found in microbiome samples, and it serves as an instrument for studying microbial communities, their biodiversities, and the relationships to their host environments. Creating, interpreting, and understanding microbial community profiles produced from microbiome samples is a challenging task as it requires large computational resources along with innovative techniques to process and analyze datasets that can contain terabytes of information.

The community profiles are critical because they provide information about what microorganisms are present in the sample, and in what proportions. This is particularly important as many human diseases and environmental disasters are linked to changes in microbiome compositions.

In this work we propose novel approaches for the creation and interpretation of microbial community profiles. This includes: (a) a cloud-based, distributed computational system that generates detailed community profiles by processing large DNA sequencing datasets against large reference genome collections, (b) the creation of microbiome maps: interpretable, high-resolution visualizations of community profiles, and (c) a machine learning framework for characterizing microbiomes from the microbiome maps that delivers deep insights into microbial communities.

The proposed approaches have been implemented in three software solutions: Flint, a large scale profiling framework for commercial cloud systems that can process millions of DNA sequencing fragments and produces microbial community profiles at a very low cost; Jasper, a novel method for creating microbiome maps, which visualizes the abundance profiles based on the Hilbert curve; and Amber, a machine learning framework for characterizing microbiomes using the microbiome maps generated by Jasper with high accuracy.

Results show that Flint scales well for reference genome collections that are an order of magnitude larger than those used by competing tools, while using less than a minute to profile a million reads on the cloud with 65 commodity processors. Microbiome maps produced by Jasper are compact, scalable representations of extremely complex microbial community profiles with numerous demonstrable advantages, including the ability to display latent relationships that are hard to elicit. Finally, experiments show that by using images as input instead of unstructured tabular input, the carefully engineered software, Amber, can outperform other sophisticated machine learning tools available for classification of microbiomes.

TABLE OF CONTENTS

CHAPTER PAGE

1 INTRODUCTION 1

1.1 Motivation and Goals 2

1.2 Research Contributions 4

1.3 Hardware and Software Environments 6

1.3.1 Spark EMR Cluster 7

1.3.2 EMR Cluster Provisioning 8

1.4 Road Map for the Dissertation 8

2 BACKGROUND AND REVIEW 11

2.1 High-Throughput DNA Sequencing 11

2.2 Cloud Computing 14

2.2.1 Spark and MapReduce 15

2.3 Streaming Techniques 18

2.4 Visualizing Data 19

2.5 Space Filling Curves 20

2.5.1 Hilbert Curve 22

2.6 Artificial Intelligence and Machine Learning 24

2.6.1 Deep Learning 24

3 MICROBIAL REFERENCE GENOMES AND CLOUD INFRASTRUCTURE 28

3.1 Reference Genomes 28

3.2 Reference Genome Preprocessing 29

3.2.1 Partitioning and Indexing the Reference Genomes 30

3.3 Cloud Infrastructure 32

3.3.1 AWS Cloud Cluster 32

4 LARGE-SCALE MICROBIAL COMMUNITY PROFILING IN THE CLOUD 35

4.1 Microbial Community Profiling 36

4.2 Approach 37

4.2.1 Spark and MapReduce 37

4.2.2 Streaming Techniques 39

4.3 Methods 42

4.3.1 A “Double” MapReduce 42

4.3.2 Reference Genome Preparation 44

4.3.3 Cluster Provisioning 45

4.4 Results and Discussion 46

4.4.1 Benchmarking the Mapping of Reads 46

4.4.2 Mapping Reads on the Cloud 46

4.4.3 Comparing Flint to Existing Tools 47

4.4.4 Reference Genome Collections 48

4.4.5 Experimental Setup 51

4.4.6 Measuring Accuracy Using Simulated Datasets 52

4.4.7 Human Metagenomic Samples 54

4.4.8 Scaling to Larger Reference Collections 58

4.5 Conclusion 62

5 MICROBIOME MAPS: HILBERT CURVE VISUALIZATIONS OF MICROBIAL COMMUNITY PROFILES 63

5.1 Introduction 63

5.2 Approach 64

5.2.1 Space-Filling Curves 66

5.2.2 The Hilbert Curve 66

5.2.3 Ordering the Genomes Along the Space-Filling Curve 68

5.3 Methods 69

5.3.1 Microbial Neighborhoods 70

5.4 Results and Discussion 77

5.4.1 Comparison to Other Methods 77

5.4.2 Metagenomic Visualizations 78

5.5 Summary and Conclusion 81

6 APPLICATIONS OF MICROBIOME MAPS 82

6.1 Existing Tools for Visualizing Microbial Profiles 83

6.2 Overview of Experiments with Microbiome Maps 84

6.2.1 Experiments with Orderings 84

6.2.2 Reference Genome Collections 92

6.3 Processing the Microbiome Datasets 94

6.3.1 HMP Sample Processing 94

6.3.2 Chronic Kidney Disease Sample Processing 97

6.3.3 Infant Gut Sample Processing 98

6.4 Visualizations with Microbiome Maps 99

6.4.1 HMP Taxonomic Ordering 99

6.4.2 HMP Labeled Ordering 101

6.5 Visualizations - CKD Study 104

6.5.1 CKD Taxonomic Ordering 105

6.5.2 CKD Labeled Ordering 106

6.6 Variations on Microbiome Maps 119

6.6.1 Differential Profiles 119

6.6.2 Animated Movies 119

6.6.3 Composite Maps 120

6.7 Metagenomic Hilbert Inspector 123

7 CHARACTERIZING MICROBIOMES WITH CONVOLUTIONAL NEURAL NETWORKS 125

7.1 Introduction 125

7.2 Approach 126

7.2.1 Convolutional Neural Networks 126

7.2.2 U-Net 129

7.3 Methods 132

7.3.1 Convolutions for Microbial Neighborhoods 132

7.3.2 The Amber CNN Model 134

7.4 Results and Discussion 138

7.4.1 Comparison to other Classifiers 138

7.4.2 Training with HMP Body Site Maps 142

7.4.3 Training with Chronic Kidney Disease Maps 150

7.4.4 Training with No Borders 154

7.4.5 Training with Two Datasets 154

7.5 Profile Classification Comparison 156

7.6 Summary & Conclusion 158

8 CONCLUSION 160

8.1 Microbial Community Profiling in the Cloud 160

8.2 Visualizations of Microbial Community Profiles 161

8.3 Characterizing Microbiomes 163

8.4 Future Work 164

8.4.1 The Future of Flint 164

8.4.2 Peering Down the Road for Jasper 165

8.4.3 Variations on the Theme of Amber 172

APPENDIX 189

VITA 190

LIST OF TABLES

TABLE PAGE

4.1 Reference Database Partitions. Average size of a partition ("Partition Disk Size (GB)") after splitting the reference collection, along with estimated running times for aligning 66k HMP reads against the partition ("Bowtie2 Map Time"). 46

4.2 HiSeq Synthetic Datasets. Average alignment times and alignment rates for three synthetic datasets aligned against Ensembl's 44K bacterial genomes. Alignment rate measures the number of paired reads that were correctly aligned, and sensitivity is the proportion of aligned paired reads that were mapped correctly to the genome from which they were generated. Evaluations were performed on a 64 worker-node Spark cluster. Note that the number of reads aligned can be obtained by multiplying the number of paired reads in the first column by 2. . 53

4.3 Initial Cluster Benchmarks. Average alignment times on a 64 worker-node cluster for a set of randomly selected reads from a HMP throat sample. The number of alignments column contains the output alignments that were generated by each set of reads; these alignments were processed as soon as they were produced and were not stored, therefore minimizing the local storage requirements necessary for profiling metagenomic samples. 55

4.4 HMP Sample Analysis. Diversity classes were established based on the number of unique genera in 753 HMP samples. Three samples were selected from each diversity class and analyzed in a 64 worker-node cluster. Alignment execution time measures the total time to align all the sample reads against Ensembl's 44K bacterial genomes. The "Streamed Shards" are the number of 250K read sets that are streamed into the cluster, and the "Alignments per Shard" is the average number of alignments produced by each shard. 56

6.1 HMP Samples. Eight (8) body sites were selected from the HMP (version 1 project repository) based on their representative number of samples (how many samples each body site contained), as well as the number of DNA sequencing reads that passed a HMP complexity filter (MAPQ score > 20) for each sample. 95

6.2 Infant Gut Samples. Fifteen days of a patient's antibiotic treatment (Vancomycin and Ticarcillin-Clavulanate). Day 28.2 through day 34.5 are before treatment, while day 37 through 64.5 are after treatment. . 98

7.1 HMP Classification Report (12 Classes). Evaluation of the performance of the Amber CNN model in classifying the twelve classes of the HMP dataset. The model performs very good for many of the classes, but fails notably for a couple of classes (those with an F1 score of 0.00). Classes with an F1 score of 0.00 should probably be dropped, as their number of samples is not enough to be used. The "No. Samples" column contains the number of input samples for each class. The "Precision" column contains the proportion of positive classifications that was correct, the "Recall" column contains the proportion of actual positive classifications, and the "F1 Score" column contains the harmonic mean of the precision and recall columns. 147

7.2 HMP Classification Report (8 Classes). Evaluation of the performance of the Amber CNN model in classifying a reduced set of eight classes of the HMP dataset. The model performs very good achieving a training accuracy of 0.99% and a testing accuracy of 0.94%. 148

7.3 CKD Classification Report. Evaluation of the performance of the Amber CNN model in classifying the six classes of the CKD dataset. The model performs best when discriminating among the CKD 2 and CKD 3 classes, and worst for the CKD 5 class. The "No. Samples" column contains the number of input samples for each class. The "Precision" column contains the proportion of positive classifications that was correct, the "Recall" column contains the proportion of actual positive classifications, and the "F1 Score" column contains the harmonic mean of the precision and recall columns. 153

LIST OF FIGURES

FIGURE PAGE

1.1 Cluster layout of the basic distributed architecture in our application. The Bowtie2 mapper is pre-installed in each worker node during the cluster provisioning step. Reference database partitions ("Index") are also copied into each worker node. 7

2.1 Cost of Sequencing 1 Million DNA Nucleobases. NHGRI estimated costs through the years (2001-2019) of generating 1 million DNA nucleic bases (A, T, G, C). Costs only include wet-lab resources, and does not reflect costs of analyzing the generated data [111]. 12

2.2 Cost of Sequencing a Human-Sized Genome. NHGRI estimated costs from 2001-2019 of generating a human-sized genome, or of generating over 3 billion DNA nucleobases [111]. 13

2.3 RefSeq Bacteria. Number of unique accession identifiers in NCBI's RefSeq database. The blue line tracks the growth of new bacterial species submitted to the RefSeq project database. The red line, "Total Accessions", tracks the overall number of unique identifiers for bacterial sequence records in RefSeq. The orange line, "Nucleotides", tracks the number of unique nucleotide bacterial records derived from the International Nucleotide Sequence Database Collaboration (INSDC) project. The green and purple lines track the number of unique records for bacterial "Transcripts" (mRNA sequences) and bacterial "Proteins" in RefSeq. 14

2.4 Distributed Execution Framework. Typical layout of a distributed computational cluster such as the ones available in AWS. The Master node contains the Driver Program, a piece of software that contains the main logic. The Cluster Manager handles communication between the master node and any Worker Nodes, whose primary task is to execute jobs. Each Worker Node is usually a cheap, off-the-shelf machine that contains moderate resources. The power of the such a setup is that all the Worker Nodes run at the same time, and developers can utilize this parallelization to process very large datasets quickly and at reduced costs [153]. 17

2.5 Visualizing Microbial Community Profiles with Bar Charts. An analysis of the structure and function of healthy human microbiomes. The bar charts do a great job of summarizing large amounts of data for the conditions in the study, but details are lost for individual samples. Figure 2 from the paper by Huttenhower et al. [72], and reproduced with permission. 20

2.6 Circos Plot for Microbial Ecology. A Circos diagram showing the relationships between genus-level microbes in solar panels. A Circos plot uses a circular backbone structure to display entities, and ribbons drawn between pairs of entities to display relationships. Figure 5 from Porcar et al. [121], and reproduced with permission. 21

2.7 Microbial Ecology Sankey Diagram. A taxonomic analysis using a sankey diagram showing a hierarchical structure of microbes. Figure 3 from Porcar et al. [121], and reproduced with permission. 22

2.8 Hilbert Curve Levels. First four iterations (levels) of the Hilbert curve show the 2D square recursively partitioned into 4 sections at each iteration (level) of the curve. 23

2.9 Artificial Intelligence, Machine Learning, and Deep Learning. The relationship between the fields of Artificial Intelligence, Machine Learning, and Deep Learning reproduced and based on the book by François Chollet, Deep Learning with Python [31]. 25

2.10 Character Classification. A stylized depiction of a Deep Learning neural network for classifying character letters (“A”, “B”, “C”, etc.) Each layer in the model can learn to represent the network’s input into increasing levels of relevant and abstract representations. Inspired by Figure 1.6 from the book by François Chollet, Deep Learning with Python [31]. 26

3.1 Bacterial genome reference database partitions. Red labels indicates the average disk size in gigabytes (GB, y-axis) of a single partition when partitioning the full reference database into N-number of partitions (x-axis). 31

4.1 Overview of the Flint System. Reference genomes are partitioned so that a large reference set is be distributed across a Spark cluster, and the number of partitions matches the number of worker nodes. Samples are streamed into the cluster to avoid storage overheads as shards of 250K reads. Reads are aligned to the distributed reference genomes using a double MapReduce pipeline that continually updates metagenomic profiles as samples streamed into the cluster. Read alignments are never stored, and processed by each worker node as soon as they are produced. 41

4.2 MapReduce Workflow. Metagenomic samples can be streamed in from a distributed filesystem into the cluster were they are stored in an RDD. The first map step generates alignments through Bowtie2 and feeds its resulting pairs to the first reduce step, which aggregates the genomes that a single reads aligns to. The second map step generates read contributions that are used in the second reduce step to aggregate all the read contributions for a single genome. This results in an abundance matrix containing the abundances of each genome in the sample being analyzed. 43

4.3 Bowtie2 Partition Map Time. Run times for aligning 66k HMP reads using Bowtie2 against different reference database partitions. 47

4.4 Kraken2 Comparison. Community abundance profile comparison for an HMP anterior nares sample (SRS019067) generated by Flint and Kraken. Red branches are lineages identified only by Flint, blue branches are lineages identified by Kraken only, and green branches are lineages identified by both. The outermost ring represents the species identified by either Flint (red), Kraken (blue), or both (green). 49

4.5 Phylogenetic tree showing taxa identified by Flint using 44K Ensembl bacterial genomes (blue), and using 5K NCBI's Genomes references (red) with an input of 1M randomly selected reads from the HMP anterior nares sample (SRS015996). Genomes are considered as identified if the average coverage in their genomic sequence is 80% or more. . . 50

4.6 Initial Benchmarks. Figure A displays the running time for 12 paired-end read datasets on a 64 worker-node cluster. These 12 datasets were used to estimate the optimal number of reads that a 64 worker-node cluster could handle without memory saturation or network overloading. Note that while 1 million paired-end reads can be mapped in 67 seconds against 44K bacterial strains, it is not ideal to ingest 1M reads at once as the cluster's memory will be overwhelmed with the alignments that are produced. Figure B displays the logarithmic running time of the 12 datasets, and the 250K paired-end read dataset was chosen as a good trade-off between speed and resource availability. . . 57

4.7 Modified Cluster Architecture. Support for larger reference collections such as RefSeq could be achieved by having each worker node in the cluster work on multiple partitions, rather than a single one as is the case in the current architecture. 60

5.1 Hilbert Curve Visualization of Metagenomic Samples. (A) The first five iterations of the Hilbert curve: the "Level 1" curve is obtained by connecting the centers of the four initial squares as shown; the Level k curve is obtained by a recursive partitioning of each square from Level k-1, creating four Level k-1 curves and connecting them as outlined by the "Level 1" curve, rotated appropriately. At level k, the original square is divided into $2^k \times 2^k$ small squares, each of whose centers is visited by the Level k Hilbert curve. (B) A representative image of a mWGS Buccal Mucosa sample (SRS045254) from the Human Microbiome Project (HMP) created using a "taxonomic ordering" of 44K reference genomes from the Ensembl database. The intensity of each position in the image represents the abundance of one microbial genome. Groups of segments are labeled by the groups induced by the ordering of the taxa on the Hilbert curve. 65

5.2 Detailed Hilbert Curve Levels. First four iterations (levels) of the Hilbert curve show the 2D square recursively partitioned into 4 sections at each iteration (level) of the curve. 67

5.3 Taxonomic Ordering. A Taxonomic Ordering of 44,048 microbial reference genomes from Ensembl Bacteria. Microbial neighborhoods are drawn based on a taxonomic tree for microbial classification. The image depicts the distribution of Genera in the reference genome database, and the size of each neighborhood is consistent with the number of genomes that belong to it. (B) A taxonomic hierarchical tree with three (3) levels of rankings for the genomes in the reference database: Genus, Species, and Strain. The taxonomic tree is linearized to create a 1D linear order for the tree's leaves (Strains). (C) The 1D linear order is then laid out onto a 2D plane using a Hilbert curve, which creates microbial neighborhoods of related taxa. 69

5.4 Labeled Ordering. (A) Samples are processed in a M x N matrix that contains M labeled samples, and N microbial taxa. The user specifies the ordering of the biological conditions that the M samples belong to. For each labeled grouping, the taxa with the highest mean relative abundance is identified and used as an anchor for a linear order in the Hilbert curve. (B) Resulting Hilbert curve image for 3 samples (SRS024557, SRS045254, SRS063478) for a buccal mucosa condition, built using an averaging ordering scheme from 8 HMP body sites. . . 71

5.5 Chronic Kidney Disease Stages. Hilbert curve visualizations for 16S samples of five stages of chronic kidney disease, along with a control sample. Each panel represents the mean relative abundance of 3 samples for each stage, and displays 5,127 species. Regions marked A shows a group of microbes that appear in all stages, while region marked B appears in almost all stages except CKD3 (absent), and Control (lowered abundance). 76

6.1 Tongue Dorsum Profile. A tabular representation of a typical community abundance profile as generated by the Flint software. Only the first 10 rows out of 44K are shown. 84

6.2 Tongue Dorsum Profile - Pie Chart. A pie chart of the abundance values for the full profile of Figure 6.1 generated by the Tableau [141] software. 85

6.3 Tongue Dorsum Profile - Bubble Plot. A bubble plot of the abundance values for the tongue dorsum profile of Figure 6.1. The size of the bubble represents a relative abundance value of the profiled genome at the strain level (note that labels display the genus and species, and are truncated for display purposes). Plot generated by the Tableau [141] software. 86

6.4 Metagenomic Profile - Krona Plot. A screenshot of an interactive Krona plot displaying the abundance profile of the tongue dorsum sample from Figure 6.1. The black bands at the bottom and bottom-right of the plot are caused by the need to pack a large number of reference genomes in a small section of the circle. 87

6.5 16 Samples - Stacked Bar Chart. A stacked bar chart representation of a community abundance profile for 16 time points generated by the Tableau software [141] for the study from [51]. 88

6.6 Taxonomic Ordering. Basic layout of the taxonomic ordering. Each unit of the map is assigned to one taxon in the taxonomic ordering. Each clade in the taxonomy tree appears consecutively on the taxonomic ordering and gets assigned a region in the map. As shown, these clades can be outlined, if needed, creating microbial neighborhoods of taxonomically related taxa. The size of an outlined region is proportional to the number of species in that taxonomic clade. 89

6.7 Taxonomic Ordering at the genus level. Clades at the genus level are outlined as microbial neighborhoods. The sizes of the neighborhoods correspond to the number of child species in the taxonomy tree. . . . 90

6.8 Labeled Ordering. A labeled ordering of an example profile that contains 7 biological conditions. The ordering is for 44K genomes from Ensembl, and they are ordered based on their mean relative abundance in the samples with that condition. 91

6.9 HMP Samples. Sample class distribution for the collection from the Human Microbiome Project. Four body sites dominate the distribution – Buccal Mucosa, Gastrointestinal Tract, Nares, and Posterior Fornix. 96

6.10 Sample SRS024557 from the HMP Buccal Mucosa set. 100

6.11 Sample SRS014683 from the HMP Gastrointestinal Tract set. 101

6.12 Sample SRS011105 from the HMP Nares set. 102

6.13 Sample SRS015061 from the HMP Palatine Tonsils set. 103

6.14 Sample SRS014343 from the HMP Posterior Fornix set. 104

6.15 Sample SRS017088 from the HMP Supragingival Plaque set. 105

6.16 Sample SRS013948 from the HMP Throat set. 106

6.17 Sample SRS013502 from the HMP Tongue Dorsum. 107

6.18 Buccal Mucosa Average Abundance 108

6.19 Gastrointestinal Tract Average Abundance 109

6.20 Nares Average Abundance 110

6.21 Palatine Tonsils Average Abundance 111

6.22 Posterior Fornix Average Abundance 112

6.23 Supragingival Plaque Average Abundance 113

6.24 Throat Average Abundance 114

6.25 Tongue Dorsum Average Abundance 115

6.26 CKD Stage 1, Taxonomic Ordering 116

6.27 CKD Stage 2, Taxonomic Ordering 116

6.28 CKD Stage 3, Taxonomic Ordering 116

6.29 CKD Stage 4, Taxonomic Ordering 117

6.30 CKD Stage 5, Taxonomic Ordering 117

6.31 CKD Control Normals: Taxonomic Ordering 117

6.32 CKD Stage 1 & 2, Labeled Ordering 118

6.33 CKD Stage 3 & 4, Labeled Ordering 118

6.34 CKD Stage 5 & Normal Control, Labeled Ordering 118

6.35 Differential Abundance. P-value microbiome map of an analysis of differentially abundant taxa in buccal mucosa and gastrointestinal tract. Color intensity corresponds to p-value significance with lower p-values being more bright. Only p-values ≤ 0.05 are displayed. 120

6.36 Animated Microbiome Map. Selected frames of an animated time-series visualization of 12,116 strains for a single patient from [51]. Panel (A): zoomed regions of the Enterococcus neighborhood as it progresses through the antibiotic response. Panel (B): Full resolution animated microbiome map is available at the project's repository via links from `biorg.cs.fiu.edu/jasper` 121

6.37 Buccal Mucosa Average Map. A composite map created from the averaged abundance profiles of six HMP buccal mucosa samples. Note the Streptococus neighborhood, a signature region for buccal mucosa samples. 122

6.38 Image Viewer. The Jasper image viewer utility being used to inspect a CKD image. The hand (top) clicked on the Streptomyces neighborhood, and the program displayed the image segment Streptomyces sp. gba 94-10 as the species that was clicked on. 124

7.1 Convolutional Operations. Input images are turned into an input volume and small patches (convolutional kernels) and are slid across the input. The kernels compute the dot product with the input values at the same location, and the result is a output feature map of filters applied to the image. 129

7.2 Higher Levels of Abstractions. A simple look at how a CNN would arrive at the answer for a Dog. Initial layers will learn to recognize local features such as lines, curves, and blobs of color; while succeeding layers will learn to recognize legs, ears, and snouts. 130

7.3 U-Net Architecture. The Ronneberger U-Net used for image segmentation. The architecture gets its name from the distinctive "U" shape formed by its two paths: the left side performs the convolutional operations, while the right side performs the deconvolutional operations. The output of a U-Net is a feature segmentation map. 131

7.4 Convolutions on Microbial Neighborhoods. A microbiome map (panel (A)) created using the Hilbert curve, and scaled for display purposes. The native resolution for each image is 256×256, which corresponds to a Hilbert curve at level eight. Panel (B) shows how the Hilbert curve is recursively built by recursive partitioning of a square into four, and the first three levels of the curve are shown. Panel (C) shows the first stage of our convolutional architecture that we use for our microbiome classifier. Patches of size (5×5) are used in the initial convolutional layers, and they create 64 filters which comprise the "Output Feature Map". A `max()` pooling layer is used to downsample the feature map in the following round of convolutional layers. 133

7.5 Amber CNN Architecture I. The primary architecture of the Amber CNN model when 44K genomes are profiled. The network starts with the typical CNN architecture of having alternating convolutional and pooling layers (lines 5-23). After the last pooling layer, we introduce a dropout layer (line 25) and flatten its output (line 27). The flattened input is used as input to a densely connected layer (line 29) which feeds into another dropout layer before delivering its output to a softmax layer (line 33) which contains 12 neurons (one for each class in the HMP dataset). 135

7.6 Amber CNN Architecture II. Illustrated version of the Amber CNN model. The model starts with an input volume measuring width $\times$ height $\times$ depth, where the width and height are the corresponding width and height of the input images (256×256 for mWGS, and 128×128 for 16S), and the depth corresponds to the number of channels: three for RGB images, one for grayscale. The model starts by using a 7×7 kernel, and learning 16 filters. Notice that as the model gets deeper and deeper, we learn more and more filters (16, 32, 64, and 128) and the dimensions of the output volumes get smaller and smaller (250, 125, 62, 31, and 15) which is typical for a CNN that is learning higher levels of abstraction with each subsequent layer. 137

7.7 Synthetic Data Classification. Seven classifiers were tested with synthetic datasets created with the scikit-learn library [117], using code adapted from scikit-learn examples [118]. The "Original Data" depicts the original data points (x,y) created for the trials with two classes, A and B, for three datasets. Overall accuracy for each classifier is displayed in each box (label in dark font). 139

7.8 HMP Data. Mean accuracy of the seven classifiers when run with profiles created with Flint. The community profiles contained 44K microbial genomes (features), and 12 classes from HMP human body sites. Panel (A) contains the results of the classifiers with a tabular (numerical) profile used as input. Panel (B) is used for comparison, and displays the mean accuracy of running the Amber CNN with the two orders (taxonomic and labeled) using the same tabular profiles converted to microbiome maps. 141

7.9 Chronic Kidney Disease Data. Mean accuracy of the seven classifiers when run with profiles created with Kraken 2. Unlike the profiles created with Flint, these profiles only contained 5K microbial genomes, and 6 classes (CKD stages plus control). Panel (A) contains the results of the classifiers with a tabular (numerical) profile used as input. Panel (B) is used for comparison, and displays the mean accuracy of running the Amber CNN with the two orders (taxonomic and labeled) using the same tabular profiles converted to microbiome maps. 142

7.10 Training with HMP Data. Accuracy (purple, gray lines) and loss (red, blue lines) for training and testing the Amber CNN with the 328 HMP samples for 100 epochs. While both training (99%) and testing (92%) accuracy are high, the diverging paths of the training and testing loss indicate that the model is overfitting the data. 143

7.11 Augmented microbiome maps. Representative images of performing data augmentation for reducing overfitting. The HMP training dataset used by the Amber CNN was augmented by a set of augmented images that were modified by rotating them, as well as adjusting the luminosity, and transparency properties, as well as RGB color channel levels. 145

7.12 HMP Training Corrected for Overfitting. Accuracy (purple, gray lines) and loss (red, blue lines) for training and testing the Amber CNN (with dropout layers) with 328 augmented HMP samples for 200 epochs (all 12 classes). Training and testing accuracies decreased to 87% and 89% respectively, but there was no separation in the trajectories of the training and testing losses. 146

7.13 HMP CNN Activations. Visualization of the convolutional layer's activations for the HMP buccal mucosa class. The buccal mucosa class is characterized by having a significantly abundant Streptococcus genus, and we can see that the model is learning that in the initial convolutional layers (top) when it learns 16 and 32 filters. 149

7.14 Bull Abstractions, 1945. A series of lithographs by Pablo Picasso to represent higher levels of abstractions of a bull. The images show that to represent a bull, it is not necessary to learn fine details, but sometimes, a simple set of lines suffices 150

7.15 Training with CKD Data. Accuracy (purple, gray lines) and loss (red, blue lines) for training and testing the Amber CNN with the 200 HMP samples for 200 epochs. For the CKD dataset, the training accuracy was high (99%), however, the testing accuracy was low (40%). Note the diverging paths of the training and test losses, which indicate that the model is overfitting the data substantially. 151

7.16 CKD Data Corrected for Overfitting. Accuracy (purple, gray lines) and loss (red, blue lines) for training and testing the Amber CNN (with dropout layers) with 200 augmented CKD samples for 200 epochs. Training accuracy was retained at 99%, while the testing accuracy dropped slightly to 39%. Overfitting was reduced, but not completely eliminated, which suggests that more training data should be used. . 152

7.17 Alternative Training Images. Training and testing accuracies were much higher when training with microbiome maps with borders around neighborhoods. 155

7.18 Accuracy with No Border. Training and testing accuracy when evaluation the Amber CNN for 160 epochs with the HMP dataset, but the microbiome maps contained no visible neighborhood borders. Training accuracy finished at 94%, while testing finished at 84%. 156

7.19 Loss with No Border. Training and testing loss when evaluation the Amber CNN for 160 epochs with the HMP dataset, but the microbiome maps contained no visible neighborhood borders. 157

7.20 Profile Comparison. Testing accuracies for seven classifiers and the Amber CNN model. Panel (A) contains seven classifiers whose input is numerical tab-separated data, and panel (B) contains results for the Amber CNN model which uses microbiome map images as input. "Neural Net (baseline)" represents the simplest model for a neural network which consisted of a single 10-neuron layer trained for 10 epochs. The "Neural Net" classifier consisted of 10, 100-neuron layers trained for 1,000 epochs. 158

8.1 A 3D Hilbert Curve. The top two volumes represent two variants of the first iteration of the curve in 3D space, while the bottom two volumes represent the second iteration of the top volume above it. 167

8.2 Two Maps Side by Side. A visual DIFF of two microbiome maps for a HMP buccal mucosa (left) and anterior nares (right) samples. 168

8.3 Difference Slices. Overlaid differences in Kaleidoscope of two microbiome maps for a HMP buccal mucosa (left) and anterior nares (right) samples. The two images are layered on top of each other, and the pendulum in the middle (black line) allows the user to see what is "behind" (right image) and "above" (left image) the two images. . . . 169

8.4 Color Difference. A Kaleidoscope color difference image of two microbiome maps for a HMP buccal mucosa (left) and anterior nares (right) samples. The two images are layered on top of each other two display what is common on both images (green) and what is different (dark areas), with red-tinted areas being what is unique to the left image, and blue-tinted images being what is unique to the right image. . . . 170

8.5 Zoom. Kaleidoscope can zoom into the image from Figure 8.4, and users can navigate by using the navigation map in the top-right. . . . 171

CHAPTER 1

INTRODUCTION

Microbes are everywhere [171], and a microbiome is the collection of all the genetic material of all microbes that inhabit a particular environmental niche such as the human body, earth soil, and the water in oceans and lakes. When we think of microbes, we usually think about the bad ones, pathogens, the ones that cause disease [3] and make us sick. But not all microbes are bad: some help us digest foods [100, 101], while others keep us safe even before we are born [1]. Studying and understanding the relationships between us and the microbes that inhabit us is a crucial step in understanding our relationship to our surroundings as many human diseases and environmental disasters are linked to changes in microbiome compositions [63, 80, 168, 175].

Metagenomics is one of the the lens through which we study microbiomes. Profiling a microbiome is a critical step to probe the microbial diversity and to tell us what microorganisms are present, and in what proportions. When combined with other omics studies, it can serve as an instrument to go beyond microbial biodiversities and to study their complex relationships to their hosts and environments.

A powerful tool for creating microbial community profiles from microbiome samples is high-throughput DNA sequencing [105]. Advances in recent years have steadily reduced the cost of sequencing, and have led to the generation of an increasing number of extremely large and complex metagenomic datasets [13, 26]. Current approaches for generating community profiles from metagenomic whole-genome sequencing (mWGS) datasets do not "scale", often opting to deal with: (*i*) a smaller, curated collection of microbial reference genomes, and/or (*ii*) mWGS datasets that contain a small amount of sequenced genetic material. However, the arrival of low-cost DNA sequencing has created a "data deluge" – the genome collections have been growing exponentially, as have the mWGS dataset sizes and cohorts. To put things in perspective, the RefSeq database (v.92) [109, 113] contains 173,055 bacterial genomes, comprising over 400 GB of data (see Figure 2.3); also, the sequence repository for the Human Microbiome Project (HMP) [71] contains many terabytes of sequence data from thousands of healthy human microbiome samples. The consequence of readily available low-cost DNA sequencing has been that microbiome samples are out-pacing the computational resources that are required to analyze them.

Standard genomics and transcriptomics analyses for DNA sequencing datasets usually begin by aligning sequencing reads to a reference genome [147, 158], and producing abundance counts [148]; but in microbial community profiling analyses, the alignment step is performed against a collection of reference genomes that can be extremely large, slowing down the entire operation. Fast k-mer based strategies have been developed to improve metagenomic profiling [132, 167], but they achieve high speeds by confining their searches to small, curated reference databases created from a subset of selected genomes from a larger database. Alternative k-mer index-building strategies exist, but only at species-level resolutions [177]. LiveKraken [142] is a real-time classification tool for metagenomics, and is based on the Kraken [167] method for profiling microbiome samples. It is implemented as a instrument module that classifies reads as soon as they are produced by the sequencer, and uses the same approach as the HiLive [95] mapper, but extends it to metagenomic datasets. Since LiveKraken uses a k-mer based index for the reference genomes, its scalability is limited by the size of the index that can be stored, and is therefore not well suited for large genome collections [109].

1.1 Motivation and Goals

The analysis of massive metagenomic datasets poses a computational challenge as current tools generate large intermediate results, and require the creation of enormous indexes to catalog the ever increasing sets of reference genome collections [156]. New analysis paradigms are needed that leverage modern streaming architectures for processing large amounts of data, and that use cloud-based distributed computational pipelines. Such computational strategies must be matched with visualization strategies to create interpretable results from all the data that low-cost DNA sequencing is generating. The goal of this dissertation is to address these needs, and to create powerful computational tools that create interpretable results from microbial community profiles. This dissertation focuses on three questions for processing and analyzing metagenomics DNA sequencing data.

Profile: How to design a scalable distributed computational framework for profiling large mWGS datasets in a commercial cloud environment using a large collection of reference microbial genomes?

Microbial community profiling for mWGS datasets usually starts by aligning the sequenced DNA reads to a collection of microbial reference genomes. Current profiling tools are designed to work against a small representative collection of microbial genomes, and do not scale well to larger genome collections. However, large reference genome collections are capable of providing a more complete and accurate profile of the bacterial population in a microbiome sample. Chapter 4 presents Flint, a scalable, efficient, and affordable approach to this problem that runs on commercial cloud platforms, thus bringing big data solutions within the reach of laboratories with modest resources.

Visualize: How to visualize microbial community profiles from microbiome samples at high resolution?

Microbial community profiles from mWGS datasets synthesize information from billions of sequenced DNA reads coming from the genomes of the thousands of microbes present in a microbiome. Analyzing and understanding these profiles can be a challenge since the data they represent is complex. Particularly challenging is their visualization and interpretability, as existing techniques are inadequate when the number of identified taxa is in the thousands. Chapter 5 presents Jasper, a technique for visualizing microbial community profiles succinctly using a space-filling Hilbert curve that transforms community profiles into interpretable 2D images called *Microbiome Maps*.

Characterize: How to build a distributed machine learning framework for characterizing microbiomes and creating interpretable results?

Chapter 7 presents Amber, a custom machine learning framework and architecture for recognizing patterns in microbial community profiles from microbiome samples. The method uses as input the 2D microbial images that are created using the Hilbert curve (Chapter 5), and is able to classify microbiome samples with high accuracy. As described later, by using the Hilbert order imposed on the 2D images, the classifier is able to outperform competing classifiers by learning the unique characteristics, and latent relationships, present in a microbiome.

1.2 Research Contributions

The contributions of this dissertation are as follows: a scalable approach for profiling microbiome samples against a large collection of microbial reference genomes, a scalable technique for visualizing microbial community profiles using a space-filling Hilbert curve, and a machine learning framework based on custom-built artificial neural networks that capitalizes on the properties of scalable visualizations. Finally, we present novel applications of the approaches described above, some of which are explored in detail, while others are only briefly discussed. Specifically, the work in this dissertation resulted in the following software systems:

1. Flint: is a pipeline for fast real-time profiling of microbiome samples. It is designed to process large mWGS datasets mapped against a large collection of reference genomes. It takes advantage of parallel computational pipelines and streaming architectures to quickly align DNA reads against a reference collection of 44K microbial reference genomes. Flint runs on Amazon's Elastic MapReduce cloud service, and is able to profile 1 million Illumina paired-end reads against 44K genomes on 64 machines in 67 seconds – an order of magnitude faster than the state of the art.

2. Jasper: is a system that offers techniques for creating *Microbiome Maps* by transforming microbial community profiles into a 2D image using a space-filling Hilbert curve. We describe how we convert mWGS, and 16S rRNA gene sequencing, microbial community profiles into synthetic 2D images using two different arrangements: a *Taxonomic Ordering* which creates taxonomic neighborhoods of related taxa, and a *Labeled Ordering* which creates neighborhoods of related taxa relevant to specific biological conditions.

3. Amber: is a machine learning framework for characterizing microbiome samples from images constructed from microbial community profiles. Our framework takes advantage of the custom metagenomic Hilbert images created by Jasper and as a first step, we implement a custom convolutional neural network (CNN) that can accurately classify samples using a custom network architecture tuned to perform well with metagenomic Hilbert images as inputs. We show that the microbiome maps created by Jasper give Amber an advantage over other classifiers that use raw tabular data as inputs.

This dissertation addresses the lack of powerful computational tools for profiling large microbiome samples against large collections of bacterial genomes. In addition, this proposal also addresses the need to develop improved analysis pipelines that produce results that are readily interpreted and deciphered without the assistance of expert data scientists.

1.3 Hardware and Software Environments

Here we describe the software and hardware environments used in our experiments. Some of the concepts and technical terms used in the description here are reviewed and described in more detail in Chapter 2.

All software development and experiments with data sets were carried out on multiple systems spanning single machines, local servers, and remote clusters hosted on third-party cloud providers. A GitHub repository is available at `https://github.com/camilo-v` that contains all the source code, along with documentation and instructions on how to run the software, and set up any computational clusters.

All software development was carried out on a MacBook Pro, running macOS Mojave (version 10.14) with an Intel Core i7 processor, 16 GB of RAM, and 512 GB of storage. Initial testing and experiments were also carried out in a server machine at the Computer Science Department at Florida International University [133]; the machine has 792 GB of RAM, and 48 Intel Xeon processors (E5-2650, 2.20 GHz). Testing and development in a cloud environment was performed in Amazon Web Services [11] using the Amazon S3 storage service, Elastic Compute Cloud (EC2), and Elastic Map Reduce (EMR) service. Multiple cluster configurations were tested in EMR, and the instance type "`c4.2xlarge`" was selected for offering a good balance of performance and cost (see section 1.3.1 for more information). All the necessary Python packages, and external libraries, are documented in [52], as well as the default runtime parameters and configurations for launching the EMR cluster. The project websites for the three software systems resulting from this dissertation are located at the following three URLs: `https:/biorg.cs.fiu.edu/flint`, `https:/biorg.cs.fiu.edu/jasper`, and `https:/biorg.cs.fiu.edu/amber`.

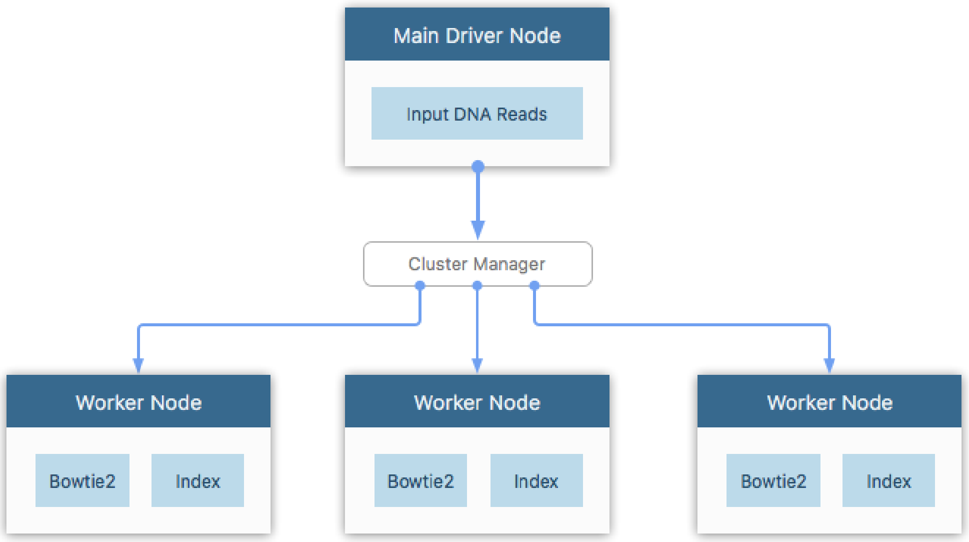


Figure 1.1: Cluster layout of the basic distributed architecture in our application. The Bowtie2 mapper is pre-installed in each worker node during the cluster provisioning step. Reference database partitions ("Index") are also copied into each worker node.

1.3.1 Spark EMR Cluster

The computational framework for the work described in Chapter 4 was primarily implemented using the *MapReduce* model [34], and deployed in a cluster launched in Amazon Web Services's [11] *Elastic Map Reduce* (EMR) service. The cluster consisted of multiple worker machines (a computational "worker" *node*), each with 15 GB of RAM, 8 vCPUs (each being a hyperthread of a single Intel Xeon core), and 100 GB of disk storage. Each of the worker computational nodes work in parallel to align the input sequencing DNA reads to a shard of the reference database (Figure 1.1); after the alignment step is completed, each worker node acts as a regular Spark executor node. By leveraging the work of multiple machines working at the same time, we are able to align a large number of reads to a large database of reference genomes much more efficient manner than only using a single powerful machine.

1.3.2 EMR Cluster Provisioning

A Spark [173] cluster was created using the AWS Console with the following software configuration: EMR-5.7.0, Hadoop 2.8.4, Ganglia 3.7.2, Hive 2.3.3, Hue 4.2.0, Spark 2.3.1, and Pig 0.17.0. The cluster was composed of homogeneous machines for both the driver node and worker nodes, and each machine was an Amazon machine instance of type "`c4.2xlarge`". These instances contain 8 vCPUs, 15 GB of RAM, 100 GB of EBS storage, and each cost on average $0.123 USD to run per hour on the "us-east" availability zone on the Spot market as of this writing. Newer instances ("`c5.2xlarge`") are also available for use, but their availability is infrequent in large numbers, in addition to having a higher cost per hour to run.

A custom provisioning bash shell script was used to pre-install all the necessary utilities, security certificates, third-party library/packages, and DNA mapper executables in the cluster; the shell script was included as a "Bootstrap Action" in the EMR provisioning step to copy all the run-time assets into the local file system of each worker node so that it was available to each worker during runtime. Once the cluster was created, each reference database shard was copied into each worker node directly from the S3 bucket storage. Deploying the EMR cluster was relatively fast, and the entire process took about 15 minutes using 64 "`c4.2xlarge`" instances. The machines were acquired from Amazon's EC2 Spot Market [10], which offers machine instances at reduced cost.

1.4 Road Map for the Dissertation

The dissertation is organized as follows. Chapter 2 contains a review of the relevant literature on the methods, datasets, and systems used by researchers for the analysis of microbiome samples. These methods, datasets, and systems span the spectrum from computational pipelines and infrastructure, to visualization techniques for displaying community profiles, and also machine learning frameworks and architectures for characterizing microbiomes.

Chapter 3 describes the primary data foundation for our projects: a large collection of microbial reference genomes from the Ensembl repository [42]. These reference genomes are the bedrock upon which Flint (Chapter 4), Jasper (Chapter 5), and Amber (Chapter 7) are built on. Chapter 3 contains detailed information on how we process the microbial genome collections, and the steps that are required for preparing the genomes so that we can use them to profile microbiomes samples.

Chapter 4 describes the Flint system which was developed to support this work, and published in [153]. Flint is a large scale microbial community profiling system that runs on cloud infrastructure such as Amazon Web Services [11], and can profile millions of whole-genome sequencing reads very fast and at reduced cost. The community abundance profiles from Flint have strain-level resolution for a collection of 44K microbial reference genomes.

Chapter 5 describes the Jasper framework for visualizing microbial community profiles. It describes the basic space-filling curve technique that we use to convert microbial community profiles into 2D image representations that form the basis of a *Microbiome Maps*. The resulting 2D images are easy to interpret, and can synthesize the very complex information present in a microbiome sample.

Chapter 6 contains some applications for the *Microbiome Maps* created by the technique in Chapter 5. These applications are animated movies of time-series microbiome samples that show how different microbiomes behave across time; how microbiomes change from site to site in the human body; and finally, how the microbiome of patients with chronic kidney disease (CKD) progresses through different stages of disease.

Chapter 7 describes a machine learning framework and architecture that can learn and identify microbial patterns present in microbiomes by learning the patterns found in *Microbiome Maps*. We create and train custom artificial neural networks that can learn to recognize patterns in large collections of 2D microbial images generated with the technique in Chapter 5, and also discuss how the interpretability of the microbial community profile models can be a useful device for characterizing microbiome samples.

Lastly, in Chapter 8, we provide some concluding remarks and perspectives, and suggestions for future work.

CHAPTER 2

BACKGROUND AND REVIEW

Metagenomic DNA sequencing datasets are very complex, and their analysis necessitates a collection of tools and technologies that straddle the boundaries between the fields of computer science and biology. This chapter describes some of the underlying technologies and methods that researchers use to generate the samples in laboratories, as well as some of the analytical tools and computational infrastructure that are required to profile, visualize, and characterize them.

2.1 High-Throughput DNA Sequencing

High-throughput DNA sequencing technologies provide powerful tools that can be used for profiling microbiomes [105]. These technologies have revolutionized the study of organisms by giving researchers powerful instruments with which to study their genomes. These technologies became widely available to researchers around 2008 [108] and have become increasingly affordable over the years. The National Human Genome Research Institute (NHGRI) has tracked the costs of sequencing technologies since 2001 [111]. Figures 2.1 and 2.2 show the trajectories of sequencing costs for generating 1 million DNA bases and 3 billion DNA bases (roughly the size of the human genome), respectively. NHGRI notes that sequencing costs have plummeted since about 2008 when high-throughput technologies became widely available, and they superimpose the cost of sequencing with that of Moore's Law [107], or the observed trend of computing hardware to reduce in cost by a factor of two about every 18 months.

The key to Figures 2.1 and 2.2 is that the cost of producing DNA sequencing data keeps out-pacing the costs of the computing resources that are required to analyze

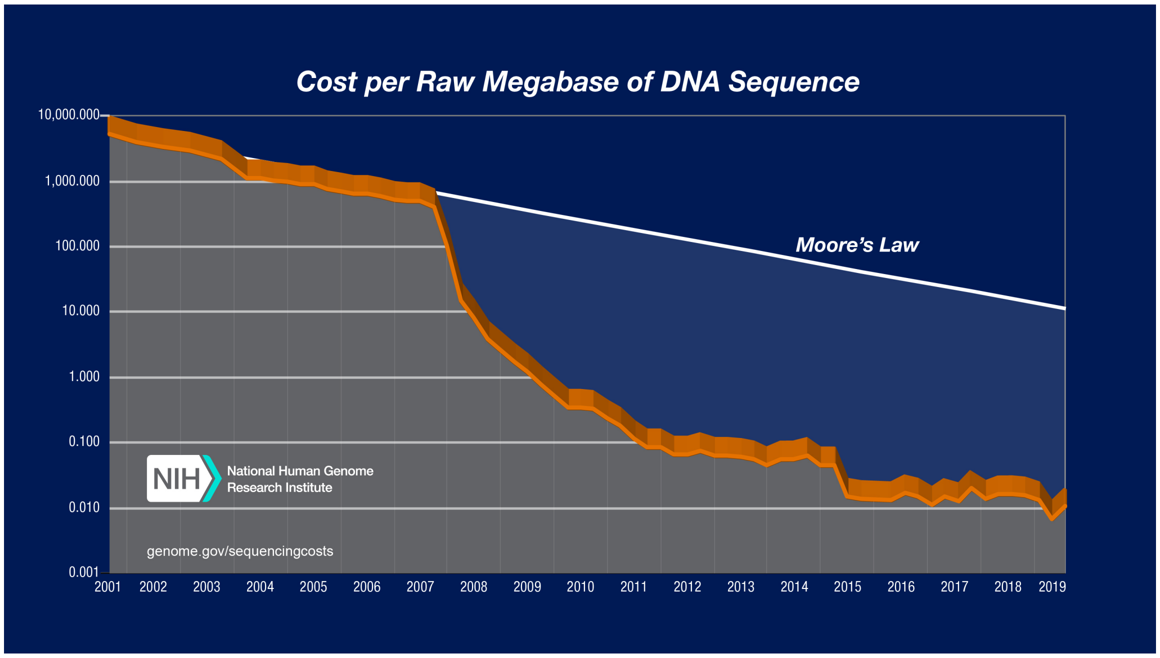


Figure 2.1: Cost of Sequencing 1 Million DNA Nucleobases. NHGRI estimated costs through the years (2001-2019) of generating 1 million DNA nucleic bases (A, T, G, C). Costs only include wet-lab resources, and does not reflect costs of analyzing the generated data [111].

the data, and the general observation is that there is a "DNA data deluge" as DNA sequencing keeps getting cheaper and cheaper.

Reduced sequencing costs has created a boom in the sequencing of new microbial genomes [110, 97], and as of January 6, 2020, the RefSeq Bacteria [113] database from the National Center for Biotechnology Information (NCBI) has cataloged over 60K bacterial species (Figure 2.3).

The types of datasets that high-throughput DNA sequencing technologies create depend on the extraction protocols that are used by laboratories to generate the necessary sequencing libraries that are used as input to the sequencing instruments [13, 67, 105]. Protocols exist for capturing messenger-RNA [108, 138, 158, 162], as well as the raw DNA sequence of an organism [65, 99, 134]. In the context of microbial studies, technologies that capture the raw DNA sequence generate "Whole Genome

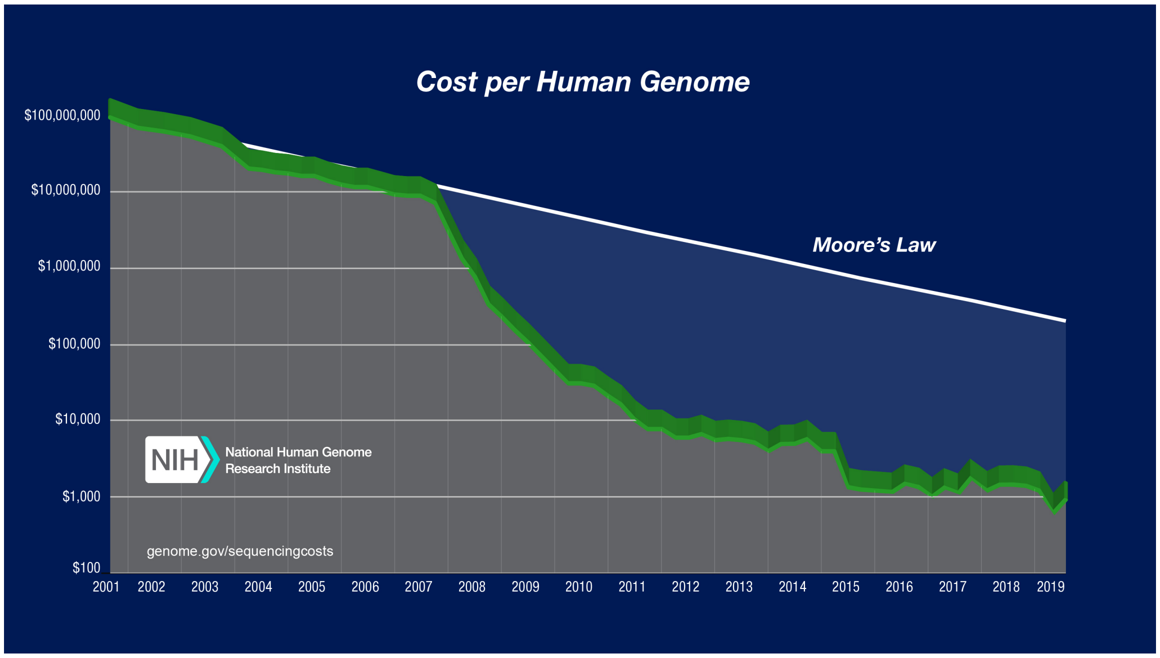


Figure 2.2: Cost of Sequencing a Human-Sized Genome. NHGRI estimated costs from 2001-2019 of generating a human-sized genome, or of generating over 3 billion DNA nucleobases [111].

Sequencing" data, while technologies that capture DNA from gene regions generate "16S rRNA gene sequencing" data. These two technologies, whole-genome and 16S rRNA gene sequencing, capture different parts of a microbe's genome: whole-genome sequencing captures pieces of the microbe's genome, while 16S rRNA gene sequencing captures parts of the 16S rRNA gene. From a analysis perspective, the taxonomic tree lineage that defines relationship among close microbial organisms is the main factor that differentiates the two technologies: whole-genome sequencing datasets provide a higher resolution of detection than 16S rRNA gene sequencing experiments [124]. 16S rRNA gene sequencing experiments are cheaper to produce, and also generate much less data [124], and their advantage is that we can generate a larger number of samples to study a given biological condition. Standard genomics and transcriptomics analyses for sequencing datasets usually begin by aligning sequencing reads to a reference genome [147, 158], and producing abundance counts [148]; but

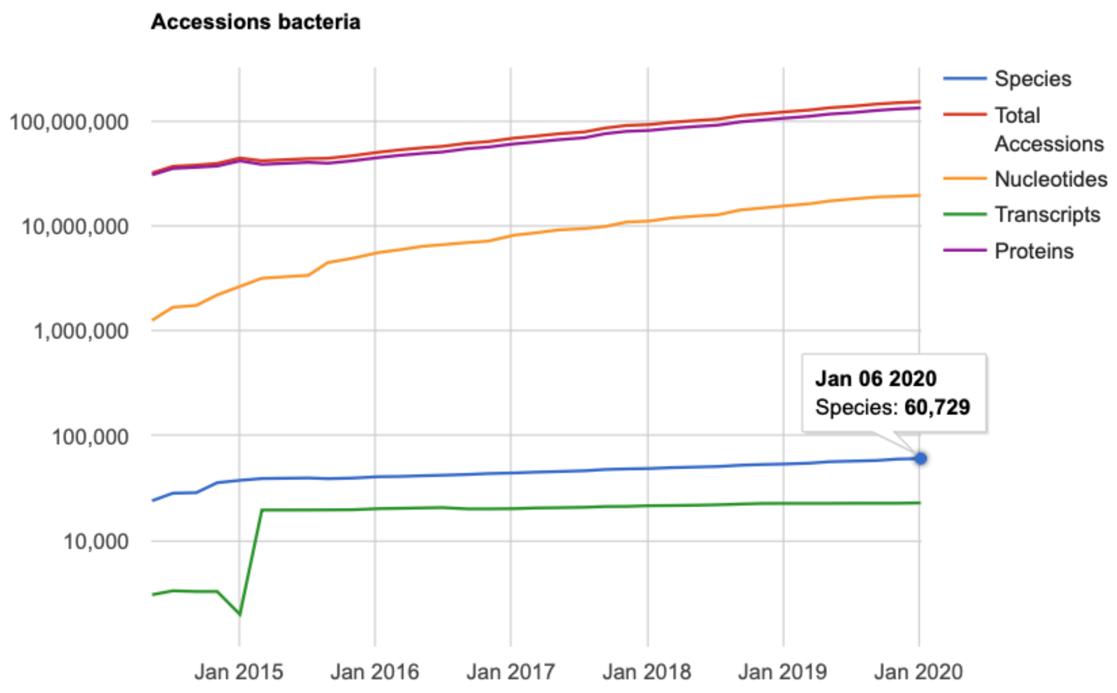


Figure 2.3: RefSeq Bacteria. Number of unique accession identifiers in NCBI's RefSeq database. The blue line tracks the growth of new bacterial species submitted to the RefSeq project database. The red line, "Total Accessions", tracks the overall number of unique identifiers for bacterial sequence records in RefSeq. The orange line, "Nucleotides", tracks the number of unique nucleotide bacterial records derived from the International Nucleotide Sequence Database Collaboration (INSDC) project. The green and purple lines track the number of unique records for bacterial "Transcripts" (mRNA sequences) and bacterial "Proteins" in RefSeq. More information can be accessed at `https://www.ncbi.nlm.nih.gov/refseq/statistics/`.

in metagenomic analyses, the alignment step is performed against a collection of reference genomes that can be extremely large, slowing down the entire operation.

2.2 Cloud Computing

To deal with the barrage of DNA sequencing data, distributed cloud computing platforms and frameworks such as Amazon Web Services [11], Apache Hadoop [143], and Apache Spark [173] have been used by researchers to take advantage of parallel computational pipelines, and economies of scale: large DNA sequencing workloads can be distributed in a computational cluster that is comprised of many cheap, off-the-shelf computational nodes, that is located off-site and requested on demand, i.e., using “cloud services”.

These cloud-based solutions have been successfully used in the past for human genomics [85], human transcriptomics [127], and more recently for microbial community profiling applications [70, 178]. An example of one of these cloud setups is a cluster deployed in AWS that is composed of homogeneous Amazon Machine Instances (AMI) of type c4.2xlarge: These instances contain 8 vCPUs, 15 GB of RAM, 100 GB of EBS storage, and cost on average $0.123 USD each to run per hour through the Spot [10] market. Newer instances (such as those of c5.2xlarge variety) are available for use, but their availability is infrequent in large numbers, in addition to having a higher cost per hour to run.

Another advantage of these cloud-based solutions is their support for software assets and resources that provide researchers with powerful tool-chains with which to build scalable solutions that can handle large datasets. The following sub-sections describe some of the more recent software libraries that are used to deal with large amounts of data in a distributed computational infrastructure.

2.2.1 Spark and MapReduce

The MapReduce model [34] provides a fast computational model for developing programs that execute in parallel across a cluster of machines (see Figure 2.4). The Spark framework [173] extended the MapReduce model by offering a rich API and a computational engine that allows in-memory computation of all steps in a map-reduce pipeline. Both the MapReduce model and the Spark framework have been popular in DNA sequencing contexts as they have been used in speeding up the crucial read-alignment steps in the analysis of single-organism sequencing datasets, as researchers have framed the read-alignment and quantification tasks in terms of *map* and *reduce* operations. For example, Langmead et al. used it to align human sequencing reads using the Bowtie read-mapping utility [86] and searched for single nucleotide polymorphisms (SNPs); while Roberts et al. [127] used it to speed up the quantification of human gene transcripts by the expectation-maximization (EM) algorithm.

Spark has been used in metagenomic analyses [60] for mapping sequencing reads against small genome reference and for clustering metagenomes [125]. Zhou et al. developed MetaSpark [178] to align metagenomic reads to reference genomes. The tool employs Spark's Resilient Distributed Dataset (RDD) [174] to cache reference genome and read information across worker nodes in the cluster. MetaSpark was developed with two reference datasets: a 0.6 GB reference, and a larger 1.3 GB from RefSeq's archived bacterial repository. SparkHit [70] developed by Huang et al. includes a metagenomic mapping utility called "SparkHit-recruiter" that performs much faster than MetaSpark, albeit with similar small sets of reference genomes.

Zhou et al. developed MetaSpark [178] to align metagenomic reads to reference genomes. The tool employs Spark's Resilient Distributed Dataset (RDD) [174] – the main programming abstraction for working with large datasets – to cache reference genome and read information across worker nodes in the cluster. By using Spark's RDD, MetaSpark is able to align more reads than previous tools. MetaSpark was developed with two reference datasets of bacterial genomes: a 0.6 GB reference, and the larger 1.3 GB from RefSeq's bacterial repository. These reference sets are small compared to the 170 GB reference set of Ensembl, and because of MetaSpark's use of an RDD to hold its index, it is unlikely that MetaSpark can scale to use them: the contents of an RDD are limited to available memory, and large reference sets would require correspondingly large memory allocations. It is worth pointing out the RDD memory limitations of MetaSpark in aligning reads: it took 201 minutes (3.35 hours) to align 1 million reads to the small 0.6 GB reference using 10 nodes [178].

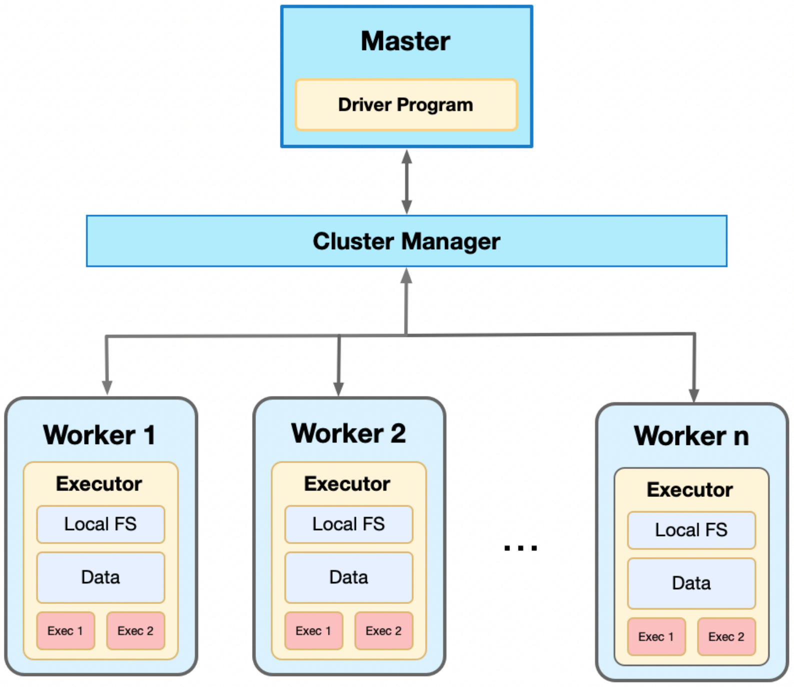


Figure 2.4: Distributed Execution Framework. Typical layout of a distributed computational cluster such as the ones available in AWS. The *Master* node contains the *Driver Program*, a piece of software that contains the main logic. The *Cluster Manager* handles communication between the master node and any *Worker Nodes*, whose primary task is to execute jobs. Each *Worker Node* is usually a cheap, off-the-shelf machine that contains moderate resources. The power of the such a setup is that all the *Worker Nodes* run at the same time, and developers can utilize this parallelization to process very large datasets quickly and at reduced costs [153].

SparkHit [70] was developed by Huang et al. as a toolbox for scalable genomic analysis and also included the necessary optimizations for the preprocessing. SparkHit includes a metagenomic mapping utility called "SparkHit-recruiter" that performs much faster than MetaSpark with similar sets of reference genomes. SparkHit performs well with large dataset of reads and small reference genome sets — the authors profiled 2.3 TB of whole genome sequencing reads against only 21 genomes in a little over an hour and a half. The limitation of SparkHit is that it builds its reference index using a k-mer strategy that does not scale to large collections of reference genomes [109], as well as having to rebuild the index for the reference database with each run, as it assumes that the reference genomes being profiled will be changed with each new analysis as is the case with metagenomic sequence assembly experiments [50]. This assumption, and the method of index building, makes SparkHit unsuitable for profiling large metagenomic datasets against large collections of reference genomes.

2.3 Streaming Techniques

In order to process the large quantities of both input metagenomic datasets, and the large collections of reference genomes to profile against, new paradigms are required that take advantage of highly parallelizable cloud infrastructure, as well as real-time data streams for consuming large input datasets that cannot be held in memory.

LiveKraken [142] was developed as a real-time classification tool that improves overall analysis times, and is based on the popular Kraken [167] method for profiling metagenomic samples in Kraken-based workflows. LiveKraken uses the same approach as the HiLive [95] real-time mapper for Illumina reads, but extends it to metagenomic datasets. LiveKraken can ingest reads directly from the sequencing instrument in illumina's binary basecall format (BCL) before the instrument's run finishes, allowing real-time profiling of metagenomic datasets. Reads are consumed as they are produced at the instrument, and the metagenomic profile produced by LiveKraken is continuously updated. LiveKraken points the way to future classification systems that use streams of data as input, but its limitation is that it uses a k-mer based reference index — in its publication, LiveKraken was tested with an archived version of RefSeq (circa 2015) that only contained 2,787 bacterial genomes. Since then, RefSeq has grown to over 50k genomes in the latest release (version 92), and creating a K-mer based index of it would require substantial computational resources.

More recently, a Spark streaming-based aligner has been developed that uses streams of data to map reads single reference genomes. The tool, StreamAligner [126], is implemented with Spark and the Spark-streaming API, and uses novel MapReduce-based techniques to align reads to the reference genome of a single organism. Unlike other methods, it creates its own reference genome index using suffix arrays in a distributed manner that reduces index-build times, and can then be stored in memory during an analysis run. By using the Spark streaming API, StreamAligner can continuously align reads to a single reference genome without the need of storing the input reads in local storage, and although StreamAligner has high performance when using a single genome, there is no evidence if it can scale to metagenomic workflows where tens of thousands of genomes are used, and the footprint of the reference genomes are much larger than could be fit in memory.

2.4 Visualizing Data

Visualizing the results of an analysis is a critical step that can help reveal patterns and structures that are sometimes not evident in the raw numbers alone. Many techniques exist for visualizing tabular data and their corresponding descriptive statistics, and tools such as box plots, gene pies, scatter plots, bar charts, and histograms are all very popular for visualizing biological data [39]. These tools are also very popular in the field of metagenomics because they can be used to aggregate the results of many microbial community profiles, creating large descriptive charts that summarize a lot of information, albeit at higher levels of abstraction [49, 72, 149], see Figure 2.5.

Box plots, histograms, and other tools for generic tabular data can also be used in metagenomics studies to display the results of individual microbiome samples, however, because of the hierarchical nature of microbial taxonomies, they are not suitable for displaying the underlying structure and relationships that are present in the large reference genome collections that are being profiled.

Alternative plotting devices exist that can display relationships among entities, and the Circos tool [82] has been very popular for its ability to create "ribbons"

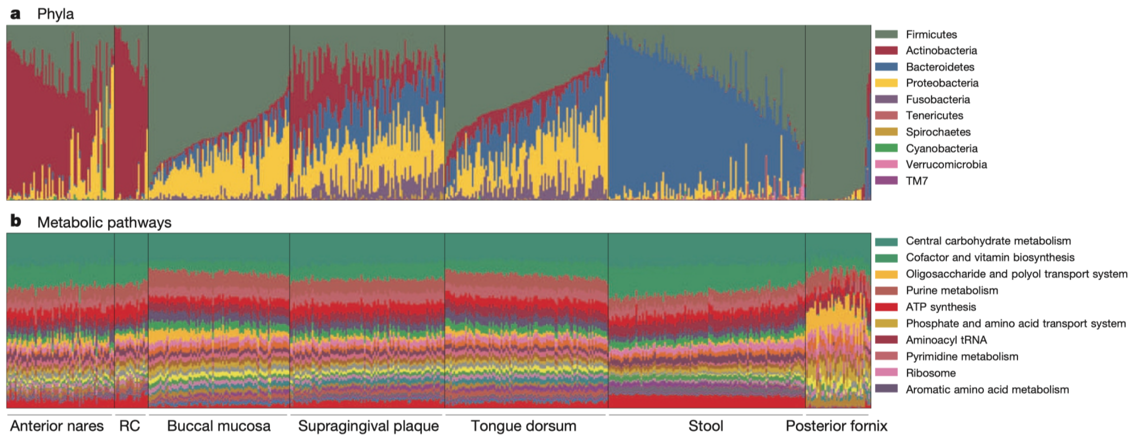


Figure 2.5: Visualizing Microbial Community Profiles with Bar Charts. An analysis of the structure and function of healthy human microbiomes. The bar charts do a great job of summarizing large amounts of data for the conditions in the study, but details are lost for individual samples. Figure 2 from the paper by Huttenhower et al. [72], and reproduced with permission. Copyright 2012, Springer Nature.

among pairs of related objects. In metagenomics, Circos has been used to display the relationship of microbes, and the study by Porcar et al. [121]

Plotting hierarchical data can be achieved using many types of plots such as trees, and a particularly effective one is a Sankey diagram [161], which creates a *flow diagram* (usually from left to right) of bifurcating structures that clearly depict a parent-child relationship — it is similar to a tree drawn sideways. The Plotly library [120] for the R language [123] is a popular library for creating such plots. Figure 2.7 contains such a plot from a paper by Porcar [121] displaying a hierarchy of microbes in a study of taxonomic composition in solar panels from Berkeley, CA, United States.

2.5 Space Filling Curves

Space-filling curves are popular in scientific computing applications for their ability to speed-up computations, optimize complex data structures, and simplify algorithms [18]. Trees are particularly interesting structures that can be optimized with space-filling curves because it is possible to generate sequential orderings of the nodes

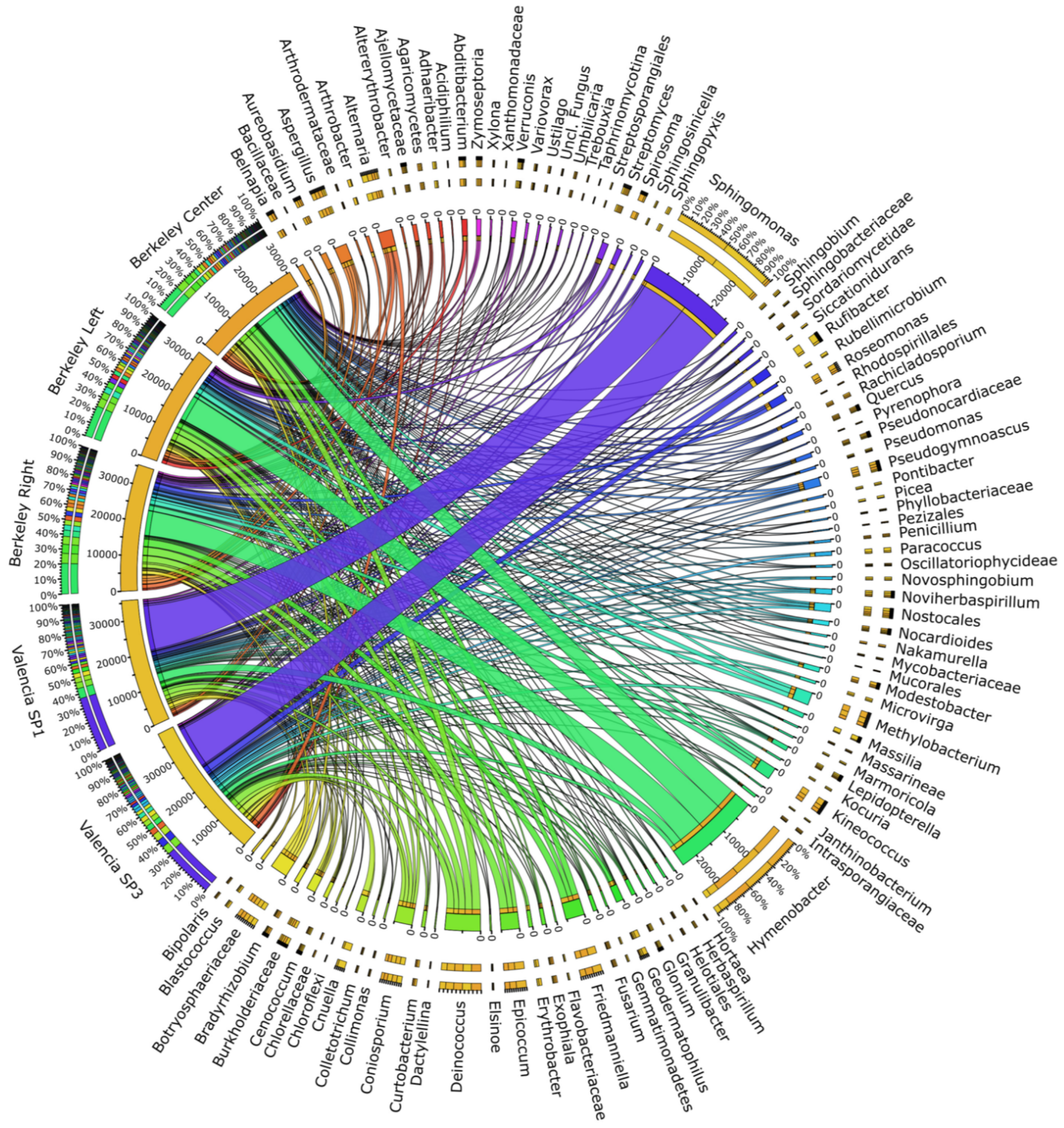


Figure 2.6: Circos Plot for Microbial Ecology. A Circos diagram showing the relationships between genus-level microbes in solar panels. A Circos plot uses a circular backbone structure to display entities, and *ribbons* drawn between pairs of entities to display relationships. Figure 5 from Porcar et al. [121], and reproduced with permission. Copyright 2018, Porcar, Louie, Kosina, Van Goethem, Bowen, Tanner and Northen.

of the tree in which parent and children nodes are neighbors in a two-dimensional plane. The combination of trees and space-filling curves is useful in many fields [16]. In metagenomics we can use them because the taxonomy of bacteria is represented as a classification tree based on the biological characteristics of bacterial taxa, with the leaf nodes representing individual bacterial strains.

2.5.1 Hilbert Curve

The Hilbert Curve [66] is one of the more prominent examples of space-filling curves; and its construction is based on a recursive partitioning of a 2D plane into four sub-squares (Figure 2.8). Many applications exploit the order that space-filling curves impose on data, and a particular application has been the visualization of high-dimensional data. The first use of the Hilbert Curve as a visualization tool was proposed by Keim in 1996 [77] to represent stock market data, and since then, it has been used for visualizing genomic data [35, 12], and also for visualizing DNA alignments of whole bacterial genomes [165].

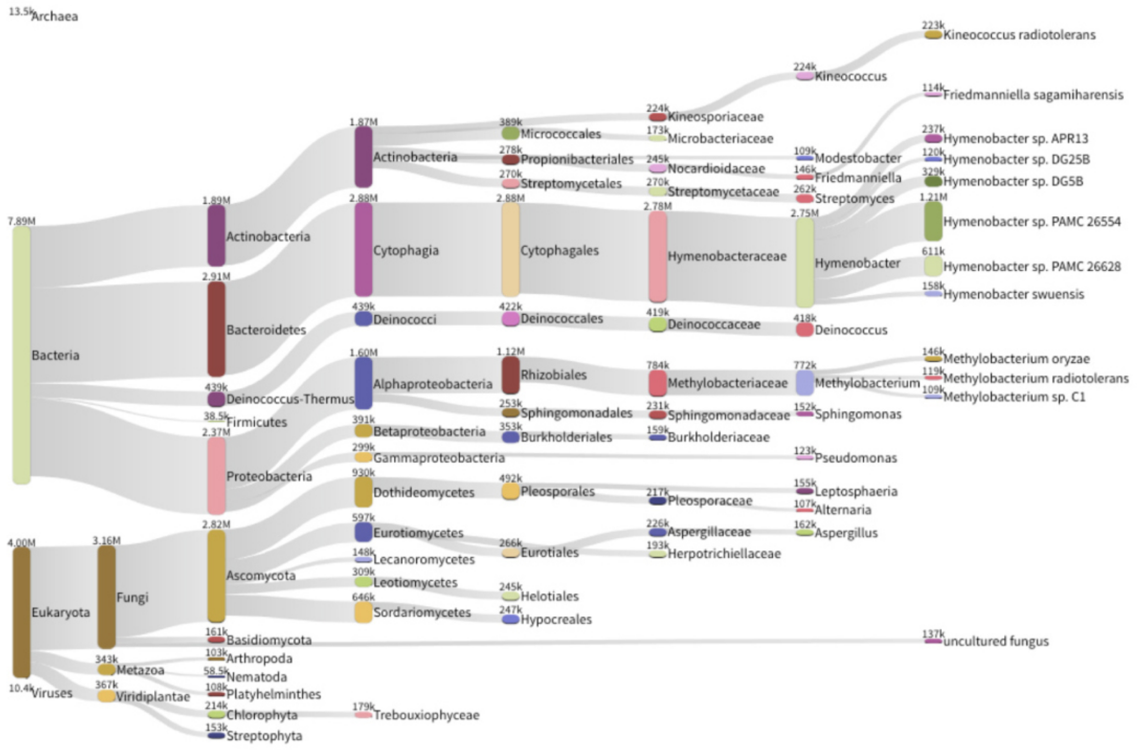


Figure 2.7: Microbial Ecology Sankey Diagram. A taxonomic analysis using a sankey diagram showing a hierarchical structure of microbes. Figure 3 from Porcar et al. [121], and reproduced with permission. Copyright 2018, Porcar, Louie, Kosina, Van Goethem, Bowen, Tanner and Northen.

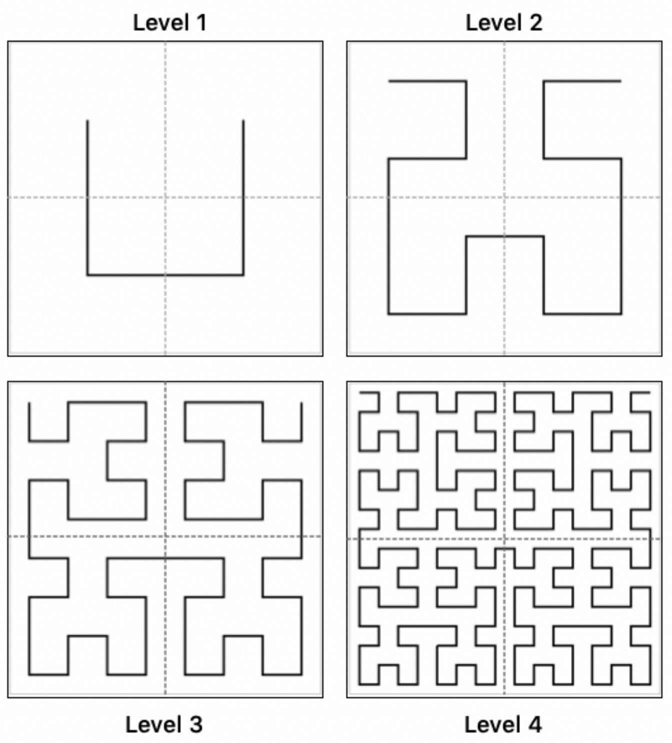


Figure 2.8: Hilbert Curve Levels. First four iterations (levels) of the Hilbert curve show the 2D square recursively partitioned into 4 sections at each iteration (level) of the curve.

In human genomics, using the HCV is a straightforward affair as the natural linear order of genomic positions can be easily visualized by the curve, and there are tools and software libraries for creating Hilbert curve images from human sequencing experiments such as HilbertVis [12] and HilbertCurve [59]. Both HilbertVis and HilbertCurve apply HCV to genomics in the context of a single human genome: a single genome scaffold is modeled as a one-dimensional (1D) line in which each interval is taken to be a single genomic position, but the HilbertCurve library can also stitch together multiple chromosomes so that they can all be displayed on a single image, but the display is nevertheless that of a single genome. To date, the HCV technique has not been applied to metagenomics datasets.

2.6 Artificial Intelligence and Machine Learning

The field of artificial intelligence (AI) is a field in computer science that studies so-called *thinking* by machines, and can be defined as "the effort to automate intellectual tasks normally performed by humans" [31]. AI has practical applications in the fields of computer vision [40], self-driving cars [29, 45, 130], and medical imaging [61, 64] to name a few. *Machine Learning* is a sub-field of artificial intelligence (see Figure 2.9) and is a set of technologies that can learn to recognize patterns in a dataset by being exposed to those patterns in a different training dataset [31, 54]. Machine learning techniques differentiate themselves from other techniques in AI in that machine learning systems do not have a predefined set of rules that govern the behavior of the system — the system automatically *learns the rules.* The *Deep Learning* technique itself is a sub-field of machine learning (Figure 2.9), and differentiates itself from other *shallow learning* techniques by creating successive layers of representations [31, 54]. Consider the case of trying to classify pictures of a dog using a *Deep Learning* system: different layers would learn different characteristics and traits, so say the first layer would learn to identify lines, another layer would learn to identify shapes such as eyes, noses, and ears; and another layer would learn to identify faces, and so on until the final layer would decide if the picture we are looking at contains a dog or not. The following section, section 2.6.1, contains a detailed explanation of the technique.

2.6.1 Deep Learning

Deep learning is a Machine Learning technique that allows for the development of systems that accumulate knowledge from experience by modeling problems as a hierarchy of concepts, with higher level concepts defined by connected simpler concepts [54]. The method is called *Deep Learning* because if we were to visualize the procedure, then we end up with concepts layered one on top of each other, spread

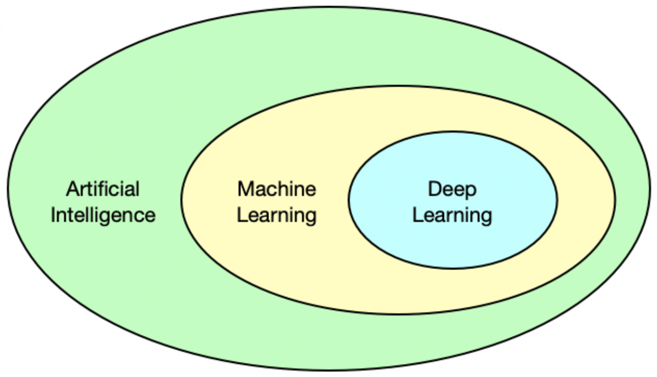


Figure 2.9: Artificial Intelligence, Machine Learning, and Deep Learning. The relationship between the fields of Artificial Intelligence, Machine Learning, and Deep Learning reproduced and based on the book by François Chollet, *Deep Learning with Python* [31].

out in many layers (Figure 2.10). Artificial Neural Networks (ANNs) are prototypical examples of deep learning networks. The recent advent of accessible low-cost computing, especially with inexpensive graphical processing units (GPU), has allowed the development of large deep learning systems that can extract patterns from raw data; researchers have successfully used this capacity in fields as wide ranging as law enforcement [136], biomedical image segmentation [62], and the analysis of non-coding RNA sequences [14].

A type of specialized neural network known as a Convolutional Neural Network (CNN) [89] has been found to be particularly effective for processing data that have grid-like topologies. Unlike traditional neural networks which use matrix multiplication to model the interactions between input and output neurons in each layer, CNNs use convolution operations in at least one layer, which has the effect of creating a sparse connectivity graph between the input and output units. The benefit of using convolutions over matrix multiplications is that it allows CNNs to describe complex interactions between many variables only using simple sparse connections between the units in adjacent layers. CNNs also have the advantage of producing high-level ab-

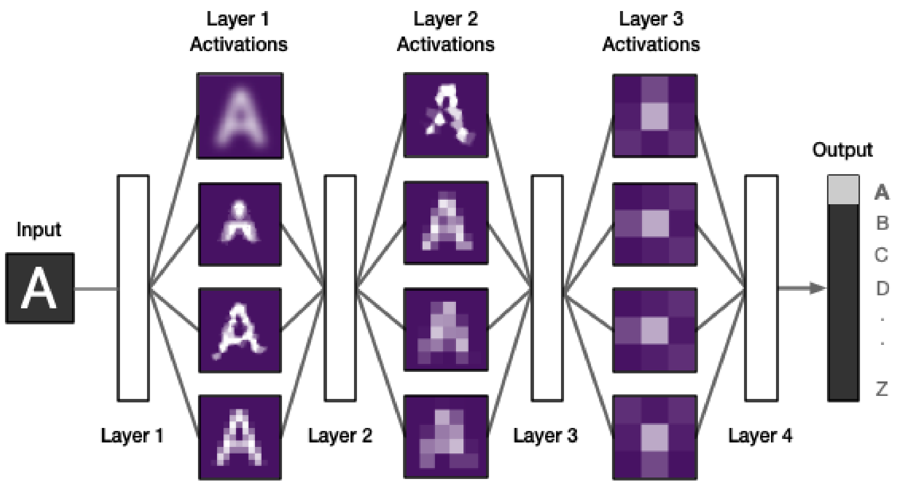


Figure 2.10: Character Classification. A stylized depiction of a *Deep Learning* neural network for classifying character letters (“A”, “B”, “C”, etc.) Each layer in the model can learn to represent the network’s input into increasing levels of relevant and abstract representations. Inspired by Figure 1.6 from the book by François Chollet, *Deep Learning with Python* [31].

stract objects rather than just a class for a classification job, or continuous real-values for regression problems.

Modern deep learning frameworks [2] are able to detect, classify, and diagnose diseases such as skin cancer [41] (using 2D images), segment brain tumors [64] (using 2D image slices taken from CT or MRI scans), detect anomalies in electrocardiograms [36, 150] (using numerical signal data) and more recently, they have also been used in metagenomics to classify multiple diseases [87, 112] (using OTUs) and to classify disease stages [46] (using numerical OTU abundance and phylogenetic distance matrices). These deep learning frameworks are remarkably robust, and CNNs in particular are particularly powerful when working with image datasets as can be seen in many applications such as hand-written digit recognition [89], the modeling of digital pathology images [98], and the segmentation of 2D and 3D microscopy biomedical imaging datasets [62].

Generative adversarial networks (GAN) [55] are a compelling combination of two CNN models working against each other (hence the word “adversarial”) to create new content. A GAN is comprised of two models, a *generator* which learns to create feasible data, and a *discriminator* which learns to identify the generator’s data from real data. As the GAN is trained with real data, the generator model becomes progressively better at generating data that looks “real”, while the discriminator models becomes progressively worse at distinguishing real data from generated data. GANs are useful for simulating real-like images, and they have applications in driving simulators [130], medical image analysis [170], and reducing overfitting via data augmentation [22].

CHAPTER 3

MICROBIAL REFERENCE GENOMES AND CLOUD INFRASTRUCTURE

This chapter describes the primary genomic reference data that we used to build the software systems described in this dissertation, i.e., the collection of microbial reference genomes. It also provides details on the preprocessing applied to these datasets. Finally, this chapter also describes the cluster infrastructure used for carrying out our experiments. When appropriate, the chapter provides details on the commands, parameters, and environments used in the steps.

3.1 Reference Genomes

Bacterial Genomes for our reference database were downloaded from the Ensembl Bacteria [42] repository. Multiple versions of the Ensembl Bacteria database were processed and analyzed, from Ensembl Bacteria versions 38 through 45. The version we use, 45, contains 44,048 bacterial genomes (strain level), and these were downloaded in FASTA format, accounting for over 4.7 million individual FASTA reference files. The collection included reference sequences for fully assembled chromosomes and plasmids, as well as sequences for draft-quality supercontigs, the latter accounting for most of the reference files in our database.

In total, the 44K bacterial genomes had a data footprint of 170 GB. It took four days to download the data from the Ensembl repository with a custom user script that employed the *wget* command.

In addition, genome annotations were also downloaded for each of the 44K genomes; the annotations were downloaded from the Entrez database [131] at the National Center for Biotechnology Information (NCBI) using the *Bio* package in a Python script.

3.2 Reference Genome Preprocessing

Before the microbial genomes could be used with our analysis pipelines, and before they were deployed on our cluster, they were prepared appropriately. The preprocessing of the genomes involved expanding/extracting the downloaded reference sequence files, reformatting FASTA headers so that the reference files contained specific taxonomic lineage information, partitioning the big FASTA file into a specific number of partitions (denoted by the parameter N, and finally indexing each of the resulting partitions. In summary, our preprocessing pipeline consisted of the following steps:

1. Genome Download
2. FASTA Extraction
3. Annotation Preparation
4. Reference Database Partitioning
5. Reference Database Indexing

The following subsections describe the particulars of each of the aforementioned steps. The machine named "`castalia`" at the School of Computing and Information Sciences at Florida International University [133] was used in all the preprocessing steps described below, as well as to obtain a benchmark performance with a single machine. The machine runs the Linux operating system, has over 10 TB of disk storage, 768 GB of RAM, and 48 Intel Xeon CPUs (E5-2650 v4, 2.20GHz).

Downloading the Genomes

We downloaded the Ensembl database (v.45) from the public FTP site located at `ftp.ensemblgenomes.org`. Ensembl stores the genome FASTA files in "collection" directories, and we recursively downloaded the "dna" directory in each of the bacterial sub-folders. In total, 4,672,683 FASTA files were downloaded, with a data footprint on disk of just over 170 GB — these accounted for 44,048 bacterial strains.

FASTA Extraction

Ensembl distributes the FASTA files in the "gzip" [47] format, and in order for us to be able to further enhance them for use in our cluster, we needed to extract them. Extracting all the 4.6 million FASTA files took over one day on the "castalia" machine.

Annotation Preparation

Each FASTA record from the Ensembl database contains minimal information about the lineage of each bacteria. In order to create meaningful reports, as well as to compare the advantages and disadvantages of using a large bacterial reference database, we added additional annotations to each of the header strings of the FASTA records in all of the sequences. The additional information that we added included information such as the "Taxonomic ID", "start coordinate", "end coordinate", and a optimized "Sequence ID" string that we could efficiently parse in the *MapReduce* operations [34].

3.2.1 Partitioning and Indexing the Reference Genomes

Reference Database Partitioning

Note that if all the sequences were to be concatenated into a single file, it would be of size 170 GB, then the indexers such as Bowtie2 [84] and Kallisto [24] would

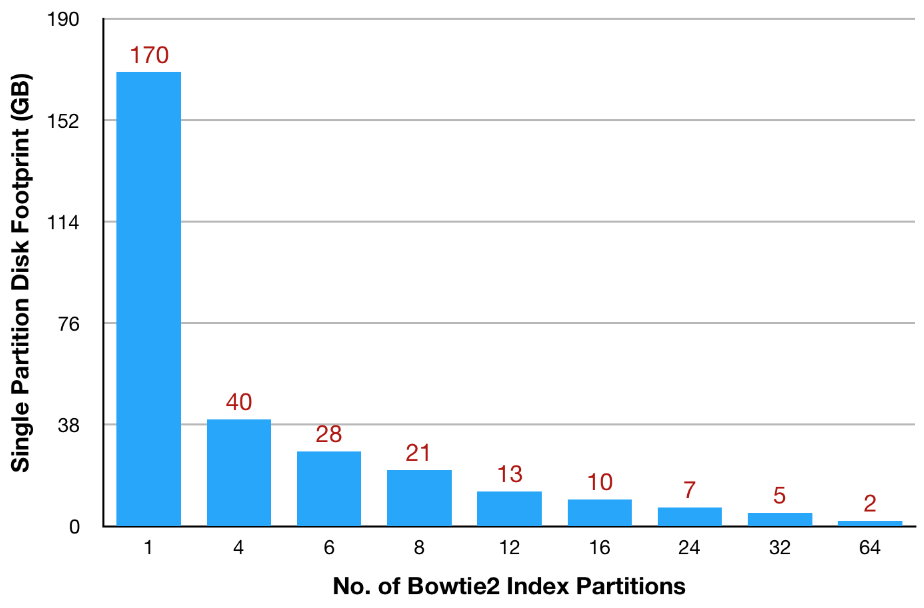


Figure 3.1: Bacterial genome reference database partitions. Red labels indicates the average disk size in gigabytes (GB, y-axis) of a single partition when partitioning the full reference database into N-number of partitions (x-axis).

either be too slow, run out of memory, or would crash since they were designed for human genomes, which are approximately of size 3 GB.

Partitioning is a powerful technique that has been shown to be useful for detecting bacterial genomes [152] and mapping long-reads from nanopore sequencing [48] in resource-constrained platforms.

The idea of partitioning is to split the reference database such that each partition when indexed is just small enough to fit in the memory space of a small to medium-sized Amazon Elastic Compute Cloud (EC2) machine, since otherwise the individual machines would end up slowing down due to thrashing. The average partition sizes as a function of the number of partitions are shown in Figure 3.1). Once partitioned, the index of each partition would be placed on a Spark worker nodes [173]. Note that the partitions are not all exactly the same size because each partition has to contain only complete genomes, not partial ones as in [48].

Indexing the Reference Database

Part of the process of preparing a reference database of genomes for DNA mapping is to create a reference *Index* [147] — an optimized data structure that the DNA mapper uses to map the input sequencing DNA reads to the putative reference sequences of origin. The process is resource intensive and often took multiple days to complete. The resulting reference database shards were each indexed independently and sequentially on the "`castalia`" machine, a process that took roughly two and a half days using the "`bowtie2-build`" utility. Since the partitions were not all of the same exact size, the resulting indexes showed some variations in size, as shown in Table 4.1).

3.3 Cloud Infrastructure

Once all the shards of the reference genome database have been indexed, we upload all of them into a bucket in Amazon's Simple Storage Service (S3) [11]. Uploading the partitioned reference genome through the AWS website is not a feasible option as the web user-interface is designed for relatively small files, so instead, we used the AWS command line utility (CLI) to perform the upload to the S3 bucket. Uploading the reference genome's shards took 50 minutes with 57 seconds using the command in Listing 3.1.

```
aws s3 cp /ensembl/partitions s3://references/ensembl/partitions
    --recursive
```

Listing 3.1: The command used to upload partitions to the S3 bucket.

3.3.1 AWS Cloud Cluster

The cluster consists of multiple "commodity" worker machines (a computational "worker" *node*), each with 15 GB of RAM, 8 vCPUs (each being a hyperthread of a single Intel Xeon core), and 100 GB of disk storage. A Spark [173] cluster was created using the AWS Console with the following software configuration: EMR-5.7.0, Hadoop 2.8.4, Ganglia 3.7.2, Hive 2.3.3, Hue 4.2.0, Spark 2.3.1, and Pig 0.17.0 in the US East (N. Virginia) region.

The cluster is composed of homogeneous machines for both the driver node and worker nodes, and each machine is an Amazon machine instance of type c4.2xlarge. These instances contain 8 vCPUs, 15 GB of RAM, 100 GB of EBS storage, and each cost on average $0.123 USD to run per hour on the "us-east" availability zone on the Spot [10] market as of this writing in January 2019. Newer instances (c5.2xlarge) are also available for use, but their availability is infrequent in large numbers, in addition to having a higher cost per hour to run.

Cost Analysis

All experiments were conducted using Amazon's Elastic MapReduce service (EMR) [6] and used the `c4.2xlarge` machine instance type. These machines contain 8 vCPUs, 15 GB of RAM, and 100 GB of EBS storage, and provide a good balance between cost and performance. At the time of this writing, each machine instance type costs around $0.123 USD in the Amazon's Spot market [10]. The Spot market offers machines at a reduced cost as they are machines that have been reserved by other customers but are currently not being used, and rather than have the machines be idle, the purchasing customer offers them at a reduced cost.

Read Data Streaming

Resilient Distributed Datasets (RDD) [174] are robust programming abstractions that can be used to persist data across a cluster of machines. The primary source of input to the cluster are sequencing reads taken from a datastream in batches that are processed by the MapReduce pipeline. Reads are consumed either directly from their location in an Amazon S3 bucket, the local file system of the master node, or from a datastream source. The stream of reads in transformed into an RDD that is consumed by the master node, which then broadcasts it out into all the worker nodes in the cluster. Note that the input RDD of reads is partitioned into sets of reads that are each independently aligned to a reference genome partition in each of the worker nodes of the cluster.

CHAPTER 4

LARGE-SCALE MICROBIAL COMMUNITY PROFILING IN THE CLOUD

Bacterial metagenomics profiling for metagenomic whole sequencing (mWGS) usually starts by aligning sequencing reads to a collection of reference genomes. Current profiling tools are designed to work against a small representative collection of genomes, and do not scale well to larger reference genome collections. However, large reference genome collections are capable of providing a more complete and accurate profile of the bacterial population in a metagenomics dataset. In this paper, we discuss a scalable, efficient, and affordable approach to this problem, bringing big data, high-performance computing solutions within the reach of laboratories and researchers with modest resources.

We developed Flint, a metagenomics profiling pipeline that is built on top of the Apache Spark framework [173], and is designed for fast real-time profiling of metagenomic samples against a large collection of reference genomes. Flint takes advantage of Spark's built-in parallelism and streaming engine architecture to quickly map reads against a large (170 GB) reference collection of 44,048 bacterial genomes from Ensembl. Flint runs on Amazon's Elastic MapReduce service, and is able to profile 1 million Illumina paired-end reads against over 40K genomes on 64 machines in 67 seconds – an order of magnitude faster than the state of the art, while using a reference genome collection that is an order of magnitude larger than that used by the competing tools. Streaming the sequencing reads allows this approach to sustain mapping rates of 55 million reads per hour, at an hourly cluster cost of $8.00 USD, while avoiding the necessity of storing large quantities of intermediate alignments. Such prices make microbiome profiling using high-performance computing affordable for all researchers.

4.1 Microbial Community Profiling

Microbes are ubiquitous and a microbiome is a community of microbes that inhabit a particular environmental niche such as the human body, earth soil, and the water in oceans and lakes. Metagenomics is the study of the combined genetic material found in microbiome samples, and it serves as an instrument for studying microbial biodiversities and their relationships to humans. Profiling a microbiome is a critical task that tells us what microorganisms are present, and in what proportions; this is particularly important as many human diseases are linked to changes in human microbiome composition [63, 80, 168, 175], and large research projects have started to investigate the relationships between the two [33].

A powerful tool for profiling microbiomes is high-throughput DNA sequencing [105], and whole metagenome sequencing experiments generate data that give us a lens through which we can study and profile microbiomes at a higher resolution than 16S amplicon-based sequencing analyses [124].

Advances in sequencing technologies have steadily reduced the cost of sequencing and have led to an ever increasing number of extremely large and complex metagenomic data sets [13, 26]. The resulting computational challenges for analysis tools include the storage management of even larger intermediate results and larger indexes of the reference genome collections [156], and the resulting increase in processing time that make it impossible to process the data sets on commodity workstations or laptops. Powerful multi-user servers and clusters are an option, but the cost of systems with higher processor speeds, greater storage volumes, and huge memory sizes are out of reach for small laboratories.

To deal with the deluge of sequencing data, distributed cloud computing platforms and frameworks such as Amazon Web Services [11], Apache Hadoop [143], and Apache Spark [173] have been used by researchers to take advantage of parallel computation and economies of scale. Thus, large sequencing workloads are distributed on a cloud cluster that is comprised of many inexpensive, off-the-shelf compute nodes. These cloud-based solutions have been successfully used for human genomics [85], transcriptomics [127], and more recently for metagenomics applications [178, 70] (see Section 4.2).

Standard genomics and transcriptomics analyses for sequencing datasets usually begin by aligning sequencing reads to a reference genome [147, 158], and producing abundance counts [148]; but in metagenomic analyses, the alignment step is performed against a collection of reference genomes that can be extremely large, slowing down the entire operation. The MapReduce model [34] along with the Spark framework have been popular in speeding up these crucial steps when dealing with sequence datasets from a single organism. This was achieved by framing the read alignment and quantification tasks in terms of `map` and `reduce` operations [86, 127].

4.2 Approach

4.2.1 Spark and MapReduce

The MapReduce model was originally developed by researchers from Google [34], and most notably popularized by the Apache Hadoop [143] open-source project from the Apache Foundation [144]. The Apache Spark [173] project further expanded the Hadoop project and introduced new optimizations for enhanced calculation speeds and improved programming paradigms [174]. The MapReduce model abstracts away much of the boiler-plate programming details of developing distributed applications, freeing developers to focus more on algorithm-specific issues. The framework requires the user to flesh out two distinct steps: the `map()` step, and the `reduce()` step. These functions are executed on two clusters of processors, each `map()` working on a different fragment of the dataset. The `map()` function produces as output a set of tuples, each consisting of a `(key, value)` pair; while the `reduce()` function merges the output of the `map()` function by coalescing or aggregating tuples with the same `key`.

The MapReduce model and the Spark framework have been employed in a number of sequence data analysis workflows [27, 60]. The Crossbow project from 2009 used Spark's MapReduce implementation to identify Single Nucleotide Polymorphisms (SNPs) in human samples [85]; eXpress-D used Spark to implement the expectation maximization (EM) algorithm for ambiguous DNA-fragment assignment [127]. Spark has also been used in metagenomic analyses [60] for mapping sequencing reads against small reference databases and for clustering metagenomes [125].

A natural approach to use the Spark framework for the analysis of mWGS datasets is to partition the input of reads into smaller subsets of reads to be processed by worker nodes in a Spark cluster. This strategy works well when the dataset of reads is large. The limitation of this strategy is that it does not scale to large collections of reference genomes because a data structure (index) of the reference collection of genomes must either be duplicated in each of the worker nodes, or multiple passes of the input will need to be used. Indexes built from large reference collections using a k-mer based strategy are often too large to be accommodated on a single commodity machine on the cloud [109]. Fast k-mer based strategies have been used for profiling mWGS datasets [132, 167]. But they trade-off speed for the size of the indexes. More recently, alternative index-building strategies were developed [177]. However, these were achieved only at species-level resolutions and were not designed for use with a cloud-based infrastructure.

Zhou et al. developed MetaSpark [178] to align metagenomic reads to reference genomes. The tool employs Spark's Resilient Distributed Dataset (RDD) [174] – a critical programming abstraction for working with large datasets. RDDs were used to cache information on reference genomes and reads across worker nodes in the cluster. By using Spark's RDD, MetaSpark is able to align more reads than previous tools.

MetaSpark was developed with two reference datasets of bacterial genomes: a small reference collection of size 0.6 GB, and a larger one from RefSeq's bacterial repository of size 1.3 GB. These reference sets are small compared to the 170 GB reference set of Ensembl. Since MetaSpark uses an RDD to hold the index of the reference genome collection, it is unlikely that MetaSpark can scale to use them because the contents of an RDD are limited by the available memory, which are not standard in commodity machines on the cloud. More specifically, MetaSpark took 201 minutes (3.35 hours) to align 1 million reads to the small 0.6 GB reference using 10 nodes [178].

SparkHit [70] was developed by Huang et al. as a toolbox for scalable genomic analysis and also included the necessary optimizations for the preprocessing. SparkHit includes a metagenomic mapping utility called "SparkHit-recruiter" that performs much faster than MetaSpark with similar sets of reference genomes. SparkHit performs well with large dataset of reads and small reference genome sets — the authors profiled 2.3 TB of whole genome sequencing reads against only 21 genomes in a little over an hour and a half. The limitation of SparkHit is that it builds its reference index using a *k*-mer strategy that does not scale to large collections of reference genomes [109], especially if the reference database changes with each study that is analyzed. Thus, SparkHit does not scale well and is unsuitable for profiling large metagenomic datasets against large collections of reference genomes.

4.2.2 Streaming Techniques

In order to process the large quantities of both input metagenomic datasets, and the large collections of reference genomes to profile against, new analysis paradigms are required that take advantage of highly parallelizable cloud infrastructure, as well as real-time data streams for consuming large input datasets.

LiveKraken [142] was developed as a real-time classification tool that improves overall analysis times, and is based on the popular Kraken [167] method for profiling metagenomic samples in Kraken-based workflows. LiveKraken uses the same approach as the HiLive [95], a real-time mapper for Illumina reads, but extends it to metagenomic datasets. LiveKraken can ingest reads directly from the sequencing instrument in Illumina's binary basecall format (BCL) before the instrument's run finishes, allowing real-time profiling of metagenomic datasets. Reads are consumed as they are produced at the instrument, and the metagenomic profile produced by LiveKraken is continuously updated. LiveKraken points the way to future classification systems that use streams of data as input, but its limitation is that it uses a k-mer based reference index. In fact, as reported in their publication, LiveKraken was tested with an archived version of RefSeq (circa 2015) that only contained 2,787 bacterial genomes. Since then, RefSeq has grown to over 50K genomes in the latest release (version 92), and creating a k-mer based index of it would require substantial computational resources, not available on commodity machines.

More recently, a Spark streaming-based aligner was developed that uses streams of data to map reads single reference genomes. The tool, StreamAligner [126], was implemented with the Spark-streaming API and used novel MapReduce-based techniques to align reads to the reference genome of a single organism. Unlike other methods, it creates its own reference genome index using suffix arrays in a distributed manner that reduces index-build times, and can then be stored in memory during an analysis run. With the Spark streaming API, StreamAligner can continuously align reads to a single reference genome without the need to store the input reads in local storage. Although StreamAligner has high performance when using a single genome, there is no evidence that it can be scaled up to metagenomic workflows where tens of thousands of genomes are involved, and the footprint of the index of the reference genomes are too large to fit in memory.

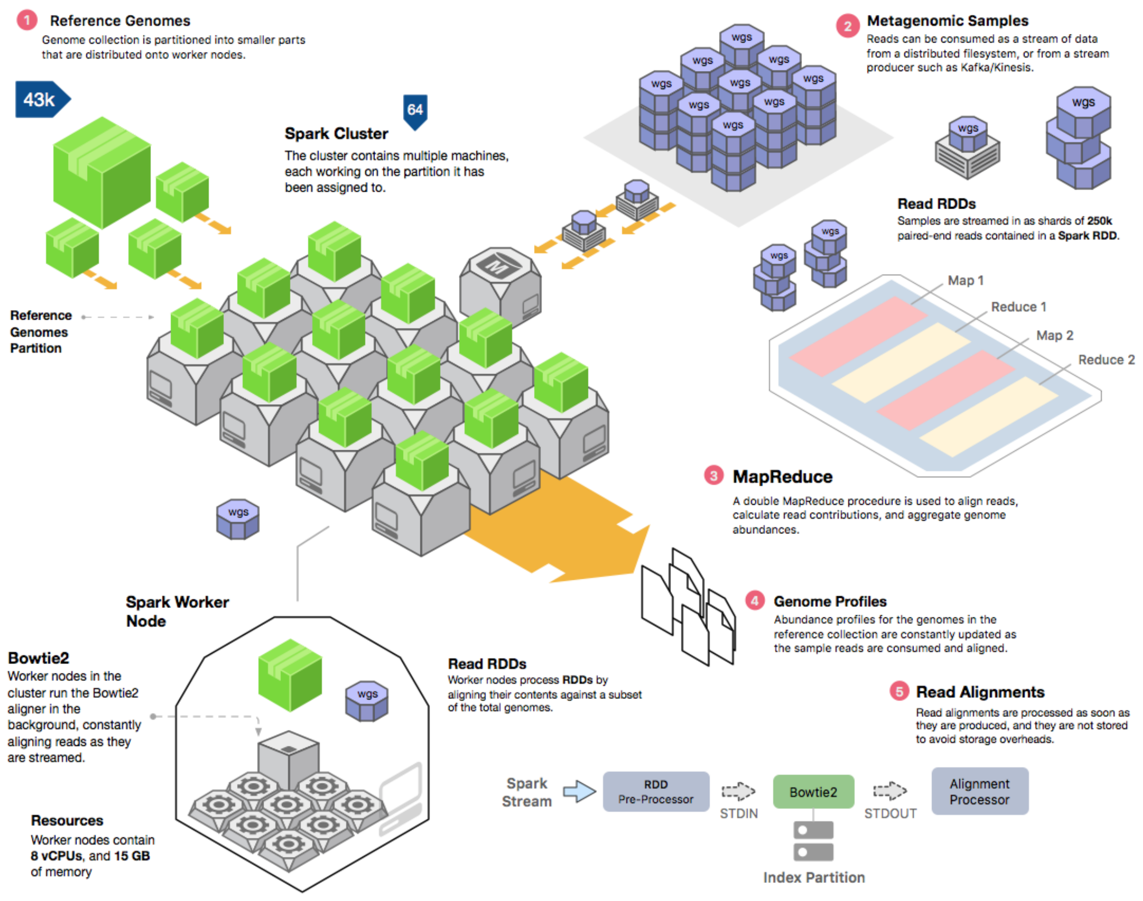

1 Reference Genomes
Genome collection is partitioned into smaller parts that are distributed onto worker nodes.
43k
64
Spark Cluster
The cluster contains multiple machines, each working on the partition it has been assigned to.
2 Metagenomic Samples
Reads can be consumed as a stream of data from a distributed filesystem, or from a stream producer such as Kafka/Kinesis.
wgs
Read RDDs
Samples are streamed in as shards of 250k paired-end reads contained in a Spark RDD.
Map 1
Reduce 1
Map 2
Reduce 2
Reference Genomes Partition
3 MapReduce
A double MapReduce procedure is used to align reads, calculate read contributions, and aggregate genome abundances.
4 Genome Profiles
Abundance profiles for the genomes in the reference collection are constantly updated as the sample reads are consumed and aligned.
Spark Worker Node
Bowtie2
Worker nodes in the cluster run the Bowtie2 aligner in the background, constantly aligning reads as they are streamed.
Read RDDs
Worker nodes process RDDs by aligning their contents against a subset of the total genomes.
5 Read Alignments
Read alignments are processed as soon as they are produced, and they are not stored to avoid storage overheads.
Resources
Worker nodes contain 8 vCPUs, and 15 GB of memory
Spark Stream
RDD Pre-Processor
STDIN
Bowtie2
STDOUT
Alignment Processor
Index Partition

Figure 4.1: Overview of the F<span style="font-variant:small-caps">lint</span> **System.** Reference genomes are partitioned so that a large reference set is be distributed across a Spark cluster, and the number of partitions matches the number of worker nodes. Samples are streamed into the cluster to avoid storage overheads as shards of 250K reads. Reads are aligned to the distributed reference genomes using a double MapReduce pipeline that continually updates metagenomic profiles as samples streamed into the cluster. Read alignments are never stored, and processed by each worker node as soon as they are produced.

4.3 Methods

4.3.1 A "Double" MapReduce

A natural approach to using MapReduce for large metagenomic analyses tasks is as follows. The `map` step divides the task of mapping the reads against a genomic index by having each map processor work on a part of the index, and the `reduce` step aggregates all the hits to each genome and constructs the microbial profile of the metagenomic sample. Since the full genomic index is too large to be farmed out to each map node in the cluster, we adopt the approach of building indexes from each "shard" of the reference genome database and providing each cluster node with a smaller index. Such an approach allows for much larger reference databases to be processed for mapping the reads.

Following up on the aforementioned idea, one way to perform the MapReduce is to have the `map` function produce tuples of the form $\langle g, 1\rangle$, for every read r that is aligned to genome g, while the `reduce` function aggregates all tuples of the form $\langle g, 1\rangle$ to obtain the abundance of genome g in the sample being analyzed, effectively generating output tuples of the form $\langle g, \mathcal{A}(g)\rangle$, where $\mathcal{A}(g)$ is the reported abundance of genome g in the sample being analyzed.

Unfortunately, a read may align to multiple genomes. Instead of counting a hit for every genome that the read aligns to, or counting it for only one of the genomes that the read aligns to, we follow the algorithm of Valdes et al. [152], which assigns fractional counts for the genomes that a read aligns to. In order to implement fractional counts, we employ a novel double MapReduce algorithm. In the modified MapReduce, the `map` function generates alignments in SAM format [93] by dispatching a subprocess of the Bowtie2 aligner and produces tuples of the form $\langle r, (g, 1)\rangle$, for every read r that is aligned to genomes g. All tuples for the same read are aggregated by the first `reduce` step to generate tuples of the form $\langle r, (g, 1/\mathcal{C}(r))\rangle$. The second map step generates contributions of reads for a given genome, and the second reduce step aggregates all tuples of the form $\langle g, c\rangle$ to obtain the abundance of genome g in

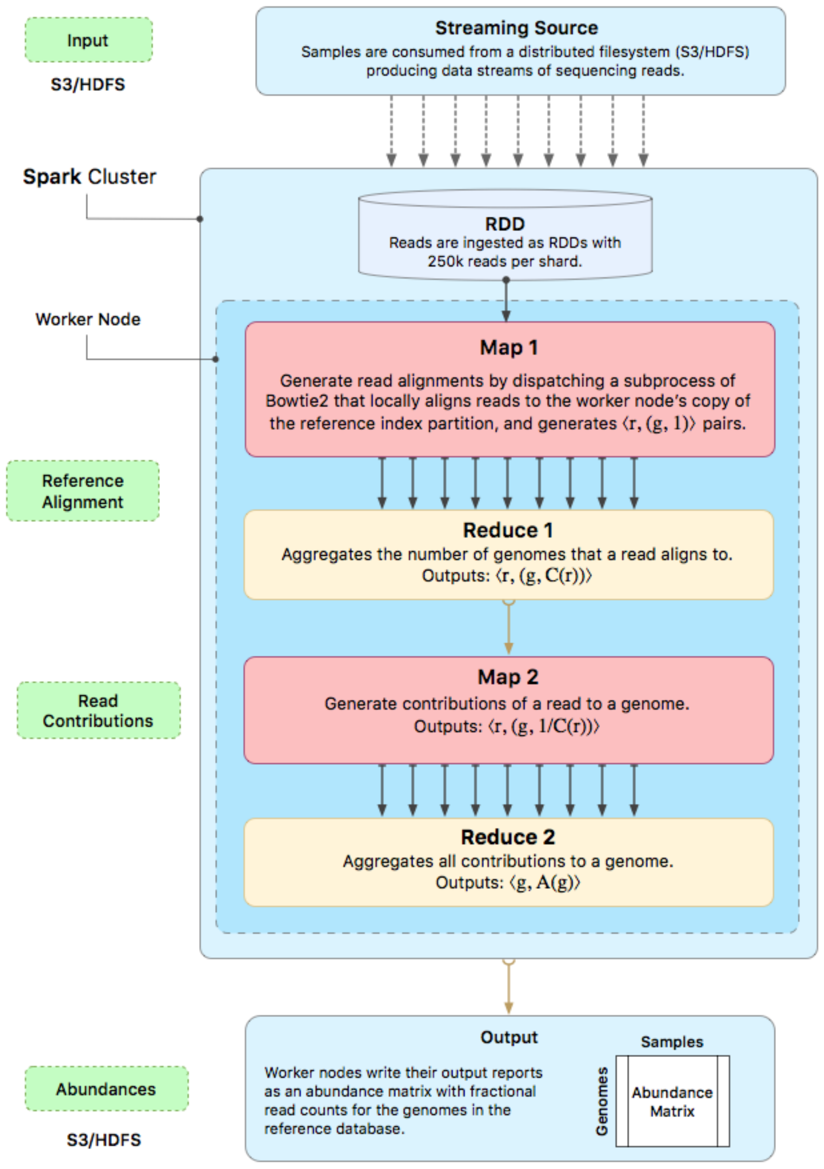

Input
S3/HDFS
Streaming Source
Samples are consumed from a distributed filesystem (S3/HDFS) producing data streams of sequencing reads.
Spark Cluster
RDD
Reads are ingested as RDDs with 250k reads per shard.
Worker Node
Map 1
Generate read alignments by dispatching a subprocess of Bowtie2 that locally aligns reads to the worker node's copy of the reference index partition, and generates ⟨r, (g, 1)⟩ pairs.
Reference Alignment
Reduce 1
Aggregates the number of genomes that a read aligns to. Outputs: ⟨r, (g, C(r))⟩
Read Contributions
Map 2
Generate contributions of a read to a genome. Outputs: ⟨r, (g, 1/C(r))⟩
Reduce 2
Aggregates all contributions to a genome. Outputs: ⟨g, A(g)⟩
Output
Worker nodes write their output reports as an abundance matrix with fractional read counts for the genomes in the reference database.
Samples
Genomes
Abundance Matrix
Abundances
S3/HDFS

Figure 4.2: MapReduce Workflow. Metagenomic samples can be streamed in from a distributed filesystem into the cluster were they are stored in an RDD. The first map step generates alignments through Bowtie2 and feeds its resulting pairs to the first reduce step, which aggregates the genomes that a single reads aligns to. The second map step generates read contributions that are used in the second reduce step to aggregate all the read contributions for a single genome. This results in an abundance matrix containing the abundances of each genome in the sample being analyzed.

the sample being analyzed, effectively generating output tuples of the form $\langle g, \mathcal{A}(g)\rangle$, where $\mathcal{A}(g)$ is the reported abundance of genome g in the sample being analyzed obtained by aggregating all the fractional contributions of reads that map to that genome. Note that all intermediate tuples are stored in RDDs, one for each step.

4.3.2 Reference Genome Preparation

Before we can use the bacterial genomes in the cluster, they need to be prepared. The process entails creating a Bowtie2 index for each shard of the reference database, and specific details on this procedure can be found in Section 3.2. Briefly, the reference genomes are divided into smaller partitions that are each independently indexed by Bowtie2. The index preparation step can take considerable computational resources and time with a single machine. A parallel version of the indexing system can greatly improve performance and will be automated in the next release of Flint. As described in Section 3.2.1, once the partitions have been indexed they are then copied to an Amazon S3 [8] bucket that serves as a staging location for the reference shards. The staging S3 bucket holds the index so that worker nodes can copy it during their provisioning step and the analysis can start; the S3 bucket is also public, and researchers can download copies of the prepared indices for their use.

It should be noted that Ensembl's bacterial genome collections have grown only modestly in the last couple of releases to minimize redundancy, and reference indices for new Ensembl releases can be built relatively quickly with utility scripts provided by Flint. The cost of building a partitioned reference index is only accrued the first time it is built for a cluster of a particular size, and as part of the release of the Flint project, we are making available partitioned indices of Ensembl (v.45) with 48, 64, 128, 256, and 512 fragments. We anticipate that such precomputed indexes would be useful for researchers employing clusters of those sizes. These indices, along with the scripts necessary to build future versions, can be found in the GitHub repository through the project website.

We currently use minimal annotations that keep track of basic attributes for each bacterial strain; these include taxonomic identifiers, assembly lengths, etc. Future releases of the software will include a more robust annotations package that will contain data on gram staining, pathogenicity, and other properties.

FLINT uses a streaming model to quickly map a large number of reads to a large collection of reference bacterial genomes by using a distributed index. Resilient Distributed Datasets (RDD) [174] are robust programming abstractions that can be used to persist data across a cluster of machines. We ingest reads from datastreams in batches of 500,000 reads that are processed by our mapreduce pipeline. Reads are consumed either directly from their location in an Amazon S3 bucket, or from a datastream source. An RDD of the input read stream is created in the master node that is then broadcasted out into all the worker nodes in the cluster. The input RDD of reads is partitioned into sets of reads that are each independently aligned to a reference genome partition in each of the worker nodes. The Bowtie2 DNA aligner is used internally in Spark worker nodes to align reads to the local partition of the reference index, by using a MapReduce that continuously streams reads into worker nodes. Output alignments are parsed and tabulated by worker nodes, and then sent back to master node as alignment tasks finish. FLINT can be deployed on any Spark cluster, as long as the necessary software dependencies are in place; the partitioned reference index for Ensembl's 44K genomes is made available at the FLINT website, and scripts are provided as part of the provisioning step that copy the partitions into worker nodes.

4.3.3 Cluster Provisioning

Our computational framework is primarily implemented using the *MapReduce* model [34], and deployed in a cluster using the *Elastic Map Reduce* (EMR) service offered by AWS (Amazon Web Services) [11]. Each of the worker computational nodes will execute in parallel to align the input sequencing DNA reads to a "shard" of the reference database (Figure 4.2); after the alignment step is completed, each worker node acts as a regular Spark executor node. By leveraging the work of multiple machines working at the same time, FLINT is able to align a large number of reads to a large database of reference genomes in a much more efficient manner than that achieved by using a single powerful machine.

4.4 Results and Discussion

4.4.1 Benchmarking the Mapping of Reads

With the goal of benchmarking our experimental setup, we used a prototypical microbiome data set from HMP – reads from a Nasopharynx sample (sample ID SRS072346), which contained 66,780 paired-end reads. Bowtie2 took 12 hours to complete the mapping using up almost all the 768 GB RAM available on the "`castalia`" machine. Note that "`castalia`" is not a commodity machine because it has 48 processors and has a large RAM.

4.4.2 Mapping Reads on the Cloud

No. Partitions	Partition Disk Size (GB)	Bowtie2 Map Time
1	170	12:05:31
4	40	0:30:24
6	28	0:15:28
8	21	0:10:53
12	13	0:06:48
16	10	0:05:27
24	7	0:05:00
32	5	0:04:25
48	3.5	0:01:30
64	2.6	0:00:52
128	1.3	0:00:35

Table 4.1: Reference Database Partitions. Average size of a partition ("Partition Disk Size (GB)") after splitting the reference collection, along with estimated running times for aligning 66k HMP reads against the partition ("Bowtie2 Map Time").

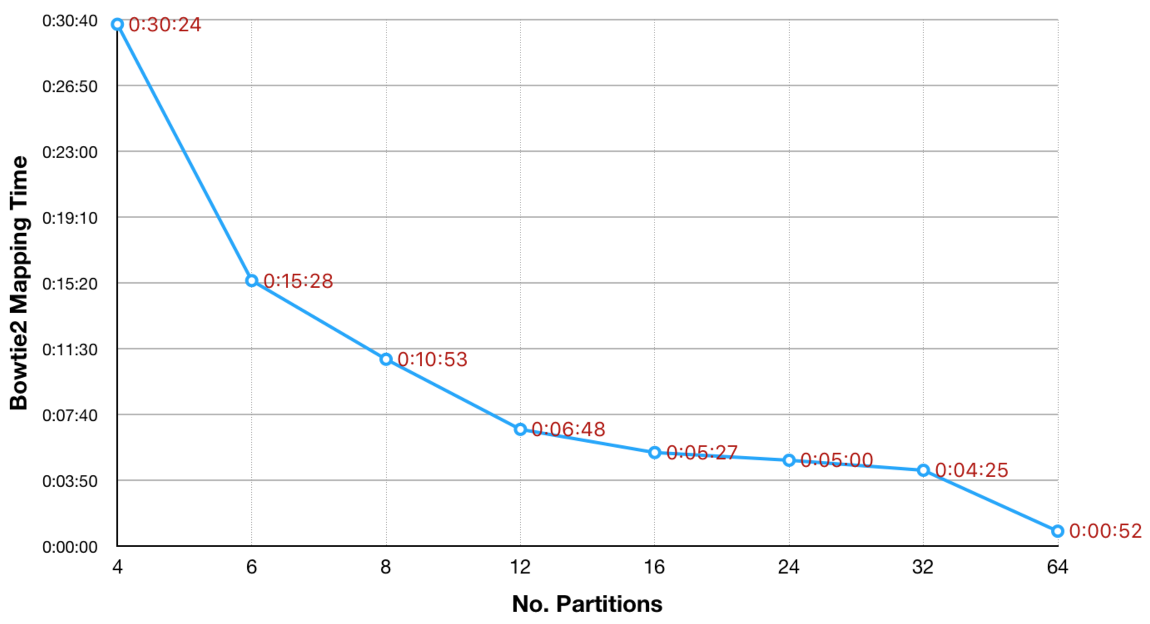


Figure 4.3: Bowtie2 Partition Map Time. Run times for aligning 66k HMP reads using Bowtie2 against different reference database partitions.

Table 4.1 shows the time taken to map all the reads against the indexed partitions for different values of N, the size of the cluster on the cloud. Our benchmarking experiments with a partition into four shards took roughly 30 minutes, while with a partition into 64 shards, it took less than a minute to map all the reads. This is shown in Figure 4.3.

Note that the benchmarking runtimes from Table 4.1 are for only one randomly chosen partition, and they do not reflect the actual times that we achieved in the cluster using all the partitions for a given partition size class.

4.4.3 Comparing Flint to Existing Tools

As part of its evaluation, we compared the abundance profiles generated by Flint to those provided by HMP and those generated by Kraken [167]. Note that Kraken is a k-mer-based algorithm to align reads to genomic sequences and is known to be one of the most accurate ones [103].

We selected an anterior nares sample (SRS019067) with 528k reads from the Human Microbiome Project (HMP) and analyzed it with Kraken (2.0.7-beta) and Flint and compared the results to those provided by HMP in their community abundance profiles. HMP reported 36.7% aligned reads using a bacterial database of 1,751 genomes, while Kraken was able to classify 36% of the reads using their RefSeq bacterial database of 14,506 genomes; in contrast, Flint was able to align 81% of the reads using Ensembl's 44K bacterial genomes. The increased number of aligned reads is due to the larger number of genomes in Ensembl. Kraken uses RefSeq's bacterial genomes collection, while Ensembl contains many draft genomes that increases the probability for mapping a read. Flint also aligns reads with Bowtie2 directly to the bacterial strain genomes, and does not apply lowest common ancestor (LCA) assignment to reads as Kraken does, which should mitigate any database diversity influences (genus, species, and strain ratios) as noted by Nasko et al. [109]. As shown in Figure 4.4, both Flint and Kraken identify roughly the same set of genera, but at the species level, Flint identifies significantly more species.

MetaSpark [178] and SparkHit [70] are methods built on the Spark framework with a cluster infrastructure similar to that of Flint, but their lack of support for large genome references makes direct comparison impossible. MetaSpark has a 201 minute runtime for 1 million reads with 10 nodes, profiled against a 0.6 GB reference of bacterial genomes from NCBI. In comparison, Flint takes 67 seconds to profile 1 million paired-end reads against Ensembl's 44,048 genomes (170 GB) with 64 nodes.

4.4.4 Reference Genome Collections

Creating the Bowtie2 index for the sharded collection of bacterial genomes is a one-time operation and the index can be reused across cluster deployments until an updated collection is available. With a 64 worker-node cluster, we created 64 reference shards, each of average size 2.6 GB. The total sequential indexing time for the 64 shards was 1d 20h 4m 33s on a single machine. When we used an LSF cluster

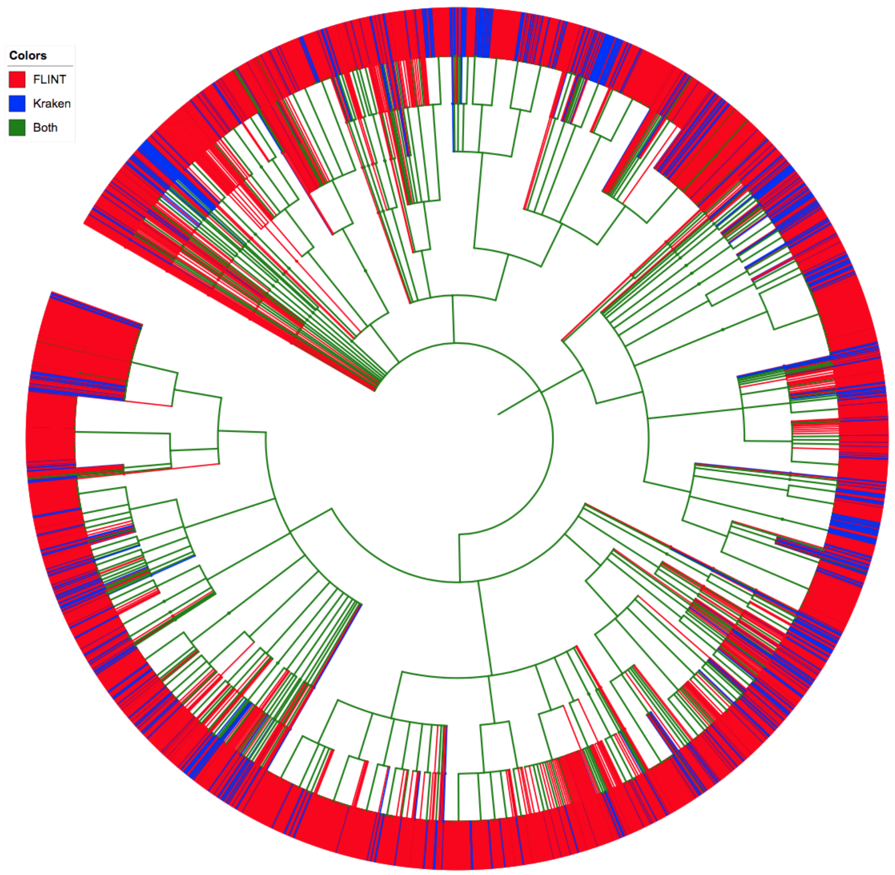


Figure 4.4: Kraken2 Comparison. Community abundance profile comparison for an HMP anterior nares sample (SRS019067) generated by FLINT and Kraken. Red branches are lineages identified only by FLINT blue branches are lineages identified by Kraken only, and green branches are lineages identified by both. The outermost ring represents the species identified by either FLINT (red), Kraken (blue), or both (green).

[145] that indexed the 64 shards in parallel, the total indexing time came down to just over three hours.

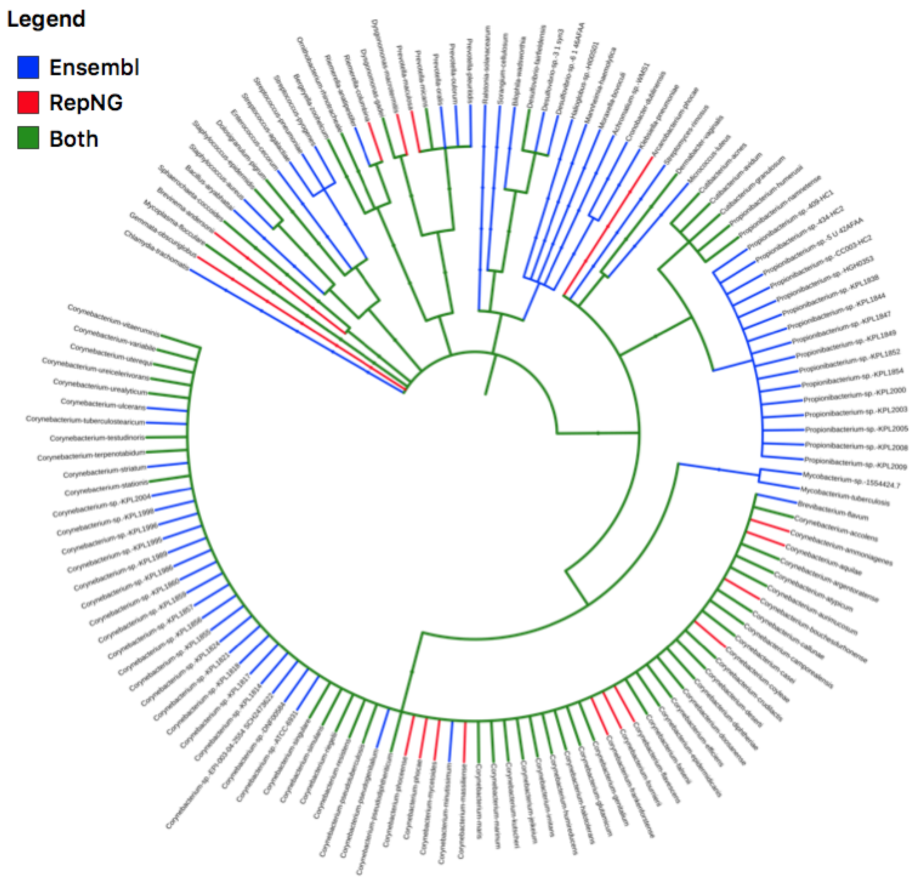


Figure 4.5: Phylogenetic tree showing taxa identified by Flint using 44K Ensembl bacterial genomes (blue), and using 5K NCBI's Genomes references (red) with an input of 1M randomly selected reads from the HMP anterior nares sample (SRS015996). Genomes are considered as identified if the average coverage in their genomic sequence is 80% or more.

Existing metagenomic profiling tools such as MetaSpark and SparkHit use an archived version of RefSeq as their reference genomes database — MetaSpark's collection of RefSeq bacterial references has a size of 1.3 GB and contains about 5K genomes. The Ensembl database used by Flint contains roughly nine times the number of genomes (and has size 170 GB), and we looked into how a metagenomic profile could be different by looking at how many genomes are identified by using a

large or small reference collection. To do this, we randomly selected 1M reads from an HMP anterior nares sample (SRS015996) and aligned its reads using Bowtie2 to two genome reference indices: the large collection created from the 44K Ensembl bacterial genomes, and a small collection created from 5,591 bacterial representative and reference genomes from NCBI's Genomes (RepNG) (these are not the same 5K genomes from the MetaSpark collection, as our collection was sourced from the bacterial genomes in GenBank at
`ftp://ftp.ncbi.nih.gov/genomes/genbank/bacteria`).

We investigated how many clades were identified by both references. Figure 4.5 displays the results and shows a phylogenetic tree (created with the Interactive Tree Of Life (iTOL) visualization tool [92]) showing the differences in the phylogenetic diversity of the taxa identified in the anterior nares sample. Genomes are called as "present" by selecting only those genomes that have an average coverage greater than 80% along their genomic sequence. Nodes at the inner level of the figure represent the phylum taxonomic level, while nodes in the outer rings are at the species level. Green branches represents the clades identified by both references, blue branches represent clades identified only using Ensembl, and red branches are clades identified only using the RepNG reference set. Note that the number of clades identified by Ensembl at the higher Class and Genus taxonomic levels outnumber those identified when only using the RepNG subset. We had expected RepNG to be contained in the Ensembl collection. However, this turned out not to be the case.

4.4.5 Experimental Setup

As mentioned earlier, the computational framework is primarily implemented using the *MapReduce* model [34], and deployed on a cluster launched using Amazon Web Services [11] *Elastic Map Reduce* (EMR) service. Each of the worker computational nodes will work in parallel to align the input sequencing DNA reads to a shard of the reference database; after the alignment step is completed, each worker node acts as a regular Spark executor node. By leveraging the work of multiple machines working at the same time, we are able to align millions of reads to the over 44K reference genomes in a much more efficient manner than either using only a single machine with considerable computational resources, or using other parallel computation approaches. Benchmarking tests were performed on Spark clusters of size 48, 64, and 128 worker nodes, all deployed using Amazon's EMR service at very low costs.

4.4.6 Measuring Accuracy Using Simulated Datasets

To get a measure of the accuracy of FLINT's read-alignment pipeline and to test the robustness of the streaming infrastructure, we simulated synthetic Illumina reads using the InSilicoSeq metagenomic simulator [58]. These were meant to test the accuracy of the overall pipeline, to verify that the streaming system would not introduce any duplicate artifacts, and to verify that the `reduce` steps in the Spark cluster did not exclude any of the output alignments. We created 12 datasets ranging in size from one read to one million reads, created with a log-normal abundance profile, and using the default error model for the HiSeq sequencing instrument available in InSilicoSeq. For each size setting, we created three replicates.

Table 4.2 outlines the results for the synthetic datasets described above. Dataset evaluations were performed on a 64 worker-node cluster on AWS, with each worker node containing 8 vCPUs and 15 GB of memory. FLINT achieves good performance with the HiSeq dataset achieving 99% sensitivity across all three HiSeq replicates. Alignment times on the 64 node Spark cluster using the database of 44K Ensembl bacterial genomes show that 1 million reads are aligned in just over 1 minute with no loss of sensitivity. The "Reads" column contains the number of reads that were generated; note that reads are mated in a unique pair, as either the *left* mate (oriented $3' - 5'$), or the *right* mate (oriented $5' - 3'$), forming a "paired-end fragment". The "Alignments" column contains the number of alignments that are produced by correctly aligning a single unique pair of reads with the correct orientation: the *left*

Paired Reads (×2)	Number of Alignments	Running Time	Alignment Rate	% Sensitivity
1	1	2s 344ms	100%	100%
5	23	2s 400ms	100%	100%
50	172	2s 376ms	100%	100%
500	1,356	2s 455ms	100%	100%
2,500	8,592	2s 517ms	90%	98%
5,000	23,791	3s 193ms	94%	99%
25,000	74,543	5s 138ms	96%	100%
50,000	103,835	8s 320ms	93%	99%
125,000	187,349	15s 788ms	95%	100%
250,000	275,917	29s 18ms	93%	97%
375,000	513,954	45s 91ms	95%	99%
500,000	617,933	1m 14s 713ms	96%	99%

Table 4.2: HiSeq Synthetic Datasets. Average alignment times and alignment rates for three synthetic datasets aligned against Ensembl's 44K bacterial genomes. Alignment rate measures the number of paired reads that were correctly aligned, and sensitivity is the proportion of aligned paired reads that were mapped correctly to the genome from which they were generated. Evaluations were performed on a 64 worker-node Spark cluster. Note that the number of reads aligned can be obtained by multiplying the number of paired reads in the first column by 2.

read aligns $3' - 5'$, and the *right* read aligns $5' - 3'$, with an *insert* gap in between them (the gap varies, and can be as low as 50 bases, or as high as 250). Note that these output alignments are not stored by the system, but rather they are processed as soon as they are generated by the worker nodes in the cluster. The "Alignment Rate" is calculated by dividing the number of correctly aligned read pairs by the total number of pairs generated for each dataset (note that a read can only be found in a single pair). The "Sensitivity" is conditional on the number of correctly aligned read pairs, and measures the proportion of aligned paired-end reads that were mapped correctly to the genome from which they were generated by InSilicoSeq.

4.4.7 Human Metagenomic Samples

After verifying the performance of the Flint system on simulated datasets, we tested the capabilities of the system on real metagenomic samples from the Human Microbiome Project (HMP) [71], which was generated using an Illumina-based sequencing system. We therefore expected a comparable performance with the HMP data as with the synthetic dataset.

Cluster Benchmarks

Before testing the system with sequencing data from metagenomic studies of human microbiome samples, we ran a benchmark of randomly sampled paired-end reads from a HMP anterior nares sample (SRS015996) to confirm our previous observations on the synthetic datasets. Each of these read datasets was then processed through the Flint system running on a 64 worker-node cluster in AWS. Table 4.3 presents the runtimes for each of the datasets. As expected, Flint processed 1 million reads from this HMP sample in about 67 seconds.

Sequencing Data from Human Samples

For the experiments reported here, we considered 753 HMP samples for their abundance profiles and surveyed the number of unique genera (i.e., genus richness) as reported in the community abundance profiles provided by HMP. Based on the distribution of genus richness values, we selected three representative samples with varying richness values (and, thus, varying diversity). We picked (*i*) an anterior nares sample (SRS019067, 528k reads) with 8 unique genera (low diversity class), (*ii*) a stool sample (SRS065504, 116M reads) with 60 unique genera (medium diversity class), and (*iii*) a supragingival plaque sample (SRS017511, 56M reads) with 133 unique genera (high diversity class). We speculated that the high diversity samples would contain

Paired Reads (×2)	Number of Alignments	Running Time	Cluster Memory (GB)
1	0	2s 320ms	4
5	36	2s 422ms	4
50	902	2s 336ms	4
500	9252	2s 316ms	4.3
2,500	53,918	2s 455ms	4.5
5,000	106,160	2s 700ms	4.9
25,000	538,594	5s 437ms	5.2
50,000	1,006,122	8s 318ms	5.8
125,000	2,349,518	17s 164ms	6.4
250,000	5,327,040	33s 950ms	7.6
375,000	8,439,356	50s 880ms	9.5
500,000	10,710,420	1m 7s 609ms	10.3

Table 4.3: Initial Cluster Benchmarks. Average alignment times on a 64 worker-node cluster for a set of randomly selected reads from a HMP throat sample. The number of alignments column contains the output alignments that were generated by each set of reads; these alignments were processed as soon as they were produced and were not stored, therefore minimizing the local storage requirements necessary for profiling metagenomic samples.

reads from a larger number of organisms, and the alignment system would spend more time and resources locating their origins in comparison to the samples with low diversity.

Table 4.4 contains the results from processing the reads from the three selected samples using Flint. The sample with the biggest number of paired-end reads, sample SRS065504 with 116 million paired-end reads, was profiled against Ensembl's 44K genomes in about 105 minutes. The sample with 56 million paired-end reads was profiled in about 94 minutes; while the sample with the lowest number of paired-end reads was profiled in 53 seconds. Note that the sample with 56 million paired-end reads had the highest genus richness of the 3 samples and took the most amount of time per read.

We conjecture that among all the HMP body sites, the gut is the most rigorously studied with the most sequenced genomes in the Ensembl database. Consequently, a much larger number of strains of each of the gut microbiomes are likely to be present in the Ensembl collection. As a result, the indexes are likely to be more dense, making the mapping of every read more time consuming.

Diversity Class	Unique Genera	Paired-End Reads	Alignment Time	Streamed Shards	Alignments per Shard
Low	8	528,988	0h 0m 53s	2	1,763,227
Medium	60	116,734,970	1h 45m 30s	234	1,471,036
High	133	56,085,526	1h 34m 51s	113	1,535,626

Table 4.4: HMP Sample Analysis. Diversity classes were established based on the number of unique genera in 753 HMP samples. Three samples were selected from each diversity class and analyzed in a 64 worker-node cluster. Alignment execution time measures the total time to align all the sample reads against Ensembl's 44K bacterial genomes. The "Streamed Shards" are the number of 250K read sets that are streamed into the cluster, and the "Alignments per Shard" is the average number of alignments produced by each shard.

Streaming Performance

The samples in Table 4.4 were streamed into the cluster through Spark's streaming engine. The entire sample is not ingested all at once, but rather, we stream in shards of each sample so that clusters do not get overwhelmed with too much data and cause a cluster failure. To determine the ideal number of reads to include in a stream shard, we studied the results in Table 4.3 and Figure 4.6. Panel B in Figure 4.6 displays a logarithmic curve of the alignment times for all 12 sizes of the paired-end read datasets, and while we can align one million reads in about 67 seconds, doing so creates so many alignments that each of the Spark executor processes running in each worker node could run out of memory. We looked for the "knee-in-the-curve" in

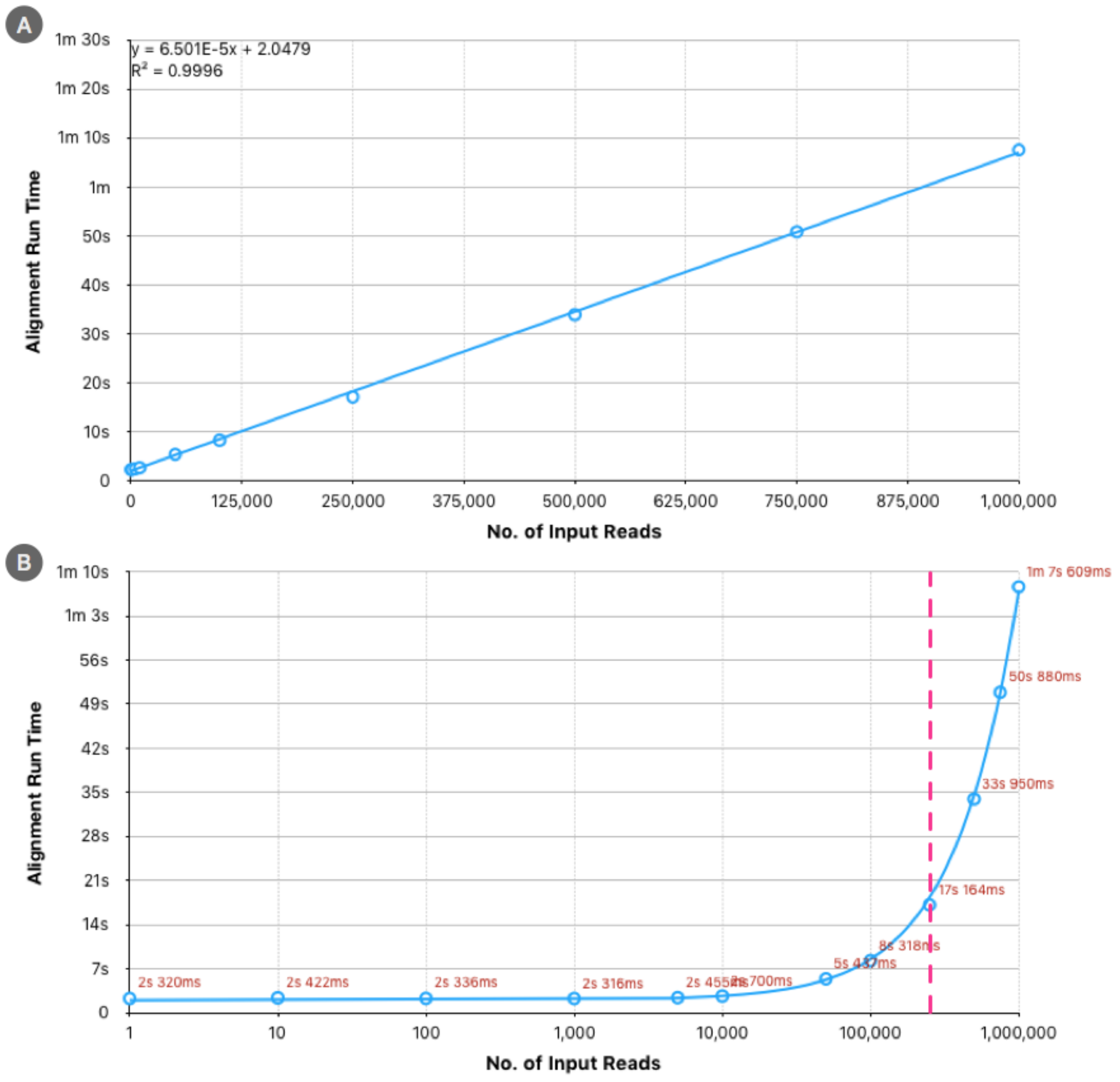


Figure 4.6: Initial Benchmarks. Figure A displays the running time for 12 paired-end read datasets on a 64 worker-node cluster. These 12 datasets were used to estimate the optimal number of reads that a 64 worker-node cluster could handle without memory saturation or network overloading. Note that while 1 million paired-end reads can be mapped in 67 seconds against 44K bacterial strains, it is not ideal to ingest 1M reads at once as the cluster's memory will be overwhelmed with the alignments that are produced. Figure B displays the logarithmic running time of the 12 datasets, and the 250K paired-end read dataset was chosen as a good trade-off between speed and resource availability.

Figure 4.6 panel B, marked by the vertical magenta line, and identified a size of 250K paired-end reads as a good trade-off between shard size and cluster performance. When we analyzed the three HMP samples in table 4.4 we set the streaming shard size to this optimal size of 250K reads, and 2 shards were created for the anterior nares sample (low diversity, 500K reads), 234 shards were created for the stool sample (medium diversity, 116M reads), and 113 shards for the supragingival plaque sample (high diversity, 56M reads). The performance results suggest that the choice of shard size was good.

Cloud Cost Analysis

All experiments were conducted using Amazon's Elastic MapReduce service (EMR) [6] and used the `c4.2xlarge` machine instance type. At the time of the experimental runs, each machine cost \$0.123 USD in the Amazon's Spot market [10]. All results reported here were obtained on a cluster of 65 total machines (64 worker-nodes, one master node) with a cost of \$0.123 USD per node, for an overall cluster cost of \$8.00 per hour. Thus, processing cost averages to \$ 0.15 USD per million reads and \$ 150 USD for a billion reads, making it extremely affordable for even the average researcher across the globe.

4.4.8 Scaling to Larger Reference Collections

The current architecture of FLINT is designed to work with large reference collections such as Ensembl's "Bacteria" collection. Figure 4.1 illustrates the current architecture of the system, and one of the primary concepts of the current cluster layout is that each worker node is provided with a single piece of the partitioned reference collection. This layout works well with the current size of Ensembl (44K genomes) but much larger collections such as RefSeq's full catalog might pose a problem as they contain upwards of 170K genomes.

It must be noted though, that RefSeq's full catalog contains a lot of redundancy as closely related genomes are part of the catalog; this is in contrast to Ensembl's strategy of maintaining a very well curated database that aims to have a wide breadth of taxonomic coverage, while mitigating redundant species. Our discussion here is not specific to RefSeq's 170K genome catalog, but to how our architecture will handle scalability as the reference genome collection grows in the future.

One approach to handle larger reference genome collections is to modify the system so that each worker node is given multiple partitions of the reference collection, rather than just a single one like in the current implementation (Listing 4.1). In this modified layout, RefSeq's 170K genomes could be partitioned into N partitions (say 256), and each worker node could be given a small subset of these (say four per node). As long as we maintain the size of each partition small enough (say 3 GB), we should be able to keep using clusters of size 64 with the current configurations from Section 4.3.3. However, if we wanted to support larger partitions, then we would need to increase the capacity of the worker machines so that we could handle the increase partition sizes (we would need more RAM memory and CPU resources).

Creating the necessary number of partitions to support a larger collection would proceed with the same steps as those outlined in Section 3.2.1, with the only difference being that the number of partitions to generate would be much higher. Before partitioning the reference collection, we would have to establish what type of cluster we would be using (number of CPUs, RAM, etc.) and how many worker machines the cluster would have. Care would have to be taken so that the number of partitions given to each worker node is uniform, this would mitigate any issues with workers being idle because they have a small number of partitions compared to others. Indexing would remain unchanged from the steps in Section 3.2.1 as the indexing step follows the partitioning step.

Each worker node in the cluster would now be responsible for aligning the input set of reads across multiple partitions, instead of aligning them across a single one as before. The process of aligning reads against multiple partitions would cause a delay in getting data back to the master node, so the streaming settings and task managers would have to be modified to account for the delays, but the distributed architecture of the cluster should be able to support it without much engineering modifications.

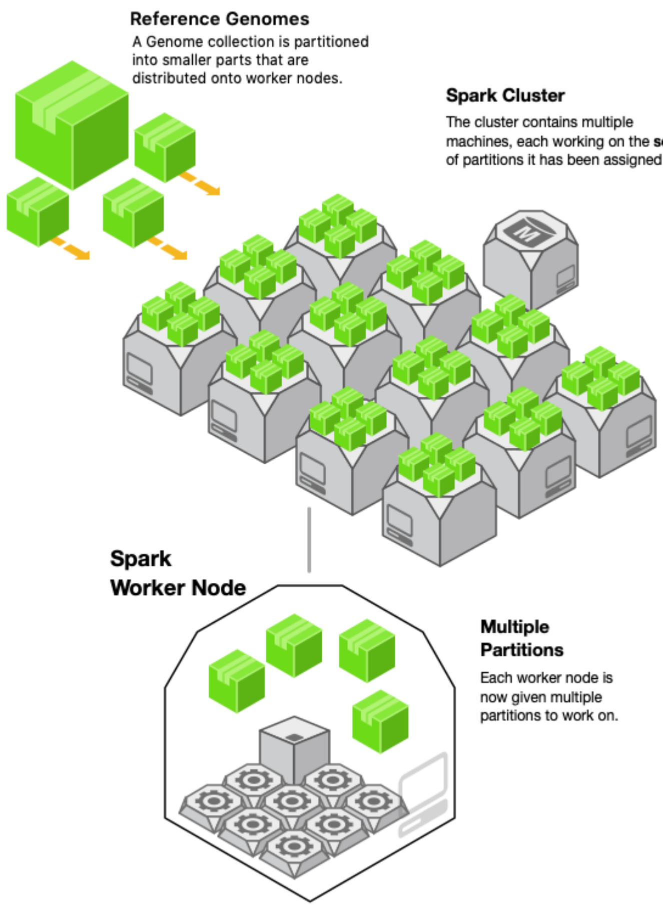

Reference Genomes
A Genome collection is partitioned into smaller parts that are distributed onto worker nodes.
Spark Cluster
The cluster contains multiple machines, each working on the **set** of partitions it has been assigned to.
Spark Worker Node
Multiple Partitions
Each worker node is now given multiple partitions to work on.

Figure 4.7: Modified Cluster Architecture. Support for larger reference collections such as RefSeq could be achieved by having each worker node in the cluster work on multiple partitions, rather than a single one as is the case in the current architecture.

The script for handling the alignment to a reference genome is `spark_jobs.py`, and the function `align_with_bowtie2()` is main code block that needs modification. The current function dispatches a Bowtie2 subprocess to align the reads. The function relies on having a single `index_path` and a single `index_name`. To support multiple reference indices, we would need to modify the function in Listing 4.1 so that it can loop through a collection of multiple partitions. Lines 4 through 14 in Listing 4.1 would be placed in a loop that would iterate through the different partitions, and append its set of resulting alignments to the `alignments[]` list declared in line two.

```
1  def align_with_bowtie2(iterator):
2      alignments = []
3      reads_list = broadcast_sample_reads.value
4      bowtieCMD = getBowtie2Command(bowtie2_node_path=bowtie2_node_path,
5                                    bowtie2_index_path=bowtie2_index_path,
6                                    bowtie2_index_name=bowtie2_index_name,
7                                    bowtie2_number_threads=bowtie2_number_threads)
8      try:
9          align_subprocess = sp.Popen(bowtieCMD, stdin=sp.PIPE,
               stdout=sp.PIPE, stderr=sp.PIPE)
10         pickled_reads_list = pickle.dumps(reads_list)
11         alignment_output = align_subprocess.communicate(
12             input=pickled_reads_list.decode('latin-1')
13         )
14         for a_read in alignment_output.strip().decode().splitlines():
15             alignment = a_read.split("\t")[0] + "\t" +
                   a_read.split("\t")[2]
16             alignments.append(alignment)
17     except sp.CalledProcessError as err:
18         sys.exit(-1)
19
20     return iter(alignments)
```

Listing 4.1: Original alignment function (single partition).

4.5 Conclusion

In this chapter we have shown how large metagenomic samples comprising millions of paired-end reads can be profiled against a large collection of reference bacterial genomes in a fast and economical way. Our implementation relies on the Spark framework to distribute the job of aligning millions of sequencing reads against Ensembl's large collection of 44K bacterial genomes.

By combining streaming, a sophisticated double MapReduce algorithm, affordable cloud computing services, FLINT brings sophisticated metagenomic analyses within reach of small research groups with modest resources.

Additional materials, simulation datasets, partitioned reference indices, and links to the GitHub website with the source code and installation instructions can be found on the FLINT project website at
`http://biorg.cs.fiu.edu/flint`.

CHAPTER 5

MICROBIOME MAPS: HILBERT CURVE VISUALIZATIONS OF MICROBIAL COMMUNITY PROFILES

5.1 Introduction

Microbiome samples are now routinely created by means of low-cost, high throughput metagenomics DNA sequencing. The next steps in the analysis is the creation of microbial community abundance profiles [25], obtained by mapping the sequenced reads against a collection of microbial genomes from a reference genome collection like Ensembl [42] or RefSeq [113]. Tools such as Flint [153] and Kraken 2 [166] facilitate the process to create detailed microbial abundance profiles for tens of thousands of genomes for both metagenomic whole-genome DNA sequencing (mWGS) as well as 16S-amplicon sequencing (16S).

Metagenomic profiles contain relative abundance values for the entire collection of microbial taxa present in a sample, and these profiles can be easily viewed with stacked bar charts, pie charts, Krona plots [115], or using data analysis software suites such as Tableau [141], or MS Excel [106]. Such tools are readily available to the public and allow for data exploration, but are designed for the analysis of generic tabular data, and do not consider domain-specific information (taxonomic, phylogenetic, etc.) that may be crucial for the interpretation of metagenomics datasets. Recent tools such as WHAM! [38] allow for explicit metagenomics-focused analyses, making it possible to dig down into the data and create useful visualizations for descriptive analyses.

We argue that the complex latent properties of microbiomes embedded in community abundance profiles such as taxonomic hierarchies and other relationships are not easily described and visualized in traditional generic plotting mechanisms such as stacked bar-charts or line-plots. The problem gets more acute as the sizes of our reference genome collections continues to grow exponentially over time [109], and even novel tools such as Krona are not able to keep up and display large amounts of reference genomes (Figure 6.4).

In this work we consider the problem of succinctly visualizing a microbiome, and specifically, visualizing metagenomic community abundance profiles, along with their latent structured labels (e.g., taxonomic description or biological property) with the use of a visualization technique called the *Hilbert Curve Visualization* (HCV). We discuss the challenges of scalable visualization of billions of measurements for hundreds of thousands of microbial genomes, and argue that alternative visualization techniques are useful when trying to combine many factors of metagenomic information in order to create interpretable images that can lead to improved understanding.

5.2 Approach

As mentioned earlier, we use a technique called the Hilbert curve visualization (HCV) to succinctly visualize the microbial community abundance profiles of a large number of genomes (up to 44K). These profiles contain the relative abundance measurements of thousands of genomes, and they are ordered along a space-filling curve in a 2D square using the Hilbert curve [66], thus making it possible to visualize the profile of a single metagenomic sample. In the resulting Hilbert image, each position is a genome from the reference database, and the intensity of the position's color value represents the abundance of a genome in the sample.

As discussed below, depending on the ordering of the genomes that is selected, different "microbial neighborhoods" are created, allowing for different interpretations of the clusters of bright segments, i.e., "hotspots", of abundant genomes in the images. Fixing the position of a genome results in visualizations that allow for quick comparisons of the abundance of the same genome or sets of genomes in multiple microbiome samples.

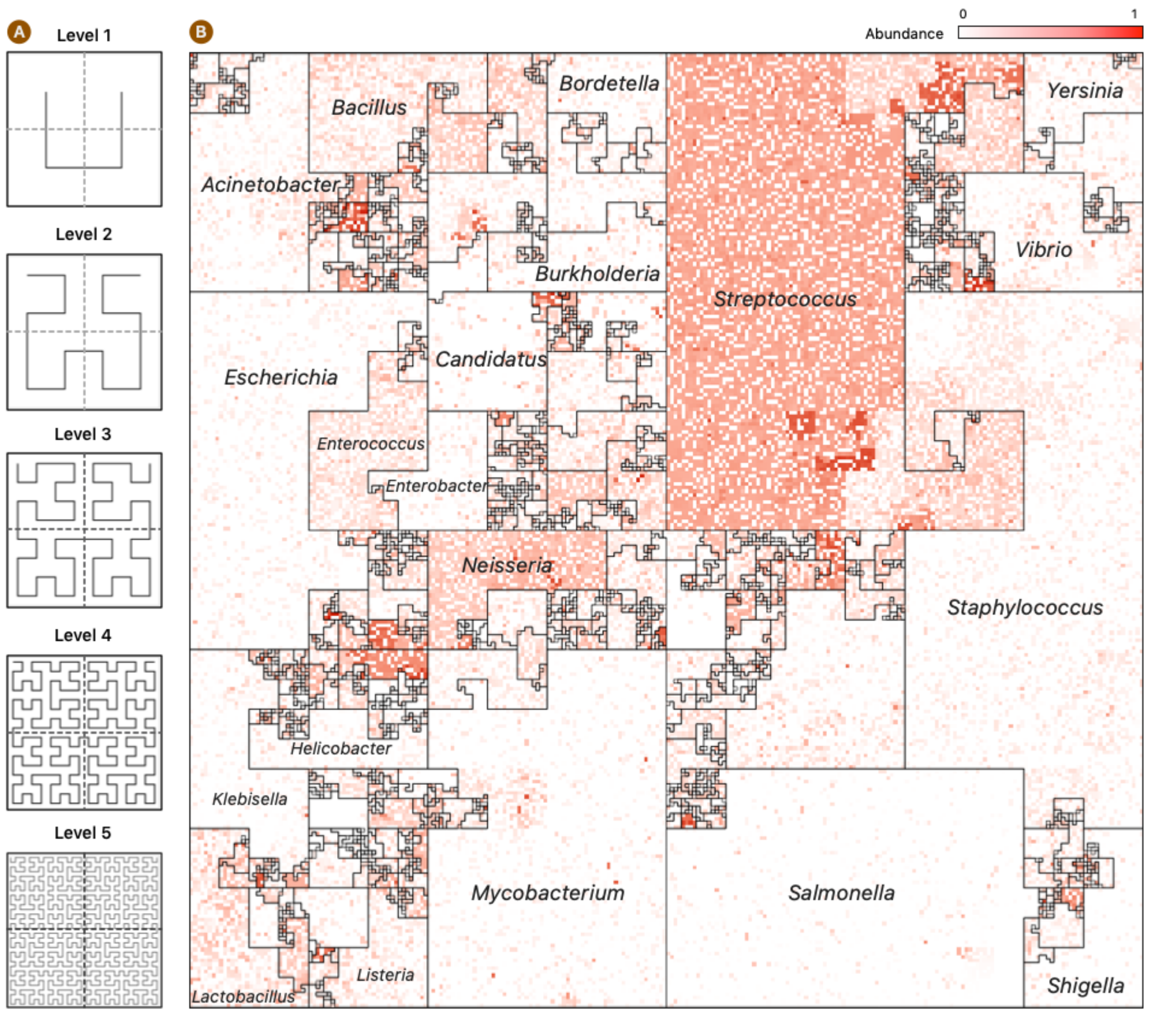


Figure 5.1: Hilbert Curve Visualization of Metagenomic Samples. (A) The first five iterations of the Hilbert curve: the "Level 1" curve is obtained by connecting the centers of the four initial squares as shown; the Level k curve is obtained by a recursive partitioning of each square from Level $k-1$, creating four Level $k-1$ curves and connecting them as outlined by the "Level 1" curve, rotated appropriately. At level k, the original square is divided into $2^k \times 2^k$ small squares, each of whose centers is visited by the Level k Hilbert curve. **(B)** A representative image of a mWGS Buccal Mucosa sample (SRS045254) from the Human Microbiome Project (HMP) created using a "taxonomic ordering" of 44K reference genomes from the Ensembl database. The intensity of each position in the image represents the abundance of one microbial genome. Groups of segments are labeled by the groups induced by the ordering of the taxa on the Hilbert curve. Additional Hilbert curve images for more mWGS samples, as well as other 16S samples at full resolution, are available via links from `biorg.cs.fiu.edu/jasper`

5.2.1 Space-Filling Curves

Space-filling curves are popular in scientific computing applications for their ability to speed-up computations, optimize complex data structures, and simplify algorithms [18]. Trees are particularly interesting structures that can be optimized with space-filling curves because it is possible to generate sequential orderings of the nodes of the tree in which parent and children nodes are neighbors in a 2D plane. The combination of trees and space-filling curves has been shown to be useful in many fields [16], and in metagenomics this can be useful because the microbial genomes in a reference database are classified using a taxonomy tree with a hierarchy of levels (Strain, Species, Genus, etc.). For data from mWGS experiments, we can presume the leaf nodes of the taxonomy tree to be microbial strains (Figure 5.3, panel (B)); for 16S data, the leaf nodes are usually species or genera. Clades of the taxonomy tree correspond to microbial neighborhoods in the resulting visualization.

5.2.2 The Hilbert Curve

The Hilbert Curve is one of the more prominent examples of space-filling curves, and its construction is based on a recursive partitioning of a square region in the 2D plane and connecting the centers of these squares in a specific order. To provide a recursive definition of the curve, Figure 5.1 (A) shows the curve at Level 1 when there are only four squares to connect. The curve at Level k is defined recursively by dividing the original square into four squares, each with a Level $k-1$ curve in it and then connecting these pieces using the template of the Level 1 curve after appropriate rotations of the four curves (Figure 5.2).

Many applications exploit the order that space-filling curves impose on data, and a particular application has been the visualization of high-dimensional data. The first use of the Hilbert Curve as a visualization tool was proposed by Keim in 1996 [77] to represent stock market data, and since then, it has been used for visualizing genomic

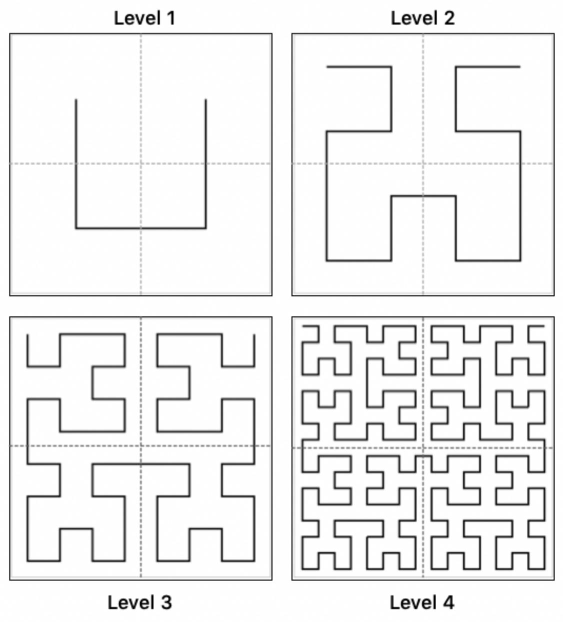


Figure 5.2: Detailed Hilbert Curve Levels. First four iterations (levels) of the Hilbert curve show the 2D square recursively partitioned into 4 sections at each iteration (level) of the curve.

data [35, 12], and also for visualizing DNA alignments of whole bacterial genomes [165].

In human genomics, using the HCV is a straightforward affair as the natural linear order of genomic positions can be easily visualized by the curve, and there are tools and software libraries for creating Hilbert curve images from human sequencing experiments such as HilbertVis [12] and HilbertCurve [59]. Both HilbertVis and HilbertCurve apply HCV to genomics in the context of a single human genome: a single genome scaffold is modeled as a one-dimensional (1D) line in which each interval is taken to be a single genomic position, but the HilbertCurve library can also stitch together multiple chromosomes so that they can all be displayed on a single image, but the display is nevertheless that of a single genome. To date, the HCV technique has not been applied to metagenomics datasets.

5.2.3 Ordering the Genomes Along the Space-Filling Curve

Visualizing microbial community profiles is non-trivial as there exists no natural linear ordering for the thousands of genomes in the reference database, and we argue that in trying to order thousands of genomes the answer is not a single "true" order, but rather, multiple different orderings that provide different perspectives of the metagenomics data being visualized.

Custom orderings based on a single sample (say, ordered by decreasing abundance values in a single sample) are not that informative when using a HCV, since the main advantage of these visualizations is in the ability to locate the same genome in the same position on the image for different related samples. Even using the average abundance value from a cohort is not so useful because different cohorts cannot be readily visually compared.

Thus, the value of our visualizations lies in specifying useful orderings that are not dependent on a single attribute of one, or multiple samples, but rather, on some global properties and biological interpretations that are useful for the researcher.

Linear Orderings: Below, we discuss two classes of orderings that are shown to be useful for the visualization.

1. **Taxonomic Ordering:** Ordering of reference genomes as per the taxonomic tree from Ensembl Genomes [43].
2. **Labeled Ordering:** Ordering as per a user-supplied labeling scheme.

We show with experiments in Chapter 6 that the microbiome maps resulting from the orderings mentioned above are expressive in that researchers can use them to quickly identify groups of taxa that are abundant or differentially abundant without losing any context imposed by the ordering (i.e., taxonomic hierarchies or biological

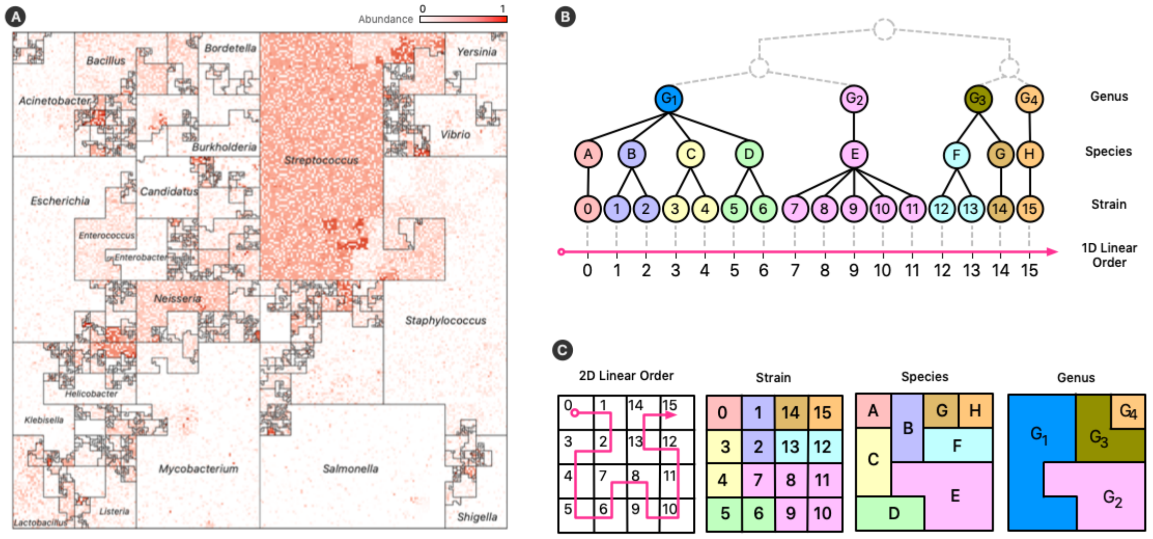


Figure 5.3: Taxonomic Ordering. (A) A *Taxonomic Ordering* of 44,048 microbial reference genomes from Ensembl Bacteria. *Microbial neighborhoods* are drawn based on a taxonomic tree for microbial classification. The image depicts the distribution of Genera in the reference genome database, and the size of each neighborhood is consistent with the number of genomes that belong to it. **(B)** A taxonomic hierarchical tree with three (3) levels of rankings for the genomes in the reference database: Genus, Species, and Strain. The taxonomic tree is linearized to create a 1D linear order for the tree's leaves (Strains). **(C)** The 1D linear order is then laid out onto a 2D plane using a Hilbert curve, which creates *microbial neighborhoods* of related taxa.

conditions). Other useful orderings may exist and JASPER provides a basic framework that allows one to creatively explore other possible orderings.

Microbiome Maps offer a quick and visual way to readily identify "hotspots" in "microbial neighborhoods", i.e., groups of related microbes, that can contextualize the important features of the metagenomic samples under study.

5.3 Methods

Visualizing a microbiome's abundance profile starts by aligning, or classifying, sequencing reads against a reference collection of microbial genomes, and creating counts of the number of reads that align (or are classified) to each genome. The Flint

[153] software facilitates the profiling of mWGS datasets, while the Kraken 2 software does it for 16S datasets. For the images shown in this paper, Flint uses a reference collection of 44,408 microbial genomes from the Ensembl Bacteria database [42], while Kraken 2 uses a "16S" reference of 5,127 genomes which contains references from Greengenes [37], SILVA [122], and RDP [32]. Once the profiles have been created, we use them as input to the Jasper tool which creates a Hilbert image for each sample.

The input to Jasper can represent a single metagenomic sample, or multiple ones if given in a labeled matrix, and its output will be metagenomic Hilbert images for each sample provided as input. Documentation on the type of profiles, along with matrix formatting guidelines, are all available in the documentation page of the GitHub repository, and detailed parameters, along with sample command calls for both Flint and Kraken 2 are available in the project website.

Different linear orderings of the taxa on the Hilbert curve result in different images. Users may select from two options: a *taxonomic ordering* (Figure 5.3) which uses the linear order from Ensembl's taxonomic tree, or a *user-defined labeled ordering* (Figure 5.4) which determines the order based on a labeling from a biological grouping of samples provided by the user. Both orderings place microbial genomes along the curve in a specific order. Note that unlike other genomic HCV techniques, metagenomic Hilbert images do not depict single genomic positions, but rather, they display thousands of genomes.

5.3.1 Microbial Neighborhoods

Different linear orderings produce different Hilbert visualizations, with each resulting in clusters of related microbes along neighboring regions in the 2D plane. The clustering creates unique areas that resemble community neighborhoods in popular consumer mapping applications like Google Maps [56], and we term these areas *Microbial Neighborhoods* (Figure 5.3, panel (A)) as they represent microbes belonging to

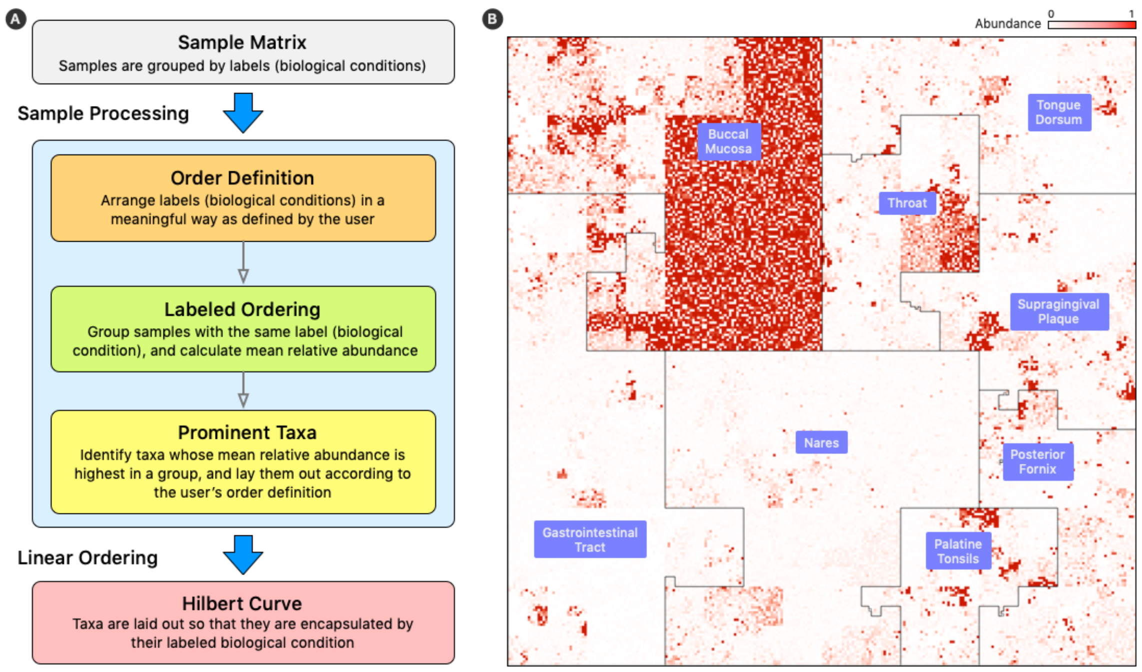

A
Sample Matrix
Samples are grouped by labels (biological conditions)
Sample Processing
Order Definition
Arrange labels (biological conditions) in a meaningful way as defined by the user
Labeled Ordering
Group samples with the same label (biological condition), and calculate mean relative abundance
Prominent Taxa
Identify taxa whose mean relative abundance is highest in a group, and lay them out according to the user's order definition
Linear Ordering
Hilbert Curve
Taxa are laid out so that they are encapsulated by their labeled biological condition
B
Abundance
0
1
Buccal Mucosa
Tongue Dorsum
Throat
Supragingival Plaque
Nares
Posterior Fornix
Gastrointestinal Tract
Palatine Tonsils

Figure 5.4: Labeled Ordering. **(A)** Samples are processed in a M x N matrix that contains M labeled samples, and N microbial taxa. The user specifies the ordering of the biological conditions that the M samples belong to. For each labeled grouping, the taxa with the highest mean relative abundance is identified and used as an anchor for a linear order in the Hilbert curve.**(B)** Resulting Hilbert curve image for 3 samples (SRS024557, SRS045254, SRS063478) for a buccal mucosa condition, built using an averaging ordering scheme from 8 HMP body sites.

either the same taxonomic group, or the same biological condition — the idea being that they are clustering around a common theme (taxonomic or biological).

One advantage of the *Microbial Neighborhoods*, using the *Taxonomic Ordering* is that when the genomes are laid out in the Hilbert image without any abundance color mappings, the distribution of the reference genome database can be quickly illustrated (Figure 5.3, panel (A)). Larger clades with more taxa (leaves) occupy larger tracts of the image. Thus, we can see from Figure 5.1 (B) and 5.3 (A) that the Ensembl database contains a large number of strains from the *Streptococcus* and *Staphylococcus* genera. Displaying a reference genome database using the Hilbert curve is a way of understanding the microbial diversity of the database.

Taxonomic Neighborhoods

The first option for ordering genomes along the Hilbert curve is the *taxonomic ordering* which determines a 1D linear order based on a genome's taxonomic lineage. In this ordering, pairs of taxa belonging to the same taxonomic group (say the same Genus or Species) are placed close to each other along the curve, and consequently, close to each other in the Hilbert image. This ordering scheme creates *Taxonomic Neighborhoods* that envelop related taxa based on their taxonomic lineage, and as seen in Figure 5.3, multiple taxonomic levels can be displayed at the same time in a single image. The ability of the Hilbert image to display multiple levels of the taxonomic tree all at once, while at the same time providing high-resolution abundance information for single genomes, is a compelling advantage over visualizing data with other means as the sheer number of data points (single genome measurements) would overwhelm any other 1D visualization.

Linearizing Trees

Illustrating a taxonomic tree as a 2D Hilbert curve starts by finding a linear order of the leaf nodes in the tree. Figure 5.3, panel (B), depicts a fictitious taxonomic tree with 16 leaf nodes (microbial strains in this example) that are ordered along a 1D line using a *taxonomic ordering* scheme (section 5.3.1) which groups the 16 strains according to their parent species and genus groups. Figure 5.3, Panel (C), illustrates how the 16 strains would be laid out on a 2D plane, and how the taxonomic hierarchies are represented as strain, species, and genus areas in the Hilbert image.

Note that establishing a linear order for a tree structure is non-trivial as trees do not have a "start" or "finish", nor do they have a "right" side or a "left" side. Different orderings result by performing a permutation of the children at any given node of the tree. Algorithms for finding an optimal order have been proposed such as the one described in [17], but the optimal order relies on the tree having a certain property such as it being a binary tree, and for a distance measure to be calculated.

The *taxonomic ordering* linearizes a tree by using taxonomic data taken from Ensembl's Pan-taxonomic Compara [44] and Ensembl Genomes [43] databases, and we use Ensembl's taxonomic information as the basis for building a taxonomic tree structure that is linearized and used as the foundation for the Hilbert image. The genomes in the reference database are annotated so that we can establish a lineage up to the phylum level, and in the case when we profile mWGS data, the leafs of the tree are taken to be microbial strains; in the case when we profile 16S data, then the leafs of the tree are taken to be microbial species. For both mWGS data and 16S data, the 2D square that bounds the Hilbert image represents all the genus-level groups in our taxonomic tree; we do this because we found that drawing the curve at the genus level amounts to a good compromise between information and visual appeal.

Condition Neighborhoods

The second option for ordering genomes along the Hilbert curve is the *labeled ordering* (Figure 5.4), which creates "*Condition Neighborhoods*" by using an ordering scheme that determines the 1D linear order based on a user-supplied labeling of samples provided as a labeled $m \times n$ sample matrix M, where m are sample rows, and n are the genomes in the reference database; if we have i samples, and j reference genomes, then the cells in the matrix correspond to abundance values for genome n_j in sample m_i.

Establishing the 1D linear order for multiple biological conditions (Figure 5.4, panel (A)) starts by ordering the number of conditions k in some meaningful order, $C_1, C_2, ..., C_k$, provided by the user. Note that the conditions may represent different disease states (if we are comparing disease samples), time intervals (if we are comparing a time series), or locations (if we are comparing body sites, or environmental sites); the conditions are not limited to the aforementioned list, as users can supply their own. Once we have a condition ordering established, the next task is to identify taxa whose average relative abundance is highest in C_1 and order them first, followed by taxa whose average relative abundance is highest in C_2, and so on, until we terminate the ordering by taxa that are not abundant in any of the conditions. Once we have established the ordering, we can then draw the Hilbert images for each of the samples from the input sample matrix M.

Although one can argue that every taxon must be highest in one of the k conditions, this is not meaningful unless its presence is above the threshold of noise, which we determine when we normalize the input matrix M. In general, if a taxon is most abundant in multiple conditions (something that we have not seen in practice), then we assign it to the first condition as determined by the input ordering criteria. After the conditions have been organized along the curve, taxonomic information is used to order genomes within the range of the condition.

Note that in this ordering, the Hilbert image is still visualizing one sample, but it is displaying the abundance characteristics of the biological condition for which the sample is most prevalent in — "*hotspots*" will therefore appear in one of the conditions, and users can readily tell what condition the sample belongs by identifying the area, i.e., neighborhood, that represents the condition in the image. Clusters of bright positions will also appear in other neighborhoods (Figure 5.5), as other conditions will contain taxa with high relative abundances, but not in the same quantities as for the condition that the sample belongs to.

Adding New Samples and Genomes

For the taxonomic order, the preservation of a genome's locality in the plot is a key advantage when adding new samples, as they can be added to a dataset of microbiome maps without having to modify the map's topology significantly: the dimensions of each microbiome map ($width \times height$) correspond to the number of genomes in our reference database, and as long as the reference database does not change, then the microbiome map's underlying topology should remain constant for new samples.

For the labeled ordering, the preservation of locality for a genome is a little more delicate as the assignment of a genome to a neighborhood is done by identifying taxa whose mean relative abundance is highest in a group of samples. If the new sample to be added changes a taxa's mean relative abundance in the sample's group, then it could affect the map's topology. Note that this is a disadvantage to all mapping orders that rely on precomputing a value to place taxa in a neighborhood — specially when that value is computed across a group of samples, as new samples will require their re-evaluation.

Adding new genomes to the reference collection could affect all existing plots because the positions of the existing genomes in the map could change when new genomes are added in the middle of the ordering. One possible solution is to leave "blank" (i.e., unassigned) pixels or areas on the map to allow for future additions to the reference database. This could be implemented by inserting the gap so that the next clade always starts at the boundary of a square region of size $2^k \times 2^k$, for a predetermined value of k. Nevertheless, the addition of new taxa into an existing ordering will also change the relative abundance of all or almost all taxa changing the intensities of the pixels by some amount.

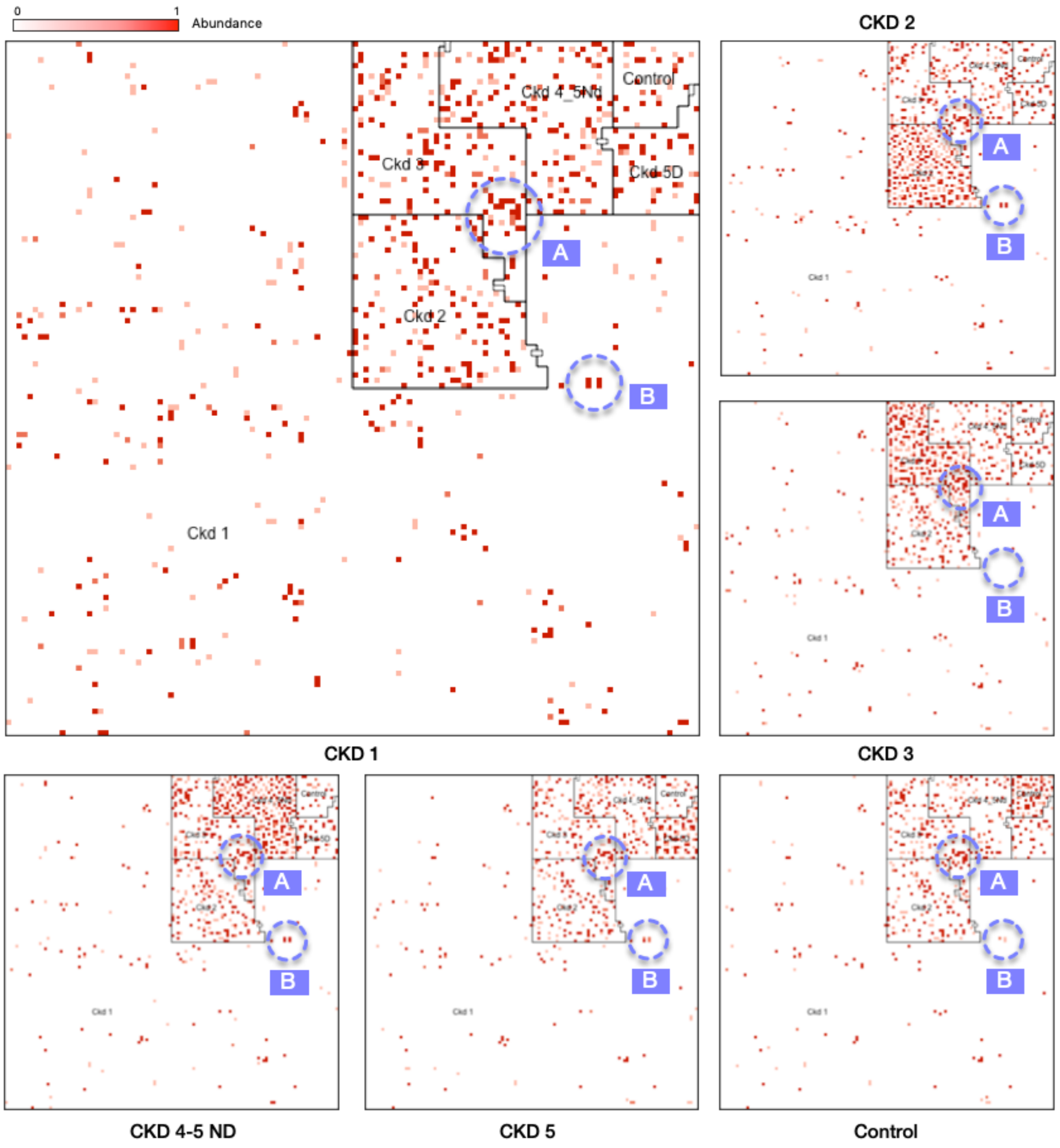


Figure 5.5: Chronic Kidney Disease Stages. Hilbert curve visualizations for 16S samples of five stages of chronic kidney disease, along with a control sample. Each panel represents the mean relative abundance of 3 samples for each stage, and displays 5,127 species. Regions marked *A* shows a group of microbes that appear in all stages, while region marked *B* appears in almost all stages except CKD3 (absent), and Control (lowered abundance).

5.4 Results and Discussion

We created "microbiome maps" for two groups of metagenomic datasets: 24 mWGS normal samples taken from the Human Microbiome Project (HMP) [71], and 18 fecal samples (16S) from a collaboration with Kangwon National University and Seoul National University in Korea. The 24 samples from HMP represent 8 different body sites, and the 18 samples from the Korea study represent five stages of Chronic Kidney Disease (CKD), along with a normal control set. We analyzed the mWGS HMP samples with the Flint software [153], and the 16S CKD samples with Kraken 2 [166]. For the HMP samples, the metagenomic profiles contained relative abundance measurements for 44,048 microbial strains, and for the CKD samples, the metagenomic profiles contained relative abundance measurements for 5,127 microbial species.

Three samples were selected for our study from HMP from each of the eight following body sites: Buccal Mucosa, Gastro-Intestinal Tract, Nares, Palatine Tonsils, Posterior Fornix, Supragingival Plaque, Throat, and Tongue Dorsum.

Eighteen fecal samples were obtained from CKD patients of Kangwon and Seoul National University Hospitals. The samples were selected based on their glomerular filtration rate (see Kidney Disease Improving Global Outcomes (KDIGO) [78]), and a total of six groups were created: Control, CKD Stage 1 (CKD 1), CKD Stage 2 (CKD 2), CKD Stage 3 (CKD 3), CKD Stage 4 & 5 non-dialysis dependent (CKD 4-5ND), and CKD Stage 5 dialysis dependent (CKD 5). The CKD stages were determined based on the worsening function of the kidney patients, and three samples from each group were used.

5.4.1 Comparison to Other Methods

The WHAM! and iMAP suite of tools for exploring the profiles create standard visualizations including stacked-bar plots, cluster heatmaps, etc., which are helpful for condensing the summarized information for multiple samples. However, they lack the ability to convey nuanced information on about 44,000 bacterial strains in a single sample while retaining the perspective of the latent metagenomic relationships (common and/or unique taxa). The Hilbert curve visualizations are easy to interpret: with a quick glance, one can capture the prominent taxonomic groups and even capture groups of co-occurring taxa. With a shift in perspective, the visualization can help to identify the body site (for HMP) or disease condition (CKD) that characterizes the samples.

5.4.2 Metagenomic Visualizations

The community abundance profiles from FLINT and Kraken 2 for the HMP and CKD datasets were converted into Hilbert curve visualizations: for both datasets we created a set of images using a *taxonomic ordering*, and another set using a *labeled ordering*. Note that the images with the *taxonomic ordering* contain taxonomic clade border lines at the Genus level, because we found this level of resolution to be a good trade-off between image interpretability and taxonomic lineages. Going to a higher level would have resulted in images with vast neighborhoods, and going a level deeper would have resulted in images that contained too many borders. For images done with the *labeled ordering*, the clade border lines are drawn so as to fence off the different labels specified by the user.

Metagenomic Hilbert images for mWGS and 16S data communicate abundance information at different levels of a genome's lineage: for mWGS samples, each position in the image displays information about microbial strains (the resolution at which abundances are reported by FLINT[153]); for 16S samples, each position in the image displays information about microbial species (the resolution at which abundances are reported by Kraken 2 [166]).

Figure 5.3, panel (A), contains a representative image from one sample of the HMP dataset (sample SRS019119, Nares) ordered using the *taxonomic ordering* scheme. In this image we can clearly see that the *Streptococcus* and *Staphylococcus* groups are abundant in the Nares sample. While the dominant group would have been obvious even in a traditional 1D plot, the Hilbert curve visualization ensures that the smaller taxonomic groups are not overshadowed by the more abundant groups. Identifying the most abundant taxonomic clades in a sample only takes a quick glance at the image.

Figure 5.4, panel (B), contains another representative image from a Buccal Mucosa sample from the set of 18 HMP samples. This image also contains the same 44K reference genomes from 5.3, panel (A), but they have been ordered using the *labeled ordering* scheme. Using this scheme, we have ordered the reference genomes based on the highest mean relative abundance of a genome in its labeled cohort, and the resulting image can then display the profile of any sample (or the cohort's average) using the computed order. The advantage of this ordering scheme is that identifying the biological condition that the sample belongs to is effortless: one need only look at the neighborhood that contains the most *hotspots* (Buccal Mucosa in this case). Additionally, one can readily see that the buccal mucosa site shares some microbes with those typically abundant in the throat, palatine tonsils, posterior fornix, supragingival plaque, tongue dorsum, and the GI tract.

Figure 5.5 comprises 16S samples from the CKD analysis arranged using the *labeled ordering* scheme. Here, we observe five stages of CKD, and the control, and are displaying the abundances for 5,127 species. Just as we did with the ordering in Figure 5.4, we ordered each of the species in the profiles based on mean relative abundance of each CKD stage: the most prominent taxa in CKD Stage 1 are surrounded by the labeled border of the CKD 1 area, the most prominent taxa in CKD Stage 2 are surrounded by the border of the CKD 2 area, and so on. Each image on the panel corresponds to the highest mean relative abundance in the samples from each of the CKD conditions. It is not surprising that the respective regions are more lit up when the samples are from the appropriate cohort. Thus, we can readily identify the stage of each sample by looking at the density of *hotspots* in each labeled area.

What is more significant is the way these plots show the microbes shared by different stages of the disease. For example, region marked A in the figure shows a group of microbes that appear in all stages, while the region marked B appears in almost all stages except CKD3 (absent) and Control (lowered abundance).

A significant point to note is that by fixing the orderings of the taxa, Jasper's visualizations can be used to effectively present groups of metagenomic samples that can be partitioned temporally (longitudinal studies), spatially (body sites or environmental sites), by disease types or subtypes (e.g., ulcerative colitis vs Crohn's disease), by disease stages (as in CKD), and by developmental stages (infant gut at different stages of development). Additionally, it is readily possible to create *average* microbiome maps (Figure), *aggregate* maps (Figure), and *differential* maps (Figure 6.35) showing either average, aggregate, or differential abundances, respectively. Finally, animations can help enhance the visual appeal for some of these groups of samples, as discussed in Chapter 6.6.2.

Visual Inspector Tool

Jasper also offers the *Visual Inspector*, a tool for interactively inspecting and exploring the "microbiome maps". The inspector can load an image along with a set of annotations, and users can mouse over and click on a location of the loaded "microbiome map" to get information about the microbial genome represented at that location. Details on the inspector can be found at the project's website and Section 6.7. An obvious enhancement to the visual inspector tool is the ability to zoom into specific regions of the “microbiome map”, and/or to select a region of the image for inspection and further exploration.

5.5 Summary and Conclusion

In this chapter we have shown how the Hilbert curve visualization technique can be used to compactly visualize metagenomic community abundance profiles from both mWGS and 16S rRNA gene sequencing datasets. The resulting microbiome maps display the relative abundance of microbial genomes in an interpretable manner, and can convey information about multiple latent factors of the reference genomes in the samples under study.

The Hilbert curve is used to lay out the microbes from the reference database in two different ordering schemes that can be used to draw a microbiome map image: the first, the *taxonomic ordering* relies on taxonomy information from the Ensembl Genomes database, and can be used to create images that express abundance values in the context of the taxonomic clades that the microbial genomes belong to. The second, the *labeled ordering* is dependent on a user-specified labeling of biological conditions for a cohort of samples, and can express the abundance values of the profile in the context of a user-defined biological interpretation.

CHAPTER 6

APPLICATIONS OF MICROBIOME MAPS

As discussed in Chapter 5, JASPER uses a technique called the Hilbert curve visualization (HCV) to visualize the microbial community abundance profiles of a large number of genomes (up to 44K). These profiles contain the relative abundance measurements of thousands of genomes. These genomic taxa are ordered along a space-filling curve in a 2D square using the Hilbert curve and the relative abundance values are translated into intensities of pixel values along the curve. In the resulting *Microbiome Map*, each position corresponds to a genome from the reference database, and the intensity of the position's color value represents the abundance of that genome in the sample. The impetus for the JASPER project is that traditional visualization techniques that display community abundance profiles do not take into account underlying information.

To showcase the potential of our technique, we analyzed 12 samples from the Human Microbiome Project (HMP) and created detailed abundance profiles and microbiome maps. We also analyzed a set of 18 samples from a Chronic Kidney Disease (CKD) study, and 15 samples from a Infant Gut Microbiome study.

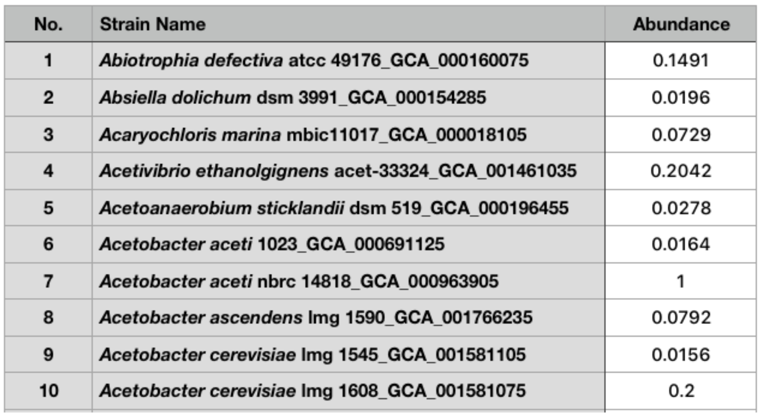
For the experiments reported here, community abundance profiles from FLINT and Kraken 2 for the HMP and CKD sample datasets were converted into microbiome maps using JASPER. The input to JASPER can represent a single metagenomic sample, or multiple ones if given in a labeled matrix, and its output will be microbiome maps for each sample provided as input.

This chapter contains detailed discussions on the results of our experiments with JASPER on visualizing these data sets.

6.1 Existing Tools for Visualizing Microbial Profiles

Visualizing a microbiome's abundance profile is achieved by mapping each read against a reference collection of microbial genomes and creating counts of the number of reads that map to each genome. A typical community abundance profile of a microbiome sample (HMP tongue dorsum, SRS019389) is shown in Figure 6.1, and it is simply stored as a text file with strain names and abundance values. We can choose to plot the single tongue dorsum profile from Figure 6.1 using a pie chart (Figure 6.2), or a bubble plot (Figure 6.3), but what we immediately notice is that these plots cannot display the abundance profiles of large genome collections such as the 44K from Ensembl. Recently, the Krona visualization software [115] was developed for interactively displaying microbial abundance profiles (Figure 6.4), but the software is also not able to display profiles with a large number of genomes; Krona (version 2.4) will not even run with the full profile for the tongue dorsum sample from Figure 6.1 which contains 44K genomes: the plot in Figure 6.4 is only able to display 20K genomes, as trying to use the software with 44K genomes crashes the program. Bar charts are another popular tool for displaying abundance profiles, and they can even be used to display multiple samples, and not just a single one. Figure 6.5 contains such a plot, and it displays fifteen bars representing fifteen time-points at which the microbiome of a single patient was sampled [51].

In all of the aforementioned visualization techniques, one can readily see the *top players*, i.e., the taxa with the highest abundance. However, all of them have poor resolution for the taxa with relatively small abundance, making visual comparisons of two samples difficult — this is exacerbated when one sample contains a different set of highly abundant taxa than other samples, as trying to only focus on the most abundant would result in a different set of taxa displayed. In particular, quickly determining the change in abundance of a single taxon is not convenient. Furthermore, none of them have the ability to visually compare clades or groups of taxa between two samples. These differences are further discussed in the sections to follow.



No.	Strain Name	Abundance
1	*Abiotrophia defectiva* atcc 49176_GCA_000160075	0.1491
2	*Absiella dolichum* dsm 3991_GCA_000154285	0.0196
3	*Acaryochloris marina* mbic11017_GCA_000018105	0.0729
4	*Acetivibrio ethanolgignens* acet-33324_GCA_001461035	0.2042
5	*Acetoanaerobium sticklandii* dsm 519_GCA_000196455	0.0278
6	*Acetobacter aceti* 1023_GCA_000691125	0.0164
7	*Acetobacter aceti* nbrc 14818_GCA_000963905	1
8	*Acetobacter ascendens* lmg 1590_GCA_001766235	0.0792
9	*Acetobacter cerevisiae* lmg 1545_GCA_001581105	0.0156
10	*Acetobacter cerevisiae* lmg 1608_GCA_001581075	0.2

Figure 6.1: Tongue Dorsum Profile. A tabular representation of a typical community abundance profile as generated by the Flint software. Only the first 10 rows out of 44K are shown.

6.2 Overview of Experiments with Microbiome Maps

6.2.1 Experiments with Orderings

In a metagenomics context, different linear orderings of the taxa on the Hilbert curve result in different microbiome maps. Users may select from one of two options: a *taxonomic ordering* which uses the linear order from Ensembl's taxonomic tree, or a user-defined *labeled ordering* which determines the order based on a labeling from a biological grouping of samples provided by the user. For both datasets (HMP and CKD) we created both a *taxonomic ordering* and a *labeled ordering*, converting each into corresponding visualizations.

Taxonomic Ordering

The taxonomic ordering scheme is based on a taxonomic hierarchy of microbial lineages. Each level of the taxonomic tree is parsed and laid out on the curve (see Figure 5.2 from Chapter 5). We take taxonomic information from Ensembl Genomes

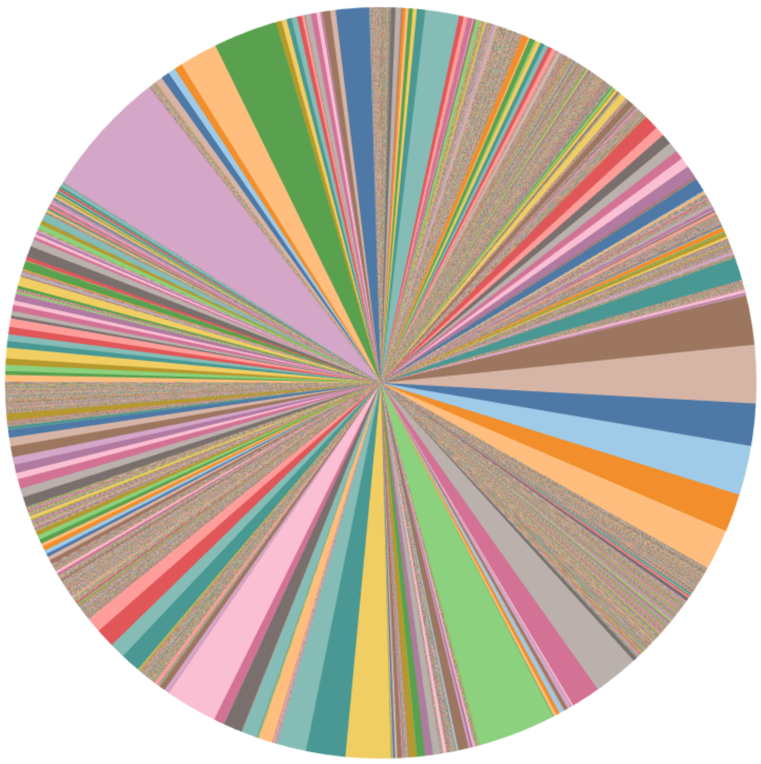


Figure 6.2: Tongue Dorsum Profile - Pie Chart. A pie chart of the abundance values for the full profile of Figure 6.1 generated by the Tableau [141] software.

[43] and use it to create the basic structure for our microbiome maps. For optimization purposes, we compute the orderings at the Genus, Species, and Strain levels. In our visualizations, one image represents one sample, and the outermost 2D square represents a union of all the different genera in our reference database.

Lines are drawn to fence off different genera, and these create the "microbial neighborhoods" mentioned in 5.3.1 and that encompass related clades of microbial species and strains in the taxonomy 6.6. Thus, a useful byproduct of the *taxonomic ordering* and its microbiome map is that the area of a "microbial neighborhood" corresponding to a specific genus reflects its size (i.e., the number of reference genomes from that genus) in the database. In Figure 6.7, we can see that the *Streptococcus* and *Staphylococcus* genera contain a lot of genomes in our database, while the *Yersinia*

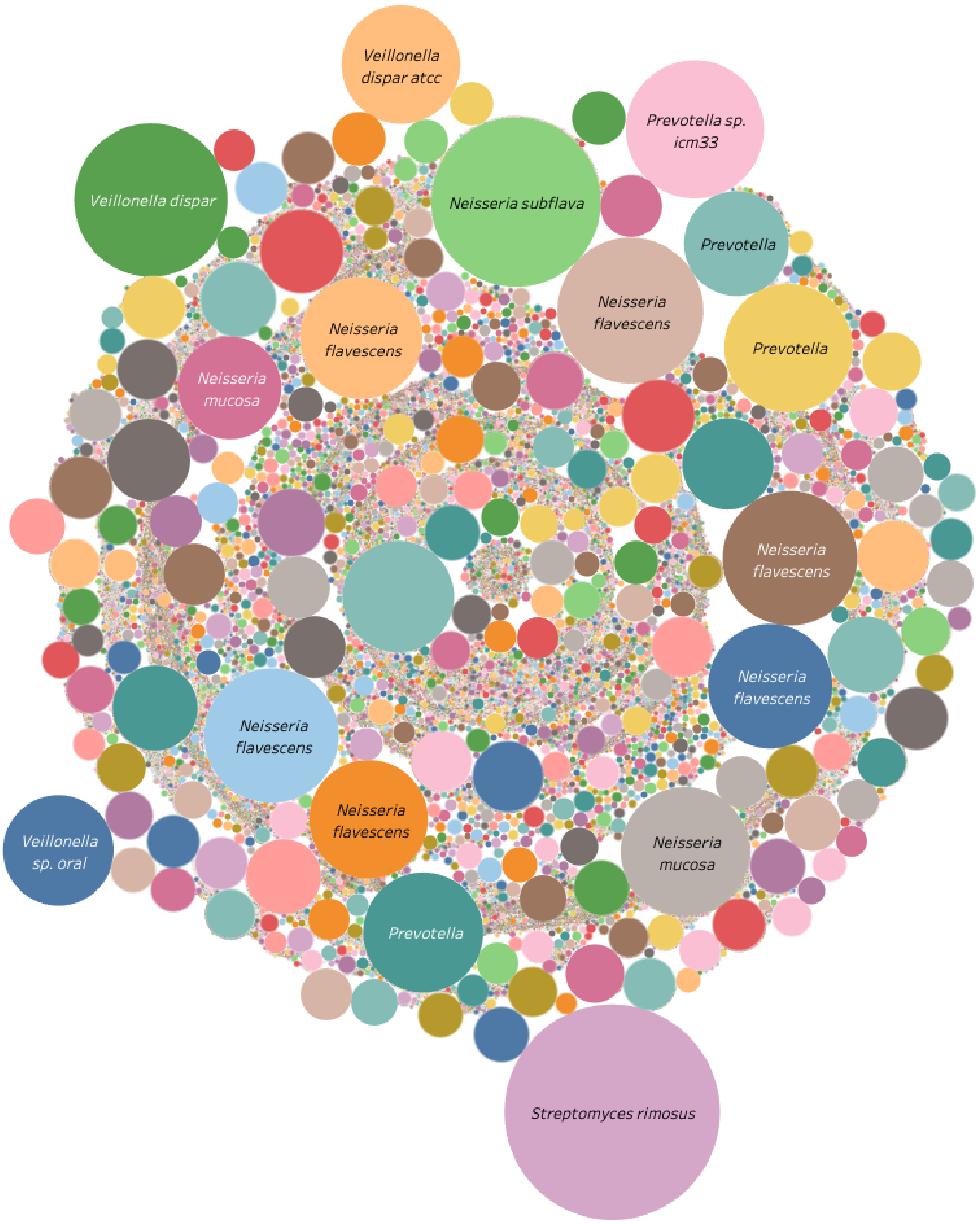


Figure 6.3: Tongue Dorsum Profile - Bubble Plot. A bubble plot of the abundance values for the tongue dorsum profile of Figure 6.1. The size of the bubble represents a relative abundance value of the profiled genome at the strain level (note that labels display the genus and species, and are truncated for display purposes). Plot generated by the Tableau [141] software.

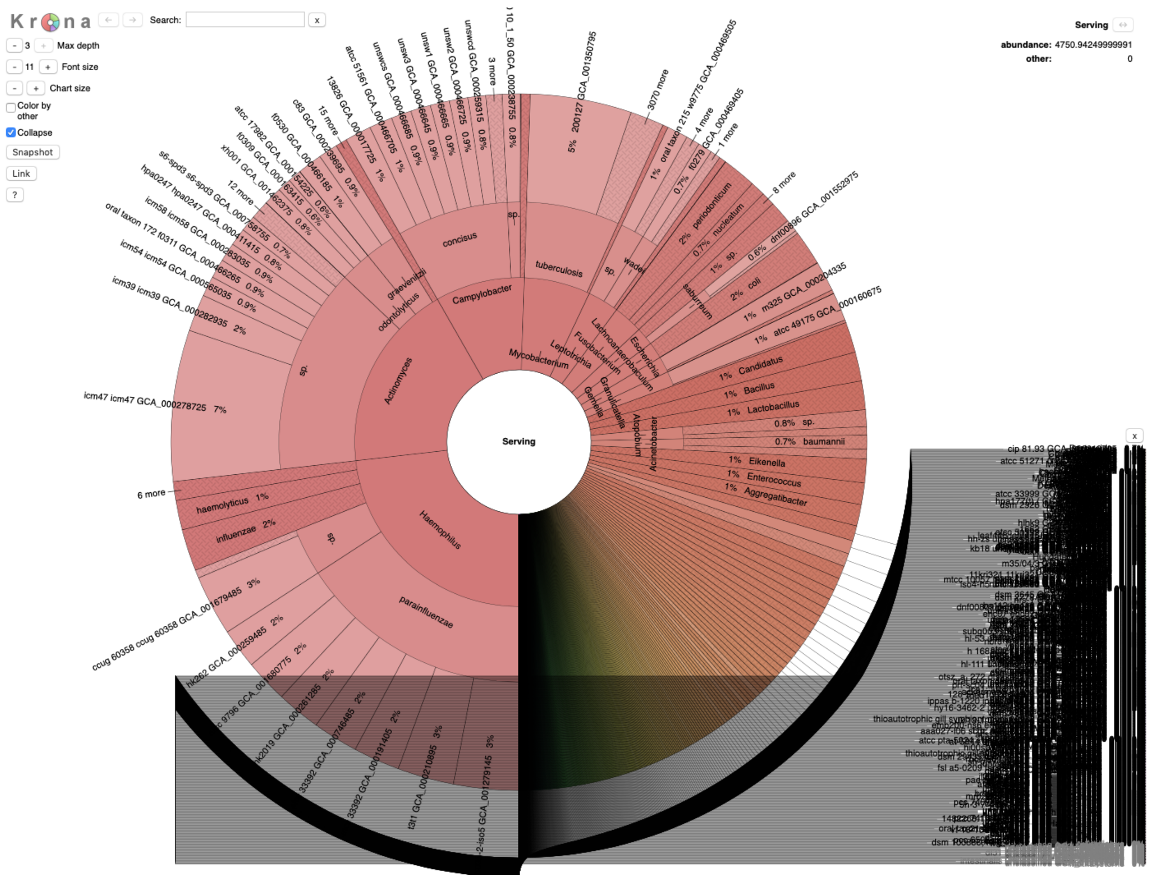


Figure 6.4: Metagenomic Profile - Krona Plot. A screenshot of an interactive Krona plot displaying the abundance profile of the tongue dorsum sample from Figure 6.1. The black bands at the bottom and bottom-right of the plot are caused by the need to pack a large number of reference genomes in a small section of the circle.

genus contains a relatively smaller set. Visualizations such as these for a reference database are helpful because they inform future sequencing projects if we wanted to achieve better parity for the references.

Labeled Ordering

The second option for ordering genomes along the Hilbert curve is the *labeled ordering*. This is an ordering scheme that determines the 1D linear order based on a user-supplied labeling of samples provided as a labeled $m \times n$ sample matrix M, where each of the m rows correspond to samples, and each of the n columns correspond to genomes in the reference database; if we have m samples, and n reference genomes, then the cells in the matrix corresponds to abundance values for a genome n_i in a sample m_j. Figure 6.8 contains the basic layout for a case of 8 body sites from HMP.

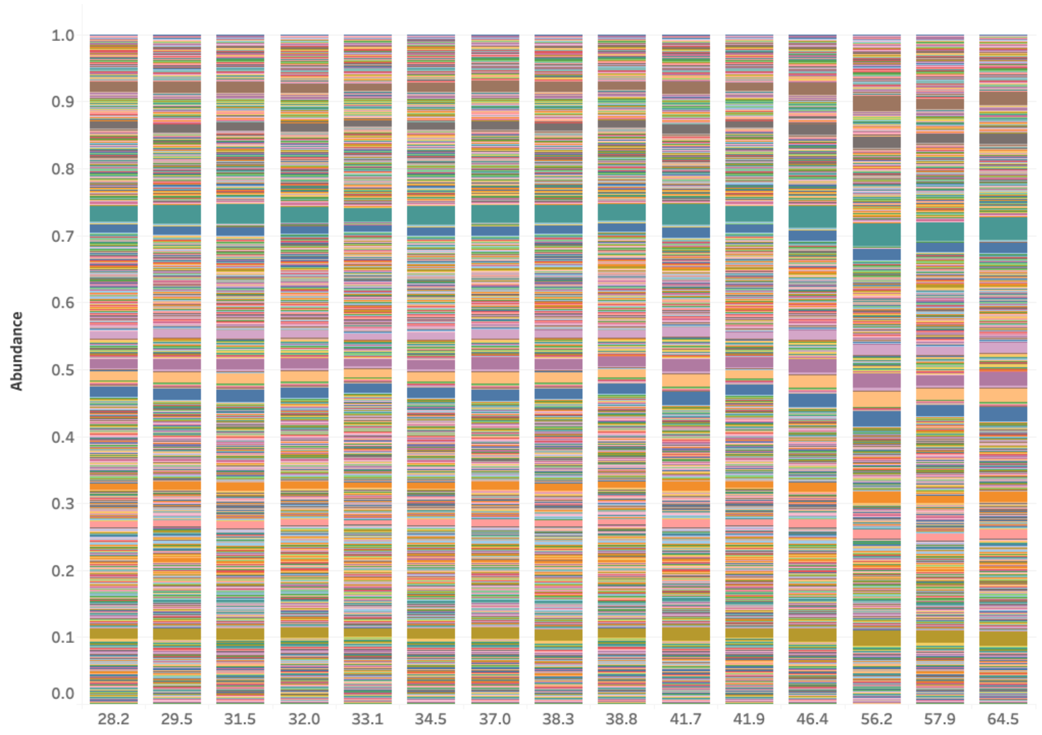


Figure 6.5: 16 Samples - Stacked Bar Chart. A stacked bar chart representation of a community abundance profile for 16 time points generated by the Tableau software [141] for the study from [51].

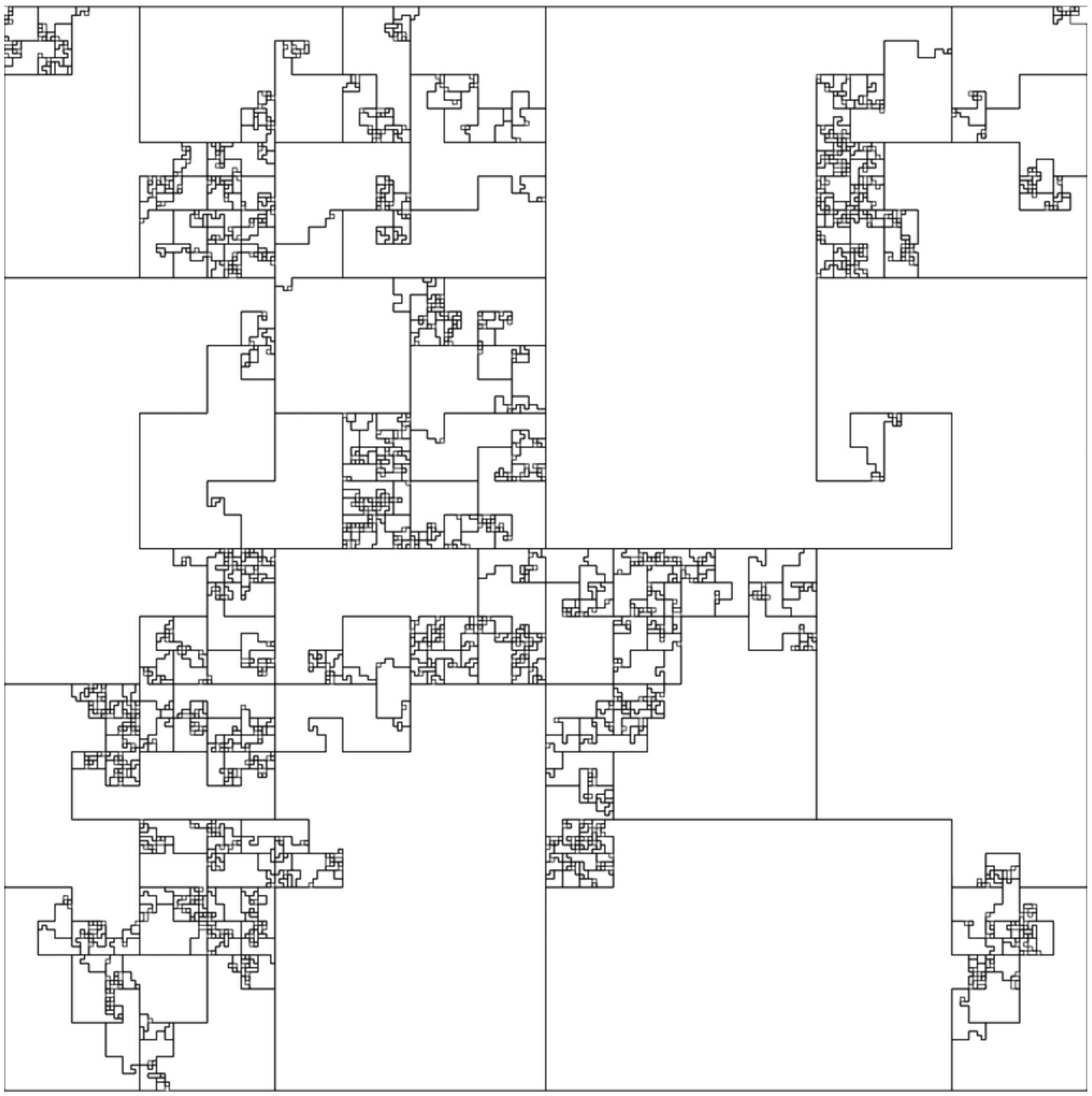


Figure 6.6: Taxonomic Ordering. Basic layout of the taxonomic ordering. Each unit of the map is assigned to one taxon in the taxonomic ordering. Each clade in the taxonomy tree appears consecutively on the taxonomic ordering and gets assigned a region in the map. As shown, these clades can be outlined, if needed, creating microbial neighborhoods of taxonomically related taxa. The size of an outlined region is proportional to the number of species in that taxonomic clade.

Note that in this ordering, each taxon is assigned a condition that it best "represents". A taxon is grouped under condition i if its average abundance is highest in the samples corresponding to condition i. Then, the conditions are ordered along

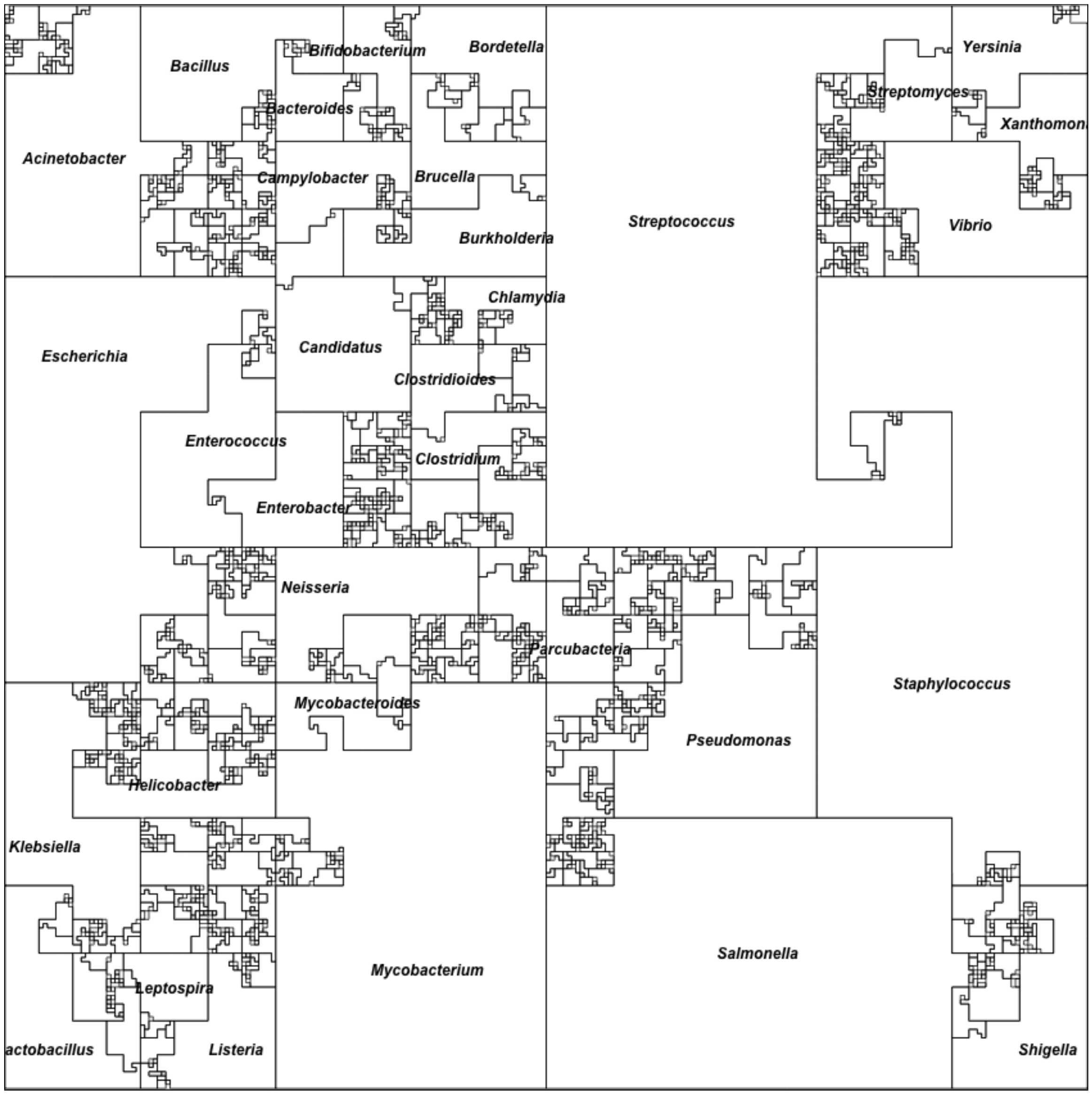


Figure 6.7: Taxonomic Ordering at the genus level. Clades at the genus level are outlined as microbial neighborhoods. The sizes of the neighborhoods correspond to the number of child species in the taxonomy tree.

the Hilbert curve. Within the condition, all taxa are ordered using the taxonomic ordering described earlier. Note that the *labeled* arrangement is still visualizing one sample, but it is effectively displaying the abundance characteristics of the biological condition to which the sample belong. In other words, if the sample belongs to condition i, then it is most likely to display "hotspots" in the region of the map cor-

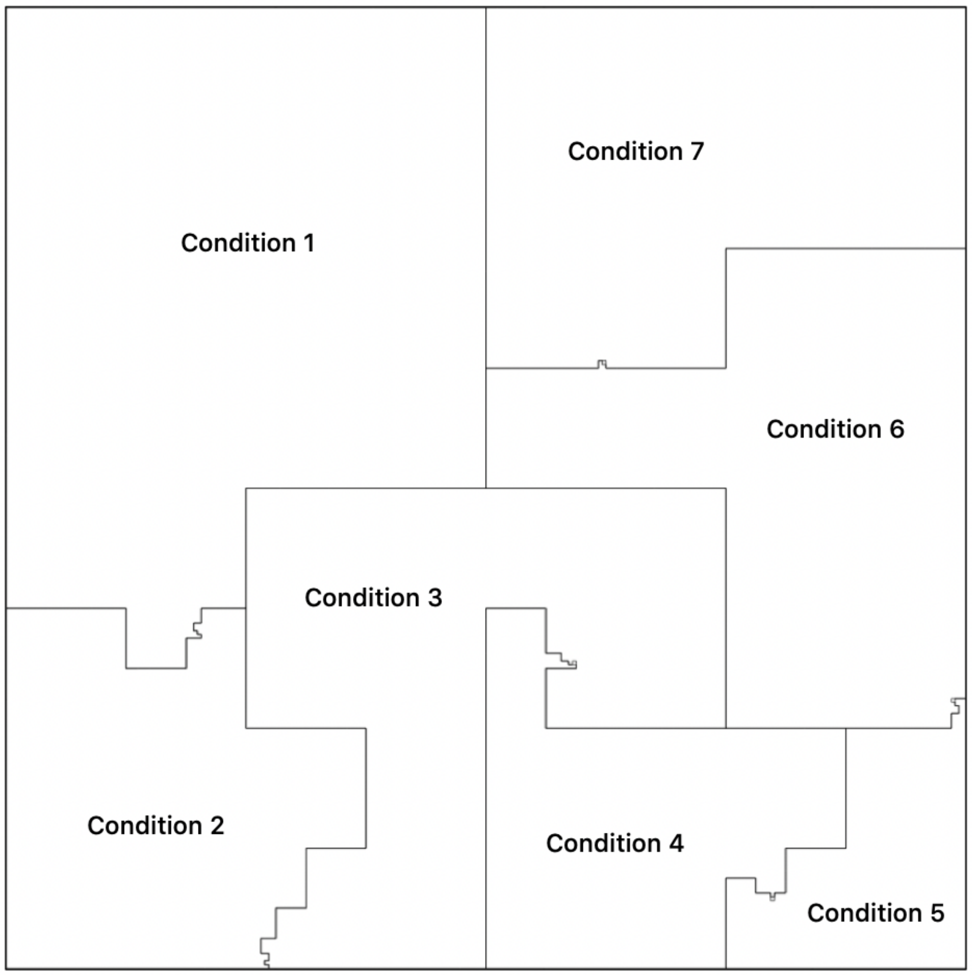


Figure 6.8: Labeled Ordering. A labeled ordering of an example profile that contains 7 biological conditions. The ordering is for 44K genomes from Ensembl, and they are ordered based on their mean relative abundance in the samples with that condition.

responding to that condition, making it easy for users to visually classify samples by their visualizations. With real samples that may be part of more than one condition or in between two conditions (e.g., stages of a disease), hotspots may also appear in other neighborhoods. However, the utility is best recognized with the examples shown. See Figures 6.18 - 6.25 for some examples.

As mentioned above, the conditions need to be ordered along the Hilbert curve. Deciding the ordering of the biological conditions, $C_1, C_2, ..., C_k$, is obvious if a natural ordering exists based on the biological description or if it is provided by the user. For example, natural orderings exist if the conditions represent different stages of disease progression, time points in a longitudinal study, or sampling locations (ordering of the organs within the digestive system, or in environmental sites). After ordering the conditions, we identify all taxa whose average relative abundance is highest in C_1 and order them first, followed by taxa whose average relative abundance is highest in C_2, and so on, breaking ties arbitrarily. Once we have established the ordering, we can then draw the *labeled ordering* for each of the samples from the input matrix M.

6.2.2 Reference Genome Collections

Ensembl Genomes

Our starting reference database of microbial genomes was downloaded from the Ensembl Bacteria [42] repository (version 45). A total of 44,048 bacterial genomes (Strain level) were downloaded in FASTA format, accounting for about 4.7 million individual FASTA references. The collection included reference sequences for fully assembled chromosomes and plasmids, as well as sequences for draft-quality super-contigs, the latter amounting for most of the reference files in our database. We followed the same preprocessing steps used in Valdes et al. for the Flint project [153], and in Section 3.2.

Kraken 2 Genomes

For processing 16S amplicon sequencing data, we used the RDP database of bacterial and archaeal 16S rRNA sequences [32]. Downloading the library with taxonomy information and indexing reference genomes were performed by Kraken 2 [166] taxonomic classification tool by utility “`kraken2-build"`” using the command in Listing 6.1.

```
THREADS=32

kraken2-build --threads ${THREADS} --download-taxonomy \
            --db /path/to/Kraken2/database
kraken2-build --threads ${THREADS} --download-library bacteria
    --no-masking \
            --db /path/to/Kraken2/database
kraken2-build --threads ${THREADS} --build --db
    /path/to/Kraken2/database
```

Listing 6.1: The command used to process data with Kraken 2.

To estimate relative abundance from Kraken 2 classification results, we used Bracken [96]. Therefore, we built the Bracken database of read-length *k*-mers derived from the Kraken 2 RDP database mentioned above using the "`bracken-build`" command in Listing 6.2.

```
THREADS=32

bracken-build -d /path/to/Kraken2/database -t ${THREADS}
```

Listing 6.2: The command used to process data with Bracken.

This completes the description of the processing of the genomes used in the profiling by Kraken 2.

RefSeq Bacterial Genomes

A total of 12,116 "complete" (i.e., their genomes have no gaps) bacterial genomes were downloaded from RefSeq v.92 [113]. We will refer to this set as the *RefSeq* collection. The indexing and annotation step was performed with the FLINT prepossessing module [153].

6.3 Processing the Microbiome Datasets

We created microbiome map visualizations for three groups of datasets: 24 mWGS samples from the Human Microbiome Project (HMP) [71] obtained from healthy individuals, 18 16S disease samples from a collaboration with Kangwon National University School of Medicine in Korea, and 15 mWGS gut microbiome samples taken from a single patient ([51]). The 24 samples from HMP represent 8 different body sites, the 18 samples from the Kangwon study represent 5 stages of Chronic Kidney Disease (CKD), along with a normal control set, and the 15 gut microbiome samples represent 15 days from an infant patient's antibiotic treatment with Vancomycin and Ticarcillin-Clavulanate (6 days before treatment, 9 days after treatment). We analyzed the mWGS HMP samples with the Flint software [153], Since Flint is not designed for profiling data from 16S rRNA sequencing data, the 16S CKD samples were analyzed with Kraken 2 [166], and the 15 gut microbiome samples were profiled directly with Bowtie2 in a single machine (no Flint cluster) to test the visualization of a reduced set of reference genomes. For the HMP samples, the abundance profiles contained relative abundance measurements for 44,048 microbial strains (details in Section 6.3.1); for the CKD samples, the abundance profiles contained relative abundance measurements for 5,127 microbial species (dtails in Section 6.3.2); and for the gut microbiome samples, the abundance profiles contained measurements for 12,116 microbial strains (details in Section 6.3.3).

6.3.1 HMP Sample Processing

Samples from the human microbiome project (Figure 6.9) were downloaded and then processed using the Flint software. Table 6.1 shows the samples and their body sites that were selected from HMP for processing.

HMP samples were processed using Flint (version RC4.B20191120) running on AWS [9]. Samples were downloaded and preprocessed, and then analyzed with

Body Site	Sample 1	Sample 2	Sample 3
Buccal Mucosa	SRS045254	SRS024557	SRS063478
Gastrointestinal Tract	SRS014683	SRS050422	SRS064276
Nares	SRS011105	SRS014901	SRS019119
Palatine Tonsils	SRS015061	SRS019126	SRS063351
Posterior Fornix	SRS014343	SRS047335	SRS078197
Supragingival Plaque	SRS017088	SRS047265	SRS065310
Throat	SRS013948	SRS015062	SRS065335
Tongue Dorsum	SRS045127	SRS013502	SRS055495

Table 6.1: HMP Samples. Eight (8) body sites were selected from the HMP (version 1 project repository) based on their representative number of samples (how many samples each body site contained), as well as the number of DNA sequencing reads that passed a HMP complexity filter (MAPQ score > 20) for each sample.

the following command-line parameters (Listing 6.3) using a "`c4.8xlarge`" EMR instance.

```
URL_FOR_SPARK_CLUSTER="yarn"
YARN_QUEUE="default"
DEPLOY_MODE="client"

spark-submit --jars ${KINESIS_LIB_PATH} \
            --master ${URL_FOR_SPARK_CLUSTER} \
            --deploy-mode ${DEPLOY_MODE} \
            --queue ${YARN_QUEUE} \
            ${PROJECT_DIR}/flint.py --samples ${CONF_SAMPLES_JSON} \
                                  --output_local \
                                  --report_all \
                                  --coalesce_output \
                                  --stream_dir
```

Listing 6.3: Flint parameters for AWS EMR.

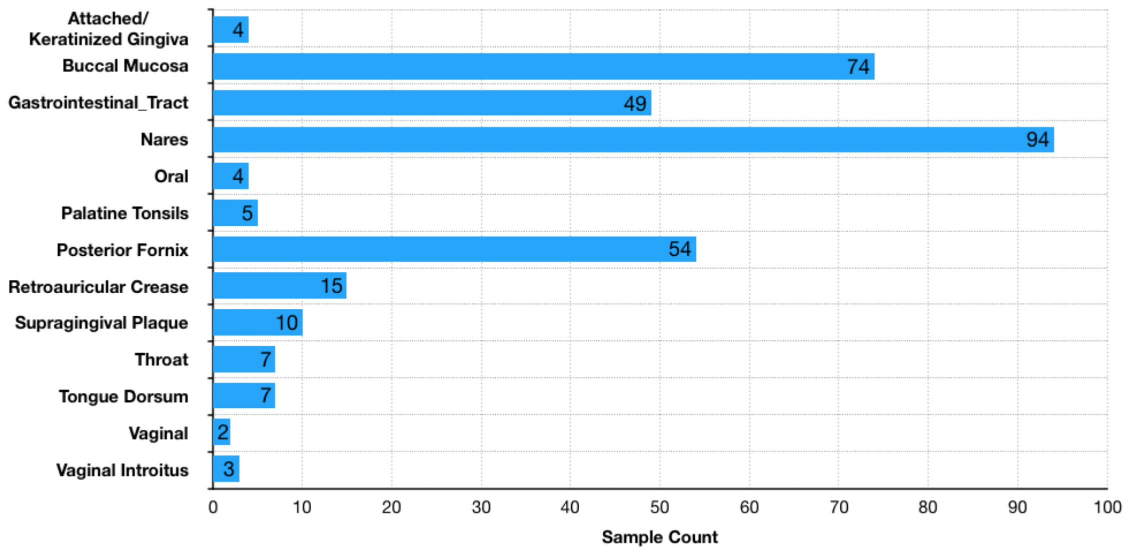


Figure 6.9: HMP Samples. Sample class distribution for the collection from the Human Microbiome Project. Four body sites dominate the distribution – *Buccal Mucosa*, *Gastrointestinal Tract*, *Nares*, and *Posterior Fornix*.

The JSON configuration from Listing 6.4 was used to run the FLINT software in an EMR cluster in AWS. Note the use of Ensembl version 45, and the use of 32 threads for the read-alingment step using Bowtie2.

```
{
   "partition_size": "64",
   "bowtie2_path": "/home/hadoop/apps/bowtie2-2.3.4.1-linux-x86_64",
   "bowtie2_index_path": "/mnt/bio_data/index",
   "bowtie2_index_name": "ensembl_v45",
   "bowtie2_threads": "32",
   "annotations": {
      "bucket": "your_bucket_name",
      "path": "path/to/ensembl/annotations.txt" },
   "streaming_app_name": "HMP Analysis",
   "batch_duration": "0.25",
   "output_dir": "/home/hadoop/flint/output",
   "samples_bucket": "hmp-samples",
   "samples":[
      {  "id": "SRS0XXXXX",
         "sample_format": "tab5",
         "sample_type": "paired",
         "sample_dir": "batch-analysis/batch-01",
         "shard_dir": "shards"
      }
   ]
}
```

Listing 6.4: JSON configuration for Flint.

6.3.2 Chronic Kidney Disease Sample Processing

18 samples were obtained from a study from Kangwon National University School of Medicine for Chronic Kidney Disease (CKD). The samples represented samples from healthy subjects along with samples from patients at one of five stages of CKD, where disease stage is defined by Kidney Disease Improving Global Outcomes (KDIGO) [78].

The taxonomic profiles were obtained using the Kraken 2 taxonomic classifier, followed by a Bracken abundance estimator with the following commands:

```
kraken2 --db /path/to/Kraken2/database \
      --report-zero-counts \
      --threads ${THREADS} \
      --report /path/to/Kraken2/report \
      --paired /path/to/fastq/pair1 /path/to/fastq/pair2 >
         /path/to/output/Kraken-report

bracken -d /path/to/Kraken2/database \
      -i /path/to/Kraken2/report \
      -o /path/to/output/Braken-report
```

Listing 6.5: Kraken and Bracken run parameters for the CKD dataset.

6.3.3 Infant Gut Sample Processing

Premature Infant Gut mWGS samples were obtained from a study (BioProject ID PRJNA301903) by Gibson et al. [51]. For our analysis, we have selected the patient with anonymized ID "107.2" who was sampled at 15 time points, or Days of Life (DOL), as listed in Table 6.2.

Sample ID	Day of Life (DOL)
SRX1551579	28.2
SRX1551431	29.5
SRX1551337	31.5
SRX1551595	32
SRX1551531	33.1
SRX1551823	34.5
SRX1551847	37
SRX1552075	38.3
SRX1551997	38.8
SRX1551533	41.7
SRX1551367	41.9
SRX1551561	46.4
SRX1551739	56.2
SRX1552021	57.9
SRX1552055	64.5

Table 6.2: Infant Gut Samples. Fifteen days of a patient's antibiotic treatment (Vancomycin and Ticarcillin-Clavulanate). Day 28.2 through day 34.5 are before treatment, while day 37 through 64.5 are after treatment.

To obtain the abundance profile for each sample, we mapped raw metagenomic reads against 12,116 RefSeq bacterial reference genomes with Bowtie2 [84]. Then, using the alignment file, we calculated the average coverage of each genome using the following formula:

$$C = \frac{NL}{G}, \tag{6.1}$$

where C is the average coverage, N is the total number of reads that align to the given sequence, L is the length of the metagenomic read, G is the genome sequence length. Finally, we obtained a relative abundance by normalizing average coverage.

6.4 Visualizations with Microbiome Maps

This section discusses microbiome maps for the HMP samples mentioned in Section 6.3.1. Two sets of images were created using the *taxonomic ordering* and *labeled ordering*. Full resolution images are available for download via a github link from the project's website, `biorg.cs.fiu.edu/jasper/`. The goal here is to use a small sample of visualizations to highlight the strengths and weaknesses of Jasper.

6.4.1 HMP Taxonomic Ordering

Figures 6.10 - 6.17 contain 3 sets of microbiome map images for each of the 8 conditions that we analyzed from HMP. Captions contain the HMP sample IDs for each image. The community abundance profiles for each of these were generated with Flint. We can see in these figures one of the inherent advantages of using the Hilbert curve with a static taxonomic order: the location of the genomes in each of the figures does not change, and we can easily compare and contrast images quickly by just glancing at them. The color intensity of each pixel region corresponds to the same genome across all the images, so we can identify what genomes are present, or absent, in the images by focusing on an area. Overlay of images can readily help to identify differences between images. Neighborhoods remain fixed for all maps, so we can easily see when a neighborhood (say *Streptococus*) is missing, i.e., not colored, from a sample as in Figure 6.11.

Figure 6.10 shows that for the buccal mucosa sample, the dominant genus is *Streptococcus*, and is quite distinct form the gut microbiome, which is dominated by *Bifidobacterium* (Figure 6.11), or anterior nares, which consists overwhelmingly of *Staphylococcus* species (Figure 6.12). We can see that for the posterior fornix sample (Figure 6.14) the dominant genus is *Lactobacillus*, but the *Streptococcus* and *Eneterococcus* genera also turn up, but not at the intensity of *Lactobacillus.* The supragingival plaque sample (Figure 6.15) has the *Streptococcus* neighborhood showing a strong presence, which contrasts to the throat sample (Figure 6.16) in which a lot of smaller neighborhoods predominate. Lastly, the tongue dorsum sample (Figure 6.17) has high abundance across the entire *Neisseria* and *Streptococus* neighborhoods, and interestingly, both neighborhoods contain hotspots corresponding to species and strains from these genera whose abundance is high (relative to the rest of that genus).

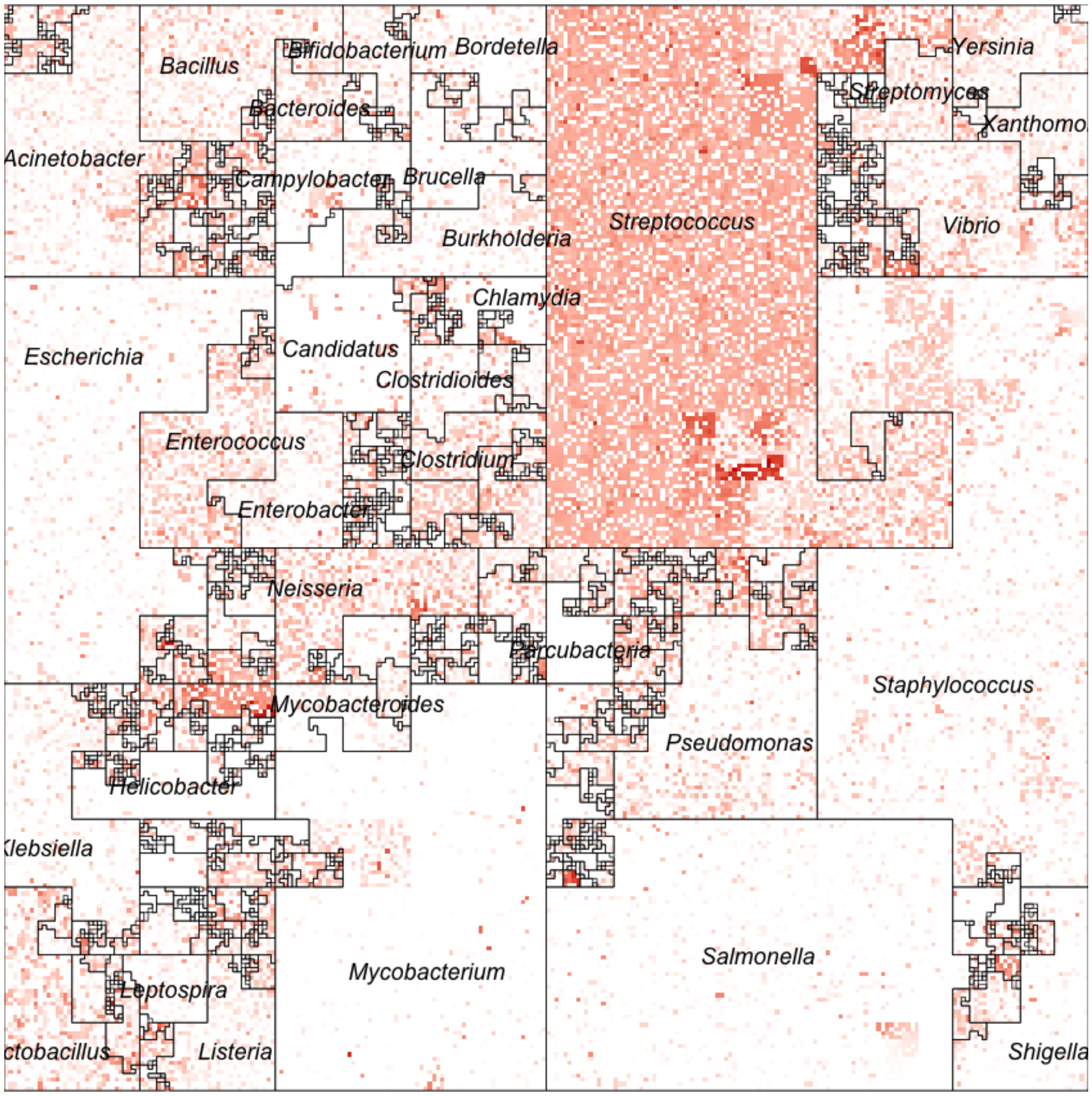


Figure 6.10: Sample SRS024557 from the HMP Buccal Mucosa set.

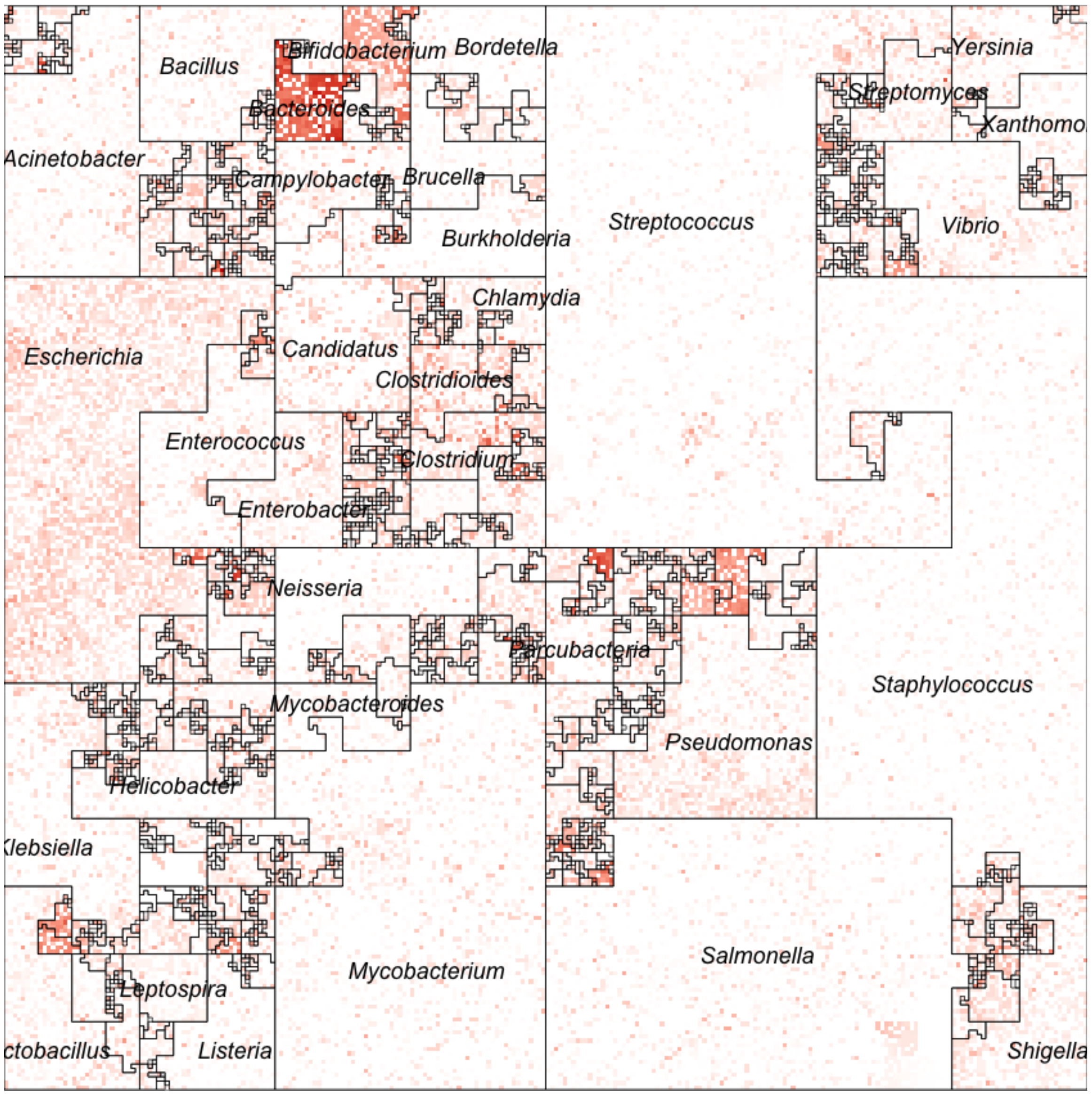


Figure 6.11: Sample SRS014683 from the HMP Gastrointestinal Tract set.

6.4.2 HMP Labeled Ordering

Figures 6.18 - 6.25 contain two sets of metagenomic Hilbert images that represent the average relative abundance of each taxon (i.e., averaged over all HMP samples from Table 6.1 for that site). Each image represents the 44K genomes from Ensembl, and their ordering is based on the highest mean relative abundance for the samples

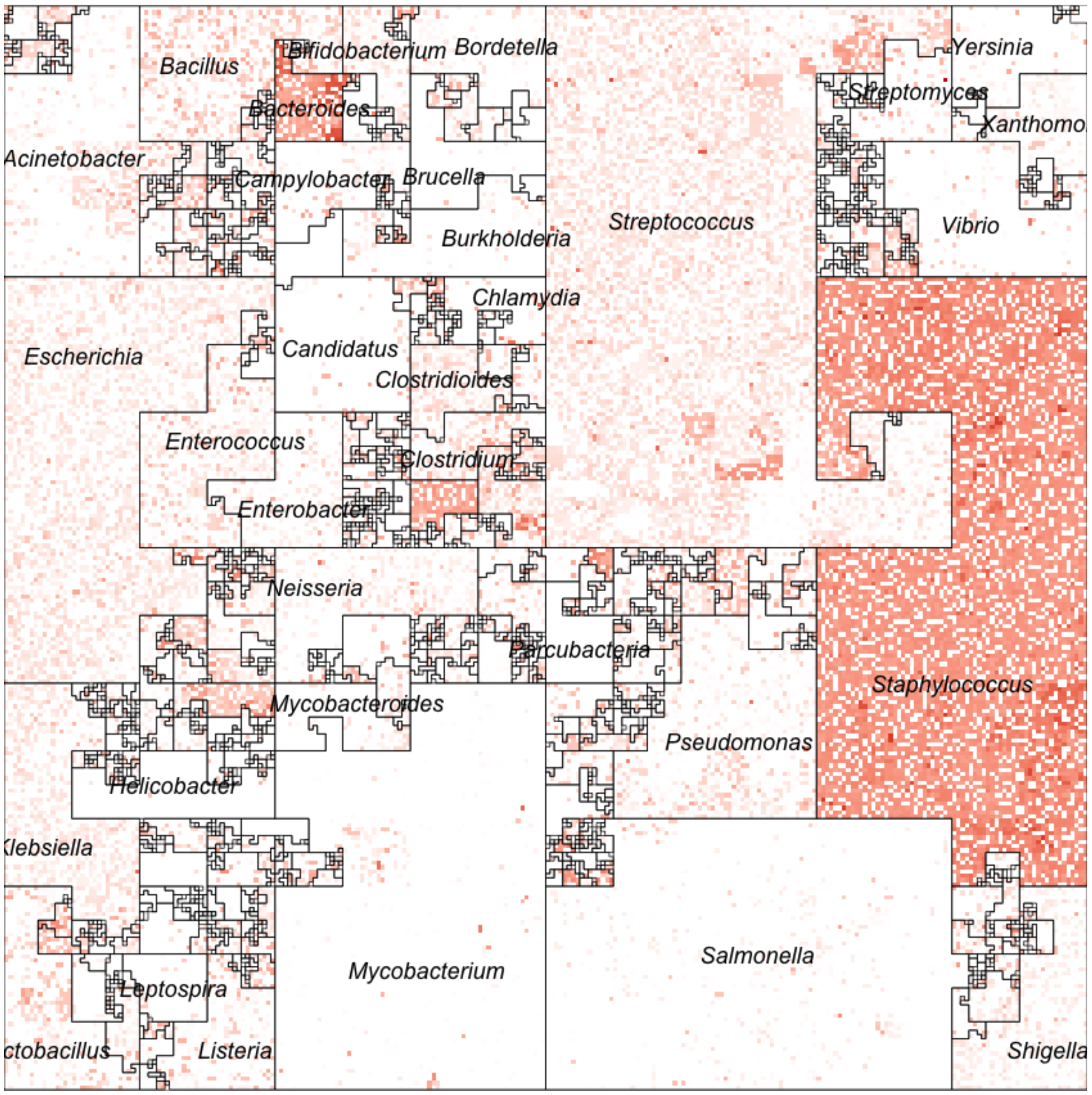


Figure 6.12: Sample SRS011105 from the HMP Nares set.

in each condition. Captions contain the HMP body site that the samples belong to. Community abundance profiles for each of these were generated with Flint.

It is interesting to see that for samples from a specific body site the hotspots are mostly confined to that labeled region on the microbiome map. For example, the hotspots in Figure 6.18 and 6.20 are largely confined to the region labeled "Buccal Mucosa" and "Nares", respectively. However, due to the proximity of the other

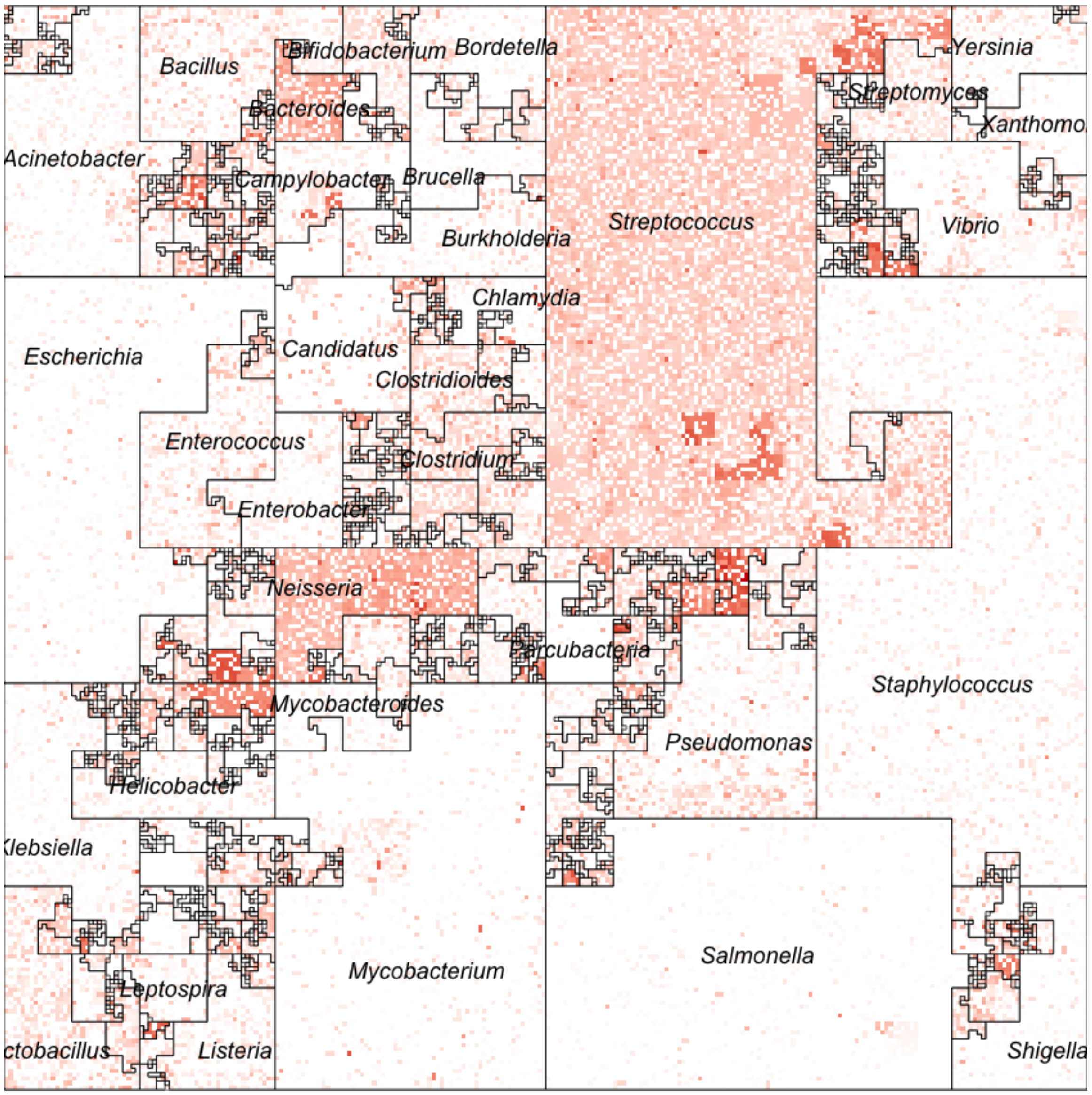


Figure 6.13: Sample SRS015061 from the HMP Palatine Tonsils set.

oral sampling sites, it does show smaller hotspots in other regions such as throat, supragingival plaque, etc. The samples from the GI tract appear to share fewer hotspots with other regions 6.19. Similar conclusions can be drawn from Figures 6.21 through 6.25.

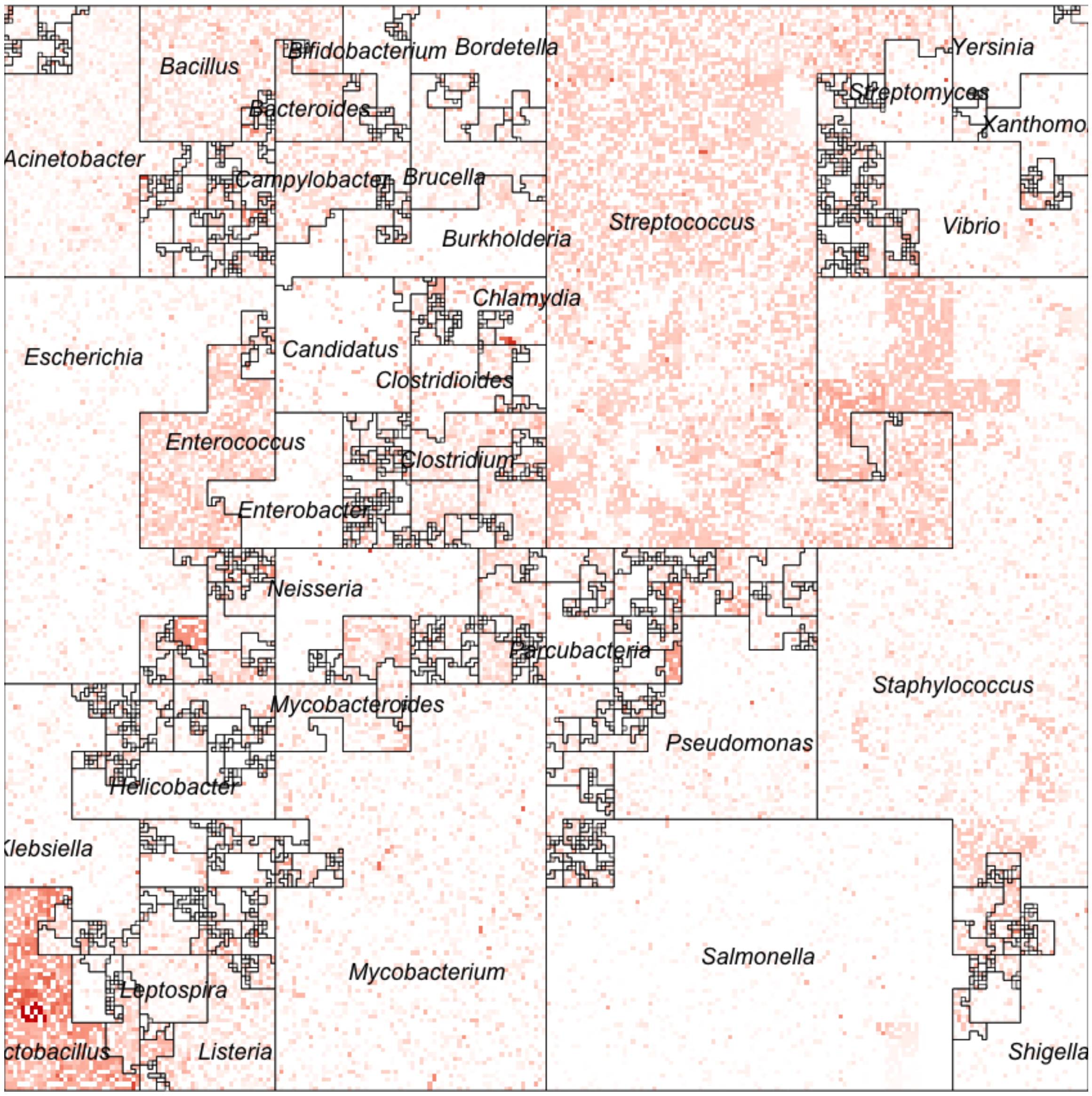


Figure 6.14: Sample SRS014343 from the HMP Posterior Fornix set.

6.5 Visualizations - CKD Study

This section contains metagenomic Hilbert curve visualizations for the CKD samples from section 6.3.2. As with the HMP samples, two sets of images were created using the *taxonomic ordering* and *labeled ordering*. Full resolution images are available for download via a github link from the project's website: `biorg.cs.fiu.edu/jasper`.

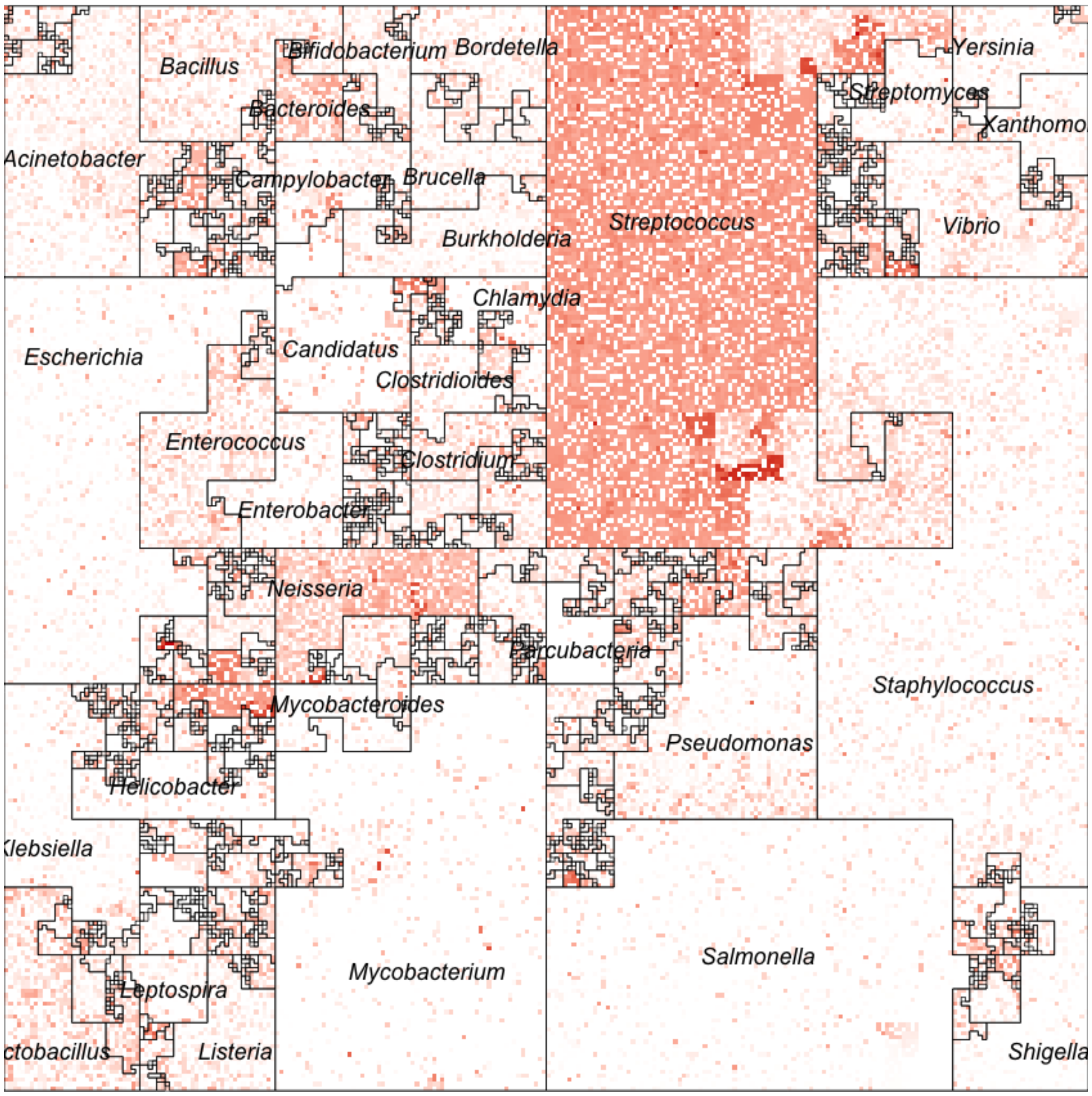


Figure 6.15: Sample SRS017088 from the HMP Supragingival Plaque set.

6.5.1 CKD Taxonomic Ordering

Figures 6.26 - 6.31 contain three sets of metagenomic Hilbert images for each of the five CKD stages, and also for the control normal. Community abundance profiles for each of these CKD images were generated with Kraken 2, and each image segment represents the relative abundance of 5,127 bacterial species.

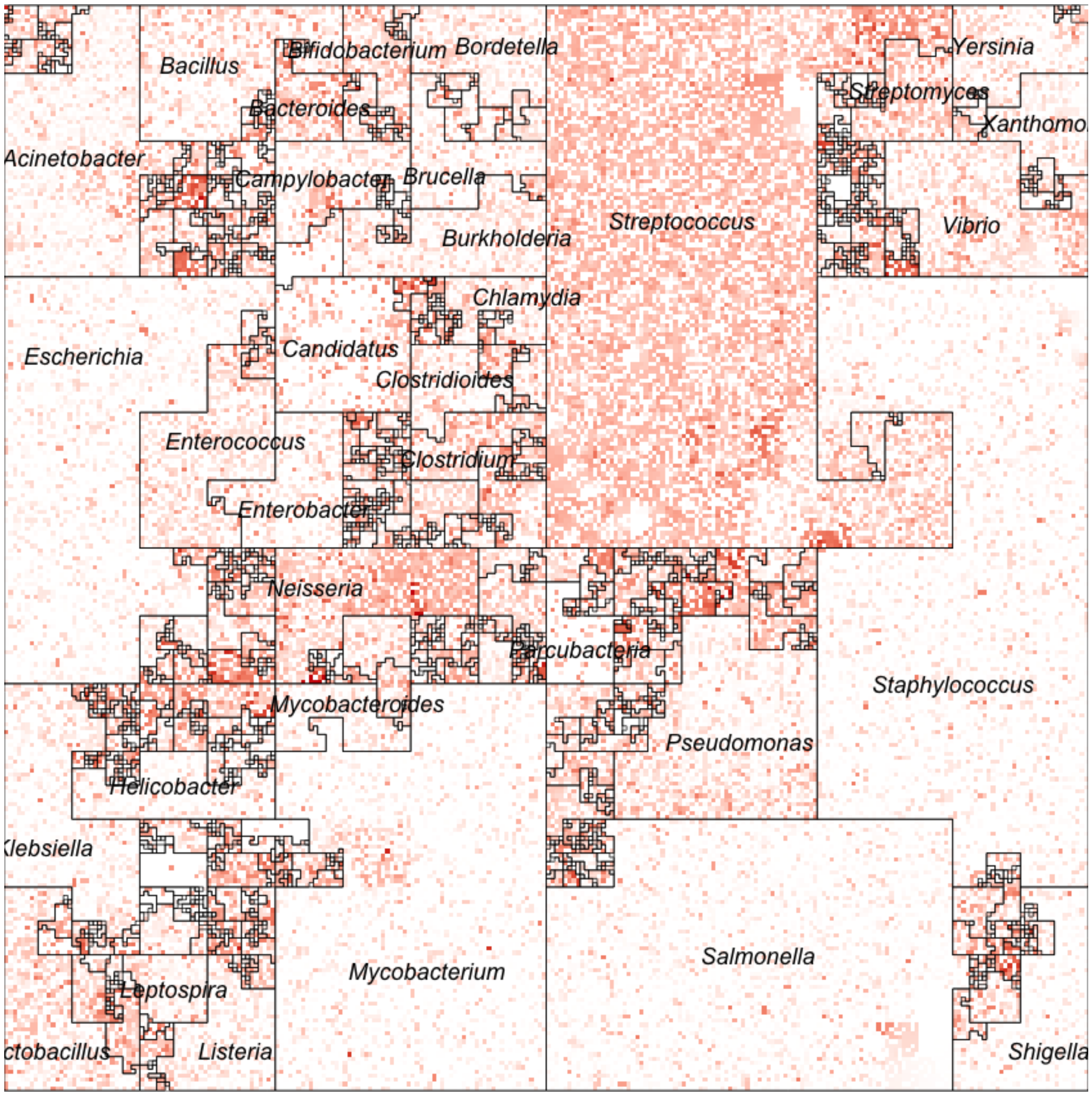


Figure 6.16: Sample SRS013948 from the HMP Throat set.

6.5.2 CKD Labeled Ordering

Figures 6.32 - 6.34 contain two sets of metagenomic Hilbert images that represent the average relative abundnace for the CKD samples. Each image represents the 5,127 genomes from the Kraken 2 16S database, and the genome's ordering is based on the highest mean relative abundance for the samples in each CKD stage. Community abundance profiles for each of these were generated with Kraken 2.

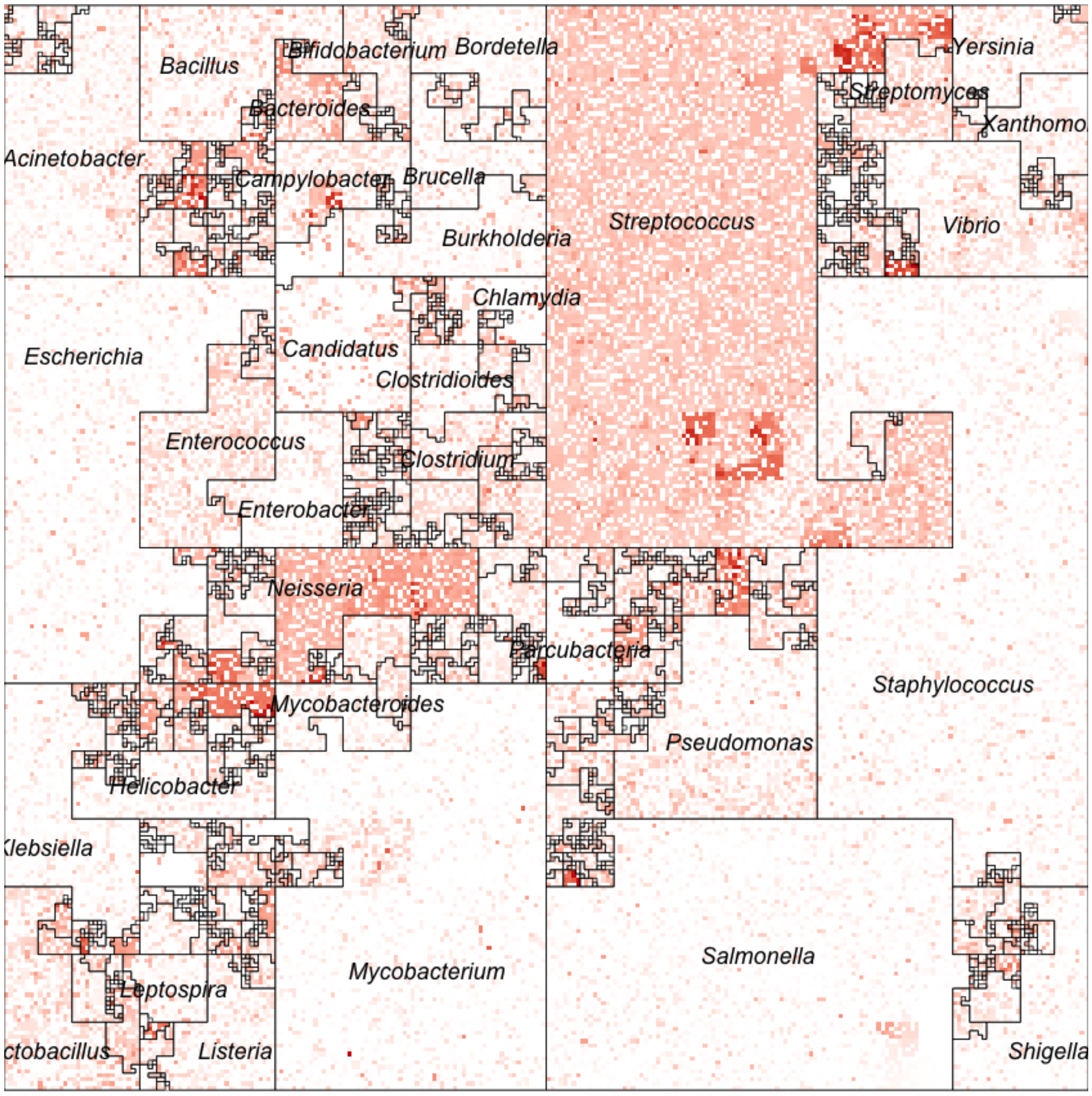


Figure 6.17: Sample SRS013502 from the HMP Tongue Dorsum.

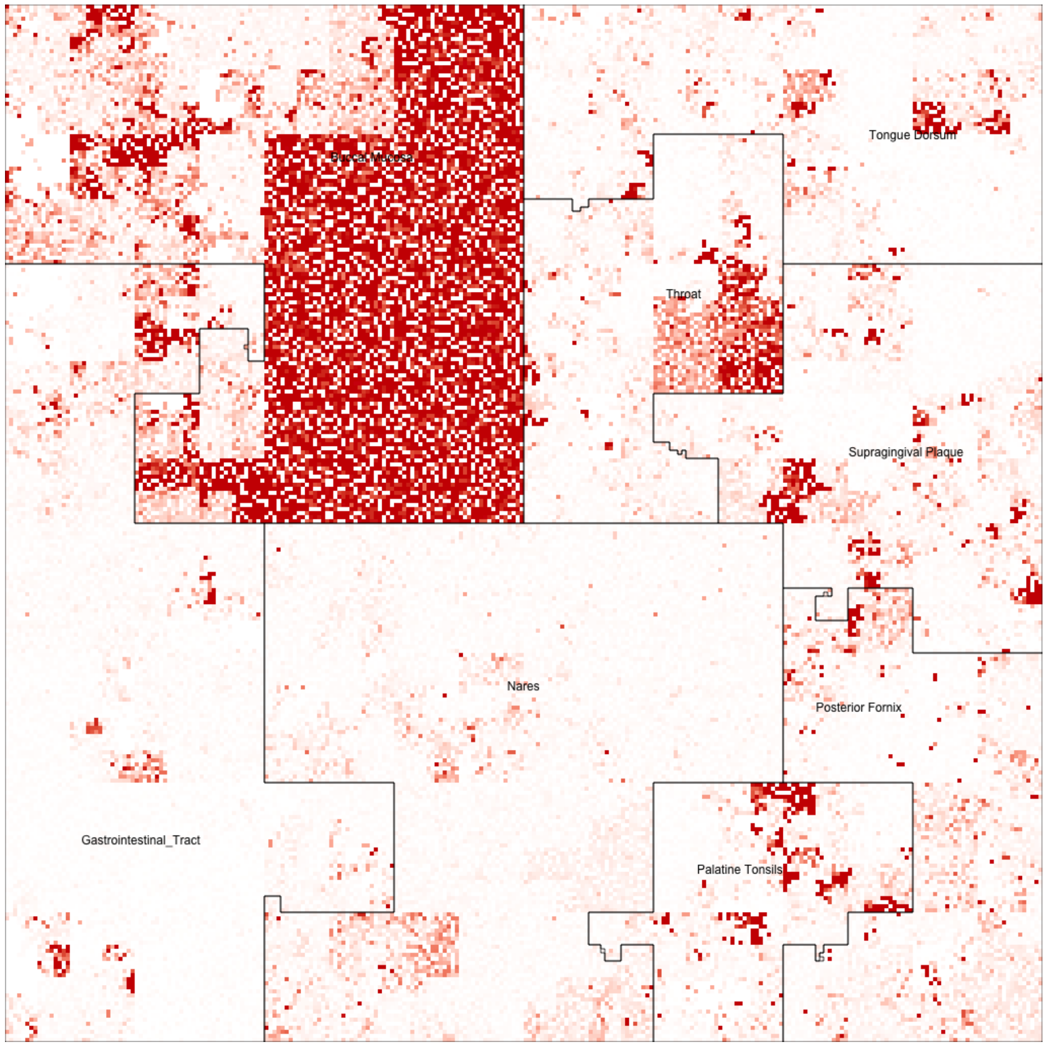


Figure 6.18: Buccal Mucosa Average Abundance

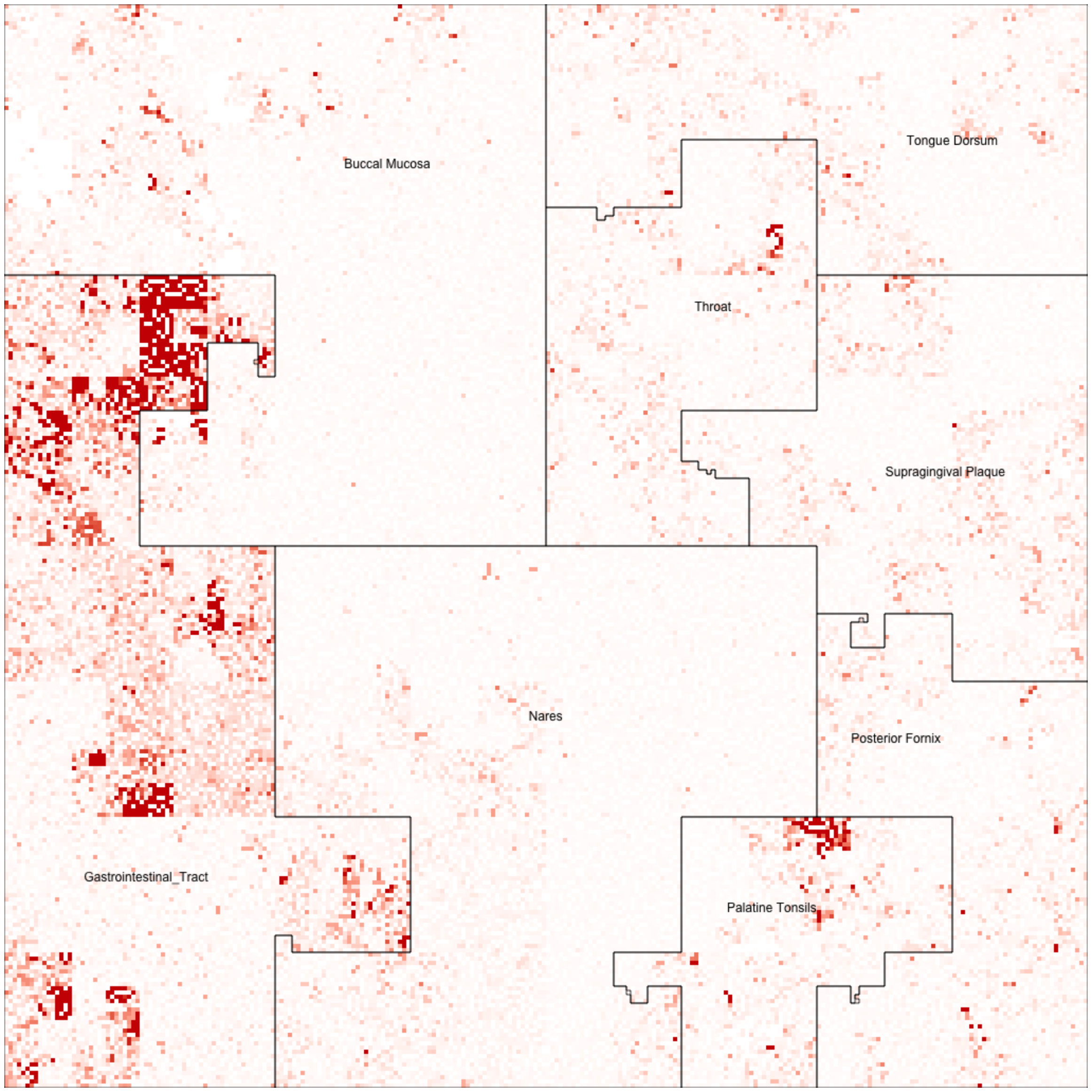


Figure 6.19: Gastrointestinal Tract Average Abundance

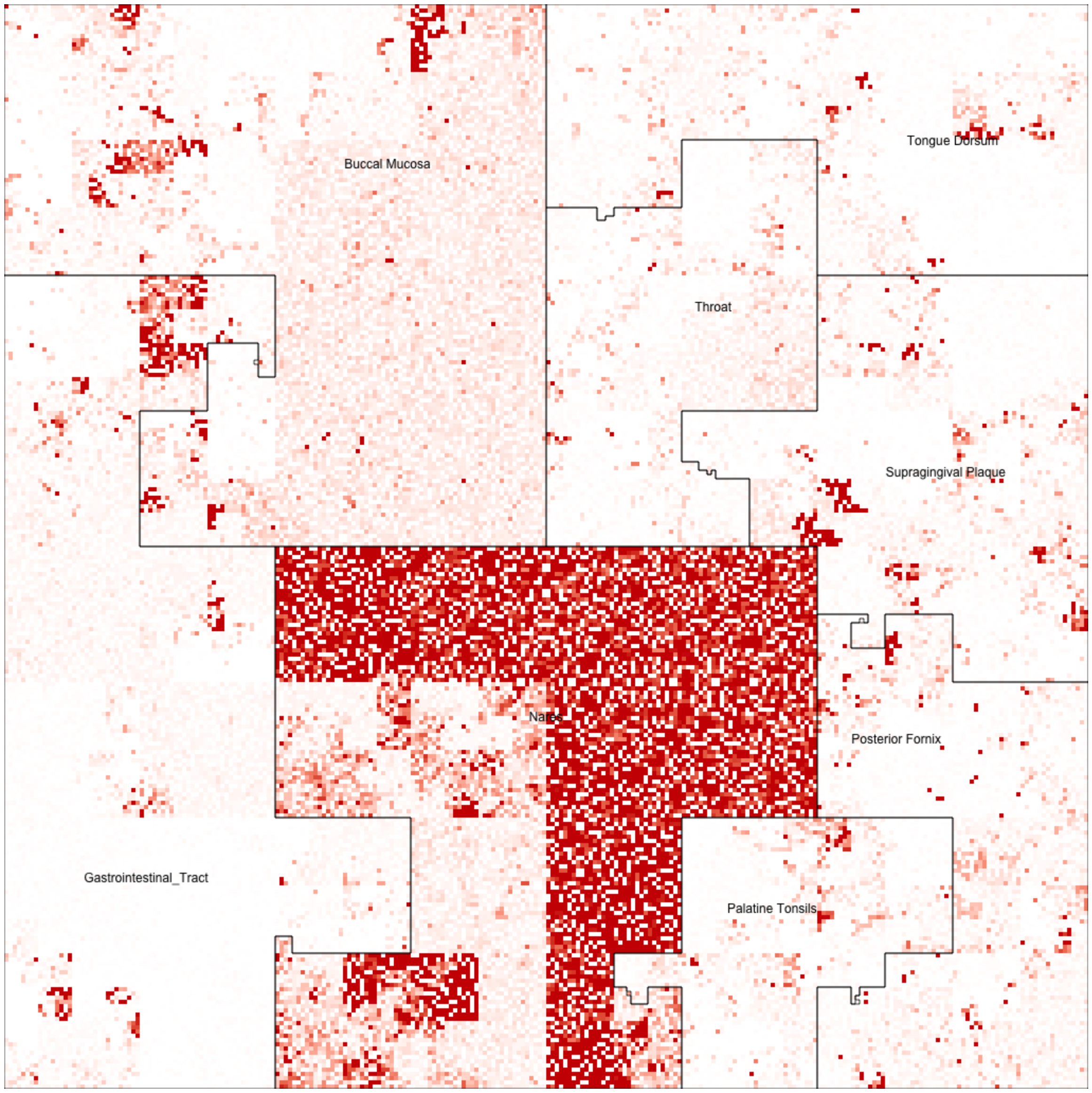


Figure 6.20: Nares Average Abundance

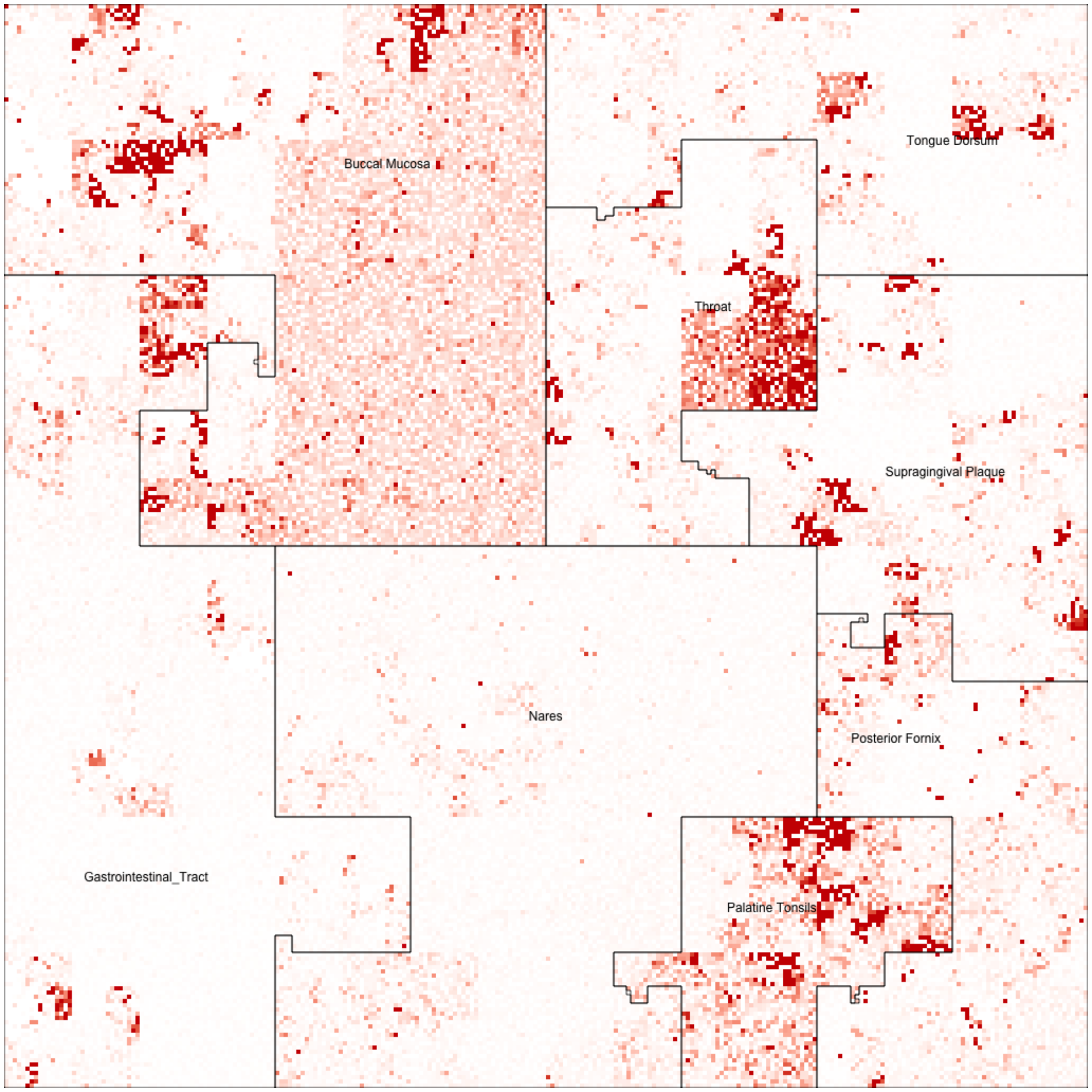


Figure 6.21: Palatine Tonsils Average Abundance

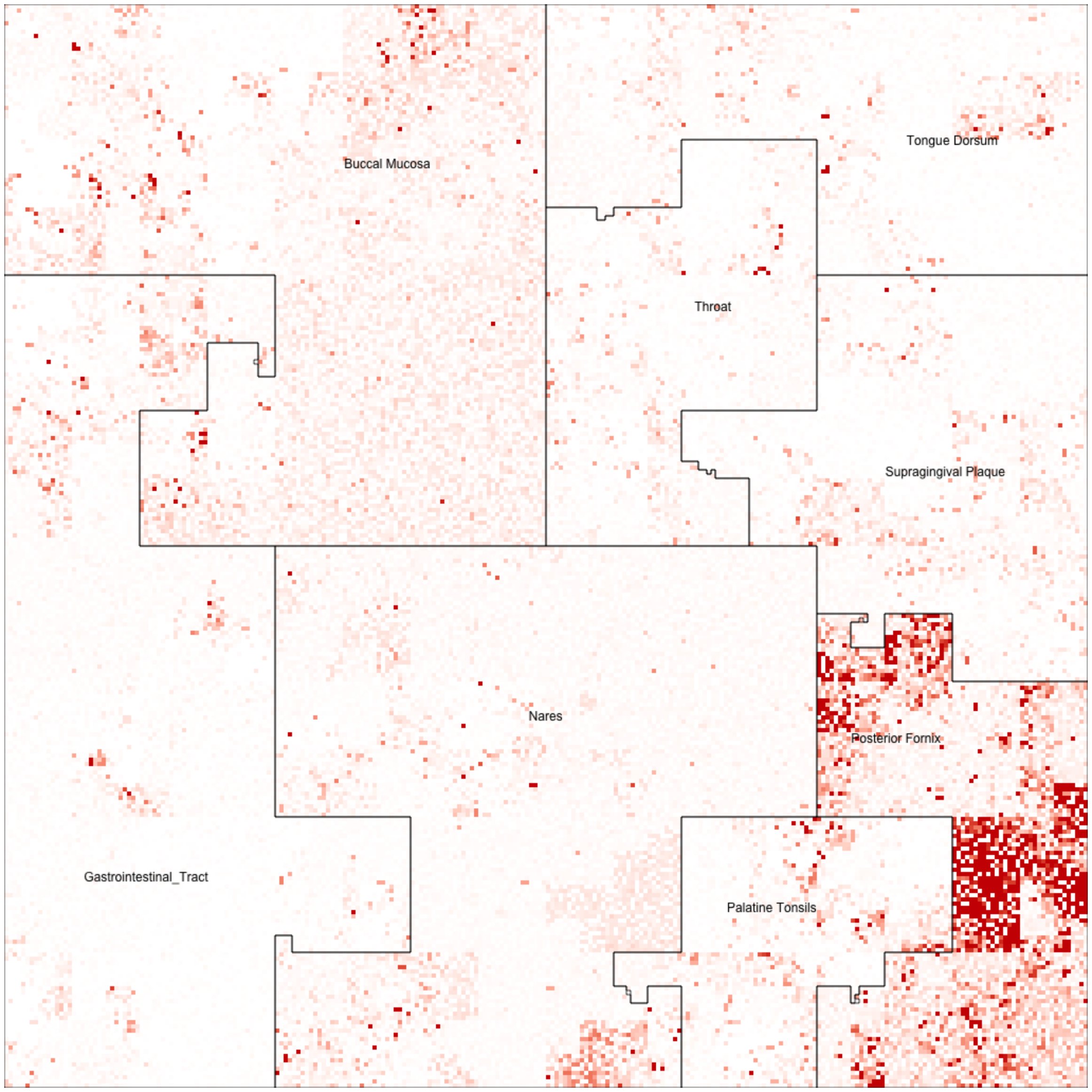


Figure 6.22: Posterior Fornix Average Abundance

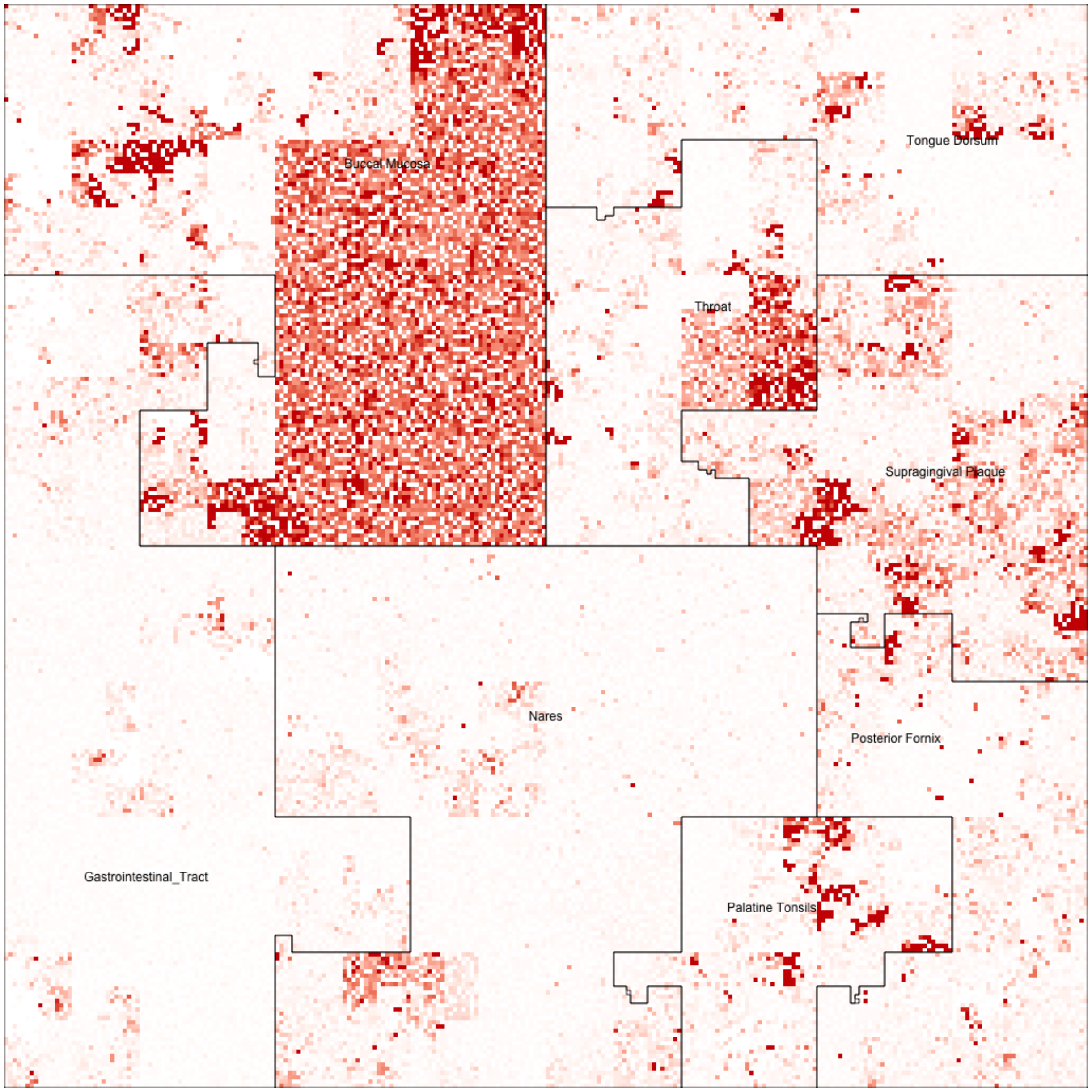


Figure 6.23: Supragingival Plaque Average Abundance

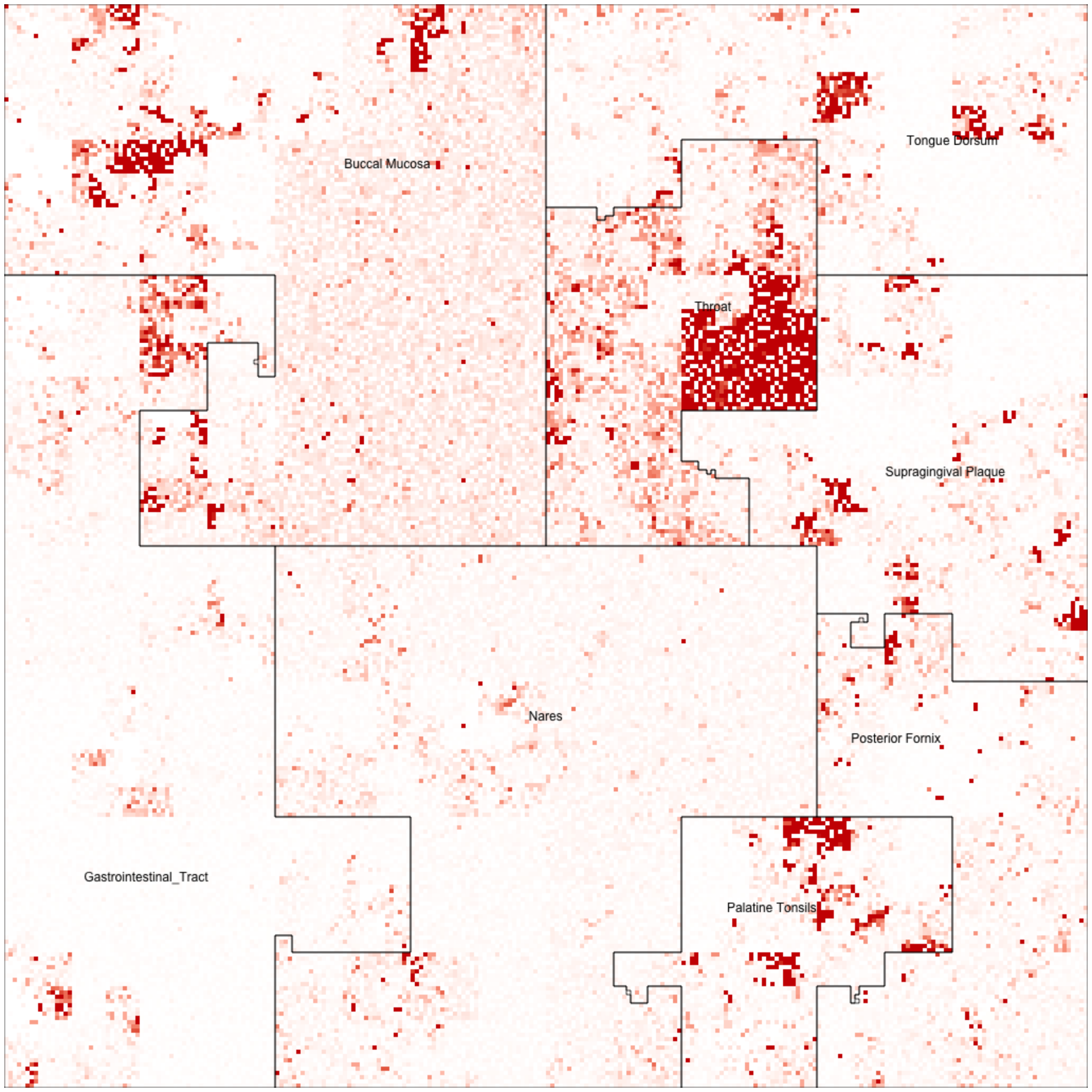


Figure 6.24: Throat Average Abundance

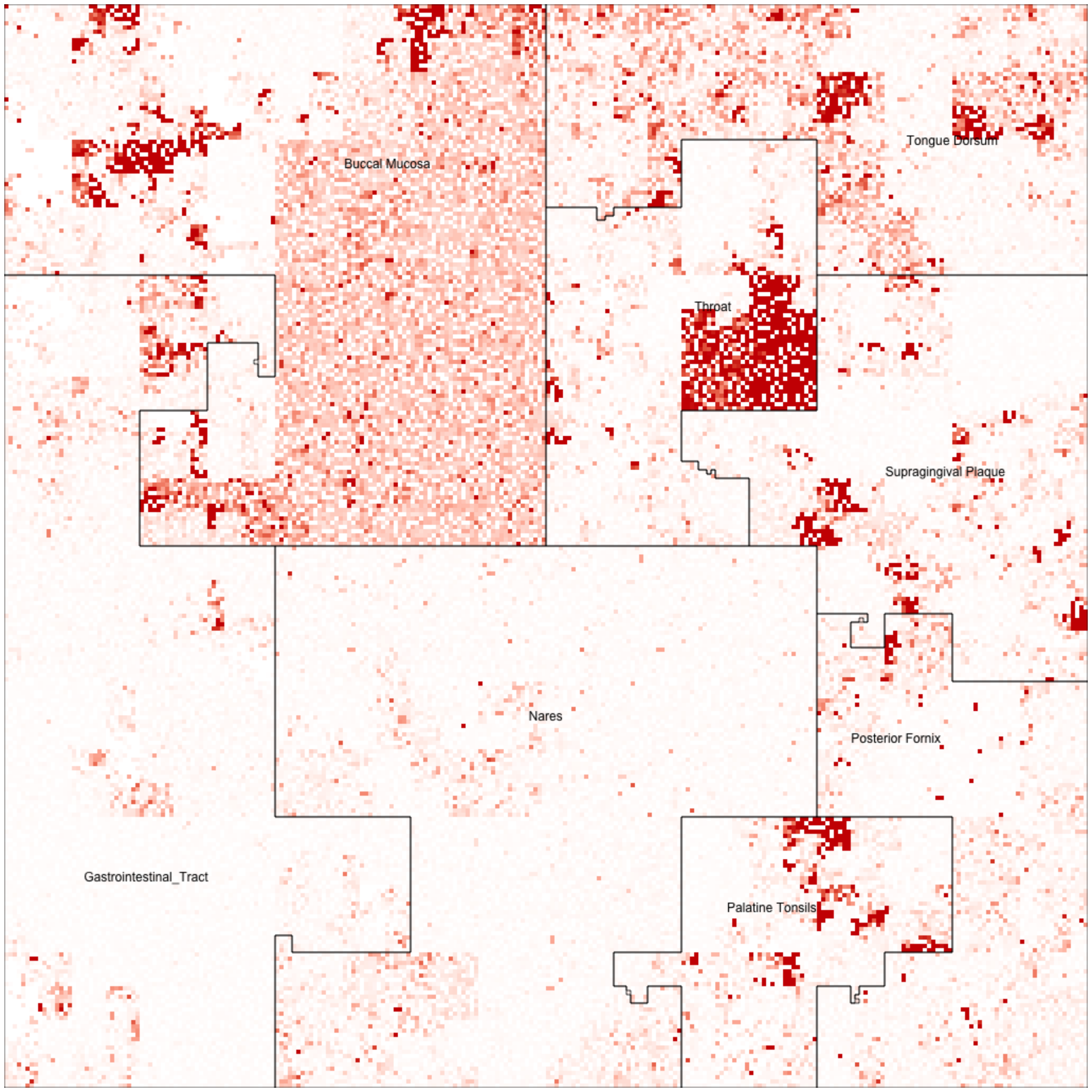
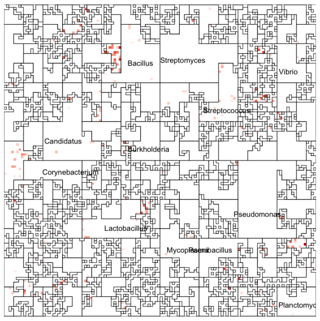
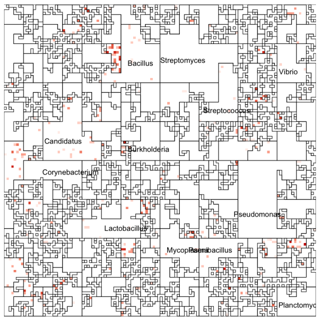


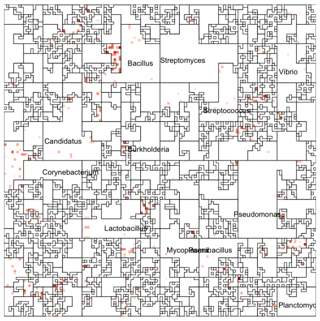
Figure 6.25: Tongue Dorsum Average Abundance



(a) Sample 1

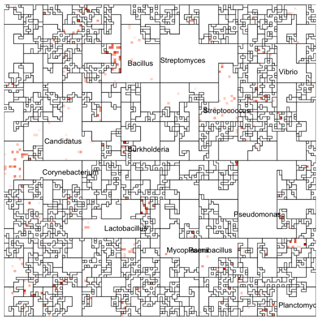


(b) Sample 2

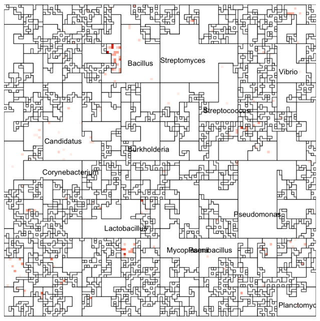


(c) Sample 3

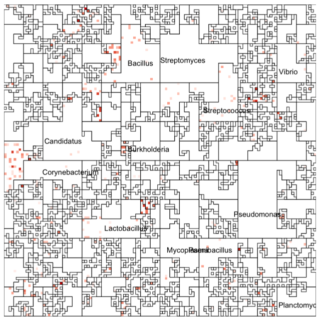
Figure 6.26: CKD Stage 1, Taxonomic Ordering



(a) Sample 4

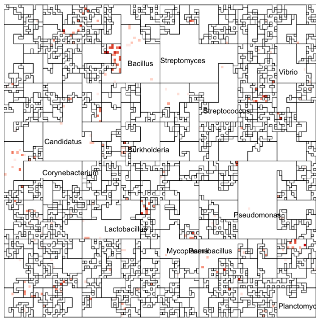


(b) Sample 5

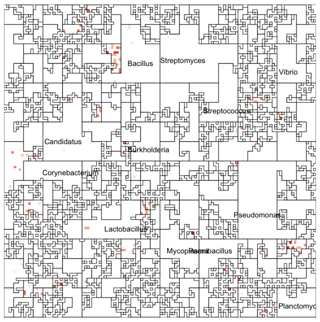


(c) Sample 6

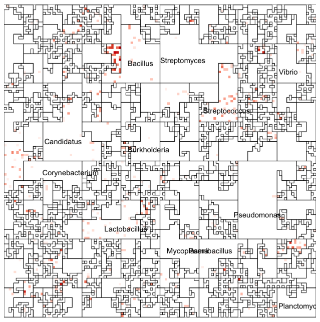
Figure 6.27: CKD Stage 2, Taxonomic Ordering



(a) Sample 7

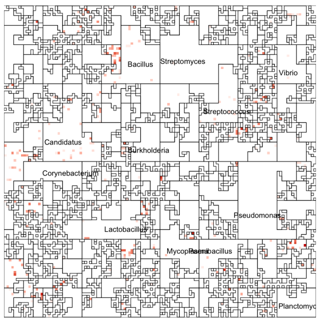


(b) Sample 8

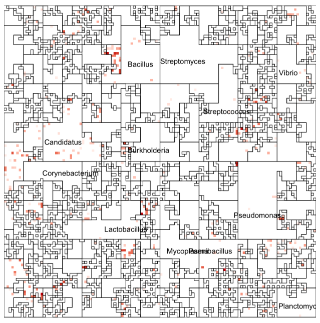


(c) Sample 9

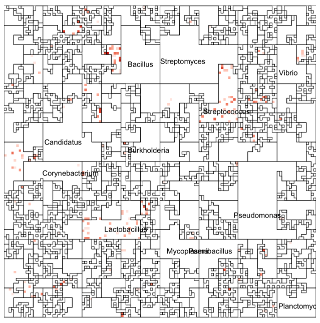
Figure 6.28: CKD Stage 3, Taxonomic Ordering



(a) Sample 10

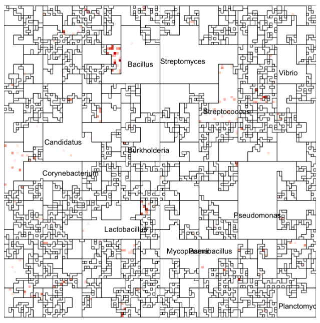


(b) Sample 11

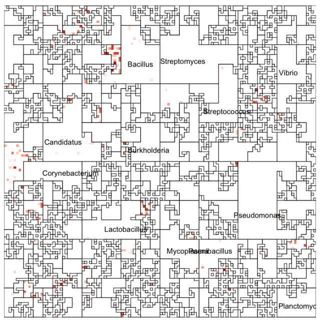


(c) Sample 12

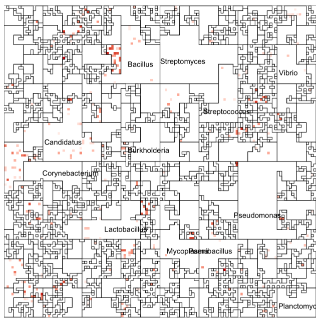
Figure 6.29: CKD Stage 4, Taxonomic Ordering



(a) Sample 13

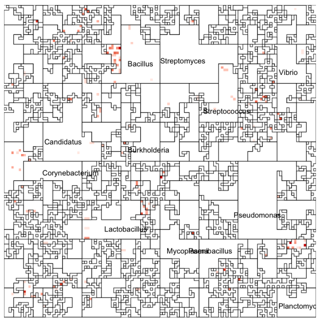


(b) Sample 14

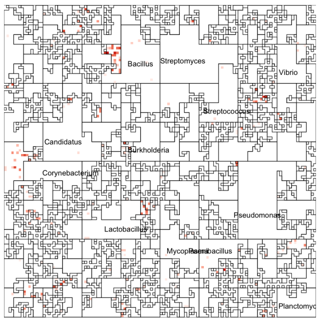


(c) Sample 15

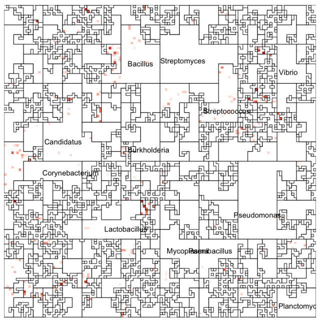
Figure 6.30: CKD Stage 5, Taxonomic Ordering



(a) Sample 16

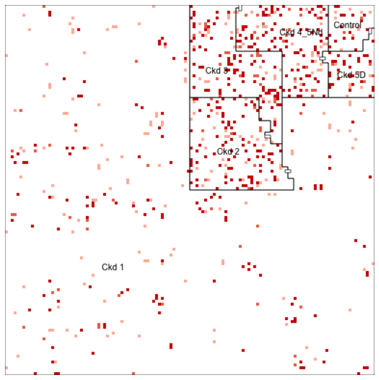


(b) Sample 17

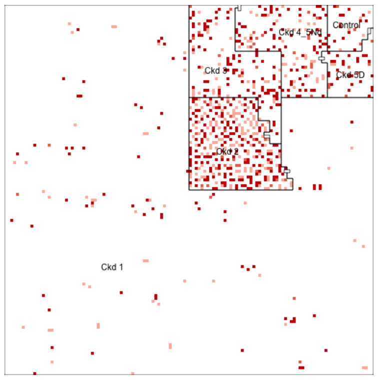


(c) Sample 18

Figure 6.31: CKD Control Normals: Taxonomic Ordering

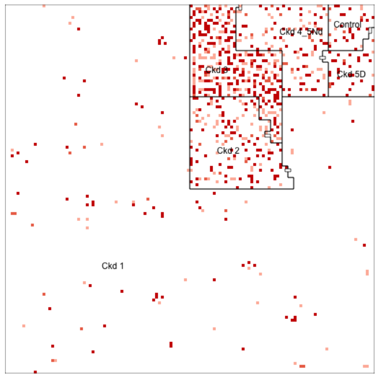


(a) CKD 1 Average Abundance

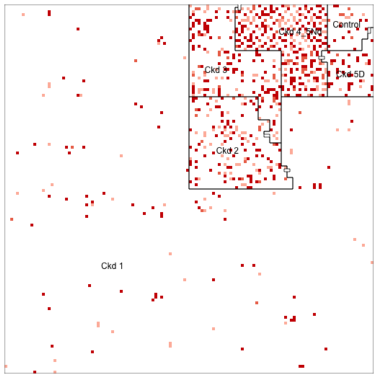


(b) CKD 2 Average Abundance

Figure 6.32: CKD Stage 1 & 2, Labeled Ordering

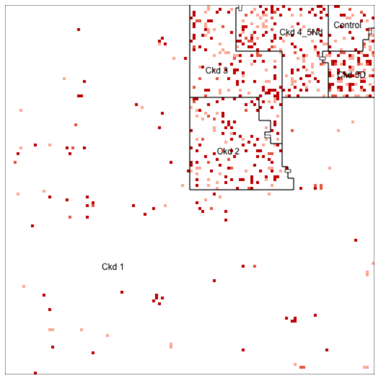


(a) CKD 3 Average Abundance

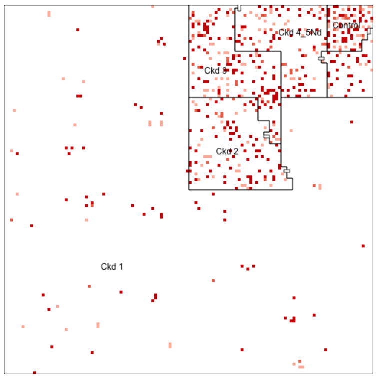


(b) CKD 4 Average Abundance

Figure 6.33: CKD Stage 3 & 4, Labeled Ordering



(a) CKD 5 Average Abundance



(b) CKD Control Average Abundance

Figure 6.34: CKD Stage 5 & Normal Control, Labeled Ordering

6.6 Variations on Microbiome Maps

6.6.1 Differential Profiles

In addition to being able to display relative abundances, a microbiome map can also be useful do display the p-values from a differential abundance analyses. Figure 6.35 contains a figure which displays the p-values from a differential analysis of the buccal mucosa and gastrointestinal tract datasets from HMP. Each image location corresponds to a genome, but the color of the location is the intensity of the p-value for the hypothesis that the relative abundance for the two conditions are different, with lower p-values having a brighter intensity, and only p-values ≤ 0.05 being colored. Note how one can visually identify taxa that are significantly different between the two body sites.

6.6.2 Animated Movies

JASPER produces a single image for each sample it is given as input, using either a *taxonomic ordering* or a *labeled ordering.* Images can be used as a single frame of animations that show abundance *hotspots*, and their evolution across samples or biological conditions. Figure 5.5 displays an example of how the microbiome "evolves" as the disease stage progresses. Figure 6.36 contains image frames from a study by [51], and shows how the microbiome of a single patient evolves over the course of several days. One can see the changes in the map due to the administration of antibiotics (Vancomycin & Ticarcillin-Clavulanate) on day 35, and the changes during the recuperation period that ensues (days 38 - 64). In particular, the figure highlights how one could focus on the genus *Enterococcus* and see how it is nearly exterminated after the antibiotic is administered, but bounces back with a vengeance, barely three days layer (see day 41), and remaining at that level for a month (see day 64). Neighboring genera seem to remain unaffected by the treatment. Full resolution images and movies are available on the project's website.

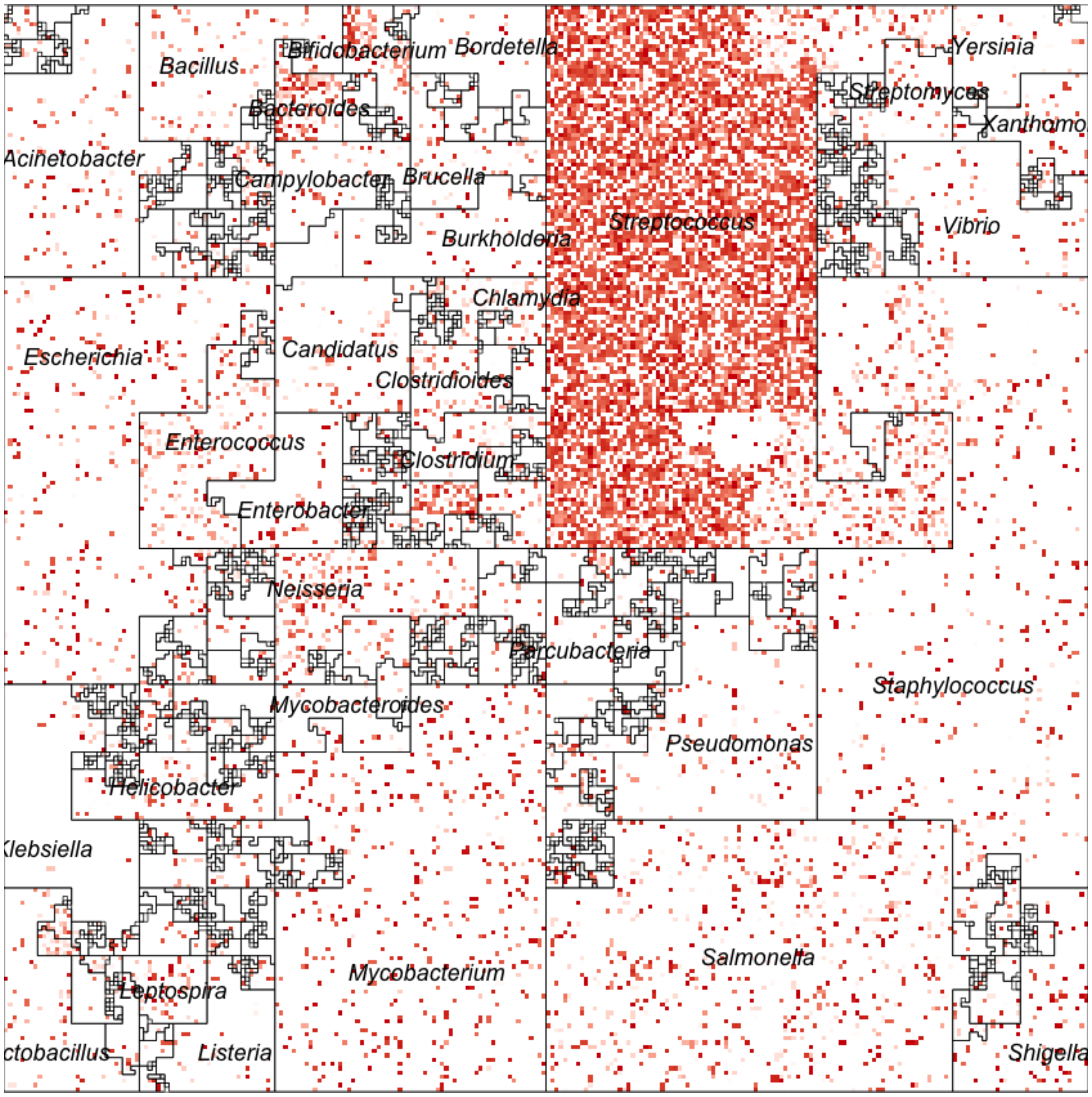


Figure 6.35: Differential Abundance. P-value microbiome map of an analysis of differentially abundant taxa in buccal mucosa and gastrointestinal tract. Color intensity corresponds to p-value significance with lower p-values being more bright. Only p-values ≤ 0.05 are displayed.

6.6.3 Composite Maps

A meaningful advantage of fixing the locations of taxa in the microbiome maps is that visualizations can be used to effectively present composite images of multiple maps. One concept for a composite map is that of an "average" map that displays the mean abundance values of a set of samples, and Figure 6.37 contains such a plot

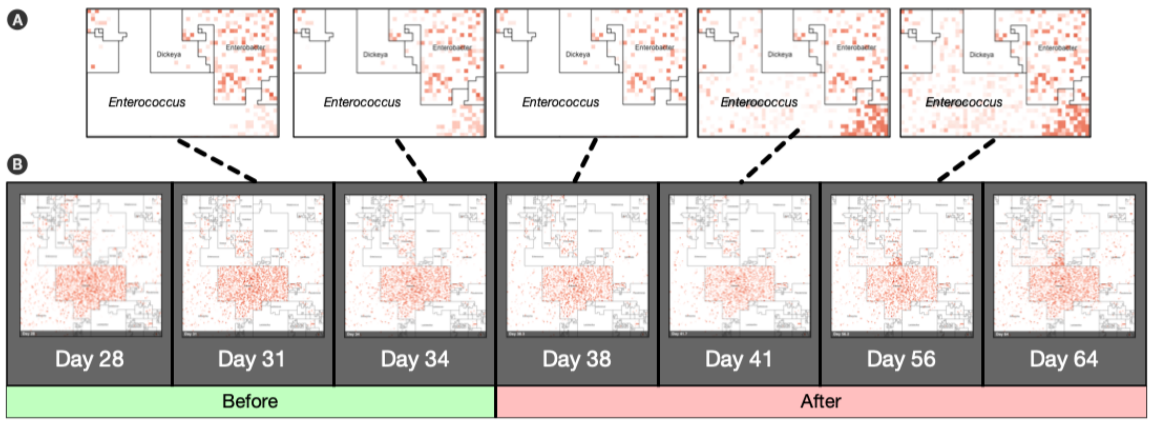


Figure 6.36: Animated Microbiome Map. Selected frames of an animated time-series visualization of 12,116 strains for a single patient from [51]. **Panel (A)**: zoomed regions of the *Enterococcus* neighborhood as it progresses through the antibiotic response. **Panel (B)**: Full resolution animated microbiome map is available at the project's repository via links from `biorg.cs.fiu.edu/jasper`

created from six (HMP) buccal mucosa samples (SRS024557, SRS045254, SRS063478, SRS014683, SRS050422, and SRS064276). Figure 6.37 displays the mean abundance of the six samples (as pixel intensities). In the buccal mucosa average map, the signature *Streptococus* neighborhood of the buccal mucosa can be clearly seen as the neighborhood with the largest concentration of *hot spots* of abundant taxa. Also note that almost all the neighborhoods have a *hot spot*, but their density is not as high as in *Streptococus*. Another type of composite plot is that of a "aggregate" plot that collects the abundance values for a group of samples — grouped in any meaningful way that is valuable to a user. An example would be to create an aggregate map of a set of samples that have a common factor to them, say the general location from which they where collected: one could see the creation of an aggregate map of the *human mouth* that displayed the aggregate abundance values of samples taken from the "palatine tonsils", "buccal mucosa", and "supragingival plaque" regions, as an example. To aggregate the samples in such a map, we could use a `SUM()` function that sums all the abundance values for the strains found. Note that an aggregate map that uses the SUM() would be similar to an average map that use the mean, the only difference would be that the intensities shown in the average map would be normalized by the number of samples it represents.

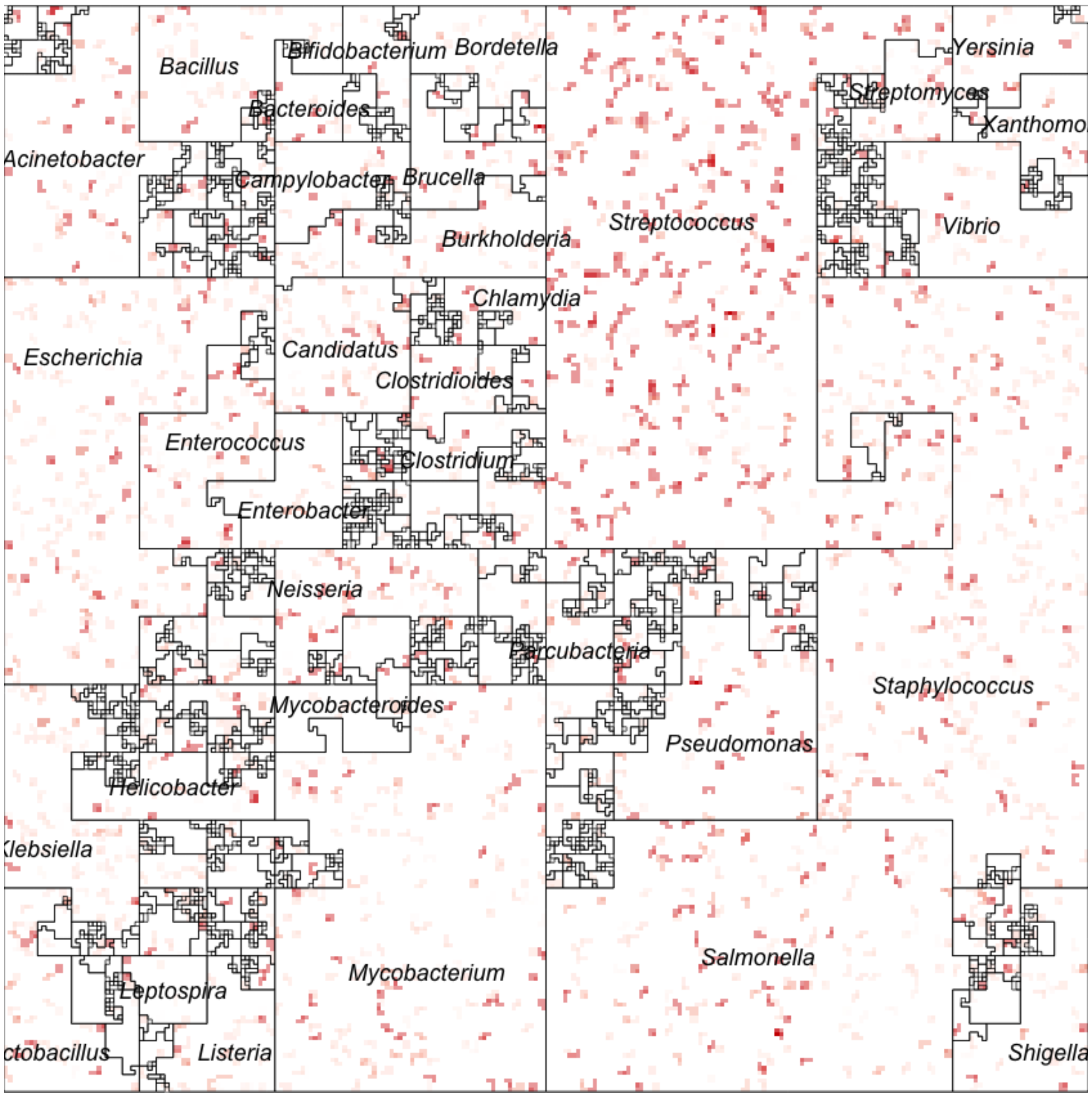


Figure 6.37: Buccal Mucosa Average Map. A composite map created from the averaged abundance profiles of six HMP buccal mucosa samples. Note the *Streptococus* neighborhood, a signature region for buccal mucosa samples.

6.7 Metagenomic Hilbert Inspector

As part of the Jasper release, we also offer a way for users to interactively inspect the microbiome maps they have created. We provide a tool with which users can load an image along with a set of annotations, and then click on a given segment of the Hilbert image and get information about the underlying microbial genome, along with other metadata. Details on the inspector can be found at the project's website and Section 6.7.

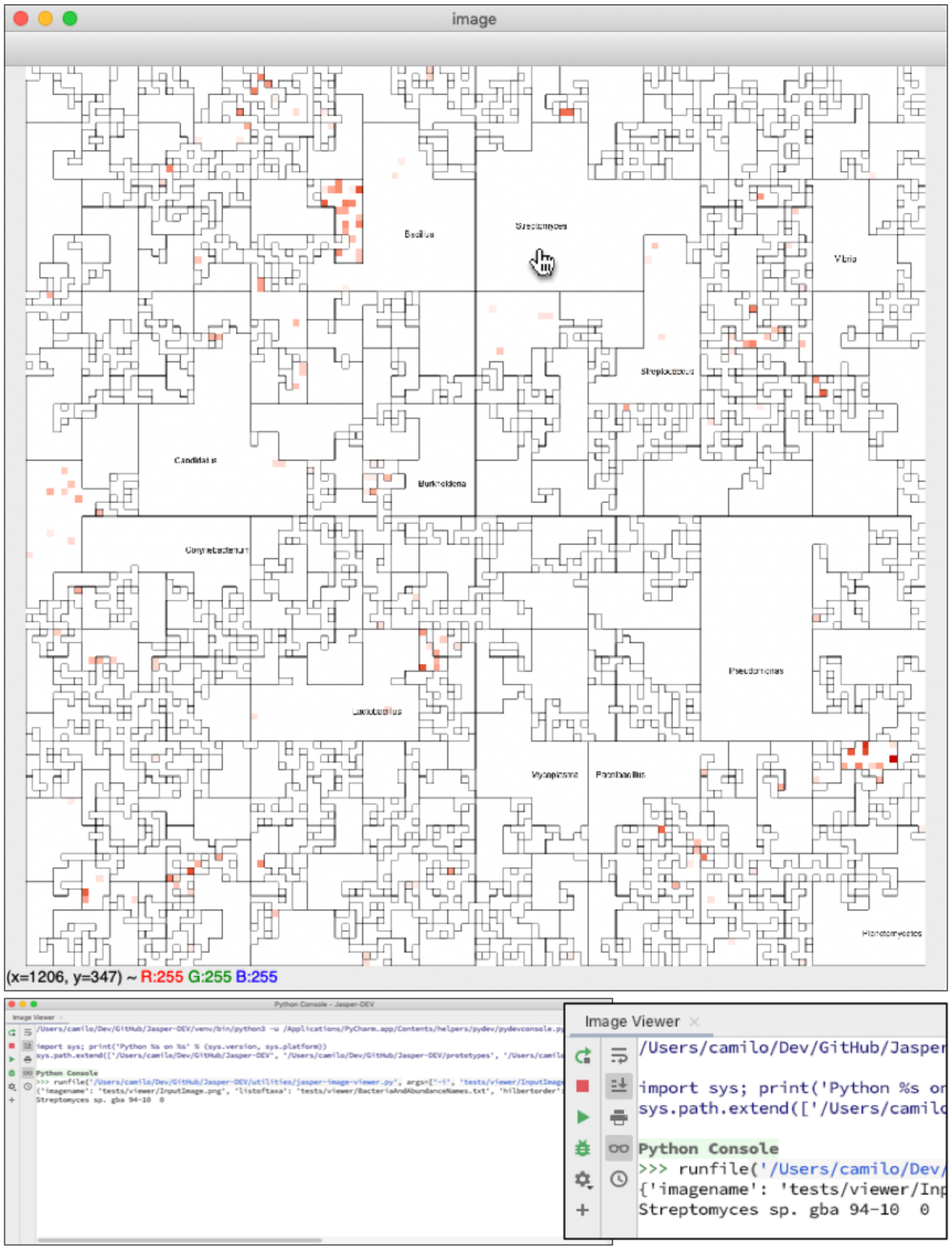
The inspector tool runs on python and relies on the OpenCV [23] and hilbertcurve [4] libraries. Users are able to load a microbiome map generated by Jasper by calling the "`jasper-image-viewer.py`" script supplied in the "`utilities`" folder of the main project directory. The program relies on three parameters to run:

1. "`-i`" The path to an image created by Jasper in .png format.
2. "`-l`" A text file with an ordered set of annotations that describes the order of the genomes in the image (this is provided by Jasper as part of its output).
3. "`-p`" The level of the curve that Jasper used to create the image (also provided as part of the output).

The basic command call for the image viewer is outlined in Listing 6.6, and users can click on any area of the image and the program will provide information on the genome that was located at the position of the mouse click. Figure 6.38 contains a screenshot of the image viewer displaying an image from the CKD dataset.

```
jasper-image-viewer.py -i /path/to/jasper-image.png
   -l /path/to/ordered-annotations.txt
   -p 7
```

Listing 6.6: Command call for launching the Jasper visual inspector.



```
(x=1206, y=347) ~ R:255 G:255 B:255
```

```
/Users/camilo/Dev/GitHub/Jasper-DEV/venv/bin/python3 -u /Applications/PyCharm.app/Contents/helpers/pydev/pydevconsole.py

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/camilo/Dev/GitHub/Jasper-DEV', '/Users/camilo/Dev/GitHub/Jasper-DEV/prototypes', '/Users/camilo

Python Console
>>> runfile('/Users/camilo/Dev/GitHub/Jasper-DEV/utilities/jasper-image-viewer.py', args=['-i', 'tests/viewer/InputImage
{'imagename': 'tests/viewer/InputImage.png', 'listoftaxa': 'tests/viewer/BacteriaAndAbundanceNames.txt', 'hilbertorder':
Streptomyces sp. gba 94-10  0
```

```
/Users/camilo/Dev/GitHub/Jasper

import sys; print('Python %s on
sys.path.extend(['/Users/camilo

Python Console
>>> runfile('/Users/camilo/Dev/
{'imagename': 'tests/viewer/Inp
Streptomyces sp. gba 94-10  0
```

Figure 6.38: Image Viewer. The JASPER image viewer utility being used to inspect a CKD image. The hand (top) clicked on the *Streptomyces* neighborhood, and the program displayed the image segment *Streptomyces sp. gba 94-10* as the species that was clicked on.

CHAPTER 7

CHARACTERIZING MICROBIOMES WITH CONVOLUTIONAL NEURAL NETWORKS

7.1 Introduction

Community abundance profiles give us a lens through which we can study microbiomes in varying levels of detail. In previous chapters we showed how to compute these community profiles and how to visualize them succinctly. Here we consider the problem of studying collections of microbiome profiles. An important followup challenge is to identify distinctive sets of features of cohorts of microbiomes so that these cohorts can be characterized. Characterizing microbiomes is an important task because it can help us understand the micro-ecosystem, and its functional and compositional characteristics under different biological and environmental conditions. This has practical applications in several fields including food safety and agronomy [19, 20, 57], and monitoring of foods during fermentation [76, 90, 102, 119, 140]. For example, in planetary exploration, it is important for NASA to characterize all Mars-bound probes and rovers so that they do not contaminate Mars with Earth's microbes [163, 28, 83, 146, 155]. Characterizing cohorts of microbiomes thus has applications in associating the features with spatial location, identifying features that characterize diseased samples from healthy, one disease subtype from another, treated samples from untreated, different types or dosages of drugs, stages of development, and so much more.

Characterizing microbiomes can be done in one of two ways. One possible approach is to identify microbiome-related *biomarkers* for cohorts of samples. For example, the presence of *Gardnerella* species in vaginal microbiomes is often a biomarker for bacterial vaginosis [15, 104]. A second and more general approach is to build a *computational model*, i.e., a machine learning (ML) model that can automatically *label* or *classify* a new microbiome sample based on its community abundance profile. We follow the latter approach of building computational models.

In this chapter, we consider the problem of characterizing microbiomes using existing and well-known ML techniques on their abundance profiles. Our favored method represents a departure from the standard ML approaches in the following sense. While the obvious approach is to consider the abundance profiles as a matrix of values to be input into any of multiple standard ML tools, we use the *microbiome maps* computed in Chapter 5 and then train a convolutional neural network (CNN) model [89]. We discuss the challenges of training CNNs when the number of samples is small, as well as how to optimize microbiome maps for CNNs. We discuss how current machine learning frameworks can be successfully employed in the field of metagenomics, and used to interpret the diverse relationships in a microbiome.

7.2 Approach

Modern deep learning frameworks [2] are able to detect, classify, and diagnose diseases such as skin cancer [41] (using 2D images), segment brain tumors [64] (using 2D image slices taken from CT or MRI scans), detect anomalies in electrocardiograms [36, 150] (using numerical signal data) and more recently, they have also been used in metagenomics to classify multiple diseases [87, 112] (using OTUs) and to classify disease stages [46] (using numerical OTU abundance and phylogenetic distance matrices). These deep learning frameworks are robust. CNNs are particularly powerful at classification and segmentation tasks when working with image datasets [54, 62, 89, 98, 157]. CNN architectures are primarily designed to work with images, and we take advantage of this assumption to analyze microbiome map visualizations.

7.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) [89] are a specific type of neural network that work well with data whose structure has a grid-like topology. Unlike traditional neural networks in which each neuron of each layer uses matrix multiplication to model the interactions between its inputs and output, CNNs use convolution operations in at least one layer, which has the effect of creating a sparse connectivity graph between the input and output units. The benefit of using convolutions over matrix multiplications is that it allows CNNs to describe complex interactions between many variables only using simple sparse connections between the units in adjacent layers — this gives CNNs the ability to learn local patterns, unlike traditional fully-connected networks that learn global patterns [31]. CNNs also have the advantage of producing high-level abstract objects rather than just a class for a classification job, or continuous real-values for regression problems.

Convolutional layers in a CNN are specifically designed to work well with images, and take as input a volume with shape "`width, height, depth`" where "`width`" and "`height`" are the width and height of the input images, and "`depth`" is the number of channels that the image has: a color image would have "`depth=3`", the depth being a channel for each color in the RGB spectrum. CNNs are primarily built using three types of layers: convolutional layers, pooling layers, and fully-connected layers. These layers are defined below:

Fully Connected Layer: Typically found in traditional neural networks, each neuron in such a layer is connected to every neuron in the previous layer, allowing the entire previous layer to impact its output.

Pooling Layer: This layer downsamples its inputs from the previous effectively decreasing the "`width`" and "`height`" without changing the "`depth`".

Convolutional Layer: In this type of layer each neuron is only connected to a subset (neighborhood) of neurons in the preceding layer.

Convolutional networks transform the input volume of images (“`width, height, depth`”) into an output volume of feature maps: convolutional layers create small patches (usually 3×3, or 5×5) over the input feature map and *slide* them across every position. These patches are called *convolutional kernels* and they compute the dot product of the kernel at each position with the values of the input at the same position. The depth of the output volume is arbitrary, and it is set by a parameter in the convolutional layer — each depth-level in the output volume represents a *filter* on the image that creates a feature map of the kernel that was used to create the filter. Figure 7.1 contains a visual representation of how a convolutional layer works.

Pooling layers as mentioned previously, downsample the feature maps that they take as input. Similar to convolutional layers, these layers operate through small patches (usually 2×2) that are slid across the feature map, but unlike the convolutional layers, they use a hard-coded `max()` function, a type of aggregation or averaging function to transform the output of the previous layer. The purpose of these layers is to make succeeding convolutional layers look at wider patches of the original image — this downsampling is what makes convolutional networks understand higher levels of abstractions (Figure 7.2).

The last set of layers in most CNNs are fully connected layers. These layers operate in the same manner as traditional feed-forward neural networks, and their function is to classify the input images. The very last output layer will often be composed of neurons that have a `softmax` activation function that will create a probability distribution across the classes that are being graded: if we were trying to classify images that contained “dogs” using a convolutional network (does the image contain a dog? Yes? No?), then the output layer would have a single neuron whose output would be the probability that the image contains a dog.

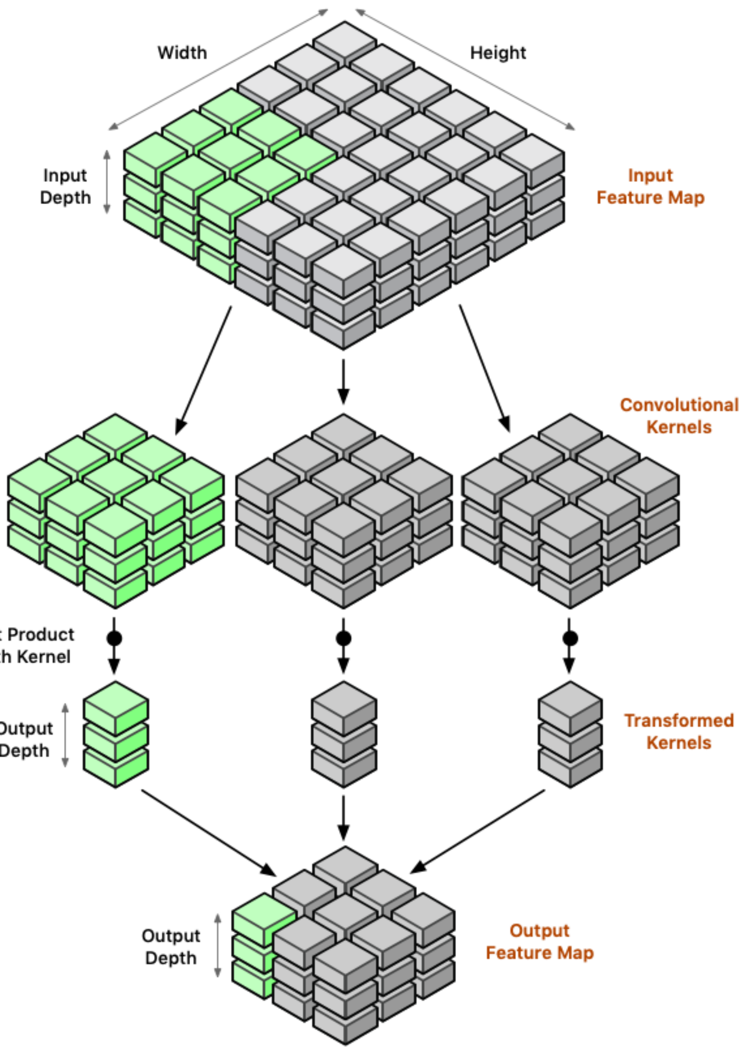


Figure 7.1: Convolutional Operations. Input images are turned into an input volume and small patches (convolutional kernels) and are slid across the input. The kernels compute the dot product with the input values at the same location, and the result is a output feature map of filters applied to the image. Figure inspired by Francois Chollet in *Deep Learning with Python*, (Figure 5.4, page 125) [31].

7.2.2 U-Net

The strength of convolutional networks is in their ability to perform classification tasks on images, and their traditional applications concern the assignment of a class label to an image. A "U-Net" is a modified convolutional architecture proposed by Ronneberger in 2015 [128] that contains two symmetrical paths of layers that first

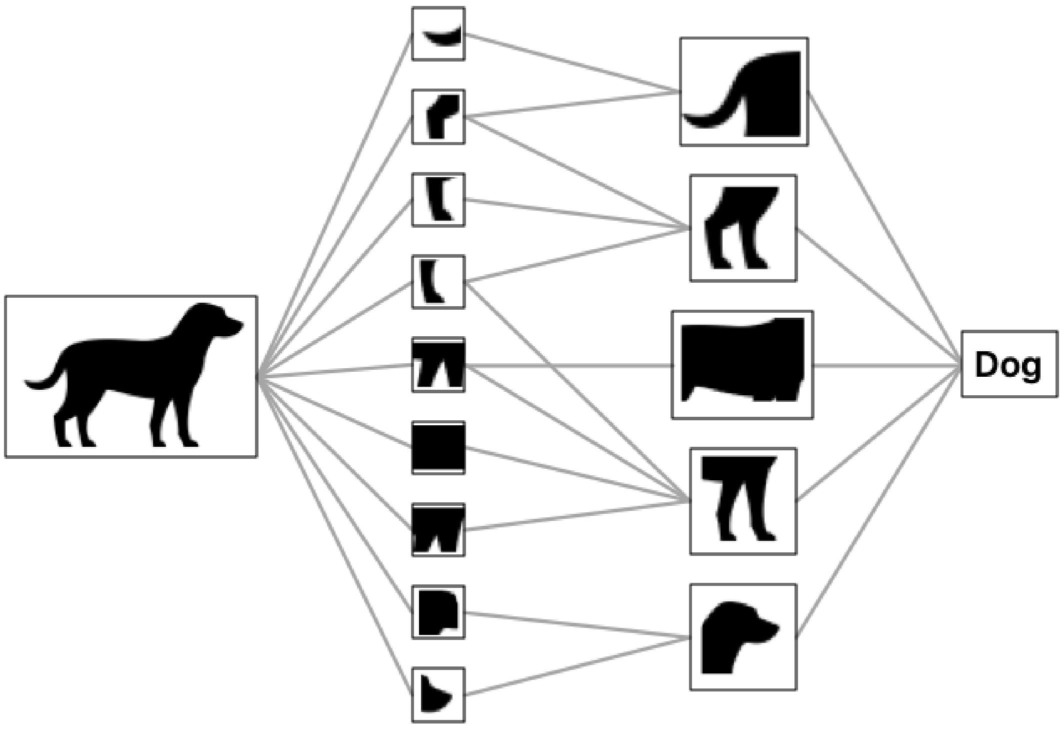


Figure 7.2: Higher Levels of Abstractions. A simple look at how a CNN would arrive at the answer for a *Dog*. Initial layers will learn to recognize local features such as lines, curves, and blobs of color; while succeeding layers will learn to recognize legs, ears, and snouts. Inspired by Figure 5.2 (page 123) from *Deep Learning with Python* by Francois Chollet [31]

perform a contraction and downsampling of the input, and then perform an expansion and upsampling. The architecture has been popular for segmenting objects of interest in images, and it has been particularly useful in biomedical imaging applications [69, 81, 151, 169].

Figure 7.3 contains the Ronneberger U-Net architecture from [128], and we can see that the network creates a distinct "U" shape by the union of its contracting path (left side) and expanding path (right side). The left side of a U-Net is a convolutional network with convolutional and pooling layers, but unlike a full convolutional network, it does not have any fully connected layers. The interesting part of this contracting path is that the depth of the output volume after each convolutional layer is doubled from that of the preceding convolutional layer, and this is done to make the network sensitive to the complex objects in the input. In the contracting path, the first four levels of contraction are comprised by two convolutional layers followed by a pooling layer until we reach the "bottom" of the "U" where the two convolutional layers are followed by a "up-convolution" layer, sometimes called a "deconvolution" layer, but more accurately described as a *transposed* convolutional layer. It is at this point that the expanding path starts, and the right side of the U-Net is basically the mirror image of the left side, but going in the opposite direction. The function of the *transposed* convolutional layers is to upsample the input feature maps coming from the contracting path on the left side of the U-Net, and each upsampling of the input is followed by two convolution operations to reduce upsampling artifacts.

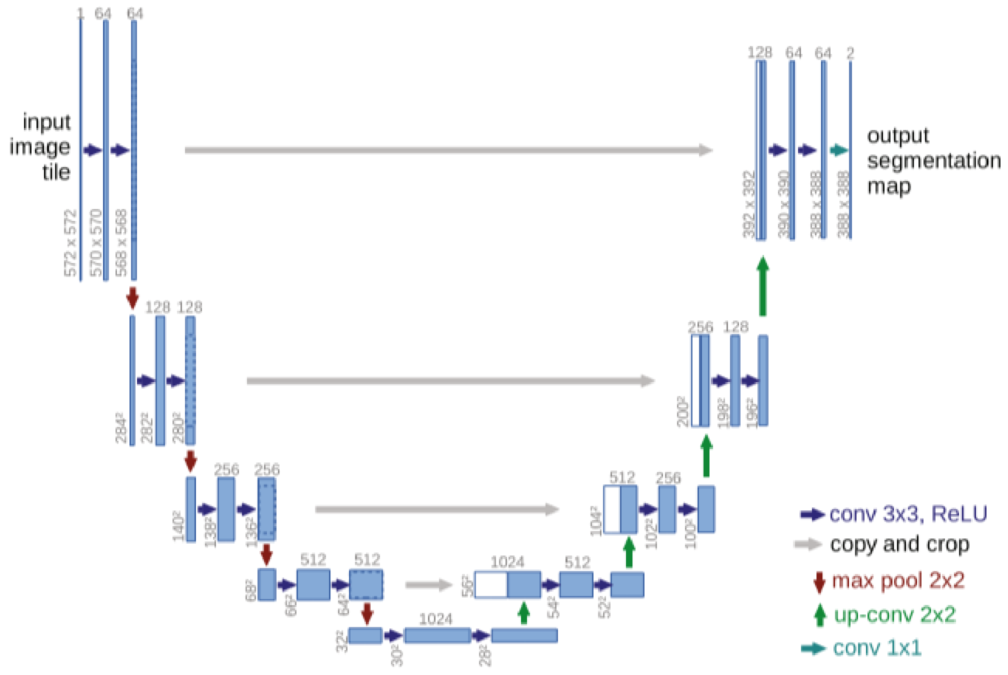


Figure 7.3: U-Net Architecture. The Ronneberger U-Net used for image segmentation. The architecture gets its name from the distinctive "U" shape formed by its two paths: the left side performs the convolutional operations, while the right side performs the deconvolutional operations. The output of a U-Net is a feature segmentation map. Figure 1 from the paper by Ronneberger [128], and reproduced with permission. Copyright 2015, Springer Nature.

7.3 Methods

The CNN architecture in the Amber framework is designed specifically for the characterization of microbiome samples generated from DNA sequencing experiments. Our characterization framework takes advantage of a convolutional network's powerful set of techniques for image classification (through a convolutional network architecture). Our framework is able to process samples using whole-genome or 16S rRNA gene metagenomic sequencing datasets. We start by creating detailed microbial community profiles with tools such as Flint and Kraken 2. The community profiles for both mWGS and 16S rRNA data are then transformed into microbiome maps by the Jasper system using the taxonomic ordering. A CNN-based neural network is then trained to learn patterns in those images by designing a CNN architecture. CNNs can take advantage of the proximity of related taxonomic groups in a microbial neighborhood of a microbiome map (see Section 5.3.1 for more on microbial neighborhoods).

Our framework is designed around the traditional layer architecture of a convolutional network (Section 7.2.1). We use the convolutional architecture to develop a classification framework that is able to classify microbiome samples using synthetic 2D microbiome maps.

After the training is completed with profile images of labeled microbiome samples, the trained CNNs are able to classify or label new and unlabeled samples by analyzing the presence or absence of the same patterns that are coded into the CNNs.

7.3.1 Convolutions for Microbial Neighborhoods

The primary instrument employed in our framework is convolutional operations on microbiome maps created with the Hilbert curve. The convolutional architecture and a typical microbiome map are shown in Figure 7.4. The native resolution of the image shown here is 256×256, which corresponds to a Hilbert curve at level 8. (For more on the level of the curve, refer to Figure 5.1 and Section 5.2.2.) The divisions in

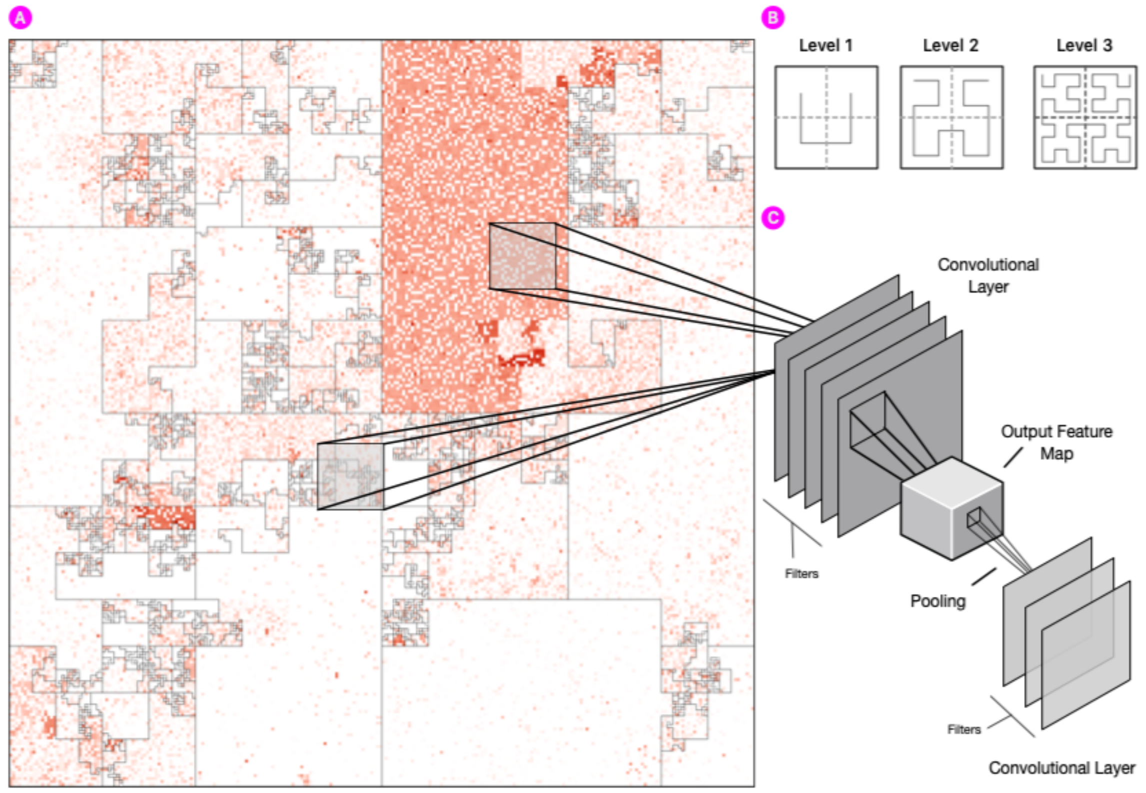


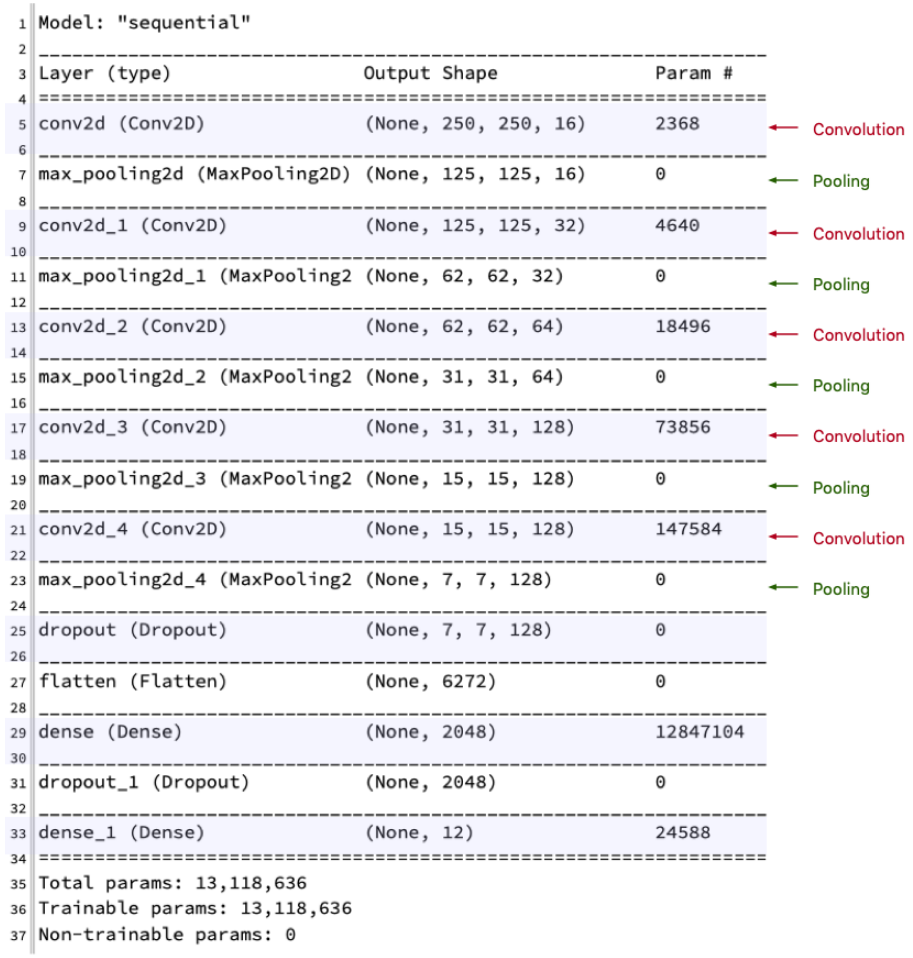
Figure 7.4: Convolutions on Microbial Neighborhoods. A microbiome map (panel **(A)**) created using the Hilbert curve, and scaled for display purposes. The native resolution for each image is 256 × 256, which corresponds to a Hilbert curve at level eight. Panel **(B)** shows how the Hilbert curve is recursively built by recursive partitioning of a square into four, and the first three levels of the curve are shown. Panel **(C)** shows the first stage of our convolutional architecture that we use for our microbiome classifier. Patches of size (5 × 5) are used in the initial convolutional layers, and they create 64 filters which comprise the "Output Feature Map". A `max()` pooling layer is used to downsample the feature map in the following round of convolutional layers.

the image called *Microbial Neighborhoods* represent different taxa as per a taxonomic ordering of microbial genomes. (See Section 5.3.1 for more on taxonomic ordering.) The patterns in the same neighborhoods in cohorts of images are spotted and learned by the CNN. We start by creating convolutional kernels of size 5 × 5 and slide them across the image to create 64 filters. All the filters taken together represent the output feature map of the first layer, which is then feed to a pooling layer that downsamples the images. Note that the architecture for the transposed convolutional layers is similar to the ones depicted in Figure 7.4, but we use a transposed convolutional layer rather than a convolutional one.

7.3.2 The Amber CNN Model

We implemented our framework using the TensorFlow library [2] version 2.1.0 in Python. The images that we are using are of relatively small sizes, as each pixel in the image represents a bacterial genome from a reference genome collection. Two sizes of images are used which correspond to the sizes of the reference genome database that is used to create the community profiles: a 256×256 image is created for community profiles that use 44K genomes, and a 128×128 image is used when we use 5K genomes; these sizes also correspond to the underlying DNA sequencing technology that was used to create them: the whole-genome datasets profile a larger set of microbes which result in the larger images (256×256 pixels), while the 16S rRNA gene sequencing dataset profiles a smaller set of microbes which result in the smaller image (128×128 pixels).

Figure 7.5 contains a textual description of the Amber CNN model. In typical CNN fashion, the model starts with a set (five here) of alternating convolutional and pooling layers. Note that in the figure, we see that the first convolutional layer's output shape is $250 \times 250 \times 16$ which corresponds to the model used when 44K genomes (mWGS data) are used in the microbiome map; the output shapes changes to $120 \times 120 \times 16$ when 5K genomes are used (16S rRNA). Following the convolutional and pooling layers, we see that a pair of dropout layers are used to prevent overfitting — these layers randomly set a fraction of its input neurons to zero during the weight updates throughout the training run. The last layer of the network is a densely-connected layer with a `softmax` activation function that creates a probability distribution over the output classes of the dataset (12 body sites for the HMP dataset, and six conditions for the CKD dataset). The model uses the Adam optimizer [79] to update the network's weights, with a learning rate set to 0.001.



```
 1 Model: "sequential"
 2 _________________________________________________________________
 3 Layer (type)                 Output Shape              Param #
 4 =================================================================
 5 conv2d (Conv2D)              (None, 250, 250, 16)      2368
 6 _________________________________________________________________
 7 max_pooling2d (MaxPooling2D) (None, 125, 125, 16)      0
 8 _________________________________________________________________
 9 conv2d_1 (Conv2D)            (None, 125, 125, 32)      4640
10 _________________________________________________________________
11 max_pooling2d_1 (MaxPooling2 (None, 62, 62, 32)        0
12 _________________________________________________________________
13 conv2d_2 (Conv2D)            (None, 62, 62, 64)        18496
14 _________________________________________________________________
15 max_pooling2d_2 (MaxPooling2 (None, 31, 31, 64)        0
16 _________________________________________________________________
17 conv2d_3 (Conv2D)            (None, 31, 31, 128)       73856
18 _________________________________________________________________
19 max_pooling2d_3 (MaxPooling2 (None, 15, 15, 128)       0
20 _________________________________________________________________
21 conv2d_4 (Conv2D)            (None, 15, 15, 128)       147584
22 _________________________________________________________________
23 max_pooling2d_4 (MaxPooling2 (None, 7, 7, 128)         0
24 _________________________________________________________________
25 dropout (Dropout)            (None, 7, 7, 128)         0
26 _________________________________________________________________
27 flatten (Flatten)            (None, 6272)              0
28 _________________________________________________________________
29 dense (Dense)                (None, 2048)              12847104
30 _________________________________________________________________
31 dropout_1 (Dropout)          (None, 2048)              0
32 _________________________________________________________________
33 dense_1 (Dense)              (None, 12)                24588
34 =================================================================
35 Total params: 13,118,636
36 Trainable params: 13,118,636
37 Non-trainable params: 0
```

Figure 7.5: Amber CNN Architecture I. The primary architecture of the Amber CNN model when 44K genomes are profiled. The network starts with the typical CNN architecture of having alternating convolutional and pooling layers (lines 5-23). After the last pooling layer, we introduce a dropout layer (line 25) and flatten its output (line 27). The flattened input is used as input to a densely connected layer (line 29) which feeds into another dropout layer before delivering its output to a softmax layer (line 33) which contains 12 neurons (one for each class in the HMP dataset).

Although the images are small, training on a laptop computer is not feasible as the underlying hardware components are not designed for training with a large set of images. A powerful machine with 48 CPUs and 768 GB of RAM memory might seem like a good option, but such machines are slow to train a large CNN model — we tried such a setup using the "`castalia`" machine (see Section 3.2) and it took over 14 hours to train a test network of four convolutional layers and four pooling layers, using a subset of 250 (256×256) images. Training CNNs is much faster using computing machines with graphical processing units (GPU) [94, 137, 154], as the GPUs have a lot of small processing cores that can perform many simple operations very quickly and in parallel; this benefits CNNs because the weight updates in the training step are nothing more than matrix multiplications which can be performed by the GPU extremely quickly. We first trained our model using the "`castalia`" machine and saw training runs taking multiple hours: as low as six hours, and as high as 10 hours for 200 epochs (the discrepancy being due to the load that the machine was under while running the experiments as other users were also executing jobs). We then moved the training of the model to another machine equipped with two NVIDIA GeForce GTX 1080 Ti GPUs, each with $3,584$ cores and 11GB or RAM memory. Training on the NVIDIA GPU was substantially faster as we saw average training runs lasting no more than 5 minutes depending on the machine's job loads.

Training in the Cloud

We previously developed the capability of running the FLINT system on Amazon Web Services (AWS) [9] (Section 4.2) using a distributed cluster architecture that runs on commodity machines (Section 4.3.3) running in a *master-worker* configuration. The primary building block of a cloud cluster is a simple machine (worker node) that is configured with the desired software libraries. Such machines are usually quite affordable through the AWS Spot Market [10]. AWS provides a basic building block for machine learning workflows called a Amazon Deep Learning AMI [5] which we use to bootstrap the process of creating a cluster used to run our training models.

Training a TensorFlow model in a distributed cluster is similar to the processing previously performed in Chapter 3 and 4, with a cluster architecture similar to the layout in Figure 1.1. A master node is deployed which will serve as the primary execu-

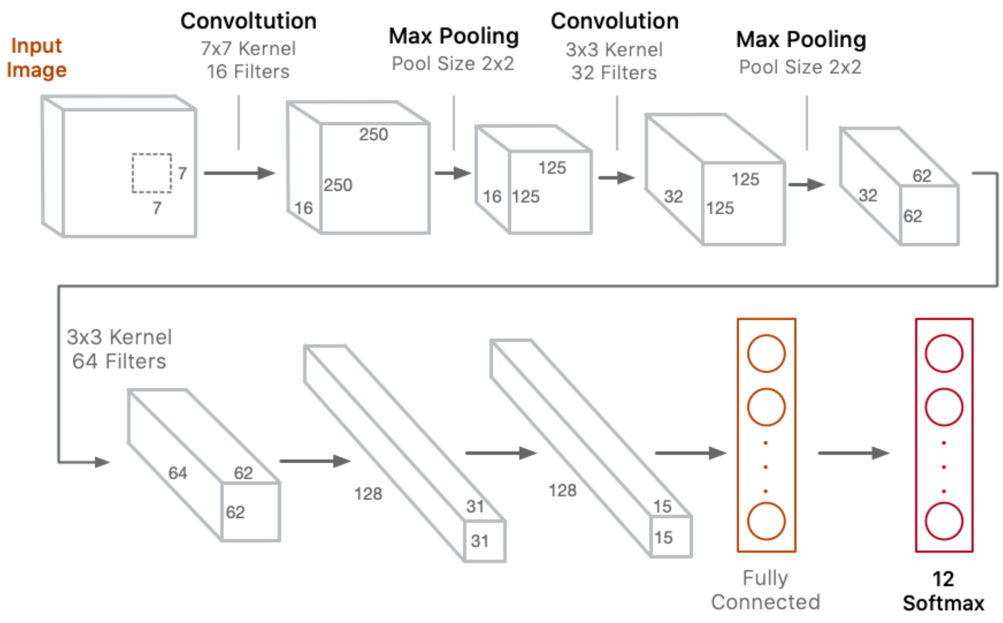


Figure 7.6: Amber CNN Architecture II. Illustrated version of the Amber CNN model. The model starts with an input volume measuring *width* × *height* × *depth*, where the width and height are the corresponding width and height of the input images (256 × 256 for mWGS, and 128 × 128 for 16S), and the depth corresponds to the number of channels: three for RGB images, one for grayscale. The model starts by using a 7 × 7 kernel, and learning 16 filters. Notice that as the model gets deeper and deeper, we learn more and more filters (16, 32, 64, and 128) and the dimensions of the output volumes get smaller and smaller (250, 125, 62, 31, and 15) which is typical for a CNN that is learning higher levels of abstraction with each subsequent layer.

tion point for the training script. Multiple worker nodes are also deployed, and these nodes are the ones that will perform the bulk of the computation. A configuration script runs during the cluster provisioning step that loads all the necessary software libraries that we will be using; libraries such as Pandas, Numpy, Scikit-Learn, Scikit-Image, and others are all installed on both the master and worker nodes during the provisioning step.

Training runs are started on the master node using a Python script that contains the code for creating the convolutional network using TensorFlow. This script performs any preprocessing that is required and prepares the training, test, and validation datasets. Once everything is assembled, and the convolutional network has been created, the script compiles the network and initiates the training run on the worker nodes. The Tensorboard tool [2] is available in the cluster for visualizing the graph that TensorFlow has created and for debugging purposes.

7.4 Results and Discussion

Flint [153] (Chapter 4) and Jasper (Chapter 5) make the process of creating microbial profiles and microbiome maps relatively straightforward. We used Flint and Jasper to create profiles and microbiome maps from 328 samples from healthy human subjects from the Human Microbiome Project (HMP) [149]. We also used Kraken 2 and Jasper to create profiles and maps from a dataset of samples from patients at one of five different stages of Chronic Kidney Disease (CKD). The 328 samples from the HMP dataset were sampled from 12 different body sites: "Attached & Keratinized Gingiva", "Buccal Mucosa", "Gastrointestinal Tract", "Nares", "Oral", "Palatine Tonsils", "Posterior Fornix", "Retroauricular Crease", "Supragingival Plaque", "Throat", "Tongue Dorsum", and "Vaginal". The CKD samples were selected based on their glomerular filtration rate (see Kidney Disease Improving Global Outcomes (KDIGO) [78]), and a total of six classes were created: Control, CKD Stage 1 (CKD 1), CKD Stage 2 (CKD 2), CKD Stage 3 (CKD 3), CKD Stage 4 & 5 non-dialysis dependent (CKD 4-5ND), and CKD Stage 5 dialysis dependent (CKD 5). The CKD stages were determined based on the worsening function of the kidney patients [78].

7.4.1 Comparison to other Classifiers

The ability to correctly classify microbiome samples using *Microbiome Maps* was tested by first evaluating seven well-known classifiers with the community abundance profiles that were used to create the microbiome maps. The community profiles were used as input in their tabular form to the following classifiers: "Nearest Neighbor",

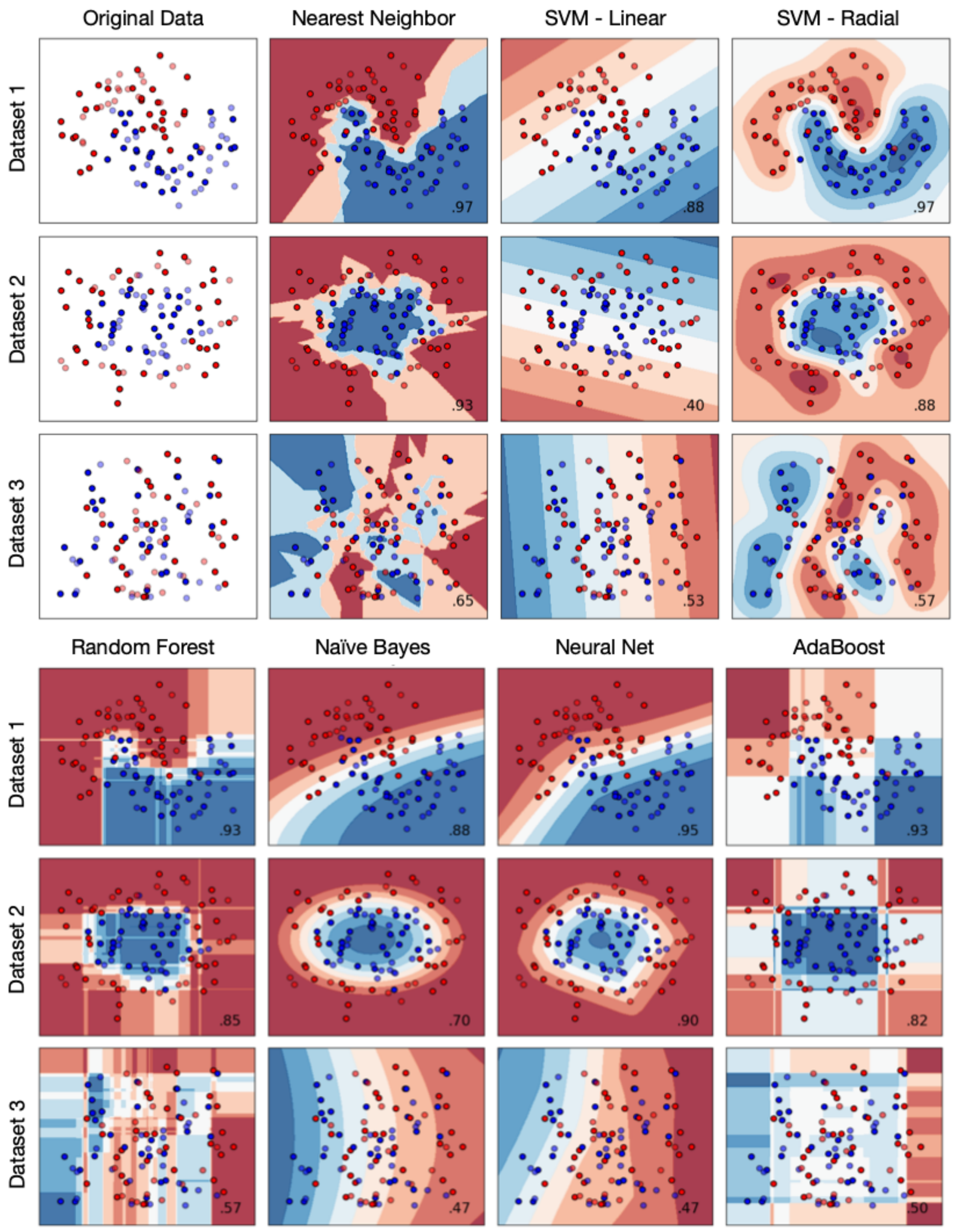


Figure 7.7: Synthetic Data Classification. Seven classifiers were tested with synthetic datasets created with the scikit-learn library [117], using code adapted from scikit-learn's examples [118]. The "Original Data" depicts the original data points (x, y) created for the trials with two classes, A and B, for three datasets, *Dataset 1* (created using the `make_moons()` function), *Dataset 2* (created using the `make_circles()` function), and *Dataset 3* (created using a randomized `make_classification()` function). Overall accuracy for each classifier is displayed in each box (label in dark font).

"Support Vector Machine (linear)", "Support Vector Machine (Radial)", "Random Forest", "Naïve Bayes", "Multi-Layer Perceptron (Neural Net)", and "AdaBoost".

To start the comparison, we created three synthetic datasets of random (x, y) coordinate points using the $scikit-learn$ library [117]. To start the comparison, we created three datasets of random (x, y) coordinate points using the $scikit-learn$ library [117] (Figure 7.7). The three datasets were created using the `make_moons()`, `make_circles()`, and `make_classification()` functions from scikit-learn. These functions create a series of pseudo-random points suitable for classification trials by first creating points along a sine wave (`make_moons()`), or along a circle (`make_circles()`), and then adding Gaussian noise to spread them out. The third dataset was created by calling the `make_classification()` function and modifying the return set of points with a random set of numbers drawn from a uniform distribution using the *numpy* library [114] (note that we use `make_classification()` as a convenience function to get the correct input shape as required by the classifiers). These three datasets were benchmarked against the seven classifiers, and their classification boundaries plotted (Figure 7.7). The goal of this experiment was to get familiar with the classifier's decision boundaries, as well as to try to understand on what type of dataset each classifier would perform the best. Note that detailed parameters for the classifiers tested are available in Appendix 1.

After running the seven classifiers with synthetic data, we tested their performance with the 328 samples from HMP and 12 classes (body sites). We took the community profiles generated by FLINT in their tabular form and used it as input. Figure 7.8 contains the results of the trial. When using the tabular-numerical profile, the "Neural Net" classifier performed the best with 85% accuracy, and "Naïve Bayes" came in second with 84%. The "AdaBoost" classifier performed the worst with 51%. The AMBER CNN obtained a classification accuracy of 92% when trained with a set

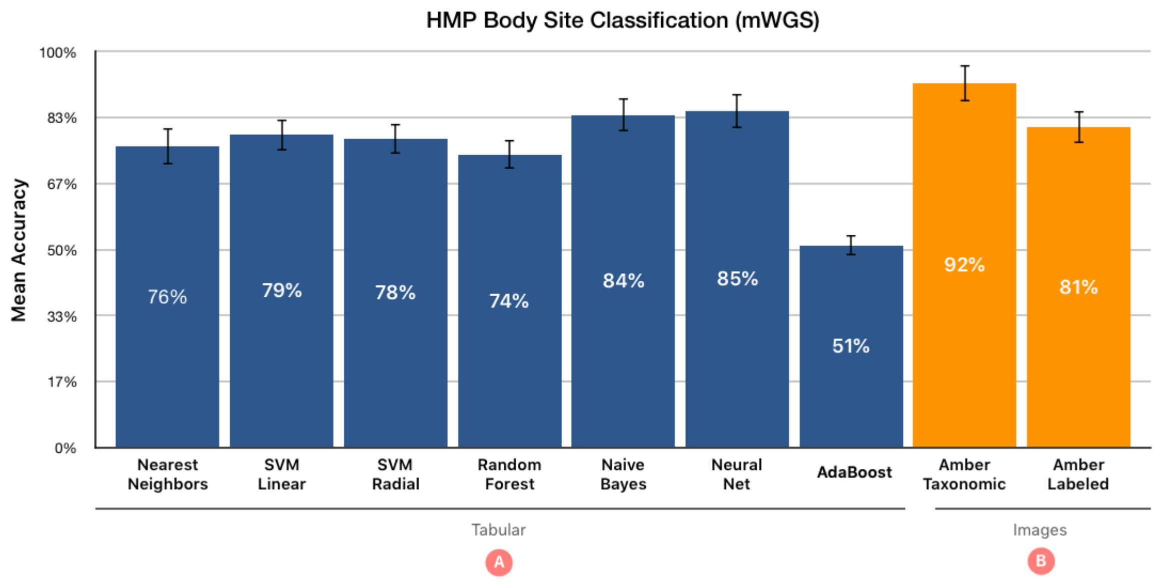


Figure 7.8: HMP Data. Mean accuracy of the seven classifiers when run with profiles created with FLINT. The community profiles contained 44K microbial genomes (features), and 12 classes from HMP human body sites. Panel (A) contains the results of the classifiers with a tabular (numerical) profile used as input. Panel (B) is used for comparison, and displays the mean accuracy of running the AMBER CNN with the two orders (taxonomic and labeled) using the same tabular profiles converted to microbiome maps.

of images created the taxonomic ordering scheme; AMBER obtained a accuracy of 81% when trained with labeled-order images.

We also tested the seven classifiers with community profiles for the CKD set. The data for these profiles was created from 16S rRNA gene sequencing and therefore had a lower resolution (less microbes to profile against) than the mWGS profiles from the HMP set. We used the Kraken 2 software [166] to create the profiles using their "16S" database. The results for this trial are displayed in Figure 7.9. Note that the overall accuracy for this set is below that of the HMP set, a likely result due to the smaller number of microbes present in the profile which is to be expected since the 16S rRNA gene sequencing experiments are very different from HMP's whole-genome sequencing. Just like in the HMP trial, the "Neural Net" classifier is the one with the highest accuracy at 38%. Section 7.4.3 contains more details on training the model using the 16S CKD dataset.

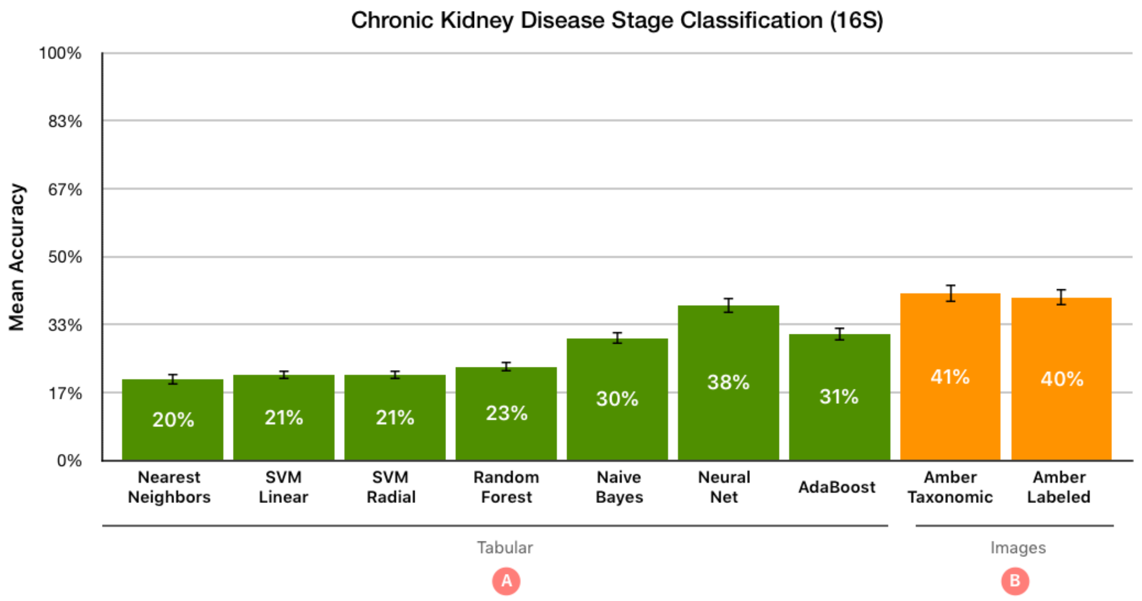


Figure 7.9: Chronic Kidney Disease Data. Mean accuracy of the seven classifiers when run with profiles created with Kraken 2. Unlike the profiles created with Flint these profiles only contained 5K microbial genomes, and 6 classes (CKD stages plus control). Panel (A) contains the results of the classifiers with a tabular (numerical) profile used as input. Panel (B) is used for comparison, and displays the mean accuracy of running the Amber CNN with the two orders (taxonomic and labeled) using the same tabular profiles converted to microbiome maps.

7.4.2 Training with HMP Body Site Maps

After gauging the performance of the seven classifiers with tabular data (Section 7.4.1), we proceeded to test the performance of our custom convolutional neural network (Section 7.3.2) trained with two datasets of microbiome map images: the first, a collection of 328 healthy human subjects from the human microbiome project (HMP); the second, a collection of 200 samples from patients with multiple stages of chronic kidney disease collected from Kangwon and Seoul National University Hospitals (Section 7.4.3). Each of the two previously mentioned datasets were divided out corresponding *training* and *testing* sets, with a 70/30 split, or 70% of the dataset being distributed into the training set, and 30% for the testing set.

HMP Body Sites

The HMP dataset is comprised of 12 classes representing the collection sites from healthy human subjects: “Attached & Keratinized Gingiva”, “Buccal Mucosa”, “Gastrointestinal Tract”, “Nares”, “Oral”, “Palatine Tonsils”, “Posterior Fornix”, “Retroauricular Crease”, “Supragingival Plaque”, “Throat”, “Tongue Dorsum”, “Vaginal”, and “Vaginal Introitus”. Figure 6.9 displays a distribution of the number of samples in each class.

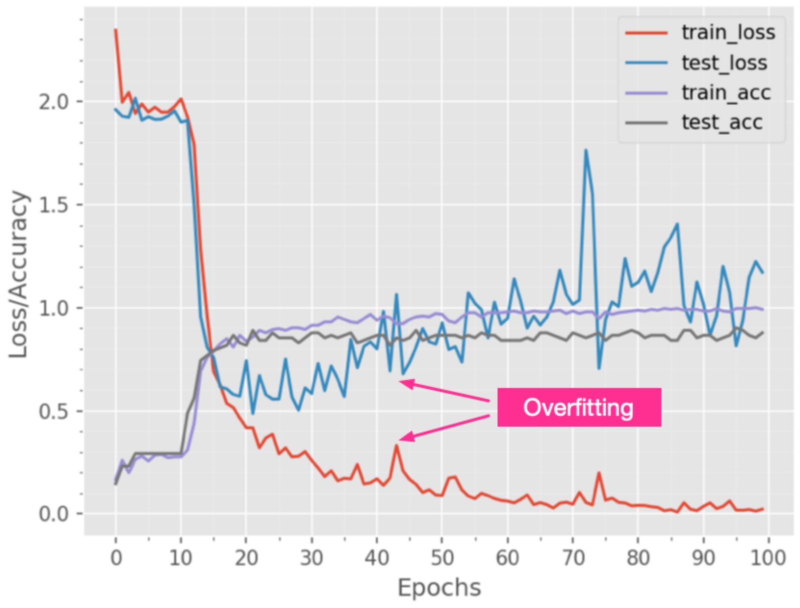


Figure 7.10: Training with HMP Data. Accuracy (purple, gray lines) and loss (red, blue lines) for training and testing the AMBER CNN with the 328 HMP samples for 100 epochs. While both training (99%) and testing (92%) accuracy are high, the diverging paths of the training and testing loss indicate that the model is overfitting the data.

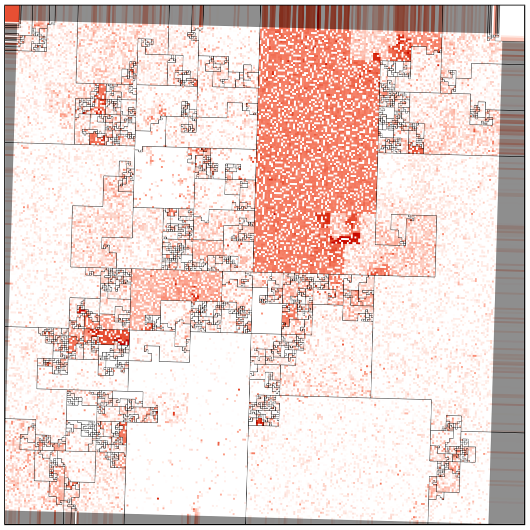
Microbial community abundance profiles were generated for each of the samples in the HMP dataset using the FLINT software (Chapter 4), and then the profile for each sample was converted into a set of microbiome maps (Chapter 5) using the two ordering schemes described in Sections 5.3.1. Note that the microbiome maps in this dataset consist of the transformed abundance profiles for 44K genomes profiled against each of the HMP samples using mWGS data. For these microbiome maps, each image has dimension 256×256 and contains 3 RGB color channels. Each pixel in the image corresponds to a genome in the database (see Section 5.3.1). The training accuracy of the model with 12 classes (each a body site from which the sample was collected from) after 160 epochs of training is 0.99, and testing accuracy is 0.85.

Figure 7.10 displays a representative training experiment done with the HMP dataset (12 body site classes). The experimental run consisted of training the AMBER CNN for 100 epochs with the HMP microbiome maps. After the 100 epochs of training, we see that the training accuracy of the model is 99%, and the testing accuracy is 92%. However, we can see from the traces of the training and testing losses that our model overfits the data by a large amount. We can see this by the characteristic overfitting indication of diverging training and testing losses. Overfitting starts from about epoch 15, and lasts for the duration of the training run. The loss outlines tell us how good the model is performing after each epoch, and while our training loss is decreasing after each epoch, our testing loss is increasing, which tells us right away that our model is not generalizing well.

After analyzing the results for the HMP body site dataset, and determining that the model was overfitting the data, we applied two well-known techniques [91] for minimizing overfitting: *regularization* [30, 172, 176] and *data augmentation* [68, 116, 135]. We implemented the regularization technique by adding two dropout layers [75] to the network: the first layer right after the last pooling layer, and the second layer right before the last densely connected softmax layer (lines 25 and 31 in Figure 7.5). Data augmentation was implemented by using a custom `ImageDataGenerator` class using the Keras and Tensorflow libraries. Appendix 1, Section A1.2 contains the particulars on the data augmentation procedures.

Figure 7.12 displays the results after applying the two techniques to address overfitting. The overall accuracy for the model went down, to 87% training accuracy and 89% testing accuracy, but we can see the effect that the dropout layers and the data augmentation had: the trajectory of both the training and test losses follow the same path without any divergence as in the original model. Note that it is unusual for the training accuracy to be lower than the testing accuracy as the model is being optimized with the training set. This effect is usually the result of applying the dropout regularization technique as dropout layers set the input of some fraction of neurons to zero, effectively disabling them. However, dropout layers only apply their effects during the training step, and not during the testing step, and their effect is generally reflected in the training accuracy being lower than the testing accuracy.

Table 7.1 contains a detailed classification report for the HMP dataset. In the table, the "No. Samples" column contains the number of representative samples for


(a) Augmented buccal mucosa microbiome map.

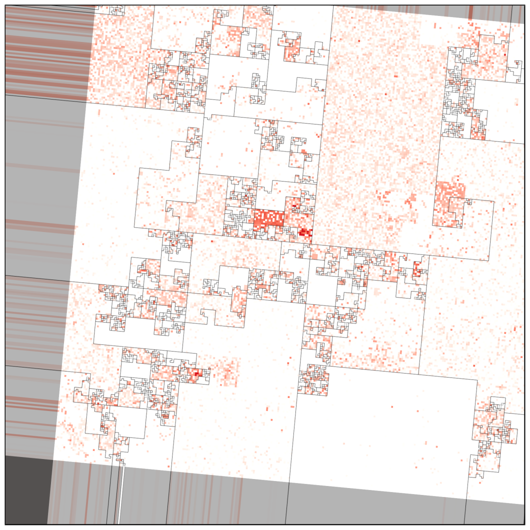

(b) Augmented nares microbiome map.

Figure 7.11: Augmented microbiome maps. Representative images of performing data augmentation for reducing overfitting. The HMP training dataset used by the AMBER CNN was augmented by a set of augmented images that were modified by rotating them, as well as adjusting the luminosity, and transparency properties, as well as RGB color channel levels.

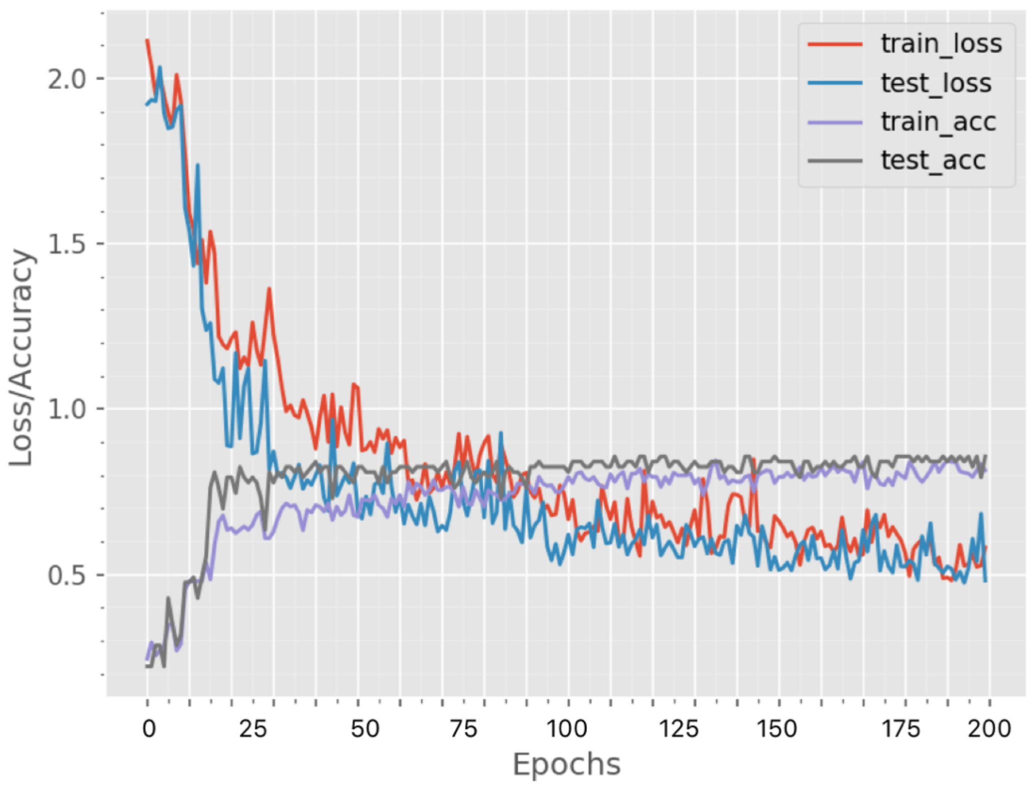


Figure 7.12: HMP Training Corrected for Overfitting. Accuracy (purple, gray lines) and loss (red, blue lines) for training and testing the Amber CNN (with dropout layers) with 328 augmented HMP samples for 200 epochs (all 12 classes). Training and testing accuracies decreased to 87% and 89% respectively, but there was no separation in the trajectories of the training and testing losses.

a given class, and we can see that when the number of samples is greater than 5, then the performance of the model is generally very good (for the exception of the tongue dorsum class). Classes with such a small number of samples are probably not being represented very well in the training and testing datasets, and the model might not be being exposed to a sufficiently large number of samples to learn from them. To test this hypothesis, we removed the classes that had less than seven samples (the classes that had a f1 score of 0.00), and this resulted in a new dataset that contained only eight classes and 310 samples (the "Attached/Keratinized Gingiva", "Oral", "Palatine Tonsils", and "Vaginal" classes and their samples were removed). We re-trained the Amber CNN model with the new eight-class dataset and saw

Class	No. Samples	Precision	Recall	F1 Score
Attached/Keratinized Gingiva	4	0.00	0.00	0.00
Buccal Mucosa	74	1.00	1.00	1.00
Gastrointestinal Tract	49	0.67	1.00	0.80
Nares	94	0.82	0.95	0.88
Oral	4	0.00	0.00	0.00
Palatine Tonsils	5	0.00	0.00	0.00
Posterior Fornix	54	0.96	0.92	0.94
Retroauricular Crease	15	0.50	1.00	0.67
Supragingival Plaque	10	1.00	1.00	1.00
Throat	7	1.00	0.92	0.96
Tongue Dorsum	7	0.25	0.50	0.33
Vaginal	5	0.00	0.00	0.00

Table 7.1: HMP Classification Report (12 Classes). Evaluation of the performance of the AMBER CNN model in classifying the twelve classes of the HMP dataset. The model performs very good for many of the classes, but fails notably for a couple of classes (those with an F1 score of 0.00). Classes with an F1 score of 0.00 should probably be dropped, as their number of samples is not enough to be used. The "No. Samples" column contains the number of input samples for each class. The "Precision" column contains the proportion of positive classifications that was correct, the "Recall" column contains the proportion of actual positive classifications, and the "F1 Score" column contains the harmonic mean of the precision and recall columns.

that the training accuracy went up to 99%, and the testing accuracy went to 94%, suggesting that the model sees a substantial improvement in accuracy when classes are represented in sufficient (i.e., at least seven) classes. Table 7.2 contains a detailed report on the model's performance across the eight classes.

In order to better understand how the AMBER CNN model is learning to characterize the HMP classes, we created a visualization of the model's convolutional layer's activation functions. Figure 7.13 displays the activations at each layer, starting at the initial layers (top of figure) all the way to the fifth layer (bottom). Following the design of our model (Figure 7.5), the initial layers of our model start by learning to identify classes by focusing on concrete visual features that we can readily identify.

Class	No. Samples	Precision	Recall	F1 Score
Buccal Mucosa	74	1.00	0.88	0.93
Gastrointestinal Tract	49	1.00	1.00	1.00
Nares	94	1.00	1.00	1.00
Posterior Fornix	54	0.93	1.00	0.96
Retroauricular Crease	15	0.33	0.50	0.40
Supragingival Plaque	10	0.95	0.95	0.95
Throat	7	1.00	1.00	1.00
Tongue Dorsum	7	0.80	1.00	0.89

Table 7.2: HMP Classification Report (8 Classes). Evaluation of the performance of the AMBER CNN model in classifying a reduced set of eight classes of the HMP dataset. The model performs very good achieving a training accuracy of 0.99% and a testing accuracy of 0.94%.

As we go deeper and deeper into the convolutional layers, we start to see that the model is learning to identify classes (buccal mucosa in Figure 7.13) by focusing more on higher levels of abstraction, and less on details that we can understand visually. This is the result of applying the pooling layers after each convolutional layer, as we are forcing the deeper layers to focus more on higher levels of abstractions.

A great example of what is happening in Figure 7.13 comes from the Spanish artist Pablo Picasso [160]: in 1945 he created a series of lithographs to portray a bull at various levels of abstraction. Figure 7.14 shows the Picasso images, with the bull drawn in great detail in the drawings of the first column, and then progressively being drawn at higher levels of abstraction in the remaining columns, finishing with the bull in its most simple (and unmistakable) form in the last drawing of the last column. The bull abstractions from Picasso are analogous to how the CNN is learning to identify microbiome map classes: the early layers of the model are learning to identify maps in great detail, but by the time we get to the deeper layers ("Convolution 5" in Figure 7.13), the CNN is learning to identify maps at higher levels of abstractions.

To human eyes, the abstracted layers in the deeper layers are not as attractive as the ones sketched by Picasso, but we can "observe" the abstracted microbiome images.

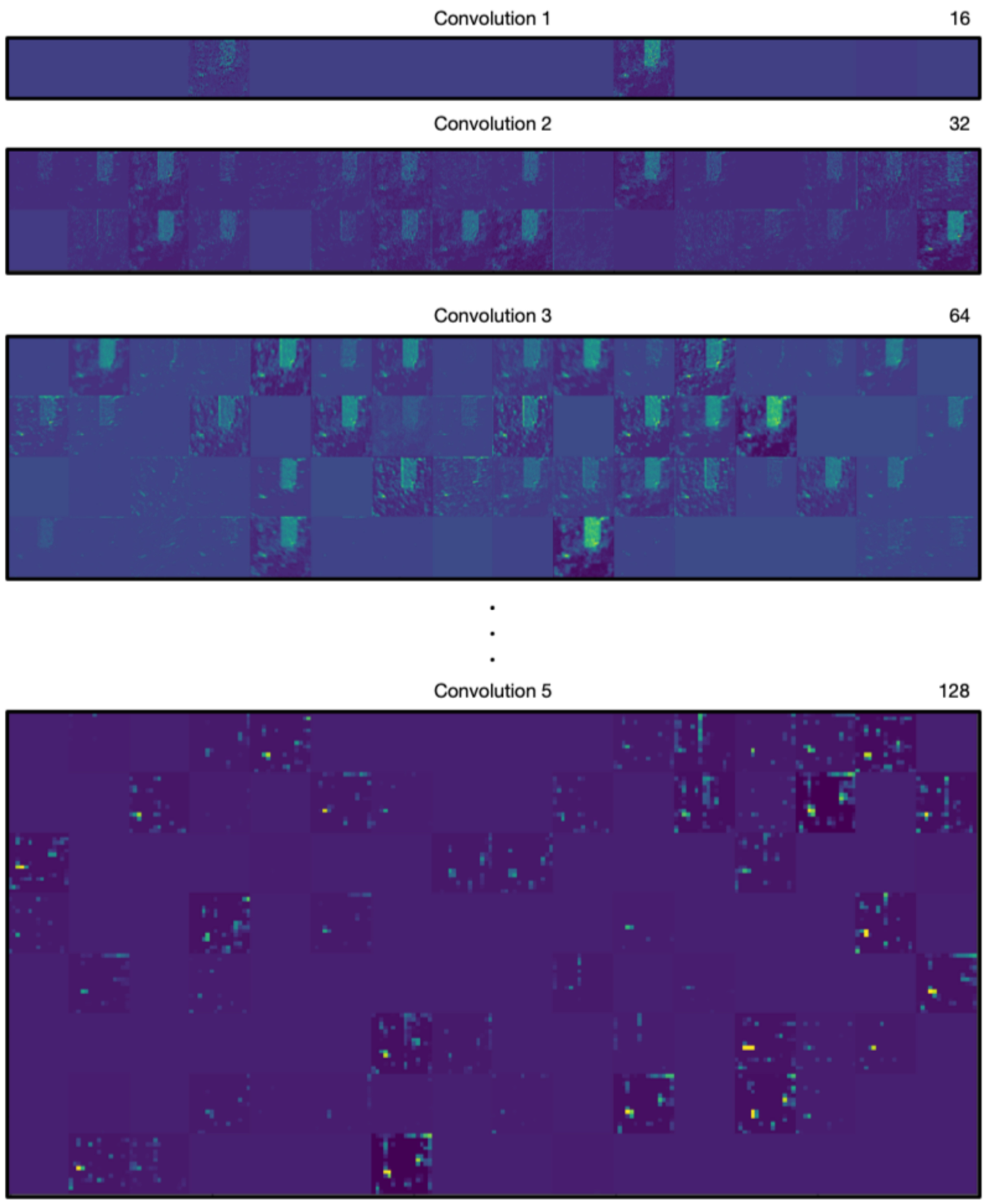


Figure 7.13: HMP CNN Activations. Visualization of the convolutional layer's activations for the HMP buccal mucosa class. The buccal mucosa class is characterized by having a significantly abundant *Streptococcus* genus, and we can see that the model is learning that in the initial convolutional layers (top) when it learns 16 and 32 filters.

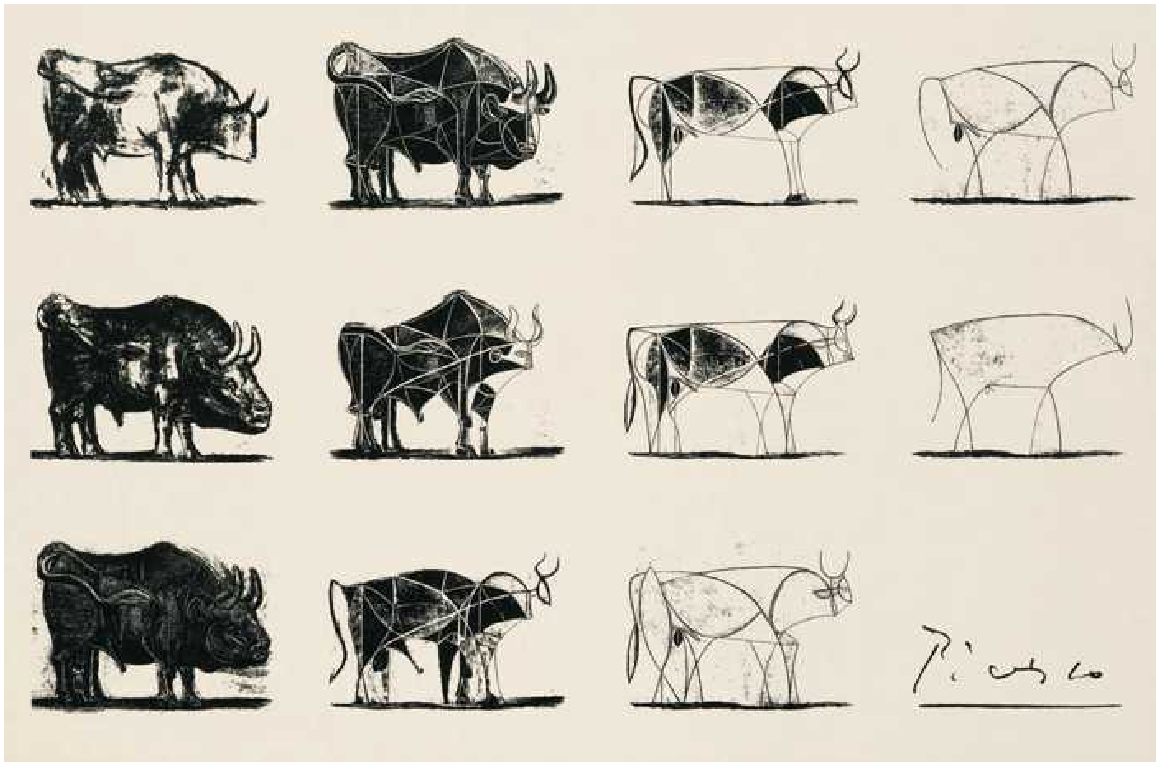


Figure 7.14: Bull Abstractions, 1945. A series of lithographs by Pablo Picasso to represent higher levels of abstractions of a bull. The images show that to represent a bull, it is not necessary to learn fine details, but sometimes, a simple set of lines suffices.
Image taken from `https://drawpaintacademy.com/the-bull`

7.4.3 Training with Chronic Kidney Disease Maps

After training the Amber CNN with the HMP/mWGS datasets, we proceeded to train with the 16S rRNA gene sequencing data for the chronic kidney disease (CKD) patients. This dataset consisted of 200 samples obtained from CKD patients of Kangwon and Seoul National University Hospitals, and comprised a total of 6 classes, (5 stages of CKD, and control). Representative microbiome map images for this dataset can be seen in Section 6.5, and they depict the community abundance profiles generated by the Kraken 2 software using their “16S” database of 5K microbial genomes. The native resolution of the images is 128×128 pixels.

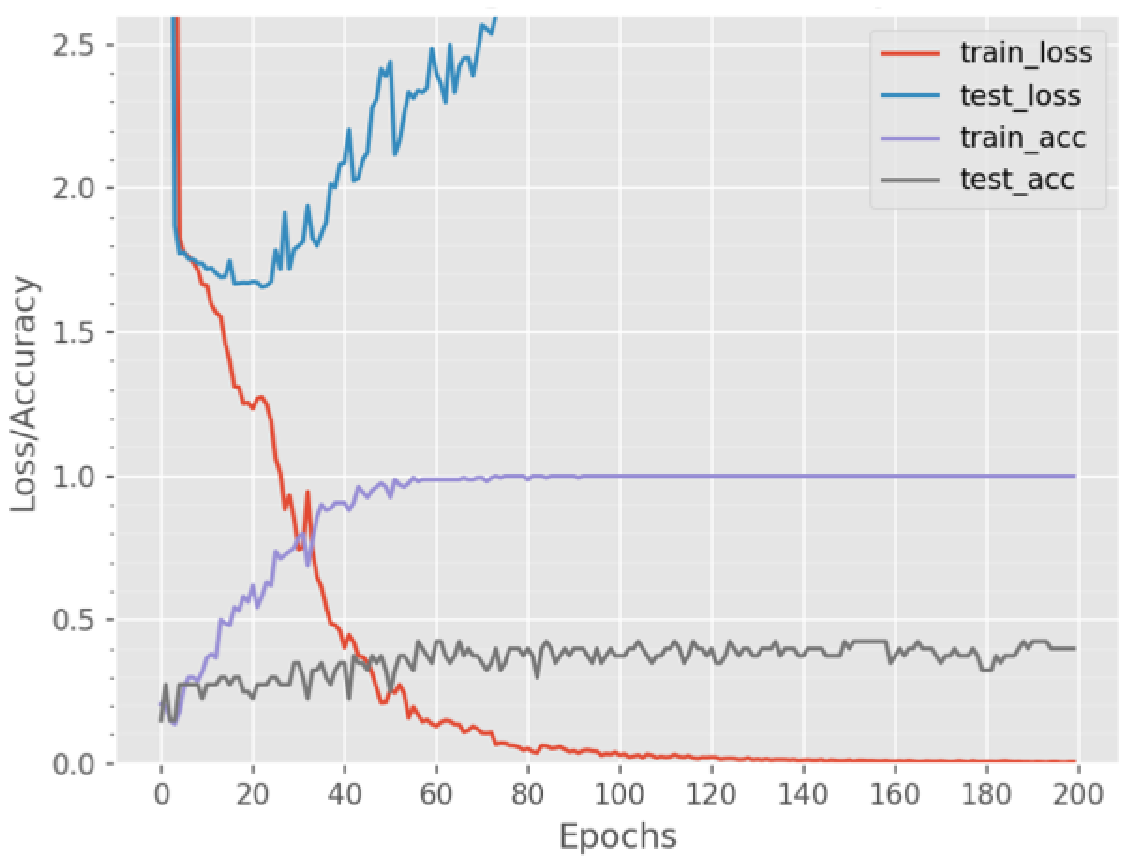


Figure 7.15: Training with CKD Data. Accuracy (purple, gray lines) and loss (red, blue lines) for training and testing the AMBER CNN with the 200 HMP samples for 200 epochs. For the CKD dataset, the training accuracy was high (99%), however, the testing accuracy was low (40%). Note the diverging paths of the training and test losses, which indicate that the model is overfitting the data substantially.

Figure 7.15 displays the accuracy and loss for training the AMBER model with the CKD dataset across 200 epochs. Note that training accuracy converges to 99% after about 80 epochs, but the testing accuracy does not match it, and it settles to 41% — this is slightly higher than the "neural net" classifier from figure 7.9 which achieved 38% testing accuracy. An issue that is present with training the model using the CKD dataset is that the model overfits the data, and this can be seen in the characteristic diverging training and testing loss lines in Figure 7.15. We can see that the loss traces start to diverge early on, and their gap grows considerably by the end of the training.

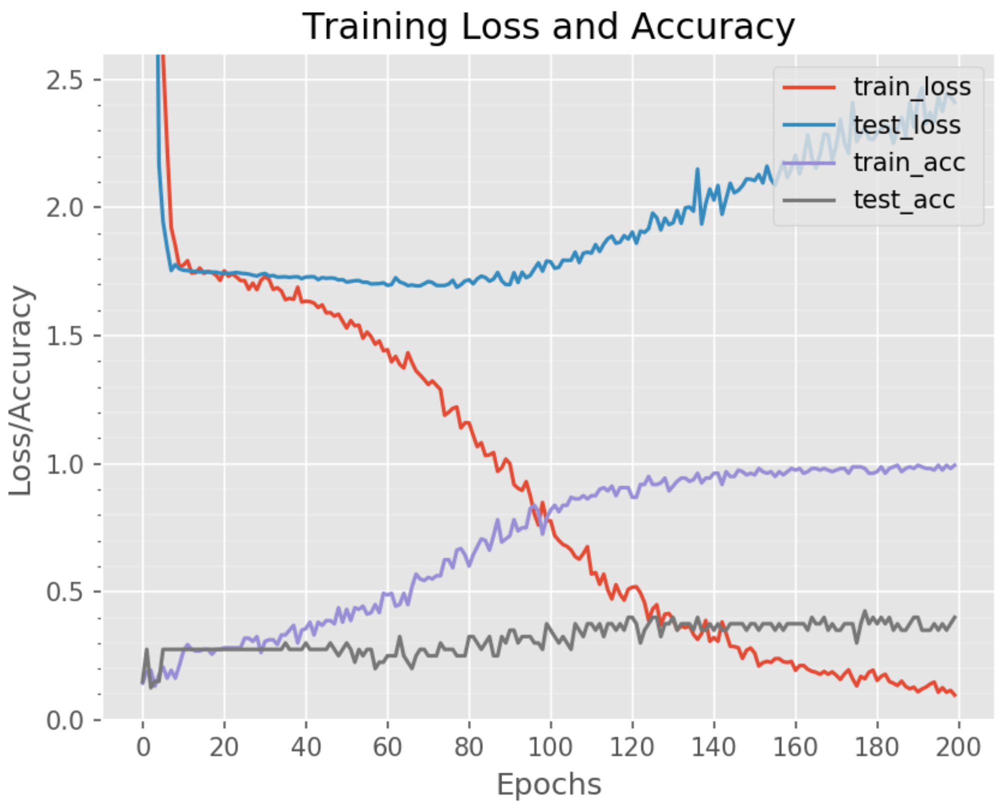


Figure 7.16: CKD Data Corrected for Overfitting. Accuracy (purple, gray lines) and loss (red, blue lines) for training and testing the AMBER CNN (with dropout layers) with 200 augmented CKD samples for 200 epochs. Training accuracy was retained at 99%, while the testing accuracy dropped slightly to 39%. Overfitting was reduced, but not completely eliminated, which suggests that more training data should be used.

We applied the same two techniques to tackle overfitting as we did with the HMP dataset, namely the use of dropout layers in the model and data augmentation for the microbiome maps. Figure 7.16 contains the updated results for training the model, and we can see that trajectories of the training and testing losses are not diverging as much as they did in Figure 7.15. The model is still overfitting the data, but not to the level it was before. Table 7.3 contains detailed performance

Class	No. Samples	Precision	Recall	F1 Score
CKD 1	27	0.25	0.17	0.20
CKD 2	31	0.44	0.73	0.55
CKD 3	28	0.75	0.50	0.60
CKD 4	28	0.25	0.17	0.20
CKD 5	31	0.17	0.20	0.18
Control	55	0.50	0.33	0.40

Table 7.3: CKD Classification Report. Evaluation of the performance of the Amber CNN model in classifying the six classes of the CKD dataset. The model performs best when discriminating among the CKD 2 and CKD 3 classes, and worst for the CKD 5 class. The "No. Samples" column contains the number of input samples for each class. The "Precision" column contains the proportion of positive classifications that was correct, the "Recall" column contains the proportion of actual positive classifications, and the "F1 Score" column contains the harmonic mean of the precision and recall columns.

measures for the model's accuracy over the six classes, and we can see that the model performs best at distinguishing between CKD 2 and CKD 3 (F1 scores of 0.55 and 0.60 respectively), and worst at distinguishing CKD 5 (F1 score of 0.18). While the dropout and data augmentation techniques helped mitigate overfitting in the HMP data, it is not helping as much in this case. A reason for this could be that the CNN model is too complex to generalize well to 16S data and its reduced number of profiled genomes. The current Amber model contains 13 million trainable parameters (Figure 7.5) and this could just prove to be an overly-complex model for training with 16S data and its smaller number of profiled genomes. A solution could be to design the Amber model to be adaptive to the number of genomes that are represented in a microbiome map: large reference collections could use the current CNN model with its 15 layers and 13M parameters, but smaller collections could use a modified CNN model with less layers and parameters.

The resulting accuracy for the CKD dataset is not as high as those achieved with the HMP dataset. We attribute this to multiple factors. The images from the

HMP dataset generated by FLINT were larger and more detailed than the ones from the CKD dataset, since each image from the HMP dataset represented abundance information from 44K strain-level genomes. In contrast, the CKD images generated by Kraken 2 were coarser and represented abundance information on only about 5K genus, or species-level, with lot less differentiation between the genomes [124]. Consequently, there are more subtle patterns in the HMP images from which the CNN could learn and exploit for its performance. We see further evidence of this claim in Figure 7.20 below, which shows that AMBER performs better with FLINTś profiles of 44K genomes than with profiles from the same samples but constructed with only 2K genomes in the reference collection.

Alternative Training Strategies

In addition to the main training schemes outlined in Sections 7.4.2 and 7.4.3, we also explored alternative strategies for training the AMBER CNN with modified HMP and CKD datasets. These alternative training strategies are described in the following subsections.

7.4.4 Training with No Borders

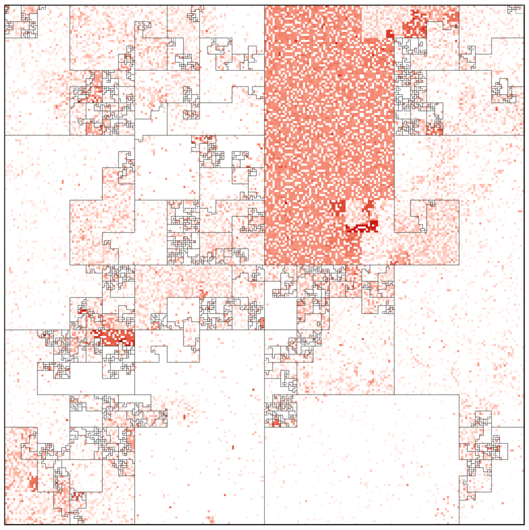
As part of the development of the AMBER CNN model, we experimented with training our network with microbiome maps that contained no visible borders around their respective neighborhoods. Our initial hypothesis was that the neighborhood demarcation lines would have a negative impact on the accuracy of our network, but as we found out, both the training and testing accuracies were much higher when we trained our network with images with borders around the neighborhoods.

7.4.5 Training with Two Datasets

In the primary training scheme for the AMBER CNN, we trained the network 7.4.2 with two independent datasets: the HMP dataset and the CKD dataset. The

CNN was trained independently with each dataset, that is, we trained and evaluated the CNN with the HMP dataset and we separately trained and evaluated the CNN with the 16S rRNA gene dataset.

We explored a scheme that combined both sets of images and trained and evaluated the CNN with both. Our hypothesis was that the increased number of images (by combining the datasets) would give us a larger dataset to train our network. Unfortunately that was not the case as both datasets are generated with different reference collections (44K for HMP, 5K for CKD) and different sequencing technologies (mWGS vs 16S). The result is that the microbiome maps for both datasets have different dimensions (256×256 for HMP/mWGS, and 128×128 for CKD/16S) which caused the CNN to not generalize well for either dataset. The best training accuracy we could achieve was 71%, with a testing accuracy of 67%. We tried to normalize both sets of images to the same dimensions (scale both datasets up to 256×256, scale down to 128×128) but it did not help increase accuracy of the model as the HMP


(a) Microbiome Map with Border.

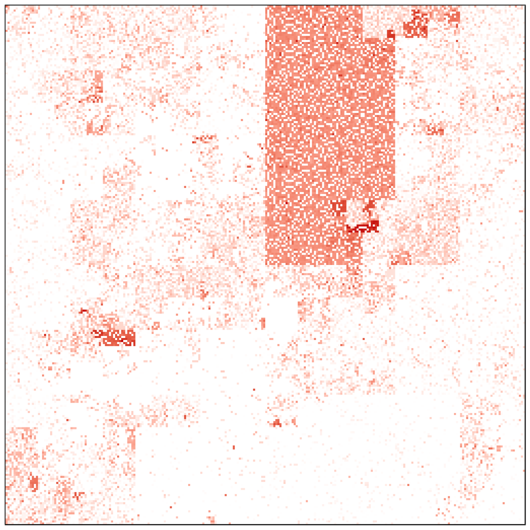

(b) Microbiome Map with No Border.

Figure 7.17: Alternative Training Images. Training and testing accuracies were much higher when training with microbiome maps with borders around neighborhoods.

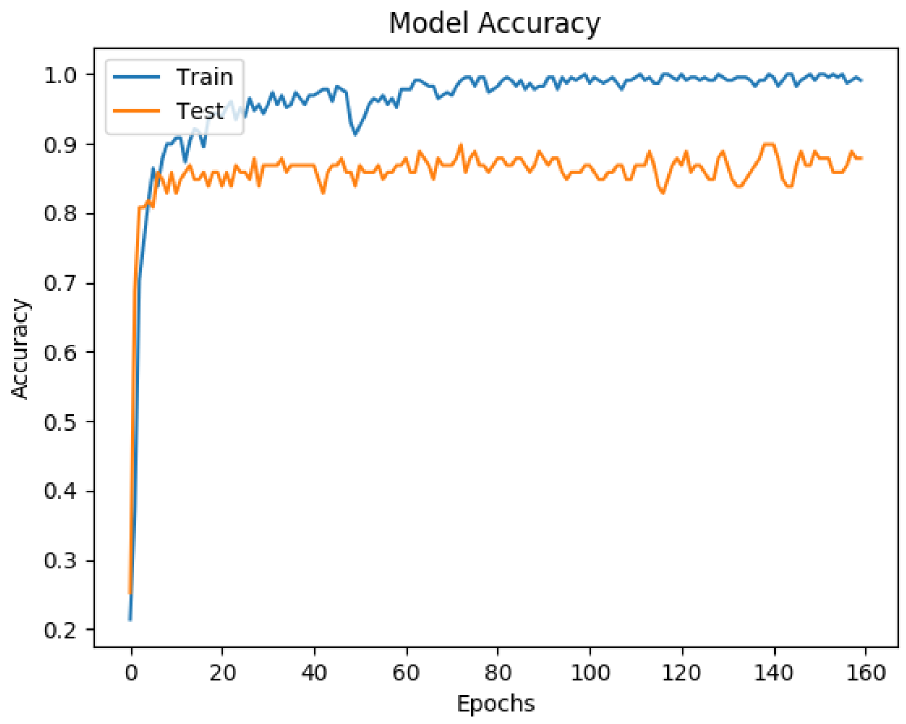


Figure 7.18: Accuracy with No Border. Training and testing accuracy when evaluation the AMBER CNN for 160 epochs with the HMP dataset, but the microbiome maps contained no visible neighborhood borders. Training accuracy finished at 94%, while testing finished at 84%.

maps were too distorted and compacted (scaling down), or the CKD maps were too blurry (scaling up).

7.5 Profile Classification Comparison

Recall that the HMP dataset is comprised of 12 classes representing the collection sites from healthy human subjects: “Attached & Keratinized Gingiva”, “Buccal Mucosa”, “Gastrointestinal Tract”, “Nares”, “Oral”, “Palatine Tonsils”, “Posterior Fornix”, “Retroauricular Crease”, “Supragingival Plaque”, “Throat”, “Tongue Dorsum”, and “Vaginal”.

We compared the profiles generated by FLINT (using 44K genomes) to those available publicly available from the HMP website. The HMP profiles are publicly

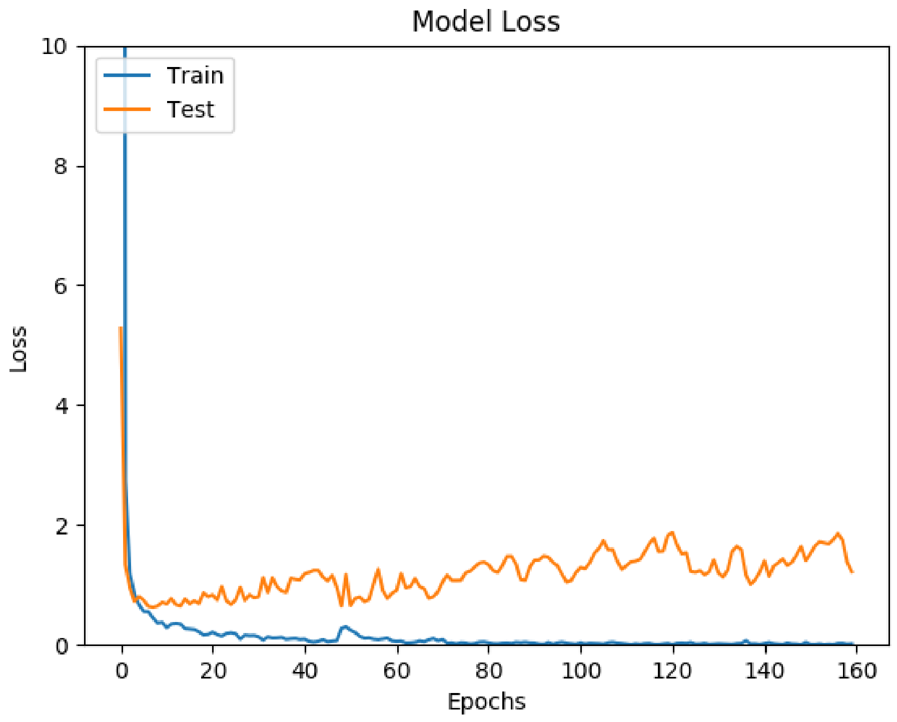


Figure 7.19: Loss with No Border. Training and testing loss when evaluation the Amber CNN for 160 epochs with the HMP dataset, but the microbiome maps contained no visible neighborhood borders.

available and were generated by profiling a custom curated reference genome database of 2K genomes. Both sets of profiles use mWGS data. Figure 7.20 depicts the results of training with the HMP profiles (orange bars) and training with profiles created with Flint (blue bars). We can see that for the tabular data (panel (A)), the results are comparable, with the HMP profiles generating better results with the "Nearest Neighbors", "SVM-Linear", and "AdaBoost" classifiers; Flint profiles generate slightly better results with "Naive Bayes", and both basically being the same for "Random Forest", and the "Neural Net". For the image data (panel (B)) the results are a little different. The results in these bars show the performance of the Amber CNN model using microbiome maps created by both Flint and the public HMP profiles using the taxonomic 5.3.1 and labeled orderings 5.3.1. We can see that the profiles created by FLINT yield a higher accuracy than those from HMP in both sets of AMBER orderings, with the AMBER taxonomic set with FLINT profiles producing a testing accuracy of 92%, compared to 82% for HMP. These results are encouraging as they demonstrate the utility of converting the numerical tabular profiles to a microbiome map, and not just using the tabular profiles directly.

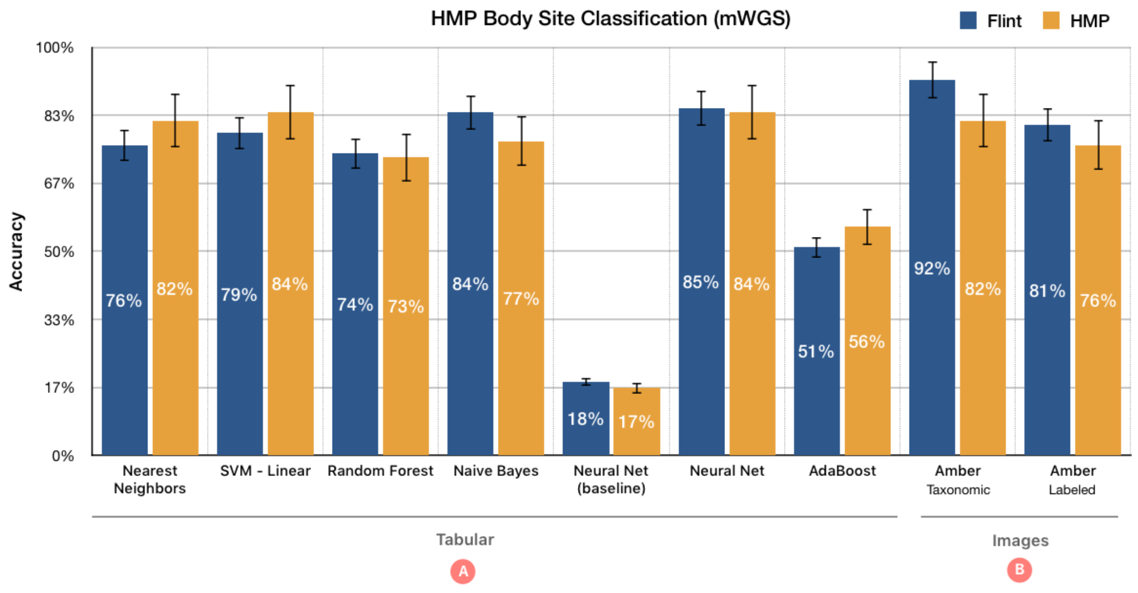


Figure 7.20: Profile Comparison. Testing accuracies for seven classifiers and the AMBER CNN model. Panel **(A)** contains seven classifiers whose input is numerical tab-separated data, and panel **(B)** contains results for the AMBER CNN model which uses microbiome map images as input. "Neural Net (baseline)" represents the simplest model for a neural network which consisted of a single 10-neuron layer trained for 10 epochs. The "Neural Net" classifier consisted of 10, 100-neuron layers trained for 1,000 epochs.

7.6 Summary & Conclusion

This chapter highlights the importance of the microbiome maps presented in Chapters 5 by showing that its value goes well beyond succinct representations and the useful applications discussed in Chapter 6. This chapter shows the enormous value of the resulting images in machine learning applications such as AMBER. This chapter shows that it is possible to leverage the information and patterns in microbiome maps to create a convolutional neural network model that is able to classify microbiome samples with high accuracy. The resulting software called AMBER performs its best when the number of classes is sufficiently large (seven or more samples per class), and also when the number of genomes that are part of the profile is large. The results of our experiments with the 16S CKD dataset once again showed that the microbiome maps lead to machine learning models with higher accuracies. However, the experiments with 16S datasets suggested a lot of room for improvement because the accuracy was much lower than the ones achieved with wMGS data.

The AMBER CNN model continues to undergo fine-tuning and improvements. The source code, along with all training datasets and resources, will be made available at the project's website at
`http://biorg.cs.fiu.edu/amber`.

CHAPTER 8

CONCLUSION

The analysis of massive metagenomic datasets poses a computational challenge because (*i*) current tools generate large intermediate results and (*ii*) require the creation of enormous indexes to catalog the ever increasing sets of reference genome collections [156]. New paradigms are needed that go beyond exploiting basic parallelism in computations, that leverage modern streaming architectures for processing large amounts of data, and that use cloud-based distributed computational pipelines. Such computational strategies must be matched with visualization strategies to create interpretable results from all the data that low-cost DNA sequencing is generating. Finally, sophisticated machine learning techniques can be fine-tuned and engineered to learn more patterns from the metagenomic datasets and microbiome maps than has been achieved so far.

8.1 Microbial Community Profiling in the Cloud

Microbial community profiling for mWGS datasets usually starts by aligning the sequenced DNA reads to a collection of microbial reference genomes. Current profiling tools are designed to work against a small representative collection of microbial genomes, and do not scale well to larger genome collections. However, large reference genome collections are capable of providing a more complete and accurate profile of the bacterial population in a microbiome sample.

The Flint system outlined in Chapter 4 address the computational challenge by showing how large metagenomic datasets can be profiled against a large collection of reference bacterial genomes in a fast and economical way. Our implementation relies on the freely available Spark framework to distribute the alignment of millions of sequencing reads against Ensembl's collection of 44K bacterial genomes. The reference genomes are partitioned in order to distribute the genome sequences across worker machines, and this allows us to use large collections of reference sequences. By using the well-known Bowtie2 aligner under the hood in the worker-nodes, we are able to maintain fast alignment rates, without loss of accuracy.

To date, profiling metagenomic samples against thousands of reference genomes has not been possible for research groups with access to modest computing resources. This is due to the size of the reference genomes and the financial costs of the computing resources necessary to employ them. By using distributed frameworks such as Spark, along with affordable cloud computing services such as Amazon's EMR, we are able to distribute a large collection of reference genomes (totaling 170 GB of reference sequence, and 4.6 million assembly FASTA files) and use a MapReduce strategy to profile millions of metagenomic sequencing reads against them in a matter of hours, and at minimal financial costs, thus bringing sophisticated metagenomic analyses within reach of small research groups with modest resources and achieving a small measure of much-needed democratization in metagenomics.

FLINT is open source software written in Python and available under the MIT License (MIT). The source code can be obtained at the GitHub repository which includes instructions and documentation on provisioning an EMR cluster, deploying the necessary partitioned reference genome indices into worker nodes, and launching an analysis job. Links to additional materials, simulation datasets, source code, and partitioned reference indices can be found on the FLINT project website at
`http://biorg.cs.fiu.edu/`.

8.2 Visualizations of Microbial Community Profiles

Microbial community profiles from mWGS datasets synthesize information from billions of sequenced DNA reads coming from the genomes of the thousands of microbes present in a microbiome. Analyzing and understanding these profiles can be a challenge since the data they represent is complex. Particularly challenging is their visualization and interpretability, as existing techniques are inadequate when the number of identified taxa is in the thousands. The JASPER system (Chapters 5 and 6) shows how the Hilbert curve visualization technique can be used to visualize metagenomic community abundance profiles from both mWGS and 16S DNA sequencing datasets. The resulting "microbiome maps" display the relative abundance of microbial genomes in a succinct, intuitive and interpretable manner, and can communicate multiple latent factors of the reference genomes in the samples under study.

The Hilbert curve is used to lay out the microbes from the reference database in two different ordering schemes that can be used to draw a microbiome map. The first is the *taxonomic ordering*, which relies on taxonomy information from the Ensembl Genomes database, and can be used to create maps that express abundance values in the context of the taxonomic clades that the microbial genomes belong to. The second is the *labeled ordering*, which is dependent on a user-specified labeling of biological conditions for each sample, and can express the abundance values of the profile in the context of a biological interpretation for a cohort of samples.

JASPER is accessible through the command line, but we are working on creating a full graphical user interface (GUI) that will allow users to explore and immerse themselves in the maps by clicking on microbial neighborhoods that will bring a neighborhood into full view, allowing users to inspect individual members of the microbiome in greater detail. This envisions a *Google Maps* for *microbiomes*. JASPER is open source software written in Python and R, and uses OpenCV [23], hilbert curve [4], and HilbertCurve (R) [59]. It is available under a GPL 3 license, and obtained from `biorg.cs.fiu.edu/jasper`, where additional materials are also available.

8.3 Characterizing Microbiomes

The microbiome maps generated with the technique from Chapter 5 are a powerful visualization tool for representing the dynamicity of microbiome samples: they can display thousands of microbial genomes and their corresponding relative abundances, while at the same time embedding the context of microbial relationships and their taxonomies under multiple biological conditions. The embedding of this information creates distinctive visual patterns in a microbiome map that could be used to uniquely identify the biological conditions for the sample from which the map originated. Tha maps could also be used to identify the microbial taxonomic clades dominant in a set of samples.

Chapter 7 addresses the problem of characterizing microbiomes using microbiome maps created from the microbial community abundance profiles generated with DNA sequencing data from whole-genome and 16S rRNA gene sequencing technologies. We addressed the problem of developing and training convolutional neural networks (CNNs) [89] directly with microbiome maps. CNNs are a type of artificial neural network (ANN) useful for characterization of cohorts of inputs.

The CNN architecture in the Amber framework is designed specifically for the characterization of microbiome maps generated from microbiome sequencing data. The framework takes advantage of a CNN's established ability to segment and classify images. The framework automatically takes advantage of patterns in the images, which appear in the images because of the proximity of related taxonomic groups in the various microbial neighborhoods of a microbiome map.

The Amber framework is available under the GNU public license (version 3.0), and is available to download at the project's url: `biorg.cs.fiu.edu/amber`, which also contains links to the GitHub source.

8.4 Future Work

8.4.1 The Future of Flint

The Flint system described in Chapter 4 outlines a fast and efficient software framework for creating microbial community profiles. The output of the system is a set of abundance profiles for each sample given as input. An enhancement for the system that we have explored is the real-time reporting of the abundance values as they are being generated. Such a system would allow users to see microbial community profiles in real time. A pipeline currently exists for this type of system with the use of data warehousing solutions such as Amazon RedShift [7]. The RedShift service is a cloud service for maintaining data warehouses that can easily connect to client applications such as visual analytics tools like Tableau [141]. The Flint system works by accumulating microbial abundances from worker machines, and we could modify it to also send data to a RedShift instance; a client machine such as a desktop or laptop computer could run Tableau at the same time, and use a data connector to connect to the RedShift instance that is receiving data from Flint. By doing the above, the Tableau tool would be able to visualize the community profile in real-time as its being generated, and this would create a powerful visualization and reporting solution for researchers in the field.

A Fleet of Cooperating Clusters

The Flint system (Chapter 4) describes a single cluster of commodity machines running in the cloud. An exciting extension of the system would be to create a fleet (pipeline or network) of clusters: one could deploy a task force of multiple clusters, each working on a different task, but cooperating to analyze multiple datasets and synchronizing their output. The system could activate clusters as they are needed, and decommission them after they are no longer needed. The fleet would not have to be composed of the same types of cluster, as different roles could be assigned to different clusters. Some clusters could be dedicated to performing abundance profiling tasks such as the current Flint cluster, but others could be provisioned to support other types of tasks such as annotation and visualization (similar to the PlotTwist web app [53], but built on a cloud cluster and not as a Shiny RStudio app [129]). Another interesting role for clusters in the fleet could be that of knowledge-guided analysis clusters – a take on the platform provided by KnowEnG [21]; in this role, clusters in the fleet could be assigned a task that relies on prior knowledge to enhance the analysis based on a pre-existing knowledge bank of microbial signatures [159], gene-set enrichment [139], and functional annotations [88].

Sequencing Instrument Integration

The current implementation of Flint assumes that the input files are stored in a distributed file system such as Amazon's S3 [8] or Hadoop's HDFS [143]. The files are assumed to be a collection of pre-processed sequencing read that were created by a high-throughput sequencing instrument such as an Illumina Hi-Seq instrument [74]. An extension to the Flint system could be a instrument plug-in that streams the reads as soon as they are created inside the instrument; this could create a highly automated and integrated system that is constantly producing and analyzing data.

8.4.2 Peering Down the Road for Jasper

The Jasper system from Chapter 5 creates microbiome maps as 2D image representations of microbial community profiles. As part of the first release of Jasper we released a basic image inspector that users can use to inspect the microbiome maps they create: users can load an image alongside a set of annotations, and the inspector tool will allow them to use a computer-mouse to point to specific locations in the image and have the tool display the name of the microbial genome that they clicked on. The tool is a proof of concept and a natural next-step is to create a fully featured tool with a graphical user interface. We envision a tool that will allow users to not only click on a region of the image and get feedback, but also to mouse-over regions to get quick visual feedback, as well as to allow them to zoom into particular microbial neighborhoods — a behavior similar to that of a consumer mapping application like Google Maps [56], and a visual "DIFF" tool like Kaleidoscope [73].

The tool could also have a back-end connected to multiple annotation providers, API servers, and online resources. Similar to the proposed connections of prior-knowledge resources in the FLINT fleet, the JASPER toolkit could integrate with a data source of microbial signatures [159], gene-set enrichment collections [139], and functional annotations [88], but also contain additional sources for inspecting individual genomes in details such as connections to organism-specific databases such as the `pseudomonas.com` database [164].

As a visual tool, the JASPER toolkit could facilitate context-switching for users so that they can analyze their microbiome maps through different factors. This could be done by giving users the ability to switch between different orderings with the click of a button, and the mechanism would give users the ability to apply different orderings and not have to decide between one or the other: a window could be implemented that displays two orders side-by-side (similar to Figure 8.2), and give the user the ability to mark a microbial strain in one order and have it highlighted in the other.

Having an interactive visual tool would open the possibilities for exploring more immersive experiences for creating Hilbert curves in higher dimensions. We could start by generalizing the Hilbert curve to three dimensions, so that we construct a space filling curve to occupy the unit cube, and not the unit square. This would give us an extra dimension to use in the building of our microbiome maps, and we could construct *Microbiome Cubes* that contain an additional layer of information. Time could be used as the extra dimension such that the same microbial strain occupies the a sub-volume of the cube, but the values at each vertex point in the 3D folding of the curve contain the abundance values of that strain across a time-series analysis. Having a 3D *Microbiome Cube* could also enhance the Amber framework as we could use 3D convolutions in the model.

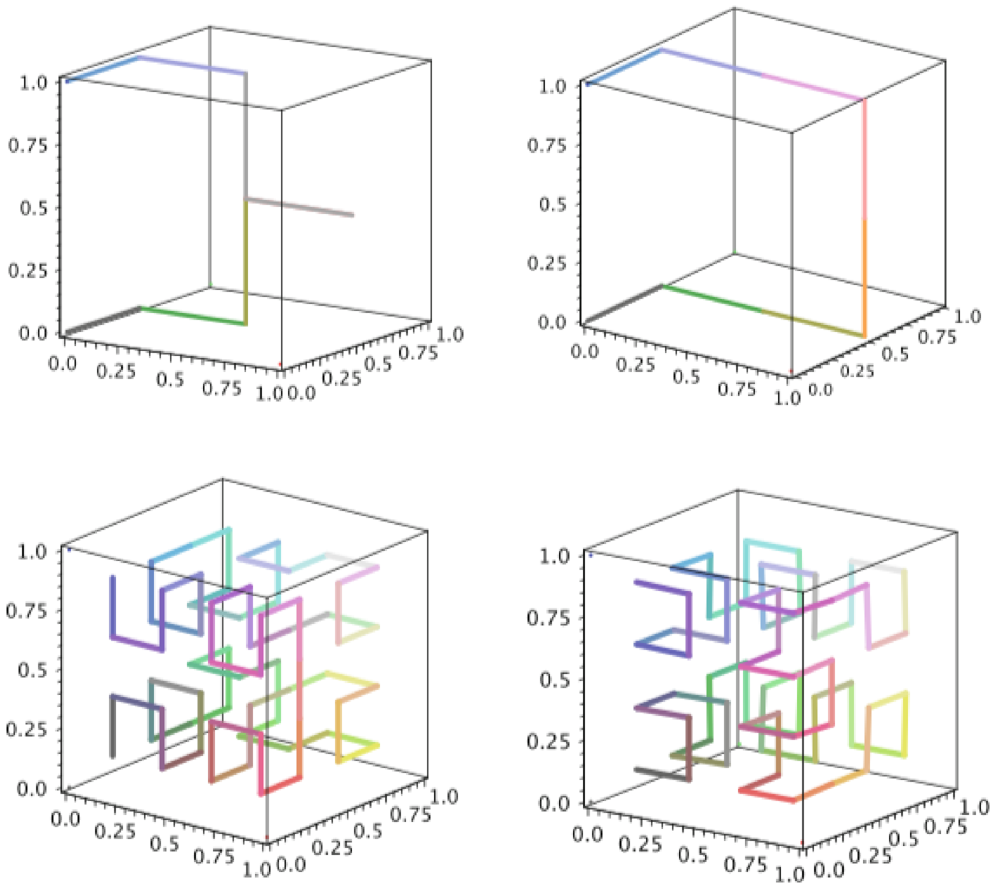


Figure 8.1: A 3D Hilbert Curve. The top two volumes represent two variants of the first iteration of the curve in 3D space, while the bottom two volumes represent the second iteration of the top volume above it. Figure 8.2 from the book by Michael Bader, "Space-Filling Curves" [16] and reproduced with permission. Copyright 2016, Springer.

Kaleidoscope is a visual "DIFF" tool that can identify differences between two text files, two images, and two directories. We present it here as a example of what a GUI Jasper toolkit app can be, and Figures 8.2 through 8.5 contain a set of microbiome map images that have been loaded into it. The Kaleidoscope tool allows for a visual inspection of two images side by side, but we see a JASPER toolkit app that can interactively do what Kaleidoscope is doing: zoom into the images and clicking on a pixel to get an inspector with copious details about the genome that was clicked. The inspector would contain data from multiple sources, and could present the user with data about the microbe's pathogenicity, shape, taxonomic lineage, metabolism, etc. The screenshots presented in Figures 8.2 through 8.5 are used as an example of what the interactivity model for the JASPER toolkit could be like, and while Kaleidoscope is great to prototype such a tool, it was not meant to explore microbiome maps as it is just a "DIFF" tool designed for tracking changes (but it does give us a glimpse of what could be built).

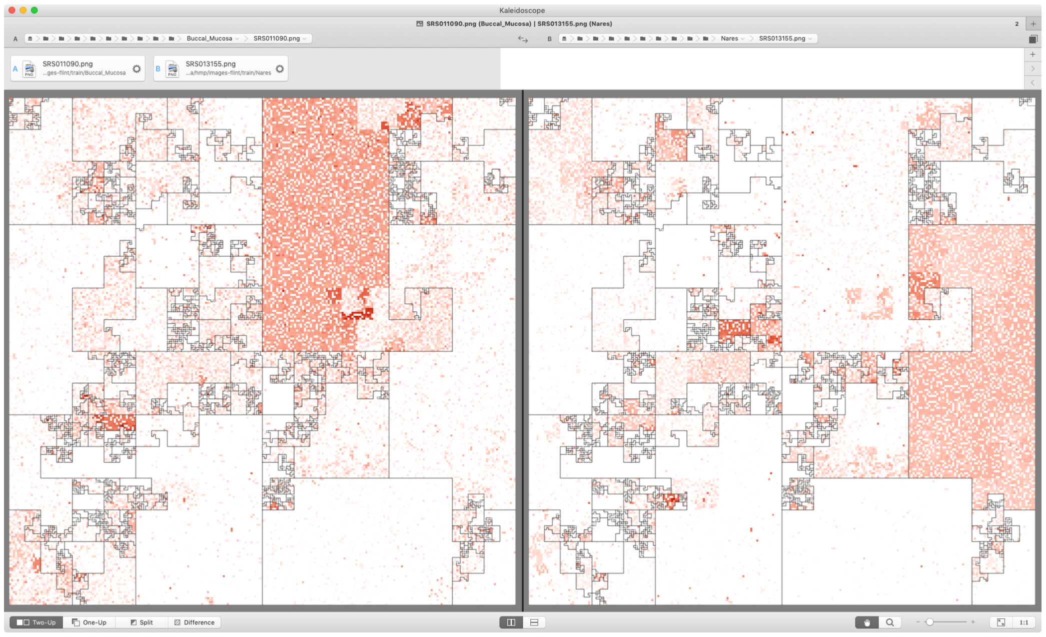

Kaleidoscope
SRS011090.png (Buccal_Mucosa) | SRS013155.png (Nares)
Buccal_Mucosa
SRS011090.png
Nares
SRS013155.png
SRS011090.png
...ges-flint/train/Buccal_Mucosa
SRS013155.png
...a/hmp/images-flint/train/Nares
Two-Up
One-Up
Split
Difference
1:1

Figure 8.2: Two Maps Side by Side. A visual DIFF of two microbiome maps for a HMP buccal mucosa (left) and anterior nares (right) samples.

Exploring Optimal Orderings

Establishing a linear order for a tree structure is an important step for setting the order that microbial strains are laid out in the microbiome map using the Hilbert

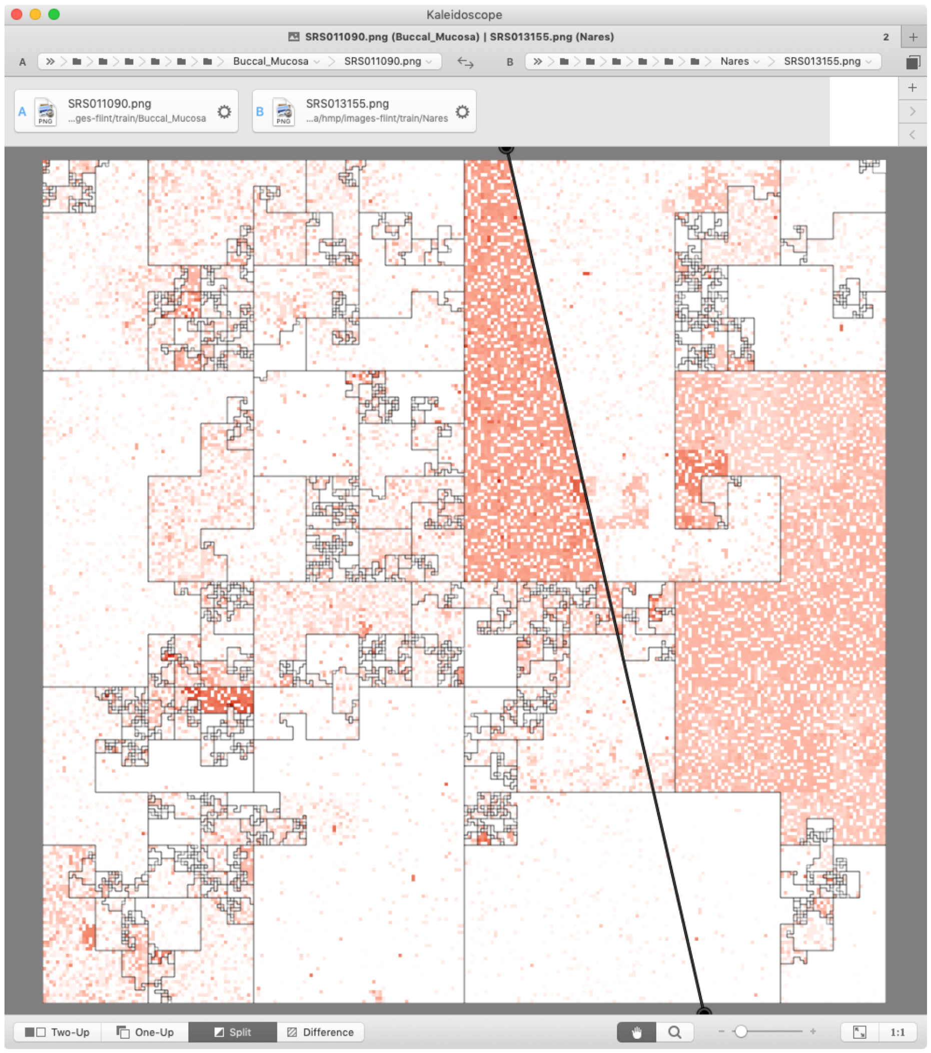

Kaleidoscope
SRS011090.png (Buccal_Mucosa) | SRS013155.png (Nares)
Buccal_Mucosa
SRS011090.png
Nares
SRS013155.png
SRS011090.png
...ges-flint/train/Buccal_Mucosa
SRS013155.png
...a/hmp/images-flint/train/Nares
Two-Up
One-Up
Split
Difference
1:1

Figure 8.3: Difference Slices. Overlaid differences in Kaleidoscope of two microbiome maps for a HMP buccal mucosa (left) and anterior nares (right) samples. The two images are layered on top of each other, and the pendulum in the middle (black line) allows the user to see what is "behind" (right image) and "above" (left image) the two images.

curve. The set of terminal nodes in the taxonomic tree (microbial strains) do not have a "start" or "finish", nor does it have a "right" side or a "left" side. Different orderings of this set can result by performing a permutation of tree's nodes, or of any node, at any given level of the taxonomic tree. Algorithms for finding an optimal order have been proposed such as the one described in [17], but they rely on the tree having a certain property such as it being a binary tree, and for a distance measure

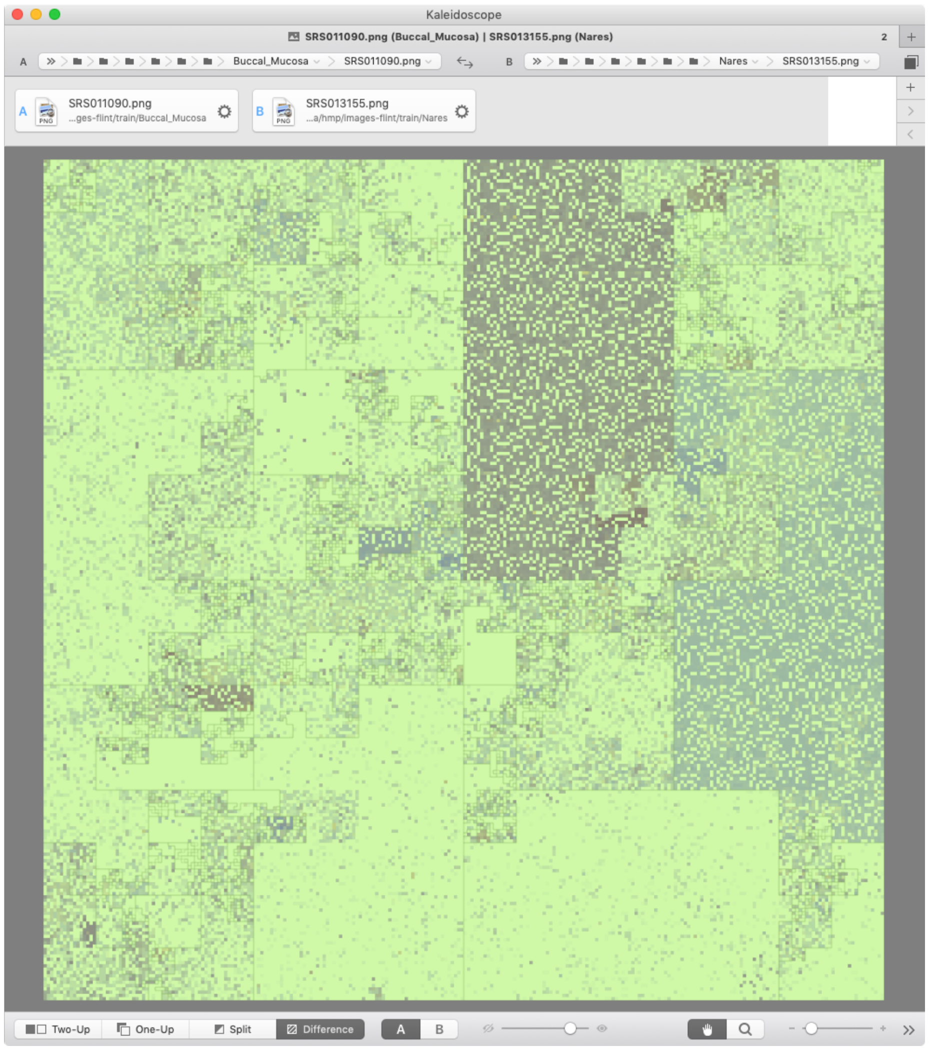

Kaleidoscope
SRS011090.png (Buccal_Mucosa) | SRS013155.png (Nares)
Buccal_Mucosa
SRS011090.png
Nares
SRS013155.png
SRS011090.png
...ges-flint/train/Buccal_Mucosa
SRS013155.png
...a/hmp/images-flint/train/Nares
Two-Up
One-Up
Split
Difference
A
B

Figure 8.4: Color Difference. A Kaleidoscope color difference image of two microbiome maps for a HMP buccal mucosa (left) and anterior nares (right) samples. The two images are layered on top of each other two display what is common on both images (green) and what is different (dark areas), with red-tinted areas being what is unique to the left image, and blue-tinted images being what is unique to the right image.

to be calculated. We can continue to explore this space in future work, and look into ways of representing taxonomic trees as binary trees such that we can establish a linear order.

Recall that in the labeled ordering 5.3.1, each taxon is assigned a condition that best "represents" it (a condition given by the user). A taxon is grouped under con-

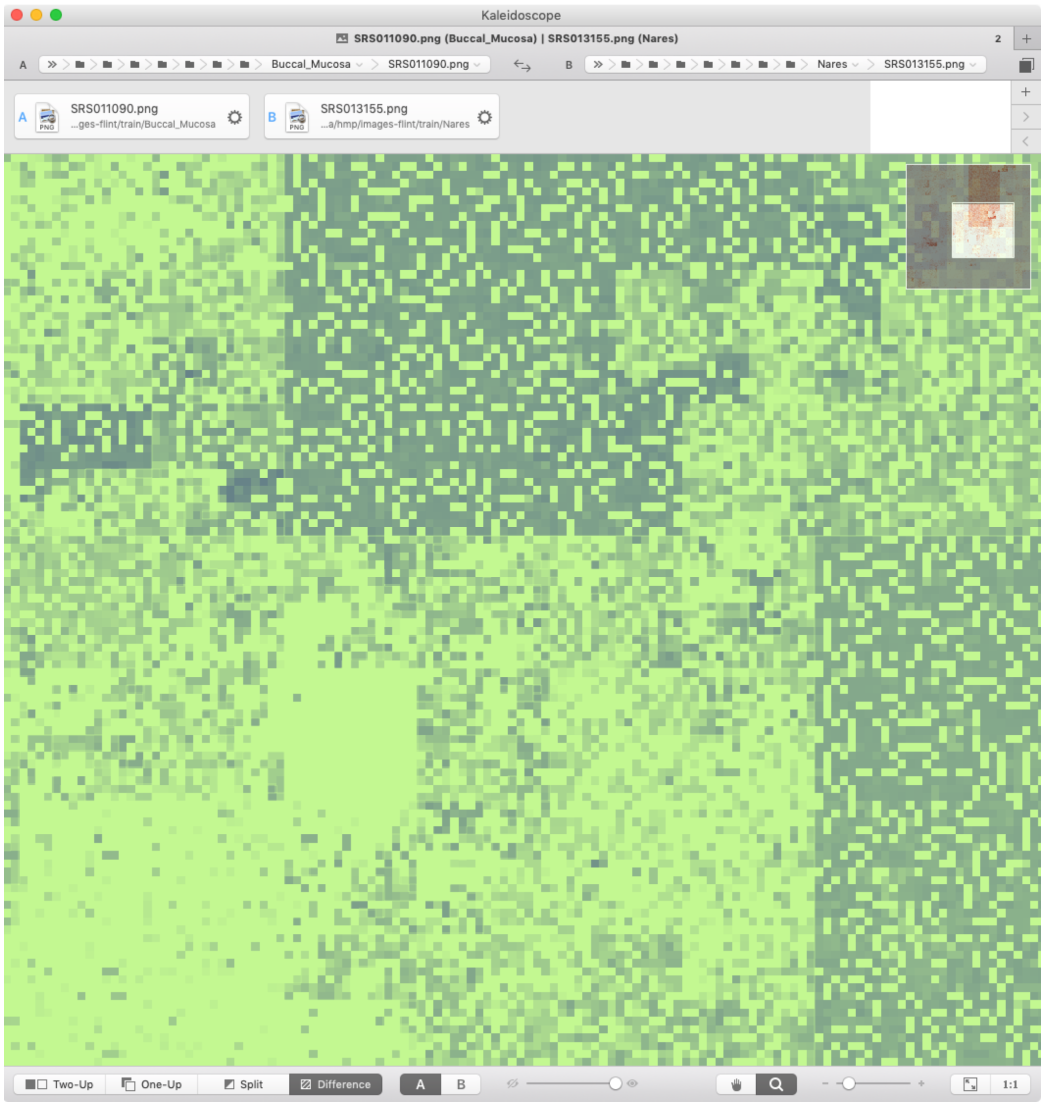

Kaleidoscope
SRS011090.png (Buccal_Mucosa) | SRS013155.png (Nares)
Buccal_Mucosa
SRS011090.png
Nares
SRS013155.png
SRS011090.png
...ges-flint/train/Buccal_Mucosa
SRS013155.png
...a/hmp/images-flint/train/Nares
Two-Up
One-Up
Split
Difference
A
B
1:1

Figure 8.5: Zoom. Kaleidoscope can zoom into the difference image from Figure 8.4, and users can navigate by using the navigation map in the top-right.

dition i if its average abundance is highest in the group of sample corresponding to condition i. One issue with this construction is that there are taxa that should belong to another condition on account of being "lowest" for that condition, or that "should" belong to another condition owing to them being prominent biomarkers for that condition. More work here is needed to allow for representative taxa to be anchored to a user-specified condition, or to a condition that is supported by the literature —

this could be achieved by automatically connecting JASPER to a reference annotation authority database that contains such information.

8.4.3 Variations on the Theme of AMBER

Finally, the AMBER framework from Chapter 7 is a machine learning framework for characterizing microbiomes using microbiome maps. The model proves quite effective at discriminating the HMP body sites, and we show that it outperforms other models even when we select a smaller number of genomes as in Section 7.4.2. The model does not perform as well with the smaller 16S CKD dataset 7.4.3, and part of the future work for this project is to create an adaptive model that takes into account the number of genomes that are being profiled. The current model architecture in Section 7.3.2 contains a lot of convolutional-pooling layer pairs that might not be suitable for abundance profiles with a smaller amount of profiled genomes, and an enhancement to the implementation would be a architecture that grows and shrinks depending on the size of the profile (the number of genomes), so that profiles with a smaller set of genomes would use a less-complex model, i.e., a model with a smaller set of convolutional-pooling layer pairs.

We also want to explore the area of data augmentation for time-series analyses by means of generative adversarial nets (GAN) [55]. In some time-series analyses, data points are not able to be collected, and by training a GAN with a set of related microbiome maps, we could explore how to create completely synthetic samples to interpolate between two time points. A GAN could also be used as the basis for a simulator that generates simulated community profiles for benchmarking purposes, a valuable tool that could be useful for researchers doing tool development.

[1] Kjersti Aagaard, Jun Ma, Kathleen M. Antony, Radhika Ganu, Joseph Petrosino, and James Versalovic. The Placenta Harbors a Unique Microbiome. *Science Translational Medicine*, 6(237):237ra65–237ra65, May 2014.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software available from tensorflow.org.

[3] Jennifer Ackerman. The Ultimate Social Network. *Scientific American*, 306(6):36–43, June 2012.

[4] Gabriel Altay. The Hilbert Curve Python package. `https://pypi.org/project/hilbertcurve`. Accessed: 2020-01-25.

[5] Amazon Web Services. Amazon Deep Learning AMI. `https://aws.amazon.com/machine-learning/amis`. Accessed: 2020-02-17.

[6] Amazon Web Services. Amazon EMR. `https://aws.amazon.com/emr`. Accessed: 2018-11-16.

[7] Amazon Web Services. Amazon RedShift. `https://aws.amazon.com/redshift`. Accessed: 2020-02-18.

[8] Amazon Web Services. Amazon Simple Storage Service, S3. `https://aws.amazon.com/s3`. Accessed: 2018-11-16.

[9] Amazon Web Services. Amazon Web Services. `https://aws.amazon.com`. Accessed: 2019-11-17.

[10] Amazon Web Services. EC2 Spot Market. `https://aws.amazon.com/ec2/spot`. Accessed: 2018-11-16.

[11] Amazon.com Inc. Amazon Web Services. `https://aws.amazon.com`. Accessed: 2018-10-17.

[12] Simon Anders. Visualization of Genomic Data with the Hilbert Curve. *Bioinformatics (Oxford, England)*, 25(10):1231–1235, May 2009.

[13] Wilhelm J Ansorge. Next-generation DNA Sequencing Techniques. *New biotechnology*, 25(4):195–203, April 2009.

[14] Genta Aoki and Yasubumi Sakakibara. Convolutional Neural Networks for Classification of Alignments of Non-Coding RNA Sequences. *Bioinformatics (Oxford, England)*, 34(13):i237–i244, July 2018.

[15] Alla Aroutcheva, Jose Simoes, Kian Behbakht, and Sebastian Faro. *Gardnerella vaginalis* Isolated from Patients with Bacterial Vaginosis and from Patients with Healthy Vaginal Ecosystems. *Clinical Infectious Diseases*, 33(7):1022–1027, 2001.

[16] Michael Bader. *Space-Filling Curves: An Introduction with Applications in Scientific Computing.* Springer Publishing Company, Incorporated, 2012.

[17] Ziv Bar-Joseph, David K Gifford, and Tommi S Jaakkola. Fast Optimal Leaf Ordering for Hierarchical Clustering. *Bioinformatics*, 17(suppl_1):S22–S29, 2001.

[18] John J. Bartholdi and Loren K Platzman. Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space. *Management Science*, 34(3):291–305, March 1988.

[19] Roeland Berendsen, Corne Pieterse, and Peter Bakker. The Rhizosphere Microbiome and Plant Health. *Trends in Plant Science*, 17(8):478–486, 2012.

[20] Gabriele Berg, Armin Erlacher, and Martin Grube. The Edible Plant Microbiome: Importance and Health Issues. *Principles of Plant-Microbe Interactions*, 1(1):419–426, 2014.

[21] Charles Blatti III, Amin Emad, Matthew J Berry, Lisa Gatzke, Milt Epstein, Daniel Lanier, Pramod Rizal, Jing Ge, Xiaoxia Liao, Omar Sobh, Mike Lambert, Corey S Post, Jinfeng Xiao, Peter Groves, Aidan T Epstein, Xi Chen, Subhashini Srinivasan, Erik Lehnert, Krishna R Kalari, Liewei Wang, Richard M Weinshilboum, Jun S Song, C Victor Jongeneel, Jiawei Han, Umberto Ravaioli, Nahil Sobh, Colleen B Bushell, and Saurabh Sinha. Knowledge-guided Analysis of "Omics" Data using the KnowEnG Cloud Platform. *PLoS biology*, 18(1):e3000583, January 2020.

[22] Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. GAN Augmentation: Augmenting Training Data using Generative Adversarial Networks. *ArXiv*, October 2018.

[23] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[24] Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal Probabilistic RNA-seq Quantification. *Nature biotechnology*, 34(5):525–527, May 2016.

[25] M. Luz Calle. Statistical Analysis of Metagenomics Data. *Genomics & Informatics*, 17(1), March 2019.

[26] Gregory Caporaso, Christian L Lauber, William A Walters, Donna Berg-Lyons, James Huntley, Noah Fierer, Sarah M Owens, Jason Betley, Louise Fraser, Markus Bauer, Niall Gormley, Jack A Gilbert, Geoff Smith, and Rob Knight. Ultra-high-throughput Microbial Community Analysis on the Illumina HiSeq and MiSeq Platforms. *The ISME journal*, 6(8):1621–1624, March 2012.

[27] Giuseppe Cattaneo, Raffaele Giancarlo, Stefano Piotto, Umberto Ferraro Petrillo, Gianluca Roscigno, and Luigi Di Biasi. MapReduce in Computational Biology - A Synopsis. In *Advances in Artificial Life, Evolutionary Computation, and Systems Chemistry*, pages 53–64. Springer, Cham, October 2016.

[28] Aleksandra Checinska, Alexander J. Probst, Parag Vaishampayan, James R. White, Deepika Kumar, Victor G. Stepanov, George E. Fox, Henrik R. Nilsson, Duane L. Pierson, Jay Perry, and Kasthuri Venkateswaran. Microbiomes of the Dust Particles Collected from the International Space Station and Spacecraft Assembly Facilities. *BMC Microbiome*, 1(1), 2015.

[29] T Chen, Z Chen, Q Shi, and X Huang. Road Marking Detection and Classification Using Machine Learning Algorithms. *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 617–621, 2015.

[30] Junghee Cho, Junseok Kwon, and Byung-Woo Hong. Adaptive Regularization via Residual Smoothing in Deep Learning Optimization. *IEEE Access*, 2019.

[31] Francois Chollet. *Deep Learning with Python*. Manning Publications Co., USA, 1st edition, 2017.

[32] James R Cole et al. Ribosomal Database Project - Data and Tools for High Throughput rRNA Analysis. *Nucleic Acids Research*, 42(D1):D633–D642, 2014.

[33] The Integrative HMP iHMP Research Network Consortium. The Integrative Human Microbiome Project: Dynamic Analysis of Microbiome-Host Omics Profiles during Periods of Human Health and Disease. *Cell Host & Microbe*, 16(3):276–289, September 2014.

[34] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, 51(1):107–113, January 2008.

[35] Xuegong Deng, Xuemei Deng, Simon Rayner, Xiangdong Liu, Qingling Zhang, Yupu Yang, and Ning Li. DHPC: A New Tool to Express Genome Structural Features. *Genomics*, 91(5):476 – 483, 2008.

[36] Elif Derya Übeyli. Recurrent Neural Networks Employing Lyapunov Exponents for Analysis of ECG Signals. *Expert Systems with Applications*, 37(2):1192–1199, March 2010.

[37] Todd DeSantis, Hugenholtz Philip, Neils Larsen, Mark Rojas, Eoin Brodie, Keith Keller, Thomas Huber, Daniel Dalevi, Ping Hu, and Gary Andersen.

Greengenes, a Chimera-checked 16S rRNA Gene Database and Workbench Compatible with ARB. *Applied and Environmental Microbiology*, 72(7):5069–5072, 2006.

[38] Joseph C Devlin, Thomas Battaglia, Martin J Blaser, and Kelly V Ruggles. WHAM!: A Web-based Visualization Suite for User-defined Analysis of Metagenomic Shotgun Sequencing Data. *BMC genomics*, 19(1):493, June 2018.

[39] Sorin Drăghici. *Data Analysis Tools for DNA Microarrays*. Chapman & Hall/CRC Mathematical & Computational Biology. Taylor & Francis, 2003.

[40] Xinxin Du and Kok Kiong Tan. Comprehensive and Practical Vision System for Self-Driving Vehicle Lane-level Localization. *IEEE Trans. Image Processing*, 25(5):2075–2088, 2016.

[41] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level Classification of Skin Cancer with Deep Neural Networks. *Nature*, 542(7639):115–118, February 2017.

[42] European Bioinformatics Institute (EMBL-EBI). Ensembl Bacteria. `https://bacteria.ensembl.org/index.html`. Accessed: 2018-10-15.

[43] European Bioinformatics Institute (EMBL-EBI). Ensembl Genomes. `http://ensemblgenomes.org`. Accessed: 2018-10-17.

[44] European Bioinformatics Institute (EMBL-EBI). Pan Taxonomic Compara. `http://ensemblgenomes.org/info`. Accessed: 2018-10-17.

[45] Nathaniel Fairfield and Chris Urmson. Traffic Light Mapping and Detection. *Google*, pages 1–6, February 2011.

[46] Diego Fioravanti, Ylenia Giarratano, Valerio Maggio, Claudio Agostinelli, Marco Chierici, Giuseppe Jurman, and Cesare Furlanello. Phylogenetic Convolutional Neural Networks in Metagenomics. *BMC bioinformatics*, 19(Suppl 2):49, March 2018.

[47] Jean-loup Gailly and Mark Adler. GNU GZIP. `https://www.gzip.org`. Accessed: 2018-10-23.

[48] Hasindu Gamaarachchi, Sri Parameswaran, and Martin Smith. Featherweight Long Read Alignment Using Partitioned Reference Indexes. *bioRxiv*, page 386847, August 2018.

[49] Dirk Gevers, Rob Knight, Joseph F Petrosino, Katherine Huang, Amy L McGuire, Bruce W Birren, Karen E Nelson, Owen White, Barbara A Methé, and Curtis Huttenhower. The Human Microbiome Project: A Community Resource for the Healthy Human Microbiome. *PLoS biology*, 10(8), August 2012.

[50] Dirk Gevers, Mihai Pop, Patrick D Schloss, and Curtis Huttenhower. Bioinformatics for the Human Microbiome Project. *PLoS Computational Biology*, 8(11):e1002779, 2012.

[51] Molly K. Gibson, Bin Wang, Sara Ahmadi, Carey-Ann D. Burnham, Phillip I. Tarr, Barbara B. Warner, and Gautam Dantas. Developmental Dynamics of the Preterm Infant Gut Microbiota and Antibiotic Resistome. *Nature Microbiology*, 1(4):16024, March 2016.

[52] GitHub, Inc. GitHub, Hosting and Version Control using Git. `https://github.com`. Accessed: 2018-11-15.

[53] Joachim Goedhart. PlotTwist: A Web App for Plotting and Annotating Continuous Data. *PLoS biology*, 18(1):e3000581, January 2020.

[54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[55] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[56] Google, LLC. Google maps. `https://www.google.com/maps`. Accessed: 2018-11-17.

[57] Chiara Gorni, Donatella Allemand, Dario Rossi, and Paola Mariani. Microbiome Profiling in Fresh-Cut Products. *Trends in Food Science and Technology*, 46(2):295–301, 2015.

[58] Hadrien Gourlé, Oskar Karlsson-Lindsjö, Juliette Hayer, and Erik Bongcam-Rudloff. Simulating Illumina Metagenomic Data with InSilicoSeq. *Bioinformatics (Oxford, England)*, July 2018.

[59] Zuguang Gu, Roland Eils, and Matthias Schlesner. HilbertCurve - An R/Bioconductor Package for High-Resolution Visualization of Genomic Data. *Bioinformatics*, 32(15):2372–2374, 2016.

[60] Runxin Guo, Yi Zhao, Quan Zou, Xiaodong Fang, and Shaoliang Peng. Bioinformatics Applications on Apache Spark. *GigaScience*, 7(8), August 2018.

[61] Zhe Guo, Xiang Li, Heng Huang, Ning Guo, and Quanzheng Li. Medical Image Segmentation Based on Multi-Modal Convolutional Neural Network: Study on Image Fusion Schemes. *arXiv.org*, October 2017.

[62] Matthias G Haberl, Christopher Churas, Lucas Tindall, Daniela Boassa, Sébastien Phan, Eric A Bushong, Matthew Madany, Raffi Akay, Thomas J Deerinck, Steven T Peltier, and Mark H Ellisman. CDeep3M-Plug-and-Play

cloud-based deep learning for image segmentation. *Nature methods*, 15(9):677–680, September 2018.

[63] Henry J Haiser, David B Gootenberg, Kelly Chatman, Gopal Sirasani, Emily P Balskus, and Peter J Turnbaugh. Predicting and Manipulating Cardiac Drug Inactivation by the Human Gut Bacterium Eggerthella lenta. *Science (New York, NY)*, 341(6143):295–298, 2013.

[64] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain Tumor Segmentation with Deep Neural Networks. *Medical image analysis*, 35:18–31, January 2017.

[65] Hawkins, R David, Hon, Gary C, and Ren, Bing. Next-generation Genomics: an Integrative Approach. *Nature reviews Genetics*, 11(7):476–486, June 2010.

[66] David Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. In *Dritter Band: Analysis · Grundlagen der Mathematik · Physik Verschiedenes*, pages 1–2. Springer, Berlin, Heidelberg, 1935.

[67] Robert Holt and Steven Jones. The New Paradigm of Flow Cell Sequencing. *Genome Research*, 18(1):839–846, 2008.

[68] Benlin Hu, Cheng Lei, Dong Wang, Shu Zhang, and Zhenyu Chen. A Preliminary Study on Data Augmentation of Deep Learning for Image Classification. *CoRR*, cs.CV, 2019.

[69] Xuegang Hu and Hongguang Yang. DRU-net - A Novel U-net for Biomedical Image Segmentation. *IET Image Processing*, 14(1):192–200, 2020.

[70] Liren Huang, Jan Krüger, and Alexander Sczyrba. Analyzing Large Scale Genomic Data on the Cloud with Sparkhit. *Bioinformatics (Oxford, England)*, 34(9):1457–1465, 2018.

[71] Human Microbiome Project Consortium. A Framework for Human Microbiome Research. *Nature*, 486(7402):215–221, June 2012.

[72] Curtis Huttenhower, Dirk Gevers, Rob Knight, Sahar Abubucker, Jonathan Badger, Asif Chinwalla, Heather Huot Creasy, Earl AM, Michael Fitzgerald, Robert Fulton, Michelle Giglio, Kymberlie Pepin, Lobos EA, Ramana Madupu, Vincent Magrini, John Martin, Makedonka Mitreva, Muzny DM, Sodergren EJ, and Owen White. Structure, Function and Diversity of the Healthy Human Microbiome. *Nature*, 486:207–214, 06 2012.

[73] Hypergiant, LLC. Kaleidoscope, Spot the Differences, Merge in Seconds. `https://www.kaleidoscopeapp.com`. Accessed: 2020-4-10.

[74] Illumina, Inc. Illumina Sequencing Platforms. `https://www.illumina.com/systems/sequencing-platforms.html`. Accessed: 2020-4-08.

[75] Ishan Jindal, Matthew S Nokleby, and Xuewen Chen. Learning Deep Networks from Noisy Labels with Dropout Regularization. *16th IEEE International Conference on Data Mining (ICDM)*, 1(1):967–972, 2016.

[76] Ji Young Jung, Se Hee Lee, Hyun Mi Jin, Yoonsoo Hahn, Eugene Madsen, and Che Ok Jeon. Metatranscriptomic Analysis of Lactic Acid Bacterial Gene Expression During Kimchi Fermentation. *International Journal of Food Microbiology*, 163(2):171–179, 2013.

[77] Daniel A Keim. Pixel-Oriented Visualization Techniques for Exploring Very Large Data Bases. *Journal of Computational and Graphical Statistics*, 5(1):58–77, February 1996.

[78] Kidney Disease Improving Global Outcomes Foundation. Kidney Disease Improving Global Outcomes Guidelines. `https://kdigo.org/guidelines`. Accessed: 2018-11-17.

[79] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ArXiv*, 1(1), 2014.

[80] Robert A Koeth, Zeneng Wang, Bruce S Levison, Jennifer A Buffa, Elin Org, Brendan T Sheehy, Earl B Britt, Xiaoming Fu, Yuping Wu, Lin Li, Jonathan D Smith, Joseph A DiDonato, Jun Chen, Hongzhe Li, Gary D Wu, James D Lewis, Manya Warrier, J Mark Brown, Ronald M Krauss, W H Wilson Tang, Frederic D Bushman, Aldons J Lusis, and Stanley L Hazen. Intestinal Microbiota Metabolism of L-Carnitine, a Nutrient in Red Meat, Promotes Atherosclerosis. *Nature medicine*, 19(5):576–585, May 2013.

[81] Tadashi Kondo, Junji Ueno, and Shoichiro Takao. Medical Image Analysis of Brain X-ray CT Images By Deep GMDH-Type Neural Network. *JRNAL*, 3(1):17–23, 2016.

[82] Martin I Krzywinski, Jacqueline E Schein, Inanc Birol, Joseph Connors, Randy Gascoyne, Doug Horsman, Steven J Jones, and Marco A Marra. Circos: An Information Aesthetic for Comparative Genomics. *Genome Research*, 2009.

[83] Myron La Duc, Wayne Nicholson, Roger Kern, and Kasthuri Venkateswaran. Microbial Characterization of the Mars Odyssey Spacecraft and its Encapsulation Facility. *Environmental Microbiology*, 5(10):977–985, 2003.

[84] Ben Langmead and Steven L Salzberg. Fast Gapped-Read Alignment with Bowtie 2. *Nature methods*, 9(4):357–359, 2012.

[85] Ben Langmead, Michael C Schatz, Jimmy Lin, Mihai Pop, and Steven L Salzberg. Searching for SNPs with Cloud Computing. *Genome biology*, 10(11), 2009.

[86] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. Ultrafast and Memory-Efficient Alignment of Short DNA Sequences to the Human Genome. *Genome biology*, 10(3):R25, 2009.

[87] Nathan LaPierre, Chelsea J-T Ju, Guangyu Zhou, and Wei Wang. MetaPheno: A Critical Evaluation of Deep Learning and Machine Learning in Metagenome-Based Disease Prediction. *Methods (San Diego, Calif)*, 166:74–82, August 2019.

[88] Malo Le Boulch, Patrice Déhais, Sylvie Combes, and Géraldine Pascal. The MACADAM database - A MetAboliC Pathways Database for Microbial Taxonomic Groups for Mining Potential Metabolic Capacities of Archaeal and Bacterial Taxonomic Groups. *Database, The Journal of Biological Databases and Curation*, V2019(baz049):1–14, 2019.

[89] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989.

[90] Frederick Lee, Douglas Rusch, Frank Stewart, Heather Mattila, and Irene Newton. Saccharide Breakdown and Fermentation by the Honey Bee Gut Microbiome. *Environmental Microbiology*, 17(3):796–815, 2014.

[91] Pirmin Lemberger. On Generalization and Regularization in Deep Learning. *CoRR*, 2017.

[92] Ivica Letunic and Peer Bork. Interactive Tree of Life (ITOL) v3: An Online Tool for the Display and Annotation of Phylogenetic and Other Trees. *Nucleic Acids Research*, 44:gkw290, 04 2016.

[93] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, and 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)*, 25(16):2078–2079, August 2009.

[94] Xiaqing Li, Guangyan Zhang, H. Howie Huang, Zhufan Wang, and Weimin Zheng. Performance Analysis of GPU-based Convolutional Neural Networks. *45th International Conference on Parallel Processing*, 1(1), 2016.

[95] Martin S Lindner, Benjamin Strauch, Jakob M Schulze, Simon H Tausch, Piotr W Dabrowski, Andreas Nitsche, and Bernhard Y Renard. HiLive: Real-time Mapping of Illumina Reads while Sequencing. *Bioinformatics (Oxford, England)*, 33(6):917–319, March 2017.

[96] Jennifer Lu, Florian P Breitwieser, Peter Thielen, and Steven L Salzberg. Bracken: Estimating Species Abundance in Metagenomics Data. *PeerJ Computer Science*, 3:e104, 2017.

[97] MacLean, D, Jones, J, and Studholme, D. Application of "Next-generation" Sequencing Technologies to Microbial Genetics. *Nature Reviews Microbiology*, 7(1):96–97, 2009.

[98] Anant Madabhushi and George Lee. Image Analysis and Machine Learning in Digital Pathology - Challenges and Opportunities. *Medical image analysis*, 33:170–175, 2016.

[99] Mardis, Elaine R. Next-generation DNA Sequencing Methods. *Annual review of genomics and human genetics*, 9(1):387–402, 2008.

[100] J. Bull Matthew and Nigel Plummer. Part 1: The Human Gut Microbiome in Health and Disease. *Integrative Medicine: A Clinician's Journal*, 13(6):17–22, 2014.

[101] Emeran Mayer, Tor Savidge, and Robert Shulman. Brain–Gut Microbiome Interactions and Functional Bowel Disorders. *Gastroenterology*, 146(6):1500–1512, 2014.

[102] Joshua McCann, Ahmed Elolimy, and Juan Loor. Rumen Microbiome, Probiotics, and Fermentation Additives. *Veterinary Clinics: Food Animal Practice*, 33(3):539–553, 2017.

[103] Alexa McIntyre, Rachid Ounit, Ebrahim Afshinnekoo, Robert J Prill, Elizabeth Hénaff, Noah Alexander, Samuel S Minot, David Danko, Jonathan Foox, Sofia Ahsanuddin, Scott Tighe, Nur A Hasan, Poorani Subramanian, Kelly Moffat, Shawn Levy, Stefano Lonardi, Nick Greenfield, Rita R Colwell, Gail L Rosen, and Christopher E Mason. Comprehensive Benchmarking and Ensemble Approaches for Metagenomic Classifiers. *Genome biology*, 18(1):182, September 2017.

[104] Jean Pierre Menard, Florence Fenollar, Mireille Henry, Florence Bretelle, and Didier Raoult. Molecular Quantification of *Gardnerella vaginalis* and *Atopobium vaginae* Loads to Predict Bacterial Vaginosis. *academic.oup.com*, 47(1):33–43, 2008.

[105] Michael L. Metzker. Sequencing Technologies - The Next Generation. *Nature reviews Genetics*, 11(1):31–46, 2010.

[106] Microsoft Corporation. Microsoft Excel. `https://products.office.com/en-us/excel`. Accessed: 2020-01-14.

[107] Gordon E. Moore. Cramming More Components onto Integrated Circuits. *Electronics*, 38(8):114, 1965.

[108] Ali Mortazavi, Brian A Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. Mapping and Quantifying Mammalian Transcriptomes by RNA-Seq. *Nature methods*, 5(7):621–628, July 2008.

[109] Daniel J Nasko, Sergey Koren, Adam M Phillippy, and Todd J Treangen. RefSeq Database Growth Influences the Accuracy of k-mer-based Lowest Common Ancestor Species Identification. *Genome biology*, 19(1):165, October 2018.

[110] National Center for Biotechnology Information, U.S. National Library of Medicine. RefSeq Growth Statistics. `https://www.ncbi.nlm.nih.gov/refseq/statistics`. Accessed: 2020-02-12.

[111] National Human Genome Research Institute. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP). `https://www.genome.gov/sequencingcostsdata`. Accessed: 2020-4-13.

[112] Thanh Hai Nguyen, Edi Prifti, Yann Chevaleyre, Nataliya Sokolovska, and Jean-Daniel Zucker. Disease Classification in Metagenomics with 2D Embeddings and Deep Learning. *arXiv.org*, June 2018.

[113] Nuala A O'Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufo, Diana Haddad, Richard McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, Alexander Astashyn, Azat Badretdin, Yiming Bao, Olga Blinkova, Vyacheslav Brover, Vyacheslav Chetvernin, Jinna Choi, Eric Cox, Olga D Ermolaeva, Catherine M Farrell, Tamara Goldfarb, Tripti Gupta, Daniel H Haft, Eneida Hatcher, Wratko Hlavina, Vinita S Joardar, Vamsi K Kodali, Wenjun Li, Donna R Maglott, Patrick Masterson, Kelly M McGarvey, Michael R Murphy, Kathleen O'Neill, Shashikant Pujar, Sanjida H Rangwala, Daniel Rausch 0002, Lillian D Riddick, Conrad L Schoch, Andrei Shkeda, Susan S Storz, Hanzhen Sun, Françoise Thibaud-Nissen, Igor Tolstoy, Raymond E Tully, Anjana R Vatsan, Craig Wallin, David Webb, Wendy Wu, Melissa J Landrum, Avi Kimchi, Tatiana A Tatusova, Michael Dicuccio, Paul A Kitts, Terence D Murphy, and Kim D Pruitt. Reference sequence (RefSeq) Database at NCBI - Current Status, Taxonomic Expansion, and Functional Annotation. *Nucleic Acids Research*, 44(D1):D733–45, 2016.

[114] Travis Oliphant. NumPy: A guide to NumPy. Trelgol Publishing, 2006. Accessed: 2018-04-17.

[115] Brian Ondov, Nicholas Bergman, and Adam Phillippy. Interactive Metagenomic Visualization in a Web Browser. *BMC Bioinformatics*, 1(1), 2011.

[116] Ahmet Haydar Ornek and Murat Ceylan. Comparison of Traditional Transformations for Data Augmentation in Deep Learning of Medical Thermography. *42nd International Conference on Telecommunications and Signal Processing (TSP)*, 1(1):191–194, 2019.

[117] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[118] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Classifier Comparison., 2020. Accessed: 2020-02-14.

[119] Catia Pinto, Diogo Pinho, Remy Cardoso, Valeria dos Santos Custodio, Joana Fernandes, Susana Sousa, Miguel Pinheiro, Conceicao Egas, and Ana Gomes. Wine Fermentation Microbiome: A Landscape from Different Portuguese Wine Appellations. *Frontiers in Microbiology*, 1(1), 2015.

[120] Plotly Technologies Inc. Plotly: Collaborative Data Science. `https://plot.ly`, 2015.

[121] Manuel Porcar, Katherine Louie, Suzanne Kosina, Marc W. Van Goethem, Benjamin Bowen, Kristie Tanner, and Trent Northen. Microbial Ecology on Solar Panels in Berkeley, CA, United States. *Frontiers in Microbiology*, 9:3043, 12 2018.

[122] Christian Quast et al. The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Research*, 41(D1):D590–D596, 11 2012.

[123] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[124] Ravi Ranjan, Asha Rani, Ahmed Metwally, Halvor S McGee, and David L Perkins. Analysis of the Microbiome: Advantages of Whole Genome Shotgun versus 16S Amplicon Sequencing. *Biochemical and biophysical research communications*, 469(4):967–977, January 2016.

[125] Zeehasham Rasheed and Huzefa Rangwala. A Map-Reduce Framework for Clustering Metagenomes. In *International Symposium on Parallel & Distributed Processing*, pages 549–558. IEEE, 2013.

[126] Sanjay Rathee and Arti Kashyap. StreamAligner: A Streaming-based Sequence Aligner on Apache Spark. *Journal of Big Data*, 5(1):8, December 2018.

[127] Adam Roberts, Harvey Feng, and Lior Pachter. Fragment Assignment in the Cloud with eXpress-D. *BMC bioinformatics*, 14(1), 2013.

[128] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer, Cham, October 2015.

[129] RStudio Team. RStudio: Integrated Development Environment for R, 2015.

[130] Eder Santana and George Hotz. Learning a Driving Simulator. *arXiv.org*, August 2016.

[131] Eric W Sayers, Richa Agarwala, Evan Bolton, J Rodney Brister, Kathi Canese, Karen Clark, Ryan Connor, Nicolas Fiorini, Kathryn Funk, Timothy Hefferon, J Bradley Holmes, Sunghwan Kim 0002, Avi Kimchi, Paul A Kitts, Stacy Lathrop, Zhiyong Lu, Thomas L Madden, Aron Marchler-Bauer, Lon Phan, Valerie A Schneider, Conrad L Schoch, Kim D Pruitt, and James Ostell. Database Resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 47:D23, 2019.

[132] Lorian Schaeffer, Harold Pimentel, Nicolas Bray, Páll Melsted, and Lior Pachter. Pseudoalignment for Metagenomic Read Assignment. *arXiv.org*, October 2015.

[133] School of Computing and Information Sciences. Computer and information sciences, florida international university. `https://www.cis.fiu.edu`. Accessed: 2018-10-23.

[134] Jay Shendure and Hanlee Ji. Next-generation DNA Sequencing. *Nature biotechnology*, 26(10):1135–1145, October 2008.

[135] Connor Shorten and Taghi M Khoshgoftaar. A Survey on Image Data Augmentation for Deep Learning. *J. Big Data*, 6(1):60, 2019.

[136] Panagiotis Stalidis, Theodoros Semertzidis, and Petros Daras. Examining Deep Learning Architectures for Crime Classification and Prediction. *arXiv.org*, December 2018.

[137] Daniel Strigl, Klaus Kofler, and Stefan Podlipnig. Performance and scalability of GPU-based convolutional neural networks. *18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, 1(1), 2010.

[138] Su, Zhenqiang, Labaj, Pawel P, Li, Sheng, Thierry-Mieg, Jean, Thierry-Mieg, Danielle, Shi, Wei, Wang, Charles, Schroth, Gary P, Setterquist, Robert A, Thompson, John F, Jones, Wendell D, Xiao, Wenzhong, Xu, Weihong, Jensen, Roderick V, Kelly, Reagan, Xu, Joshua, Conesa, Ana, Furlanello, Cesare, Gao, Hanlin, Hong, Huixiao, Jafari, Nadereh, Letovsky, Stan, Liao, Yang, Lu, Fei, Oakeley, Edward J, Peng, Zhiyu, Praul, Craig A, Santoyo-Lopez, Javier, Scherer, Andreas, Shi, Tieliu, Smyth, Gordon K, Staedtler, Frank, Sykacek, Peter, Tan, Xin-Xing, Thompson, E Aubrey, Vandesompele, Jo, Wang, May D, Wang, Jian, Wolfinger, Russell D, Zavadil, Jiri, Auerbach, Scott S, Bao, Wenjun, Binder, Hans, Blomquist, Thomas, Brilliant, Murray H, Bushel, Pierre R, Cain, Weimin, Catalano, Jennifer G, Chang, Ching-Wei, Chen, Tao, Chen, Geng, Chen, Rong, Chierici, Marco, Chu, Tzu-Ming, Clevert, Djork-Arne, Deng, Youping, Derti, Adnan, Devanarayan, Viswanath, Dong, Zirui, Dopazo, Joaquín, Du, Tingting, Fang, Hong, Fang, Yongxiang, Fasold, Mario, Fernandez, Anita, Fischer, Matthias, Furio-Tari, Pedro, Fuscoe, James C, Caimet, Florian, Gaj, Stan, Gandara, Jorge, Gao, Huan, Ge, Weigong, Gondo, Yoichi, Gong, Binsheng, Gong, Meihua, Gong, Zhuolin, Green, Bridgett, Guo, Chao, Guo, Lei, Guo, Li-Wu, Hadfield, James, Hellemans, Jan, Hochreiter, Sepp,

Jia, Meiwen, Jian, Min, Johnson, Charles D, Kay, Suzanne, Kleinjans, Jos, Lababidi, Samir, Levy, Shawn, Li, Quan-Zhen, Li, Li, Li, Peng, Li, Yan, Li, Haiqing, Li, Jianying, Li, Shiyong, Lin, Simon M, Lopez, Francisco J, Lu, Xin, Luo, Heng, Ma, Xiwen, Meehan, Joseph, Megherbi, Dalila B, Mei, Nan, Mu, Bing, Ning, Baitang, Pandey, Akhilesh, Perez-Florido, Javier, Perkins, Roger G, Peters, Ryan, Phan, John H, Pirooznia, Mehdi, Qian, Feng, Qing, Tao, Rainbow, Lucille, Rocca-Serra, Philippe, Sambourg, Laure, Sansone, Susanna-Assunta, Schwartz, Scott, Shah, Ruchir, Shen, Jie, Smith, Todd M, Stegle, Oliver, Stralis-Pavese, Nancy, Stupka, Elia, Suzuki, Yutaka, Szkotnicki, Lee T, Tinning, Matthew, Tu, Bimeng, van Deft, Joost, Vela-Boza, Alicia, Venturini, Elisa, Walker, Stephen J, Wan, Liqing, Wang, Wei, Wang, Jinhui, Wang, Jun, Wieben, Eric D, Willey, James C, Wu, Po-Yen, Xuan, Jiekun, Yang, Yong, Ye, Zhan, Yin, Ye, Yu, Ying, Yuan, Yate-Ching, Zhang, John, Zhang, Ke K, Zhang, Wenqian, Zhang, Wenwei, Zhang, Yanyan, Zhao, Chen, Zheng, Yuanting, Zhou, Yiming, Zumbo, Paul, Tong, Weida, Kreil, David P, Mason, Christopher E, and Shi, Leming. A comprehensive assessment of RNA-seq accuracy, reproducibility and information content by the Sequencing Quality Control Consortium. *Nature biotechnology*, 32(9):903–914, September 2014.

[139] Aravind Subramanian, Pablo Tamayo, Vamsi K. Mootha, Sayan Mukherjee, Benjamin L. Ebert, Michael A. Gillette, Amanda Paulovich, Scott L. Pomeroy, Todd R. Golub, Eric S. Lander, and Jill P. Mesirov. Gene Set Enrichment Analysis: A Knowledge-based Approach for Interpreting Genome-wide Expression Profiles. *National Acad Sciences*, 102(43):15545–15550, 2005.

[140] Pen Sun, Jia Wang, and Li Deng. Effects of *Bacillus subtilis natto* on Milk Production, Rumen Fermentation and Ruminal Microbiome of Dairy Cows. *Animal, Cambridge Core*, 7(2):1–7, 2013.

[141] Tableau Software, LLC. Tableau. `https://www.tableau.com`. Accessed: 2020-01-14.

[142] Simon H Tausch, Benjamin Strauch, Andreas Andrusch, Tobias P Loka, Martin S Lindner, Andreas Nitsche, and Bernhard Y Renard. LiveKraken: Real-time Metagenomic Classification of Illumina Data. *Bioinformatics (Oxford, England)*, 34(21):3750–3752, June 2018.

[143] The Apache Software Foundation. Apache Hadoop. `http://hadoop.apache.org`. Accessed: 2018-10-17.

[144] The Apache Software Foundation. The Apache Projects. `https://www.apache.org`. Accessed: 2018-10-17.

[145] The International Business Machines Corporation. IBM Spectrum LSF. `https://www.ibm.com/support/knowledgecenter/en/SSWRJV/product_welcome_spectrum_lsf.html`. Accessed: 2019-03-20.

[146] Benjamin Thompson. NASA, the Spacecraft Assembly Facility, and the Extremotolerant Bacteria. *Microbiology Society*, 1(1), 2013.

[147] Cole Trapnell and Steven L Salzberg. How to Map Billions of Short Reads onto Genomes. *Nature biotechnology*, 27(5):455–457, May 2009.

[148] Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript Assembly and Quantification by RNA-Seq Reveals Unannotated Transcripts and Isoform Switching During Cell Differentiation. *Nature biotechnology*, 28(5):511–515, May 2010.

[149] Peter J Turnbaugh, Ruth E Ley, Micah Hamady, Claire M Fraser-Liggett, Rob Knight, and Jeffrey I Gordon. The Human Microbiome Project. *Nature*, 449(7164):804–810, October 2007.

[150] Elif Derya Übeyli. Combining Recurrent Neural Networks with Eigenvector Methods for Classification of ECG Beats. *Digital Signal Processing*, 19(2):320–329, March 2009.

[151] Kensuke Umehara, Junko Ota, Naoki Ishimaru, Shunsuke Ohno, Kentaro Okamoto, Takanori Suzuki, Naoki Shirai, and Takayuki Ishida. Super-resolution Convolutional Neural Network for the Improvement of the Image Quality of Magnified Images in Chest Radiographs. *Medical Imaging - Image Processing*, 2017.

[152] Camilo Valdes, Meghan Brennan, Bertrand Clarke, and Jennifer Clarke. Detecting Bacterial Genomes in a Metagenomic Sample using NGS Reads. *Statistics and Its Interface*, 8(4):477–494, 2015.

[153] Valdes, Camilo, Stebliankin, Vitalii, and Narasimhan, Giri. Large Scale Microbiome Profiling in the Cloud. *Bioinformatics (Oxford, England)*, 35(14):i13–i22, July 2019.

[154] Nicolas Vasilache, Jeff Johnson, Michael Mathieu, Soumith Chintala, Serkan Piantino, and Yann LeCun. Fast Convolutional Nets with FBFFT: A GPU Performance Evaluation. *arxiv.org*, 2014.

[155] Kasthuri Venkateswaran, Parag Vaishampayan, Jessica Cisneros, Duane Person, Scott Rogers, and Jay Perry. International Space Station Environmental Microbiome—Microbial Inventories of ISS Filter Debris. *Applied Microbiology and Biotechnology*, 98(1):6453–6466, 2014.

[156] George Vernikos, Duccio Medini, David R Riley, and Hervé Tettelin. Ten Years of Pan-genome Analyses. *Current opinion in microbiology*, 23:148–154, February 2015.

[157] Noorul Wahab, Asifullah Khan, and Yeon Soo Lee. Transfer Learning Based Deep CNN for Segmentation and Detection of Mitoses in Breast Cancer Histopathological Images. *Microscopy (Oxford, England)*, 68(3):216–233, February 2019.

[158] Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-Seq: A Revolutionary Tool for Transcriptomics. *Nature reviews Genetics*, 10(1):57–63, January 2009.

[159] Alice R Wattam, James J Davis, Rida Assaf, Sébastien Boisvert, Thomas Brettin, Christopher Bun, Neal Conrad, Emily M Dietrich, Terry Disz, Joseph L Gabbard, Svetlana Gerdes, Christopher S Henry, Ronald W Kenyon, Dustin Machi, Chunhong Mao, Eric K Nordberg, Gary J Olsen, Daniel E Murphy-Olson, Robert Olson, Ross Overbeek, Bruce Parrello, Gordon D Pusch, Maulik Shukla, Veronika Vonstein, Andrew Warren, Fangfang Xia, Hyunseung Yoo, and Rick L Stevens. Improvements to PATRIC, the All-Bacterial Bioinformatics Database and Analysis Resource Center. *Nucleic Acids Research*, 45(D1):D535–D542, January 2017.

[160] Wikipedia. Pablo picasso. `https://en.wikipedia.org/wiki/Pablo_Picasso`. Accessed: 2020-04-03.

[161] Wikipedia. Sankey Diagram. `https://en.wikipedia.org/wiki/Sankey_diagram`, 2020. Accessed: 2020-02-14.

[162] Wilhelm, Brian T, Marguerat, Samuel, Watt, Stephen, Schubert, Falk, Wood, Valerie, Goodhead, Ian, Penkett, Christopher J, Rogers, Jane, and Bähler, Jürg. Dynamic Repertoire of a Eukaryotic Transcriptome Surveyed at Single-Nucleotide Resolution. *Nature*, 453(7199):1239–1243, June 2008.

[163] Niki Wilson. A Microbial Hitchhiker's Guide to the Galaxy: Researchers Race to Understand Effects of Deep Space on the Microbiome. *Bioscience*, 69(1):5–11, 2019.

[164] Geoffrey L Winsor, Emma J Griffiths, Raymond Lo, Bhavjinder K Dhillon, Julie A Shay, and Fiona S L Brinkman. Enhanced Annotations and Features for Comparing Thousands of *Pseudomonas* Genomes in the *Pseudomonas* Genome Database. *Nucleic Acids Research*, 44(D1):D646–D653, 2016.

[165] Pak Chung Wong, Kwong Kwok Wong, H. Foote, and J. Thomas. Global Visualization and Alignments of Whole Bacterial Genomes. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):361–377, July 2003.

[166] Derrick E Wood, Jennifer Lu, and Ben Langmead. Improved Metagenomic Analysis with Kraken 2. *Genome biology*, 20(1):1–13, December 2019.

[167] Derrick E Wood and Steven L Salzberg. Kraken: Ultrafast Metagenomic Sequence Classification using Exact Alignments. *Genome biology*, 15(3):R46, March 2014.

[168] Gary D Wu and James D Lewis. Analysis of the Human Gut Microbiome and Association With Disease. *Clinical Gastroenterology and Hepatology*, 11(7):774–777, July 2013.

[169] Lin Xu, Cheng Xu, Yi Tong, and Yu Chun Su. Detection and Classification of Breast Cancer Metastates Based on U-Net. *arXiv.org*, September 2019.

[170] Xin Yi, Ekta Walia, and Paul Babyn. Generative Adversarial Network in Medical Imaging: A Review. *Medical Image Analysis*, 58, 2019.

[171] Ed Yong. *I Contain Multitudes: The Microbes Within Us and a Grander View of Life*. Life Sciences. HarperCollins, 2016.

[172] K Yu, W Xu, Y Gong Advances in Neural Information Processing, and 2009. Deep Learning with Kernel Regularization for Visual Recognition. *Advances in Neural Information Processing Systems 21*, 2009.

[173] Matei Zaharia. Apache Spark. `http://spark.apache.org`. Accessed: 2018-10-17.

[174] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J Franklin, Scott Shenker, and Ion Stoica. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. *9th USENIX Symposium on Networked Systems Design and Implementation*, pages 15–28, 2012.

[175] Yinfeng Zhang, Cheuk-Yin Lun, and Stephen Kwok-Wing Tsui. Metagenomics: A New Way to Illustrate the Crosstalk between Infectious Diseases and Host Microbiome. *International journal of molecular sciences*, 16(11):26263–26279, 2015.

[176] Z Zhang, C Xu, J Yang, Y Tai, and L Chen. Deep Hierarchical Guidance and Regularization Learning for End-to-End Depth Estimation. *Pattern Recognition*, 83:430–442, 2018.

[177] Wei Zhou, Nicole Gay, and Julia Oh. ReprDB and panDB: Minimalist Databases with Maximal Microbial Representation. *Microbiome*, 6(1):15, January 2018.

[178] Wei Zhou, Ruilin Li, Shuo Yuan, ChangChun Liu, Shaowen Yao, Jing Luo, and Beifang Niu. MetaSpark: A Spark-based Distributed Processing Tool to Recruit Metagenomic Reads to Reference Genomes. *Bioinformatics (Oxford, England)*, 33(7):1090–1092, April 2017.

APPENDIX 1

A1.1 Classifier Parameters

The code-block in Listing 1.1 declares the classifier instances and their respective paramenters. When no parameters are explicitly listed, then the default parameters are used. Details can be found at the `scikit-learn` API url `https://scikit-learn.org/stable/modules/classes.html`.

```
classifiers = [
   KNeighborsClassifier("3"),
   SVC(kernel="linear", C="0.025"),
   SVC(gamma="2", C="1"),
   RandomForestClassifier(max_depth="5", n_estimators="10",
      max_features="1"),
   GaussianNB(),
   MLPClassifier(alpha="1", max_iter="100"),
   AdaBoostClassifier()
]
```

Listing 1.1: Classifier parameters.

A1.2 Data Augmentation

Data augmentation for the Amber CNN was performed by implementing a custom `ImageDataGenerator` using the Keras and Tensorflow libraries. The main function is in Listing 1.2.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

img_h, img_w = 256

train_datagen = ImageDataGenerator(rotation_range=10,
   width_shift_range=0.1,
   height_shift_range=0.1, shear_range=0.15,
   zoom_range=0.1, channel_shift_range=0.2,
   horizontal_flip=False)

for subsite_dir in listdir(train_dir):
   for hilbert_img in listdir(subsite_dir_path):
      image_gen = train_datagen.flow(img,
          save_to_dir=subsite_augmented_dir,
         save_prefix='aug', save_format='png', batch_size=1)
      total_aug_imgs = 0
      for aug_img in image_gen:
         total_aug_imgs += 1
         overall_imgs_created += 1
         if total_aug_imgs == files_to_generate or
            overall_imgs_created == target_number_samples:
            break
```

Listing 1.2: Code for creating augmented microbiome maps.

VITA

CAMILO VALDES

2015-2020	Ph.D., Computer Science
	Florida International University
	Miami, Florida
2015-2019	M.S., Computer Science
	Florida International University
	Miami, Florida
2010	B.S., Computer Science
	Florida International University
	Miami, Florida
2010	B.S., Biological Sciences
	Florida International University
	Miami, Florida

My primary research interests lie in developing high-performance, real time distributed applications for the analysis of very large datasets. Recently, I have been involved in projects that study how microbes affect human disease, and how to visualize their results. The underlying theme to my projects is their massive datasets, and I am interested in developing algorithms and methods for improving their analysis, as well as creating new techniques for visualizing and creating interpretable results.

Lead developer and architect of Flint, a cloud profiling framework for metagenomic whole-genome sequencing. (https://github.com/camilo-v/flint)

Lead developer and architect of Jasper, a framework for visualizing and characterizing metagenomic DNA sequencing datasets. (https://github.com/camilo-v/jasper)

GRANTS & FELLOWSHIPS
Amazon AWS Cloud Credits for Research. Awarded $25,000 in AWS credits to study algorithms and methods for the profiling and characterization of sequencing datasets. The award resulted in the Flint and Jasper projects.
Dissertation Year Fellowship. Awarded a Florida International University dissertation year fellowship for the completion of PhD dissertation work (2019 Fall - 2020 Spring).

Experience in the design and implementation of probabilistic models for human gene expression datasets, resulting in research publications and open source software tools. Experience in the design and application of statistical hypothesis testing methods as applied to human genomics and transcriptomics cancer datasets, as well as very large metagenomic datasets.

PUBLICATIONS AND PRESENTATIONS

Valdes C, Stebliankin V, Ruiz-Perez D, Park J, Lee H, Narasimhan G. *Microbiome Maps: Hilbert Curve Visualizations of Metagenomic Profiles*. In Review. International Conference on Intelligent Systems for Molecular Biology (ISMB). 2020.

Valdes C, Stebliankin V, Narasimhan G. *Large Scale Microbiome Profiling in the Cloud*. Oxford Bioinformatics 35 (14), i13-i22. 2019.

Ayad N, Valdes C, Clarke J, et al. *Time Series Modeling of Cell Cycle Exit Identifies Brd4-dependent Regulation of Cerebellar Neurogenesis*. Nature Communications 10 (1), 1-11. 2019.

Dobra A, Valdes C, Clarke B, Clarke J. *Modeling Association in Microbial Communities with Clique Loglinear Models*. The Annals of Applied Statistics 13 (2), 931-957. 2019.

Zheng Xu, Valdes C, Clarke J. *Existing and Potential Statistical and Computational Approaches for the Analysis of 3D CT Images of Plant Roots*. Agronomy. 2018.

Capobianco E, Valdes C, Sarti S, Jiang Z, Poliseno L, Tsinoremas N. *Ensemble Modeling Approach Targeting Heterogeneous RNA-Seq data: Application to Melanoma Pseudogenes*. Nature, Scientific Reports. 7 (1), 17344. 2017.

Valdes C, Brennan M, Dobra A, Clarke B, Clarke J. *Detecting Bacterial Genomes in a Metagenomic Sample Using NGS Reads*. Statistics and Its Interface. Statistics and Its Interface 8 (4), 477-494. 2015.

Clarke B, Valdes C, Dobra A, Clarke J. *A Bayes Testing Approach to Metagenomic Profiling in Bacteria. Statistics and Its Interface*. Volume 8, Number 2, pp. 173-185. 2015.

Valdes, C; Capobianco, E. *Methods to Detect Transcribed Pseudogenes: RNASeq Discovery Allows Learning through Features*. Methods in Molecular Biology, Volume 1167, 2014, pp.157–183.

Valdes, C; P, Seo; N, Tsinoremas; & J, Clarke. *Characteristics of Cross-Hybridization and Cross-Alignment of Expression in Pseudo-Xenograft Samples by RNA-Seq and Microarrays*. Journal of Clinical Bioinformatics, 3, 2013, 8.

Valdes C. (July 2019) *Flint: Large Scale Microbiome Profiling in the Cloud*. Paper presented at the proceedings of the Intelligent Systems for Molecular Biology, and the European Conference on Computational Biology. Basel, Switzerland.