



# Systems Interrogation of Host-Microbiome Immunomodulation and Metabolism

## Citation

Luber, Jacob Mayne. 2020. Systems Interrogation of Host-Microbiome Immunomodulation and Metabolism. Doctoral dissertation, Harvard University Graduate School of Arts and Sciences.

## Permanent link

<https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37368931>

## Terms of Use

This article was downloaded from Harvard University's DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at <http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA>

## Share Your Story

The Harvard community has made this article openly available.  
Please share how this access benefits you. [Submit a story](#).

[Accessibility](#)

HARVARD UNIVERSITY  
Graduate School of Arts and Sciences



DISSERTATION ACCEPTANCE CERTIFICATE

The undersigned, appointed by the  
Department of the Division of Medical Sciences  
have examined a dissertation entitled  
Systems Interrogation of Host-Microbiome Immunomodulation and Metabolism

presented by Jacob M. Lubber

candidate for the degree of Doctor of Philosophy and hereby  
certify that it is worthy of acceptance.

Signature  \_\_\_\_\_


Typed name: Prof. Stephan Kissler

Signature  \_\_\_\_\_  
Jose Ordovas-Montanes (Aug 13, 2020 09:26 EDT)

Typed name: Prof. Jose Ordovas-Montanes

Signature  \_\_\_\_\_

Typed name: Prof. Jeffrey Moffitt

Signature  \_\_\_\_\_

Typed name: Prof. Morgan Langille

Signature \_\_\_\_\_

Typed name: Prof.

Date: August 10, 2020

# Systems Interrogation of Host-Microbiome Immunomodulation and Metabolism

A DISSERTATION PRESENTED

BY

JACOB M. LUBER

TO

THE DIVISION OF MEDICAL SCIENCES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN THE SUBJECT OF

BIOMEDICAL INFORMATICS

HARVARD UNIVERSITY

CAMBRIDGE, MASSACHUSETTS

AUGUST 2020

©2020 – JACOB M. LUBER  
ALL RIGHTS RESERVED.

## Systems Interrogation of Host-Microbiome Immunomodulation and Metabolism

### ABSTRACT

In this dissertation, I interrogate how scaling computational systems for complex ‘omics problems efficiently can lead to novel biological discoveries in the context of both how the microbiome modulates host metabolism and how the immune system responds to cancer. In Chapter 2, the method *Aether* is introduced. *Aether* is a tool that allows for intelligence bidding on cloud compute to reduce the cost of computational tasks in genomics by up to .7 orders of magnitude. In Chapter 3, *Aether* is utilized to handle *de novo* assembled meta‘omic data at massive scale to help yield the discovery a novel microbe present in the stomach of professional athletes. Chapter 4 shows how working with complex single cell sequencing data of immune cells at scale can yield novel insight into tumors non-invasively through the creation of machine learning algorithms that can predict whether a CD8<sup>+</sup> T cell in blood is in a shared clonal lineage as T cells in tumor. Taken together, these projects demonstrate the power of coupling experimental design with computation at scale.

# Contents

1	INTRODUCTION	I
1.1	Leveraging Computation at Scale for Experimental Biology . . . . .	2
1.2	Next Generation Sequencing . . . . .	2
1.3	Sequencing The Microbiome . . . . .	3
1.4	Single Cell Sequencing in Immunology . . . . .	5
1.5	Rotation Projects . . . . .	6
1.6	Literature Reviews . . . . .	II
2	SCALING CLOUD COMPUTING FOR ‘OMICS	23
2.1	Introducing <i>Aether</i> . . . . .	24
2.2	Cloud Compute Markets . . . . .	25
2.3	Validation of Approach . . . . .	28
2.4	Advantages of <i>Aether</i> . . . . .	32
2.5	General Implementation . . . . .	33
2.6	Bidding Algorithm . . . . .	34
2.7	Comparisons . . . . .	35
2.8	Optimization Problem Specification . . . . .	37
2.9	Conclusions . . . . .	39
2.10	Other Directions . . . . .	39
3	A PERFORMANCE ENHANCING MICROBE	40
3.1	Applying <i>Aether</i> . . . . .	41
3.2	Background on Computational Methods to Analyze the Microbiome . . . . .	42
3.3	Gut <i>Veillonella</i> Abundance is Significantly Associated with Marathon Running . .	44
3.4	<i>V. atypica</i> Gavage Improves Treadmill Run Time in Mice . . . . .	45
3.5	The Athlete Gut Microbiome is Functionally Enriched for the Metabolism of Lactate to Propionate Postexercise . . . . .	52
3.6	Serum Lactate Crosses the Epithelial Barrier into the Gut Lumen, and Colorectal Pro- pionate Instillation is Sufficient to Enhance Treadmill Run Time . . . . .	70
3.7	Coupling Computation and Experimentation . . . . .	72
3.8	Computational Methods . . . . .	74
3.9	Statistics . . . . .	82

4	SINGLE CELL IMMUNOLOGY OF CANCER	85
4.1	Single Cell Landscapes of Blood and Tumor . . . . .	86
4.2	Current Understanding of the T Cell Receptor Repertoire . . . . .	88
4.3	Using Machine Learning to Make Predictions About Clonal T Cell Lineages . . . . .	89
4.4	Computational Methods . . . . .	97
4.5	Shiny App . . . . .	99
4.6	Applications of the Shiny Application . . . . .	101
5	CONCLUSION	109
5.1	Failed Projects . . . . .	110
5.2	Influence of Dissertation Projects on the Field . . . . .	118
5.3	Conclusions . . . . .	120
5.4	Future Directions . . . . .	121
	APPENDIX A <i>AETHER</i> : SCALING CLOUD COMPUTE	122
A.1	Data Availability . . . . .	122
A.2	Tutorial . . . . .	123
A.3	Command Line Options . . . . .	123
A.4	Funding . . . . .	126
	APPENDIX B ATHLETE MICROBIOME PROJECT	127
B.1	Wet Lab Methods . . . . .	127
B.2	Data Availability . . . . .	133
B.3	Supplementary Tables . . . . .	134
B.4	Code Availability . . . . .	134
B.5	Unprocessed Western Blot . . . . .	134
B.6	Funding . . . . .	136
B.7	Author Contributions . . . . .	136
	APPENDIX C SINGLE CELL CANCER IMMUNOLOGY	138
C.1	Wet Lab Methods . . . . .	138
C.2	Funding . . . . .	142
	APPENDIX D CODE	143
D.1	Selected Code Examples . . . . .	143
	REFERENCES	177

# Listing of figures

1.1	Overview of the HiGlass Web Application . . . . .	7
1.2	Comparing The BGISEQ-500 and Illumina HiSeq 2000 for Reference Based Metagenomics . . . . .	9
1.3	Dispatch: Microbial BGC-derived Secondary Metabolites Modulate Host Proteases	15
1.4	Focus: Immunoglobulin Response to the Commensal Microbiota in Human TiD	20
2.1	AWS Instance Price History Example . . . . .	26
2.2	Illustration of Linear Programming Approach . . . . .	27
2.3	General Overview of <i>Aether</i> Infrastructure. . . . .	28
2.4	Overview of De Novo Assembly Process . . . . .	29
2.5	<i>Aether</i> Cost Comparisons . . . . .	30
2.6	Expected Value of Worst Case Bidding Outcomes . . . . .	31
3.1	Longitudinal Phylum-level Relative Abundance in Marathon Runners . . . . .	46
3.2	Longitudinal <i>Veillonella</i> Relative Abundance in Marathon Runners . . . . .	47
3.3	GLMMs Predicting Longitudinal <i>Veillonella</i> Relative Abundance . . . . .	48
3.4	Confidence Intervals for GLMM Predictions . . . . .	49
3.5	Statistical Validations of Association Between <i>Veillonella</i> Abundance and Marathon Running . . . . .	50
3.6	Microbiome Composition in Control Subjects . . . . .	51
3.7	<i>V. atypica</i> Gavage Improves Treadmill Run Time in Mice . . . . .	53
3.8	Statistical Validations of the Association Between <i>Veillonella</i> Gavage and Mouse Treadmill Run Time . . . . .	54
3.9	AB/BA Crossover Study Results Segregated by Individual Mouse . . . . .	55
3.10	Difference in Maximum Treadmill Run Time . . . . .	56
3.11	Visualization of GLMMs Showing Treadmill Run Time Improvement in Crossover	57
3.12	Mouse Serum Cytokine Response . . . . .	58
3.13	GLUT <sub>4</sub> Measurement in Mice Following Gavage . . . . .	59
3.14	Metagenomic Analysis of Athlete Gut Microbiome Samples . . . . .	60
3.15	Enzyme-level Abundance Analysis of the Methylmalonyl-CoA Pathway . . . . .	61
3.16	Differential Expression Analysis of Gene Families in the Methylmalonyl-CoA Pathway	62
3.17	Lactate Catabolism Across the Bacterial Tree of Life . . . . .	63
3.18	Prevalence of Methylmalonyl-CoA Pathway Components in Lactate Catabolizing Microbes . . . . .	64
3.19	SCFA Detection in Various Spent Media . . . . .	64



3.20	Schematic of $^{13}\text{C}_3$ Flux-tracing Experiment . . . . .	65
3.21	$^{13}\text{C}_3$ Abundance . . . . .	66
3.22	Normalized $^{13}\text{C}_3$ Abundance . . . . .	67
3.23	Intracolonic Propionate Infusion Increases Treadmill Runtime . . . . .	68
3.24	Proposed model of the microbiome–exercise interaction . . . . .	69
3.25	Lactate Clearance Following IP Injection in Mice . . . . .	72
4.1	Overview of Paired Single Cell GEX + TCR Data From Mouse . . . . .	86
4.2	Integrated Mouse Blood Single Cell Landscape . . . . .	89
4.3	Integrated Mouse Tumor Single Cell Landscape . . . . .	90
4.4	Visualization of “Tumor Matching” Cells in Blood . . . . .	91
4.5	Visualization of Blood Matching Cells in Tumor . . . . .	92
4.6	Predicting Tumor Matching Cells From All GEX Data . . . . .	93
4.7	Visualization of Blood Matching Cells in Tumor . . . . .	94
4.8	Gene Contribution to Classifier Performance . . . . .	94
4.9	Flow Cytometry Validation of Predicted Cell Surface Markers . . . . .	95
4.10	Biological Function of Top Predicted Surface Markers . . . . .	96
4.11	CITE-Seq Experiment Results . . . . .	97
4.12	Viewing Clustering in the Shiny Application . . . . .	101
4.13	Viewing T Cells in Shared Clonal Lineages Between Tissues in the Shiny Application	102
4.14	Viewing Marker Expression in the Shiny Application . . . . .	103
4.15	Viewing Clonal Expansion in the Shiny Application . . . . .	104
4.16	Viewing Number of Genes in the Shiny Application . . . . .	105
4.17	Viewing T Cell Clones in the Shiny Application . . . . .	106
4.18	Marker Expression Across Tissues . . . . .	107
4.19	Clonal Expansion Plot . . . . .	108
4.20	Coxcomb Plot Showing TCR Repertoire . . . . .	108
5.1	Improving Microbial Scaffolds with Long Reads . . . . .	111
5.2	Detecting Horizontal Gene Transfer Events Using “Chimeric” Long Reads . . . . .	112
5.3	Correction of Reference Genomes to Incorporate Horizontal Gene Transfer Events	113
5.4	Attempted Recovery of Low Abundance Microbial Genomes . . . . .	115
B.1	Unprocessed Western Blot for Figure 3.13 . . . . .	135

I DEDICATE THIS DISSERTATION TO MY PARENTS AND SISTER: KATIE, PHIL AND DIANA.

# Acknowledgments

I owe the privilege of completing this dissertation to a practically uncountable number of people, and I apologize if I have somehow neglected to acknowledge anyone. I have received excellent mentorship from two brilliant dissertation advisors, Meromit Singer and Aleksander Kostic. Without my undergraduate research advisors Matt Hibbs and Carol Bult, I would never have pursued a PhD. None of my computational method development would have been possible without singular experimental and/or clinical collaborators such as Jonathan Scheiman, Ted Chavkin, Kristen Pauken, Kelly Mahuron, Adil Daud, and Michael Rosenblum. Many productive collaborations have resulted from discussions with Marsha Wibowo, Braden Tierney, Zhen Yang, Juhi Kuchroo, Osmaan Shahid, Kaitlyn Lagattutta, and Conor Delaney. I am grateful to the NIH, NSF, and Amazon for funding my PhD. I owe my solid fundamentals in CS theory and programming to Mark Lewis, Seth Fogarty, Paul Myers, and Albert Jiang. Chirag Patel has been an unparalleled computational collaborator. My dissertation committee that has been comprised of Maha Farhat, George Church, Arlene Sharpe, Souyma Raychaudhuri, and Stephan Kissler has offered me excellent advice. My cohort of Jake, Ryan, Eric, Zach, Qinbo, Heather, and Gaurav has been incredibly supportive. I am particularly indebted to Stephan Kissler for going above and beyond in helping me navigate the challenges of completing a PhD. I have incredible friends that have made me a better person for close to a decade: Evan, Evan, Daniel, Zach, and Diego. In Boston my tight circle of friends has al-

ways been there when I needed to get away from work: Ángel, Chris, and Wilson. In Eda, I have the kindness and most motivated partner that one could ask for. Finally, I owe everything to my family: Diana, Phil, and Katie.

# 1

## Introduction

COMPUTATIONAL TRAINING HABITUATES ONE TO BREAK DOWN COMPLEX PROBLEMS INTO  
MANAGEABLE CHUNKS in order to find the fastest, most logical, and elegant solution.\* The primary  
focus of my PhD has been building tools to handle large scale genomic data and utilizing computa-

---

\*Portions of this chapter have been previously published in *Nature*, *Genome Biology*, *Science Immunology*, *Current Biology*, and *GigaScience*.

tional and machine learning approaches to integrate messy human data with more straightforward data from model organisms. Applying the logical premise of Occam's razor to applications in biology and genomics is inherently dangerous as evolution does not design things neatly.

## 1.1 LEVERAGING COMPUTATION AT SCALE FOR EXPERIMENTAL BIOLOGY

The first two projects of my dissertation (Chapter 2 and Chapter 3) focus on computational methods to handle large scale meta-omic data. The third project of my dissertation focuses on using single cell sequencing to study clonal lineages of T cells in blood and tumor (Chapter 4). In this introduction chapter, I go over some basic background information covering Next Generation Sequencing, how this relates to the microbiome, background on single cell sequencing in cancer immunology, some small literature reviews on these topics that I previously published (Luber & Kostic, 2017, 2019), as well as some minor projects I contributed to early in my PhD.

## 1.2 NEXT GENERATION SEQUENCING

While these topics are seemingly very different, a common thread that links them is that all require optimized computation to account for the fact that the rate in which sequencing technologies are decreasing in cost is far greater than the rate that computational resources are decreasing in cost (Muers, 2011). Next generation sequencing methods commonly rely on recovering "shot-gun" fragments of genomic material, which can then either be mapped to reference genomes or *de novo* assembled into larger contiguous regions ('contigs'). Computational methods in metagenomics lag behind

genomics. In the context of the microbiome, as dozens or hundreds of unique microbial strains exist together in environmental samples that often have many genes that are evolutionarily similar to each other, it is difficult to associate sequenced genomic material with a specific strain, and therefore it is much harder to reconstruct the entire community from sequencing data than it is to reconstruct a single human genome.

### 1.3 SEQUENCING THE MICROBIOME

Gut microbes play a critical role in human health through production of secondary metabolites, small RNA interactions with the host, and perturbation of colonization trends of other species (Devlin & Fischbach, 2015; Magnúsdóttir et al., 2017). For the past decade, the field of metagenomics has made a series of tradeoffs to deal with the complexity of parsing shotgun sequencing data that encompasses thousands of potential species in one sample that has resulted in a lack of ability to recover low abundance microbes.

The microbes that constitute an average human gut microbiome are difficult to study as estimates of the number of species present have increased during the past decade from being in the hundreds to being in the thousands (Lloyd-Price et al., 2016). The true range of their abundances is unknown as there are possibly microbes whose genome has never been recovered but whose secondary metabolites still functionally interact with the host (see more in depth literature review on this later in the introduction chapter). Most metagenomics analysis to date almost exclusively relies on tools that utilize alignment to a limited set of a few thousand reference genomes previously obtained from

isolates (Lloyd-Price et al., 2017). The methods derived from this philosophy of align then analyze are extremely memory efficient and allow for easy analysis of large patient cohorts with reasonable amounts of computational resources (Quince et al., 2017; Truong et al., 2015). A small number of metagenomics studies to-date such as metaHIT have relied more on algorithms that utilize de novo assembly methods on relatively smaller datasets, which require much more computational resources but yield more informative results (Qin et al., 2010; Li et al., 2014). The assembly approach is better overall than the alignment approach for finding new microbes under complex conditions but there remains room for improvement in terms of data standardization, scale of databases, and studies of niches other than the gut microbiome (Nielsen et al., 2014). Specifically, the size of cohorts that can be studied with the assembly approaches are limited by computational considerations.

The continual reliance on reference based alignment methods means that even if complete metagenomes are recovered, the tools to computationally analyze them have not been developed yet. There are approximately the same number of microbes in a human than there are human cells (Sender et al., 2016a,b). Full characterization of the gut microbiome remains impossible without developing methods to account for the complexity of the unannotated portions of the microbiome not present in reference genomes; Chapter 2 and Chapter 3 focus on developing such methods in specific contexts related to scaling computation as well as discovering novel host associated function in microbial gene catalogs.



#### 1.4 SINGLE CELL SEQUENCING IN IMMUNOLOGY

Tumor immunotherapy has revolutionized the clinical course of cancer treatment, harnessing the power of the immune system to eliminate malignant cells without the need for more toxic treatment methods (Giladi & Amit, 2018; Keir et al., 2008; Pauken et al., 2016; Sharpe & Pauken, 2018; Daud et al., 2016; Lubber, 2015). There is significant clinical interest in tracking host immune responses to cancer immunotherapy in the peripheral blood since this site can be easily and repeatedly be sampled for immune monitoring purposes, and likely contains a significant population of tumor-antigen specific T cells that are en route to the tumor (Yost et al., 2019; Zhang et al., 2018). However, tracking tumor-associated immune responses in the peripheral blood has been challenging due to (1) the low number of tumor-specific T cells that are likely present in the blood at any given time and (2) challenges in tracking known tumor antigen-specific T cell responses using traditional methods (e.g. tetramers) (Ráki et al., 2007; Robert et al., 2014). Overcoming these obstacles to allow routine tracking of T cell populations that are actually responding to the tumor would provide a powerful tool to the field, allowing a focused analysis of the most relevant T cell populations during the anti-tumor response rather than relying on bulk analyses on T cells in a non-antigen specific way.

Peripheral blood serves as a potential window into the systemic anti-tumor immune response in patients. Blood can easily be repeatedly sampled, and there is therefore intense interest in using blood samples to track the evolution of anti-tumor responses and correlate changes in immune populations in the blood to tumor outcomes following immunotherapy. However, immune populations (e.g. T cells) extracted from blood are heavily diluted for signals of anti-tumor response

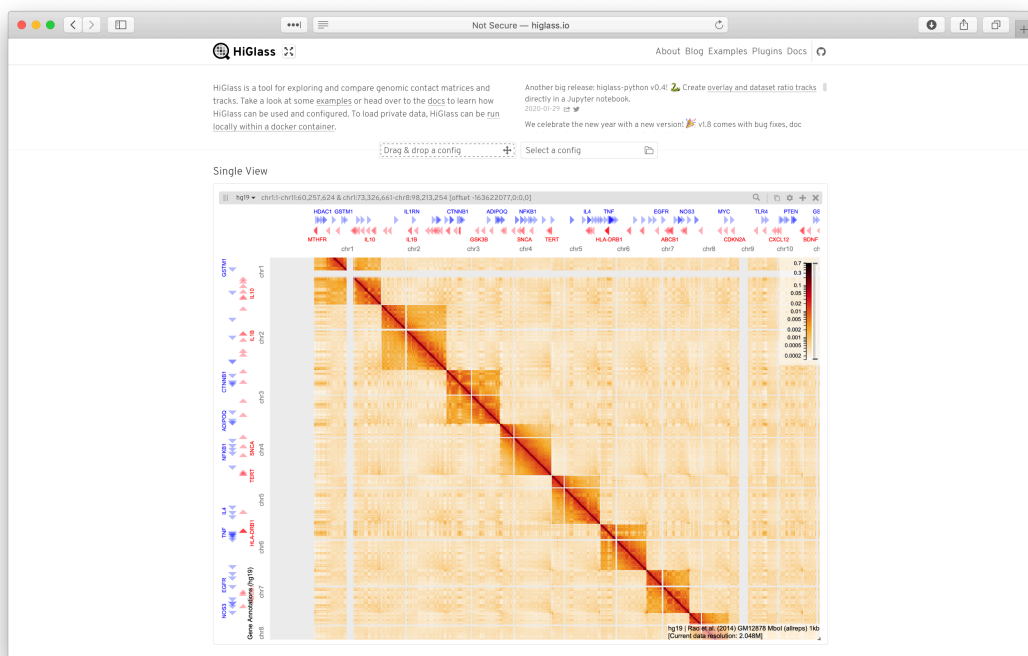
because only a small fraction of the cells in blood are relevant to an ongoing anti-tumor immune response. Chapter 4 asks whether the improved sensitivity afforded by single cell RNA sequencing (scRNAseq) paired with T cell receptor (TCR) sequencing could be used as a method to identify tumor-antigen specific T cells in the peripheral blood by using the TCR as a molecular barcode for T cells that share specificity with tumor-infiltrating lymphocyte (TIL) populations.

## 1.5 ROTATION PROJECTS

I was quite fortunate to have contributed to three projects during my rotations prior before beginning work in my dissertation labs that ended up being published. I have briefly summarized this work below.

### 1.5.1 4D NUCLEOME CONSORTIUM/HIGLASS

During my first rotation, I implemented some of the backend code for HiGlass <http://higlass.io> (Kerpedjiev et al., 2018). HiGlass is a fast visualization tool for large Hi-C and other genomic data sets. Hi-C data maps chromatin contact points genome wide, so HiGlass can be thought of as “google maps” for viewing the 3D structure of genomes (Figure 1.1). Many of the backend problems that I wrote code for handled how to efficiently serve “tiles” of contact map through an API when a user zoomed in or out. Through my work on this project, I also contributed to the 4D Nucleome (3D nucleomes over time) perspective piece in *Nature* (Dekker et al., 2017).



**Figure 1.1:** An example view of a Hi-C contact map in the HiGlass Web Application. HiGlass is a fast visualization tool for large Hi-C and other genomic data sets. HiGlass can be accessed at <http://higlass.io>.

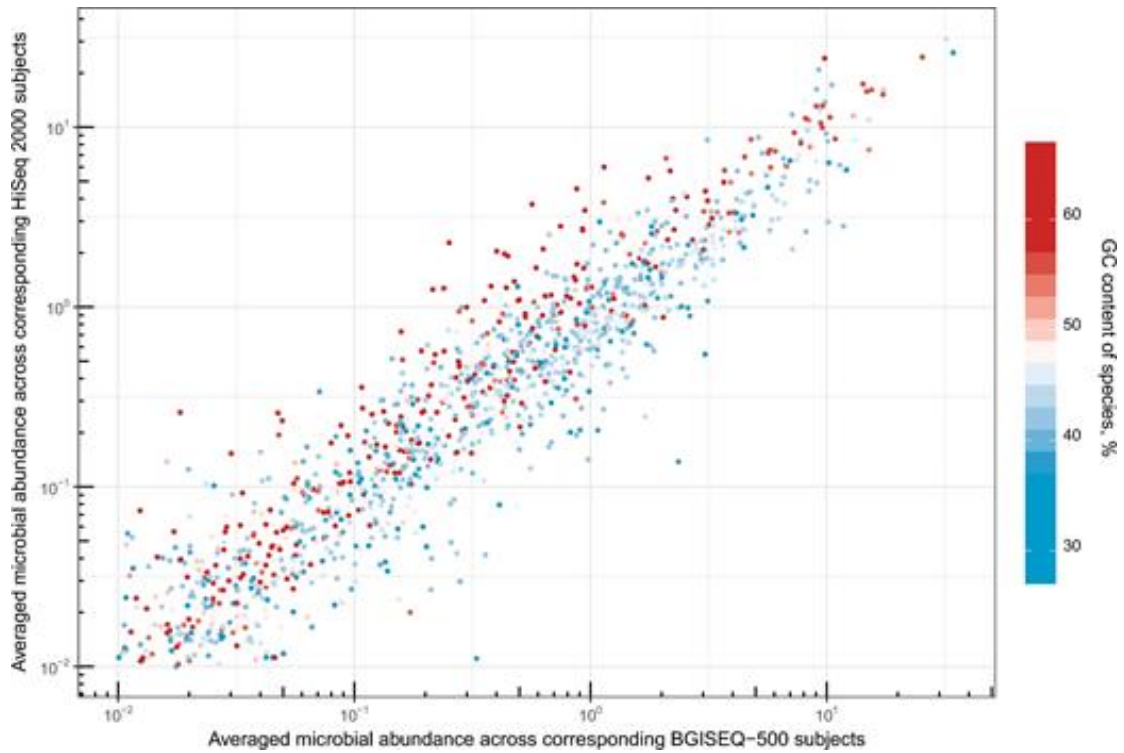
## 1.5.2 GENERALIZED LINEAR MODEL (GLM) REGRESSION ANALYSIS FOR RELATIVE SPECIES ABUNDANCE AND THEIR GC CONTENT ACROSS TWO PLATFORMS

For this project I worked with BGI to benchmark how their at the time unreleased BGISEQ-500 platform performed when using reference based metagenomic tools compared to the Illumina HiSeq 2000 (Fang et al., 2017).

A generalized linear model (GLM) regression analysis was conducted to investigate the associations between approximate relative species abundance and GC content across the 2 platforms. MetaPhlan2 (Truong et al., 2015) was utilized to generate estimates of relative abundance for each species in each sample. The GC content of each species was retrieved from NCBI. Samples were classified as either high/low abundance (above/below median = 0.2844), either high/low GC content (above/below median = 43.8%) with respect to sequencing platform (BGISEQ-500 or Illumina) (Figure 1.2). A log-linear model was used to model the total number of species in each of the 8 categories (abundance high/low, GC content high/low, BGI/Illumina), and a likelihood ratio test then suggested that the association between relative species abundances and their GC content did not vary across the BGISEQ-500 and HiSeq 2000 sequencing platforms ( $P = 0.323$ , chi-square test)

### DETAILED METHODS

Generalized linear model (GLM) regression was used to assess whether the association between relative abundance and GC content varied between the BGISEQ-500 and the HiSeq 2000 sequencing platforms. Samples were categorized into eight categories based on abundance (high or low), GC



**Figure 1.2:** Comparison of relative species abundance between BGISEQ-500 and HiSeq 2000. Averaged microbial abundance calculated with MetaPhlan2 across BGI replicates plotted against microbial abundance for the corresponding Illumina replicates for all samples. Species are colored by GC content.

content (high or low), and sequencing platform (BGISEQ-500 or HiSeq 2000). High relative abundance and GC content were classified as being above the median level (abundance median = 0.2844, GC content median = 43.8%). A log linear model was used to model the expected number of samples in each of the 8 categories. Let  $X$ ,  $Y$ , and  $Z$  be random variables describing GC content, relative abundance, and sequencing platform for each sample:

$$X \sim \text{GC content (0=below median, 1=above median), } i=0/1$$

$$Y \sim \text{Abundance (0=below median, 1=above median), } j=0/1$$

$$Z \sim \text{Sequencing (0=BGISEQ-500, 1=HiSeq 2000), } k=0/1$$

Then, the probability of being in category  $ijk$  is denoted as:

$$\pi_{ijk} = \mathbb{P}(X = i, Y = j, Z = k)$$

and the expected number of counts in category  $ijk$  is

$$\mu_{ijk} = \mathbb{E}[\eta_{ijk}] = n\pi_{ijk}$$

where  $n$  is the total sample size.

Two models were used to model expected counts. The first (model 1) models the null hypothesis that the association between relative abundance and GC content does not vary across sequencing

platform. The second (model 2) represents the alternative hypothesis that the association between relative abundance and GC content vary across sequencing platform.

Model 1:

$$\log(\mu_{ijk}) = \lambda + \lambda_i^X + \lambda_j^Y + \lambda_k^Z + \lambda_{ij}^{XY} + \lambda_{jk}^{YZ} + \lambda_{ik}^{XZ}$$

Model 2:

$$\log(\mu_{ijk}) = \lambda + \lambda_i^X + \lambda_j^Y + \lambda_k^Z + \lambda_{ij}^{XY} + \lambda_{jk}^{YZ} + \lambda_{ik}^{XZ} + \lambda_{ijk}^{XYZ}$$

A likelihood ratio test was used to test the significance of the addition of the three-way interaction term ( $\lambda_{ijk}^{XYZ}$ ). A significant test statistic would suggest that the null hypothesis that the association between relative abundance and GC content does not vary across the BGISEQ-500 and the HiSeq 2000 sequencing platforms can be rejected.

## 1.6 LITERATURE REVIEWS

A general theme of my dissertation is bridging computation, metagenomics, and immunology. The below section contains two small published literature reviews from early in my PhD covering these topics.

### 1.6.1 DISPATCH: GUT MICROBIOTA: SMALL MOLECULES MODULATE HOST CELLULAR FUNCTIONS

The human gut metagenome has been recently discovered to encode vast collections of biosynthetic gene clusters (BGCs) with diverse chemical potential, almost none of which are yet functionally

validated (Cimermanic et al., 2014; Donia et al., 2014).<sup>†</sup> Recent work by Chun-Jun Guo et al. elucidates common microbiome-derived BGCs encoding peptide aldehydes that inhibit human proteases (Guo et al., 2017).

Biosynthetic gene clusters (BGCs) are tightly clustered groups of genes in bacteria that encode pathways capable of producing small molecule natural products without cellular machinery such as ribosomes (Cimermanic et al., 2014). The array of diverse natural products produced by BGCs are vast; current databases hold more than one million predicted BGCs, a substantial fraction of which are found in the human microbiome (Medema et al., 2015; Hadjithomas et al., 2015). A class of antibiotics in clinical trials was simultaneously discovered to be produced by a human microbiome-encoded BGC, which suggests that understanding and subsequently engineering BGCs could yield highly effective new drug pipelines (Donia et al., 2014). However, small perturbations in the genomes encoding these biosynthetic pathways can yield radically different functional secondary metabolites (Medema et al., 2011). This biochemical diversity is achieved through small genetic perturbations that yield chemical variants in the resulting metabolites, which makes interpreting BGCs both an interesting computational problem in terms of combinatorics as well as a challenge to experimentally validate.

Research into secondary metabolites thus far has aimed almost exclusively to provide a window into microbe-microbe interactions in environmental organisms (Li & Vederas, 2009; Courtois et al., 2003). Guo et al. attempt to identify the most commonly shared BGC family in the human microbiome by mining the NIH Human Microbiome Project Phase I dataset, a large study profiling the

---

<sup>†</sup>Portions of this section were previously published in *Current Biology*



“healthy” human gut microbiome in Americans, and then subsequently determine the function of this BGC family (Guo et al., 2017; Human Microbiome Project Consortium, 2012). This synergistic approach couples computational analysis of large metagenomic datasets, synthetic biology to express genetic elements from uncultured bacteria in laboratory strains, and chemistry to identify and interpret the BGC products. The BGC family identified consists of thirty-seven similar clusters shared among ninety percent of individuals in the cohort that are also rarely present in environmental bacterial isolates, which suggests a possible direct role for the BGC products in human physiology. However, the gut metagenome is massively complex and elucidating fine grained interactions with the host in a sea of trillions of bacteria is far from easy.

For fourteen members of said family, the resulting product was reconstructed by either directly cloning from native hosts or synthesizing the entire BGC de novo and transforming the construct into either *E. coli* or *Bacillus subtilis*. Several of these BGCs, derived from Gram-positive bacteria, were not only transcribed but shown to produce unique small molecule products in *E. coli* driven by an *E. coli* promoter, an encouraging result that meant it is possible to have functioning BGCs even when they’re expressed in vastly divergent bacteria (Guo et al., 2017). With these fourteen BGCs reconstructed in vivo, the next step was to elucidate their cognate small molecule product. Liquid Chromatography-Mass Spectrometry was used to identify new peaks that show up in the BGC product—seven out of the fourteen BGCs produced new peaks that corresponded to thirty-two unique compounds (Guo et al., 2017).

With these compounds identified, two important questions remained: Are these compounds produced in the native host bacteria? Which of these compounds have activity that interact with

host function and which are products of degradation or other artifacts? Through careful experimental validation, dipeptide aldehyde fit the clear role of the active product as it is stable long enough to be active, known to be a cell-permeable protease inhibitor, and subsequently demonstrated cell protease inhibition activity (Guo et al., 2017).

How does the inhibition of a human cellular process by a natural product produced non-ribosomally by gut microbes potentially enhance mutualism, if in fact it does? It seems highly counterintuitive that inhibiting host cellular proteases that break down metabolic debris and unnecessary proteins would have a positive effect on mutualism. However, one of the dipeptide aldehydes, Phe-Phe-H, has specific activity against cathepsins, which are an important component of the antigen presentation machinery. This suggests that inhibition of hosts proteases may facilitate the ability of mutualists to reside in the gut without being targeted by the immune system. Figure 1.3 illustrates this concept. The discovery of a family of BGCs that are present in the vast majority of healthy human adults that are also generally unique to humans suggests that the resulting product from this BGC plays a direct role in signaling to the host. If this finding is validated, it represents the first example of a microbe-derived BGC responsible for maintaining the symbiosis between humans and their microbiome.

Natural products from BGCs are an important source of FDA-approved drugs. However, these are all derived from environmental microbes and plants rather than human-associated microbes (Li & Vederas, 2009; Courtois et al., 2003). The conventional focus of the field has been on discovering gene products with antimicrobial properties which can be segued into new antibiotic pipelines (Heidrich et al., 1998; Lubelski et al., 2008). The Fischbach group has previously found large numbers

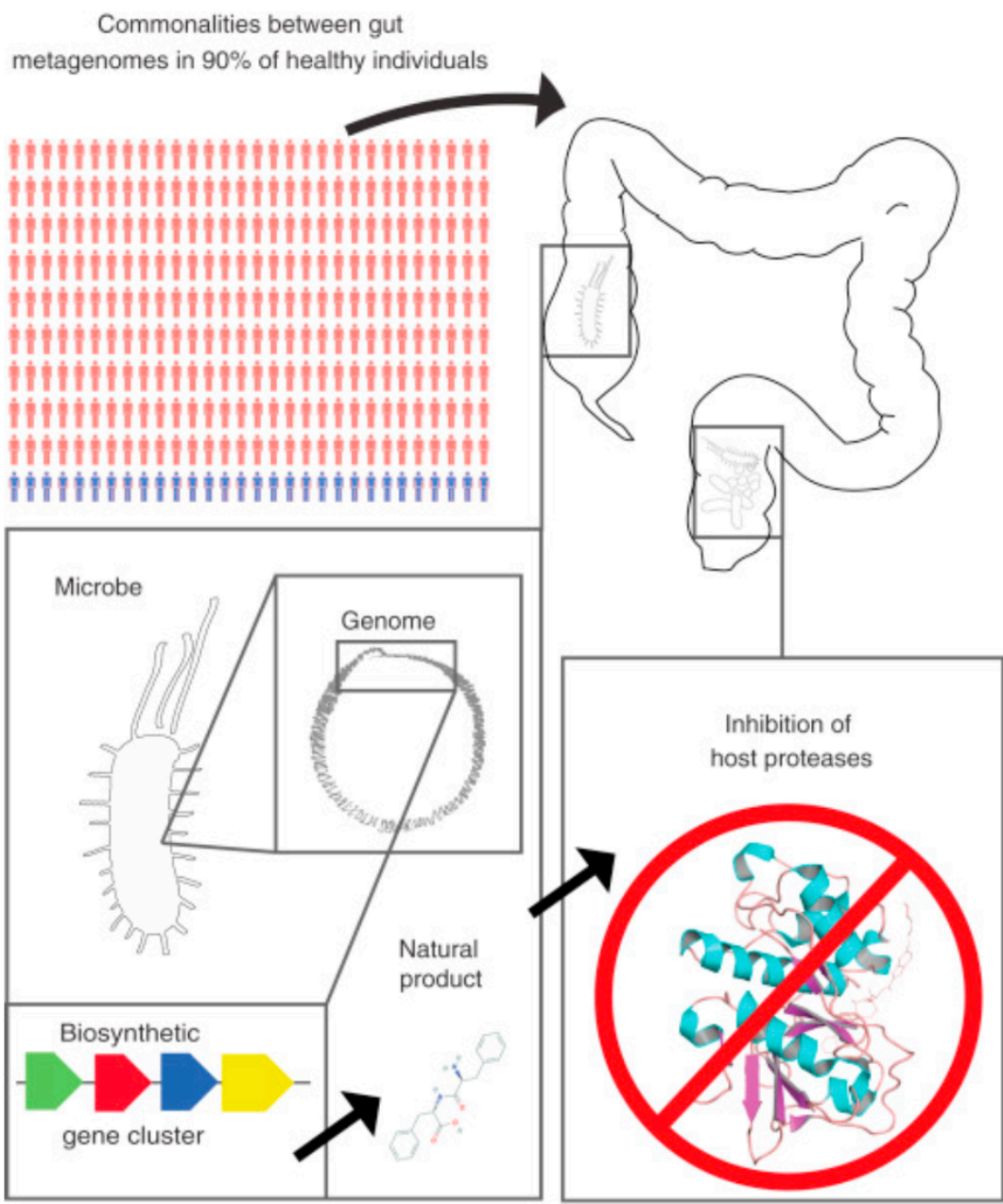


Figure 1.3: Microbial BGC-derived Secondary Metabolites Modulate Host Proteases

of BGCs in the human microbiome and subsequently found the resulting product[s] to modulate host function in some way (Cimermancic et al., 2014; Zimmermann & Fischbach, 2010). One BGC identified in this prior work that is commonly present in the vaginal microbiome was shown to endogenously produce lactocillin, of which there was a synthetic version simultaneously in clinical trials (Donia et al., 2014). In extending this type of analysis to healthy cohorts, Fishbach's group has clearly shown that small-molecule products from gene clusters in the human gut microbiome that are widely shared across healthy individuals exist. However, it is still not known how many other health-associated BGCs exist. On a molecular level, it is still not fully conclusive what dipeptide aldehydes actually do and whether or not they act on immune cells as hypothesized.

The evidence of widespread microbiome-encoded BGCs being present in healthy individuals and likely modulating human physiology is an important result, yet poses many challenges to further exploration. Gut microbes clearly have immunomodulatory and inflammatory effects on humans (Round & Mazmanian, 2009; Cullen et al., 2015). However, the discovery of a unary BGC family present across healthy individuals required a massive amount of data mining coupled with complex experimental validation. The groundbreaking methods used by Fischbach's group to analyze BGCs present across a healthy human cohort are necessary to prove their existence; the next step is to understand their role. Mining large metagenomic databases can yield putative common BGCs of large effect, but whether the products of these BGCs have additive effects remains an open question. To fully discern the role of the complete microbiome in modulating human health, bioinformatics must be further coupled with experimental design so that BGC calling algorithms inform experimentation and vice versa (Medema et al., 2012; Smanski et al., 2016). Understanding the functional

complexities of BGCs on health will require the utilization of 3rd generation long read sequencing as a first step for calling putative BGCs that are rare in gut microbial populations. BGC enzymes are highly conserved with strong homology across distant taxa. Additionally, horizontal gene transfer events for BGCs are likely. Therefore, short reads often cannot be uniquely mapped to BGCs even at extremely high coverage. Long reads and methods to assemble them that span the entire BGC represent a potential leap forward (Koren et al., 2017).

To understand the full functional landscape of BGCs, perturbing and engineering minimal communities of microbes (i.e. “toy” microbiomes) in model organisms will help reduce the complexity involved with the human microbiome, and make it possible to understand the complete metagenome, metatranscriptome, metaproteome, and metabolome of closed system. To perform perturbation at this scale, high throughput CRISPR based methods will likely need to be utilized, which presents other implicit challenges (Xu et al., 2015). Extensions of this initial work by Guo et al. will be the first steps in elucidating the function of poorly understood elements of the metagenome. Further efforts into understanding natural products produced by BGCs, and engineering microbes that produce them, could eventually yield a deeper understanding of how host-microbe interactions modulate human health.

## 1.6.2 FOCUS: A PERFECT STORM: GENETICS AND ANTI-COMMENSAL ANTIBODIES SHORE

### UP TYPE 1 DIABETES

HLA haplotypes in conjunction with serum anti-commensal antibody responses are predictive of type 1 diabetes progression.<sup>‡</sup>

Many recent studies have found associations between the gut microbiota and autoimmune diseases such as type 1 diabetes (T1D). In this issue of *Science Immunology*, studies by Paun et al. hint at possible mechanisms behind immunopathogenesis of T1D by associating HLA haplotypes with anti-commensal antibody (ACAb) responses and islet autoantibodies (IABs) in two distinct cohorts (Paun et al., 2019), and crucially observes a genetic determinant in these associations.

The hygiene hypothesis postulates that at the population level, the frequency of autoimmune disorders and diseases related to allergy are inversely correlated with the frequency of infectious disease (Strachan, 2000). Epidemiological associations support the hygiene hypothesis, but a causal mechanism has never been conclusively validated; a widely supported theory is that tolerance in Toll Like Receptors (TLRs) recognizing both commensals and pathogens leads to increased incidence of autoimmunity in specific populations based on their aggregate gut microbiome composition (Bach, 2018).

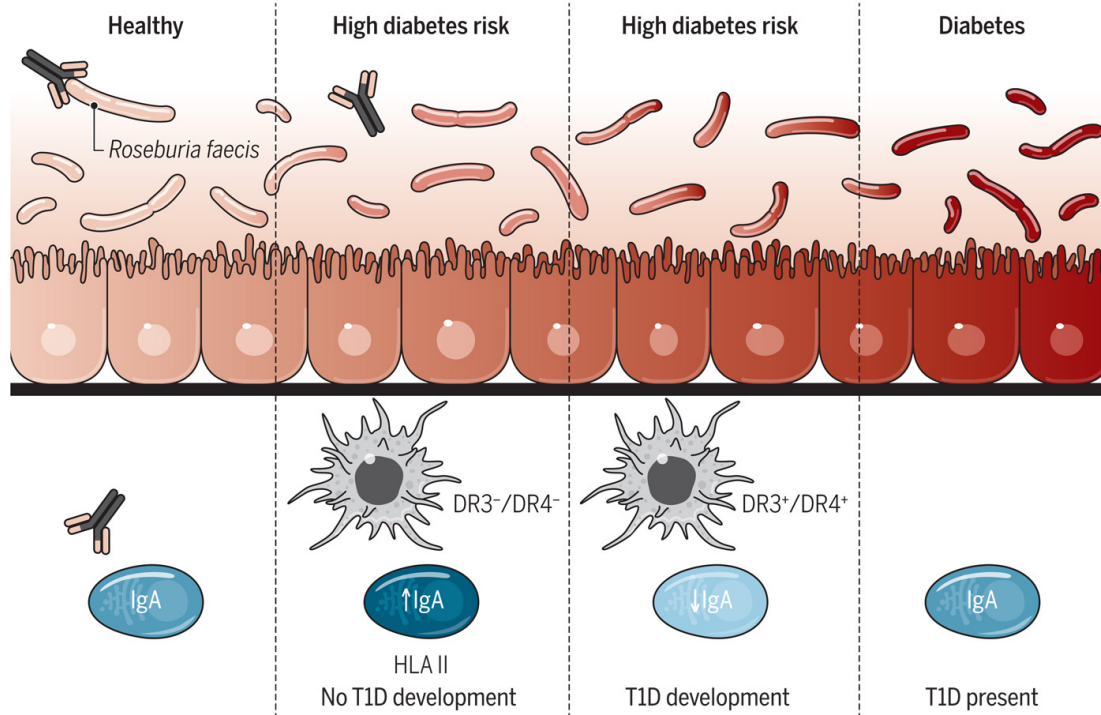
Recent work in the microbiome field has found associations between both low diversity in gut microbiota composition at the taxa level and longitudinal progression of autoimmune disorders like T1D (Kostic et al., 2016), as well as an association between autoimmunity and lipopolysaccharide (a

---

<sup>‡</sup>Portions of this section were previously published in *Science Immunology*

potential TLR4 agonist) from microbes in countries with lesser incidence of infectious disease (Vatanen et al., 2016). This association is not found in other countries (of shared common genetic background) with geopolitically limited genetic admixture that in contrast have comparatively higher incidence of infectious disease (Vatanen et al., 2016). These findings have recently nudged the microbiome field towards pursuing massive collections of metagenomes without genome or exome sequencing of component individuals in these cohorts, such as in the The Environmental Determinants of Diabetes in the Young (TEDDY) study (Vatanen et al., 2018; TEDDY Study Group, 2007). The association between HLA haplotypes, gut microbes, and IABs presented by Paun et al. in this issue raises the compelling point that more nuanced approaches are needed to account for genetic determinants in host-microbe interactions related to autoimmunity and implicitly posits that for sustained forward progress in elucidating autoimmune disease pathogenesis silos need to come down between the fields of computational metagenomics, human genetics, and experimental immunology.

In order to measure ACAb responses from serum, Paun et al. created an assay where serially diluted serum is combined with bacterial targets and fluorophore-labeled antibodies, after which flow cytometry is conducted to calculate a “response” index for each sample (Paun et al., 2019). This assay showed elevated ACAb responses to a diverse group of commensals in Crohn’s disease patients compared to healthy controls. The authors subsequently applied their assay to T1D cohorts: one simply comparing serum from pediatric T1D patients to age matched healthy controls and one comparing serum obtained during a pre-diabetic (seropositive for IABs but normal glucose levels) phase in at risk patients from the Type 1 Diabetes TrialNet (long term sample collections from family mem-



**Figure 1.4:** Using two separate cohorts of pediatric T1D, Paun et al. (Paun et al., 2019) found that the anticommensal antibody (ACAb) response against multiple bacteria is able to discriminate between healthy controls and individuals recently diagnosed with T1D. Although the ACAb response on its own cannot distinguish healthy controls from samples taken before the onset of diabetes, serum IgG2 antibodies against commensal bacteria in combination with the HLA DR3/DR4 haplotype were associated with future diagnosis of T1D.

bers of existing T1D patients) who also went on to develop T1D (Skyler et al., 2008). A summary of the underlying biology is shown in Figure 1.4.

In the first cohort comparing age matched pediatric T1D patients and healthy controls, the authors found significant differences in both IgG and IgA ACAb responses against a range of commensals and were able to use LDA to distinguish healthy controls from age matched T1D patients. When the assay was applied to the cohort comparing healthy controls and pre-diabetes TrialNet patients, no differences in ACAb responses to commensals were found between the two groups. However,



when HLA genotype information was included in the multivariate statistical comparison between the two groups, the cohorts stratified with certain ACAbs predicting future T1D status in a HLA haplotype dependent manner (Paun et al., 2019). The HLA locus encodes both the MHC complex and other machinery that assists with antigen processing and presentation; certain MHC class II haplotypes have large effect size in predicting T1D risk (Dendrou et al., 2018). The authors subsequently went on to show that in addition to IAB antigen specificity that is dependent on high risk HLA haplotypes, an association also exists between IAB antigen specificity and ACAb responses (Paun et al., 2019).

This association, while not validated to be causal, raises vital and perhaps incommensurable questions about how study design for future longitudinal cohorts aiming to look at host-microbiome interactions in T1D immunopathogenesis should be conducted. Perhaps the no holds barred approach of sequencing as many metagenomic samples as possible for greater statistical power in downstream analyses (Vatanen et al., 2018) is misapprehending crucial underlying biology in that human genetic determinants likely play a significant role in disease. Coupling results from experimental assays with clinical genotyping like Paun et al is laudable; their findings should serve as impetus to include either full exome or genome sequencing in future longitudinal studies looking at the interface between the metagenome and disease.

We have learned much from the non-obese diabetic mouse, the most widely used animal model of T1D, but unlike the human it has been cured of diabetes in over 700 different ways and has many nuanced differences in disease pathogenesis compared to humans (Jayasimhan et al., 2014). Therefore, understanding the human genetic determinants of how the microbiome affects T1D im-

munopathogenesis is critical in pursuing future treatments and cures for T1D . Integrating analyses of longitudinal human microbiome studies and human genetic determinants of disease to inform, characterize, modify, and improve experiments utilizing mechanistic animal models of autoimmune disease will be critical to developing treatments for T1D.

*No physical quantity can continue to change exponentially forever. Your job is delaying forever.*

Gordon Moore

# 2

## Scaling Cloud Computing For ‘omics

ACROSS BIOLOGY, WE ARE SEEING RAPID DEVELOPMENTS IN SCALE OF DATA production without a corresponding increase in data analysis capabilities. In this chapter, the design, implementation, and validation of the computational tool *Aether* is presented (Luber et al., 2017).\*

---

\*Portions of this chapter were previously published in *Bioinformatics*. This paper has an additional co-first author, Braden T. Tierney. Braden and I co-conceived the algorithm, I implemented the algorithm, and Braden conceptualized test cases where the algorithm could be benchmarked.

(<http://aether.kosticlab.org>) is an intuitive, easy-to-use, cost-effective and scalable framework that uses linear programming to optimally bid on and deploy combinations of underutilized cloud computing resources. This approach simultaneously minimizes the cost of data analysis and provides an easy transition from users' existing HPC pipelines.

## 2.1 INTRODUCING *AETHER*

Data accumulation is exceeding Moore's law, which only still progresses due to advances in parallel chip architecture (Esmailzadeh et al., 2013). Moore's law states that the every two years the number of transistors that can fit on a computer chip doubles, while the cost of the chip itself halves (Moore, 1965). Fortunately, the shift away from in-house computing clusters to cloud infrastructure has yielded approaches to computational challenges in biology that both make science more reproducible and eliminate time lost in high-performance computing queues (Beaulieu-Jones & Greene, 2017; Garg et al., 2011); however, existing off-the-shelf tools built for cloud computing often remain inaccessible, cumbersome, and in some instances, costly.

### 2.1.1 WHERE MOORE'S LAW FALLS SHORT IN BIOLOGY

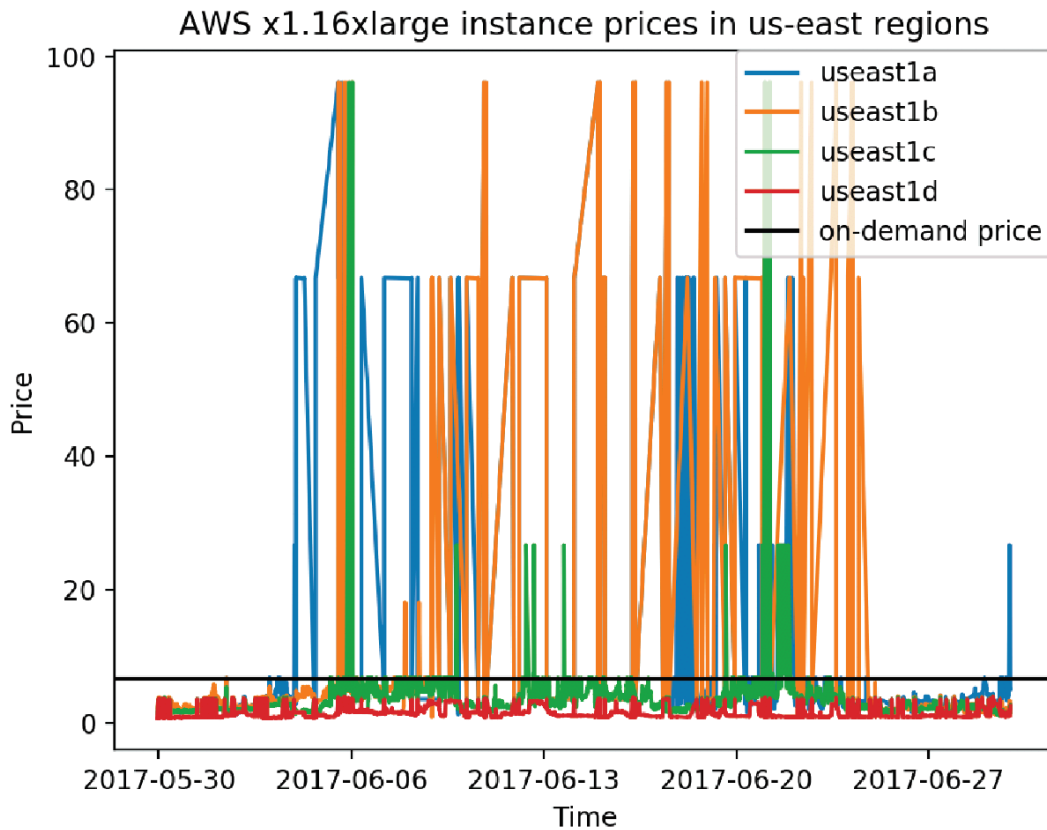
Solutions to parallelizable compute problems in computational biology are increasingly necessary; however, batch job-oriented cloud computing systems, such as Amazon Web Services (AWS) Batch, Google preemptible Virtual Machines (VMs), Apache Spark and MapReduce implementations are either closed source, restrictively licensed, or locked in their own ecosystems making them inaccessible to many bioinformatics labs (Shvachko et al., 2010; Yang et al., 2007). Other approaches for

bidding on cloud resources exist, but they neither provide implementations nor interface with a distributed batch job process with a backend implementation of all necessary networking (Andrzejak et al., 2010; Tordsson et al., 2012; Zheng et al., 2015).

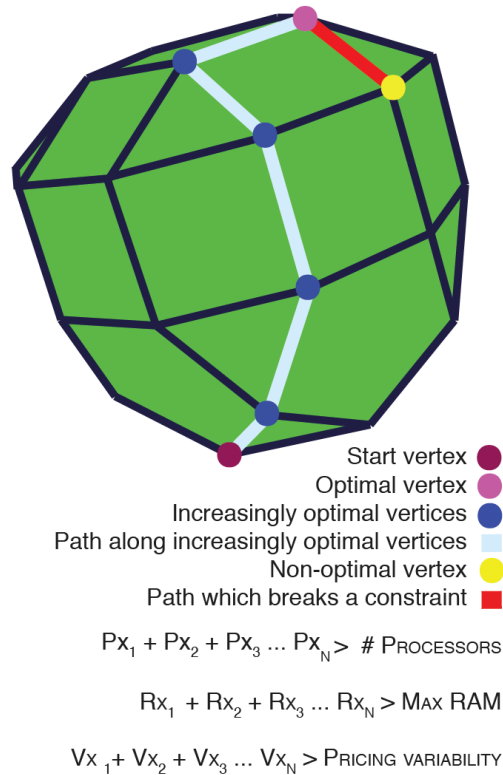
## 2.2 CLOUD COMPUTE MARKETS

Our proposed tool, *Aether*, leverages a linear programming (LP) approach to minimize cloud compute cost while being constrained by user needs and cloud capacity, which are parameterized by the number of cores, RAM, and in-node solid-state drive space. Specifically, certain types of instances are allocated to large web service providers (e.g. Netflix) and auctioned on a secondary market when they are not fully utilized (Zheng et al., 2015). Users bid amongst each other for use of this already purchased but unused compute time at extremely low rates (up to 90% off the listed price; <https://aws.amazon.com/ec2/pricing/>). However, this market is not without its complexities. For instance, significant price fluctuations, up to an order of magnitude, could lead to early termination of multi-hour compute jobs (Figure 2.1). Clearly, bidding strategies must be dynamic to overcome such hurdles.

*Aether* consists of bidder and batch job processing command line tools that query instance metadata from the vendor application programming interface (APIs) to formulate the LP problem. LP is an optimization method that simultaneously solves a large system of equations to determine the best outcome of a scenario that can be described by linear relationships. The *Aether* bidder, described in detail in the Supplementary Methods, generates and solves a system of 140 inequalities using the sim-



**Figure 2.1:** Pricing history of an x1.16xlarge EC2 Instance showcasing variability of an order of magnitude, in both directions, for spot prices.



**Figure 2.2:** Simplified example showing three constraints on a sample bidding approach minimizing an objective function  $cTx$  considering cost according to a system of constraints represented as inequalities.  $x_1, x_2, \dots, x_n$  represent the number of specific types of compute nodes to solve for. Each inequality represents a constraint and adds another dimension to the space which the simplex algorithm needs to traverse vertices in to find ideal solution. The green line represents the optimal solution.

plex algorithm (Figure 2.2). For the purposes of reproducibility, an implementation of the bidder using CPLEX is also provided as an optional command line flag.

Subsequently, the replica nodes specified by the LP result are placed under the control of a primary node, which assigns batch processing jobs over transmission control protocol, monitors for any failures, gathers all logs, sends all results to a specified cloud storage location, and terminates all compute nodes once processing is complete (Figure 2.3). Additionally, *Aether* is able to distribute

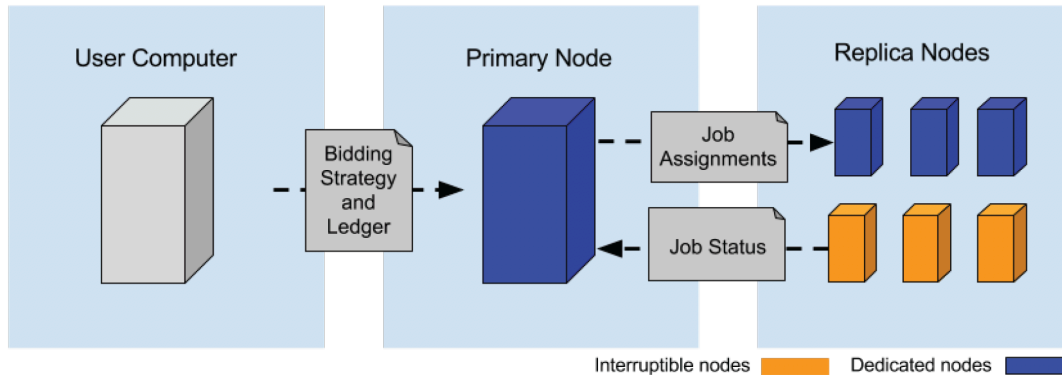


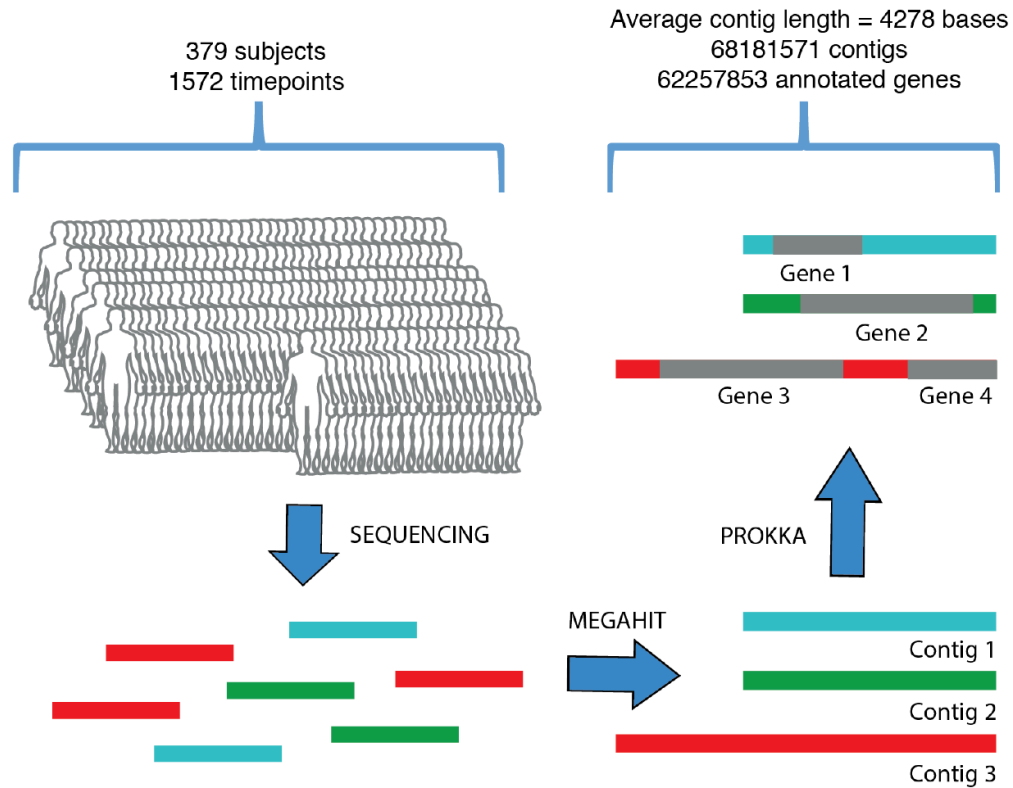
Figure 2.3: General overview of *Aether*.

compute across multiple cloud providers. Sample code for this is provided with the *Aether* implementation although it was not utilized in our reported tests due to cost feasibility. Our implementation runs on any Unix-like system; we ran our pipeline and cost analysis using AWS but have provided code to spin up compute nodes on either Microsoft Azure or on a user’s local physical clusters.

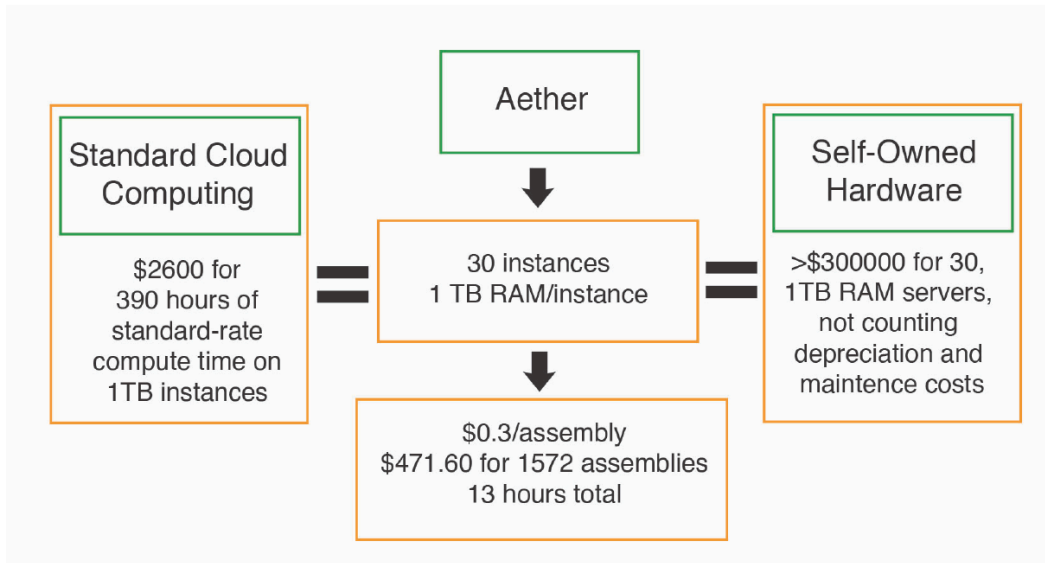
### 2.3 VALIDATION OF APPROACH

To test our bidding approach and batch job pipeline at scale, we used our framework to de novo assemble and annotate 1572 metagenomic, longitudinal samples from the stool of 222 infants in Northern Europe Figure (Figure 2.4) (Bäckhed et al., 2015; Kostic et al., 2016; Vatanen et al., 2016; Yassour et al., 2016). The sequencing data within datasets from the DIABIMMUNE consortium ranged from 4680 to 22, 435, 430 reads/sample with a median of 19, 020, 036 reads/sample. Assemblies were performed with MEGAHIT and annotations were done with PROKKA (Li et al., 2015; Seemann, 2014). Metagenomic data, typically shotgun DNA sequencing of microbial communities,





**Figure 2.4:** Overview of the assembly process. A total of 1572 fecal samples were collected and sequenced at various timepoints during the first 3 years of 379 individuals lives. These were assembled with MEGAHIT into 68,181,571 contigs. Across all samples, a total of 62,257,853 genes, 1,000,000 of which were unique, were then annotated using Prokka. Only contigs that were over 1000 bases long were used. The mean length of this group was 4278 bases.

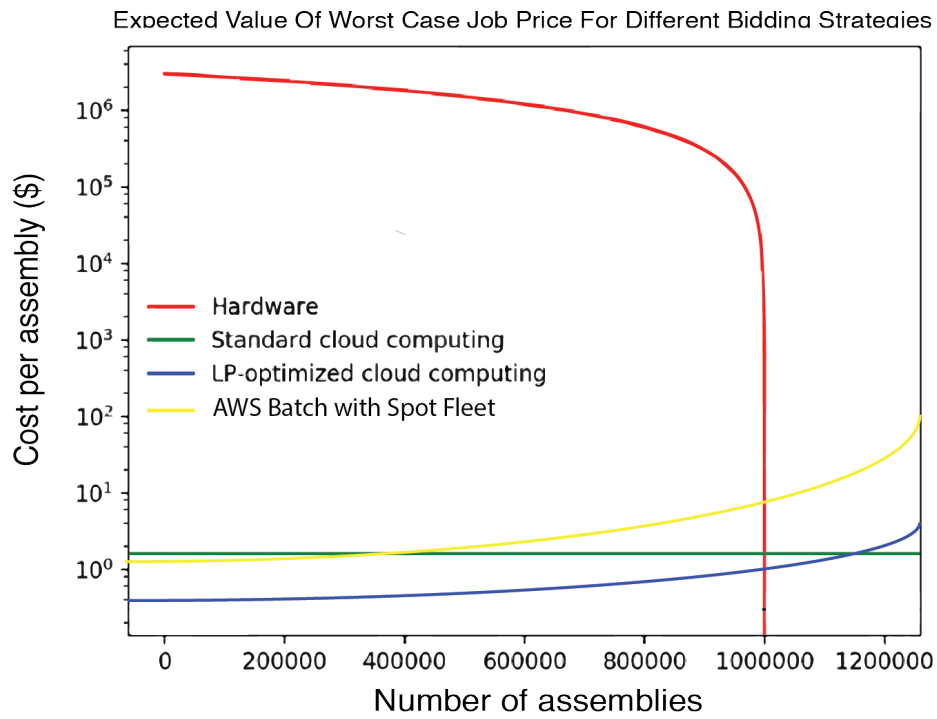


**Figure 2.5:** Cost comparison between *Aether*, standard cloud computing, and user-maintained hardware. Total assembly cost was 18% (\$471.60) of what it would have been using on-demand instances. We estimated the upfront cost of a server equivalent to those used to analyze the data being \$10,000. Given that we used 30 instances of these servers, the total cost of hardware would be \$300,000 according to pricing information from Penguin Computing and Dell, not counting system maintenance and depreciation.

is difficult to analyze because of the enormous amounts of compute required to naively assemble short sequence reads into large contiguous spans (contigs) of DNA. To accomplish our assemblies, our bidding algorithm suggested that the optimal strategy would be to spin up 30 TB of RAM across underutilized compute nodes. Our networked batch job processing module utilized these nodes for 13 h and yielded an assembly and annotation cost of US \$0.30 per sample (Figure 2.5).

Theoretically, the pipeline can complete in the time it takes for the longest sub-process (i.e. assembly in this case) to finish (7 h). Spinning up the same nodes for this long without a bidding approach would cost US \$1.60 per sample (Figure 2.6).

In order for on-site hardware to achieve the same cost efficiency as our pipeline, one would have



**Figure 2.6:** Plot of the Expected Value of worst case job price for different bidding strategies to showcase the advantage of Aether using a price “lookback” as a constraint for optimization.

to carry out on the order of 1 million assemblies over the lifespan of the servers, a practically insurmountable task (Figure 2.5). Such efficiency in both time and cost at scale is unprecedented. In fact, due to resource paucity, computational costs have forced the field of metagenomics to rely on algorithmic approaches that utilize mapping back to reference genomes rather than de novo methods (Truong et al., 2015).

#### 2.4 ADVANTAGES OF *AETHER*

Additional testing of *Aether* showed marginally better relative cost savings (compared to the assembly example) when tasked with aligning raw reads to the previously assembled genomes with BWA-MEM (Li & Durbin, 2009); this is not surprising as shorter computational tasks are less sensitive to the risks of early spot instance termination. Additionally, in simulated runs of the bidder incorporating pricing history from periods where ask prices were approximately an order of magnitude higher than normal on the east coast of the United States (Figure 2.1), *Aether* suggested utilization of different instance types that would have resulted in similar cost and time to completion as our actual run. To allow users to make optimal usage of these benefits, the ability to simulate bidding for different timeframes is included as a feature. By not having to potentially re-run analysis pipelines (due to being outbid on compute during runtime), we claim that utilizing *Aether* leads to a reduction of market inefficiencies. We have both qualitatively and empirically compared *Aether* to existing AWS tools such as AWS Batch and Spot Fleet Pricing (Figure 2.6). Additionally, where empirical validation of benefit was possible, we have iterated on previous work and incorporated strategies

such as basing a subset of constraints on service level agreements (Andrzejak et al., 2010). Future directions include training the bidding algorithm to predict its own effect on pricing variability when being utilized at massive scale as well as distributing compute nodes across datacenters when enough resources are being spun up to strongly influence the market.

## 2.5 GENERAL IMPLEMENTATION

**Implementation Details** Computational resources and monetary costs are mapped to each available instance type at run-time by querying the cloud providers web-based public APIs. To identify the ideal resource selection, we feed these data, along with constraints provided by the user, into our multi-objective optimization procedure. The user-defined set of jobs is subdivided into computational workloads according to the resources available to each node, and distributed across the worker nodes by a central server. In a single nodes workload, jobs are executed in parallel but may complete asynchronously. Upon completion of a job, the replica node notifies the central server, which then schedules another task for the replica. To prevent scheduling errors, we synchronized changes in the primary nodes job ledger, and used at-least-once message delivery. We controlled access to computational resources and accounts with AWS Identity & Access Management (IAM) security groups and Azure Identity and Access Management (IAM), which their respective providers recommend for authentication and authorization. Additional details regarding *Aether*'s implementation are available on the project website (<http://aether.kosticlab.org>).

## 2.6 BIDDING ALGORITHM

Due to pricing variability, it can be optimal to bid on non-auctioned instances in certain regions. To properly handle this case, we include additional linear constraints for both an instances on-demand and at-auction prices. The solution vector is bounded by the number of currently running instances as well as limits due to provider capacity. Finally, to avoid bidding on instances that will spike in price, the algorithm looks at pricing history and sets a final constraint corresponding to a users maximum tolerable pricing variability. For each run of the bidder, this system of 140 inequalities is converted to slack (standard) form and then solved with the simplex algorithm as implemented in Python's `scipy.linprog` library (Figure 2.2) (Jones et al., 2001). This naively outputs suggested compute bids as floats; obviously, a fraction of an instance is not a valid bid and generating integer solutions to linear programming problems is NP-hard. However, a true integer linear programming solution is not required, as the constraints still hold if the floor is taken from each bid, provided that preprocessing is done to remove underutilized instance types and those that cannot process a unary compute job. To reach this optimal integer pseudo-solution, the linear programming solver is run recursively such that these non-feasible fractional bids are iteratively removed. Additionally, adhering to the pricing variability constraint is not guaranteed to yield the optimal value, so the simplex algorithm is applied iteratively, setting the pricing variability from zero to the maximum specified value until either the optimal value is found or it is determined that there is no solution to the system. In the event of finding no solution, the user must re-run the program with a higher maximum cost. This approach results in a tractable average case runtime, which yields essentially instant bidding

suggestions given the small size of the system being solved.

The key code implementing this algorithm can be found in Appendix D.

## 2.7 COMPARISONS

Information about how to access a tutorial to run *Aether* as well as documentation of the command line interface (CLI) options are in Appendix A.

### 2.7.1 COMPARISON WITH OTHER THEORETICAL ALGORITHMS

While we did mention that there are other bidding strategies available, direct comparisons are difficult as universally every method cited either provides a theoretical algorithm and not an implementation (Zheng et al., 2015; Andrzejak et al., 2010) or an implementation with an algorithm that is closed source (i.e. AWS Batch with Spot Fleet Pricing). *Aether* already incorporates ideas proposed by existing algorithms that do not have implementations that we were able to empirically validate (i.e. incorporation of SLA constraints as in (Andrzejak et al., 2010)). In the case where there was a closed source implementation (AWS Batch), we were able to make both qualitative and empirical comparisons to *Aether*.

### 2.7.2 COMPARING *AETHER* WITH AWS BATCH

The problem of comparing two bidding algorithms on AWS cloud is difficult—one could easily sample outliers to paint a comparison picture that is not true (in either direction). Thus, as authors the burden of proof is on us to prove mathematically that *Aether*'s bidding algorithm yields better

prices than the closed source algorithm in AWS Batch. It is difficult to accurately compare two algorithms (and easy to fake—both algorithms running at the same time would affect the other; if they are running at separate times they are subject to different conditions. It is financially not feasible to run the same Batch process at realistic scale on both platforms with sufficient replicates to ensure statistical significance). Thus, we have attempted to logically showcase the features that make *Aether* a superior algorithm through examples comparing mechanisms of querying the Expectation of batch job runtime in both bidding approaches. The reasons that *Aether*'s bidding approach is superior is quite simple—AWS Batch never asks the user how long they expect each of their batch jobs to take. The goal of any arbitrary bidding algorithm looking at AWS spot instances (those that can terminate if you are outbid on price; *Aether*'s advantage comes from bidding on these types of instances efficiently) is to minimize the amount of “wasted” compute that occurs when you bid on an instance, run your job for an arbitrary amount of time, and have somebody outbid you before your job completes. Two important things to note here are 1) that AWS benefits if you bid incorrectly—they can very easily make more money having users bidding incorrectly and use more compute on spot instances than anticipated compared with initially utilizing on a fixed price instance and 2) unlike the stock market, past prices in a region do correlate well with future prices—it is quite common for the price of a region to fall into a “local minima” where it is underutilized for some reason. Besides parameterizing its Linear Programming (LP) bidding strategy with the characteristics of the compute that a user will require for a batch job, *Aether* also asks the user how long that they estimate each batch job will take. AWS Batch does not inquire as to an estimate of how long each batch job will take. The *Aether* bidding backend subsequently takes this estimate to parameterize a “lookback”



that assigns essentially serves as a regularization parameter penalizing regions that are not under-utilized with price in a stable local minima. Without this lookback, it is not feasible to have a spot instance bidding strategy that provides more utility per dollar than utilizing fixed price instances. Giving Amazon the benefit of the doubt, they could possibly be using extra compute to run machine learning algorithms that predict the Expectation of the amount of time a job will run for any AWS Batch user; we would argue non-objectively that the variability of runtime between bioinformatics tasks (i.e. de novo assembly versus alignment) would make such a hypothetical implication useless. Finally, it is worth noting that there is value in having a fully open source bidding algorithm as it holds AWS accountable in the context of providing the best service to their users by making the potential effects from the conflicts of financial interest in AWS running a tool to compete in a market that they fully control less opaque. Steps needed to run AWS batch are listed in Appendix A.

## 2.8 OPTIMIZATION PROBLEM SPECIFICATION

For bidding on the Amazon cloud, *Aether's* approach minimizes

$$\sum_{i=1}^n Q_i O_i + \sum_{j=1}^m W_j S_j$$

where there are  $n$  types of on demand instances ,  $m$  types of spot instances available (there is not a spot instance for every type of on demand instance),  $O = \{o_1, o_2, \dots, o_n\}$  is the set of prices for on demand instances,  $S = \{s_1, s_2, \dots, s_m\}$  is the set of maximum prices for spot instance types over a user specified “lookback” period,  $Q = \{q_1, q_2, \dots, q_n\}$  is the set of “bids” (number of instances requested)

for on demand instances, and  $S = \{s_1, s_2, \dots, s_m\}$  is the set of “bids” (number of instances requested)

for spot instances. This minimization is constrained by:

$$\begin{aligned} \sum_{i=1}^n Q_i O_i^{ram} + \sum_{j=1}^m W_j S_j^{ram} &\geq R \\ \sum_{i=1}^n Q_i O_i^{cpus} + \sum_{j=1}^m W_j S_j^{cpus} &\geq P \\ \sum_{i=1}^n Q_i O_i^{ephemeral-storage} + \sum_{j=1}^m W_j S_j^{ephemeral-storage} &\geq E \end{aligned}$$

where  $R$  is the minimum amount of total ram,  $P$  is the total number of processors, and  $E$  is the amount of scratch space available without need for purchasing EBS storage. To reach the 140 inequalities mentioned in the paper, an additional constraint is added for each instance type that bounds the number that can be bid upon (SLA Agreement constraints inspired by the work in (Andrzejak et al., 2010)) to the number of that type of instance the user currently has spun up subtracted for the users service limit for that type of instance. As mentioned previously in the supplemental methods section, the default *Aether* solve utilizes a recursive set reduction heuristic to reach an estimated ILP solution (code is available here: <https://github.com/kosticlab/aether/blob/master/lp/lp.py>). To allow for reproducibility, the ILP solution can also be generated with the CPLEX solver (code is available here: <https://github.com/kosticlab/aether/blob/master/lp/ilp.py>).

## 2.9 CONCLUSIONS

To our knowledge, this is the first implementation of a bidding algorithm for cloud compute resources that is tied both to an easy-to-use front-end as well as a distributed backend that allows for spinning up purchased compute nodes across multiple providers. Conceivably, this tool can be applied to any number of disciplines, bringing cost-effective cloud computing into the hands of scientists in fields beyond biology.

## 2.10 OTHER DIRECTIONS

After publication of *Aether* (Luber et al., 2017), I utilized the efficiency gains to probe the complexity of the athlete microbiome at scale (see Chapter 3), which Braden T. Tierney (*Aether* co-first author) also assisted on. While I was spearheading this project, Braden was utilizing *Aether* to 1) prove that the number of unique ORFs in the human associated microbiome is larger than previously estimated by a few million (Tierney et al., 2019) and 2) benchmark various machine learning models for metagenomic applications using data that is *de novo* assembled (Le Goallec et al., 2020). Like Braden assisted with my followup project, I also assisted with these projects. The interrogation and discovery of the larger than expected complexity of the genetic landscape of the human associated microbiome (Tierney et al., 2019) potentially explains why some of the failed projects included in Chapter 1 failed.

# 3

## A Performance Enhancing Microbe

THE HUMAN GUT MICROBIOME IS LINKED TO MANY STATES OF HUMAN HEALTH AND DISEASE (Gilbert et al., 2018).<sup>\*</sup> The metabolic repertoire of the gut microbiome is vast, but the health implications of these bacterial pathways are poorly understood. This chapter will cover a study

---

<sup>\*</sup>Portions of this chapter were previously published in *Nature Medicine* (Scheiman et al., 2019). I was co-first authors on this paper with Jonathan Scheiman and Ted Chavkin. Jonathan and Ted performed the experiments, and I did all computational work.

where we identify a link between members of the genus *Veillonella* and exercise performance. We observed an increase in *Veillonella* relative abundance in marathon runners postmarathon and isolated a strain of *Veillonella atypica* from stool samples. Inoculation of this strain into mice significantly increased exhaustive treadmill run time. *Veillonella* utilize lactate as their sole carbon source, which prompted us to perform a shotgun metagenomic analysis in a cohort of elite athletes, finding that every gene in a major pathway metabolizing lactate to propionate is at higher relative abundance postexercise. Using  $^{13}\text{C}_3$ -labeled lactate in mice, we demonstrate that serum lactate crosses the epithelial barrier into the lumen of the gut. We also show that intrarectal instillation of propionate is sufficient to reproduce the increased treadmill run time performance observed with *V. atypica* gavage. Taken together, these studies reveal that *V. atypica* improves run time via its metabolic conversion of exercise-induced lactate into propionate, thereby identifying a natural, microbiome-encoded enzymatic process that enhances athletic performance.

### 3.1 APPLYING *AETHER*

After completing the development of *Aether* (Luber et al., 2017), finding a biological application to prove the applicability of the method was the logical next step. The current state of the art in the microbiome has focused on combining the use of reference and *de novo* assembly based computational analysis of metagenomic samples (Pasolli et al., 2019). Furthermore, coupling understanding of computational analyses with that of bacterial metabolism is starting to yield higher resolution understanding towards how the microbiome interacts with the host (Shepherd et al., 2018; Mallick

et al., 2019). Many new microbiome analysis methods that go beyond using reference databases require lots of computing power; such methods were utilized in the work described later in this chapter. This is where *Aether* provided much utility for accelerating analyses.

### 3.2 BACKGROUND ON COMPUTATIONAL METHODS TO ANALYZE THE MICROBIOME

For much of the last decade, advances in the computational analysis of the microbiome have focused on improving reference based tools. Initial work utilizing 16S amplicon sequencing allowed for computational tools that could utilize raw sequencing reads and make predictions about taxonomic abundance and predictive functional profiling of a given sample (Douglas et al., 2018; Langille et al., 2013). From the groundwork that these 16S based methods provided, the idea of creating reference databases that raw reads could be aligned against cheaply and easily without complicated downstream analyses was expanded to utilize whole metagenome shotgun sequencing (McIver et al., 2018).

The first series of tools that implemented this idea were able to take raw NGS shotgun reads, align them to a database that associated genomic sequences with location on a bacterial taxonomic tree, and then infer the composition of a microbiome sample down to the species level (Segata et al., 2012). Further iterations of these methods improved their sensitivity and introduced predictive profiling at the strain level (Truong et al., 2015; Franzosa et al., 2018). These methods allowed for some of the first large scale consortium studies profiling the human associated microbiome (Turnbaugh et al., 2007; Lloyd-Price et al., 2017).

At the same time these reference based microbial methods were being developed, other consortiums were working on methods that relied on *de novo* assembly of reads from microbial samples as opposed to utilizing reference based approaches (Dusko Ehrlich & The MetaHIT Consortium, 2011). Initial work here focused on building larger and larger non-redundant gene catalogs from assemblies generated from metagenomic samples to better understand the diversity of the microbiome (Qin et al., 2010; Li et al., 2014). Commonly, the CD-HIT algorithm was utilized for construction of these non-redundant gene catalogs (Fu et al., 2012). These gene catalogs have been utilized to conduct metagenome wide association studies (Qin et al., 2012). Other recent work has shown that there are many more genes in the microbial universe than previously estimated by a few ten million (Tierney et al., 2019). Groups have also been able to reconstruct thousands of complete bacterial genomes from public databases of metagenomic studies that have mostly been used for past reference based analysis (Parks et al., 2017; Pasolli et al., 2019).

Studies have begun to combine computational analyses of the microbiome with synthetic biology based perturbation methods that depend on a deep understanding of microbial enzymatic chemistry and metabolism (Shepherd et al., 2018). A major criticism of *de novo* assembly based metagenomic analysis methods is that the vast majority of what they produce (i.e. non-redundant gene catalogs or metagenome assembled genomes) have no useful functional annotation and do not contribute to understanding of host-microbiome interaction. The study that I present in the rest of this chapter provides an example of how using massive computational resources coupled with a deep interrogation of microbial metabolism shines light on a previously functionally unannotated interaction between the microbiome and the host.

### 3.3 GUT *VEILLONELLA* ABUNDANCE IS SIGNIFICANTLY ASSOCIATED WITH MARATHON RUNNING

Human microbiome studies have generally examined individuals who are ‘healthy’ or diseased and identified features of the microbiome associated with these states (Lax et al., 2014; Rothschild et al., 2018; Dusko Ehrlich & The MetaHIT Consortium, 2011). Athlete microbiomes have been found to contain distinct microbial compositions defined by elevated abundances of *Veillonellaceae*, *Bacteroides*, *Prevotella*, *Methanobrevibacter* or *Akkermansia* (Petersen et al., 2017; Clarke et al., 2014). These studies show that exercise is associated with changes in microbiome composition, although the effects of these microbial genera on phenotype remain unknown.

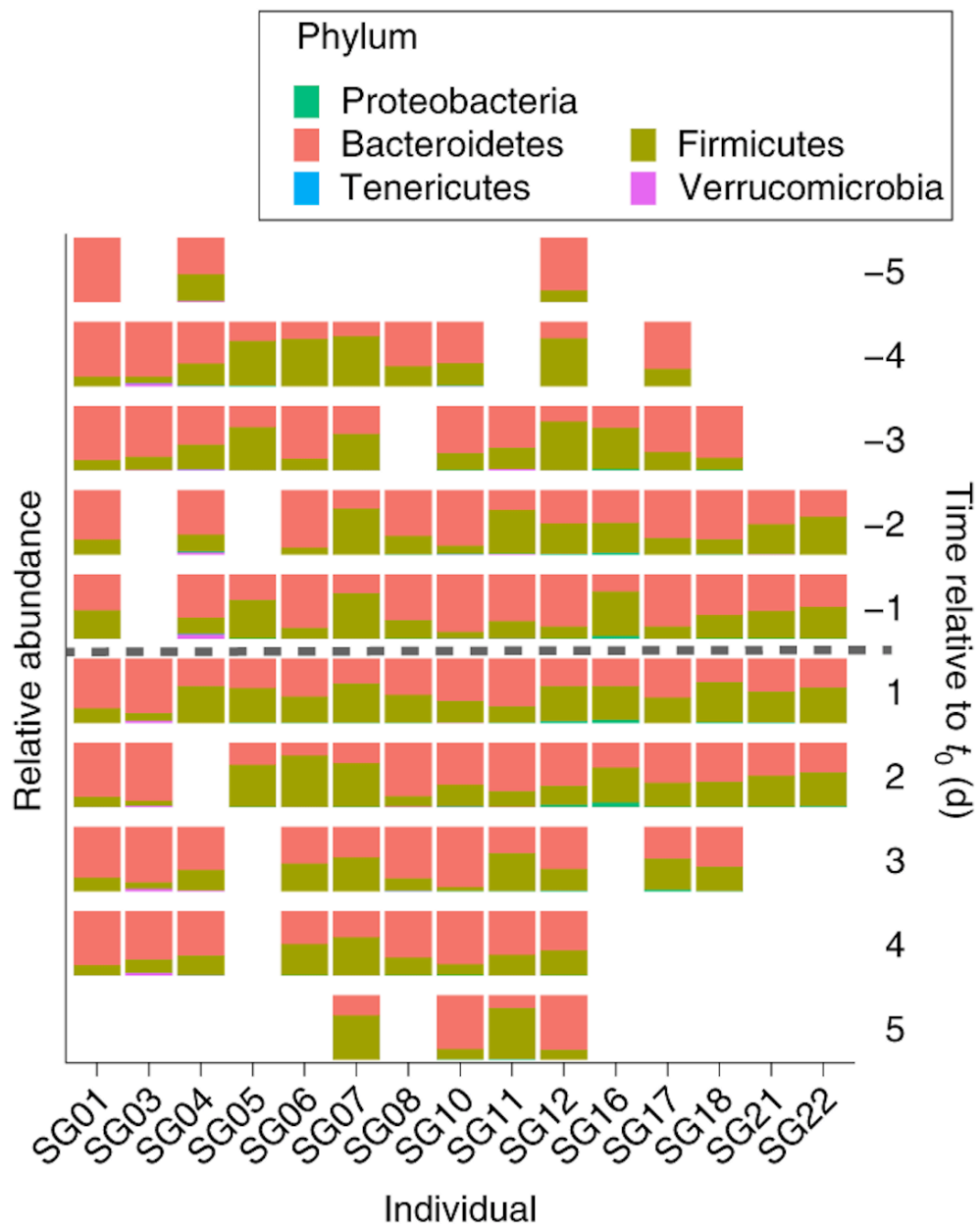
To identify gut bacteria associated with athletic performance and recovery states, we recruited athletes (n = 15) who ran in the 2015 Boston Marathon, along with a set of sedentary controls (n = 10), and conducted 16S ribosomal DNA (rDNA) sequencing on approximately daily samples collected up to one week before and one week after marathon day (n = 209 samples; access Supplementary Tables 1 and 2 in Appendix /refAppendixB). Phylum-level relative abundance partitioned by individual, time (-5 to +5 d in relation to running the marathon), and whether the participant was an athlete (Figure 3.1) showed that, at this high-level taxonomic view, any orthogonal differences were likely to be due to variation at the level of the individual. The bacterial genus *Veillonella* was the most differentially abundant microbiome feature between pre- and postexercise states (access Supplementary Table 2 in Appendix B). There was a significant difference in relative *Veillonella* abundance (P = 0.02, Wilcoxon rank-sum test with continuity correction) between samples collected be-



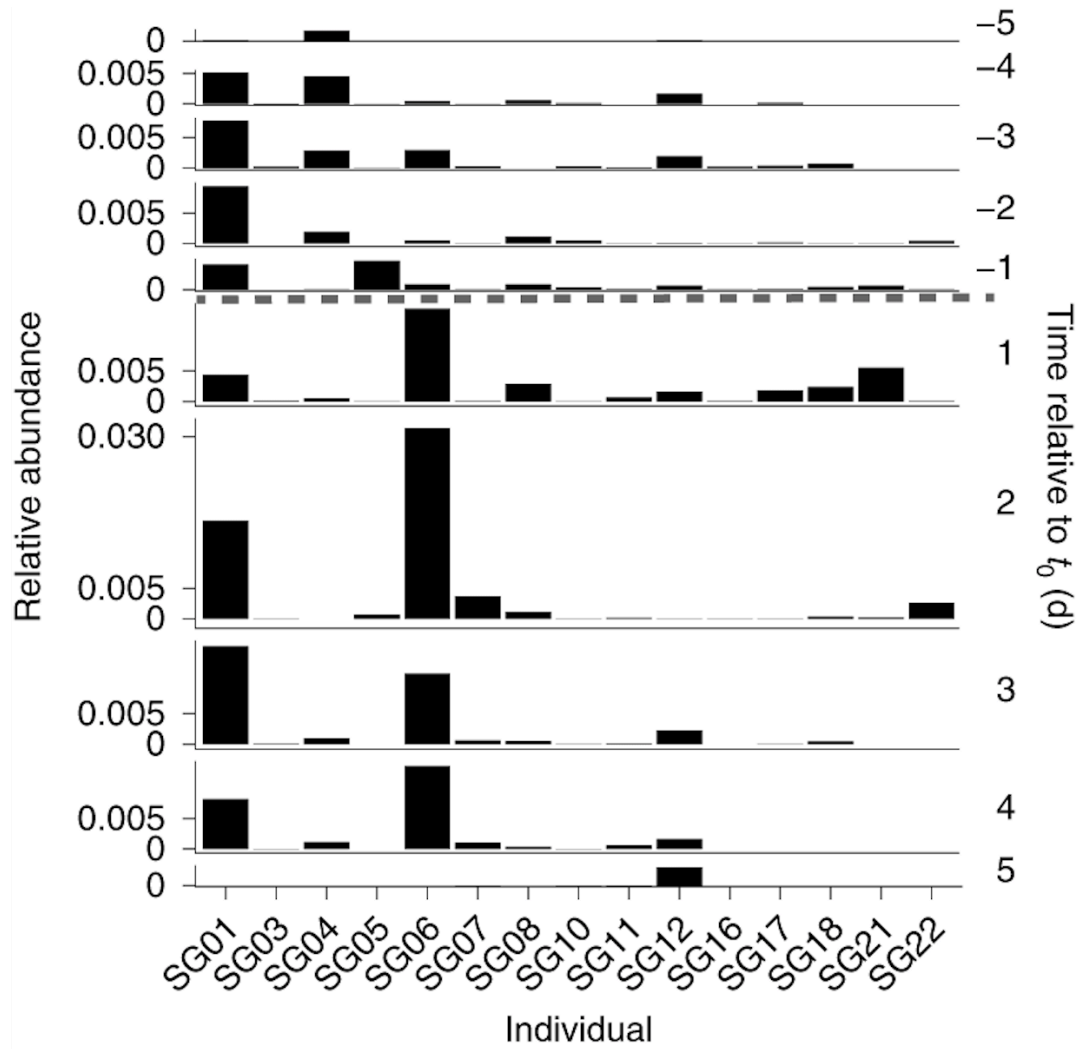
fore and after exercise (Figure 3.2). To validate the significance of the association between *Veillonella* and postmarathon state, we constructed a series of generalized linear mixed-effect models (GLMMs) to predict *Veillonella* relative abundance in the marathon participants (Figure 3.3). Subsequently, significance was calculated using Wald Z-tests for all of the coefficients included in the GLMM (Figure 3.4), revealing that no coefficients were significant except time in relation to (Figure 3.5). Additionally, it appears that *Veillonella* is more prevalent among runners than non-runners (Figure 3.6), although this was not statistically significant. These correlations raise the question of whether there is a causal link between *Veillonella* and marathon runners' performance, but no conclusions can be made without proper validation.

#### 3.4 *V. ATYPICA* GAVAGE IMPROVES TREADMILL RUN TIME IN MICE

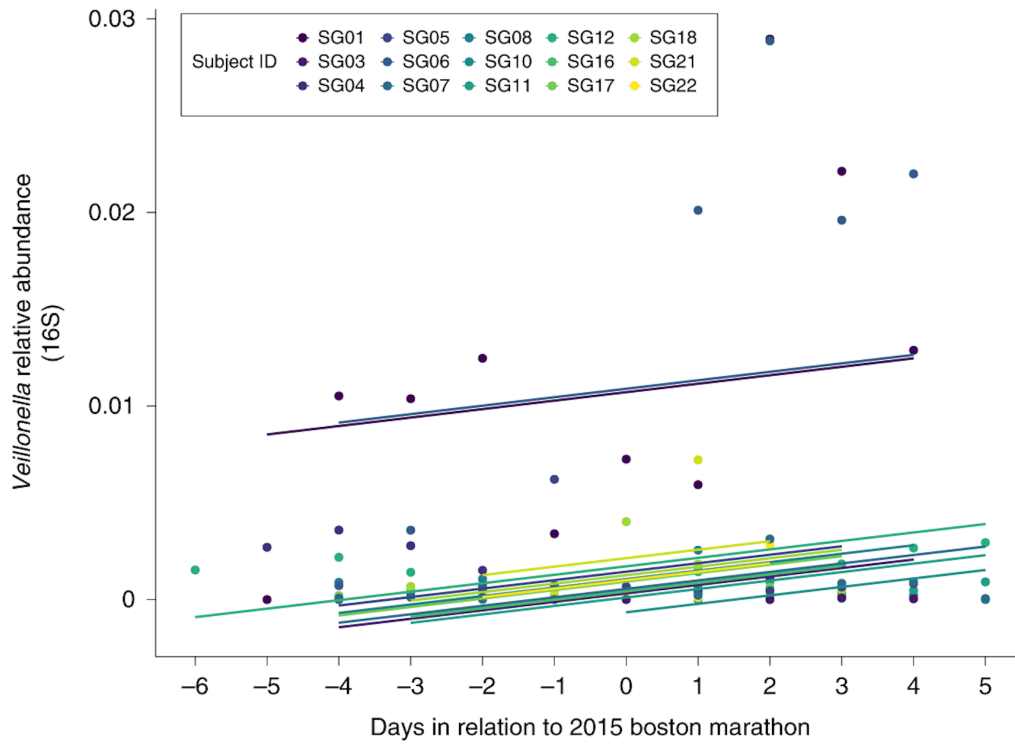
To assess whether there are any potential benefits of *Veillonella* on performance in an animal exercise model, we designed an AB/BA crossover mouse experiment spanning 2 weeks, consisting of a control group (*Lactobacillus bulgaricus* gavage; n = 16) and a treatment group (*Veillonella atypica* gavage; n = 16), with a treatment/control crossover happening between weeks (n = 32 mice in total). *L. bulgaricus* was chosen as a control due its inability to catabolize lactate, thus mimicking the bacterial load but without impacting lactate metabolism (Garvie, 1980). The *Veillonella* strain used, *Veillonella atypica*, was directly isolated from one of the marathon runners. Mice were administered either *V. atypica* or *L. bulgaricus* and run to exhaustion 5 h later. In aggregate, on both sides of the crossover, mice gavaged with *V. atypica* had statistically significantly longer maximum run times



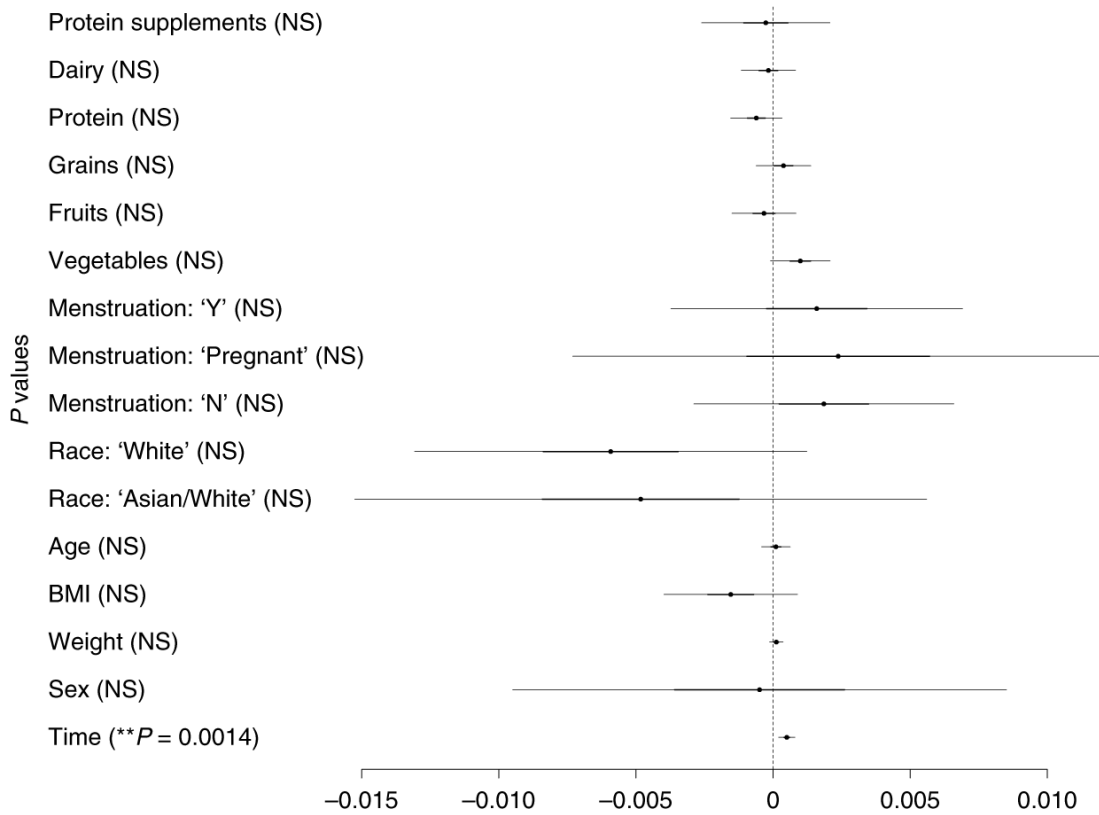
**Figure 3.1:** Phylum-level relative abundance in marathon runners, partitioned by individual and time (-5 to +5 d in relation to running the marathon ( $t_0$ ), where negative values are premarathon and positive values are postmarathon), showing few global differences in composition.



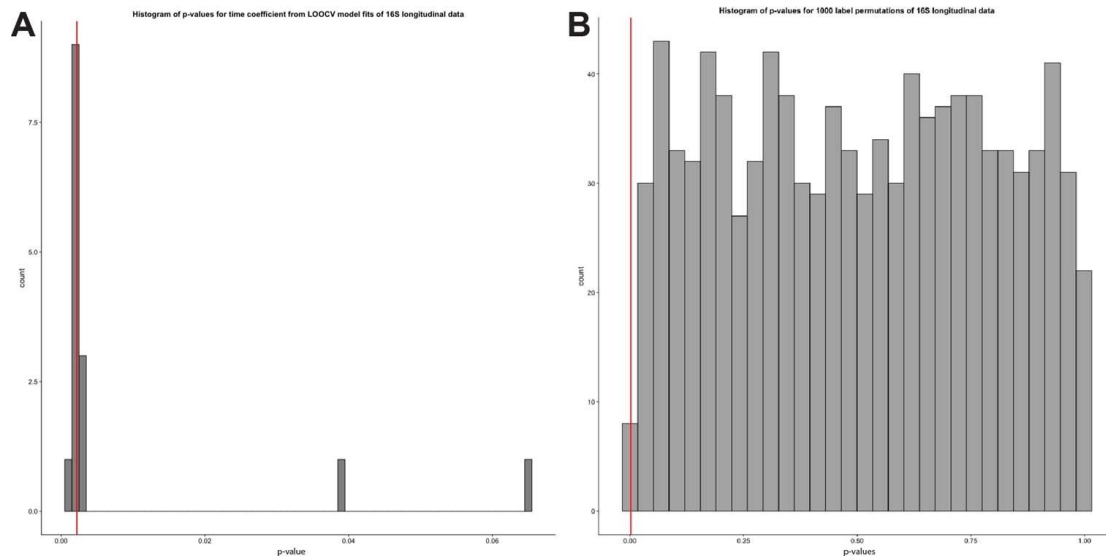
**Figure 3.2:** *Veillonella* relative abundance at the genus level, partitioned by individual and time (-5 to +5 d in relation to running the marathon), showing that there is a significant difference in *Veillonella* relative abundance ( $P = 0.02$ , two-sided Wilcoxon rank-sum test with continuity correction;  $n = 15$  individuals) between samples collected before and after exercise.



**Figure 3.3:** GLMMs predicting longitudinal *Veillonella* relative abundance in the marathon participants. Differences in intercepts between fits for different marathon runners represent random effects.



**Figure 3.4:** 95% confidence intervals for all of the fixed effects (coefficients) included in the GLMMs. All coefficients except time ( $P = 0.0014$ , Wald Z-test; time postmarathon corresponds to increased *Veillonella* relative abundance) were not significant (NS), suggesting that *Veillonella* blooms in runners correspond to exercise state and not other fixed effects ( $n = 15$  individuals).



**Figure 3.5:** **a** Histogram of P values (Wald Z-tests) for time coefficient from LOOCV models predicting 16S *Veillonella* abundance. The red line represents the P value for the model trained without any hold outs. **b** Histogram of P values for time coefficients from 1,000 label permutations in GLMM models predicting *Veillonella* relative abundance. The red line represents the P value for the model trained without any label permutation.

than mice gavaged with *L. bulgaricus* ( $P=0.02$ , paired t-test; Figure 3.7, Figure 3.8, access Supplementary Table 3 from Appendix B). Both LOOCV and iterative permutation of labels were conducted as part of the GLMM analysis (Figure 3.8). Per-mouse run times overlaid on the GLMM fits (Figure 3.9), as well as the difference between the maximum run times in *L. bulgaricus* versus *V. atypica* gavage, showed a distinction between ‘responders’ and ‘non-responders’ to *V. atypica* gavage (Figure 3.10). Mice treated with *V. atypica* ran, on average, 13% longer than the control group (Figure 3.7). Testing the significance of coefficients in the GLMM for their contribution to treadmill run time (Wald Z-test) showed that the sequence effect was not significant ( $P=0.758$ ), while treatment day ( $P=0.031$ ; negative effect on run time) and *Veillonella* treatment ( $P=0.016$ ; positive effect on run time) were significant (Figure 3.11 and Figure 3.8). In a separate experiment, levels of inflamma-

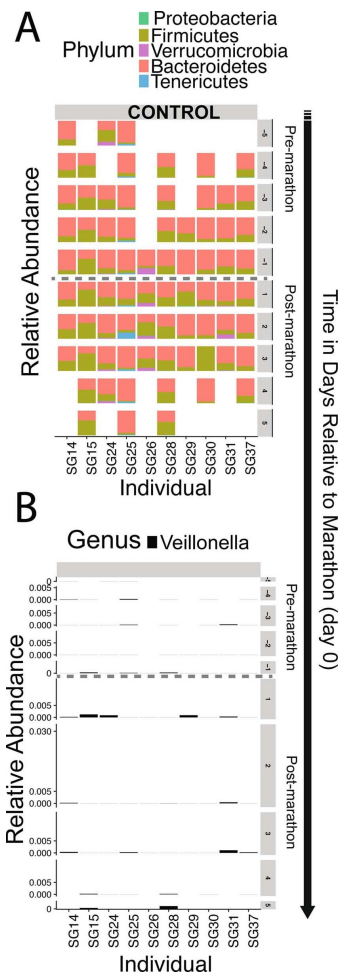


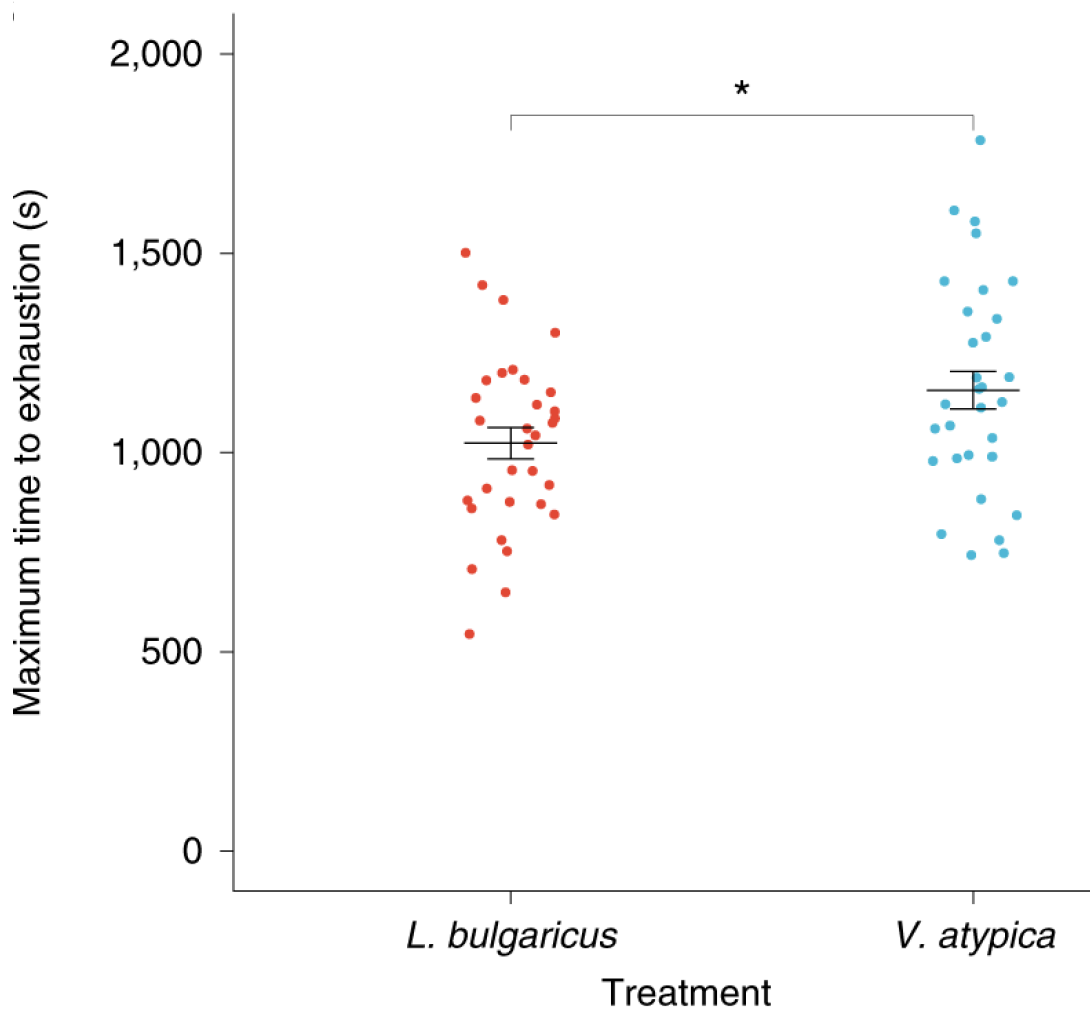
Figure 3.6: a 16S composition in control subjects. b *Veillonella* relative abundance in control subjects.

tory cytokines were quantified postexercise, and were significantly reduced in *Veillonella*-treated animals compared with *L. bulgaricus* or phosphate buffered saline (PBS) (Figure 3.12; access Supplementary Table 4 from Appendix B). To assess changes in muscle physiology, the glucose transporter GLUT<sub>4</sub> was quantified via western blot, but we observed no changes regardless of treatment (Figure 3.13).

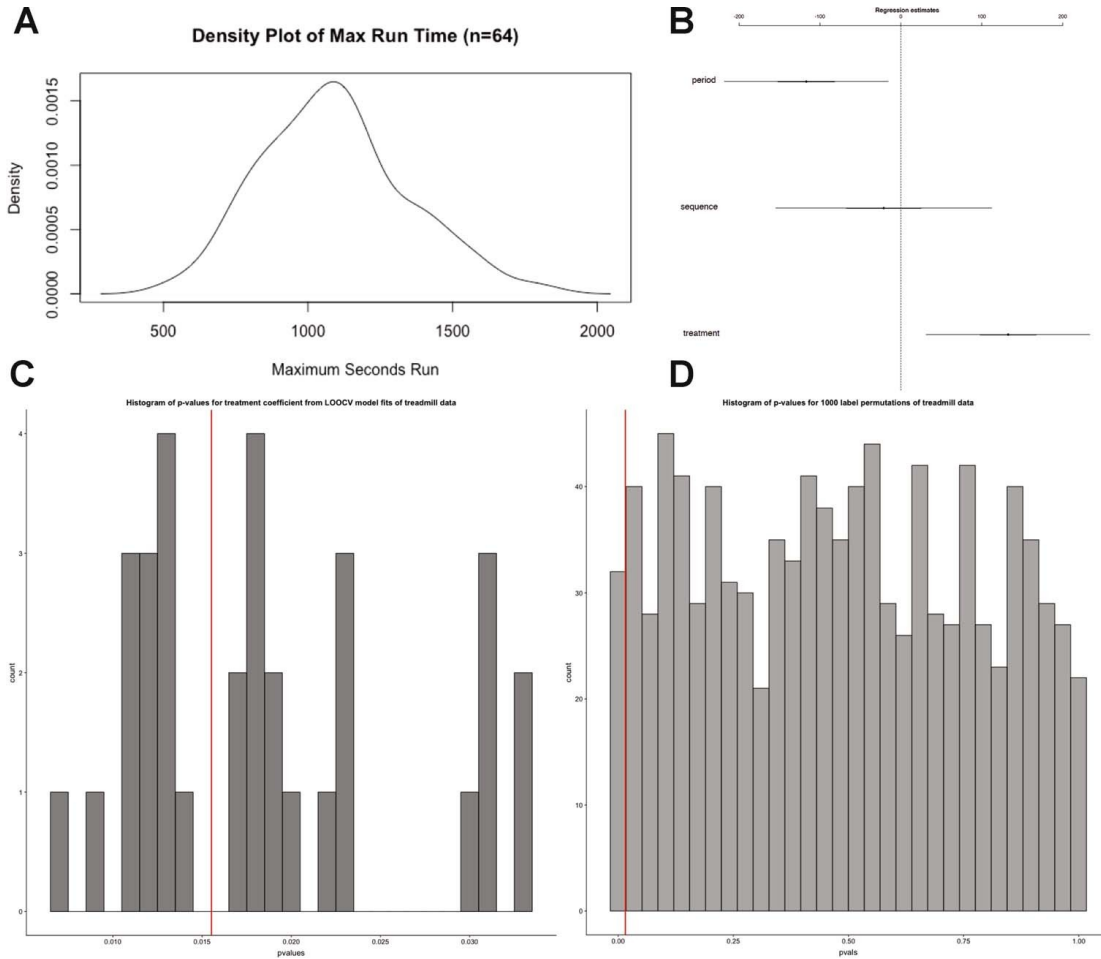
### 3.5 THE ATHLETE GUT MICROBIOME IS FUNCTIONALLY ENRICHED FOR THE METABOLISM OF LACTATE TO PROPIONATE POSTEXERCISE

To test whether our results would be replicated in an independent cohort of human athletes, we performed shotgun metagenomic sequencing of stool samples (n = 87) from ultramarathoners and Olympic trial rowers both before and after exercise (access Supplementary Table 5 from Appendix B). Putative taxonomic abundances reproduced the previous 16S sequencing-based association with *Veillonella* (Figure 3.14) (Truong et al., 2015). By utilizing novel algorithms that allow for cheap construction of metagenomic gene catalogs at a massive scale through the efficient use of cloud computing, we investigated phenotypic modulating effects of millions of microbial genes on athletes by building a sample (n = 87) by gene (n = 2,288,155) relative abundance matrix (Figure 3.14) (Luber et al., 2017; Li et al., 2015; Seemann, 2014; Fu et al., 2012; Qin et al., 2012). The inability of *Veillonella* to ferment carbohydrates, coupled with the high observed abundance of the lactate import permease in previously sequenced isolates, suggests that metabolic enzymes facilitating lactate breakdown are likely conserved (van den Bogert et al., 2013). Across the entire ultramarathon and rower cohorts,

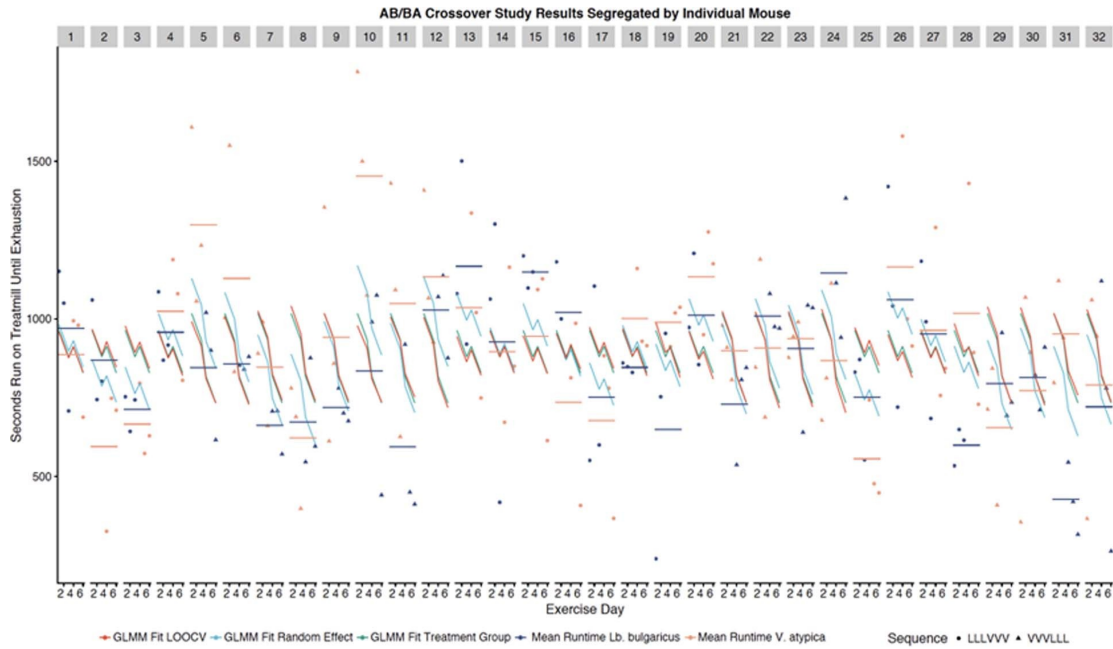




**Figure 3.7:** Mice gavaged with *V. atypica* had greater maximum run times per week than mice gavaged with *L. bulgaricus* in an AB/BA crossover trial. The graph shows the maximum run times out of 3 d of consecutive treadmill running for a given treatment (all mice switched treatments for the second week). The jitter plot shows each mouse as an individual point, with the central bar representing the mean and error bars representing s.e.m. (n = 32). \*P = 0.02, as determined by two-sided paired t-test.



**Figure 3.8:** a Density plot of maximum run times in the AB/BA crossover study. A two-sided Shapiro–Wilk normality test on the maximum run times for each mouse in each treatment group resulted in  $P = 0.67$ , with the null hypothesis that the distribution of data is normal ( $n = 64$ ). b 95% confidence intervals for the coefficient effect on treadmill run time in AB/BA crossover (Wald Z-tests,  $n = 64$ ). Center values are the regression estimate for each coefficient. Error bars represent the 95% confidence interval. c Histogram of P values for the treatment coefficient from LOOCV models predicting treadmill run time. The red line represents the P value for the model trained without any hold outs (Wald Z-tests,  $n = 64$ ). d Histogram of P values for the treatment coefficient from 1,000 label permutations in GLMM models predicting treadmill run time. The red line represents the P value for the model trained without any label permutation (Wald Z-tests,  $n = 64$  per permutation).



**Figure 3.9:** Each of the 32 facets (each representing an individual mouse) has six longitudinal treadmill run times plotted (three pre- and three post-treatment crossover). The shapes of the points represent the treatment sequence. Each mouse facet has two horizontal lines showing the mean run time when dosed with *L. bulgaricus* (light blue) or *V. atypica* (light red). Each facet has a GLMM fit to all data in a treatment sequence (green), a LOOCV GLMM fit trained on all mice except for the mouse the facet represents (red), and a GLMM fit showing the change in intercept related to random effect for each mouse (blue).

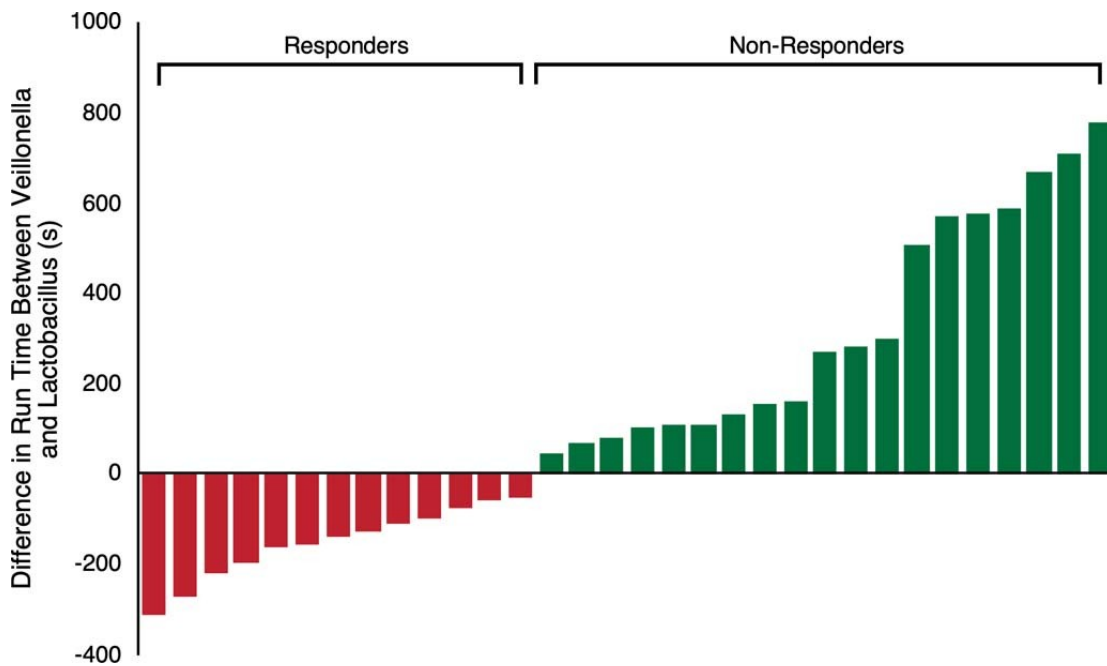
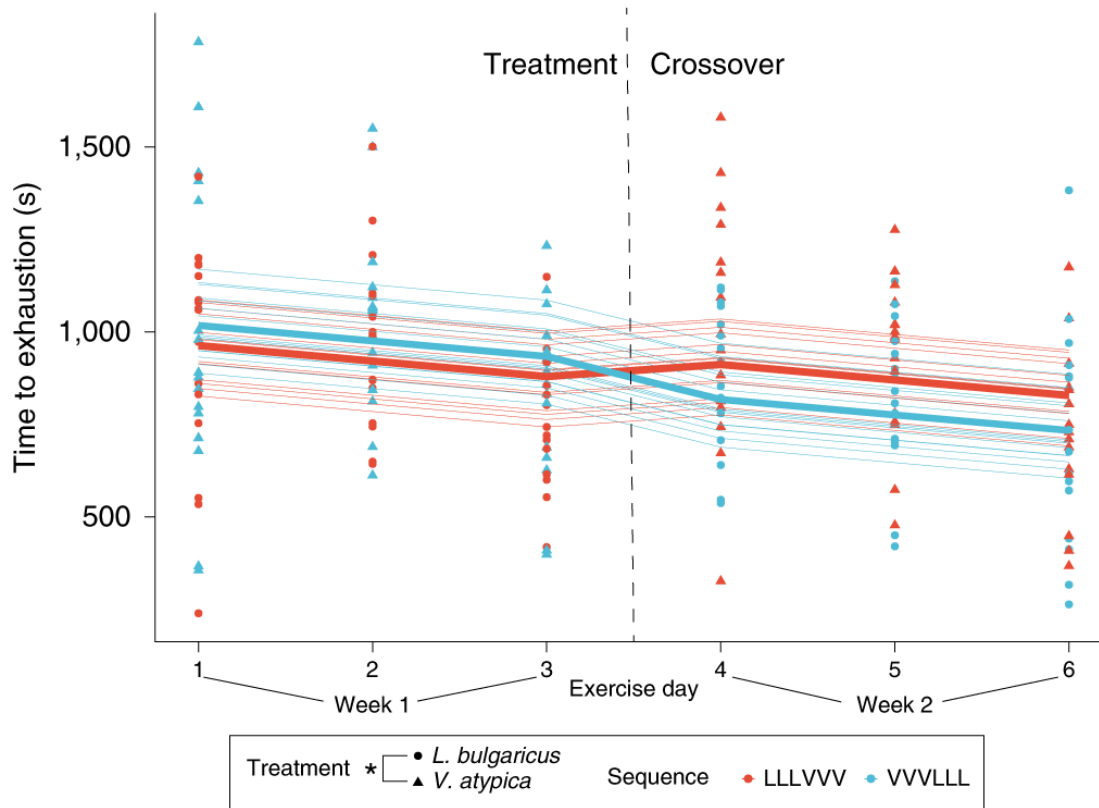
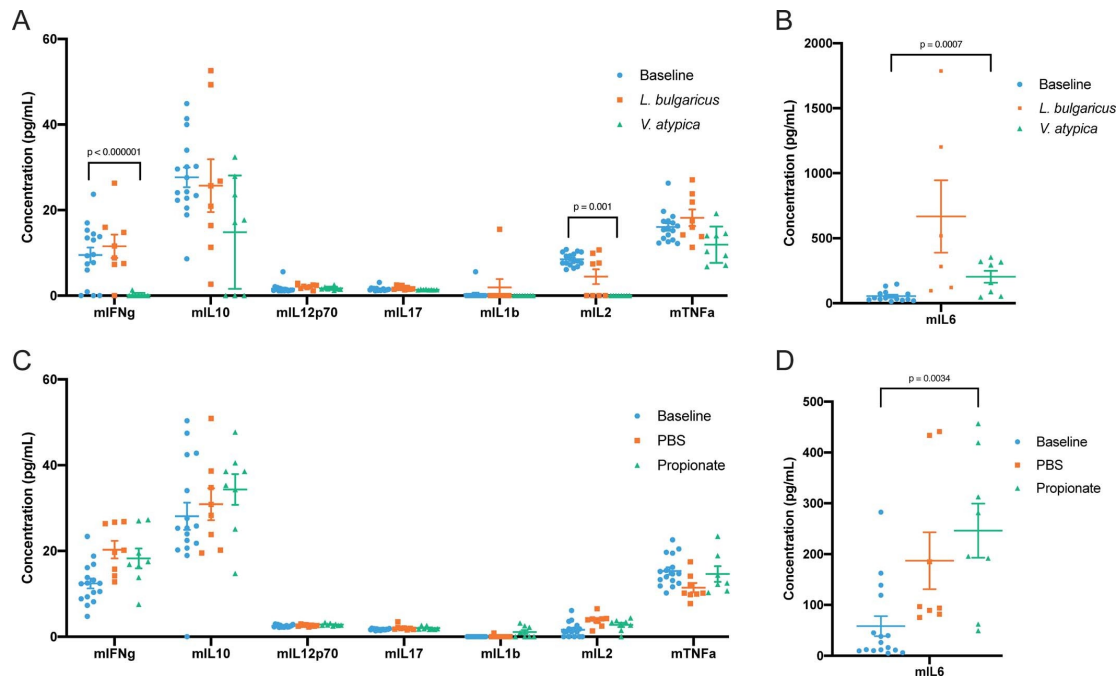


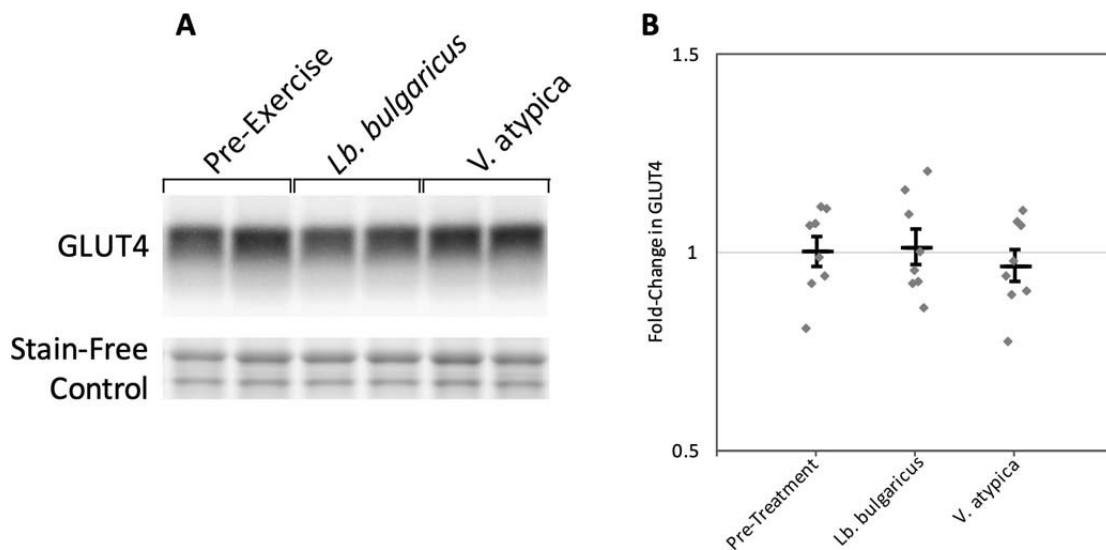
Figure 3.10: Difference in maximum run time between *V. atypica* and *L. Bulgaricus* gavage treatment periods, segregated into 'responders' and 'non-responders' to *V. atypica* treatment (n = 32).



**Figure 3.11:** GLMMs predicting run time in the 2-week AB/BA crossover trial. The colors of the lines (GLMM fits) and points (runs by an arbitrary mouse) represent the treatment sequence (in the legend, L represents *L. bulgaricus* and V represents *Veillonella atypica*). The shapes of the points represent the treatment at a given time point. These models incorporate both random effects (individual variation per mouse that manifests longitudinally) and fixed effects (treatment day, treatment sequence and treatment given). Visualization of all of the longitudinal data points with the GLMM predictions overlaid shows the effect of *V. atypica* increasing performance on both sides of the crossover when aggregated by treatment group (thick lines), as well as the trends for each of the 32 individual mice (thin lines). \*P = 0.016, as determined by Wald Z-test on model coefficients.

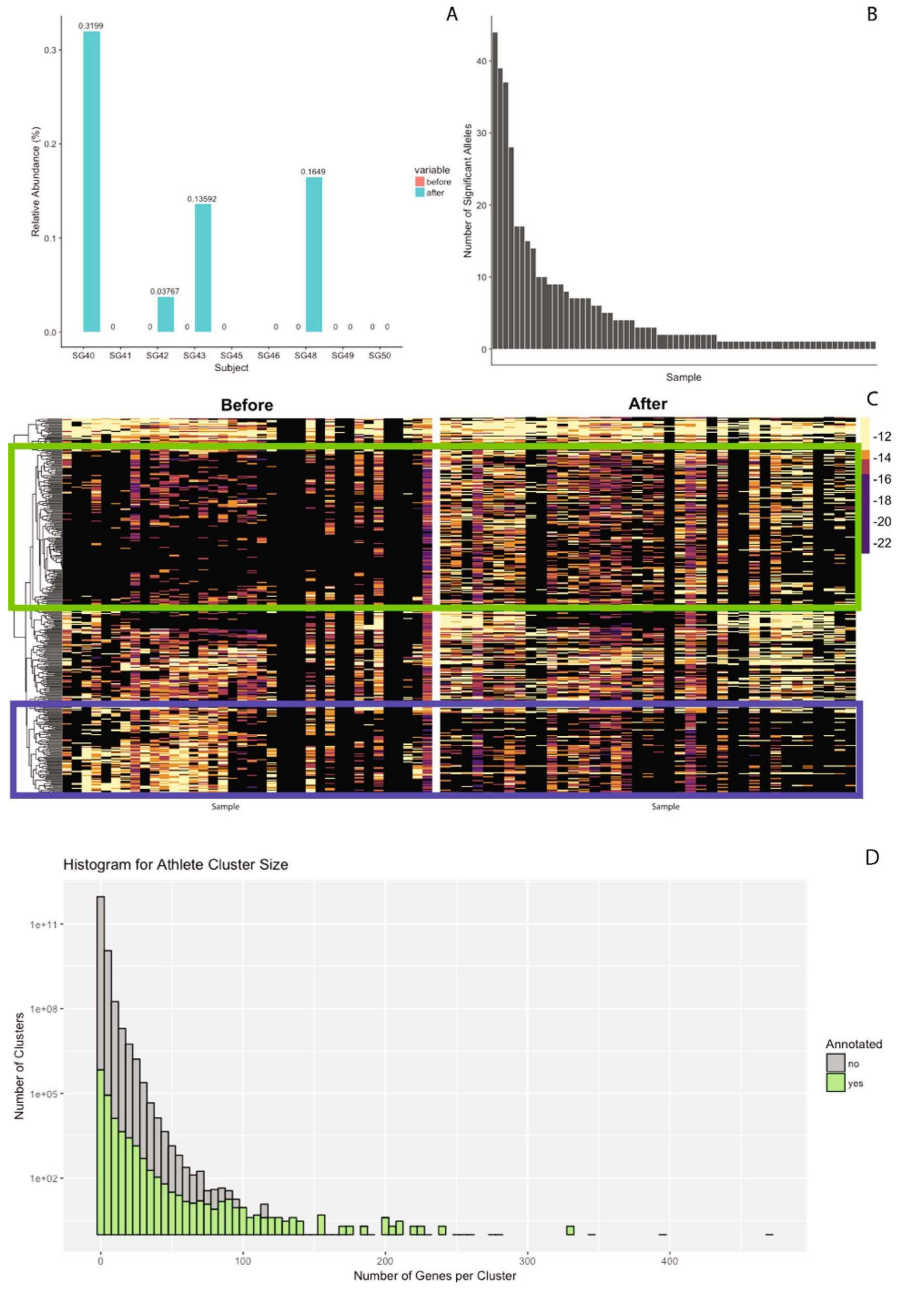


**Figure 3.12:** a,b Cytokines after *V. atypica* and *L. bulgaricus* gavage. Each mouse sample is represented as an individual point, with the central bar representing the mean and error bars representing s.e.m. ( $n = 64, 32$  and  $32$  for baseline, *L. bulgaricus* and *V. atypica*, respectively). c,d Cytokines after intrarectal propionate instillation. Each mouse sample is represented as an individual point, with the central bar representing the mean and error bars representing s.e.m. ( $n = 32, 16$  and  $16$  for baseline, *L. bulgaricus* and *V. atypica*, respectively). P values were determined by one-way ANOVA followed by Tukey's posthoc test.



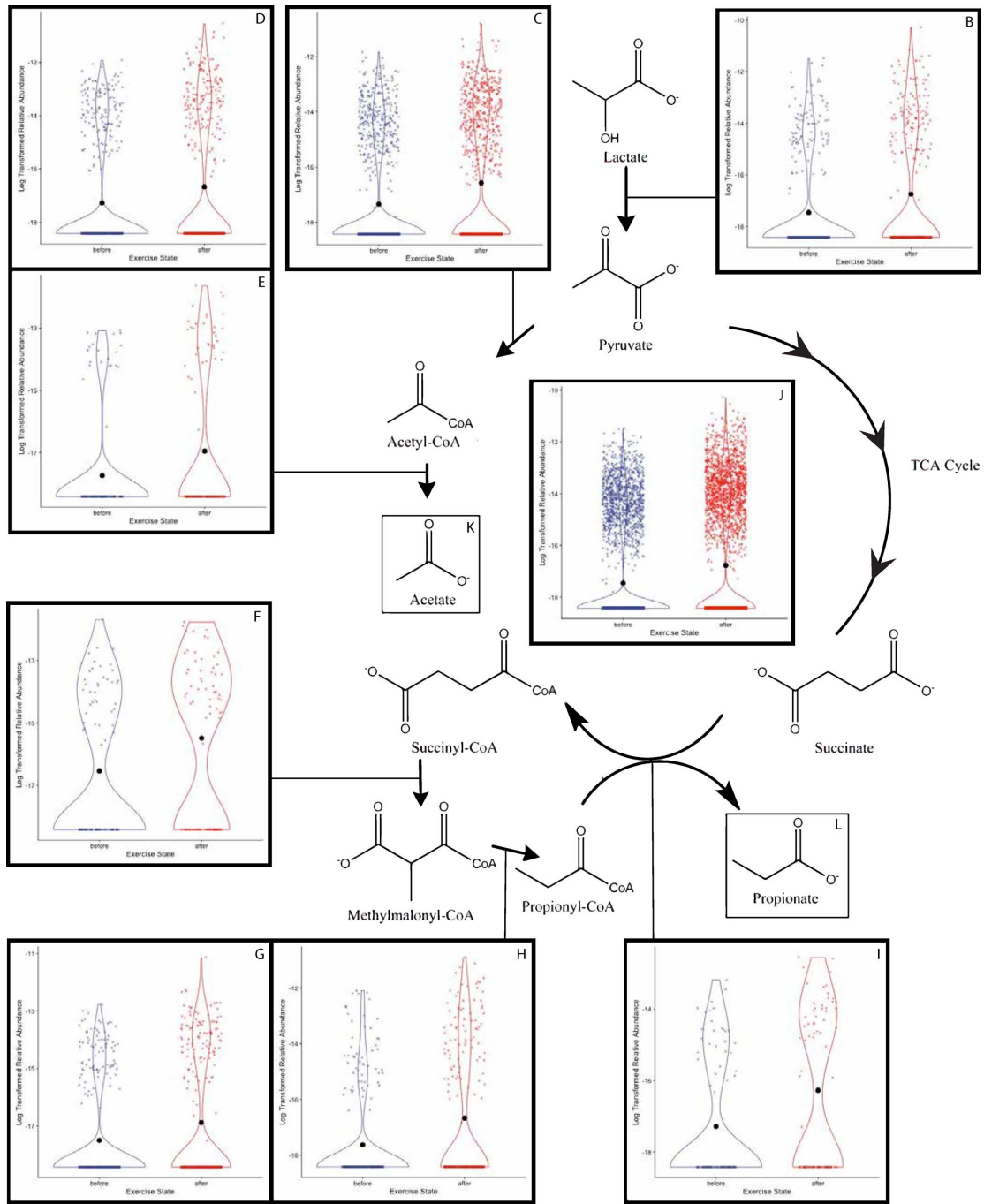
**Figure 3.13:** **a** Representative section of western blot showing GLUT4 abundance in pre-exercise states, as well as following *L. bulgaricus* and *V. atypica* gavage. A stain-free control was used to normalize the densitometry analysis shown. The experiment was performed once (n = 8). **b** Fold-change in GLUT4 abundance. Each point represents an individual mouse sample, the centre bar represents the mean and error bars represent s.e.m. (n = 8).

there exists a group of gene families with differential relative abundance pre- and postexercise (Figure 3.15), representing every step of the enriched methylmalonyl-CoA pathway ( $P = 0.00147$ ), degrading lactate into propionate, as assigned by Enzyme Commission (EC) ID numbers (Figure 3.16). Given the limited prevalence of the methylmalonyl-CoA pathway across lactate-utilizing microbes (Figure 3.17 and Figure 3.18, this enrichment postexercise may implicate Veillonella in causing functional changes in the metabolic repertoire of the gut microbiome. We verified strong production of acetate and propionate by performing mass spectrometry on spent media collected after growing three Veillonella strains isolated from the human athletes (*V. parvula*, *V. dispar* and *V. atypica*) in lactate-supplemented brain–heart infusion media (BHIL) and semi-synthetic lactate media (Figure 3.19; access Supplementary Table 6 from Appendix B).

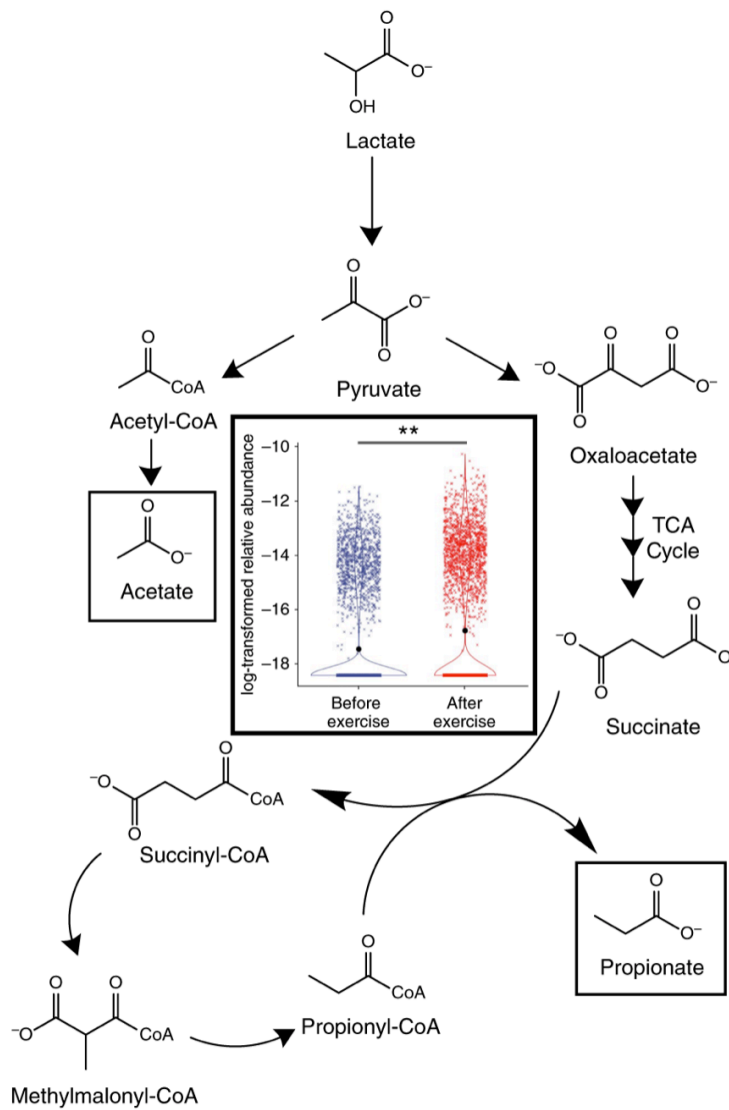


**Figure 3.14:** a Fraction of putative *Veillonella* relative abundance from metagenomics (calculated utilizing MetaPhlan2(Truong et al., 2015)) before and after exercise in rowers and runners. b Significant alleles (calculated from pairwise ANOVA) that are present in each of the 87 samples. c The aforementioned 396 significant alleles segregated by exercise state and sample. d Histogram comparing non-redundant gene family size and annotation fraction.

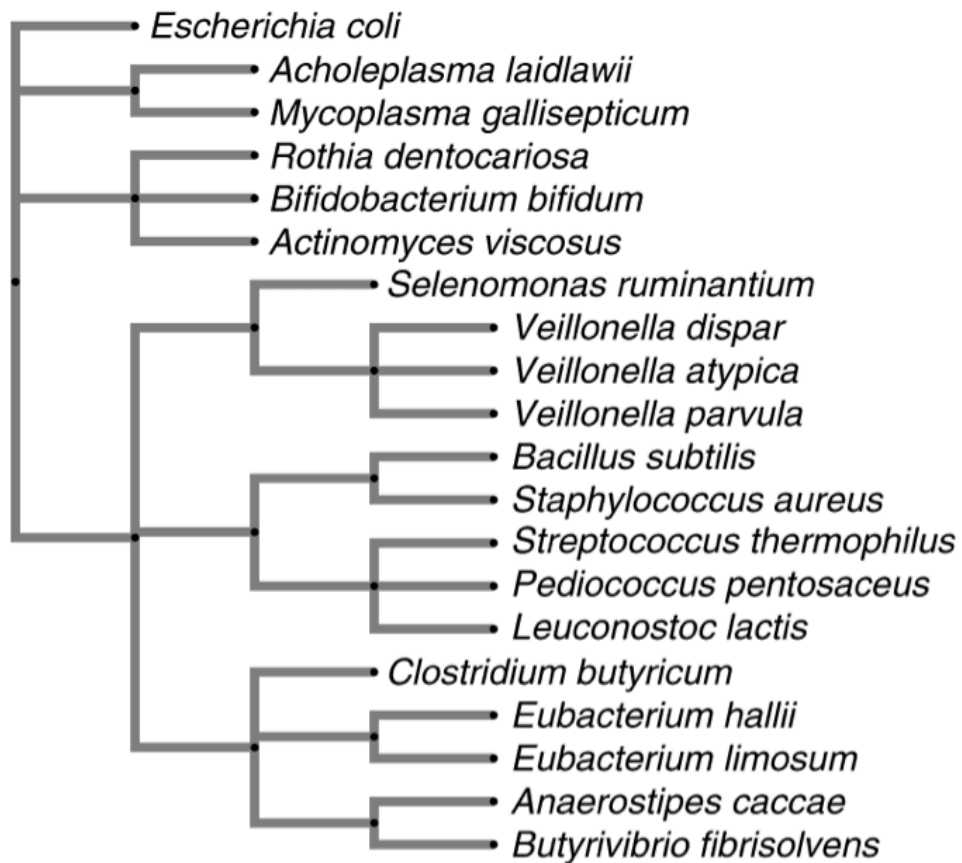




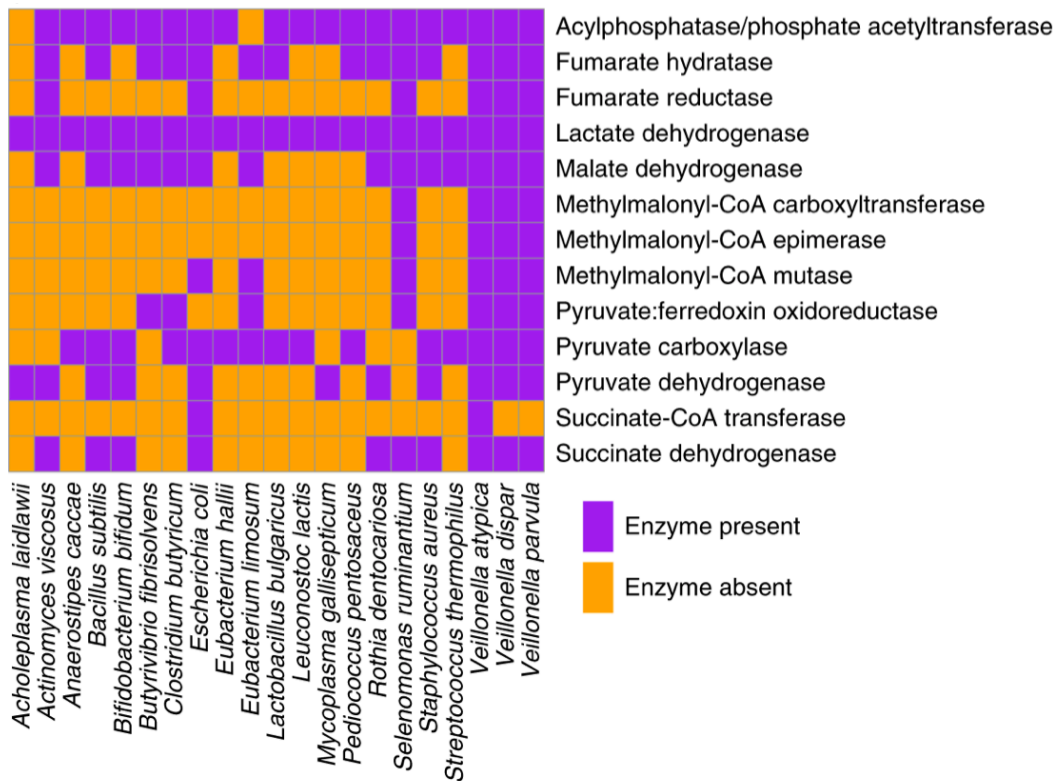
**Figure 3.15:** Enzyme-resolution, log-transformed relative abundances of differentially abundant non-redundant gene families mapped by EC ID to methylmalonyl-CoA pathway components. a Pathway in aggregate. b-i Individual reactions in the pathway (n = 8). Data are represented as violin plots, which display the distribution of data as a rotated kernel density distribution.



**Figure 3.16:** The methylmalonyl-CoA pathway, with conserved steps of the tricarboxylic acid (TCA) cycle abbreviated. Inset, significantly differentially expressed gene families within the methylmalonyl-CoA pathway in a pair of non-redundant gene catalogs created from metagenomic sequencing of athlete stool samples. The log-transformed relative abundance increases after exercise for every enzyme in the methylmalonyl-CoA pathway.  $**P = 0.00147$ , as determined by two-sided Fisher's exact test ( $n = 8$  enzymes in the pathway). Data are represented as a violin plot, which displays the distribution of data as a rotated kernel density distribution



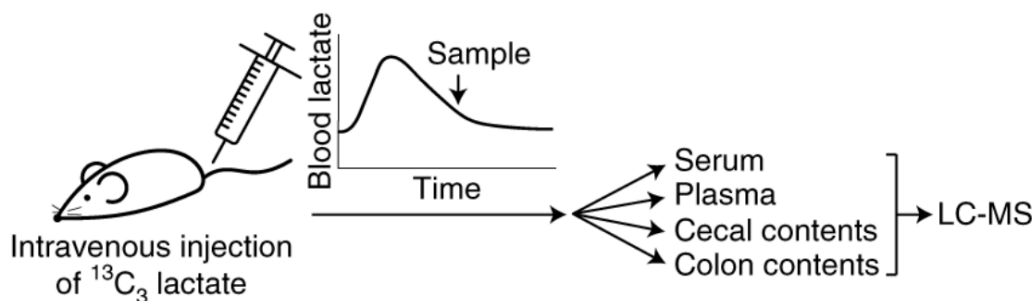
**Figure 3.17:** Bacterial phylogenetic tree showing the diversity of microbes that have the ability to utilize lactate as a carbon source.



**Figure 3.18:** Prevalence of enzymes in the methylmalonyl-CoA pathway that break down lactate into acetate and propionate in reference genomes from the representative subset of lactate-processing microbes in Figure 3.17.

	Butyrate ( $\mu\text{M}$ )	Propionate ( $\mu\text{M}$ )	Pyruvate ( $\mu\text{M}$ )	Lactate ( $\mu\text{M}$ )	Acetate ( $\mu\text{M}$ )
<i>V. atypica</i> LM	19 $\pm$ 0.3***	10,611 $\pm$ 584**	4 $\pm$ 0.4***	431 $\pm$ 5.6***	92,432 $\pm$ 3,129**
<i>L. bulgaricus</i> LM	9 $\pm$ 0.5*	3 $\pm$ 1.5	32 $\pm$ 2.7	737 $\pm$ 45	4,985 $\pm$ 247**
LM alone	11 $\pm$ 0.1	4 $\pm$ 0.5	41 $\pm$ 0.4	851 $\pm$ 6.6	1,741 $\pm$ 12.2
<i>V. atypica</i> BHIL	69 $\pm$ 0.5*	2,286 $\pm$ 68*	NA	NA	4,149 $\pm$ 118*
BHIL alone	147 $\pm$ 4.1	160 $\pm$ 1.4	NA	NA	7,557 $\pm$ 30

**Figure 3.19:** SCFAs detected in spent media after 48 h of growth with the indicated strain. LM, semi-synthetic lactate media; NA, not quantified. Each table entry shows the mean  $\pm$  s.e.m. (BHIL, n = 2; LM, n = 3). P values from left to right, row by row were: \*\*\*P = 0.0008, \*\*P = 0.003, \*\*\*P =  $4.4 \times 10^{-7}$ ; \*\*\*P =  $1.4 \times 10^{-6}$ ; \*\*P = 0.001; \*P = 0.023; \*\*P = 0.006; \*P = 0.03; \*P = 0.02; and \*P = 0.015, compared with the media control, as determined by two-sided Welch's t-test.



**Figure 3.20:** Schematic of the  $^{13}\text{C}_3$  flux-tracing experimental design. Mice were injected with  $^{13}\text{C}_3$  sodium lactate, then sacrificed after 12 minutes. Serum and plasma were collected via cardiac puncture. Cecum and colon contents were collected by dissection.

*Veillonella* species metabolize lactate into the short-chain fatty acids (SCFAs) acetate and propionate via the methylmalonyl-CoA pathway (Ng & Hamilton, 1973). Lactate dehydrogenase—the enzyme responsible for the first step of lactate metabolism—is present in a phylogenetically diverse group of bacteria (Figure 3.17). Querying microbial isolate strain genome annotations from National Center for Biotechnology Information (NCBI) shows that, unlike *V. atypica*, many other microbes are theoretically capable of utilizing lactate through lactate dehydrogenase, but do not possess the full pathway to convert lactate into propionate (Figure 3.18). Other obligate anaerobes, such as *Anaerostipes caccae* and *Eubacterium hallii* commonly ferment lactate into butyrate via different pathways (Figure 3.18). *E. hallii* can also produce propionate; however, this has been demonstrated as a biotransformation of 1,2-propanediol, rather than a complete pathway from lactate to propionate. Of note, the reference genomes on NCBI for both *Veillonella dispar* and *Veillonella parvula* are not annotated to have the succinate-CoA transferase needed for propionate production to occur; this is likely to be due to an annotation error, as we validated the production of propionate via mass

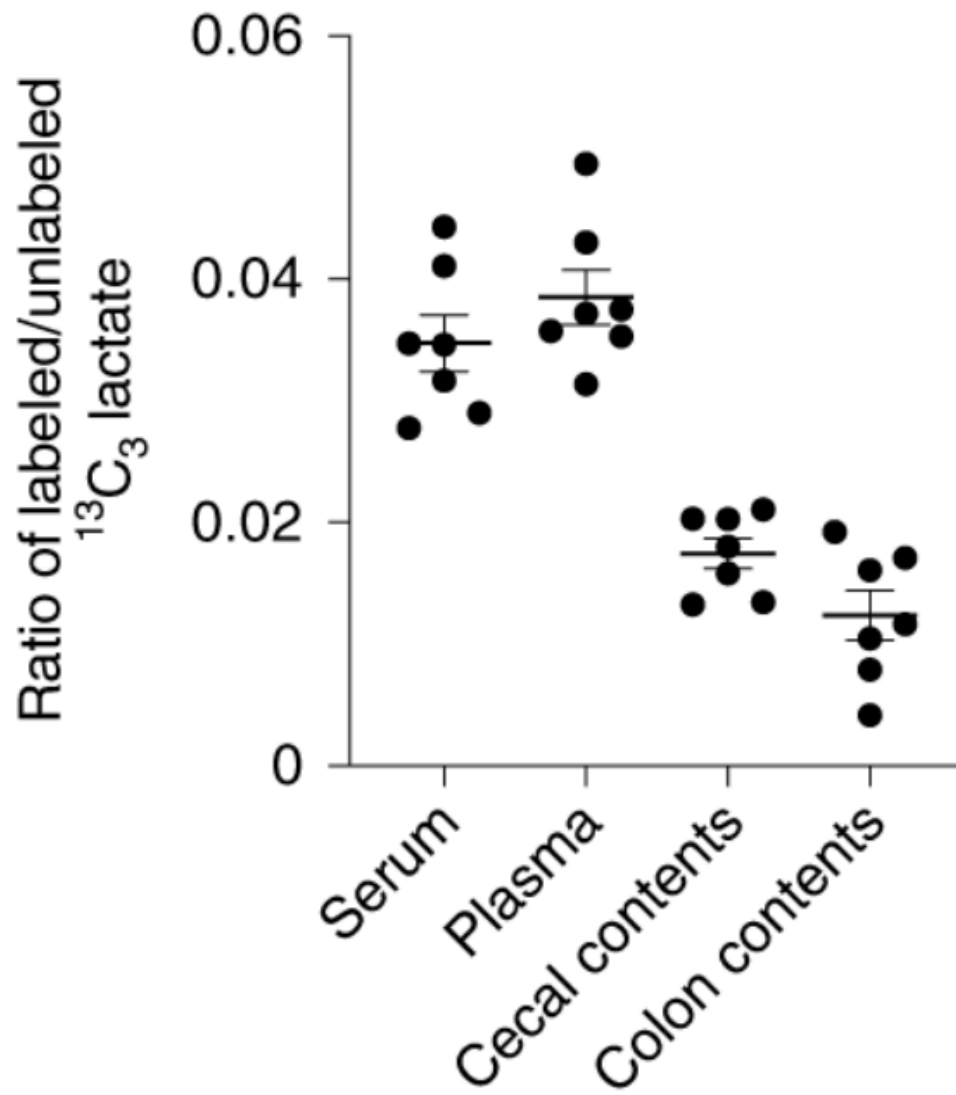
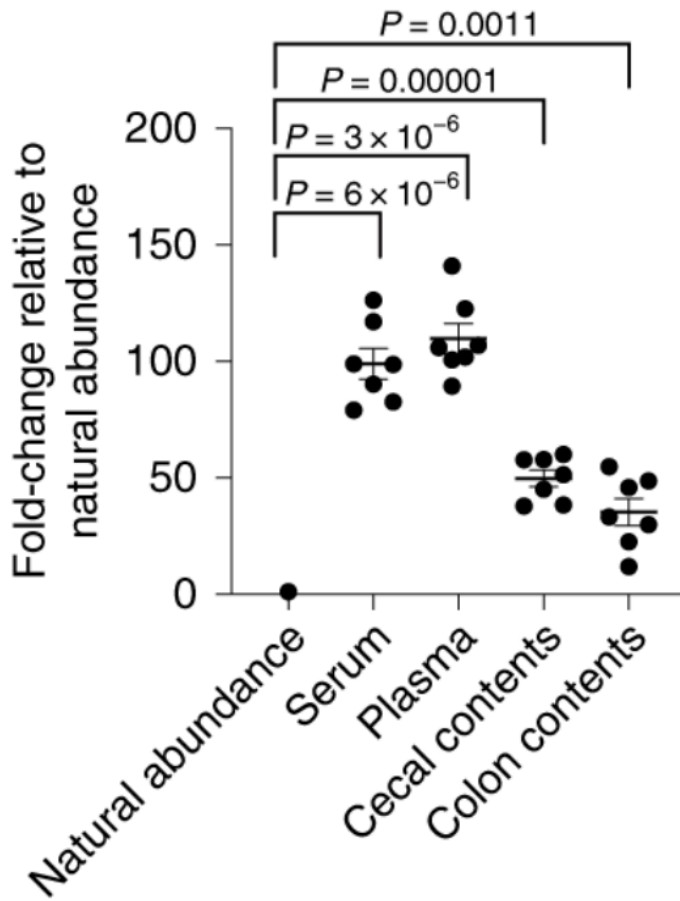
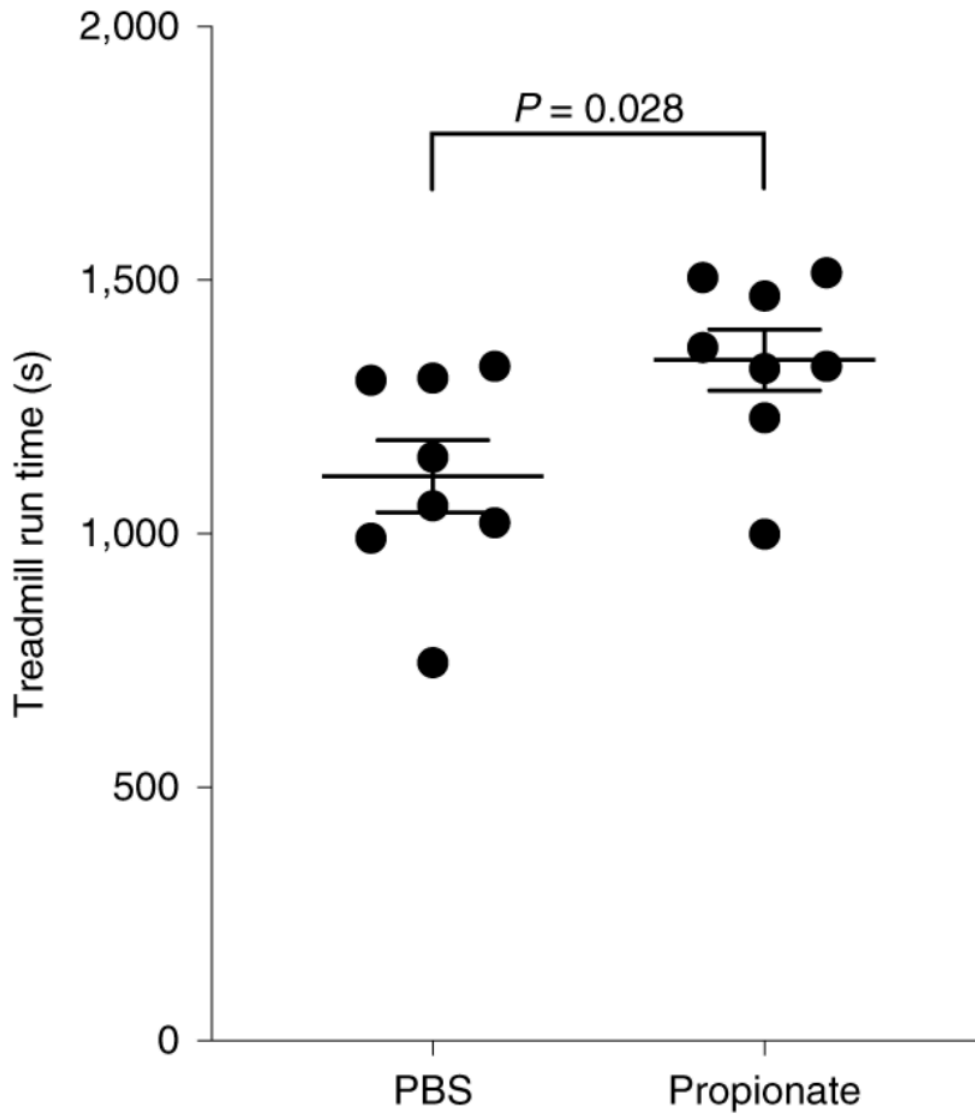


Figure 3.21: Abundance of  $^{13}\text{C}_3$  lactate quantified relative to the abundance of unlabeled lactate.

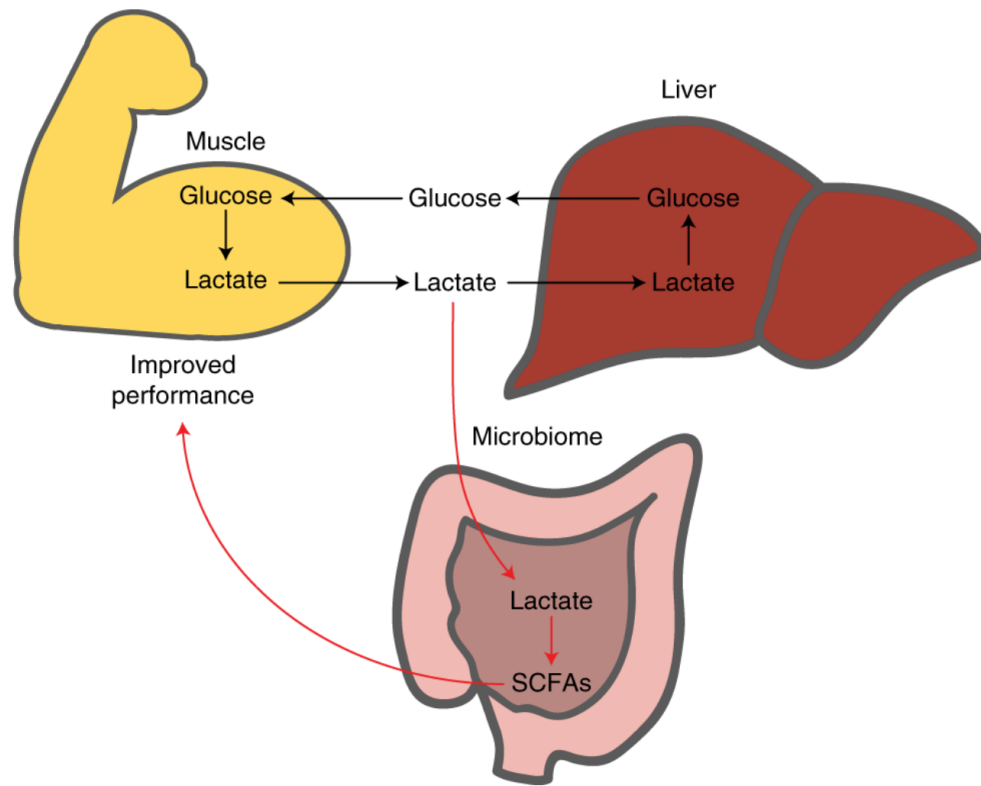


**Figure 3.22:**  $^{13}\text{C}_3$  lactate abundance normalized to the expected natural abundance of  $^{13}\text{C}_3$  lactate. The ratio of labeled/unlabeled lactate was quantified for experimental samples, as well as for the unlabeled lactate standard. Experimental samples are represented as the fold-change relative to the unlabeled standard. In Figure 3.21 and this figure, each mouse sample is represented as an individual point, with the central bar representing the mean and error bars representing s.e.m. ( $n = 7$ ). P values were determined by two-sided, one-sample t-test versus natural abundance.



**Figure 3.23:** Intracolonic infusion of propionate improves the maximum run time in mice. The graph shows the maximum run times out of 3 days of consecutive treadmill running. The jitter plot shows each mouse as an individual point, with the central bar representing the mean and error bars representing s.e.m. (n = 8). The P value was determined by two-sided unpaired t-test.





**Figure 3.24:** Proposed model of the microbiome–exercise interaction. Black arrows represent the well-known steps of the Cori cycle, where glucose is converted to lactate in the muscle, enters the liver via blood circulation, and is then converted back to glucose in the liver via gluconeogenesis. Red arrows represent the steps proposed in this work. First, lactate produced in the muscle enters the intestinal lumen via the blood circulation. In the intestine, it acts as a carbon source for specific microbes, including *Veillonella* species. This causes the observed bloom in intestinal *Veillonella*, as well as the production of SCFA byproducts (predominantly propionate), which are taken up by the host via the intestinal epithelium. The presence of microbiome-sourced SCFAs in the blood improves athletic performance via an unknown mechanism. Together, this creates an addendum to the Cori cycle by converting an exercise byproduct into a performance-enhancing molecule, mediated by naturally occurring members of the athlete gut microbiome.

spectrometry on isolates of these species (access Supplementary Table 6 from Appendix B).

Taken together, these results show that not only is the genus *Veillonella* enriched in athletes after exercise but the metabolic pathway that *Veillonella* species utilize for lactate metabolism is also enriched. This result raises the possibility that systemic lactate resulting from muscle activity during exercise may enter the gastrointestinal lumen and become metabolized by *Veillonella*.

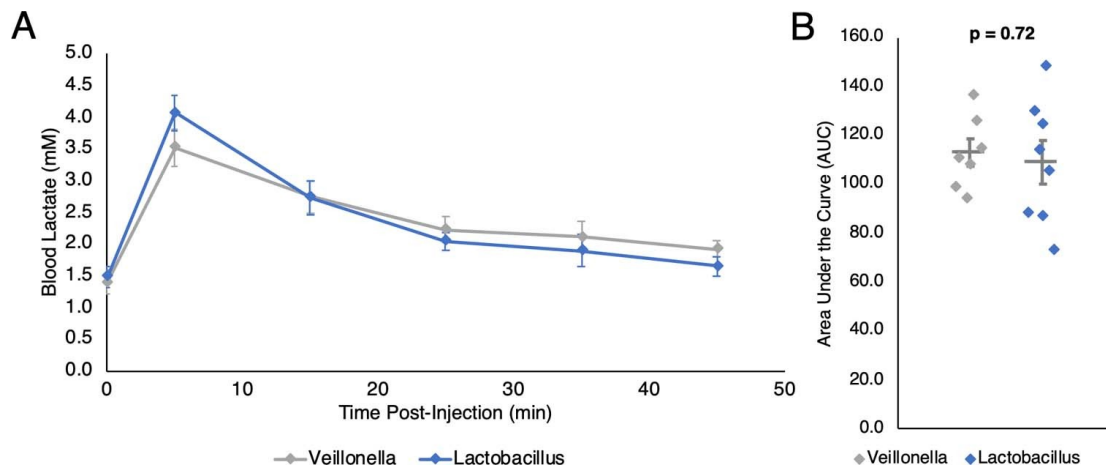
### 3.6 SERUM LACTATE CROSSES THE EPITHELIAL BARRIER INTO THE GUT LUMEN, AND COLORECTAL PROPIONATE INSTILLATION IS SUFFICIENT TO ENHANCE TREADMILL RUN TIME

Next, we sought to determine whether systemic lactate is capable of crossing the epithelial barrier into the gut lumen, as this has not been demonstrated before to our knowledge. To investigate this, we performed tail vein injections of  $^{13}\text{C}_3$  sodium lactate into mice colonized with either *V. atypica* or *L. bulgaricus*, and sacrificed them 12 minutes after injection. This time point was chosen because it was the earliest time at which we observed serum lactate levels return to baseline levels after tail vein injections in pilot experiments. At sacrifice, we immediately collected serum and plasma following cardiac puncture, and collected intestinal luminal contents by removing the colon and cecum from the mice and gently sampling the inner surface of the tissue. By performing liquid chromatography–mass spectrometry (LC-MS) on these tissues, we were able to identify  $^{13}\text{C}_3$ -labeled lactate present in both the serum and plasma, as well as in the lumen of the colons and ceca (Figure 3.20, Figure 3.21, and Figure 3.22; access Supplementary Table 7 from Appendix B). We were unable to detect any

$^{13}\text{C}_3$ -labeled propionate in these tissues; however, the 12-minute time point from tail vein injection to sacrifice is likely to have been insufficient time for labeled lactate crossing the gut barrier to be metabolized into propionate by the gut *Veillonella*.

As we have shown that serum lactate is capable of entering the intestinal lumen, we sought to determine whether *Veillonella* colonization may actively limit blood lactate levels by serving as a metabolic ‘sink’. To test the capability of *Veillonella* to accelerate blood lactate clearance in vivo, we performed intraperitoneal injections of sodium lactate in mice colonized with either *V. atypica* or *L. bulgaricus*, and monitored blood lactate over time. Neither the basal nor the peak lactate levels between the treatment groups were significantly different (Figure 3.25; access Supplementary Table 8 from Appendix B). The vast majority of lactate processing occurs in the liver (Phypers & Pierce, 2006), and although systemic lactate infiltrates the intestinal lumen, we did not observe a change in overall lactate clearance on inoculation with *Veillonella*.

Propionate has been shown to increase the heart rate and maximum rate of oxygen consumption, and to affect blood pressure in mice (Kimura et al., 2011; Pluznick et al., 2013; Pluznick, 2014), as well as raise the resting energy expenditure and lipid oxidation in fasted humans (Chambers et al., 2018). To test whether the exercise-enhancing effects of *Veillonella* may be attributable at least in part to propionate, we performed intrarectal instillation of propionate in our mouse treadmill model. Propionate was introduced intrarectally rather than orally because colonic absorption provides a more direct route for propionate to reach the systemic circulation, mirroring the location of *Veillonella*-sourced propionate. Intrarectal propionate instillation (n = 8) compared with saline vehicle (n = 8) resulted in increased treadmill run times similar to those of *V. atypica* gavage (P = 0.03; Figure 3.23).



**Figure 3.25:** a Mice were gavaged either *V. atypica* or *L. bulgaricus* and, 5 h later, injected with sodium lactate (750 mg kg<sup>-1</sup>). Blood lactate was measured 5 min postinjection and every subsequent 10 min (n = 8). Points are means ± s.e.m. b, Area under the curve (AUC) was determined for each mouse and compared between treatments. Each mouse is represented as an individual point, with the central bar representing the mean and error bars representing s.e.m. (P = 0.72 by two-sided unpaired t-test, n = 8).

As in the *Veillonella* gavage experiments, we ran the same panel of inflammatory cytokines on serum taken 40 min after treadmill running, but found no significant differences in cytokine levels (Figure 3.12; access Supplementary Table 4 from Appendix B). Therefore, the introduction of propionate into the colon is sufficient to result in an enhanced exercise phenotype via a mechanism that does not impact the inflammatory cytokines measured.

### 3.7 COUPLING COMPUTATION AND EXPERIMENTATION

Coupling computational approaches, multi'omic data collection approaches and experimental validation looks promising as a method to approach unvalidated metagenomic associations that have been proposed in the past decade. Acting on this principle, we observed that: (1) *Veillonella* abundance increased in the gut microbiome postexercise in two independent cohorts of athletes; (2) the

*Veillonella* methylmalonyl-CoA pathway is overrepresented in athlete metagenomic samples postexercise; (3) systemic lactate can cross the gut barrier into the lumen of the gut; (4) in a longitudinal AB/BA crossover study in mice, *Veillonella* inoculation improved treadmill performance; and (5) treadmill performance is improved in mice administered propionate via intracolonic infusion.

These data illustrate a model in which systemic lactate produced during exercise crosses to the gut lumen and is metabolized by *Veillonella* into propionate in the colon, which in turn serves to promote performance. Gut colonization of *Veillonella* may be augmenting the Cori cycle by providing an alternative lactate-processing method whereby systemic lactate is converted into SCFAs that re-enter the circulation (Figure 3.24). SCFAs are absorbed in the sigmoid and rectal region of the colon and enter circulation via the pelvic plexus, bypassing the liver and draining via the vena cava to reach the systemic circulation directly (Araghizadeh, F., Abdelnaby, A., 2012). Microbiome-derived SCFAs then augment performance directly and acutely, suggesting that lactate generated during sustained bouts of exercise could be accessible to the microbiome and converted to these SCFAs that improve athletic performance.

In conclusion, we have shown that the microbiome may be a critical component of physical performance, and highlight the benefits derived from it. An important question is how this performance-facilitating organism first came to be more prevalent among athletes. We propose that the high-lactate environment of the athlete provides a selective advantage for colonization by lactate-metabolizing organisms such as *Veillonella*. Future studies are needed to help explain why there is an apparent preference for *Veillonella* and not any of the many other lactate-metabolizing organisms. *Veillonella* in the physically active host therefore serves as a potential example of a symbiotic relationship in the

human microbiome.

### 3.8 COMPUTATIONAL METHODS

Wetlab methods that generated the data for the analyses below conducted by my two experimental co-first authors for this project are located in Appendix B.

#### 3.8.1 16S ANALYSIS

Each subject provided fecal samples on a daily basis, up to one week before and one week after the marathon (controls did not run in the marathon but provided fecal samples). Next, we extracted genomic DNA from these samples and performed 16S rDNA amplicon sequencing, followed by bioinformatic analysis, to obtain genus-level resolution of bacteria in each individual's microbiome (access Supplementary Tables 1 and 2 in Appendix B).

16S reads were processed with the DADA2 pipeline and phyloseq (Callahan et al., 2016; McMurdie & Holmes, 2013). There exist excellent alternatives to the software packages that were utilized (Langille et al., 2013). Default settings were used for filtering and trimming. Built-in training models were utilized to learn error rates for the amplicon dataset. Identical sequencing reads were combined through DADA2's dereplication functionality, and the DADA2 sequence-variant inference algorithm was applied to each dataset. Subsequently, paired-end reads were merged, a sequence table was constructed, taxonomy was assigned, and abundance was calculated at all possible taxonomic levels.

### 3.8.2 16S MIXED-EFFECT MODELING IN THE HUMAN COHORT

We constructed a series of GLMMs to predict *Veillonella* relative abundance in the marathon participants from both random effects (individual variation per athlete that manifests longitudinally) and fixed effects (United States Department of Agriculture (USDA) MyPlate consumption categories, protein powder supplementation, menstruation status, race, time, body mass index (BMI), weight, gender and age).

The longitudinal nature of the microbiome sampling, coupled with the unique lifestyles of athletes, means that diet, physical characteristics, age, gender, ethnicity and the menstrual cycle could potentially confound the association between postmarathon state and *Veillonella* relative abundance (Jurkowski et al., 1981; Pimentel et al., 2017). As some food compounds can selectively increase the relative abundance of *Veillonella*, 1,267 meal records logging every instance of food consumption over the course of the study (access Supplementary Table 1 from Appendix B) were quantified according to USDA MyPlate and associated with daily microbiome samples. LOOCV was performed for the GLMM analysis where the P value for the time coefficient was calculated for all permutations of eliminating one athlete, which revealed a general trend of no individual athlete driving significance, with one minor outlier (Figure 3.5; Wald Z-tests). To ensure that an arbitrary shuffling of participant labeling would not yield significant results, the GLMM was trained 1,000 times on input data with permuted labels, which generated uniformly distributed P values and showed the significance of the original labeling (Figure 3.5; Wald Z-tests). Thus, the observed significance of the association between *Veillonella* relative abundance and pre- and postmarathon state is likely not

confounded by any fixed effects. To test whether *Veillonella* has any phenotypic impact on running ability, we next introduced *Veillonella* to mice in a treadmill experiment.

Modeling of 16S *Veillonella* relative abundance for athletes participating in the marathon was done with the R nlme package (Pinheiro et al., 2014). A total of 1,267 meal records logging every instance of food consumption over the course of the study were quantified according to USDA MyPlate and associated with daily 16S samples by a nutritionist. Relative abundance was first modeled as:

$$\text{Abundance} = B_o + B_{\text{time}} + B_{\text{sex}} + B_{\text{weight}} + B_{\text{BMI}} + B_{\text{age}} + B_{\text{race}} + B_{\text{menstruation}} + B_{\text{vegetables}} + B_{\text{fruits}} + B_{\text{grains}} + B_{\text{protein}} + B_{\text{dairy}} + B_{\text{dietary protein supplementation}}$$

Subsequently, a second model was generated that included interaction terms of time:vegetables and time:menstruation. Significance was calculated for all of the coefficients included in the GLMM with Wald Z-tests (default calculation in the library utilized). Coefficients were created with the coefplot2 package (Bolker, 2012).

The code for the two models is provided below.

```
Model_1 <- lme(Veillonella~time + sex + weight + BMI + age + race + menstruation  
+ vegetables + fruits + grains + protein + dairy + dietary_protein_supp, random  
= ~1 | subjectID, data = marathon16S)
```

```
Model_2 <- lme(Veillonella~time + sex + weight + BMI + race + menstruation +  
vegetables + fruits + grains + protein + dairy + dietary_protein_supp +  
time:vegetables + time:menstruation, random = ~1 | subjectID, data = marathon16S)
```



Model predictions overlaid on the underlying data were visualized with the `ggplot2` R package (Wickham & Chang, 2015). Model results were validated with both LOOCV and permutation testing on shuffled labels.

### 3.8.3 TREADMILL RUN TIME MIXED-EFFECTS MODELING

Despite the high number of mice utilized in the AB/BA crossover experiment, comparisons of raw run times in this context could be confounded both by carryover effect (modeled as a sequence effect) inherent in the longitudinal study design, as well as unavoidably high intermouse variation. To account for this, we constructed a series of GLMMs predicting run time. These models incorporate both random effects (individual variation per mouse that manifests longitudinally) and fixed effects (treatment day, treatment sequence and treatment type given). Modeling was conducted with the `Rnlme` package (Pinheiro et al., 2014). Visualization of coefficients was conducted using the `coef2plot` R package (Bolker, 2012). Visualization of predictions overlaid on data was conducted using the `Rggplot2` package (Wickham & Chang, 2015).

Visualization of all longitudinal data points with the GLMM predictions overlaid showed both the effect of *V. atypica* increasing performance on both sides of the crossover when aggregated by treatment group (thick lines), and the trends for each of the 32 individual mice (thin lines) (Figure 3.11). LOOCV was performed for the GLMM analysis where the P value for the *V. atypica* treatment coefficient was calculated for all permutations of eliminating one mouse, which revealed that no individual mice were driving significance (Figure 3.8; Wald Z-tests). To ensure that an arbitrary shuffling of mouse labeling would not yield significant results, the GLMM was trained 1,000 times

on input data with permuted labels, which generated uniformly distributed P values and showed the significance of the original labeling (Figure 3.8; Wald Z-tests). This longitudinal modeling approach allows us to interpret that, as the treadmill runs were conducted back to back each week on subsequent days, the mice in aggregate had decreasing run times as the time to exhaustion decreased (visible as a slope of predictions in Figure 3.11), while *V. atypica* treatment independently increased run times (visible as the crossover of predictions showing the *Veillonella* treatment group having longer times to exhaustion on both sides of the crossover in Figure 3.11). To identify possible biological mechanisms for the *Veillonella* effect, we quantified levels of various inflammatory cytokines in the blood immediately following the treadmill run. We observed that several proinflammatory cytokines, including tumor necrosis factor- $\alpha$  and interferon- $\gamma$ , were significantly reduced in *V. atypica*-treated mice compared with both the baseline and the control treatment (Figure 3.12; access Supplementary Tables 4 and 13 from Appendix B). In a separate experiment, we quantified levels of the muscle glucose transporter GLUT<sub>4</sub> to assess the effects on muscle physiology, but found no difference between the *V. atypica* treatment and control (Figure 3.13). Together, taking into account intermouse variation, the longitudinal study design and the possible carryover effects of an AB/BA crossover trial, *V. atypica* treatment causes substantial increases in treadmill run time in mice.

The models were constructed to predict treadmill run time in the AB/BA crossover experiment to include the treatment effect of *Veillonella*, period effects (time of treatment), carryover effects due to the treatment crossover, and effects for naturally occurring mouse variation. In general, we can model expected run time as:

$$\text{Sequence: } V. atypica \rightarrow L. bulgaricus \text{ Week 1: } \mu + \pi_1 + \alpha_A \text{ Week 2: } \mu + \pi_2 + \alpha_B + \lambda_A$$

Sequence: *L. bulgaricus* → *V. atypica* Week 1:  $\mu + \pi_1 + \alpha_B$  Week 2:  $\mu + \pi_2 + \alpha_A + \lambda_B$

Where  $\alpha_A$  and  $\alpha_B$  are treatment effects,  $\lambda_A$  and  $\lambda_B$  are carryover effects, and  $\pi_1$  and  $\pi_2$  are period effects.

We initially attempted to model carryover effect as a sequence effect or a period-specific treatment effect (interaction term). The R code for the models is provided below:

```
Model_1 <- lme(seconds_run ~ treatment + sequence + period, random
= ~1 | subject, data = datain)

Model_2 <- lme(seconds_run ~ treatment * period, random
= ~1 | subject, data = datain)
```

By gauging the correlation of coefficients, we selected Model\_1 for the analysis in Figure 3.11.

#### 3.8.4 METAGENOMIC ANALYSIS

All of the steps in the processing of raw metagenomic data were done utilizing the Aether package (Luber et al., 2017). Raw reads were de novo assembled using megahit (Li et al., 2015). Open reading frames and annotations were generated using prokka (Seemann, 2014). A gene family catalog was generated from the called open reading frames at 95% identity utilizing the CD-HIT software package (Fu et al., 2012). A raw abundance count matrix was generated utilizing the gene family catalog, Bowtie 2 and SAMtools (Langmead & Salzberg, 2012; Li et al., 2009). The raw abundance count matrix was normalized both by sample and by gene length (Qin et al., 2012). Metabolic pathways were queried using MetaCyc, and EC IDs were pulled from prokka annotations (Seemann, 2014;

Caspi et al., 2010). R was utilized to perform the majority of statistical tests, with the exception of pairwise analysis of variance (ANOVA) tests, for which the SciPy library in Python was used (Jones et al., 2001). Root mean square error calculations were performed using the plotrix package (Lemon et al., 2007).

### 3.8.5 METAPHLAN2 TAXONOMY IN METAGENOMICS DATA

Putative taxonomic abundances were calculated with MetaPhlAn2 (Truong et al., 2015) and found to have the same association between *Veillonella* and exercise status as the previous marathon runner results ( $P = 0.03$ ; Figure 3.14).

### 3.8.6 ANNOTATIONS

To compare trends in the aggregate microbiome with the metabolic processes of microbes that had elevated 16S abundance in the previous experiment, a pairwise ANOVA was performed on all 2.3 million genes in the catalog to look for significant differences before and after exercise. A total of 396 gene families with unique annotations showed statistically different relative abundance ( $P < 0.005$ ). While false discovery rate correction did not yield significant individual genes, of these 396 gene families, 391 share functional annotations with the reference assemblies of the *V. atypica*-type strain on NCBI. Of the significant genus-level results from the 16S data, *Veillonella* has extremely high-quality assemblies of cultured isolates.

Significant alleles are present in each of the 87 samples (Figure 3.14). Interestingly, when all 396 significant alleles are segregated by exercise state and sample, discordant shifts of relative abundance

are observed (Figure 3.14). This suggests that changes in global microbiome function are associated with *Veillonella* abundance, and that conserved *Veillonella* genes may generally play metabolic roles.

### 3.8.7 COMPARATIVE GENOMICS

Genome annotations were retrieved from NCBI reference genomes. Phylogenetic trees were generated from NCBI taxonomy and visualized with phylo.io (Robinson et al., 2016). Heat maps were generated with the pheatmap package in R (Kolde, 2012).

### 3.8.8 GENE CATALOG CREATION

Raw reads were processed and de novo assembled into 4,802,186 contigs (Luber et al., 2017; Li et al., 2015). A total of 4,792,638 total open reading frames were called, which were subsequently clustered into 2,288,155 gene families with a threshold of 95% identity to create a gene catalog alongside putative annotations assigned by homology (Seemann, 2014; Fu et al., 2012). Of these gene families, 801,307 were assigned annotations and 1,486,948 were putatively classified as hypothetical proteins. Comparing annotation state versus gene family size yields the expected result that larger families, which are likely to be present in more microbes, tend to have many more annotations (Figure 3.14). Raw reads were then aligned back to the gene catalog to create a raw count abundance matrix (Langmead & Salzberg, 2012; Li et al., 2009). This matrix was normalized both per sample and by gene length to create a relative abundance matrix (Qin et al., 2012).

### 3.8.9 PATHWAY ELUCIDATION

Reactions involved in the breakdown of lactate to both propionate and acetate were manually associated with EC IDs using MetaCyc (Caspi et al., 2010).

## 3.9 STATISTICS

### 3.9.1 16S ANALYSIS

For Figure 3.1 and Figure 3.2, Wilcoxon rank-sum tests with continuity correction were used to investigate differences in taxonomic composition before and after exercise. The mean *Veillonella* abundance was 0.9 orders of magnitude greater 1 day post-exercise compared with 1 hour before exercise.

For Figure 3.3 and Figure 3.4, longitudinal data were modeled using a GLMM approach. In our model, the random effect was individual variation per marathon runner. Fixed effects are shown in Figure 3.4. An advantage of this type of statistical analysis is that it can account for the large variation between marathon participants in this type of study.

To determine statistical significance, a Wald Z-test was used to assign P values to coefficients in the GLMM. No outliers were removed in this analysis.

For Supplementary Table 1 (access from Appendix B, P values were generated using Welch's t-test (unequal variances t-test).

### 3.9.2 CROSSOVER MOUSE EXPERIMENT

For Figure 3.7, each animal was treated with both *V. atypica* and *L. bulgaricus* as part of the AB/BA crossover. Because all 32 animals were treated twice and compared between treatments, the P value was generated using a paired t-test ( $P = 0.022$ ). The normality assumption was assessed via Shapiro–Wilk’s normality test ( $P = 0.67$ ), validating the use of the t-test.

For Figure 3.11, longitudinal data were modeled using a GLMM approach. In our model, the random effect was individual variation per mouse. Fixed effects were treatment effect, period effect (the time point at which measurements were made) and carryover/sequence effect (if the order of treatments in the crossover affected later results). An advantage of this type of statistical analysis is that it can account for the large variation between mice in this type of study.

Figure 3.11 shows the number of seconds run until exhaustion at six time points, with each of the 32 mice having one measurement per time point. For each treatment order (LLL VVV and VVV LLL), the GLMM was fitted both to each individual mouse (skinny blue and red lines; note that these are all parallel for mice in the same treatment order—the space between these lines represents the ‘random effect’ of natural variation between mice) and all mice with the same treatment order (thick blue and red lines).

To determine statistical significance, a Wald Z-test was used to assign P values to coefficients in the GLMM. No outliers were removed from this analysis.

### 3.9.3 METAGENOMIC ANALYSES

For Figure 3.16 and Figure 3.15, P values for individual genes were generated utilizing pairwise ANOVA comparing the relative abundance before and after exercise. Non-significant families were associated with homologs common in other microbes that do not change in abundance. To determine the significance of potential over-representation, 1,000 global EC IDs were randomly selected, and mean differences in relative abundance between samples taken before and after exercise were calculated. These EC IDs were used to construct an odds table to determine the probability of having a set of eight selected EC IDs with increases in mean gene level relative abundance after exercise. This calculation determined that the relative abundance changes in Figure 3.15 B-I are significant ( $P = 0.00147$ , Fisher's exact test for count data).

### 3.9.4 $^{13}\text{C}_3$ FLUX-BALANCE EXPERIMENTS

For Figure 3.22, P values were generated using a one-sample t-test. Ratios of labeled/unlabeled lactate from samples were compared with the expected ratio determined mathematically. Each sample was independently compared with the expected ratio, then multiple hypothesis correction was performed using the false discovery rate correction method of Benjamini and Hochberg (serum,  $P = 0.00001$ ; plasma,  $P = 0.00001$ ; cecum content,  $P = 0.00001$ ; colon content,  $P = 0.001$ ).

### 3.9.5 INTRACOLONIC INSTILLATION OF PROPIONATE EXPERIMENT

For Figure 3.23, the P value ( $P = 0.028$ ) was generated using Welch's t-test (unequal variances t-test).



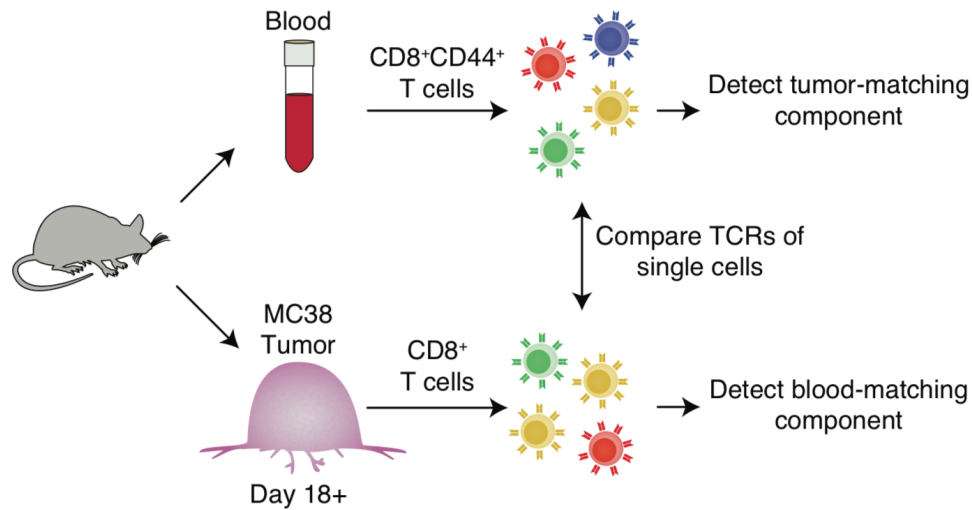
# 4

## Single Cell Immunology of Cancer

USING THE T CELL RECEPTOR AS A MOLECULAR BARCODE, we identify a population of “tumor matching” (TM) CD8<sup>+</sup> T cells in mouse peripheral blood that share TCRs with tumor infiltrating lymphocytes (TILs) in MC38 tumors.\* These TM cells have a unique transcriptional profile com-

---

\*Portions of this chapter (application of shiny app to Layilin data) were previously published in the *Journal of Experimental Medicine* (Mahuron et al., 2020), and additional portions of it were under revision in a separate publication at the time of dissertation submission.



**Figure 4.1:** Overview of how both Single Cell Gene Expression (GEX) and Single Cell TCR data is collected from both mouse tumor and mouse blood.

pared to T cells in the blood that do not have TCRs shared with TILs and represent all TIL T cell phenotypes. Using machine learning, we leveraged this unique transcriptional program to identify a marker panel that can recover TM cells via flow cytometry and subsequently validated top markers using CITE-Seq in an independent mouse cohort. A schematic of the data collected to enable these analyses is provided in Figure 4.1.

#### 4.1 SINGLE CELL LANDSCAPES OF BLOOD AND TUMOR

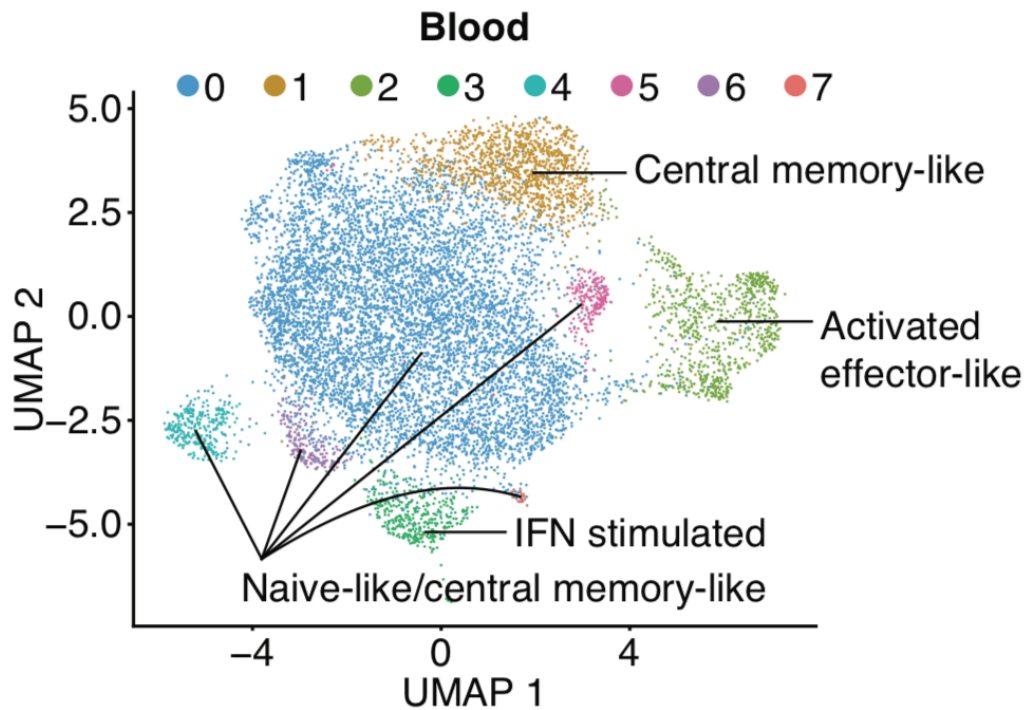
We performed scRNAseq with TCR sequencing on paired peripheral blood and tumor from B6 mice that received MC38 colon adenocarcinoma tumor cells subcutaneously in the flank. In order to enrich for antigen-experienced cells in the blood, peripheral blood CD8<sup>+</sup> T cells were sorted for CD44<sup>+</sup>. Samples were computationally integrated and the single cell transcriptomic landscape was

characterized of sorted CD44<sup>+</sup> CD8<sup>+</sup> T cells in both blood (n=10,289 cells; Figure 4.2) and tumor infiltrating lymphocytes (TILs) (n=8540 cells; Figure 4.3). 10–15 Phenotypes were defined based on gene signatures, and included a range of differentiation states including central memory, effector, effector memory, IFN-stimulated, cycling, and exhausted (Singer et al., 2017). We overlaid TCR clones between the tissues and observed a notable population of CD8<sup>+</sup> T cells in the blood that shared matching TCRs with CD8<sup>+</sup> T cells in the tumor, which hereafter are referred to as “tumor-matching” or TM cells (Figure 4.4). When overlaying the location of TCRs in the tumor that have matches in blood, cells were detected in every transcriptional cluster in the tumor (Figure 4.5), suggesting that the TM cells present in blood provide a broad window into the clones in the tumor microenvironment. Contextualizing the TM cells in the context of the entire cell landscape revealed signatures indicative of stronger proliferation and activation capacity than non-matching cells. These data suggested that TM cells in the blood provided a wide window into many phenotypes of T cell differentiation in the tumor, and that they could be clinically useful if they could be distinguished from the remainder of cells in blood. Indeed, when we created an integrated landscape of all cells from both tissues and applied trajectory inference, we observed the TM cells fitting into distinct lineage states (cycling and exhausted), but also being members of lineages that represent all T cell phenotypes in the tumor, which goes against the current understanding in the field.

## 4.2 CURRENT UNDERSTANDING OF THE T CELL RECEPTOR REPERTOIRE

The results presented in this chapter are slightly surprising when taken in context with recent studies in the field. Prominent work looking at specificity groups in the TCR repertoire has suggested that said repertoire is incredibly vast, and the TCRs do not need to have perfectly matching alpha and beta chains in order to recognize a common antigen (Glanville et al., 2017). Rather, this work suggests that TCRs containing merely similar motifs in the alpha and beta chains are able to recognize common antigens, and that it is a rare occurrence to find perfect sequence matches in both chains in different T cells. A updated version of the method developed by the same group that is designed to match these motifs between T cells to identify common antigen specificity found similar results (Huang et al., 2020).

A recent paper with a a sample collection strategy of sequencing human tumor, blood, and longitudinal human tumor introduced the idea of “clonal replacement”, where large clones in the tumor do not persist and instead are replaced over time, perhaps as a response to tumor immune evasion (Yost et al., 2019). Due to this finding of clonal replacement, this study claimed not to find representative matching clones for all major tumor cell populations in paired blood, which differs significantly from our study. This could potentially be because of differences between human and mouse studies, but it is also possible that the clonal replacement study did not perform deep enough sequencing of their  $\zeta'$  TCR library to properly identify cells with perfectly homologous chains between tissues. Thus, in the context of recent work in the field, what is presented in this chapter is significantly different and raises the question of great relevance querying whether our mouse find-

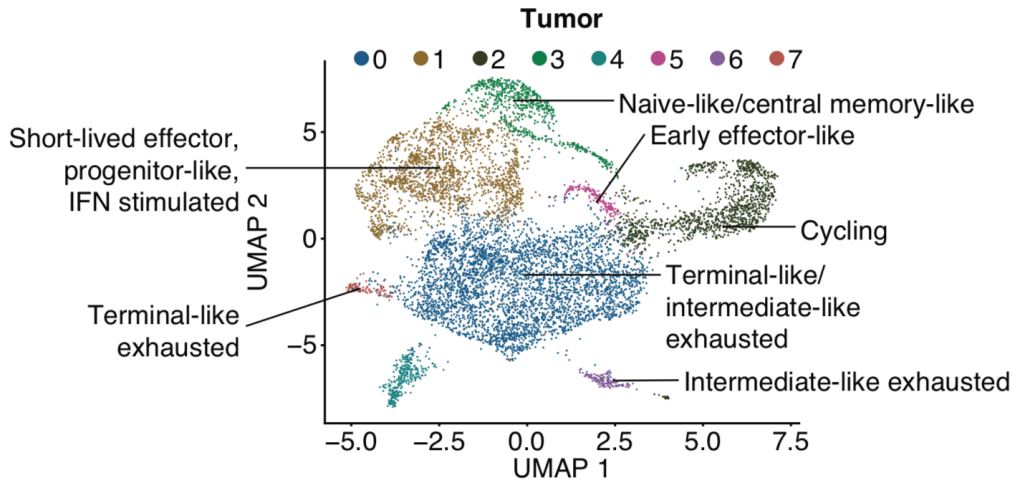


**Figure 4.2:** Clustering and UMAP visualization of peripheral blood (n=10,289 cells, paired MC38 tumor also collected) on day 18+ post tumor cell implantation. Data integrated from three mice (M1, M2, and M3) from two independent experiments (experiment 1 = M1, experiment 2 = M2, M3). Colors indicate distinct transcriptional clusters determined using Seurat clustering. Labels marking each cluster indicate the phenotypic description of the cluster based on gene expression data (see Methods).

ings will generalize to humans.

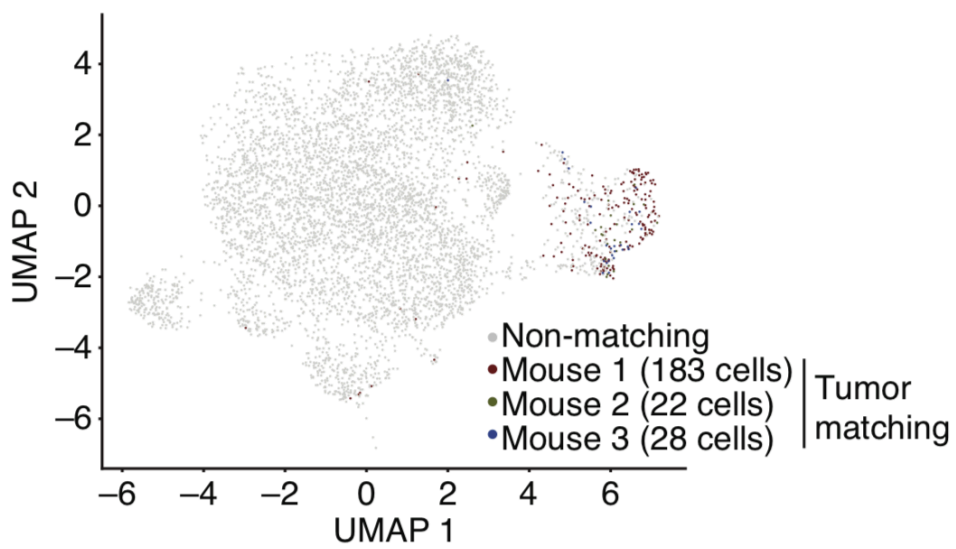
#### 4.3 USING MACHINE LEARNING TO MAKE PREDICTIONS ABOUT CLONAL T CELL LINEAGES

Considering the observation that TM cells in the blood had a unique transcriptional profile compared to non-matching cells, we determined whether classifiers could be constructed that were



**Figure 4.3:** Clustering and UMAP visualization of MC38 tumors (n=8,450 cells, paired blood also collected) on day 18+ post tumor cell implantation. Data integrated from three mice (M1, M2, and M3) from two independent experiments (experiment 1 = M1, experiment 2 = M2, M3). Colors indicate distinct transcriptional clusters determined using Seurat clustering. Labels marking each cluster indicate the phenotypic description of the cluster based on gene expression data (see Methods).

trained on gene expression (GEX) data to predict whether blood cells match to the tumor with training labels defined by TCR matches (Figure 4.6) (Pedregosa et al., 2011; Luber, 2016). Subsequently, to explore the possibility of tracking these cells using standard flow cytometry-based sorting methods, classifiers were created with training data limited to only cell surface markers (Figure 4.7, cross validated AUC=0.9849). Top contributing genes to these predictions were identified using COMET (Figure 4.8) (Delaney et al., 2019). Based on these results, a method was applied that utilizes the XL-MHG test to robustly predict flow-cytometry marker panels for specific cell populations to the TM cells (Delaney et al., 2019). Figure 4.10 shows the top predicted markers assigned into putative biological function. Flow cytometry was used to validate that many of the markers that were detected at the transcript level were also detected at the protein level. Moreover, many of the markers detected



**Figure 4.4:** UMAP showing blood CD8+ T cells colored in the integrated mouse data that have a TCR matching to CD8+ T cells found in tumor (referred to as TM cells), colored by each mouse. Numbers next to each sample in the legend indicate the number of TM cells recovered in each mouse. Grey indicates CD8+ T cells that do not have a TCR matching to T cells in tumor (referred to as non-TM).

# Tumor

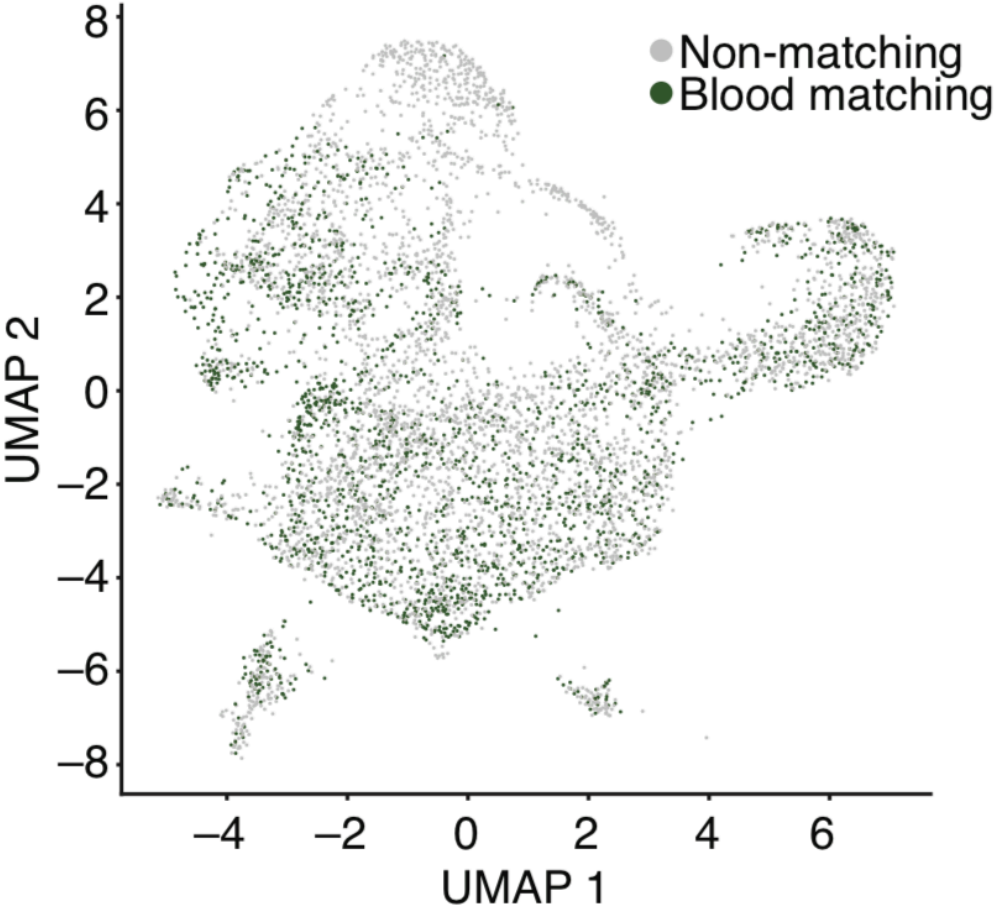
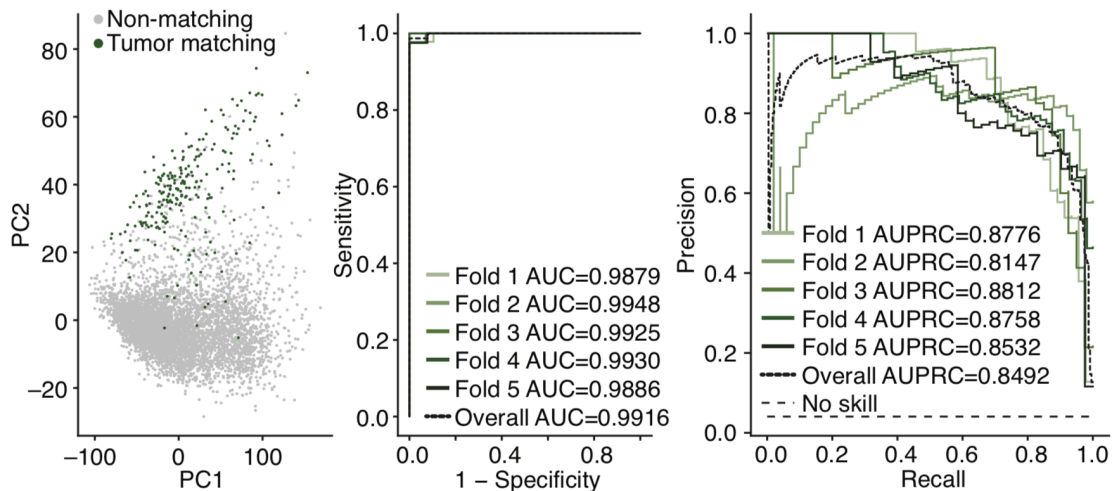


Figure 4.5: UMAP showing tumor CD8+ T cells colored in the integrated mouse data that have a TCR matching to CD8+ T cells found in blood (referred to as TM cells).

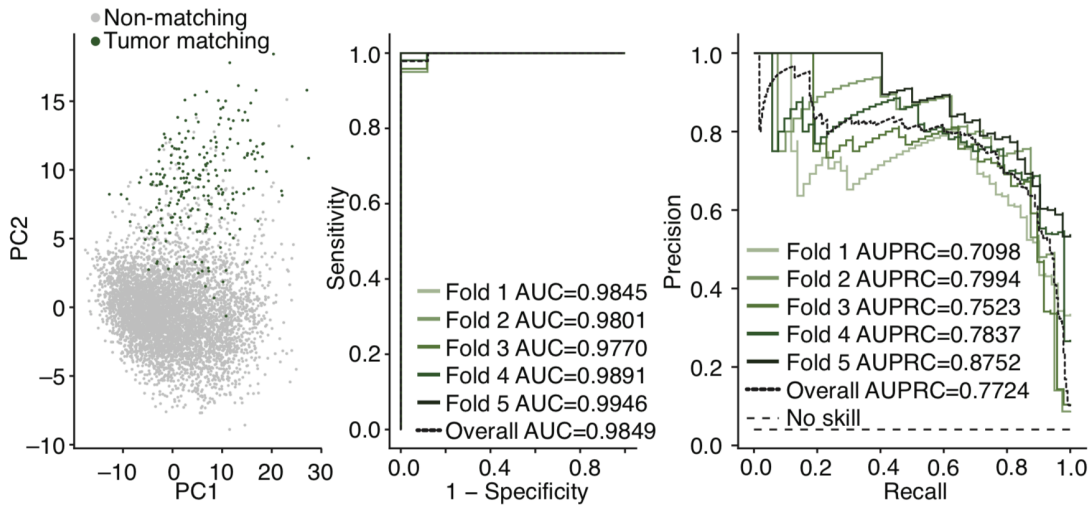




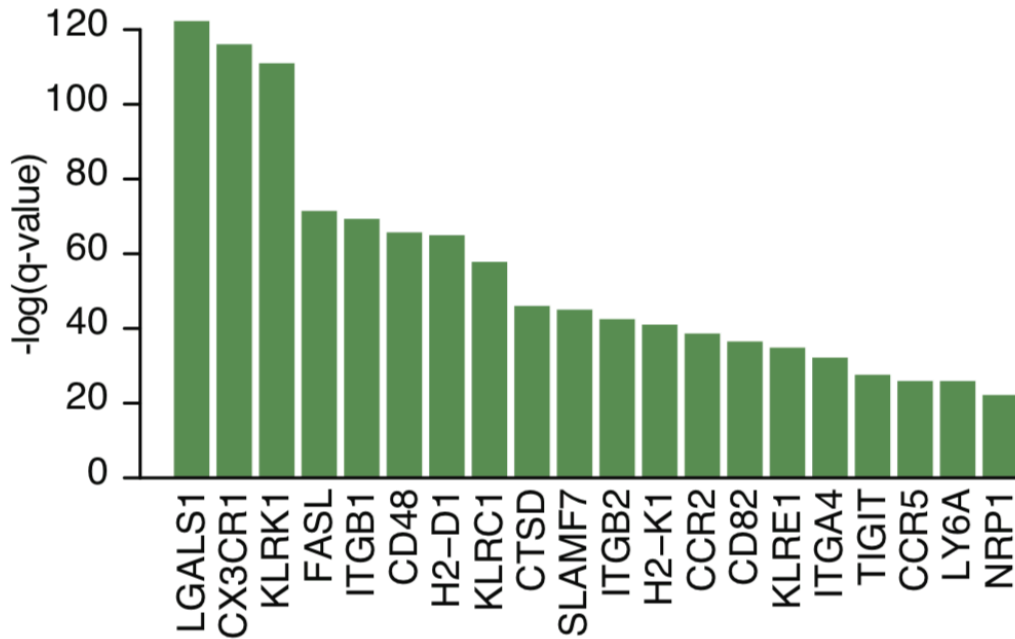
**Figure 4.6:** Logistic regression showing classification of cells as TM or non-TM based on all genes. Shown are the first two principal component projections (left), ROC curves (middle), and the Recall-Precision plots (right) with 5-fold cross validation.

in our scRNAseq data were also detected as co-expressed on CD44<sup>+</sup> CD8<sup>+</sup> T cells by flow cytometry (Figure 4.9). In order to confirm that detection of surface proteins were successfully enriching for our TM population based on having TCRs that matched to TIL, we performed a validation cohort experiment and sequenced GEX, TCR, and CITE-Seq antibodies for predicted surface markers (CD39, CX3CR1, and NKG2D) simultaneously and observed that TM cells were subsequently overrepresented and strongly aligned to the protein expression for all of these markers (Figure 4.11) (Stoeckius et al., 2017).

Our algorithmic approach to generate marker panels (perhaps patient specific) to identify TM cells coupled with future longitudinal studies will be widely applicable to the field in assisting in the creation of diagnostics that aim to predict response to immunotherapy. These data show the potential for the use of single-cell RNA-seq to identify TM CD8<sup>+</sup> T cells in blood that provide a wide



**Figure 4.7:** Logistic regression showing classification of cells as TM or non-TM based on a pre-selected list enriched for surface-marker genes (Chihara et al., 2018). Shown are the first two principal component projections (left), ROC curves (middle), and the Recall-Precision plots (right) with 5-fold cross validation



**Figure 4.8:** Top genes as calculated using COMET (Delaney et al., 2019) that contribute to performance of the classifiers (Figure 4.6 and Figure 4.7) utilized to predict tumor matching cells in blood.

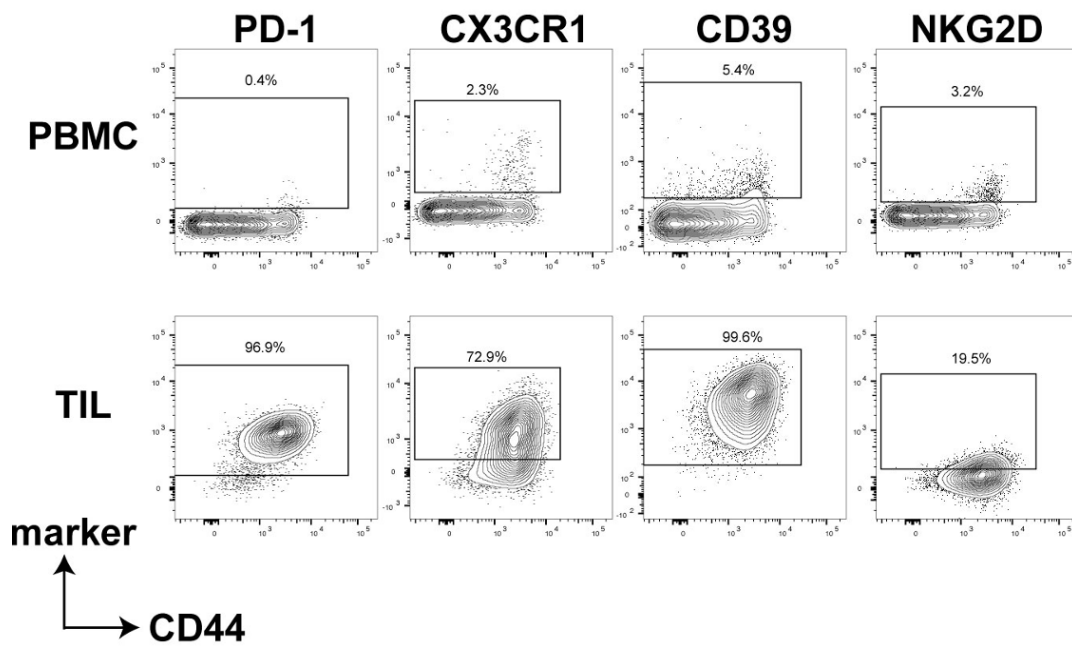
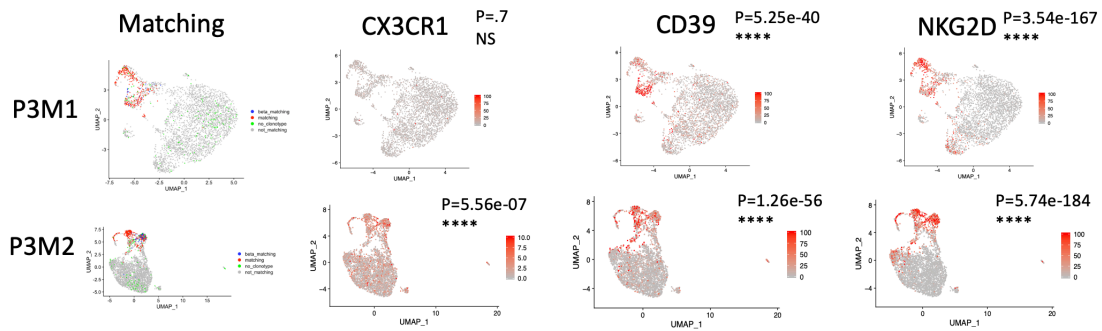


Figure 4.9: Comparison of CX3CR1, CD39, and NKG2D on CD8<sup>+</sup> T cells in blood vs MC38 tumors. This plot was created by Kristen Pauken.

Negative Regulatory Pathways
Inhibitory Receptors: Pcd1, Lag3, Tigit, Cd160 Other: Entpd1, Nrp1, Lgals1, IL10ra
Trafficking
Chemokine receptors: Cx3cr1, Ccr2, Ccr5, Cxcr6 Integrins: Itga4, Itgb1, ItgaL, Itgb2, Itgax, Itgb7 Other: Selplg
Postive Regulatory Pathways
General Activation: Cd3e, Thy1, CD247, B2m Costimulation: Icos, Cd2, Cd226, Slamf7 Cytokine receptors/signaling: Il18r1, Il18rap, Il2rg, Il2rb
NK receptors
Klrk1 (NKG2D) Klrc1 (NKG2A) Klrd1 (CD94) Klre1 (NKG2I)

**Figure 4.10:** Top markers from the classification and XL-MHG steps assigned to putative biological function from the literature (Negative Regulatory Pathways, Trafficking, Postive Regulatory Pathways, and NK Receptors). This plot was created by Kristen Pauken.



**Figure 4.11:** Results from 2 mice where GEX, TCR, and CITE-Seq (CX3CR1, CD39, and NKG2D antibodies) was collected; this data shows that the computationally selected markers have high expression on “tumor matching” cells from blood. Matching plots made by overlaying TCR data on a UMAP calculated from GEX data. P Values generated from two sided t tests considering the null hypothesis that marker expression and tumor matching status as determined by TCR sequence matching are independent.

window in the immunologic landscape of the tumor, and to provide new means to track these cells and monitor their responses to therapy. Future work needs to be done to make these computational methods that work well in mouse also work in humans, where T cell phenotypes are more complex due to the complex antigen stimulation history and lifetimes of senescent T cell memory.

#### 4.4 COMPUTATIONAL METHODS

Wetlab methods that generated the data for the analyses below conducted by my experimental co-first authors for this project are located in Appendix C.

##### 4.4.1 DEMULTIPLEXING AND READ PROCESSING

Raw reads were processed using cellranger v3.0.2 to generate raw counts matrices of gene expression and csv files corresponding to TCR clonality. Aether version 1.0 (Luber et al., 2017) was used to

process certain resource heavy jobs on compute instances rented from Amazon Web Services. TCR data were additionally processed for downstream use with a group of scripts written by the authors to determine for each cell barcode the alpha and beta chain attributes detected.

#### 4.4.2 COMPUTATIONAL PROCESSING OF GENE EXPRESSION DATA

All analyses were conducted using R version 3.6.1 and Seurat version 3 with additional utilization of the dplyr, data.table, ggplot2, cowplot, viridis, gridExtra, RColorBrewer, ggpubr, ggrepel, gtools, DescTools, doParallel, doSNOW, and tibble packages (R Core Team & Others, 2013; Butler et al., 2018; Wickham & Francois, 2015; Dowle et al., 2019; Wickham & Chang, 2015; Wilke, 2016; Auguie & Antonov, 2017; Neuwirth, 2014; Kassambara, 2018; Slowikowski, 2017; Warnes et al., 2015; Signorell et al., 2016; Analytics & Weston, 2013; Müller & Wickham, 2018). Seurat objects were created with the min.cells parameter set to 3 and the min.features parameter set to 400. Filtering cells based on expression of housekeeping genes was conducted using the human and mouse (where appropriate) gene lists maintained by the Seurat developers (available on the Satija lab website), with cells passing the filtering criteria if they had expression greater than 0 for more than half of the genes in the list. Subsequently, the MitoCarta database from the Broad institute was utilized to filter out cells based on expression of mitochondrial genes (Calvo et al., 2016). Cells were filtered out if they expressed more than 500 of the 1158 mitochondrial genes in human, or if the number of mitochondrial genes expressed was higher than 2 standard deviations from the mean in mouse.

Data were normalized using the default Seurat function (generating log-transformed transcripts-per-10K read measurements) followed by scaling, and variable genes were found using “ExpMean”

for the `mean.function` parameter and “LogVMR” for the `dispersion.function` parameter. The `RunPCA` function was run utilizing 50 principal components and then the `FindNeighbors` function was run using 30 dimensions. Subsequently, the `FindClusters` function was run with a resolution aiming to generate 5-7 biologically meaningful clusters per sample. When applicable, samples were integrated using the `SCTransform` method (Hafemeister & Satija, 2019). Upon obtaining transcriptional clusters in the integrated datasets, upregulated genes associated with each cluster were determined via the Wilcoxon Rank Sum test implemented in the `FindAllMarkers` function in Seurat.

#### 4.4.3 SINGLE-CELL TCR AND CLONAL ANALYSIS

Cells for which at least one alpha and one beta chain were annotated in the TCR data were determined as “tumor/blood-matching” or “tumor/blood-non-matching” based on whether there was a cell in the paired tissue data that had the exact same alpha and beta chain composition as the given cell. Only cells that had at least one alpha chain and one beta chain annotated were included in all of the analyses and visualizations comparing “matching” to “non-matching” cells. Two cells were assigned to be in the same clone if they had the exact same sets of alpha and beta chains assigned. This strict definition was used to ensure each pair of cells within the same clone has complete similarity of the TCR chains detected, and hence is with high probability derived from the same T cell clone.

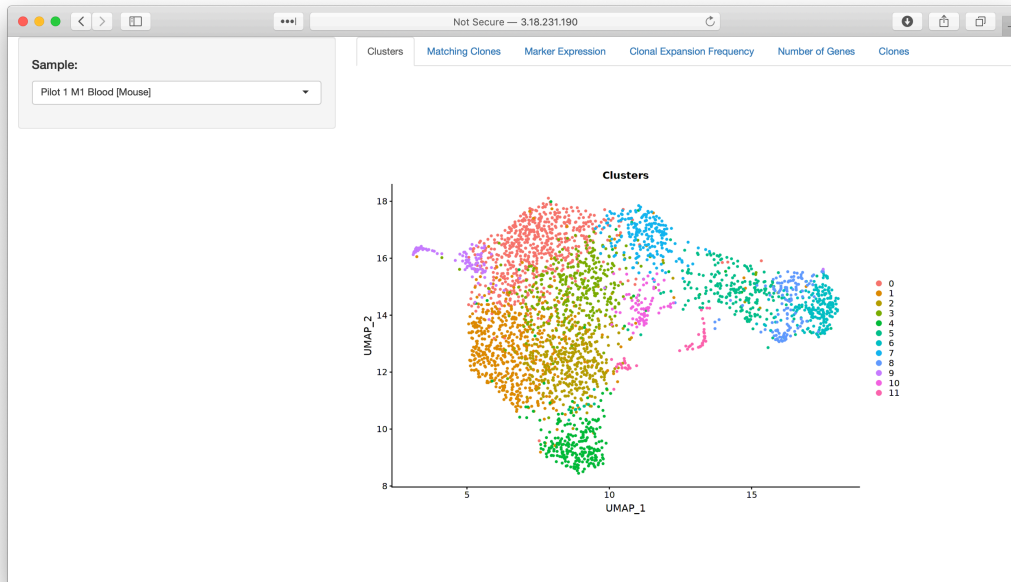
#### 4.5 SHINY APP

In the process of exploring the data for the previously described project and building machine learning models to predict whether a cell in blood is tumor matching or not, it became clear that while

there are great tools for processing and visualizing 10X 5' sequencing data (Satija et al., 2017; Butler et al., 2018; Wolf et al., 2018; Hindson et al., 2014; Freytag et al., 2018), there do not exist quality tools for visualizing 5' TCR data in the context of gene expression data. To allow for rapid hypothesis generation and collaboration with experimental collaborators, I built a Shiny web application to visualize TCR data, matching clonality, and other related plots (Beeley, 2016; R Core Team & Others, 2013).

The Shiny web application can be accessed at <http://3.18.231.190:3838/matching/>. Additionally, as this website is not guaranteed to be hosted for eternity, the code to generate all application views described below is also included in Appendix D. Data outputted from the 10X cellranger pipeline is processed with the TCR clone pipeline (available at <https://github.com/MSingerLab/blood-tcr-pipeline> and in Appendix D), then integrated into a Seurat object and lightweight CSVs metadata files (Satija et al., 2017) which are accessed by the application to reactively generate plots. Cells that are “matching” according to clonotype in the TCR pipeline are defined as cells between tissues that have exact match in TCR sequence and have at least one alpha and one beta chain. There are six views (with a tab for each in the menu bar) in the Shiny Application: viewing louvain clustering (De Meo et al., 2011) for a selected tissue (Figure 4.12), viewing “matching” T cells between two selected paired tissues (Figure 4.13), viewing marker expression for a selected tissue (Figure 4.14), viewing clonal expansion for a selected tissue (Figure 4.15), viewing number of genes per cell for a selected tissue (Figure 4.16), and viewing individual clones for a selected tissue (Figure 4.17). All of the views in the Shiny application are reactive, meaning that the page will automatically update when user input (i.e. selected tissue) is changed; plots are generated on the fly



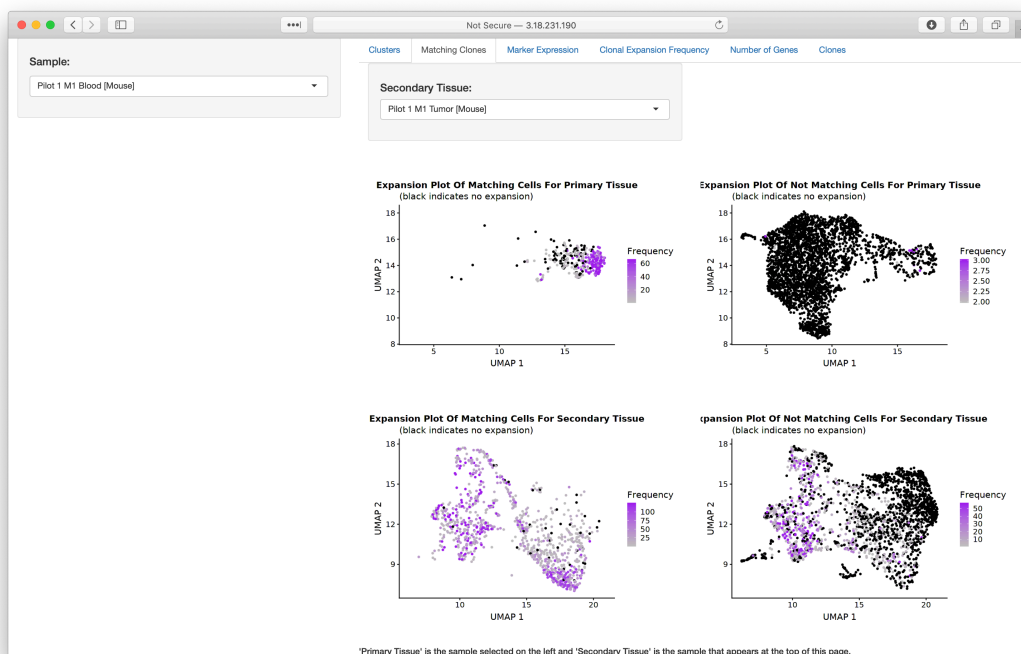


**Figure 4.12:** View of the Shiny Application showing pre-computer Louvain clustering of a sample. The view is reactive and the plots change according to the sample selection toolbar on the left.

on the backend server hosted by AWS using their EC2 product.

#### 4.6 APPLICATIONS OF THE SHINY APPLICATION

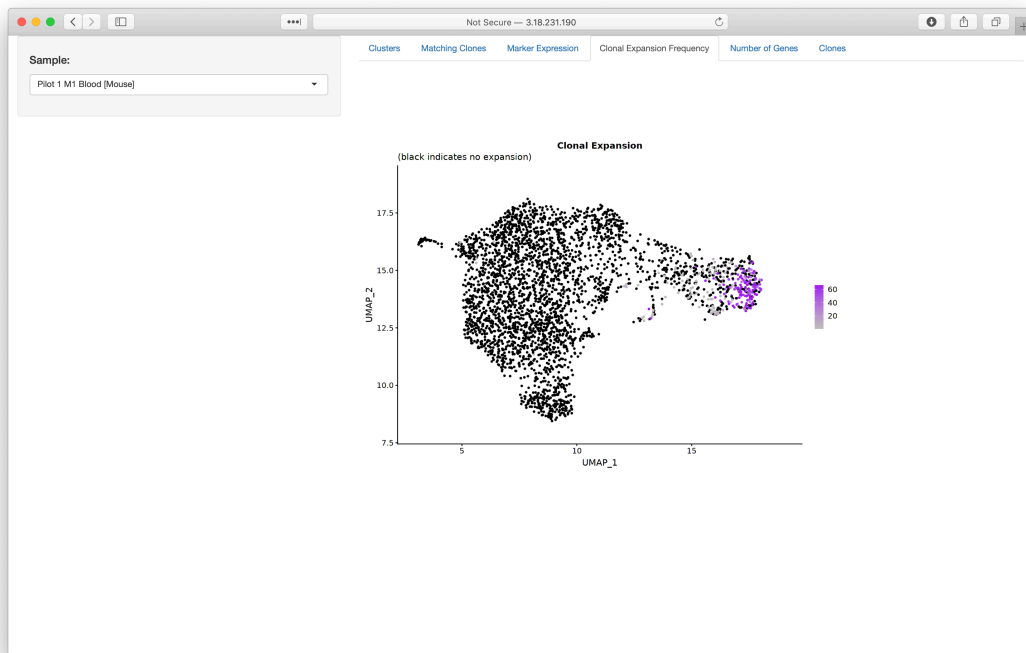
The Shiny App was utilized on human paired blood and tumor data for a project that I contributed to as a middle author showing how Layilin augments integrin activation to promote antitumor immunity (Mahuron et al., 2020). The functionality of the Shiny application (access code in Appendix D) to view marker expression across tissues (Figure 4.18) and T cell clonality (Figure 4.19) were both utilized. Additionally, the T cell clone processing pipeline output (access python code in Appendix D) was used to generate coxcomb plots showing information about the TCR repertoire (Figure



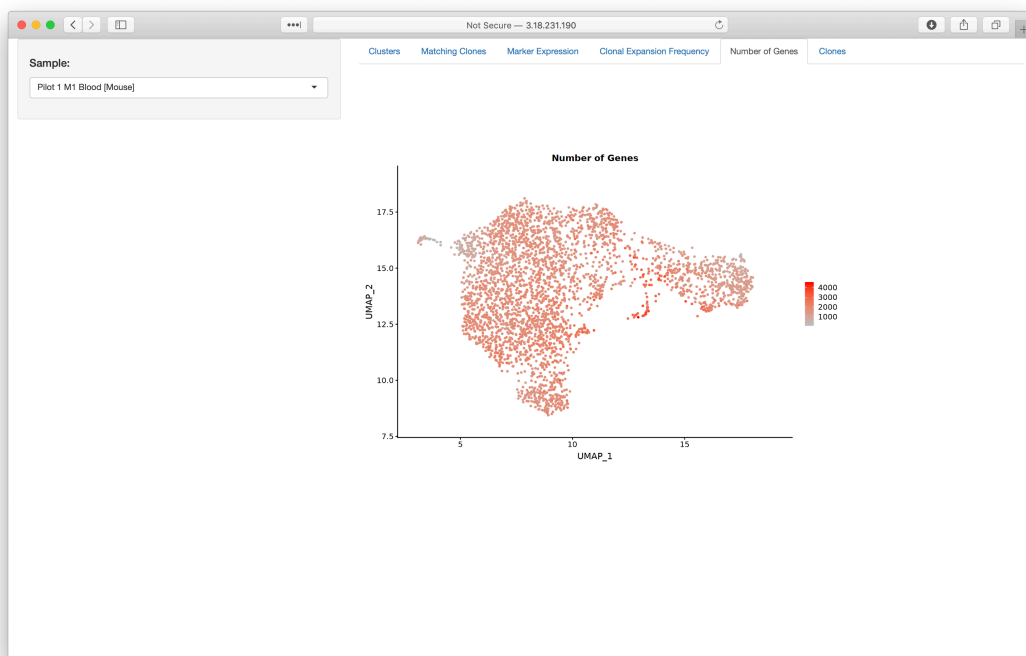
**Figure 4.13:** View of the Shiny Application showing matching T cells (defined as having a perfect match between all called alpha and beta chains while also having at least one alpha chain and at least one beta chain) between two tissues. On the left if a selector for choosing the primary tissue; after this is selected a “secondary tissue” selector is reactively generated (there is a selector in case there are more than 2 paired samples; if there is only a single secondary tissue this is automatically selected and there is no option to change it). Once the primary and secondary tissue are selected, four UMAP plots are generated on the backend and reactively added to the page. The first row contains plots for the primary tissue while the second row contains plots for the secondary tissue. The first column contains plots of matching cells and the second column contains plots of not matching cells. In each plot, if a cell is not expanded at all it is colored black. If it is expanded it is colored according to a grey to purple gradient. For a given tissue, both of the two plots (matching and not matching) are plotted in the same UMAP space.



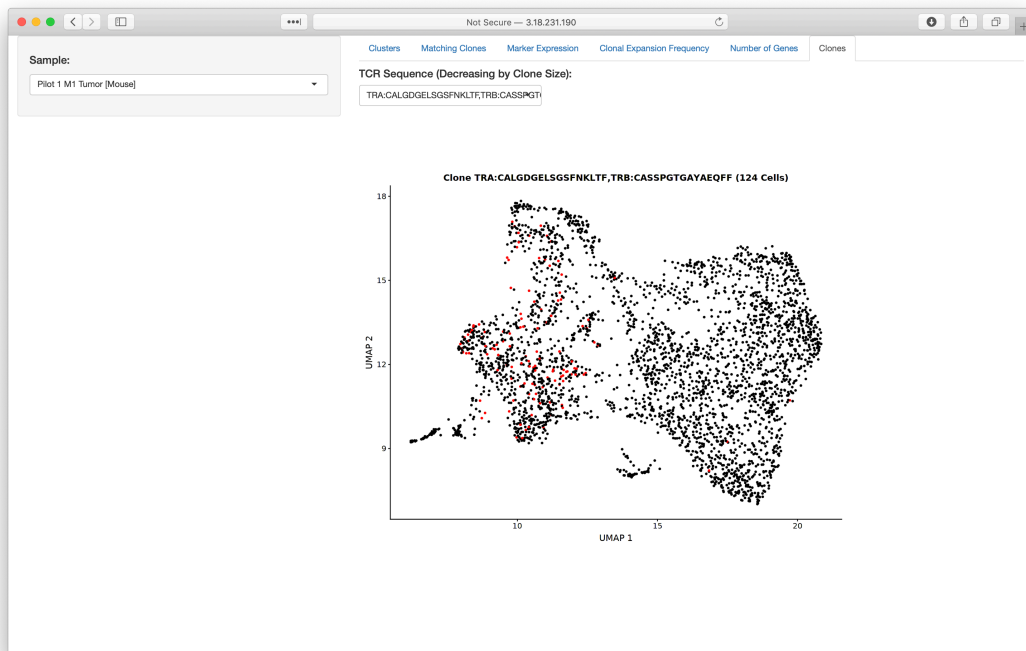
**Figure 4.14:** View of the Shiny Application showing marker expression from a tissue selected in the sample selector on the left of the screen. Gene names entered into the “Genes” text entry field will generate UMAP plots for the selected markers and automatically space multiple plots out on the page. Smart text autocompletion is also implemented for the gene entry field.



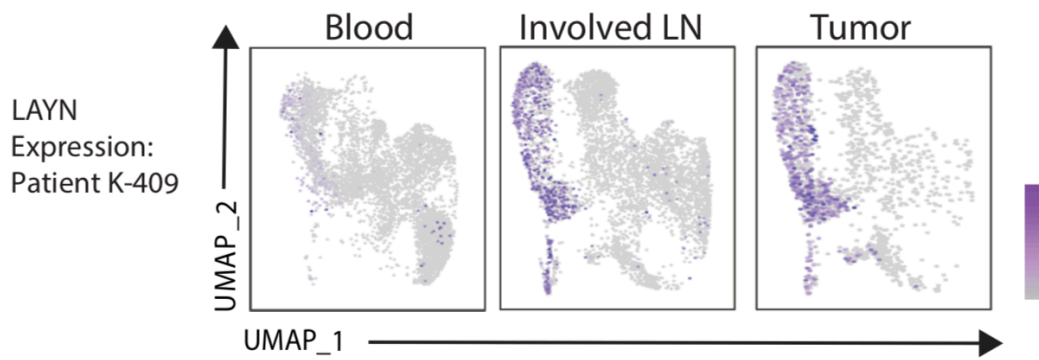
**Figure 4.15:** View of the Shiny Application showing clonal expansion from a tissue selected in the sample selector on the left of the screen. In the UMAP plot, if a cell is not expanded at all it is colored black. If it is expanded it is colored according to a grey to purple gradient. The plot generated in this menu is similar to the plots generated in Figure 4.13, with the main difference that there is no segregation of the plot into matching and not matching states as only one tissue is being plotted.



**Figure 4.16:** View of the Shiny Application showing Number of genes from a tissue selected in the sample selector on the left of the screen. In the UMAP plot, number of genes are colored on a gradient from grey to red.

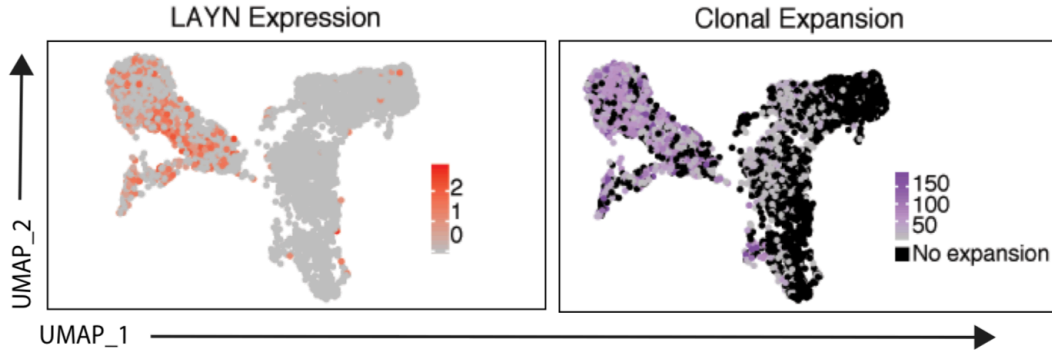


**Figure 4.17:** View of the Shiny Application showing T Cell clones from a tissue selected in the sample selector on the left of the screen. Selecting a TCR sequence from the dropdown menu (ordered according to descending clone size) will reactively generate a UMAP plot where cells in the clone are red and cells that are not in the clone are black. Clones are assigned based on perfect TCR matching in both alpha and beta chains with the requirement that a matching cell have at least one alpha chain and at least one beta chain.

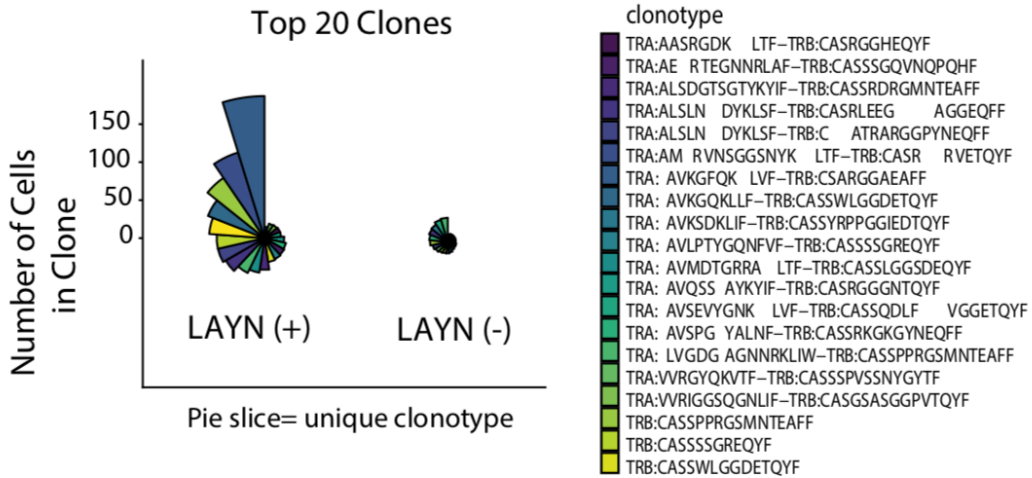


**Figure 4.18:** scRNA-seq analysis of LAYN expression in peripheral blood, metastatic LNs (involved LN), and primary tumor from a human patient.

4.20).



**Figure 4.19:** UMAP plots generated from scRNA-seq and scTCR-seq demonstrating LAYN expression and clone size from a human patient involved LN. Clones are defined as sets of cells with perfect matches for all called TCR  $\alpha$  and  $\beta$  chains from single-cell TCR data.



**Figure 4.20:** Coxcomb plots showing the 20 most expanded LAYN<sup>+</sup> and LAYN<sup>-</sup> clones in a human patient involved LN. Each pie slice represents a unique CD8<sup>+</sup> T cell clonotype, and pie slice height is proportional to clone size.



# 5

## Conclusion

EACH OF THE PROJECTS THAT I HAVE WORKED ON HAS ADVANCED OR PROVIDES POTENTIAL TO ADVANCE THE FIELD into which is was introduced. However, there were many failures that led to the successful projects in my dissertation. In this conclusion, I will briefly go over two failed projects that enabled the portions of my PhD that were published, summarize the work covered by

my dissertation, and outline some potential future directions.

## 5.1 FAILED PROJECTS

Many of my projects have failed and will never be included in the scientific record. It is still important that why they failed is documented.

### 5.1.1 DETECTING HGT IN BACTERIA FROM CHIMERIC LONG READS

#### OVERVIEW OF IDEA

The general idea for this failed project is quite simple: circa 2016 when long read sequencing from Oxford Nanopore (ONT) was starting to have actually usable base calling accuracy and the ability to generate quality microbial contigs (Figure 5.1), it seemed promising to try to infer potential Horizontal Gene Transfer (HGT) events between microbes by sequencing new samples. The basic idea here is that at some point in the past, a HGT event occurred between two microbes which were then sequenced as isolates and deposited into a database such as the ones provided by NCBI. With traditional short read sequencing, there is large probability that genetic material transferred from one microbe to another is not included in assembly scaffolds generated from short reads. However, using long read sequencing of environmental or gut samples could yield reads that have good mappings to more than one isolate reference generated from scaffolds utilizing short Illumina reads (Figure 5.2). Subsequently, potential HGT sites in reference isolate genomes could be algorithmically “imputed” (Figure 5.3).

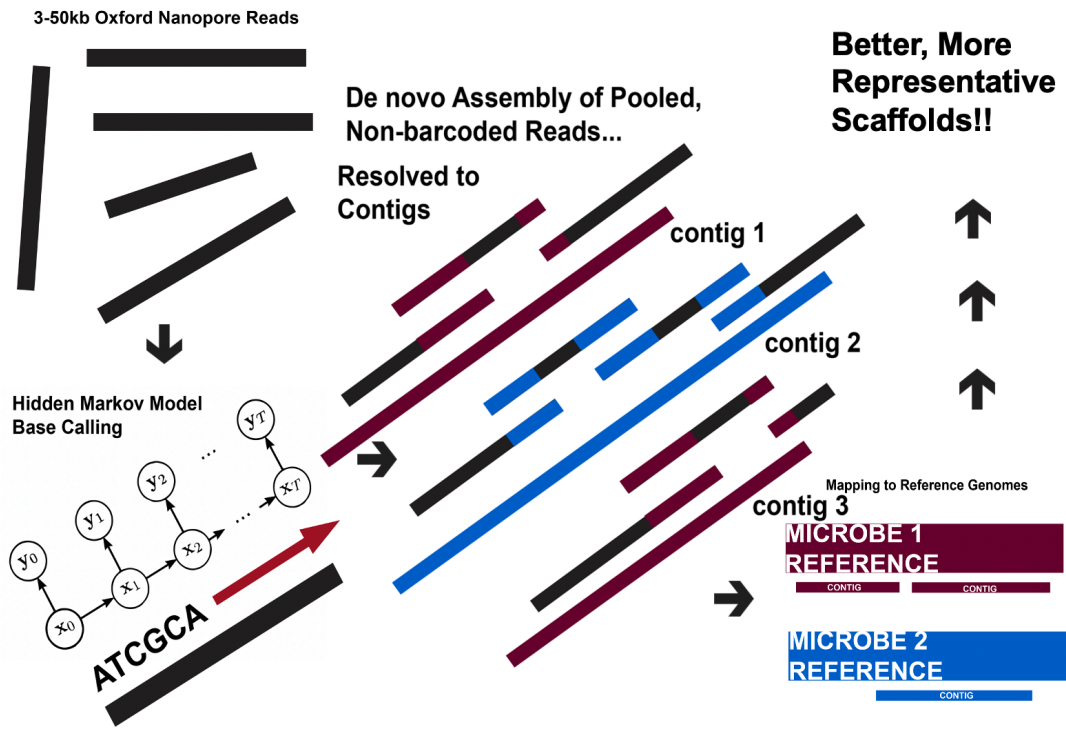
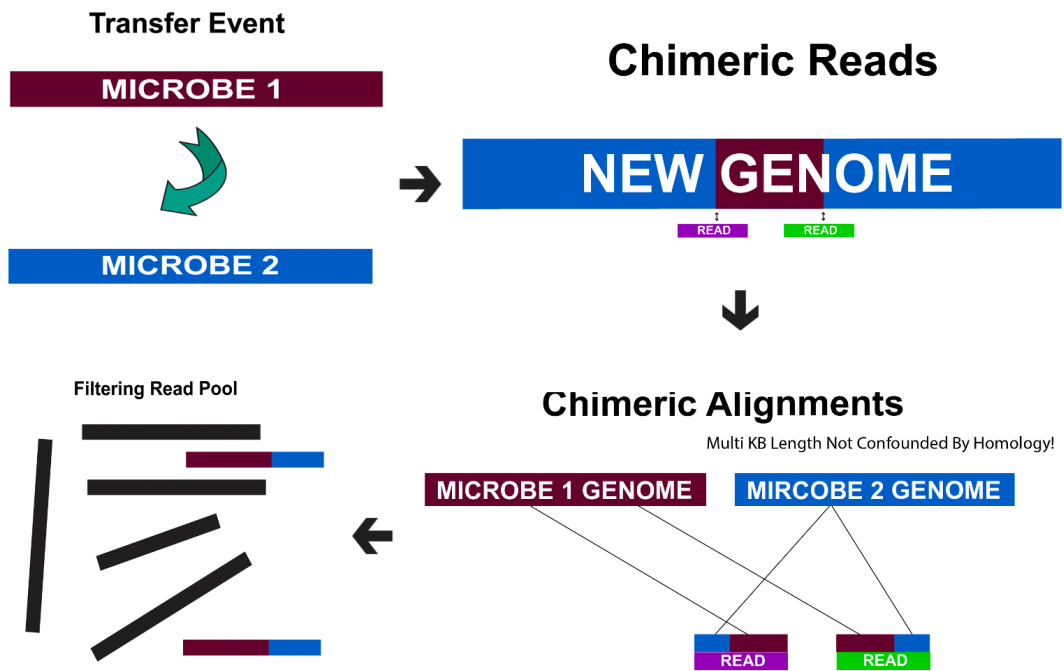
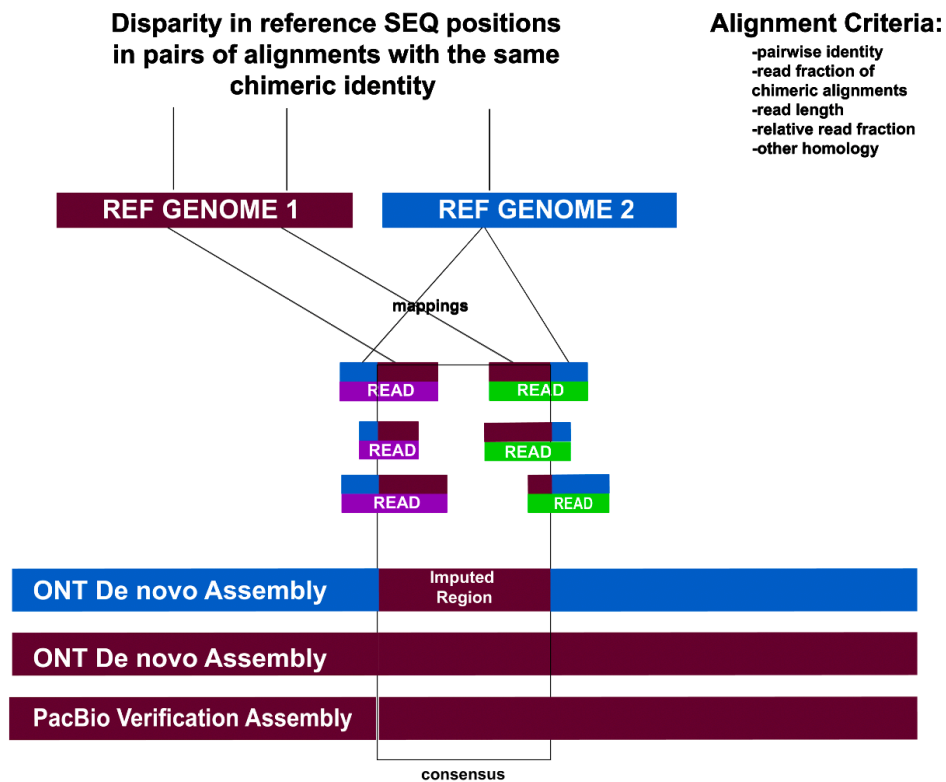


Figure 5.1: Illustration of how scaffolds for microbial assemblies could be improved through the utilization of long reads.



**Figure 5.2:** Illustration of how potential Horizontal Gene Transfer (HGT) events can be potentially identified by finding “chimeric” reads that contain good mappings to multiple reference genomes in publically available databases.

## Extracting Information From Chimeric Reads



**Figure 5.3:** Illustration of how potential HGT events could be “imputed” into existing reference genomes from information contained in “chimeric” long reads have many-fold genome coverage in

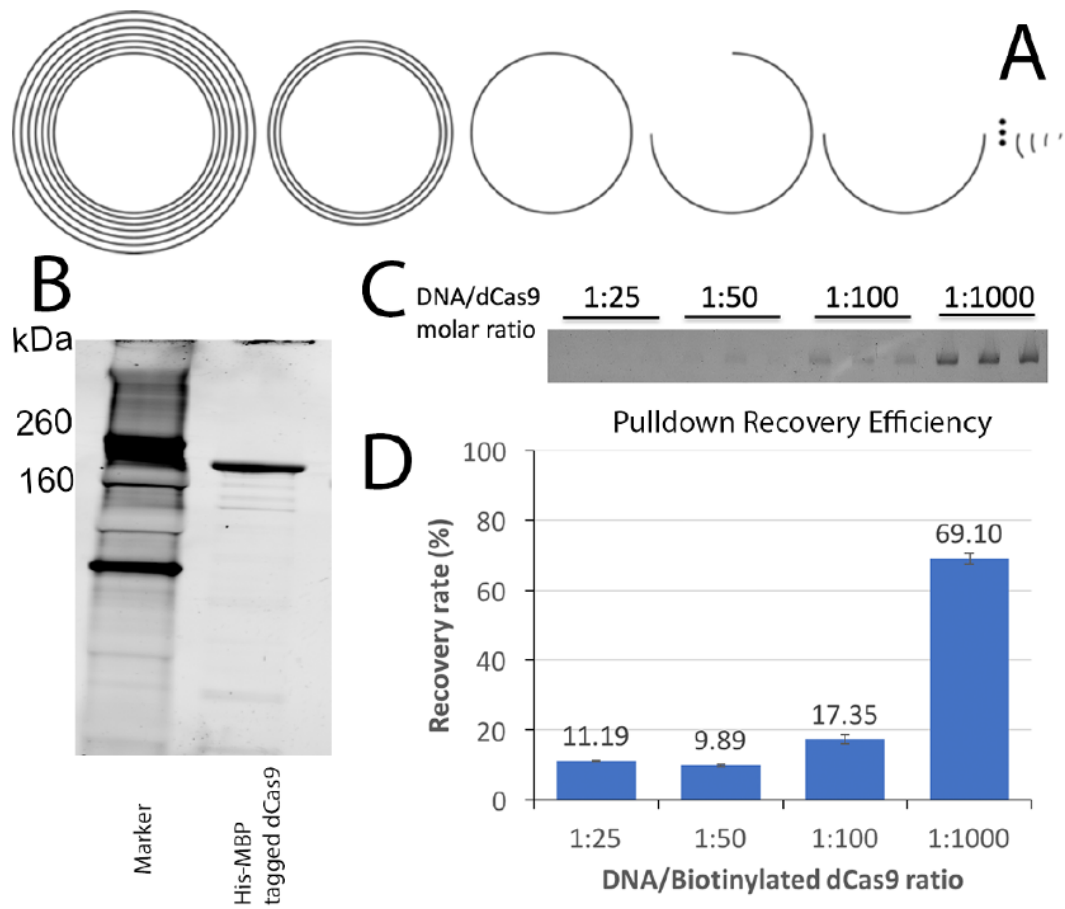
## FAILURES

After implementing a scraper to download many isolate genomes, it became apparent that the scale of this problem is quite challenging in the computational, which in part motivated some of the work in Chapter 2.

### 5.1.2 dCAS9 COMPLETE METAGENOMES

#### BACKGROUND

Low abundance microbes play a crucial role in microbial communities (Kurm et al., 2017; Dawson et al., 2017; Hausmann et al., 2016). However, current metagenomic sequencing techniques do not account for low abundance microbes as the random shotgun method recovers many-fold genome coverage of the most abundant constituents, but only a few reads from the lowest abundance constituents (see illustration in Figure 5.4). It is past time for the field of metagenomics to move beyond marker gene and 16S estimates of taxonomic abundance and develop new methods that can accurately recover the full genomes of low abundance microbes. Without such genome recovery, functional elucidation of the mechanisms underpinning how communities of microbes interact with themselves and their environment will be difficult to achieve. Developing a sequencing system that is comprised of the coupling of a gRNA selection algorithm and an engineered biotinylated catalytically-dead Cas9 (bdCas9) pulldown with the goal of recovering the currently unrecoverable genomes of low abundance microbes could potentially recover low abundance microbes.



**Figure 5.4:** a High abundance microbes have many-fold genome coverage in shotgun sequencing; low abundance organisms are hardly sampled at all. Preliminary data showing efficiency of engineered dCas9 based DNA pull-down method. b Purification of His- MBP- tagged dCas9. c Gel analysis of recovered DNA from pull-down assays at varied molar ratios of pUC19 to biotinylated dCas9. d Measurement of recovery ratios by Qubit DNA assays.

## EXPERIMENTAL WORK

To show feasibility of using a bdCas9 pulldown to retrieve desired genomic DNA, Cas9 plasmids were modified with mutations at sites D10A and H840A to disrupt the catalytic residues of both HNH and RuvC-like domains and create catalytically dead (dCas9) that has no endonuclease ability.\* Subsequently, His-MBP tagged dCas9 was purified via a resin filter (Figure 5.4 B) and a biotin bead was affixed to create bdCas9. At varying molar ratios, bdCas9 and pUC19 plasmids were combined with a constant amount of gRNA with target regions matching pUC19; bdCas9 were subsequently pulled down with a Streptavidin coated bead and bdCas9 was separated from bound pUC19 with a phenol-chloroform solution (Figure 5.4 C). Qubit assays were performed both before introduction of bdCas9 and after the post pulldown separation for varying molar ratios to measure DNA recovery efficiency (Figure 5.4 D). These results show that with a sufficient concentration of bdCas9, pulldown of targeted genomic DNA fragments large enough (pUC19 is 2686bp) to theoretically amplify and assemble circularized low abundance microbial genomes is feasible.

## COMPUTATIONAL GOALS

The computational goal of this project was to develop an algorithm to effectively identify unique genomic regions in low abundance microbes. An issue with any Cas9 based targeted binding is the selection of gRNAs whose 20bp target regions are unique to the desired genomic DNA binding site so as to minimize off-targets. The combinatorial complexity of avoiding off targets is not compu-

---

\*I worked on this project jointly with an experimental postdoc, Tao Xu



tationally scalable beyond 10Mbp of genomic material (~1 large genome) (Spoto et al., 2017). Due to properties of de bruijn graphs, when metagenomes are de novo assembled from one sample containing many microbes, reads that do not incorporate into the consensus contig are less likely to have enough homology to potentially cause an off-target effect and more likely to be part of a low abundance microbe. Thus, given reads from a metagenomic sample, my proposed approach to select gRNA target regions corresponding to low abundance microbial genomes is to perform de novo metagenome assembly, then select the set of reads that were not used. Using references databases and trained Hidden Markov Models, putative non-microbial reads (archaea, phage, eukaryotes, etc.) could be filtered out. The proposed algorithm would then select the subset of these reads that putatively minimize off-target effects. These sequences could then be utilized to create gRNAs that target low abundance microbes for any Cas9 based targeting system.

## FAILURES

While this system worked great on the pUC19 plasmid trials, pilots with minimum communities of microbes abjectly failed (Wymore Brand et al., 2015). This was likely either due to problems in the synthetic biology platform, or the algorithm not properly accounting for complex homology in microbial genomes. More information about followup projects that try to decipher and quantify the complexity of the genetic landscape of the microbiome are covered in Chapter 2.

## 5.2 INFLUENCE OF DISSERTATION PROJECTS ON THE FIELD

### 5.2.1 AETHER

Shortly after Aether was published, Amazon introduced a nearly identical algorithm on their platform. This is interesting as Amazon essentially serves as a market maker offering an illiquid asset (compute servers): they control the prices but must compete with other companies offering a similar product (Google, Microsoft, etc.). Anecdotally, shortly after *Aether* was published, there were quite substantial market distortions on the secondary cloud computing marketplace, which clearly was not in Amazon's interest. However, after AWS implemented their own version of our open source algorithm, the playing field was more level: every AWS user was now able to access predictions about whether their spot instances would terminate.

This feature release by Amazon certainly decreased the utility of Aether in obtaining extremely cheap cloud computing for genomic assemblies. In a way, Aether was taking advantage of a zero-sum game for people who used it. However, the feature introductions that Aether spurred played a role in leveling the playing field for everyone: users at small Universities without their own clusters can now attempt to utilize cloud computing without paying exorbitant prices.

### 5.2.2 THE ATHLETE MICROBIOME

The athlete microbiome paper has had significant influence on both debates about policy and sporting; it has also influenced the microbiome field. Our publication of this paper spurred widespread discussion on social media and speculation that sporting could potentially be ruined forever by hav-

ing doping be possible through a microbe that may or may not be naturally present in the guts of elite athletes. Does the microbe engraft into the GI tracts of elite athletes because of higher lactate levels or do athletes become elite because of coincidental engraftment of the microbe? This question has not yet been answered.

In the context of contributions to the microbiome field, our findings related to the discovery of *Veillonella* as a performance enhancing microbe provide a strong argument for the potential value of moving away from easy to use and off the shelf reference based metagenomics tools. Rather, conducting more challenging analyses requiring *de novo* assembly of sequencing samples such as in our study would likely benefit many other studies. Critics of non reference based approaches argue that after assembly and gene catalog creation occurs, the vast majority of genes detected have no functional annotation and no route to this without involved experimentation (Quince et al., 2017).

However, while it is true that the vast majority of genes in the catalog generated for the athlete microbiome paper had no functional annotation (approximately 2 out of 2.3 million), we were still able to discover unannotated function by inferring where poor annotations had occurred in reference databases, which led to hypothesis generation about novel microbial enzymatic chemistry and interesting metabolism. These hypotheses were subsequently experimentally validated. Leveraging our gene catalog allowed us to look at the microbiome as a system, and then improve our understanding of how it worked in the context of the human host. Such an approach would never have worked had we solely utilized pre-existing reference based approaches, as the novel biology discovered was not present in said databases.

### 5.2.3 TCR MATCHING

As the single cell immunology chapter is still in revision at the time of submission of this dissertation, it has not yet had the ability to exert any influence on the field of single cell cancer immunology. However, once it is published the results that the field will likely find most interesting are the fact that exactly matching TCR chain sequences are present between paired tissues at a greater than expected frequency. This could have potential exciting implications in the future, as this may provide some utility for future algorithms that try to detect whether a tumor will respond or not to immunotherapy based on a peripheral blood draw.

## 5.3 CONCLUSIONS

Soon after *Aether* was published (open source under a MIT license), Amazon Web Services implemented a nearly identical predictor on their AWS platform which provided a GUI that allowed users to see predictions about whether a spot instance that they chose to bid on would be terminated or not. While certainly detracting from the number of citations received, *Aether* laid the foundation for many scientists working with large scale data to be able to access cheaper and more efficient compute.

The athlete microbiome project received substantial media attention almost immediately after it was published in *ArsTechnica*, *Forbes*, *The New York Times*, *The Los Angeles Times*, *CNN*, *NPR Science Friday*, *Der Spiegel*, *Scientific American*, *PBS*, and *The Guardian* because the idea of a probiotic supplement that could yield to undetectable doping is an idea that permeates instantly into the

sports obsessed American conscious. The power of this project was showing how powerful coupling experimentation and computational approaches can be: the whole is greater than the sum of the individual parts.

The machine learning approaches for reasoning about T cells in the context of cancer immunotherapy are perhaps the most important contributions that this dissertation makes, as they lay the very initial ground work for liquid biopsies that could predict effectiveness of cancer immunotherapy treatments far in the future.

#### 5.4 FUTURE DIRECTIONS

A general theme of my dissertation work has been building computational tools to enable the handling of complexity in biological problems at a scale not previously possible. Advances in spatial transcriptomics and multiplexed imaging make relevant to the field of computational biology massive scale imaging data with greater complexity than traditional sequencing data; this area greatly needs development of computational tooling over the next few years.



# *Aether*: Scaling Cloud Compute

## A.1 DATA AVAILABILITY

Data utilized are available at <https://pubs.broadinstitute.org/diabimmune> and with EBI SRA accession ERP005989. Source code is available at (<https://github.com/kosticlab/aether>). Examples, documentation and a tutorial are available at <http://aether.kosticlab.org>.

## A.2 TUTORIAL

We have provided a tutorial to help users with migrating their computational workflows to *Aether*.

The tutorial can be accessed here: <http://aether.kosticlab.org/tutorials/>. Additionally, the tutorial page provides information about using *Aether* across multiple cloud compute service providers or with your own hardware.

## A.3 COMMAND LINE OPTIONS

Usage: `\textit{aether}` [OPTIONS]

The `\textit{Aether}` Command Line Interface

Options:

<code>-I, --interactive</code>	Enables interactive mode.
<code>--dry_run</code>	Runs <code>\textit{Aether}</code> in dry-run mode. This shows what cloud computing resources <code>\textit{Aether}</code> would use, but does not actually use them or perform any computation.
<code>--ilp</code>	Runs the LP algorithm as a dry run with the CPLEX solver instead of the default solver
<code>-A, --input-file TEXT</code>	The name of a text file, wherein each line

corresponds to an argument passed to one of the distributed batch jobs.

- L, --provisioning-file TEXT   Filename of the provisioning file.
- P, --processors TEXT        The number of cores that each batch job requires
- M, --memory TEXT            The amount of memory, in Gigabytes, that each batch job will require.
- N, --name TEXT               The name of the project. This should be unique, as an S3 bucket is created on Amazon for this project, and they must have unique names.
- E, --key-ID TEXT             Cloud CLI Access Key ID.
- K, --key TEXT                Cloud CLI Access Key.
- R, --region TEXT             The region/datacenter that the pipeline should be run in (e.g. "us-east-1").
- B, --bin-dir TEXT            The directory with applications runnable on the cloud image that are dependencies for your batch jobs. Paths in your scripts must be reachable from the top level of this directory.
- S, --script TEXT             The script to be run for every line in input-



	file and distributed across the cluster.
-D, --data TEXT	The directory of any data that the job script will need to access.
--help	Show this message and exit.

### A.3.1 STEPS NEEDED TO RUN AWS BATCH

1. An AWS Batch user one must first manually set up IAM user permissions, which is a 12 step process.
2. Subsequently, an AWS Batch user needs to create IAM roles for these IAM users, which Amazon's most detailed batch instructions do not provide clear instructions for.
3. Subsequently, an AWS Batch user then needs to follow an up to 13 step process to manually create a key pair for the AWS Batch backend to authenticate to provisioned resources that the batch process job will end up using.
4. Subsequently, an AWS Batch user must manually change non-default networking settings to create a VPC (a closed virtual network) where the provisioned AWS Batch jobs will run. This is a 5 step manual process.
5. Subsequently, an AWS Batch user must manually create a security group to allow correct port access to provisioned resources that will be utilized. This is a manual 6 step process.
6. Subsequently, an AWS Batch user must configure their job options for each job they run. This is a manual 3 step process.
7. Subsequently, an AWS Batch user must specify the run time environment on their batch job via parameterizing a virtual container. This is a manual 3 step process. Note that this has nothing to do with how long the job runs but rather run time environment.
8. Subsequently, an AWS Batch user must specify resources that their compute job will need (*Aether* also asks for this but also asks for anticipated length of time that a job will run for). This is a manual 4 step process.

9. Subsequently, an AWS Batch user must configure their computer environment type. This is a manual 3 step process.
10. Subsequently, an AWS Batch user must configure instances. This is a manual 6 step process.
11. Subsequently, an AWS Batch user must configure networking for the instances they configured in step 10. This is a manual 3 step process.
12. Subsequently, an AWS Batch user must tag their networked instances. This is a manual 3 step process.
13. Finally, an AWS Batch user must manually create their job queue.

#### A.4 FUNDING

This work was funded by National Institutes of Health/National Human Genome Research Institute (NIH/NHGRI) T32 HG002295, PI: Park, Peter J (J.M.L.); an AWS Research Credits for Education Grant (J.M.L. and A.D.K.); a Microsoft Azure for Research Grant (B.T.T. and C.J.P.), NIH National Institute of Environmental Health Sciences (NIEHS) R00 ES023504 (C.J.P.); NIEHS R21 ES025052 (C.J.P.); National Science Foundation (NSF) Big Data Spoke grant (C.J.P.), a Smith Family Foundation Award for Excellence in Biomedical Research (A.D.K.); and an American Database Association (ADA) Pathway to Stop Diabetes Initiator Award (A.D.K.).

# B

## Athlete Microbiome Project

### B.1 WET LAB METHODS

Below are methods related to work done by my two experimental co-first authors on this project, Ted Chavkin and Jonathan Scheiman. Computational methods that I developed are located in the main chapter.

### B.1.1 PARTICIPATION RECRUITMENT

All study participants were recruited following a Sports Genomics protocol (number IRB15-0869) approved by the Institutional Review Board of the Wyss Institute for Biologically Inspired Engineering. Each participant read and signed a consent form before study enrollment, and we have complied with all of the relevant ethical regulations.

### B.1.2 SAMPLE COLLECTION, EXTRACTION AND LIBRARY PREPARATION

For the collection of materials, study participants were provided with a 15 ml falcon tube with a 1 ml pipette tip inserted inside. Participants were instructed to dip the pipette tips into soiled toilet tissue, then place them back into the tubes and label the tubes with the date and time of collection. Samples were kept at 4°C for short-term storage until sample pickup, at which point they were immediately placed onto dry ice, then transferred to a -80°C freezer for long-term storage.

Fecal samples were thawed on ice and resuspended in 2–5 ml of PBS, 250 µl of which was used for DNA extraction using the Mo Bio PowerSoil high-throughput DNA extraction kit, following the manufacturer's protocol. For 16S rDNA library construction, 1–5 µl of purified DNA was used for PCR amplification of the V<sub>4</sub> variable region using Q5 Hot Start Polymerase (NEB). Primers were adapted from the Earth Microbiome Project (<http://www.earthmicrobiome.org/>), attaching Illumina paired-end adapters (forward: CTT TCC CTA CAC GAC GCT CTT CCG ATC TGT GCC AGC MGC CGC GGT AA; reverse: GGA GTT CAG ACG TGT GCT CTT CCG ATC TGG ACT ACH VGG GTW TCT AAT). Illumina barcodes were added to libraries during

a second PCR step (forward: AAT GAT ACG GCG ACC ACC GAG ATC TAC ACT CTT TCC CTA CAC GAC GCT C; reverse: CAA GCA GAA GAC GGC ATA CGA GAT GTG ACT GGA GTT CAG ACG TGT GCT C) and end products were purified via column chromatography (Zymo Research). Individual libraries were quantified and normalized for sequencing using the Quant-iT PicoGreen reagent (Thermo Fisher Scientific). For whole-genome shotgun library construction, 1 ng of purified DNA was used for Illumina's Nextera XT Tagmentation kit, following the manufacturer's protocol. Libraries were submitted to the Harvard Biopolymers core sequencing facility for bioanalyzer quality control and 150-base pair paired-end sequencing reads using either the Illumina MiSeq or HiSeq 2500 system (high output mode) for 16S rDNA and shotgun analysis, respectively.

### B.1.3 METADATA COLLECTION

Each study participant was provided with a questionnaire to collect health, dietary and athletic background information (adapted from The American Gut; <http://americangut.org/>). Additionally, for each sample collection, study participants filled out a daily annotation sheet to collect dietary, exercise and sleep information.

### B.1.4 PREPARATION OF BACTERIA FOR GAVAGE

*V. atypica* and *L. bulgaricus* were grown in 250 ml BHIL (10 ml of 60% sodium lactate per liter) and MRS broth, respectively. The optical density measured at a wavelength of 600 nm (OD<sub>600</sub>) was monitored and at an optical density of 0.4–0.6, cells were pelleted by refrigerated centrifugation at 5,000g for 10 min. The pellet was washed in PBS and resuspended in 2 ml residual PBS. Aliquots of

100  $\mu$ l were frozen at  $-80^{\circ}\text{C}$  and the numbers of colony-forming units (c.f.u.) per ml were measured by serial dilution onto BHIL agar plates. *V. atypica* was gavaged in wild-type C57BL/6 mice to determine viability and transit time through the gastrointestinal tract, observing peak viable bacterial c.f.u. counts in fecal pellets 5 h after gavage.

#### B.1.5 TREADMILL CROSSOVER EXPERIMENT

Animal research was approved by the Joslin Diabetes Center Institutional Animal Care and Use Committee and we complied with all of the relevant ethical regulations. For the treadmill experiments, 8- to 12-week-old CL57BL/6 mice ( $n = 32$ ) were acclimated to treadmill with two bouts of 30 min of 5  $\text{m min}^{-1}$  walking, split over two consecutive days. For exhaustion measurements, mice were fasted for 7 h before exercise. Then, 6 h before exercise, mice were gavaged with 200  $\mu$ l of 2.5% sodium bicarbonate to neutralize the stomach contents, and 20 min after the first gavage, mice were gavaged 200  $\mu$ l of either *V. atypica* or *L. bulgaricus*, prepared as above and normalized to 5109 c.f.u.  $\text{ml}^{-1}$ . Next, 5 hours postgavage, mice were run on the treadmill, starting at 5  $\text{m min}^{-1}$  and increasing the speed by 1  $\text{m min}^{-1}$  every minute until exhaustion. The time of exhaustion was recorded for every animal, defined as a mouse failing to return to the treadmill from the rest platform after three consecutive attempts to continue running. This protocol was repeated for two more days, followed by 4 d of rest and 3 d of crossover treatment. On the first day of treatment, serum was collected 40 min post-exhaustion via a tail vein bleed and measured via Ciraplex multiplex mouse cytokine assay (Aushon Biosystems).

## B.1.6 *IN VITRO* GROWTH AND SCFA ANALYSIS

*Veillonella* species (*V. dispar*, *V. parvula* and *V. atypica*) were isolated and purified from several study participants and grown in three different media compositions: (1) BHIL (10 ml of 60% sodium lactate per liter); (2) MRS broth (BD) supplemented with lactate (10 ml of 60% sodium lactate per liter); and (3) semi-synthetic lactate medium (per liter: 5g bacto yeast extract, 0.75g sodium thioglycolate, 25 ml basic fuchsin and 21 ml 60% sodium lactate (pH 7.5)). *Veillonella* species were inoculated into each medium, under anaerobic conditions, and allowed to grow for 48 hours to reach the stationary phase. After 48 hours, bacteria were pelleted and supernatants were collected for lactate and SCFA measurements. Approximately 10  $\mu$ l of supernatant was used to measure the lactate via the Lactate Scout ([lactate.com](http://lactate.com)). The remaining supernatants were frozen at  $-80^{\circ}\text{C}$ , then submitted to the Harvard Small Molecule Mass Spectrometry core facility for butyrate, propionate and acetate quantitative analysis.

SCFAs identified from the mass spectrometry in all three media conditions corresponded with the propionate end product suggested by the metagenomic results. Acetate was not observed in MRS or BHIL, likely due to high existing concentrations in the media making the forward reaction thermodynamically unfavorable. However, acetate production was observed in semi-synthetic lactate media (access Supplementary Table 7 from Appendix B).

### B.1.7 $^{13}\text{C}_3$ -LACTATE FLUX TRACING

Ten-week-old C57BL/6 mice were treated with sodium bicarbonate followed by 109 c.f.u. of either *V. atypica* (n = 4) or *L. bulgaricus* (n = 4), prepared as above. Then, 20% w/w  $^{13}\text{C}_3$  sodium lactate (Cambridge Isotope Laboratories) was diluted to a concentration of 400 mM in PBS. Mice were injected with 100  $\mu\text{l}$  intravenously via the tail vein and, after 9 minutes, anesthetized with isoflurane. One mouse treated with *V. atypica* was unable to be injected due to vein clamping and had to be removed. Next, 10 min post-injection, anesthesia was confirmed via foot pinch and mice were sacrificed via cardiac puncture. Whole blood was divided into two samples to obtain both serum and plasma. These were flash frozen in liquid nitrogen at 12 minutes post-injection and stored at  $-80^\circ\text{C}$ .

Immediately following cardiac puncture, mice were dissected to remove the colon and cecum, and the contents were removed by squeezing with sterilized forceps into preweighed tubes. The contents were immediately flash frozen in liquid nitrogen. The timing varied slightly, but this was done between 17 and 19 minutes post-injection.

Samples were analyzed for lactate and propionate by the Broad Institute Metabolomics Platform. LC-MS metabolomics were performed as previously described (Fujisaka et al., 2018). LC-MS traces were identified and integrated to quantify the presence of  $^{13}\text{C}_0$ - and  $^{13}\text{C}_3$ -lactate isotopes.

### B.1.8 COLORECTAL PROPIONATE INSTILLATION

Treadmilling followed the same protocol as above. Mice were fasted 7 hours before exercise to normalize their metabolic profiles. Some 30 minute before exercise, mice were treated with 200  $\mu\text{l}$  of



either PBS vehicle alone (n = 8) or 150 mM sodium propionate in PBS (n = 8), using a flexible gavage needle to introduce 200  $\mu$ l of solution into the colon. Mice were then run to exhaustion as above. This protocol was repeated for three consecutive days. On the first day of treatment, serum was collected 40 minutes post-exhaustion via tail vein bleed and measured using the Ciraplex multiplex mouse cytokine assay (Aushon Biosystems).

### B.1.9 LACTATE CLEARANCE

To measure the lactate clearance rate, mice were first fasted for 7 hours before measurement to stabilize the basal lactate levels. Then, 5 hours before measurement, mice were treated with sodium bicarbonate followed by 109 c.f.u. of either *V. atypica* or *L. bulgaricus*, prepared as above (n = 8). Next, 30 minutes before measurement, mice were weighed and individually caged, and a baseline blood lactate reading was taken using a Lactate Scout meter. Mice were administered sodium lactate via intraperitoneal injection with a dose of 750 mg/kg<sup>t</sup>, prepared as a 75 mg/ml<sup>t</sup> solution of sodium lactate in pH 7.0 PBS. Blood lactate levels were monitored with a Lactate Scout meter at 5, 15, 25, 35 and 45 minutes post-injection

### B.2 DATA AVAILABILITY

All raw sequencing data have been uploaded to NCBI and SRA in the form of the BioProjects PRJNA472785 (16S) (<http://www.ncbi.nlm.nih.gov/sra?term=PRJNA472785>) and PRJNA472768 (MGX) (<http://www.ncbi.nlm.nih.gov/sra?term=PRJNA472768>). These are linked to associated BioSamples, which in turn are linked to the paired-end read files in the SRA,

and correspond to the metadata in the Supplementary Information (<https://www.nature.com/articles/s41591-019-0485-4#MOESM2>) files.

### B.3 SUPPLEMENTARY TABLES

All supplementary tables are available at [https://static-content.springer.com/esm/art%3A10.1038%2Fs41591-019-0485-4/MediaObjects/41591\\_2019\\_485\\_MOESM2\\_ESM.xlsx](https://static-content.springer.com/esm/art%3A10.1038%2Fs41591-019-0485-4/MediaObjects/41591_2019_485_MOESM2_ESM.xlsx).

### B.4 CODE AVAILABILITY

Unless otherwise noted, all plots were generated in R version 3.4.1 with the ggplot2, dplyr, scales, grid and reshape2 packages (Wickham & Chang, 2015; Wickham & Francois, 2015; Murrell, 2002; Wickham, 2012). Large-scale data analysis was done on AWS, utilizing machines running Ubuntu 16.04. Data curation methods were coded in python version 2.7.12. The Aether package utilized for analysis is available at <https://github.com/kosticlab/aether>.

### B.5 UNPROCESSED WESTERN BLOT

Recent work by scientists doing a huge amount of unappreciated service to the greater scientific community such as Elizabeth Bik have discovered how widespread gel manipulation is in biomedical research (Bik et al., 2016). It should be the standard for authors to include unprocessed Western gels with their work to combat this. This is provided in Figure B.1.

Unprocessed Western Blot for Extended Data 7



Figure B.1: Unprocessed Western Blot for Figure 3.13

## B.6 FUNDING

This work was funded by the Synthetic Biology platform at the Wyss Institute for Biologically Inspired Engineering at Harvard University; National Institutes of Health (NIH)/National Human Genome Research Institute grant T32 HG002295 (to J.M.L.; principal investigator: P. J. Park); NIH/National Institute of Diabetes and Digestive and Kidney Diseases grant T32 DK007260 (to T.A.C.; principal investigator: T. K. Blackwell); a National Science Foundation Graduate Research Fellowship Program fellowship (to J.M.L.); National Library of Medicine BIRT grant T15LM007092 (to B.T.T.); an AWS Research Credits for Education Grant (to J.M.L. and A.D.K.); a Smith Family Foundation Award for Excellence in Biomedical Research (to A.D.K.); an American Diabetes Association Pathway to Stop Diabetes Initiator Award (to A.D.K.) and NIH/National Institute of Diabetes and Digestive and Kidney Diseases Diabetes Research Center grant P30DK036836-30 (to J.M.L., T.A.C., T.M., B.T.T., L.-D.P., Z.Y., M.C.W., S.L. and A.D.K.; principal investigator: G. L. King). We acknowledge C. J. Patel and S. R. Stein for statistical advice, and S. Softic for assistance with tail vein injections.

## B.7 AUTHOR CONTRIBUTIONS

Of the authors on the related paper(Scheiman et al., 2019), J.S., G.M.C. and A.D.K. conceived the project. J.S. collected the athlete samples and metadata. J.M.L. performed the computational analysis and modeling, with assistance from T.A.C., M.C.W., R.C.W., B.T.T., Z.Y. and M.W.H. T.A.C. designed and performed the model organism experiments, with assistance from J.M.L., T.M., A.T., L.-

D.P., M.C.W., S.P. and S.L. J.A.-P. and C.B.C. processed the metabolomic samples. J.M.L., T.A.C. and A.D.K. wrote the manuscript. G.M.C. and A.D.K. supervised the project.



# Single Cell Cancer Immunology

## C.I WET LAB METHODS

Below are methods related to work done by my two experimental co-first authors on this project.

Computational methods that I developed are located in the main chapter.

### C.1.1 MICE AND CELL LINES

Wild type (WT) female C57BL/6 mice were purchased from the Jackson Laboratory (stock number 000664). Tumor cells were implanted into mice at 8-10 weeks of age. Mice were maintained at Harvard Medical School in specific pathogen-free facilities under standard housing, husbandry, and diet conditions in accordance with Institutional Animal Care and Use Committee (IACUC) and NIH guidelines. All experimental procedures performed were approved by the IACUC at Harvard Medical School. For tumor studies, MC38 colon adenocarcinoma cells (a gift from Dario Vignali, University of Pittsburgh School of Medicine) were used. MC38 cells were grown in DMEM supplemented with 10% FBS, 100 U penicillin, and 100  $\mu$ g streptomycin in a 37°C incubator with 5% CO<sub>2</sub>. Cells were harvested at passage 2-3 after thaw, and 2.5x10<sup>5</sup> tumor cells were injected subcutaneously into the flank of mice anesthetized with 2.5% 2,2,2-Tribromoethanol (Avertin). Tumors were measured every 2-3 days using calipers, and mice were sacrificed when tumors reached 2 cm<sup>3</sup> volume, ulceration, or a body condition of >2 in accordance with IACUC guidelines. Tumor volume was determined using the formula for the volume of an ellipsoid,  $\frac{1}{2} \times D \times d^2$ , where “D” is the major axis of the tumor and “d” is the minor axis. Tumors were harvested from mice at days 19-23 after implantation for single cell RNA sequencing experiments and flow validation experiments as indicated in the Figure Legends.

### C.1.2 LYMPHOCYTE ISOLATION FROM MOUSE TISSUES

Peripheral blood was collected from mice using the retroorbital bleeding route, and blood was collected into 4% sodium citrate (Sigma) to prevent clotting. RPMI+10% FBS was added to dilute out the anti-coagulant, and then white blood cells were separated from red blood cells using centrifugation through histopaque-1083 (Sigma). The white blood cell layer at the interface between the histopaque and remaining media was subsequently washed and subjected to staining for flow cytometry analysis or sorting for single cell RNA sequencing. Tumors were dissected and mechanically disaggregated. For flow cytometry validations, a GentleMACS (Miltenyi) was used for disaggregation, whereas for single cell RNA sequencing vertical scissors used to mince the tumors instead of the GentleMACS. The dissociated tissue was digested with Collagenase Type I (400 U/ml; Worthington Biochemical) for 20-30 minutes at 37°C. Samples were then passed through a 70  $\mu$ m filter, and lymphocytes were enriched using centrifugation through a Percoll gradient (40% and 70%). The enriched lymphocyte layer at the 40%/70% interface was subsequently washed and stained for flow cytometry or sorted for single cell RNA sequencing.

### C.1.3 FLOW CYTOMETRY AND SORTING OF MOUSE SAMPLES

Single cell suspensions were generated as described above. Suspensions were labeled with LIVE/DEAD Fixable Near-IR Cell Stain in PBS (Thermo Fisher Scientific) to exclude dead cells from downstream analyses. Cells were pre-incubated with TruStain Fc Receptor Block (anti-mouse CD16/CD32, clone 93, BioLegend), then labeled with extracellular antibodies including: CD3 (clone 145-2C11)



and CD8a (clone 53-6.7) (from BD); CD11a (clone M17/4) (from Thermo Fisher Scientific); CCR2 and NKG2I (R&D Systems); Lag3 (clone C9B7W) (from Bio-Rad); and CD45.2 (clone 104), PD-1 (clone RMPI-30), CX3CR1 (clone SA011F11), CD62L (MEL-14), CD44 (IM7), CCR5 (clone HM-CCR5), CXCR6 (clone SA051D1), CD49D (clone R1-2), CD18 (clone M18/2), CD29 (clone HMB1-1), CD48 (clone HM48-1), CD94 (clone 18d3), NKG2D (clone CX5 or C7), CD39 (clone Duha59), NKG2A (clone 16A11), NK1.1 (clone PK136), Tim-3 (clone RMT3-23), CD160 (clone 7H1), Slamf7 (clone 4G2), TIGIT (clone IG9), and NRP1 (clone 3E12) (from BioLegend). Flow cytometry labeling (without inclusion of Feature Barcoding antibodies from BioLegend) was performed in PBS supplemented with 2% FBS. For CITE-seq validation experiments, cells were labeled with TotalSeqC antibodies against CD39 (TotalSeq Co834, clone Duha59) and CX3CR1 (TotalSeq Co563, clone SA011F11) as directly conjugated antibodies, and NKG2D as a biotin/streptavidin reaction (NKG2D-biotin clone C7 paired with TotalSeq Co971-Streptavidin) (from BioLegend). Labeling with Feature Barcoding antibodies was performed in PBS supplemented with 2% BSA and 0.01% Tween. Samples were acquired on a FACSymphony (BD Biosciences) and analyzed with Flow Jo software (BD Biosciences). Flow cytometry-based sorting for single cell RNA seq was performed using a FACSARIA (BD Biosciences). Blood samples were sorted based on live, CD45.2<sup>+</sup>, CD3<sup>+</sup>, CD8α<sup>+</sup>, CD44<sup>high</sup>. Tumor samples were sorted based on live, CD45.2<sup>+</sup>, CD3<sup>+</sup>, CD8α<sup>+</sup>.

#### C.1.4 SINGLE CELL RNA SEQUENCING OF MOUSE SAMPLES

Gene expression and TCR libraries for mouse samples were generated using the Chromium Single Cell 5' Library and V(D)J Reagent Kit (10X Genomics) according to the manufacturer's recommen-

dations. For samples requiring Feature Barcoding libraries to detect TotalSeqC antibodies (from BioLegend), the Chromium Single Cell 5' Feature Barcode Library Kit (10X Genomics) was used according to the manufacturer's recommendations. Following sorting as described above, approximately 10,000 cells per sample were loaded into each channel of the Chromium Chip, and recommendations were followed assuming targeted cell recovery of 2,001-6,000 cells. Libraries were sequenced on a NextSeq sequencer (Illumina) by the DFCI Sequencing Core. Gene expression libraries and Feature Barcoding libraries were sequenced using the 26 x 8 x 91 bp parameters recommended by 10X Genomics. TCR libraries were sequenced using the 150 x 8 x 150 bp parameters recommended by 10X Genomics. Based on approximate cell numbers expected, we sequenced a minimum of 20,000 reads per cell for gene expression libraries and 5,000 reads per cell for TCR and Feature Barcoding libraries.

## C.2 FUNDING

This dissertation was supported in part by the a National Science Foundation Graduate Research Fellowship.

# D

## Code

### D.1 SELECTED CODE EXAMPLES

#### D.1.1 AETHER LINEAR PROGRAMMING LOGIC

---

```
import subprocess
import os
import json
from scipy.optimize import linprog
import sys
import pickle
import math
import numpy
#from pandas import *
```

```
#this program will minimize cost per hour of distributed compute by utilizing Linear Programming to minimize cost/hour constrained by
1: minimum total cores desired at once, 2: minimum total RAM wanted at once, 3: minimum total free ephemeral storage desired, 4:
AWS account limits, 5: variability in spot bidding price. Inherently, this considers both absolute cost of resources and risk
of being outbid on the cheapest possible resources to optimize compute utilization.
```

```
global dirr
dirr='./'.join(os.path.dirname(os.path.realpath(__file__)).split('/')[:-1]+'/')

class AWS_Instance:
    def __init__(self,procs,ram,eph,name,limit,running,running_spot,historical_max,current_spot,current_od):
        self.instance_type = name
        self.procs = procs
        self.ram = ram
        self.storage = eph
        self.limit = limit
        self.running = running
        self.running_spot = running_spot
        self.historical_max = historical_max
        self.current_od = current_od
        self.current_spot = current_spot

#params for constraints
def get_user_params():
    min_cores = int(raw_input("What is the minimum number of distributed cores required?"))
    min_ram = int(raw_input("What is the minimum amount in GB of distributed RAM required?"))
    min_free_storage = int(raw_input("What is the minimum amount in GB of free ephemeral storage required?"))
    max_cost_hour = float(raw_input("What is the max cost that you are willing to pay per hour for your virtual cluster?"))
    ram_per_job = int(raw_input("What amount of RAM is required per job?"))
    procs_per_job = int(raw_input("How many Processors are required per job?"))
    return min_cores,min_ram,min_free_storage,max_cost_hour,ram_per_job,procs_per_job

def handle_grep_non_zero_output(command):
    try:
        result = subprocess.check_output(command,shell=True)
        return result
    except subprocess.CalledProcessError as e:
        result = e.output
        return result

def define_A_matrix():
    current_time = int(subprocess.check_output("date +%s",shell=True))
    weeks_back = float(raw_input("How many weeks do you anticipate running your job for?"))
    start_time = int(current_time-(weeks_back*604800))
    eph = {}
    eph_file = open(dirr+"resources/ephemeral_store_info.csv",'r')
    for line in eph_file:
        q = line.rstrip().split(',')
        eph_value = int(q[3])
        if eph_value > 0:
            eph[q[0]] = eph_value
    eph_file.close()
    retrievable_account_limits = set()
    gl_limits_file = open(dirr+"resources/gamelift_instances.txt",'r')
    for line in gl_limits_file:
        retrievable_account_limits.add(line.rstrip())
    gl_limits_file.close()
    aws_instance_file = open(dirr+"resources/instances.csv",'r')
    aws_instances = []
    os.system("aws ec2 describe-instances > ec2_instances.json")
    os.system("aws ec2 describe-spot-instance-requests > ec2_spot_instances.json")
    datacenters_fh = open(dirr+"resources/datacenters.txt",'r')
    datacenters = []
    for lines in datacenters_fh:
        datacenters.append(lines.rstrip())
    for i in range(0,len(datacenters)):
        print(str(i+1)+" "+datacenters[i])
    datacenter_idx = int(raw_input("Please enter the integer corresponding to the +amazon datacenter in which you are in:"))
    datacenter = datacenters[datacenter_idx-1]
    os.system("gunzip -c "+dirr+"resources/odprices.gz > odprices")
    print("Please visit https://console.aws.amazon.com/ec2/v2/home?region=REGION#Limits: replacing REGION with the region in which you
    plan to run this scalable cluster in and provide the requested information that is not available in the API but critical for
    proper bidding when prompted.")
    idx = 0
    pickleq = raw_input("Would you like to use a pickle file?")
    if os.path.isfile(pickleq):
        aws_instances = pickle.load( open(pickleq,"rb"))
    else:
        for line in aws_instance_file:
            split_line = line.rstrip().split(',')
            instance_name = split_line[0]
            instance_ram_float = float(split_line[2])
            instance_procs_int = int(split_line[1])
            instance_eph_int = eph[instance_name] if eph.has_key(instance_name) else 0
            running_ec2 = int(subprocess.check_output("grep \""+instance_name+"\" ec2_instances.json | wc -l",shell=True))
            running_spot = int(subprocess.check_output("grep \""+instance_name+"\" ec2_spot_instances.json | wc -l",shell=True))
            if instance_name in retrievable_account_limits:
                os.system("aws gamelift describe-ec2-instance-limits --ec2-instance-type "+instance_name+" | jq -r '.EC2InstanceLimits
                []' > i.temp.json")
                with open("i.temp.json",'r') as jsf:
                    gamelift_api_out = json.load(jsf)
                    instance_limit_pre = int(gamelift_api_out["InstanceLimit"])
                jsf.close()
            else:
                instance_limit_pre = int(raw_input("What is your account limit for "+instance_name+" in the current region being used?"))
            instance_limit = instance_limit_pre-running_spot
            historical_price_pre = handle_grep_non_zero_output("aws ec2 describe-spot-price-history --instance-types "+instance_name+"
            --end-time "+str(current_time)+" --start-time "+str(start_time)+" --product-descriptions='Linux/UNIX' --query '")
```

```

        SpotPriceHistory[*].{az:AvailabilityZone, price:SpotPrice}' | grep 'price' | sed 's/\"price\": \"/' | sed 's/^ */'
        | sed 's/\",// | uniq | sort | tail -1")
    historical_price = float(historical_price_pre)
    current_price_pre = float(handle_grep_non_zero_output("aws ec2 describe-spot-price-history --instance-types c4.large --
start-time=$(date +%s) --product-descriptions=\"Linux/UNIX\" --query 'SpotPriceHistory[*].{az:AvailabilityZone,
price:SpotPrice}' | grep 'price' | sed 's/\"price\": \"/' | sed 's/^ */' | sed 's/\",// | uniq | sort | tail -1")
    )
    current_price=float(current_price_pre)
    print("retrieved info for: "+instance_name)
    od_string = handle_grep_non_zero_output("cat odprices | grep '"+instance_name+" | grep -v 'Reserved' | grep 'Shared' |
grep -v 'SUSE' | grep -v 'Windows' | grep 'Linux' | grep '"+datacenter+"'")
    od_price = float(od_string.split(',')[9][1:-1])
    new_instance_type = AWS_Instance(instance_procs_int,instance_ram_float,instance_eph_int,instance_name,instance_limit,
running_ec2,running_spot,historical_price,current_price,od_price)
    aws_instances.append(new_instance_type)
    pickle.dump(aws_instances, open("instances.p", "wb"))
    aws_instance_file.close()
    return aws_instances

#characteristics of compute nodes (A)
def formulate_problem(aws_instances):
    od_names = map(lambda name: name+".od",map(lambda instance_object: instance_object.instance_type, aws_instances))
    spot_names = map(lambda name: name+".spot",map(lambda instance_object: instance_object.instance_type, aws_instances))
    names = spot_names+od_names
    spot_prices = map(lambda instance_object: instance_object.current_spot, aws_instances)
    od_prices = map(lambda instance_object: instance_object.current_od, aws_instances)
    prices = spot_prices+od_prices
    procs_pre = map(lambda instance_object: instance_object.procs, aws_instances)
    procs = procs_pre+procs_pre
    gbRAM_pre = map(lambda instance_object: instance_object.ram, aws_instances)
    gbRAM = gbRAM_pre+gbRAM_pre
    freestorage_pre = map(lambda instance_object: instance_object.storage, aws_instances)
    #print freestorage_pre
    freestorage = freestorage_pre+freestorage_pre
    mc_pre = map(lambda instance_object: instance_object.historical_max, aws_instances)
    max_cost_in_previous_time_window = mc_pre+od_prices
    account_limits_pre = map(lambda instance_object: instance_object.limit, aws_instances)
    account_limits = account_limits_pre+account_limits_pre
    num_types = len(procs_pre)
    return num_types,names,prices,procs,gbRAM,freestorage,max_cost_in_previous_time_window,account_limits

#setting up LP problem formulation
def run_LP(num_types,names,prices,procs,gbRAM,freestorage,max_cost_in_previous_time_window,account_limits,min_cores,min_ram,
min_free_storage,max_cost_hour,ram_per_job,procs_per_job,aws_instances):
    c = prices
    A_i = [procs,gbRAM,freestorage]
    b_i = [min_cores,min_ram,min_free_storage]
    A = map(lambda x: map(lambda y:y*-1,x),A_i)
    b = map(lambda z: z*-1,b_i)
    for i in range(0,num_types):
        append_a = [0] * num_types
        append_a[i] = 1
        add_a = append_a+append_a
        A.append(add_a)
        b.append(account_limits[i])
    A_limits = []
    b_limits = []
    for i in range(0,num_types):
        a_arr = [0]*num_types
        a_arr[i] = 1
        new_a = a_arr+a_arr
        A_limits.append(new_a)
        b_limits.append(account_limits[i])
    cost = 0
    status = 4
    while status != 0:
        A_t = A
        b_t = b
        if cost > max_cost_hour:
            break
        cost+=1
        A_t.append(max_cost_in_previous_time_window)
        b_t.append(cost)
        #print("solving:"+str(A_t)+"*x="+str(b_t))
        bounds_input = map(lambda x: (0,x),account_limits)
        lp_output = linprog(c,A_ub=A_t,b_ub=b_t,bounds=tuple(bounds_input))
        #print("x="+str(lp_output.x)+"\n")
        #lp_output = linprog(c,A_ub=A_t,b_ub=b_t,bounds=tuple(bounds_input),options={"bland": True})
        status=lp_output.status
        A_t.pop()
        b_t.pop()
        if status == 0:
            break
    lp_output_n = linprog(c,A_ub=A,b_ub=b,bounds=tuple(bounds_input),options={"bland": True})
    return lp_output,lp_output_n

def helper_recursive(input_item,names,instances):
    print input_item.x
    zipped_output = zip(names,input_item.x)
    filtered_output = filter(lambda x: x[1] != 0, zipped_output)
    remove_these_pre = map(lambda y: y[0], filter(lambda x: float(x[1]) < 1, filtered_output))
    remove_these = map(lambda x: '.'.join(x.split('.')[2:]), remove_these_pre)
    new_instances = filter(lambda x: x.instance_type not in remove_these,instances)
    return new_instances

def recursive_lp_n(lp_output,lp_output_n,min_cores,min_ram,min_free_storage,max_cost_hour,ram_per_job,procs_per_job,old_names,
aws_instances):

```

```

new_instances = helper_recursive(lp_output_n, old_names, aws_instances)
num_types, names, prices, procs, gBRAM, freestorage, max_cost_in_previous_time_window, account_limits = formulate_problem(new_instances)
new_lp_output, new_lp_output_n = run_LP(num_types, names, prices, procs, gBRAM, freestorage, max_cost_in_previous_time_window,
account_limits, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job, procs_per_job, new_instances)
if type(new_lp_output_n.x) == float:
    return lp_output_n, old_names
if len(new_lp_output_n.x) == 0:
    return lp_output_n, old_names
if list(new_lp_output_n.x) == list(lp_output_n.x):
    return lp_output_n, old_names
else:
    new_return = recursive_lp_n(new_lp_output, new_lp_output_n, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job,
procs_per_job, names, new_instances)
    return new_return

def recursive_lp(lp_output, lp_output_n, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job, procs_per_job, old_names,
aws_instances):
    new_instances = helper_recursive(lp_output, old_names, aws_instances)
    num_types, names, prices, procs, gBRAM, freestorage, max_cost_in_previous_time_window, account_limits = formulate_problem(new_instances)
    new_lp_output, new_lp_output_n = run_LP(num_types, names, prices, procs, gBRAM, freestorage, max_cost_in_previous_time_window,
account_limits, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job, procs_per_job, new_instances)
    if type(new_lp_output.x) == float:
        return lp_output, old_names
    elif len(new_lp_output.x) == 0:
        return lp_output, old_names
    elif list(new_lp_output.x) == list(lp_output.x):
        return lp_output, old_names
    else:
        new_return = recursive_lp(new_lp_output, new_lp_output_n, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job,
procs_per_job, names, new_instances)
        return new_return

#add filtering for running instances and job size
def start_bidding():
    min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job, procs_per_job = get_user_params()
    aws_instances = define_A_matrix()
    if min_free_storage > 0:
        aws_instances = filter(lambda x: x.storage > 0, aws_instances)
    aws_instances = filter(lambda x: x.procs > procs_per_job, aws_instances)
    aws_instances = filter(lambda x: x.ram > ram_per_job, aws_instances)
    num_types, old_names, prices, procs, gBRAM, freestorage, max_cost_in_previous_time_window, account_limits = formulate_problem(
aws_instances)
    lp_output, lp_output_n = run_LP(num_types, old_names, prices, procs, gBRAM, freestorage, max_cost_in_previous_time_window,
account_limits, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job, procs_per_job, aws_instances)
    lp, names = recursive_lp(lp_output, lp_output_n, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job, procs_per_job,
old_names, aws_instances)
    lp_n, names_n = recursive_lp_n(lp_output, lp_output_n, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job, procs_per_job,
old_names, aws_instances)
    return lp, lp_n, names, names_n, aws_instances, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job, procs_per_job

def find_provisioning_info(name, aws_instances):
    pre_desired_instance = filter(lambda x: x.instance_type == '.'.join(name.split('.')[2:]), aws_instances)
    assert(len(pre_desired_instance) == 1)
    desired_instance = pre_desired_instance[0]
    #print procs
    procs = str(desired_instance.procs)
    ram = str(int(desired_instance.ram))
    storage = desired_instance.storage
    name = '.'.join(name.split('.')[2:])
    return procs, ram, storage, name

def write_prov_file(lp_output, names, aws_instances):
    prov_file = open(dircr+"prov.psv", 'w')
    out_data = zip(names, lp_output.x)
    sum_deploy = 0
    print("The following is the LP generated provisioning:")
    for elem in out_data:
        pre_name = elem[0]
        procs, ram, storage, name = find_provisioning_info(pre_name, aws_instances)
        boolstr = "true" if storage > 0 else "false"
        number_to_deploy = int(round(float(elem[1])))
        sum_deploy += number_to_deploy
        for count in range(0, number_to_deploy):
            print(name+'|'+procs+'|'+ram+'|'+boolstr+"|aws\n")
            prov_file.write(name+'|'+procs+'|'+ram+'|'+boolstr+"|aws\n")
    prov_file.close()
    if sum_deploy == 0:
        sys.exit(1)
    return

def go():
    try:
        lp_output, lp_output_n, names, names_n, aws_instances, min_cores, min_ram, min_free_storage, max_cost_hour, ram_per_job, procs_per_job =
start_bidding()
        write_prov_file(lp_output, names, aws_instances)
        return ram_per_job, procs_per_job
    except:
        print "No feasible solution found, try again with different parameters"
        return "exit", 0
"""
if len(lp_output_n.x) > 0:
    naive_out = zip(names_n, lp_output_n.x)
    print "\n"
    print "Going by the seat of your pants and choosing the cheapest options that meet your criteria at the current moment would
result in this bid:"

```

```

else:
    print filter(lambda x: x[1] != 0,naive_out)
print "There is no solution"
if len(lp_output) > 0:
    print "Taking in to account pricing variability, your ideal bid is:"
    cost_out = zip(names,lp_output.x)
    print filter(lambda x: x[1] != 0,cost_out)
"""

```

---

## D.I.2 TCR CLONE PROCESSING

```

import sys
import os

#input files: 1 tumor tcr filtered contig annotations; 2 blood tcr filtered contig annotations; 3 tumor projection; 4 blood projection;
5 tumor clonotypes; 6 blood clonotypes
#output files: 7 new tumor projection; 8 new blood projection

bctr_fh = open(sys.argv[2],'r')
ttcr_fh = open(sys.argv[1],'r')
bclonotypes_fh = open(sys.argv[6],'r')
tclonotypes_fh = open(sys.argv[5],'r')

blood_clono_to_barcode = {}
tumor_clono_to_barcode = {}
blood_clonotypes = set()
tumor_clonotypes = set()

#logic to handle multiple barcodes assigned to same clonotype
def make_clono_to_barcode_dicts(fh,dictionary):
    for lines in fh:
        line = lines.rstrip().split(',')
        if line[16] != "None":
            if line[10] == "True":
                if dictionary.has_key(line[16]):
                    existing_barcodes = dictionary[line[16]]
                    existing_barcodes.append(line[0])
                    dictionary[line[16]] = existing_barcodes
            else:
                dictionary[line[16]] = [line[0]]
    return dictionary

blood_clono_to_barcode = make_clono_to_barcode_dicts(bctr_fh,blood_clono_to_barcode)
tumor_clono_to_barcode = make_clono_to_barcode_dicts(ttcr_fh,tumor_clono_to_barcode)

bctr_fh.close()
ttcr_fh.close()

class Clonotype:
    def __init__(self,chains_aa,clonotype,clono_to_barcode,frequency):
        self.barcodes = clono_to_barcode[clonotype]
        #self.frequency = frequency
        self.frequency = str(len(set(self.barcodes)))
        self.tra = []
        self.trb = []
        chains_arr = chains_aa.split(';')
        for elem in chains_arr:
            chain = elem.split(':')
            if chain[0] == "TRA":
                self.tra.append(chain[1])
            if chain[0] == "TRB":
                self.trb.append(chain[1])
    def geta(self):
        return self.tra
    def getb(self):
        return self.trb
    def getname(self):
        return '-'.join(self.tra)+'|'+'-'.join(self.trb)

def make_clonotype_sets(fh,clono_set,clono_to_barcode):
    for lines in fh:
        line = lines.rstrip().split(',')
        clonotype_id = line[0]
        chains = line[3]
        frequency = line[1]
        if clonotype_id != "clonotype_id":
            if clono_to_barcode.has_key(clonotype_id):
                clonotype = Clonotype(chains,clonotype_id,clono_to_barcode,frequency)
                clono_set.add(clonotype)
    return clono_set

blood_clonotypes = make_clonotype_sets(bclonotypes_fh,blood_clonotypes,blood_clono_to_barcode)
tumor_clonotypes = make_clonotype_sets(tclonotypes_fh,tumor_clonotypes,tumor_clono_to_barcode)

```

```

bclonotypes_fh.close()
tclonotypes_fh.close()
tp_fh = open(sys.argv[3], 'r')
bp_fh = open(sys.argv[4], 'r')
output_tp_fh = open(sys.argv[7], 'w')
output_bp_fh = open(sys.argv[8], 'w')

def find_a_chain_matches(clonotype, other_set):
    if len(clonotype.tra) > 0:
        other_matches = set()
        for tra_chain in clonotype.tra:
            matches = filter(lambda ctype: tra_chain in ctype.tra, other_set)
            for match in matches:
                other_matches.add(match)
        if len(other_matches) == 0:
            return (matches, "")
        perfect_hits = filter(lambda ctype: clonotype.tra == ctype.tra, other_set)
        if len(perfect_hits) > 0:
            return (perfect_hits, 'f')
        return (matches, '')
    return (set(), '')

def find_b_chain_matches(clonotype, other_set):
    if len(clonotype.trb) > 0:
        other_matches = set()
        for trb_chain in clonotype.trb:
            matches = filter(lambda ctype: trb_chain in ctype.trb, other_set)
            for match in matches:
                other_matches.add(match)
        if len(other_matches) == 0:
            return (matches, "")
        perfect_hits = filter(lambda ctype: clonotype.trb == ctype.trb, other_set)
        if len(perfect_hits) > 0:
            return (perfect_hits, 'f')
        return (matches, '')
    return (set(), '')

def make_tsne_output(input_fh, output_fh, matching_set, other_set):
    clonotype_dict = {}
    count = 0
    for lines in input_fh:
        line = lines.rstrip().split(',')
        barcode = line[0]
        clonotype_hits = filter(lambda ctype: barcode in ctype.barcodes, matching_set)
        assert len(clonotype_hits) < 2
        if len(clonotype_hits) > 0:
            clonotype = clonotype_hits[0]
            #find all a chain matches
            a_matches = find_a_chain_matches(clonotype, other_set)
            #find all b chain matches
            b_matches = find_b_chain_matches(clonotype, other_set)
            if b_matches[1] == a_matches[1] == 'f':
                state = False
                #perfect matches
                for x in b_matches[0]:
                    for y in a_matches[0]:
                        if x.geta() == y.geta() == clonotype.geta():
                            if x.getb() == y.getb() == clonotype.getb():
                                #check that a clonotype exists with both a perfect a chain and b chain match
                                assert len(set(b_matches[0]).intersection(set(a_matches[0]))) > 0
                                state = True
                                strkey = ",".join(y.geta()+'|'+", ".join(y.getb()))
                                if clonotype_dict.has_key(strkey):
                                    t = clonotype_dict[strkey]
                                    t.append([lines.rstrip(), "matching", clonotype.frequency, clonotype.getname()])
                                    clonotype_dict[strkey] = t
                                else:
                                    clonotype_dict[strkey] = [[lines.rstrip(), "matching", clonotype.frequency, clonotype.getname()]]
            if not state:
                #a and b chains both present and from different clonotypes --- not counted as hit
                aset = set(sum(map(lambda x:x.geta(), a_matches[0]), [])) #geta gives list of clonotypes, so map returns list of
                #lists that needs to be flattened
                bset = set(sum(map(lambda x:x.getb(), b_matches[0]), []))
                #check that both a and b chains have a clonotype match
                assert len(set(clonotype.geta()).intersection(aset)) > 0 and len(set(clonotype.getb()).intersection(bset)) > 0
                #confirm that separate a and b chain matches are assigned to different clonotypes
                assert len(set(b_matches[0]).intersection(set(a_matches[0]))) == 0
                output_fh.write(lines.rstrip()+",not_matching,"+clonotype.frequency+', '+clonotype.getname()+'\n')
            else:
                # one matching chain, missing second chain --- counted as hit
                state1 = False
                if bool(len(b_matches[0]) > 0) ^ bool(len(a_matches[0]) > 0):
                    if len(b_matches[0]) > 0:
                        if clonotype.getb() in map(lambda x: x.trb, b_matches[0]):
                            #confirm that there are no a matches
                            assert len(map(lambda x: x.tra, a_matches[0])) == 0
                            for x in b_matches[0]:
                                if len(x.tra) == 0:
                                    #confirm that full set of chains match
                                    assert clonotype.getb() == x.getb()
                                    if len(clonotype.tra) == 0:
                                        strkey = ",".join(x.geta()+'|'+", ".join(x.getb()))
                                        state1 = True
                                        if clonotype_dict.has_key(strkey):

```



```

        t = clonotype_dict[strkey]
        t.append([lines.rstrip(), "beta_matching", clonotype.frequency, clonotype.getname()])
        clonotype_dict[strkey] = t
    else:
        clonotype_dict[strkey] = [[lines.rstrip(), "beta_matching", clonotype.frequency, clonotype.
            getname()]]
    if len(a_matches[0]) > 0:
        if clonotype.geta() in map(lambda x: x.tra, a_matches[0]):
            #confirm that there are no b matches
            assert len(map(lambda x: x.trb, b_matches[0])) == 0
            for x in a_matches[0]:
                if len(x.trb) == 0:
                    #confirm that full set of chains match
                    assert clonotype.geta() == x.geta()
                    if len(clonotype.trb) == 0:
                        strkey = ".".join(x.geta())+'|'+".join(x.getb())
                        state1 = True
                        if clonotype_dict.has_key(strkey):
                            t = clonotype_dict[strkey]
                            t.append([lines.rstrip(), "matching", clonotype.frequency, clonotype.getname()])
                            clonotype_dict[strkey] = t
                        else:
                            clonotype_dict[strkey] = [[lines.rstrip(), "matching", clonotype.frequency, clonotype.getname
                                ()]]
                    if not state1:
                        assert(len(clonotype.getname()) > 1)
                        output_fh.write(lines.rstrip()+"not_matching", "+clonotype.frequency+', '+clonotype.getname()+'\n')
                else:
                    assert(len(clonotype.getname()) > 1)
                    output_fh.write(lines.rstrip()+"not_matching", "+clonotype.frequency+', '+clonotype.getname()+'\n')
    else:
        if count == 0:
            output_fh.write(lines.rstrip()+"Group, Frequency, Tcr\n")
            count = 1
        else:
            assert(len(clonotype.getname()) > 1)
            output_fh.write(lines.rstrip()+"no_clonotype, 1, notcr\n")
    return clonotype_dict

tumors = make_tsne_output(tp_fh, output_tp_fh, tumor_clonotypes, blood_clonotypes)
bloods = make_tsne_output(bp_fh, output_bp_fh, blood_clonotypes, tumor_clonotypes)

#print bloods

intersection = set(bloods.keys()).intersection(set(tumors.keys()))

def write_out_matches(d, intersection, fh):
    for key in d.keys():
        if key in intersection:
            for x in d[key]:
                fh.write(",".join(x)+'\n')
        else:
            print "hit"
            for x in d[key]:
                x[1] = "not_matching"
                #fh.write(",".join(x)+'\n')

write_out_matches(bloods, intersection, output_bp_fh)
write_out_matches(tumors, intersection, output_tp_fh)

output_tp_fh.close()
output_bp_fh.close()

tp_fh.close()
bp_fh.close()
output_tp_fh.close()
output_bp_fh.close()

```

---

## D.1.3 MOUSE MACHINE LEARNING

---

```

#!/usr/bin/env python
# coding: utf-8

# In[1]:

from __future__ import division
import numpy as np
import sys
from sklearn.decomposition import PCA

```

```

import math
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve, auc, roc_auc_score
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve

# In[17]:

arr = []
f = open("counts_2chains_surface.csv", 'r')
for lines in f:
    line = lines.rstrip().split(',')
    arr.append(line)
f.close()
nparr = np.array(arr)
print "data loaded"
nparr_t = np.transpose(nparr)[1:]
print len(nparr_t)
print "done transposing"
labels = []
dataarr = []
count = 0
for elem in nparr_t:
    print count
    count += 1
    labels.append(elem[0])
    unlabeled = map(lambda x: int(x), elem[1:])
    sumrow = sum(unlabeled)
    normalized = map(lambda x: math.log(x/sumrow) if x > 0 else 0, unlabeled)
    dataarr.append(normalized)
print "log normalized"
npdataarr = np.array(dataarr)

# In[18]:

pca = PCA(n_components=int(100))
transformed = pca.fit_transform(npdataarr)
print "transformed"

labels_d = {}
label_fh = open("metadata.csv", 'r')
for lines in label_fh:
    line = lines.rstrip().split(',')
    if line[9] == "Group":
        continue
    if line[9] != "not_matching" and line[3] != "no_clonotype":
        labels_d[line[0]] = 1
    else:
        labels_d[line[0]] = 0
label_fh.close()

Xpre = []
ypre = []

for i in range(0, len(labels)):
    if labels_d.has_key(labels[i]):
        ypre.append(labels_d[labels[i]])
        Xpre.append(transformed[i])

X = np.array(Xpre)
y = np.array(ypre)

# In[19]:

class_weight_d = {0:1, 1: 10000}

# In[20]:

FOLDS = 5
k_fold = KFold(n_splits=FOLDS, shuffle=True, random_state=32)
predictor = LogisticRegression(random_state=0, solver='liblinear', penalty="l2", C=.02, class_weight=class_weight_d)

# In[21]:

f, axes = plt.subplots(1, 3, figsize=(10, 5))
axes[0].scatter(X[y==0,0], X[y==0,1], color='blue', s=2, label='y=0')
axes[0].scatter(X[y!=0,0], X[y!=0,1], color='red', s=2, label='y=1')
axes[0].set_xlabel('X[:,0]')
axes[0].set_ylabel('X[:,1]')
axes[0].legend(loc='lower left', fontsize='small')

y_real = []
y_proba = []
y_pred = []
for i, (train_index, test_index) in enumerate(k_fold.split(X)):

```

```

Xtrain, Xtest = X[train_index], X[test_index]
ytrain, ytest = y[train_index], y[test_index]
predictor.fit(Xtrain, ytrain)
pred_proba = predictor.predict_proba(Xtest)
pred = predictor.predict(Xtest)
#explainer = shap.LinearExplainer(pred,Xtrain,feature_dependence="independent")
#shap_values = explainer.shap_values(X_test)
#X_test_array = X_test.toarray()
#shap.summary_plot(shap_values,X_test_array,feature_names=labels)
precision, recall, _ = precision_recall_curve(ytest, pred_proba[:,1])
fpr, tpr, _ = roc_curve(ytest, np.array(pred))
lab = 'Fold %d AUPRC=%0.4f' % (i+1, auc(recall, precision))
lab2 = 'Fold %d AUC=%0.4f' % (i+1, roc_auc_score(ytest, pred_proba[:,1]))
axes[2].step(recall, precision, label=lab)
axes[1].step(fpr, tpr, label=lab2)
y_real.append(ytest)
y_proba.append(pred_proba[:,1])
y_pred.append(pred)

y_real = np.concatenate(y_real)
y_proba = np.concatenate(y_proba)
y_pred = np.concatenate(y_pred)
precision, recall, _ = precision_recall_curve(y_real, y_proba)
fpr, tpr, _ = roc_curve(y_real, y_pred)
no_skill = len(y_real[y_real==1]) / len(y_real)
plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')
lab = 'Overall AUPRC=%0.4f' % (auc(recall, precision))
lab2 = 'Overall AUC=%0.4f' % (roc_auc_score(y_real, y_proba))
axes[2].step(recall, precision, lw=2,label=lab, color='black')
axes[2].set_xlabel('Recall')
axes[2].set_ylabel('Precision')
axes[2].legend(loc='lower left', fontsize='small')
axes[1].step(fpr, tpr, label=lab2, lw=2, color='black')
axes[1].set_xlabel('FPR')
axes[1].set_ylabel('TPR')
axes[1].legend(loc='lower left', fontsize='small')

f.tight_layout()
stro = "result"+"100"+" pdf"
f.savefig(stro)

print "done"

# In[22]:

clf = RandomForestClassifier(random_state=0, n_jobs=-1)
model = clf.fit(dataarr, y)

# In[23]:

importances = model.feature_importances_

# In[24]:

feature_names = []
f = open("counts_2chains.csv",'r')
for lines in f:
    if len(lines.rstrip().split(',')[0]) > 0:
        feature_names.append(lines.rstrip().split(',')[0])
f.close()

# In[25]:

indices = np.argsort(importances)[::-1]

names = [feature_names[i] for i in indices[0:20]]

plt.figure()

plt.title("Feature Importance")
imps = importances[indices][0:20]

plt.bar(range(0,20), imps)

plt.xticks(range(0,20), names, rotation=90)
stro = "result"+"_features"+" pdf"
plt.savefig(stro,bbox_inches='tight')

plt.show()

# In[190]:

indices[0:10]

# In[ ]:

```

```

# In[168]:

1

# In[162]:

importances[indices][0:9]

# In[142]:

# In[144]:

# In[ ]:

```

---

## D.I.4 MOUSE SEURAT PROCESSING

```

#!/usr/bin/env Rscript
library(Seurat)
library(dplyr)
library(data.table)
library(ggplot2)
library(cowplot)
library(viridis)
library(gridExtra)
library(RColorBrewer)
library(tibble)
#1: 10X dir, 2: freq file, 3: sample, 4: output folder loc
#Example: Rscript pipeline.R /Users/jacobluber/Desktop/ucsf/working_so/NGEX/outs/filtered_gene_bc_matrices/GRCh38 /Users/jacobluber/
Dropbox\ \(\HMS\) \Jacob\ Work\Analysis\layn\chi-square\working_so_plotting.csv working_so /Users/jacobluber/Dropbox\ \(\HMS\) /
Jacob\ Work\working\working_so/
#parse command line data
#args = commandArgs(trailingOnly=TRUE)
#tenxdir <- args[1]
#freq <- args[2]
#sample <- args[3]
setwd("/Users/jacobluber/Desktop/cwd/p1m1")

data <- Read10X(data.dir = "/Volumes/Jacob's Backup /p1m1")
#read data from Kraken and create Seurat object
#data <- Read10X(data.dir = tenxdir)

#data <- Read10X(data.dir = "/Volumes/Jacob's Backup /k468")
data2 <- CreateSeuratObject(data, min.cells = 3,min.features=400, project = "test")
housekeeping_genes_pre <- read.table("/Users/jacobluber/Desktop/cwd/HK_Satija_mus_musculs.david.converted.txt",stringsAsFactors = FALSE)
)
housekeeping_genes <- housekeeping_genes_pre$V1
housekeeping_genes <- intersect(rownames(data2), housekeeping_genes)
subset_counts_matrix <- data.frame(data2@assays$RNA@counts)[housekeeping_genes,]
subset_counts_matrix[subset_counts_matrix > 0] = 1
summary(colSums(subset_counts_matrix))
hist(colSums(subset_counts_matrix),main="> .5 selected as cutoff for HK genes")
pdf("HK_genes_hist.pdf",width=7,height=7)
hist(colSums(subset_counts_matrix),main="> .5 selected as cutoff for HK genes")
dev.off()
counts <- colSums(subset_counts_matrix)
counts2 <- data.frame(counts)
counts3 <- counts2 %>% tibble::rownames_to_column() %>% filter(counts > 48)

all_genes <- rownames(x = data2)
mito_pre <- read.table("/Users/jacobluber/Desktop/cwd/mito_genes_mouse.txt",stringsAsFactors = FALSE)
mito_genes <- mito_pre$V1
mito_genes <- intersect(rownames(data2), mito_genes)
subset_counts_matrix <- data.frame(data2@assays$RNA@counts)[mito_genes,]
subset_counts_matrix[subset_counts_matrix > 0] = 1
summary(colSums(subset_counts_matrix))
hist(colSums(subset_counts_matrix),main="< 200/1158 genes (Broad list) selected as cutoff for mito genes")
pdf("mito_genes_hist.pdf",width=7,height=7)

```

```

hist(colSums(subset_counts_matrix),main=" < 200/1158 genes (Broad list) selected as cutoff for mito genes")
dev.off()

counts4 <- colSums(subset_counts_matrix)
counts5 <- data.frame(counts4)
counts6 <- counts5 %>% tibble::rownames_to_column() %>% filter(counts4 < 200)

cells <- intersect(counts6$rowname,counts3$rowname)

#Run all processing through UMAP step
#all.genes <- rownames(x = working_so)
#housekeeping_genes_pre <- read.table("/Users/jacobluber/Desktop/cwd/HK_Satija.txt")
#housekeeping_genes <- housekeeping_genes_pre$V1
#non.mito.genes <- grep("^MT-", x = all.genes, value = TRUE, invert = TRUE)
#mito.genes <- grep("^MT-", x = all.genes, value = TRUE, invert = FALSE)

#housekeeping and mito gene plots
# hist_arr_hk = c()
# for (i in 1:length(colnames(data2@assays$RNA@counts))) {
#   hk_genes <- 0
#   for (j in 1:length(housekeeping_genes)) {
#     if (data2@assays$RNA@counts[,i][housekeeping_genes[j]] == 1){
#       old <- hk_genes
#       hk_genes <- old+1
#     }
#   }
#   hist_arr_hk<- c(hist_arr_hk,hk_genes)
# }

# hist_arr_hk2 = c()
# for (i in 1:100) {
#   print(i)
#   hk_genes <- 0
#   for (j in 1:length(housekeeping_genes)) {
#     if (data2@assays$RNA@counts[,i][housekeeping_genes[j]] >= 1){
#       hk_genes <- hk_genes+1
#     }
#   }
#   hist_arr_hk2<- c(hist_arr_hk2,hk_genes/100)
# }

# m_genes_arr <- c()
# for (i in 1:300) {
#   print(i)
#   m_genes <- 0
#   for (j in 1:length(mito.genes)) {
#     if (data2@assays$RNA@counts[,i][mito.genes[j]] == 1){
#       m_genes <- m_genes+1
#     }
#   }
#   m_genes_arr<- c(m_genes_arr,m_genes/13)
# }

working_so <- NormalizeData(data2)
working_so <- SubsetData(working_so, cells = cells)
working_so <- ScaleData(object = working_so, features = rownames(working_so))
working_so <- FindVariableFeatures(object = working_so, mean.function = ExpMean, dispersion.function = LogVMR,
do.plot = FALSE)
hv.genes <- head(x = VariableFeatures(object = working_so), 1000)
working_so <- RunPCA(object = working_so, pc.genes = hv.genes, do.print = FALSE, pcs.print = 1:5,
genes.print = 5, pcs.compute = 50)
working_so <- FindNeighbors(object = working_so, dims = 1:30)
working_so <- FindClusters(object = working_so, resolution = 1.2)
working_so <- RunUMAP(object = working_so, reduction.use = "pca", dims = 1:15, n_neighbors = 15, min_dist = 0.3)

#optionally read in existing Seurat object
#working_so <- readRDS("/Users/jacobluber/browser/k409.ln-blood.rds")

#get the tcr data to determine matching cluster status, and add various other things to the metadata
#tcr <- read.csv("/Users/jacobluber/browser/k468.blood-ln.csv")
ugh1 <- readRDS("/Users/jacobluber/browser/p1m1.blood-tumor.rds")
tcr <- ugh1@meta.data
#tcr <- read.csv(fread)
tcr.identids <- tcr$matching
#CHECK INPUT FILE
tcr$pcluster <- as.numeric(tcr$matching)
tcr$pcluster[tcr$pcluster==1] <- 1
tcr$pcluster[tcr$pcluster==2] <- 1
tcr$pcluster[tcr$pcluster==3] <- 1
tcr$pcluster[tcr$pcluster==4] <- 2
tcr$pcluster[tcr$pcluster==5] <- 2
tcr$ambig <- as.numeric(tcr$matching)
tcr$ambig[tcr$ambig==1] <- 1
tcr$ambig[tcr$ambig==2] <- 1
tcr$ambig[tcr$ambig==3] <- 2
tcr$ambig[tcr$ambig==4] <- 2
tcr$ambig[tcr$ambig==5] <- 2
tcr$consis <- as.numeric(tcr$matching)
tcr$consis[tcr$consis==1] <- 2
tcr$consis[tcr$consis==2] <- 2
tcr$consis[tcr$consis==3] <- 1
tcr$consis[tcr$consis==4] <- 2
tcr$consis[tcr$consis==5] <- 2
tcr$Barcode <- rownames(tcr)
tcr$Barcode <- gsub("(.)\\1", "\\1",tcr$Barcode)
names(tcr.identids)=tcr$Barcode
tcr.pclusters <- tcr$pcluster
tcr.ambig <- tcr$ambig

```

```

names(tcr.pclusters)=tcr$Barcode
names(tcr.ambig)=tcr$Barcode
tcr.consistis <- tcr$consistis
names(tcr.consistis)=tcr$Barcode
working_so <- AddMetaData(working_so,tcr.pclusters,col.name='pcluster')
working_so <- AddMetaData(working_so,tcr.consistis,col.name='consistent')
working_so <- AddMetaData(working_so,tcr.ambig,col.name='ambig')
working_so <- AddMetaData(working_so,tcr.identis,col.name='matching')
working_so <- AddMetaData(working_so,data.frame(working_so@reductions$umap@cell.embeddings)$UMAP_1,col.name='u1')
working_so <- AddMetaData(working_so,data.frame(working_so@reductions$umap@cell.embeddings)$UMAP_2,col.name='u2')

#determine which clusters to keep
clusters <- levels(working_so@meta.data$seurat_clusters)

#iterative function to see if a given cluster should be kept, results aggregate like a fold in Haskell
see_if_keep_cluster <- function(so, cluster, identis){
  copy_so <- so
  cso <- SubsetData(copy_so, ident.use = cluster, do.clean = TRUE, do.scale = TRUE)
  new_id <- NULL
  cd3e <- sum(as.numeric(GetAssayData(object = cso, slot = "data")["Cd3e",]>0))/nrow(cso@meta.data)
  cd3d <- sum(as.numeric(GetAssayData(object = cso, slot = "data")["Cd3d",]>0))/nrow(cso@meta.data)
  cd3g <- sum(as.numeric(GetAssayData(object = cso, slot = "data")["Cd3g",]>0))/nrow(cso@meta.data)
  cd8b <- sum(as.numeric(GetAssayData(object = cso, slot = "data")["Cd8b1",]>0))/nrow(cso@meta.data)
  cd8a <- sum(as.numeric(GetAssayData(object = cso, slot = "data")["Cd8a",]>0))/nrow(cso@meta.data)
  foxp3 <- sum(as.numeric(GetAssayData(object = cso, slot = "data")["Foxp3",]>0))/nrow(cso@meta.data)
  #cd4 <- sum(as.numeric(GetAssayData(object = cso, slot = "data")["Cd4",]>0))/nrow(cso@meta.data)
  if (cd3e > .3 || cd3d > .3 || cd3g > .3) {
    if (cd8b > .3 && cd8a > .3 && foxp3 < .05) {
      new_id <- cluster
    }
  }
  identis <- c(identis,new_id)
  return(identis)
}

identis <- c()
for (i in clusters){
  identis <- see_if_keep_cluster(working_so, as.numeric(i),identis)
}

#CHANGE
sample <- "p1m2"

working_so1 <- SubsetData(working_so, ident.use = identis, do.clean = TRUE, do.scale = TRUE)
working_so1 <- ScaleData(object = working_so1, features = rownames(working_so1))
working_so1 <- FindVariableFeatures(object = working_so1, mean.function = ExpMean, dispersion.function = LogVMR,
  do.plot = FALSE)
hv.genes <- head(x = VariableFeatures(object = working_so1), 1000)
working_so1 <- RunPCA(object = working_so1, pc.genes = hv.genes, do.print = FALSE, pcs.print = 1:5,
  genes.print = 5, pcs.compute = 50)
working_so1 <- FindNeighbors(object = working_so1, dims = 1:30)
working_so1 <- FindClusters(object = working_so1, resolution = 1.2)
working_so1 <- RunUMAP(object = working_so1, reduction.use = "pca", dims = 1:15, n_neighbors = 15, min_dist = 0.3)

#Generate UMAPs to check cluster selection
t1 <- paste0("Clusters Kept=",paste(identis, collapse=', '))
p1 <- DimPlot(object = working_so, reduction.use = "umap", no.legend = FALSE, do.return = TRUE, label = TRUE, vector.friendly = TRUE,
  pt.size = .4) + ggtitle(paste0(length(colnames(data2)), " -> ",length(colnames(working_so)), " cells (HK/mito filtering)") +
  theme(plot.title = element_text(hjust = 0.5)) + coord.fixed()
p2 <- DimPlot(object = working_so1, reduction.use = "umap", no.legend = FALSE, do.return = TRUE, label = TRUE, vector.friendly = TRUE,
  pt.size = .4) + ggtitle(paste0(length(colnames(working_so)), " -> ",length(colnames(working_so1)), " cells (CD8 filtering)") +
  theme(plot.title = element_text(hjust = 0.5)) + coord.fixed() + labs(subtitle = t1)
p3 <- FeaturePlot(working_so1,features=c("Foxp3","Cd4","Cd3g","Cd8b1","Cd8a","Cd3d","Cd3e"), reduction = "umap",cols=c("gray","red"),pt
  .size = .4, coord.fixed = TRUE,combine=FALSE)
title <- paste0(sample,"_UMAP_original.pdf")
pdf(title,width=7,height=7)
p1
dev.off()
pdf(paste0(sample,"_UMAP_cd8s.pdf"),width=7,height=7)
p2
dev.off()
pdf(paste0(sample,"_UMAP_genes.pdf"),width=7,height=7)
p3
dev.off()
pdf(paste0(sample,"_UMAP_matching_plots.pdf"),width=7,height=7)
FeaturePlot(working_so1,features=c("pcluster"), reduction = "umap",pt.size = .4, coord.fixed = TRUE) + NoLegend() + labs(title = "
  Definition Used (consistent U ambiguous), matching in grey")
FeaturePlot(working_so1,features=c("ambig"), reduction = "umap",pt.size = .4, coord.fixed = TRUE) + NoLegend() + labs(title = "
  Ambiguous Definition, matching in grey")
FeaturePlot(working_so1,features=c("consistent"), reduction = "umap",pt.size = .4, coord.fixed = TRUE) + NoLegend() + labs(title = "
  Consistent Definition, matching in grey")
dev.off()

p4 <- FeaturePlot(working_so,features=c("Foxp3","Cd4","Cd3g","Cd8b1","Cd8a","Cd3d","Cd3e"), reduction = "umap",cols=c("gray","red"),pt.
  size = .4, coord.fixed = TRUE,combine=FALSE)
title1 <- paste0(sample,"_UMAP_genes_noCD8_filtering.pdf")
pdf(title1,width=20,height=6)
p4
dev.off()

#rerun UMAP pipeline on data with non CD8 clusters removed
#working_so1 <- SubsetData(working_so, ident.use = identis, do.clean = TRUE, do.scale = TRUE)

markers <- FindMarkers(working_so1, ident.1 = 1, ident.2 = NULL, only.pos = TRUE,test.use = "MAST")
write.table(markers,"MAST_matching_markers.txt")

```

```

#write output files for COMET
write.table(GetAssayData(object = working_so1, slot = "counts"),file=paste0("counts_matrix", ".txt"),sep="\t",quote=FALSE,row.names=TRUE)
write.table(GetAssayData(object = working_so1, slot = "scale.data"),file=paste0("normalized_matrix", ".txt"),sep="\t",quote=FALSE,row.names=TRUE)
write.table(Embeddings(working_so1[["umap"]]),file=paste0("viz", ".txt"),sep="\t",quote=FALSE,row.names=TRUE)
cluster_out <- working_so1@meta.data %>% tibble::rownames_to_column('barcode') %>% select(barcode,pcluster)
write.table(cluster_out,file=paste0("clusters", ".txt"),sep="\t",quote=FALSE,row.names=FALSE)
write.csv(working_so1@meta.data,file=paste0("metadata", ".csv"))
#write out metadata
saveRDS(working_so, file = paste0(sample, ".rds"))

#p1 <- DimPlot(object = working_so, reduction.use = "umap", no.legend = FALSE, do.return = TRUE, label = TRUE, vector.friendly = TRUE,
pt.size = 1) + ggtitle("UMAP Clusters Removed") + theme(plot.title = element_text(hjust = 0.5)) + coord_fixed()

```

## D.I.5 MOUSE SAMPLE INTEGRATION

```

library(Seurat)

p1m1_tumor <- readRDS(file = "/Users/jacobluber/Dropbox (HMS)/blood-tumor_pipeline/tumor/p1m1/p1m1.rds")
p2m1_tumor <- readRDS(file = "/Users/jacobluber/Dropbox (HMS)/blood-tumor_pipeline/tumor/p2m1/p2m1.rds")
p2m2_tumor <- readRDS(file = "/Users/jacobluber/Dropbox (HMS)/blood-tumor_pipeline/tumor/p2m2/p2m2.rds")
p1m1_blood <- readRDS(file = "/Users/jacobluber/Dropbox (HMS)/blood-tumor_pipeline/blood/p1m1/p1m1.rds")
p2m1_blood <- readRDS(file = "/Users/jacobluber/Dropbox (HMS)/blood-tumor_pipeline/blood/p2m1/p2m1.rds")
p2m2_blood <- readRDS(file = "/Users/jacobluber/Dropbox (HMS)/blood-tumor_pipeline/blood/p2m2/p2m2.rds")

addLevel <- function(x, newLevel=NULL) {
  if(is.factor(x)) {
    if (is.na(match(newLevel, levels(x))))
      return(factor(x, levels=c(levels(x), newLevel)))
    }
  }
  return(x)
}

fixMatching <- function(seurat_obj,sample) {
  seurat_obj@meta.data$matching <- addLevel(seurat_obj@meta.data$matching,paste0(sample,"_matching"))
  seurat_obj@meta.data$matching <- addLevel(seurat_obj@meta.data$matching,"not_matching")
  seurat_obj@meta.data$matching[seurat_obj@meta.data$matching == "beta_matching"] <- "not_matching"
  seurat_obj@meta.data$matching[seurat_obj@meta.data$matching == "alpha_matching"] <- "not_matching"
  seurat_obj@meta.data$matching[seurat_obj@meta.data$matching == "no_clonotype"] <- "not_matching"
  seurat_obj@meta.data$matching[seurat_obj@meta.data$matching == "both_matching"] <- paste0(sample, "_matching")
  seurat_obj@meta.data$matching[seurat_obj@meta.data$matching == "matching"] <- paste0(sample, "_matching")
  sample <- rep(sample,length(seurat_obj@meta.data$matching))
  names(sample) <- rownames(seurat_obj@meta.data)
  seurat_obj<- AddMetaData(seurat_obj,sample,col.name='sample')
  return(seurat_obj)
}

p1m1_tumor <- fixMatching(p1m1_tumor,"p1m1_tumor")
p2m1_tumor <- fixMatching(p2m1_tumor,"p2m1_tumor")
p2m2_tumor <- fixMatching(p2m2_tumor,"p2m2_tumor")
p1m1_blood <- fixMatching(p1m1_blood,"p1m1_blood")
p2m1_blood <- fixMatching(p2m1_blood,"p2m1_blood")
p2m2_blood <- fixMatching(p2m2_blood,"p2m2_blood")

samples.list <- c(p1m1_tumor,p2m1_tumor,p2m2_tumor,p1m1_blood,p2m1_blood,p2m2_blood)
for (i in 1:length(samples.list)) {
  samples.list[[i]] <- SCTransform(samples.list[[i]], verbose = T)
}

options(future.globals.maxSize = 3145728000) #Jacob's note: 3000 * 1024^2 bytes to resolve error with 2GB list
samples.features <- SelectIntegrationFeatures(object.list = samples.list, nfeatures = 3000)
samples.list <- PrepSCTIntegration(object.list = samples.list, anchor.features = samples.features,
  verbose = T)

samples.anchors <- FindIntegrationAnchors(object.list = samples.list, normalization.method = "SCT",
  anchor.features = samples.features, verbose = T)
Sys.setenv('R_MAX_VSIZE' = 10000000000)
samples.integrated <- IntegrateData(anchorset = samples.anchors, normalization.method = "SCT",
  verbose = T)

# integration.anchors <- FindIntegrationAnchors(object.list = list(p1m1,p2m1,p2m2), dims = 1:30, verbose=T)
# integrated.mouse.blood <- IntegrateData(anchorset = integration.anchors, dims = 1:30)
#
#
DefaultAssay(object = samples.integrated) <- "RNA"
samples.integrated <- ScaleData(samples.integrated , verbose = T)
samples.integrated <- RunPCA(samples.integrated , npcs = 30, verbose = T)
#
#
samples.integrated <- backup
samples.integrated <- FindNeighbors(object = samples.integrated , dims = 1:30)
samples.integrated <- FindClusters(samples.integrated , resolution = .30,verbose=T)

```

```

samples.integrated <- RunUMAP(samples.integrated , reduction = "pca", dims = 1:30)

#samples.integrated@meta.data$matching <- replace(samples.integrated@meta.data$matching,is.na(samples.integrated@meta.data$matching),"
p2m2_blood_matching")

setwd("/Users/jacobluber/Dropbox (HMS)/blood-tumor_pipeline/ptime/second_clustering")

DefaultAssay(object = tumor.integrated) <- "RNA"
saveRDS(tumor.integrated, file = "mouse_tumor_integrated.RDS")
pdf("mouse_sample.pdf",width=7,height=7)
DimPlot(samples.integrated , reduction = "umap", label = FALSE,
        repel = TRUE, group.by = "sample",coord.fixed = TRUE) + ggtitle("Integrated Mouse Tumor Samples")
dev.off()

pdf("mouse_matching_p1m1.pdf",width=7,height=7)
DimPlot(samples.integrated, reduction = "umap",group.by = "matching",coord.fixed = TRUE,cols=c("gray","red","blue","gray","gray","gray",
"gray"))+ ggtitle("Integrated Mouse Tumor Cells Matching To Blood \n [8540 Total Cells, 2427 Matching Cells]")
dev.off()

pdf("mouse_matching_p2m1.pdf",width=7,height=7)
DimPlot(samples.integrated, reduction = "umap",group.by = "matching",coord.fixed = TRUE,cols=c("gray","gray","gray","red","blue","gray",
"gray"))+ ggtitle("Integrated Mouse Tumor Cells Matching To Blood \n [8540 Total Cells, 2427 Matching Cells]")
dev.off()

pdf("mouse_matching_p2m2.pdf",width=7,height=7)
DimPlot(samples.integrated, reduction = "umap",group.by = "matching",coord.fixed = TRUE,cols=c("gray","gray","gray","gray","gray","red",
"blue"))+ ggtitle("Integrated Mouse Tumor Cells Matching To Blood \n [8540 Total Cells, 2427 Matching Cells]")
dev.off()

pdf("mouse_clusters2.pdf",width=7,height=7)
DimPlot(samples.integrated, reduction = "umap",coord.fixed = TRUE)+ ggtitle("Integrated Mouse Tumor Clusters")
dev.off()

pdf("mouse_tumor_markers.pdf",width=7,height=7)
FeaturePlot(samples.integrated,features=c("Foxp3","Cd4","Cd3g","Cd8a","Cd3d","Cd3e","Pdcd1","Ctla4","Sell","Gzmb","Mki67","Il7r","Lag3",
"Tcf7","Entpd1","Ifng","Foxp3"), reduction = "umap",cols=c("gray","red"),pt.size = .4, coord.fixed = TRUE,combine=FALSE)
dev.off()

#b2<-subset(blood.integrated, subset = sample != "NA")

markers_mast <- FindAllMarkers(samples.integrated, only.pos = TRUE,test.use = "MAST")
markers <- FindAllMarkers(samples.integrated, only.pos = TRUE)
write.csv(markers_mast,"mouse_markers_MAST2.csv")
write.csv(markers,"mouse_markers2.csv")

saveRDS(samples.integrated, file = "integration.RDS")

#saveRDS(integrated.k468, file = "/Users/jacobluber/Desktop/ptime/k468_1.rds")

# blood@meta.data$matching <- addLevel(blood@meta.data$matching, "blood_not_matching")
# blood@meta.data$matching[blood@meta.data$matching == "beta_matching"] <- "blood_not_matching"
# blood@meta.data$matching[blood@meta.data$matching == "alpha_matching"] <- "blood_not_matching"
# blood@meta.data$matching[blood@meta.data$matching == "not_matching"] <- "blood_not_matching"
# blood@meta.data$matching[blood@meta.data$matching == "no_clonotype"] <- "blood_not_matching"
# blood@meta.data$matching <- addLevel(blood@meta.data$matching, "blood_matching")
# blood@meta.data$matching[blood@meta.data$matching == "matching"] <- "blood_matching"
# tumor@meta.data$matching <- addLevel(tumor@meta.data$matching, "tumor_not_matching")
# tumor@meta.data$matching[tumor@meta.data$matching == "beta_matching"] <- "tumor_not_matching"
# tumor@meta.data$matching[tumor@meta.data$matching == "not_matching"] <- "tumor_not_matching"
# tumor@meta.data$matching[tumor@meta.data$matching == "alpha_matching"] <- "tumor_not_matching"
# tumor@meta.data$matching[tumor@meta.data$matching == "no_clonotype"] <- "tumor_not_matching"
# tumor@meta.data$matching <- addLevel(tumor@meta.data$matching, "tumor_matching")
# tumor@meta.data$matching[tumor@meta.data$matching == "matching"] <- "tumor_matching"

integration.anchors <- FindIntegrationAnchors(object.list = list(p1m1_tumor,p2m1_tumor,p2m2_tumor,p1m1_blood,p2m1_blood,p2m2_blood),
        dims = 1:30, verbose=T)
integrated.mouse.blood <- IntegrateData(anchorset = integration.anchors, dims = 1:20)

backup <- integrated.mouse.blood

integrated.mouse.blood <- ScaleData(integrated.mouse.blood , verbose = FALSE)
integrated.mouse.blood <- RunPCA(integrated.mouse.blood , npcs = 30, verbose = FALSE)
integrated.mouse.blood <- FindNeighbors(object = integrated.mouse.blood, dims = 1:30)
integrated.mouse.blood <- RunUMAP(integrated.mouse.blood , reduction = "pca", dims = 1:30)

setwd("/Users/jacobluber/Dropbox (HMS)/blood-tumor_pipeline/ptime/")
pdf("sample.pdf",width=7,height=7)
DimPlot(integrated.mouse.blood , reduction = "umap", label = TRUE,
        repel = TRUE, group.by = "sample")
dev.off()

DimPlot(integrated.mouse.blood , reduction = "umap", group.by = "matching")
DimPlot(object = integrated.mouse.blood, reduction.use = "umap", no.legend = FALSE, do.return = TRUE, label = TRUE, vector.friendly =
TRUE, pt.size = .4) + theme(plot.title = element_text(hjust = 0.5)) + coord_fixed()

test <- integrated.mouse.blood
test@metadata
FeaturePlot(integrated.mouse.blood,features=c("matching"), reduction = "umap",pt.size = .4, coord.fixed = TRUE,combine=FALSE)

```



```

DimPlot(integrated.mouse.blood , reduction = "umap", group.by = "celltype", label = TRUE,
        repel = TRUE) + NoLegend()

FeaturePlot(samples.integrated,features=c("Sell"), reduction = "umap",cols=c("gray","red"),pt.size = .4, coord.fixed = TRUE,combine=
        FALSE)

DimPlot(object = integrated.mouse.blood, reduction.use = "umap", no.legend = FALSE, do.return = TRUE, label = TRUE, vector.friendly =
        TRUE, pt.size = .4)
+ theme(plot.title = element_text(hjust = 0.5))
+ coord_fixed()

saveRDS(working_soi, file = paste0(sample, ".rds"))
#saveRDS(integrated.k468, file = "/Users/jacoblubner/Desktop/ptime/k468_1.rds")

```

## D.I.6 SHINY APP

```

#
# This is a Shiny web application. You can run the application by clicking
# the 'Run App' button above.
#
# Find out more about building applications with Shiny here:
#
#   http://shiny.rstudio.com/
#

library(shiny)
library(Seurat)
library(ggplot2)
library(cowplot)
library(viridis)
library(mltools)
library(data.table)
library(dplyr)
library(tidyr)

options(shiny.sanitize.errors = FALSE)

ui <- fluidPage(

  # Application title
  sidebarLayout(
    sidebarPanel(
      selectInput("sample", h4("Sample:"),
        c("K409 LN [Human]"="k409.ln-blood.tcr.rds","K409 Tumor [Human]" = "k409.tumor-blood.tcr.rds","K411 LN [Human]"="k411.ln
        -ablood.tcr.rds","K411 LN (matching PD1 high blood) [Human]"="k411.ln-bblood.tcr.rds","K468 LN [Human]"="k468.ln
        -blood.tcr.rds", "Pilot 1 M1 Tumor [Mouse]"="p1m1.tumor-blood.tcr.rds", "K409 Blood (mathing K409 LN) [Human]"="
        k409.blood-ln.tcr.rds","K409 Blood (matching K409 Tumor) [Human]"="k409.blood-tumor.tcr.rds","K411 Blood A [Human
        ]" = "k411.ablood-ln.tcr.rds","K411 Blood B [Human]"="k411.bblood-ln.tcr.rds","K468 Blood [Human]"="k468.blood-ln
        .tcr.rds", "Pilot 1 M1 Blood [Mouse]"="p1m1.blood-tumor.tcr.rds","Pilot 2 M1 Blood [Mouse]"="p2m1.blood-tumor.tcr
        .rds","Pilot 2 M1 Tumor [Mouse]"="p2m1.tumor-blood.tcr.rds","Pilot 2 M2 Blood [Mouse]"="p2m2.blood-tumor.tcr.rds"
        ,"Pilot 2 M2 Tumor [Mouse]"="p2m2.tumor-blood.tcr.rds"))
      #selectInput("tissue", h4("Matching Tissue Selection (SAMPLE_ID MATCHING_TISSUE [ORGANISM] <-> SAMPLE_ID PRIMARY_TISSUE:"),
      # c("K409 Blood [Human] <-> K409 LN"="k409.blood-ln.csv", "K409 LN [Human] <-> K409 Blood"="k409.ln-blood.csv","K409 Tumor
      [Human] <-> K409 Blood" = "k409.tumor-blood.csv","K411 LN [Human] <-> K411 Blood A"="k411.ln-ablood.csv","K411 LN [Human] <->
      K411 Blood B"="k411.ln-bblood.csv","K468 LN [Human] <-> K468 Blood"="k468.ln-blood.csv", "Mouse Pilot 1 Tumor [Mouse] <-> Mouse
      Pilot 1 Blood"="mpl.tumor-blood.csv","K409 Blood [Human] <-> K409 Tumor"="k409.blood-tumor.csv","K411 Blood A [Human] <-> K409
      LN" = "k411.ablood-ln.csv","K411 Blood B [Human] <-> K411 LN"="k411.bblood-ln.csv","K468 Blood [Human] <-> K468 LN"="k468.blood-
      ln.csv", "Mouse Pilot 1 Blood [Mouse] <-> Mouse Pilot 1 Tumor"="mpl.blood-tumor.csv")),
      # Sidebar with a slider input for number of bins

      # Show a plot of the generated distribution
      #selectizeInput("text", c(), selected = "Pdccl1", multiple = TRUE,
      # options = NULL),
      #h4("Show Matching Clones From:"),
      #uiOutput('selectui')
      #div(id="matching")
      #h3("Gene Expression Plot[s]"),
      #h4("Gene[s]:"),
      #div(id="gene")
      #h4("TCR Sequence:"),
      #div(id="tcr")
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Clusters",plotOutput("distPlot2")),
        #tabPanel("Matching Clones",splitLayout(plotOutput("matchingPlot"),plotOutput("matchingPlot2"))),
        tabPanel("Matching Clones",fluidPage(fluidRow(column(6,wellPanel(span(h4("Secondary Tissue:"),uiOutput('selectui')))),
          fluidRow(
            column(6,
              plotOutput(outputId = "matchingPlot")
            ),

```

```

        column(6,
          plotOutput(outputId = "notMatchingPlot")
        )
      ),
      fluidRow(
        column(6,
          plotOutput(outputId = "matchingPlot1")
        ),
        column(6,
          plotOutput(outputId = "notMatchingPlot1")
        )
      ),
      fluidRow(h5("'Primary Tissue' is the sample selected on the left and 'Secondary Tissue' is the sample that
        appears at the top of this page."))
    )),
    tabPanel("Marker Expression", fluidPage( fluidRow(h4("Gene[s]:"), uiOutput('geneui')), fluidRow(plotOutput("distPlot")) )),
    tabPanel("Clonal Expansion Frequency", plotOutput("distPlot3")),
    tabPanel("Number of Genes", plotOutput("distPlot4")),
    #tabPanel("Clones", plotOutput("distPlot5"))
    tabPanel("Clones", fluidPage(fluidRow(h4("TCR Sequence (Decreasing by Clone Size):"), uiOutput('cloneui')), fluidRow(plotOutput("
      distPlot5"))) ) )
  )
  #textInput("text", label = h3("Gene"), value = "PDCD1"#,
  #sliderInput("bins",
  #  "Number of bins:",
  #    min = 1,
  #    max = 50,
  #    value = 30)
  )
)
)

# Define server logic required to draw a histogram
selector2 <- list("k409.blood-ln.csv"="K409 Blood [Human]")

server <- function(input, output) {
  k409ln <- c("K409 Blood [Human]"="k409.blood-ln.csv")
  k409tumor <- c("K409 Blood [Human]"="k409.blood-tumor.csv")
  k409blood <- c("K409 Tumor [Human]" = "k409.tumor-blood.csv")
  k409bloodl <- c("K409 LN [Human]"="k409.ln-blood.csv")
  k411ln <- c("K411 Blood A [Human]" = "k411.ablood-ln.csv")
  k411lnhigh <- c("K411 Blood B [Human]" = "k411.bblood-ln.csv")
  k411ablood <- c("K411 LN [Human]" = "k411.ln-ablood.csv")
  k411bblood <- c("K411 LN (matching PD1 high blood) [Human]" = "k411.ln-bblood.csv")
  k468ln <- c("K468 Blood [Human]" = "k468.blood-ln.csv")
  k468blood <- c("K468 LN [Human]" = "k468.ln-blood.csv")
  mp1tumor <- c("Pilot 1 M1 Blood [Mouse]" = "mp1.blood-tumor.csv")
  mp1blood <- c("Pilot 1 M1 Tumor [Mouse]" = "mp1.tumor-blood.csv")
  p2m1tumor <- c("Pilot 2 M1 Blood [Mouse]" = "p2m1.blood-tumor.csv")
  p2m1blood <- c("Pilot 2 M1 Tumor [Mouse]" = "p2m1.tumor-blood.csv")
  p2m2tumor <- c("Pilot 2 M2 Blood [Mouse]" = "p2m2.blood-tumor.csv")
  p2m2blood <- c("Pilot 2 M2 Tumor [Mouse]" = "p2m2.tumor-blood.csv")
  k471tumor <- c("K471 Blood [Human]"="k471.blood-tumor.csv")
  k471blood <- c("K471 Tumor [Human]"="k471.tumor-blood.csv")

  #selectddInput("sample", h4("Primary Tissue Selection (SAMPLE_ID PRIMARY_TISSUE [ORGANISM] <-> SAMPLE_ID MATCHING_TISSUE):"),
  #  c("K409 LN [Human] <-> K409 Blood"="k409.ln-blood.tcr.rds", "K409 Tumor [Human] <-> K409 Blood" = "k409.tumor-blood.tcr.
  #    rds", "K411 LN [Human] <-> K411 Blood A"="k411.ln-ablood.tcr.rds", "K411 LN [Human] <-> K411 Blood B"="k411.ln-bblood.tcr.rds", "
  #    K468 LN [Human] <-> K468 Blood"="k468.ln-blood.tcr.rds", "Mouse Pilot 1 Tumor [Mouse] <-> Mouse Pilot 1 Blood"="mp1.tumor-blood.
  #    tcr.rds", "K409 Blood [Human] <-> K409 LN"="k409.blood-ln.tcr.rds", "K409 Tumor [Human] <-> K409 Tumor"="k409.blood-tumor.tcr.rds
  #    ", "K411 Blood A [Human] <-> K409 LN" = "k411.ablood-ln.tcr.rds", "K411 Blood B [Human] <-> K411 LN"="k411.bblood-ln.tcr.rds", "
  #    K468 Blood [Human] <-> K468 LN"="k468.blood-ln.tcr.rds", "Mouse Pilot 1 Blood [Mouse] <-> Mouse Pilot 1 Tumor"="mp1.blood-tumor.
  #    tcr.rds"))
  selector <- list("k409.ln-blood.tcr.rds"=k409ln, "k409.tumor-blood.tcr.rds"=k409tumor, "k411.ln-ablood.tcr.rds"=k411ln, "k411.ln-bblood.
    tcr.rds"=k411lnhigh, "k468.ln-blood.tcr.rds"=k468ln, "p1m1.tumor-blood.tcr.rds"=mp1tumor, "k409.blood-ln.tcr.rds"=k409blood, "k409.
    blood-tumor.tcr.rds"=k409blood, "k411.ablood-ln.tcr.rds"=k411ablood, "k411.bblood-ln.tcr.rds"=k411bblood, "k468.blood-ln.tcr.rds"=
    k468blood, "p1m1.blood-tumor.tcr.rds"=mp1blood, "p2m1.blood-tumor.tcr.rds"=p2m1blood, "p2m1.tumor-blood.tcr.rds"=p2m1tumor, "p2m2.
    tumor-blood.tcr.rds"=p2m2tumor, "p2m2.blood-tumor.tcr.rds"=p2m2blood, "k471.tumor-blood.tcr.rds"=k471tumor, "k471.blood-tumor.tcr.
    rds"=k471blood)

  tcr_regex <- function(sof){
    sof <- sub('[^([|])', "TRA:\\1", sof)
    sof <- sub('[^\\|]', "TRB:", sof)
    sof <- sub('\\|', "TRB:", sof)
    sof <- sub('.*notcr', 'notcr', sof)
    return(sof)
  }

  tcr_regex_reverse <-function(sof){
    sof <- sub('^TRB:', '|', sof)
    sof <- sub('^TRA:', '|', sof)
    sof <- sub('|', 'TRB:', '|', sof)
    return(sof)
  }

  tcr_conv <- reactive({
    output <- lapply(sam_ordered_freq()$TCR, tcr_regex)
  })

  output$cloneui = renderUI ({
    selectizeInput("tcr", label = NULL, tcr_conv(), selected=1, multiple=FALSE, options=NULL)
  })
}

```

```

output$geneui = renderUI ({
  selectizeInput("text", label = NULL, rownames(x = sam()),selected = c(rownames(x = sam())[idx()],rownames(x = sam())[idx()-1],
    rownames(x = sam())[idx()-2],rownames(x = sam())[idx()-3]), multiple = TRUE,options = NULL)
})

output$selectui = renderUI ({
  selectInput("tissue", label = NULL, sublist(),multiple=FALSE,selected=1)
})

#output$matchingt = renderText ({
#  sublist()[1]
#})

sublist <- reactive({
  object = selector[[input$sample]]
})

sam <- reactive({
  seurat_object = readRDS(input$sample)
})

sam_ordered_freq <- reactive({
  sof <- sam()@meta.data[order(sam()@meta.data$Frequency,decreasing=TRUE),]
})

sam2 <- reactive({
  seurat_obeject = read.csv(input$tissue)
})

idx <- reactive({
  rand_num = sample(5:length(rownames(sam())),1)
})

md_oh <- reactive({
  output = one_hot(data.table(sam()@meta.data),cols=c("TCR"))
})

sam_matching <- reactive({
  output = sam()@meta.data %>% filter(matching == "both_matching" | matching == "matching" | matching == "beta_matching" |
    matching == "alpha_matching" )
})

sam_notmatching <- reactive({
  output = sam()@meta.data %>% filter(matching != "both_matching" & matching != "matching" & matching != "beta_matching" &
    matching != "alpha_matching")
})

sam2_matching <- reactive({
  output = sam2() %>% filter(matching == "matching" | matching == "both_matching" | matching == "beta_matching" | matching == "
    alpha_matching")
})

sam2_notmatching <- reactive({
  output = sam2() %>% filter(matching != "both_matching" & matching != "matching" & matching != "beta_matching" & matching != "
    alpha_matching")
})

u1max <- reactive({
  output = max(sam()@meta.data$u1)
})

u1min <- reactive({
  output = min(sam()@meta.data$u1)
})

u2max <- reactive({
  output = max(sam()@meta.data$u2)
})

u2min <- reactive({
  output = min(sam()@meta.data$u2)
})

u1max2 <- reactive({
  output = max(sam2()$u1)
})

u1min2 <- reactive({
  output = min(sam2()$u1)
})

u2max2 <- reactive({
  output = max(sam2()$u2)
})

u2min2 <- reactive({
  output = min(sam2()$u2)
})

output$matchingPlot <- renderPlot({
  ggplot(sam_matching(), aes(x=u1, y=u2))+
    geom_point(aes(color=freq),size=1)+
    xlab("UMAP 1")+
    ylab("UMAP 2")+
    ggtitle("Expansion Plot Of Matching Cells For Primary Tissue")+
    coord_fixed()+
    labs(color='Frequency')+
    scale_colour_gradient(low="gray",high="purple", na.value="black")+
    labs(subtitle = "(black indicates no expansion)")+
    xlim(u1min(),u1max())+
    ylim(u2min(),u2max())
})

```

```

output$matchingPlot1 <- renderPlot({
  ggplot(sam2_matching(), aes(x=u1, y=u2))+
    geom_point(aes(color=freq),size=1)+
    xlab("UMAP 1")+
    ylab("UMAP 2")+
    ggtitle("Expansion Plot Of Matching Cells For Secondary Tissue")+
    coord_fixed()+
    labs(color='Frequency')+
    scale_colour_gradient(low="gray",high="purple", na.value="black")+
    labs(subtitle = "(black indicates no expansion)")+
    xlim(u1min(),u1max2())+
    ylim(u2min2(),u2max2())
})

output$notMatchingPlot <- renderPlot ({
  ggplot(sam2_notmatching(), aes(x=u1, y=u2))+
    geom_point(aes(color=freq),size=1)+
    xlab("UMAP 1")+
    ylab("UMAP 2")+
    ggtitle("Expansion Plot Of Not Matching Cells For Primary Tissue")+
    coord_fixed()+
    labs(color='Frequency')+
    scale_colour_gradient(low="gray",high="purple", na.value="black")+
    labs(subtitle = "(black indicates no expansion)")+
    xlim(u1min(),u1max2())+
    ylim(u2min(),u2max2())
})

output$notMatchingPlot1 <- renderPlot ({
  ggplot(sam2_notmatching(), aes(x=u1, y=u2))+
    geom_point(aes(color=freq),size=1)+
    xlab("UMAP 1")+
    ylab("UMAP 2")+
    ggtitle("Expansion Plot Of Not Matching Cells For Secondary Tissue")+
    coord_fixed()+
    labs(color='Frequency')+
    scale_colour_gradient(low="gray",high="purple", na.value="black")+
    labs(subtitle = "(black indicates no expansion)")+
    xlim(u1min(),u1max2())+
    ylim(u2min2(),u2max2())
})

output$distPlot5 <- renderPlot({
  ggplot(md_oh(), aes(x=u1, y=u2))+
    geom_point(aes(color=as.factor(md_oh()[[paste0("TCR_", tcr_regex_reverse(input$trc))]])),size=1)+
    xlab("UMAP 1")+
    ylab("UMAP 2")+
    ggtitle(paste0("Clone ", input$trc, " (", toString(sum(md_oh()[[paste0("TCR_", tcr_regex_reverse(input$trc))]] == 1)), "
    Cells"))+
    scale_color_manual(breaks = c("0", "1"), values=c("black", "red"))+
    theme(legend.position="none")+
    coord_fixed()
},height=800,width=800)

output$distPlot <- renderPlot({
  FeaturePlot(sam(), features=c(input$text), reduction = "umap", cols=c("gray","red"), pt.size = 1, coord.fixed=T)
},height=800,width=800)

output$distPlot2 <- renderPlot({
  DimPlot(object = sam(), reduction.use = "umap", no.legend = FALSE, do.return = TRUE, vector.friendly = TRUE, pt.size = 1, label=
  FALSE, label.size=12, repel=TRUE) + ggtitle("UMAP") + theme(plot.title = element_text(hjust = 0.5))+ggtitle("Clusters")+
  coord_fixed()
},height=800,width=800)

output$distPlot3 <- renderPlot({
  FeaturePlot(sam(), features=c("freq"), pt.size = 1, reduction = "umap", coord.fixed=T, cols=c("gray","purple"))+scale_colour_gradient
  (low="gray",high="purple", na.value="black")+labs(title = "Clonal Expansion", subtitle = "(black indicates no expansion)")
},height=800,width=800)

output$distPlot4 <- renderPlot({
  FeaturePlot(sam(), features=c("nFeature_RNA"), pt.size = 1, reduction = "umap", coord.fixed=T, cols=c("gray","red"))+ggtitle("Number
  of Genes")
},height=800,width=800)
}

# Run the application
shinyApp(ui = ui, server = server)

```

# References

- Analytics, R. & Weston, S. (2013). doSNOW: Foreach parallel adaptor for the snow package. *URL <http://CRAN.R-project.org/package=doSNOW>. R package version, 1(9).*
- Andrzejak, A., Kondo, D., & Yi, S. (2010). Decision model for cloud computing under SLA constraints. In *2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems* (pp. 257–266): [ieeexplore.ieee.org](http://ieeexplore.ieee.org).
- Araghizadeh, F., Abdelnaby, A. (2012). Anatomy and physiology. In Bailey, H. R., Billingham, R. P., Stamos, M. J. & Snyder, M. J (Ed.), *Colorectal Surgery* (pp. 3–17). Elsevier Health Sciences.
- Auguie, B. & Antonov, A. (2017). gridextra: miscellaneous functions for “grid” graphics. *R package version, 2(601), 602.*
- Bach, J.-F. (2018). The hygiene hypothesis in autoimmunity: the role of pathogens and commensals. *Nat. Rev. Immunol.*, 18(2), 105–120.
- Bäckhed, F., Roswall, J., Peng, Y., Feng, Q., Jia, H., Kovatcheva-Datchary, P., Li, Y., Xia, Y., Xie, H., Zhong, H., Khan, M. T., Zhang, J., Li, J., Xiao, L., Al-Aama, J., Zhang, D., Lee, Y. S., Kotowska, D., Colding, C., Tremaroli, V., Yin, Y., Bergman, S., Xu, X., Madsen, L., Kristiansen, K., Dahlgren, J., & Wang, J. (2015). Dynamics and stabilization of the human gut microbiome during the first year of life. *Cell Host Microbe*, 17(6), 852.
- Beaulieu-Jones, B. K. & Greene, C. S. (2017). Reproducibility of computational workflows is automated using continuous analysis. *Nat. Biotechnol.*, 35(4), 342–346.
- Beeley, C. (2016). *Web Application Development with R Using Shiny*. Packt Publishing Ltd.
- Bik, E. M., Casadevall, A., & Fang, F. C. (2016). The prevalence of inappropriate image duplication in biomedical research publications. *MBio*, 7(3).
- Bolker, B. M. (2012). The coefplot2 package.

- Butler, A., Hoffman, P., Smibert, P., Papalexi, E., & Satija, R. (2018). Integrating single-cell transcriptomic data across different conditions, technologies, and species. *Nat. Biotechnol.*, 36(5), 411–420.
- Callahan, B. J., McMurdie, P. J., Rosen, M. J., Han, A. W., Johnson, A. J. A., & Holmes, S. P. (2016). DADA2: High-resolution sample inference from illumina amplicon data. *Nat. Methods*, 13(7), 581–583.
- Calvo, S. E., Clauser, K. R., & Mootha, V. K. (2016). MitoCarta2.0: an updated inventory of mammalian mitochondrial proteins. *Nucleic Acids Res.*, 44(D1), D1251–7.
- Caspi, R., Altman, T., Dale, J. M., Dreher, K., Fulcher, C. A., Gilham, F., Kaipa, P., Karthikeyan, A. S., Kothari, A., Krummenacker, M., Latendresse, M., Mueller, L. A., Paley, S., Popescu, L., Pujar, A., Shearer, A. G., Zhang, P., & Karp, P. D. (2010). The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Res.*, 38(Database issue), D473–9.
- Chambers, E. S., Byrne, C. S., Aspey, K., Chen, Y., Khan, S., Morrison, D. J., & Frost, G. (2018). Acute oral sodium propionate supplementation raises resting energy expenditure and lipid oxidation in fasted humans. *Diabetes Obes. Metab.*, 20(4), 1034–1039.
- Chihara, N., Madi, A., Kondo, T., Zhang, H., Acharya, N., Singer, M., Nyman, J., Marjanovic, N. D., Kowalczyk, M. S., Wang, C., Kurtulus, S., Law, T., Etminan, Y., Nevin, J., Buckley, C. D., Burkett, P. R., Buenrostro, J. D., Rozenblatt-Rosen, O., Anderson, A. C., Regev, A., & Kuchroo, V. K. (2018). Induction and transcriptional regulation of the co-inhibitory gene module in T cells. *Nature*, 558(7710), 454–459.
- Cimermancic, P., Medema, M. H., Claesen, J., Kurita, K., Wieland Brown, L. C., Mavrommatis, K., Pati, A., Godfrey, P. A., Koehrsen, M., Clardy, J., Birren, B. W., Takano, E., Sali, A., Lington, R. G., & Fischbach, M. A. (2014). Insights into secondary metabolism from a global analysis of prokaryotic biosynthetic gene clusters. *Cell*, 158(2), 412–421.
- Clarke, S. F., Murphy, E. F., O’Sullivan, O., Lucey, A. J., Humphreys, M., Hogan, A., Hayes, P., O’Reilly, M., Jeffery, I. B., Wood-Martin, R., Kerins, D. M., Quigley, E., Ross, R. P., O’Toole, P. W., Molloy, M. G., Falvey, E., Shanahan, F., & Cotter, P. D. (2014). Exercise and associated dietary extremes impact on gut microbial diversity. *Gut*, 63(12), 1913–1920.

- Courtois, S., Cappellano, C. M., Ball, M., Francou, F.-X., Normand, P., Helynck, G., Martinez, A., Kolvek, S. J., Hopke, J., Osburne, M. S., August, P. R., Nalin, R., Guérineau, M., Jeannin, P., Simonet, P., & Pernodet, J.-L. (2003). Recombinant environmental libraries provide access to microbial diversity for drug discovery from natural products. *Appl. Environ. Microbiol.*, 69(1), 49–55.
- Cullen, T. W., Schofield, W. B., Barry, N. A., Putnam, E. E., Rundell, E. A., Trent, M. S., Degnan, P. H., Booth, C. J., Yu, H., & Goodman, A. L. (2015). Gut microbiota. antimicrobial peptide resistance mediates resilience of prominent gut commensals during inflammation. *Science*, 347(6218), 170–175.
- Daud, A. I., Loo, K., Pauli, M. L., Sanchez-Rodriguez, R., Sandoval, P. M., Taravati, K., Tsai, K., Nosrati, A., Nardo, L., Alvarado, M. D., Algazi, A. P., Pampaloni, M. H., Lobach, I. V., Hwang, J., Pierce, R. H., Gratz, I. K., Krummel, M. F., & Rosenblum, M. D. (2016). Tumor immune profiling predicts response to anti-PD-1 therapy in human melanoma. *J. Clin. Invest.*, 126(9), 3447–3452.
- Dawson, W., Hör, J., Egert, M., van Kleunen, M., & Pester, M. (2017). A small number of low-abundance bacteria dominate plant species-specific responses during rhizosphere colonization. *Front. Microbiol.*, 8, 975.
- De Meo, P., Ferrara, E., Fiumara, G., & Provetti, A. (2011). Generalized louvain method for community detection in large networks. In *2011 11th International Conference on Intelligent Systems Design and Applications* (pp. 88–93): [ieeexplore.ieee.org](http://ieeexplore.ieee.org).
- Dekker, J., Belmont, A. S., Guttman, M., Leshyk, V. O., Lis, J. T., Lomvardas, S., Mirny, L. A., O'Shea, C. C., Park, P. J., Ren, B., Politz, J. C. R., Shendure, J., Zhong, S., & 4D Nucleome Network (2017). The 4D nucleome project. *Nature*, 549(7671), 219–226.
- Delaney, C., Schnell, A., Cammarata, L. V., Yao-Smith, A., Regev, A., Kuchroo, V. K., & Singer, M. (2019). Combinatorial prediction of marker panels from single-cell transcriptomic data.
- Dendrou, C. A., Petersen, J., Rossjohn, J., & Fugger, L. (2018). HLA variation and disease. *Nat. Rev. Immunol.*, 18(5), 325–339.
- Devlin, A. S. & Fischbach, M. A. (2015). A biosynthetic pathway for a prominent class of microbiota-derived bile acids. *Nat. Chem. Biol.*, 11(9), 685–690.

- Donia, M. S., Cimermanic, P., Schulze, C. J., Wieland Brown, L. C., Martin, J., Mitreva, M., Clardy, J., Lington, R. G., & Fischbach, M. A. (2014). A systematic analysis of biosynthetic gene clusters in the human microbiome reveals a common family of antibiotics. *Cell*, 158(6), 1402–1414.
- Douglas, G. M., Beiko, R. G., & Langille, M. G. I. (2018). Predicting the functional potential of the microbiome from marker genes using PICRUSt. *Methods Mol. Biol.*, 1849, 169–177.
- Dowle, M., Srinivasan, A., Gorecki, J., Chirico, M., Stetsenko, P., Short, T., Lianoglou, S., Antonyan, E., Bonsch, M., Parsonage, H., & Others (2019). Package ‘data.table’. *Extension of ‘data.frame’*.
- Dusko Ehrlich, S. & The MetaHIT Consortium (2011). MetaHIT: The european union project on metagenomics of the human intestinal tract. In *Metagenomics of the Human Body* (pp. 307–316). Springer, New York, NY.
- Esmailzadeh, H., Blem, E., Amant, R. S., Sankaralingam, K., & Burger, D. (2013). Power challenges may end the multicore era. *Commun. ACM*, 56(2), 93–102.
- Fang, C., Zhong, H., Lin, Y., Chen, B., Han, M., Ren, H., Lu, H., Lubber, J. M., Xia, M., Li, W., Stein, S., Xu, X., Zhang, W., Drmanac, R., Wang, J., Yang, H., Hammarström, L., Kostic, A. D., Kristiansen, K., & Li, J. (2017). Assessment of the cPAS-based BGISEQ-500 platform for metagenomic sequencing. *Gigascience*.
- Franzosa, E. A., McIver, L. J., Rahnavard, G., Thompson, L. R., Schirmer, M., Weingart, G., Lipson, K. S., Knight, R., Caporaso, J. G., Segata, N., & Huttenhower, C. (2018). Species-level functional profiling of metagenomes and metatranscriptomes. *Nat. Methods*, 15(11), 962–968.
- Freytag, S., Tian, L., Lönstedt, I., Ng, M., & Bahlo, M. (2018). Comparison of clustering tools in R for medium-sized 10x genomics single-cell RNA-sequencing data. *F1000Res*, 7, 1297.
- Fu, L., Niu, B., Zhu, Z., Wu, S., & Li, W. (2012). CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23), 3150–3152.
- Fujisaka, S., Avila-Pacheco, J., Soto, M., Kostic, A., Dreyfuss, J. M., Pan, H., Ussar, S., Altindis, E., Li, N., Bry, L., Clish, C. B., & Ronald Kahn, C. (2018). Diet, genetics, and the gut microbiome drive dynamic changes in plasma metabolites.
- Garg, S. K., Yeo, C. S., Anandasivam, A., & Buyya, R. (2011). Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. *J. Parallel Distrib. Comput.*, 71(6), 732–749.



- Garvie, E. I. (1980). Bacterial lactate dehydrogenases. *Microbiol. Rev.*, 44(1), 106–139.
- Giladi, A. & Amit, I. (2018). Single-Cell genomics: A stepping stone for future immunology discoveries. *Cell*, 172(1-2), 14–21.
- Gilbert, J. A., Blaser, M. J., Caporaso, J. G., Jansson, J. K., Lynch, S. V., & Knight, R. (2018). Current understanding of the human microbiome. *Nat. Med.*, 24(4), 392–400.
- Glanville, J., Huang, H., Nau, A., Hatton, O., Wagar, L. E., Rubelt, F., Ji, X., Han, A., Krams, S. M., Pettus, C., Haas, N., Arlehamn, C. S. L., Sette, A., Boyd, S. D., Scriba, T. J., Martinez, O. M., & Davis, M. M. (2017). Identifying specificity groups in the T cell receptor repertoire. *Nature*, 547(7661), 94–98.
- Guo, C.-J., Chang, F.-Y., Wyche, T. P., Backus, K. M., Acker, T. M., Funabashi, M., Taketani, M., Donia, M. S., Nayfach, S., Pollard, K. S., Craik, C. S., Cravatt, B. F., Clardy, J., Voigt, C. A., & Fischbach, M. A. (2017). Discovery of reactive Microbiota-Derived metabolites that inhibit host proteases. *Cell*, 168(3), 517–526.e18.
- Hadjithomas, M., Chen, I.-M. A., Chu, K., Ratner, A., Palaniappan, K., Szeto, E., Huang, J., Reddy, T. B. K., Cimermančič, P., Fischbach, M. A., Ivanova, N. N., Markowitz, V. M., Kyrpides, N. C., & Pati, A. (2015). IMG-ABC: A knowledge base to fuel discovery of biosynthetic gene clusters and novel secondary metabolites. *MBio*, 6(4), e00932.
- Hafemeister, C. & Satija, R. (2019). Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. *Genome Biol.*, 20(1), 296.
- Hausmann, B., Knorr, K.-H., Schreck, K., Tringe, S. G., Glavina Del Rio, T., Loy, A., & Pester, M. (2016). Consortia of low-abundance bacteria drive sulfate reduction-dependent degradation of fermentation products in peat soil microcosms. *ISME J.*, 10(10), 2365–2375.
- Heidrich, C., Pag, U., Josten, M., Metzger, J., Jack, R. W., Bierbaum, G., Jung, G., & Sahl, H. G. (1998). Isolation, characterization, and heterologous expression of the novel lantibiotic epicidin 280 and analysis of its biosynthetic gene cluster. *Appl. Environ. Microbiol.*, 64(9), 3140–3146.
- Hindson, B., Hindson, C., Schnell-Levin, M., Ness, K., Jarosz, M., Masquelier, D., Saxonov, S., Merrill, L., Price, A., Hardenbol, P., & Li, Y. (2014). Compositions and methods for sample processing.

- Huang, H., Wang, C., Rubelt, F., Scriba, T. J., & Davis, M. M. (2020). Analyzing the mycobacterium tuberculosis immune response by t-cell receptor clustering with GLIPH2 and genome-wide antigen screening. *Nat. Biotechnol.*
- Human Microbiome Project Consortium (2012). Structure, function and diversity of the healthy human microbiome. *Nature*, 486(7402), 207–214.
- Jayasimhan, A., Mansour, K. P., & Slattery, R. M. (2014). Advances in our understanding of the pathophysiology of type 1 diabetes: lessons from the NOD mouse. *Clin. Sci.*, 126(1), 1–18.
- Jones, E., Oliphant, T., & Peterson, P. (2001). {SciPy}: Open source scientific tools for {Python}.
- Jurkowski, J. E., Jones, N. L., Toews, C. J., & Sutton, J. R. (1981). Effects of menstrual cycle on blood lactate, O<sub>2</sub> delivery, and performance during exercise. *J. Appl. Physiol.*, 51(6), 1493–1499.
- Kassambara, A. (2018). ggpubr: “ggplot2” based publication ready plots. *R package version 0.1.7*.
- Keir, M. E., Butte, M. J., Freeman, G. J., & Sharpe, A. H. (2008). PD-1 and its ligands in tolerance and immunity. *Annu. Rev. Immunol.*, 26, 677–704.
- Kerpedjiev, P., Abdennur, N., Lekschas, F., McCallum, C., Dinkla, K., Strobelt, H., Lubner, J. M., Ouellette, S. B., Azhir, A., Kumar, N., Hwang, J., Lee, S., Alver, B. H., Pfister, H., Mirny, L. A., Park, P. J., & Gehlenborg, N. (2018). HiGlass: web-based visual exploration and analysis of genome interaction maps. *Genome Biol.*, 19(1), 125.
- Kimura, I., Inoue, D., Maeda, T., Hara, T., Ichimura, A., Miyauchi, S., Kobayashi, M., Hirasawa, A., & Tsujimoto, G. (2011). Short-chain fatty acids and ketones directly regulate sympathetic nervous system via G protein-coupled receptor 41 (GPR41). *Proc. Natl. Acad. Sci. U. S. A.*, 108(19), 8030–8035.
- Kolde, R. (2012). Pheatmap: pretty heatmaps. *R package version, 61*.
- Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., & others (2017). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *bioRxiv*.
- Kostic, A. D., Gevers, D., Siljander, H., Vatanen, T., Hyötyläinen, T., Hämäläinen, A.-M., Peet, A., Tillmann, V., Pöhö, P., Mattila, I., Lähdesmäki, H., Franzosa, E. A., Vaarala, O., de Goffau, M., Harmsen, H., Ilonen, J., Virtanen, S. M., Clish, C. B., Orešič, M., Huttenhower, C., Knip, M., & Xavier, R. J. (2016). The dynamics of the human infant gut microbiome in development and in progression toward type 1 diabetes. *Cell Host Microbe*, 20(1), 121.

- Kurm, V., van der Putten, W. H., de Boer, W., Naus-Wiezer, S., & Hol, W. H. G. (2017). Low abundant soil bacteria can be metabolically versatile and fast growing. *Ecology*, 98(2), 555–564.
- Langille, M. G. I., Zaneveld, J., Caporaso, J. G., & others (2013). Predictive functional profiling of microbial communities using 16S rRNA marker gene sequences. *Nature*.
- Langmead, B. & Salzberg, S. L. (2012). Fast gapped-read alignment with bowtie 2. *Nat. Methods*, 9(4), 357–359.
- Lax, S., Smith, D. P., Hampton-Marcell, J., Owens, S. M., Handley, K. M., Scott, N. M., Gibbons, S. M., Larsen, P., Shogan, B. D., Weiss, S., Metcalf, J. L., Ursell, L. K., Vázquez-Baeza, Y., Van Treuren, W., Hasan, N. A., Gibson, M. K., Colwell, R., Dantas, G., Knight, R., & Gilbert, J. A. (2014). Longitudinal analysis of microbial interaction between humans and the indoor environment. *Science*, 345(6200), 1048–1052.
- Le Goallec, A., Tierney, B. T., Lubber, J. M., Cofer, E. M., Kostic, A. D., & Patel, C. J. (2020). A systematic machine learning and data type comparison yields metagenomic predictors of infant age, sex, breastfeeding, antibiotic usage, country of origin, and delivery type. *PLoS Comput. Biol.*, 16(5), e1007895.
- Lemon, J., Bolker, B., Oom, S., Klein, E., Rowlingson, B., Wickham, H., Tyagi, A., Eterradossi, O., Grothendieck, G., Toews, M., & Others (2007). plotrix: Various plotting functions. URL <http://cran.r-project.org/src/contrib/Descriptions/plotrix.html>.
- Li, D., Liu, C.-M., Luo, R., Sadakane, K., & Lam, T.-W. (2015). MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de bruijn graph. *Bioinformatics*, 31(10), 1674–1676.
- Li, H. & Durbin, R. (2009). Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14), 1754–1760.
- Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., & 1000 Genome Project Data Processing Subgroup (2009). The sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16), 2078–2079.
- Li, J., Jia, H., Cai, X., Zhong, H., Feng, Q., Sunagawa, S., Arumugam, M., Kultima, J. R., Prifti, E., Nielsen, T., Juncker, A. S., Manichanh, C., Chen, B., Zhang, W., Levenez, F., Wang, J., Xu, X., Xiao, L., Liang, S., Zhang, D., Zhang, Z., Chen, W., Zhao, H., Al-Aama, J. Y., Edris, S., Yang, H.,

- Wang, J., Hansen, T., Nielsen, H. B., Brunak, S., Kristiansen, K., Guarner, F., Pedersen, O., Doré, J., Ehrlich, S. D., MetaHIT Consortium, Bork, P., Wang, J., & MetaHIT Consortium (2014). An integrated catalog of reference genes in the human gut microbiome. *Nat. Biotechnol.*, 32(8), 834–841.
- Li, J. W.-H. & Vederas, J. C. (2009). Drug discovery and natural products: end of an era or an endless frontier? *Science*, 325(5937), 161–165.
- Lloyd-Price, J., Abu-Ali, G., & Huttenhower, C. (2016). The healthy human microbiome. *Genome Med.*, 8(1), 51.
- Lloyd-Price, J., Mahurkar, A., Rahnavard, G., Crabtree, J., Orvis, J., Hall, A. B., Brady, A., Creasy, H. H., McCracken, C., Giglio, M. G., McDonald, D., Franzosa, E. A., Knight, R., White, O., & Huttenhower, C. (2017). Strains, functions and dynamics in the expanded human microbiome project. *Nature*, 550(7674), 61–66.
- Lubelski, J., Rink, R., Khusainov, R., Moll, G. N., & Kuipers, O. P. (2008). Biosynthesis, immunity, regulation, mode of action and engineering of the model lantibiotic nisin. *Cell. Mol. Life Sci.*, 65(3), 455–476.
- Luber, J. (2015). Identifying genome signatures of drug response in patient derived xenografts (PDX): a machine learning approach.
- Luber, J. M. (2016). Improved prediction of mouse pathways related to bone maintenance through machine learning utilizing diverse genomic data.
- Luber, J. M. & Kostic, A. D. (2017). Gut microbiota: Small molecules modulate host cellular functions. *Curr. Biol.*, 27(8), R307–R310.
- Luber, J. M. & Kostic, A. D. (2019). A perfect storm: Genetics and antimicrobial antibodies shore up type 1 diabetes. *Sci Immunol*, 4(32).
- Luber, J. M., Tierney, B. T., Cofer, E. M., Patel, C. J., & Kostic, A. D. (2017). Aether: Leveraging linear programming for optimal cloud computing in genomics. *Bioinformatics*.
- Magnúsdóttir, S., Heinken, A., Kutt, L., Ravcheev, D. A., Bauer, E., Noronha, A., Greenhalgh, K., Jäger, C., Baginska, J., Wilmes, P., Fleming, R. M. T., & Thiele, I. (2017). Generation of genome-scale metabolic reconstructions for 773 members of the human gut microbiota. *Nat. Biotechnol.*, 35(1), 81–89.

- Mahuron, K. M., Moreau, J. M., Glasgow, J. E., Boda, D. P., Pauli, M. L., Gouirand, V., Panjabi, L., Grewal, R., Lubber, J. M., Mathur, A. N., Feldman, R. M., Shifrut, E., Mehta, P., Lowe, M. M., Alvarado, M. D., Marson, A., Singer, M., Wells, J., Jupp, R., Daud, A. I., & Rosenblum, M. D. (2020). Layilin augments integrin activation to promote antitumor immunity. *J. Exp. Med.*, 217(9).
- Mallick, H., Franzosa, E. A., McIver, L. J., Banerjee, S., Sirota-Madi, A., Kostic, A. D., Clish, C. B., Vlamakis, H., Xavier, R. J., & Huttenhower, C. (2019). Predictive metabolomic profiling of microbial communities using amplicon or metagenomic sequences. *Nat. Commun.*, 10(1), 3136.
- McIver, L. J., Abu-Ali, G., Franzosa, E. A., Schwager, R., Morgan, X. C., Waldron, L., Segata, N., & Huttenhower, C. (2018). biobakery: a meta'omic analysis environment. *Bioinformatics*, 34(7), 1235–1237.
- McMurdie, P. J. & Holmes, S. (2013). phyloseq: an R package for reproducible interactive analysis and graphics of microbiome census data. *PLoS One*, 8(4), e61217.
- Medema, M. H., Breitling, R., Bovenberg, R., & Takano, E. (2011). Exploiting plug-and-play synthetic biology for drug discovery and production in microorganisms. *Nat. Rev. Microbiol.*, 9(2), 131–137.
- Medema, M. H., Kottmann, R., Yilmaz, P., Cummings, M., Biggins, J. B., Blin, K., de Bruijn, I., Chooi, Y. H., Claesen, J., Coates, R. C., Cruz-Morales, P., Duddela, S., Düsterhus, S., Edwards, D. J., Fewer, D. P., Garg, N., Geiger, C., Gomez-Escribano, J. P., Greule, A., Hadjithomas, M., Haines, A. S., Helfrich, E. J. N., Hillwig, M. L., Ishida, K., Jones, A. C., Jones, C. S., Jungmann, K., Kegler, C., Kim, H. U., Kötter, P., Krug, D., Masschelein, J., Melnik, A. V., Mantovani, S. M., Monroe, E. A., Moore, M., Moss, N., Nützman, H.-W., Pan, G., Pati, A., Petras, D., Reen, F. J., Rosconi, F., Rui, Z., Tian, Z., Tobias, N. J., Tsunematsu, Y., Wiemann, P., Wyckoff, E., Yan, X., Yim, G., Yu, F., Xie, Y., Aigle, B., Apel, A. K., Balibar, C. J., Balskus, E. P., Barona-Gómez, F., Bechthold, A., Bode, H. B., Borriss, R., Brady, S. F., Brakhage, A. A., Caffrey, P., Cheng, Y.-Q., Clardy, J., Cox, R. J., De Mot, R., Donadio, S., Donia, M. S., van der Donk, W. A., Dorrestein, P. C., Doyle, S., Driessen, A. J. M., Ehling-Schulz, M., Entian, K.-D., Fischbach, M. A., Gerwick, L., Gerwick, W. H., Gross, H., Gust, B., Hertweck, C., Höfte, M., Jensen, S. E., Ju, J., Katz, L., Kaysser, L., Klassen, J. L., Keller, N. P., Kormanec, J., Kuipers, O. P., Kuzuyama, T., Kyrpides, N. C., Kwon, H.-J., Lautru, S., Lavigne, R., Lee, C. Y., Linqun, B., Liu, X., Liu, W., Luzhetskyy, A., Mahmud, T., Mast, Y., Méndez, C., Metsä-Ketelä, M., Micklefield, J., Mitchell, D. A.,

Moore, B. S., Moreira, L. M., Müller, R., Neilan, B. A., Nett, M., Nielsen, J., O’Gara, F., Oikawa, H., Osbourn, A., Osburne, M. S., Ostash, B., Payne, S. M., Pernodet, J.-L., Petricek, M., Piel, J., Ploux, O., Raaijmakers, J. M., Salas, J. A., Schmitt, E. K., Scott, B., Seipke, R. F., Shen, B., Sherman, D. H., Sivonen, K., Smanski, M. J., Sosio, M., Stegmann, E., Süßmuth, R. D., Tahlan, K., Thomas, C. M., Tang, Y., Truman, A. W., Viaud, M., Walton, J. D., Walsh, C. T., Weber, T., van Wezel, G. P., Wilkinson, B., Willey, J. M., Wohlleben, W., Wright, G. D., Ziemert, N., Zhang, C., Zotchev, S. B., Breitling, R., Takano, E., & Glöckner, F. O. (2015). Minimum information about a biosynthetic gene cluster. *Nat. Chem. Biol.*, 11(9), 625–631.

Medema, M. H., van Raaphorst, R., Takano, E., & Breitling, R. (2012). Computational tools for the synthetic design of biochemical pathways. *Nat. Rev. Microbiol.*, 10(3), 191–202.

Moore, G. E. (1965). Cramming more components onto integrated circuits. *electronics*, 38 (8).

Muers, M. (2011). Technology: Getting moore from DNA sequencing. *Nat. Rev. Genet.*, 12(9), 586.

Müller, K. & Wickham, H. (2018). Tibble: Simple data frames. *R package version*, 1(2).

Murrell, P. (2002). The grid graphics package. *R News*, 2(2), 14–19.

Neuwirth, E. (2014). RColorBrewer: ColorBrewer palettes. R package version 1.1-2. *The R Foundation*.

Ng, S. K. & Hamilton, I. R. (1973). Carbon dioxide fixation by *veillonella parvula* M 4 and its relation to propionic acid formation. *Can. J. Microbiol.*, 19(6), 715–723.

Nielsen, H. B., Almeida, M., Juncker, A. S., Rasmussen, S., Li, J., Sunagawa, S., Plichta, D. R., Gautier, L., Pedersen, A. G., Le Chatelier, E., Pelletier, E., Bonde, I., Nielsen, T., Manichanh, C., Arumugam, M., Batto, J.-M., Quintanilha Dos Santos, M. B., Blom, N., Borrueal, N., Burgdorf, K. S., Boumezbear, F., Casellas, F., Doré, J., Dworzynski, P., Guarner, F., Hansen, T., Hildebrand, F., Kaas, R. S., Kennedy, S., Kristiansen, K., Kultima, J. R., Léonard, P., Levenez, F., Lund, O., Moumen, B., Le Paslier, D., Pons, N., Pedersen, O., Prifti, E., Qin, J., Raes, J., Sørensen, S., Tap, J., Tims, S., Ussery, D. W., Yamada, T., MetaHIT Consortium, Renault, P., Sicheritz-Ponten, T., Bork, P., Wang, J., Brunak, S., Ehrlich, S. D., & MetaHIT Consortium (2014). Identification and assembly of genomes and genetic elements in complex metagenomic samples without using reference genomes. *Nat. Biotechnol.*, 32(8), 822–828.

Parks, D. H., Rinke, C., Chuvochina, M., Chaumeil, P.-A., Woodcroft, B. J., Evans, P. N., Hugenholtz, P., & Tyson, G. W. (2017). Recovery of nearly 8,000 metagenome-assembled genomes substantially expands the tree of life. *Nat Microbiol*, 2(11), 1533–1542.

Pasolli, E., Asnicar, F., Manara, S., Zolfo, M., Karcher, N., Armanini, F., Beghini, F., Manghi, P., Tett, A., Ghensi, P., Collado, M. C., Rice, B. L., DuLong, C., Morgan, X. C., Golden, C. D., Quince, C., Huttenhower, C., & Segata, N. (2019). Extensive unexplored human microbiome diversity revealed by over 150,000 genomes from metagenomes spanning age, geography, and lifestyle. *Cell*, 176(3), 649–662.e20.

Pauken, K. E., Sammons, M. A., Odorizzi, P. M., Manne, S., Godec, J., Khan, O., Drake, A. M., Chen, Z., Sen, D. R., Kurachi, M., Barnitz, R. A., Bartman, C., Bengsch, B., Huang, A. C., Schenkel, J. M., Vahedi, G., Haining, W. N., Berger, S. L., & Wherry, E. J. (2016). Epigenetic stability of exhausted T cells limits durability of reinvigoration by PD-1 blockade. *Science*, 354(6316), 1160–1165.

Paun, A., Yau, C., Meshkibaf, S., Daigneault, M. C., Marandi, L., Mortin-Toth, S., Bar-Or, A., Allen-Vercoe, E., Poussier, P., & Danska, J. S. (2019). Association of HLA-dependent islet autoimmunity with systemic antibody responses to intestinal commensal bacteria in children. *Sci Immunol*, 4(32).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(Oct), 2825–2830.

Petersen, L. M., Bautista, E. J., Nguyen, H., Hanson, B. M., Chen, L., Lek, S. H., Sodergren, E., & Weinstock, G. M. (2017). Community characteristics of the gut microbiomes of competitive cyclists. *Microbiome*, 5(1), 98.

Phypers, B. & Pierce, J. M. T. (2006). Lactate physiology in health and disease. *Continuing education in Anaesthesia, critical care & pain*, 6(3), 128–132.

Pimentel, G., Burton, K. J., Rosikiewicz, M., Freiburghaus, C., von Ah, U., Münger, L. H., Pralong, F. P., Vionnet, N., Greub, G., Badertscher, R., & Vergères, G. (2017). Blood lactose after dairy product intake in healthy men. *Br. J. Nutr.*, 118(12), 1070–1077.

Pinheiro, J., Bates, D., DebRoy, S., & Sarkar, D. (2014). R core team (2014) nlme: linear and non-linear mixed effects models. R package version 3.1-117. Available at <http://CRAN.R-project.org/package=nlme>.

Pluznick, J. (2014). A novel SCFA receptor, the microbiota, and blood pressure regulation. *Gut Microbes*, 5(2), 202–207.

Pluznick, J. L., Protzko, R. J., Gevorgyan, H., Peterlin, Z., Sipos, A., Han, J., Brunet, I., Wan, L.-X., Rey, F., Wang, T., Firestein, S. J., Yanagisawa, M., Gordon, J. I., Eichmann, A., Peti-Peterdi, J., & Caplan, M. J. (2013). Olfactory receptor responding to gut microbiota-derived signals plays a role in renin secretion and blood pressure regulation. *Proc. Natl. Acad. Sci. U. S. A.*, 110(11), 4410–4415.

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

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

Quince, C., Walker, A. W., Simpson, J. T., Loman, N. J., & Segata, N. (2017). Shotgun metagenomics, from sampling to analysis. *Nat. Biotechnol.*, 35(9), 833–844.

R Core Team, R. & Others (2013). R: A language and environment for statistical computing.

Ráki, M., Fallang, L.-E., Brottveit, M., Bergseng, E., Quarsten, H., Lundin, K. E. A., & Sollid, L. M. (2007). Tetramer visualization of gut-homing gluten-specific T cells in the peripheral blood of celiac disease patients. *Proc. Natl. Acad. Sci. U. S. A.*, 104(8), 2831–2836.



Robert, L., Harview, C., Emerson, R., Wang, X., Mok, S., Homet, B., Comin-Anduix, B., Koya, R. C., Robins, H., Tumeh, P. C., & Ribas, A. (2014). Distinct immunological mechanisms of CTLA-4 and PD-1 blockade revealed by analyzing TCR usage in blood lymphocytes.

Robinson, O., Dylus, D., & Dessimoz, C. (2016). Phylo.io: Interactive viewing and comparison of large phylogenetic trees on the web. *Mol. Biol. Evol.*, 33(8), 2163–2166.

Rothschild, D., Weissbrod, O., Barkan, E., Kurilshikov, A., Korem, T., Zeevi, D., Costea, P. I., Godneva, A., Kalka, I. N., Bar, N., Shilo, S., Lador, D., Vila, A. V., Zmora, N., Pevsner-Fischer, M., Israeli, D., Kosower, N., Malka, G., Wolf, B. C., Avnit-Sagi, T., Lotan-Pompan, M., Weinberger, A., Halpern, Z., Carmi, S., Fu, J., Wijmenga, C., Zhernakova, A., Elinav, E., & Segal, E. (2018). Environment dominates over host genetics in shaping human gut microbiota. *Nature*, 555(7695), 210–215.

Round, J. L. & Mazmanian, S. K. (2009). The gut microbiota shapes intestinal immune responses during health and disease. *Nat. Rev. Immunol.*, 9(5), 313–323.

Satija, R., Butler, A., & Hoffman, P. (2017). Seurat: Tools for single cell genomics. *R package version*, 2(0).

Scheiman, J., Lubner, J. M., Chavkin, T. A., MacDonald, T., Tung, A., Pham, L.-D., Wibowo, M. C., Wurth, R. C., Punthambaker, S., Tierney, B. T., Yang, Z., Hattab, M. W., Avila-Pacheco, J., Clish, C. B., Lessard, S., Church, G. M., & Kostic, A. D. (2019). Meta-omics analysis of elite athletes identifies a performance-enhancing microbe that functions via lactate metabolism. *Nat. Med.*, 25(7), 1104–1109.

Seemann, T. (2014). Prokka: rapid prokaryotic genome annotation. *Bioinformatics*, 30(14), 2068–2069.

Segata, N., Waldron, L., Ballarini, A., Narasimhan, V., Jousson, O., & Huttenhower, C. (2012). Metagenomic microbial community profiling using unique clade-specific marker genes. *Nat. Methods*, 9(8), 811–814.

Sender, R., Fuchs, S., & Milo, R. (2016a). Are we really vastly outnumbered? revisiting the ratio of bacterial to host cells in humans. *Cell*, 164(3), 337–340.

Sender, R., Fuchs, S., & Milo, R. (2016b). Revised estimates for the number of human and bacteria cells in the body. *PLoS Biol.*, 14(8), e1002533.

- Sharpe, A. H. & Pauken, K. E. (2018). The diverse functions of the PD1 inhibitory pathway. *Nat. Rev. Immunol.*, 18(3), 153–167.
- Shepherd, E. S., DeLoache, W. C., Pruss, K. M., Whitaker, W. R., & Sonnenburg, J. L. (2018). An exclusive metabolic niche enables strain engraftment in the gut microbiota. *Nature*, 557(7705), 434–438.
- Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The hadoop distributed file system. In *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)* (pp. 1–10).: [ieeexplore.ieee.org](http://ieeexplore.ieee.org).
- Signorell, A., Aho, K., Alfons, A., Anderegg, N., Aragon, T., Arppe, A., & Others (2016). DescTools: Tools for descriptive statistics. *R package version 0.99*, 18.
- Singer, M., Wang, C., Cong, L., Marjanovic, N. D., Kowalczyk, M. S., Zhang, H., Nyman, J., Sakuishi, K., Kurtulus, S., Gennert, D., Xia, J., Kwon, J. Y. H., Nevin, J., Herbst, R. H., Yanai, I., Rozenblatt-Rosen, O., Kuchroo, V. K., Regev, A., & Anderson, A. C. (2017). A distinct gene module for dysfunction uncoupled from activation in Tumor-Infiltrating T cells. *Cell*, 171(5), 1221–1223.
- Skyler, J. S., Greenbaum, C. J., Lachin, J. M., Leschek, E., Rafkin-Mervis, L., Savage, P., Spain, L., & Type 1 Diabetes TrialNet Study Group (2008). Type 1 diabetes TrialNet—an international collaborative clinical trials network. *Ann. N. Y. Acad. Sci.*, 1150, 14–24.
- Slowikowski, K. (2017). ggrepel: Repulsive text and label geoms for ‘ggplot2’. *R pack-age version 0.5*. Online: <https://CRAN.R-project.org/package=ggrepel>.
- Smanski, M. J., Zhou, H., Claesen, J., Shen, B., Fischbach, M. A., & Voigt, C. A. (2016). Synthetic biology to access and expand nature’s chemical diversity. *Nat. Rev. Microbiol.*, 14(3), 135–149.
- Spoto, M., Fleming, E., & Oh, J. (2017). A universal, genome-wide guide finder for CRISPR/Cas9 targeting in microbial genomes.
- Stoeckius, M., Hafemeister, C., Stephenson, W., Houck-Loomis, B., Chattopadhyay, P. K., Swerdlow, H., Satija, R., & Smibert, P. (2017). Simultaneous epitope and transcriptome measurement in single cells. *Nat. Methods*, 14(9), 865–868.
- Strachan, D. P. (2000). Family size, infection and atopy: the first decade of the ‘hygiene hypothesis’. *Thorax*, 55(suppl 1), S2–10.

- TEDDY Study Group (2007). The environmental determinants of diabetes in the young (TEDDY) study: study design. *Pediatr. Diabetes*, 8(5), 286–298.
- Tierney, B. T., Yang, Z., Lubber, J. M., Beaudin, M., Wibowo, M. C., Baek, C., Mehlenbacher, E., Patel, C. J., & Kostic, A. D. (2019). The landscape of genetic content in the gut and oral human microbiome. *Cell Host Microbe*, 26(2), 283–295.e8.
- Tordsson, J., Montero, R. S., Moreno-Vozmediano, R., & Llorente, I. M. (2012). Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Gener. Comput. Syst.*, 28(2), 358–367.
- Truong, D. T., Franzosa, E. A., Tickle, T. L., Scholz, M., Weingart, G., Pasolli, E., Tett, A., Huttenhower, C., & Segata, N. (2015). MetaPhlan2 for enhanced metagenomic taxonomic profiling. *Nat. Methods*, 12(10), 902–903.
- Turnbaugh, P. J., Ley, R. E., Hamady, M., Fraser-Liggett, C. M., Knight, R., & Gordon, J. I. (2007). The human microbiome project. *Nature*, 449(7164), 804–810.
- van den Bogert, B., Boekhorst, J., Smid, E. J., Zoetendal, E. G., & Kleerebezem, M. (2013). Draft genome sequence of *veillonella parvula* HSIVP1, isolated from the human small intestine. *Genome Announc.*, 1(6).
- Vatanen, T., Franzosa, E. A., Schwager, R., Tripathi, S., Arthur, T. D., Vehik, K., Lernmark, Å., Hagopian, W. A., Rewers, M. J., She, J.-X., Toppari, J., Ziegler, A.-G., Akolkar, B., Krischer, J. P., Stewart, C. J., Ajami, N. J., Petrosino, J. F., Gevers, D., Lähdesmäki, H., Vlamakis, H., Huttenhower, C., & Xavier, R. J. (2018). The human gut microbiome in early-onset type 1 diabetes from the TEDDY study. *Nature*, 562(7728), 589–594.
- Vatanen, T., Kostic, A. D., d’Hennezel, E., Siljander, H., Franzosa, E. A., Yassour, M., Kolde, R., Vlamakis, H., Arthur, T. D., Hämäläinen, A.-M., Peet, A., Tillmann, V., Uibo, R., Mokurov, S., Dorshakova, N., Ilonen, J., Virtanen, S. M., Szabo, S. J., Porter, J. A., Lähdesmäki, H., Huttenhower, C., Gevers, D., Cullen, T. W., Knip, M., DIABIMMUNE Study Group, & Xavier, R. J. (2016). Variation in microbiome LPS immunogenicity contributes to autoimmunity in humans. *Cell*, 165(6), 1551.
- Warnes, G. R., Bolker, B., & Lumley, T. (2015). *gtools: Various R programming tools*. R package version 3.5.0.

- Wickham, H. (2012). *reshape2*: Flexibly reshape data: a reboot of the reshape package. R package version. 2012; 1.
- Wickham, H. & Chang, W. (2015). *ggplot2*: An implementation of the grammar of graphics. 2015. URL <http://CRAN.R-project.org/package=ggplot2>. R package version, 1(1).
- Wickham, H. & Francois, R. (2015). *dplyr*: A grammar of data manipulation. R package version 0.4.3.
- Wilke, C. O. (2016). *cowplot*: streamlined plot theme and plot annotations for ‘ggplot2’. *CRAN Repos*, 2, R2.
- Wolf, F. A., Angerer, P., & Theis, F. J. (2018). SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.*, 19(1), 15.
- Wymore Brand, M., Wannemuehler, M. J., Phillips, G. J., Proctor, A., Overstreet, A.-M., Jergens, A. E., Orcutt, R. P., & Fox, J. G. (2015). The altered schaedler flora: Continued applications of a defined murine microbial community. *ILAR J.*, 56(2), 169–178.
- Xu, T., Li, Y., Shi, Z., Hemme, C. L., Li, Y., Zhu, Y., Van Nostrand, J. D., He, Z., & Zhou, J. (2015). Efficient genome editing in *Clostridium cellulolyticum* via CRISPR-Cas9 nickase. *Appl. Environ. Microbiol.*, 81(13), 4423–4431.
- Yang, H.-C., Dasdan, A., Hsiao, R.-L., & Parker, D. S. (2007). Map-reduce-merge: Simplified relational data processing on large clusters. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, SIGMOD '07* (pp. 1029–1040). New York, NY, USA: ACM.
- Yassour, M., Vatanen, T., Siljander, H., Hämäläinen, A.-M., Härkönen, T., Ryhänen, S. J., Franzosa, E. A., Vlamakis, H., Huttenhower, C., Gevers, D., Lander, E. S., Knip, M., DIABIMMUNE Study Group, & Xavier, R. J. (2016). Natural history of the infant gut microbiome and impact of antibiotic treatment on bacterial strain diversity and stability. *Sci. Transl. Med.*, 8(343), 343ra81.
- Yost, K. E., Satpathy, A. T., Wells, D. K., Qi, Y., Wang, C., Kageyama, R., McNamara, K. L., Granja, J. M., Sarin, K. Y., Brown, R. A., Gupta, R. K., Curtis, C., Bucktrout, S. L., Davis, M. M., Chang, A. L. S., & Chang, H. Y. (2019). Clonal replacement of tumor-specific T cells following PD-1 blockade. *Nat. Med.*, 25(8), 1251–1259.
- Zhang, L., Yu, X., Zheng, L., Zhang, Y., Li, Y., Fang, Q., Gao, R., Kang, B., Zhang, Q., Huang, J. Y., Konno, H., Guo, X., Ye, Y., Gao, S., Wang, S., Hu, X., Ren, X., Shen, Z., Ouyang, W., &

Zhang, Z. (2018). Lineage tracking reveals dynamic relationships of T cells in colorectal cancer. *Nature*, 564(7735), 268–272.

Zheng, L., Joe-Wong, C., Tan, C. W., Chiang, M., & Wang, X. (2015). How to bid the cloud. *ACM SIGCOMM Computer Communication Review*, 45(4), 71–84.

Zimmermann, M. & Fischbach, M. A. (2010). A family of pyrazinone natural products from a conserved nonribosomal peptide synthetase in staphylococcus aureus. *Chem. Biol.*, 17(9), 925–930.



**T**HIS THESIS WAS TYPESET using  $\LaTeX$ , originally developed by Leslie Lamport and based on Donald Knuth's  $\TeX$ .

The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. The above illustration, *Science Experiment 02*, was created by Ben Schlitter and released under [CC BY-NC-ND 3.0](#). A template that can be used to format a PhD dissertation with this look & feel has been released under the permissive [AGPL](#) license, and can be found online at [github.com/suchow/Dissertate](https://github.com/suchow/Dissertate) or from its lead author, Jordan Suchow, at [suchow@post.harvard.edu](mailto:suchow@post.harvard.edu).