Interpretable and reliable statistical models for biomedicine

Jean Feng

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Noah Simon, Chair

Frederick Matsen IV, Chair

Ali Shojaie

Program Authorized to Offer Degree:
Biostatistics

University of Washington

Abstract

Interpretable and reliable statistical models for biomedicine

Jean Feng

Co-Chairs of the Supervisory Committee:

Dr. Noah Simon
Department of Biostatistics, University of Washington

Dr. Frederick Matsen IV
Computational Biology Program, Fred Hutchinson Cancer Research Center

This dissertation presents a collection of statistical tools to analyze modern biomedical datasets, which have been transformed by developments in high-throughput and high-content biology. Due to rapid growth in both scale and complexity of these datasets, there is a need for new inference procedures that combine both statistical and computational perspectives. Our results have been organized according to two major themes. In Part One, we use modern sequencing and gene-editing technologies to understand immunological and developmental processes. In Part Two, we study the accuracy and reliability of non-parametric machine learning algorithms, motivated by their growing use in medicine and healthcare.

TABLE OF CONTENTS

Page

Introduction 1

Part I: Interpretable models for biology 3

Chapter 1: Survival Analysis of DNA Mutation Motifs with Penalized Proportional Hazards 4

1 Introduction 4
2 Methods 7
3 Simulation results 22
4 Data analysis 29
5 Discussion 34
A Appendix 35

Chapter 2: Estimation of Cell Lineage Trees by Maximum-Likelihood Phylogenetics 54

1 Introduction 54
2 GESTALT model 57
3 Estimation method 71
4 Simulation engine and results 73
5 Real data analysis of a zebrafish 77
6 Discussion 84
A Appendix 85

Part II: Prediction models for medicine and healthcare 111

Chapter 3: Ensembled sparse-input hierarchical networks for high-dimensional datasets 112

1 Introduction 112
2 Related Work 115
3 Method 117
4 Empirical analyses 127
5 Discussion 133

A Appendix 136

Chapter 4: Approval Policies for Modifications to Machine Learning-Based Software as a Medical Device: A Study of Bio-Creep 150
1 Introduction 151
2 Motivating examples 153
3 Problem setup 155
4 An online hypothesis testing framework 159
5 Online error rates for aACPs 161
6 Cumulative utility of an aACP 170
7 Simulations 170
8 Discussion 175
A Appendix 177

Future Directions 192

ACKNOWLEDGMENTS

I am grateful to the many people who have supported me through my graduate studies. First and foremost, I thank my co-advisors Noah Simon and Frederick (Erick) Matsen IV. They have selflessly given me the academic freedom to explore a wide breadth of topics and chart out my research interests, and have been enthusiastic about every project and path I take. They were candid in their feedback and compassionate in their mentorship, which were essential to my success in graduate school. I thank Ali Shojaie on my reading committee as well, who has been a steady source of support and has provided insightful comments on every one of my projects.

I would also like to thank Scott Emerson and Vladimir Minin, who have been fantastic collaborators and mentors throughout my Ph.D. Scott taught me introductory statistics (starting with the T-test) and was instrumental in my research on regulatory science for machine learning. Vladimir Minin impressed his insistence on statistical rigor on me and provided generous advice during my job search. I thank both of them for their good humor.

I thank Amy Willis for being a collaborator and, more importantly, a friend. I am indebted to her willingness to share her experiences as a new assistant professor. Chats with her made me realize that an academic career was a feasible option.

Of course, many others in the UW Biostatistics Department and the Fred Hutchinson Cancer Research Center have also helped me during my graduate studies, including Ken Rice, Mauricio Saudinle, and Ruth Etzioni. I also want to give special thanks to Gitana Garofalo for letting me randomly pop in for long chats and her tireless advocacy for student health and wellbeing. Thank you to the many friends I have made along the way, including but certainly not limited to my cohort, Slab Lab, the Matsen group, and people who hang out in The Cold Room.

I would be remiss if I did not acknowledge the pivotal role Daphne Koller has played throughout my academic (and personal!) journey. She was my undergraduate advisor, professor, and my boss twice over. I am deeply indebted to her continual guidance and care. None of this would have been possible without her.

To my parents, I give my gratitude for "the principles they instilled me and for the opportunities they afforded me," and this template sentence from both of their dissertations' acknowledgments sections. I would also like to thank my sister Karen, who has been my cheerleader, public speaking drill sergeant, grammar police, and tofu cook.

Finally, I would like to thank my partner JJ, who has helped me through every struggle, think through every idea, and stood by me unfailingly through it all. This dissertation is not just dedicated to you. You deserve this Ph.D. as well.

DEDICATION

to JJ, Karen, and my parents

INTRODUCTION

This dissertation is a collection of statistical tools developed for biomedicine, against the backdrop of advancements in computation and biotechnology in the past fifty years. Exponential growth in computational power and speed has removed computational bottlenecks of existing algorithms and facilitated the growth of many new statistical techniques. High-throughput and high-content biotechnologies have rapidly increased the amount of data we can measure and the information we can capture. However, with scale comes complexity, and our need for interpretable and reliable statistical models is more important than ever.

In Part I, we build interpretable models for understanding biological processes. The models are interpretable by design, as they are built in close collaboration with domain experts and incorporate their expertise and prior knowledge. Chapter 1 adapts survival analysis methods to analyze immune receptor sequencing data and estimate the effect of local sequence context on antibody mutation rates. Chapter 2 proposes a likelihood-based phylogenetic method to analyze data from CRISPR-based cell lineage tracing technologies and reconstruct cell lineage trees of large multicellular organisms.

In Part II, we study the accuracy and reliability of non-parametric machine learning methods, motivated by their growing use in medicine and healthcare. The reliability of these models is imperative, since they may be used to make important and risky medical decisions. Chapter 3 introduces a modified neural network architecture and training procedure for analyzing datasets that are common in biomedicine, i.e. those with a large number of covariates, relatively few observations, and a small signal-to-noise ratio. We employ theoretical analyses to understand the method's properties as well as its limitations. In Chapter 4, we build a statistical framework, based on online hypothesis testing, for monitoring and regulating machine-learning algorithms that evolve over time.

Related Publiations

At the time of writing, the following chapters have been published:

Chapter 1:

Feng, Jean, David A. Shaw, Vladimir N. Minin, Noah Simon, and Frederick A. Matsen IV. 2019. "Survival Analysis of DNA Mutation Motifs with Penalized Proportional Hazards." The Annals of Applied Statistics 13 (2): 1268–94. https://doi.org/10.1214/18-AOAS1233.

Chapter 4:

Feng, Jean, Scott Emerson, and Noah Simon. 2019. "Approval Policies for Modifications to Machine Learning-Based Software as a Medical Device: A Study of Bio-Creep." Biometrics. In press.

Part I

INTERPRETABLE MODELS FOR BIOLOGY

The first part of this thesis considers two areas in biology that have seen rapid transformation in the past decade: immunology and developmental biology. Using modern immune receptor sequencing technology, we are now able to sequence millions of receptors from a single individual to perform more comprehensive immune profiling. In addition, by combining CRISPR-based gene-editing technologies with single-cell sequencing technologies, researchers are now able to perform cell lineage tracing in large multicellular organisms. We develop computationally-efficient statistical procedures in the following two chapters to model the mutation process that generate these data. Chapter 1 applies a survival analysis framework to model somatic hypermutation of immune receptor genes. Chapter 2 models the CRISPR/Cas9 mutation process as a continuous time Markov chain. We then apply these tools to gain insight into these highly complex biological processes.

Chapter 1

SURVIVAL ANALYSIS OF DNA MUTATION MOTIFS WITH PENALIZED PROPORTIONAL HAZARDS

Summary

Antibodies, an essential part of our immune system, develop through an intricate process to bind a wide array of pathogens. This process involves randomly mutating DNA sequences encoding these antibodies to find variants with improved binding, though mutations are not distributed uniformly across sequence sites. Immunologists observe this nonuniformity to be consistent with "mutation motifs", which are short DNA subsequences that affect how likely a given site is to experience a mutation. Quantifying the effect of motifs on mutation rates is challenging: a large number of possible motifs makes this statistical problem high dimensional, while the unobserved history of the mutation process leads to a nontrivial missing data problem. We introduce an ℓ_1-penalized proportional hazards model to infer mutation motifs and their effects. In order to estimate model parameters, our method uses a Monte Carlo EM algorithm to marginalize over the unknown ordering of mutations. We show that our method performs better on simulated data compared to current methods and leads to more parsimonious models. The application of proportional hazards to mutation processes is, to our knowledge, novel and formalizes the current methods in a statistical framework that can be easily extended to analyze the effect of other biological features on mutation rates.

1 Introduction

We introduce a proportional hazards model approach to study DNA mutation processes. Our study is motivated by somatic hypermutation, a mutation process that occurs in DNA sequences that encode B-cell receptors (BCRs), proteins that recognize and neutralize

pathogens. When BCRs are secreted from B cells they are known as antibodies. The immune system relies on this somatic hypermutation process to generate a diversity of BCRs that can bind to a large and continually evolving variety of pathogens. The starting material for this mutation process is a BCR sequence that is formed by recombination [48, 44]. From this sequence, a complex system of enzymes introduces mutations in a random pattern that is known to be highly sensitive to the sequence *motif*: the sequence of DNA bases surrounding the mutating position [13, 4, 42, 36].

Our goal is to develop a solid statistical framework that estimates the mutation rates of motifs and provides interpretable results for this mutation process. A better understanding of somatic hypermutation will help in designing vaccines for challenging viruses [21, 29, 51], in furthering understanding of the biological mechanisms at play [38, 43], and in gaining insight into the natural selection process occurring in the immune system [23, 49, 35, 27].

Several strategies have been used to model a motif's mutability – that is, how likely a position is to mutate given the motif at that position. The general approach is to compare a mutated sequence with its inferred ancestor sequence and model the differences between them. Cohen, Kleinstein, and Louzoun [6] and Elhanati, Sethna, Marcou, Callan, Mora, and Walczak [14] model the mutabilities of motifs as the product of the mutabilities of short subsequences (usually 1 or 2 bases). By using a log-linear model with only first-order terms they keep the parameter count low, but miss interactions between the positions. Yaari, Vander Heiden, Uduman, Gadala-Maria, Gupta, Stern, O'Connor, Hafler, Laserson, Vigneault, and Kleinstein [54] and Cui, Di Niro, Vander Heiden, Briggs, Adams, Gilbert, O'Connor, Vigneault, Shlomchik, and Kleinstein [8] do not use this log-linear assumption: they allow a separate parameter for each possible five-nucleotide motif (of which there are 4^5), and use ad-hoc methods to handle motifs with few observations. Rather than these restrictive and ad-hoc approaches, a more data-adaptive variable selection method is desirable.

Another drawback of these methods is that they ignore mutations that occur in neigh-

boring positions, even though such events can carry important information about highly mutable motifs. Indeed, these methods require counting the number of times a motif is observed to mutate: if mutations occur in neighboring positions, they cannot attribute the mutation to the correct motif. For settings with high rates of mutation, these methods end up estimating the mutabilities poorly. To properly estimate mutabilities, one needs to account for the different possible orders that mutations occurred in. Previous work has developed methods for performing various types of inference when this mutation order is unknown [28, 26], but these inference procedures make the parametric assumption that the mutation process follows a continuous time Markov process. Here we relax this model assumption and use a semiparametric model instead.

In this chapter, we advance the modeling of motif mutabilities in several directions. We propose a method to fit mutabilities using survival analysis of mutation motifs, called `samm`. We formalize the problem using Cox proportional hazards, in which mutations are the failure events to be investigated. Although survival models are used implicitly by computational immunologists for simulation [53, 45], we believe this is the first time they have been used for inference.

To estimate motif mutabilities, our method uses the Monte Carlo expectation–maximization algorithm [MCEM, 50]. Since the orders in which mutations occur are unobserved in our data, expectation–maximization [EM, 10] allows us to perform maximum likelihood while averaging over these unknown orders. However the E-step in EM requires calculating the expected log-likelihood, which is analytically intractable since we must average over all possible mutation orders; thus we estimate this expectation using Gibbs sampling. This approach is similar to that used by Goggins, Finkelstein, Schoenfeld, and Zaslavsky [19] to model interval-censored failure-time data where the order in which the failure events occur is unknown.

Our method also handles high-dimensional settings in which there are many more predictors than observations, which is important because many motifs are hypothesized to affect

the mutation rate but the specific ones are unknown. For instance, Yaari et al. [54] and Cui et al. [8] consider all motifs of length 5. We use the lasso [46] to improve estimation and perform variable selection. To provide a measure of uncertainty of our estimates, we use a two-step approach: we fit an ℓ_1-penalized Cox proportional hazards model [47] to perform variable selection and refit an unpenalized model over the selected variables to obtain our final estimates along with approximate confidence intervals.

Section 2 describes our estimation methods, starting with a simplified logistic regression model and then progressing to our full estimation method. Section 3 presents simulation results. In Section 4, we apply our method to model somatic hypermutation of BCR sequences from Cui et al. [8] and compare results with previous methods.

2 Methods

Our data consists of BCR nucleotide sequences that have mutated for an unknown period of time. Specifically, we target sequences that are undergoing mutation but not natural selection. Such data can be obtained, for example, through immunization experiments in transgenic mice designed to have a DNA segment that is carried along and mutated but not expressed as part of the BCR [56, 8].

Though we focus on modeling the somatic hypermutation process of BCRs, our approach can be framed more generally as a problem of modeling a sequence-valued mutation process. We refer to the original, unmutated sequences as "naïve" and their descendants as "mutated" sequences. Throughout, we suppose that these naïve sequences are known. In the BCR case, we restrict our attention to a computationally-identified naïve segment coded in germline DNA [the V region, 52].

More formally, the mutation process of a sequence with p positions can be described as a vector-valued stochastic process $\{X(t) = (X_1(t), ..., X_p(t)) : t \in [0, \infty)\}$ indexed by time t. Each $\{X_j(t)\}$ represents the mutation process of the jth position in the sequence. For a given time t, the state space of $X_j(t)$ is the set of nucleotides $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ and the state space of $X(t)$ is the set of length-p nucleotide sequences $\mathcal{S} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^p$. At the start of

the mutation process, $X(0)$ is fixed to be the naïve sequence.

In a context-sensitive model, the probability that a position mutates at time t depends on the current nucleotide sequence $X(t)$. In our work, we assume that only local context matters: The mutation rate at each position is affected only by the local nucleotide sequence called the motif. For motif m, we denote the length of the motif as $\text{len}(m)$, where $\text{len}(m)$ is typically much smaller than the number of nucleotides in $X(t)$. The function $I(X(t), m, j, j')$ is the binary indicator of whether motif m appears in sequence $X(t)$ from positions $j - j' + 1$ to $j - j' + \text{len}(m)$. More formally, it is defined as

$$I(X(t), m, j, j') = \prod_{k=1}^{\text{len}(m)} 1\left\{X_{j-j'+k}(t) = m_k\right\}, \tag{1.1}$$

where m_k is the nucleotide in the kth position of motif m. This is known as a $\text{len}(m)$-mer, i.e., a motif of length $\text{len}(m)$. For example, a 5-mer is a motif of length 5. In the special case where $\text{len}(m)$ is odd and $j' = (\text{len}(m) + 1)/2$, (1.1) checks if $X(t)$ has motif m centered at position j. We call this a centered motif; for all other cases we say that (1.1) is checking for an offset motif.

Define a *motif dictionary* to be a set $\mathcal{M}$ of sequence features (m, j') that may affect mutation rate. Example dictionaries include 1-mers (all length 1 motifs), offset 2-mers (length 2 motifs with $j' = 1, 2$), all of the central and offset 3-mers (length 3 motifs, with $j' = 1, 2, 3$), and all of the central 5-mers. We may also consider all possible unions of these dictionaries. Suppose we have selected a set $\mathcal{M}$. To ease exposition, we choose an arbitrarily assigned but fixed order $\left\{(m^{(1)}, j'^{(1)}), \ldots, (m^{(q)}, j'^{(q)})\right\}$ where q is the number of motif features in the dictionary $\mathcal{M}$. We may now define a function that indicates which elements in $\mathcal{M}$ occur at each position. For each position j, let $\psi_j : \mathcal{S} \mapsto \{0, 1\}^q$ be defined by $[\psi_j(X(t))]_k \equiv I(X(t), m^{(k)}, j, j'^{(k)})$ for $k = 1, ..., q$. We use ψ_j as the feature vector for modeling the mutation rate of position j (Figure 1.1).

Of course, the framework we present here generalizes to other types of dictionaries, including dictionaries that only specify bases for a subset of positions, but we will restrict

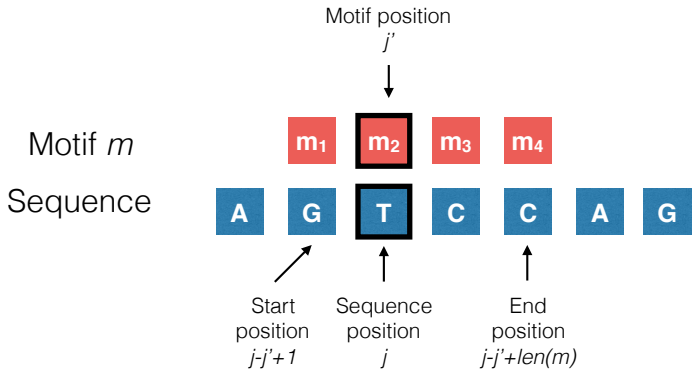


Figure 1.1: An example of how feature vectors are generated: if we believe that the mutation rate at a position depends on the 4-mer (i.e. length 4 motif) starting one position to its left, then the feature vector for position j is a one-hot encoding of the sequence that appears in position $j-1$ through $j+2$. More formally, each element in the feature vector at position j indicates whether or not a motif m appears from start position $j-j'+1$ through end position $j-j'+\text{len}(m)$ (here $m=4$ and $j'=2$). The start and end positions are derived by aligning position j of the sequence with position j' of the motif.

to the above-described dictionaries for concreteness.

2.1 Logistic regression

As a simplified approach to modeling the mutation process, one may ignore the time component and use logistic regression. In this model, each position in the sequence is independent and the probability of each position mutating only depends on the *initial* nucleotide sequence $X(0)$, i.e.

$$\Pr(\text{mutation at position } j) = \frac{1}{1+\exp(-\boldsymbol{\theta}^\top \psi_j(X(0)))} \qquad \forall j \in \{1,...,p\}. \tag{1.2}$$

Yaari, Vander Heiden, Uduman, Gadala-Maria, Gupta, Stern, O'Connor, Hafler, Laserson, Vigneault, and Kleinstein [54] essentially take this approach; the logistic model here just formalizes their intuition within a statistical framework and allows us to generalize their method to be applicable for any feature vector mapping. Moreover, we can use penalized logistic regression for handling high-dimensional models and encode various structural

assumptions regarding the mutation rates; we discuss this in detail later in Section 2.4.

Logistic regression ignores the time component in a mutation process, and as such ignores how the mutation rate of each position may change as other positions mutate (Figure 1.2). The assumption that the mutation rate only depends on the initial nucleotide sequence is most problematic when the mutation rate is high. Also, logistic regression ignores censoring: the method estimates the average mutation probability with respect to a particular sampling process. The estimates will be different if we tend to sample sequences that mutate for long vs. short periods of time. The following section addresses these issues by modeling the mutation process using a survival analysis framework.

2.2 Cox proportional hazards

We propose using a survival analysis framework to model the mutation process. We view positions in a single sequence as subjects observed for the same time period. A mutation event at position j occurs at time t if the nucleotide immediately before time t, $\lim_{s \to t^-} X_j(s)$, differs from the nucleotide at time t, $X_j(t)$. If a position never mutates, we consider its mutation time to be censored.[1] The hazard (or mutation) rate of a position is the instantaneous risk of mutating at time t given that it has been conserved up to time t. In between successive mutation times, each position has a constant hazard rate, and mutates independently from all other positions. The dependence between positions is introduced when a mutation occurs: upon a mutation event, the hazard rate for each neighboring position can change (Figure 1.2).

Accounting for how the sequence can change over time complicates our estimation procedure. Since we do not observe the order of mutation events in the data – we only observe pairs of naïve and mutated sequences – there are many possible mutation orders that could explain how the mutated sequence arose from the naïve sequence; each mutation order

[1] By using a survival analysis framework, we implicitly assume that a mutation will occur at every position given a sufficiently long period of time. This assumption is reasonable for somatic hypermutation – the complex system of enzymes has the ability to mutate any position along the sequence [4]. This assumption may not hold for other DNA mutation processes, and the method may need to be modified accordingly.

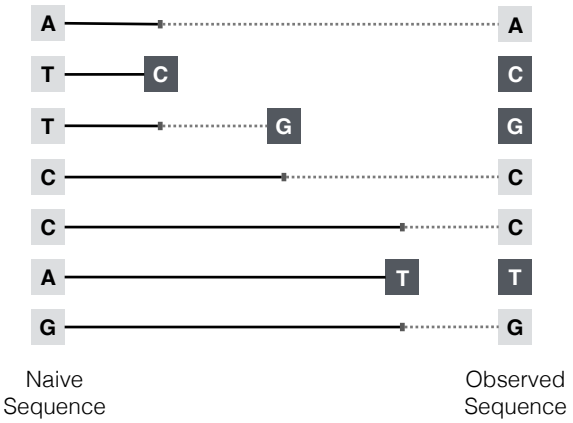


Figure 1.2: Survival analysis for BCR sequences, where the positions that have not mutated are indicated by light gray squares and those that have mutated are indicated by dark gray squares. In a context-dependent mutation model, a mutation event can change the mutation rates of other positions. Suppose the hazard (i.e. mutation) rate of a position depends on the position's two neighboring bases. Changes in the motif at a potential mutating position, and thus its hazard rate, are indicated by a change from solid to dashed lines.

corresponds to a distinct sequence of hazard rates.

For ease of exposition, we present our estimation method for the mutation process of a single pair of naïve and mutated nucleotide sequences. The method readily applies to estimating rates given many independent mutation processes (a typical application will be to thousands or more sequences).

As part of our modeling framework, we assume that each position can mutate at most once during the mutation process. This is a simplification of the somatic hypermutation process since it is possible for a position to mutate more than once, though in our data the naïve and mutated sequence typically differ in 1–5% of the positions. We think this assumption is reasonable and makes the problem easier to handle from a computational standpoint. We discuss how this assumption affects performance under model misspecification in Section A.3 of the Appendix [18].

We model the hazard rate of position j using Cox proportional hazards, which supposes

that the hazard rate j at time t is assumed to be of the form

$$h_j(t) = h_0(t) \exp\left(\boldsymbol{\theta}^\top \psi_j\left(X(t)\right)\right), \tag{1.3}$$

where $\boldsymbol{\theta} \in \mathbb{R}^q$ and the baseline hazard rate $h_0(\cdot)$ is an arbitrary unspecified baseline hazard function. Extending (1.3), we can additionally model the rate at which our process mutates to a specific nucleotide – the *target nucleotide*. Previous work [7, 43, 54, 8] suggests that the context-dependent mutation process is biased in favor of mutations to particular bases. We can take into account these preferences by considering a *per-target model*. In such a model, we additionally define vectors $\boldsymbol{\theta}_{\mathtt{N}}$ for each possible target nucleotide $\mathtt{N} \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Using a competing events framework, the rate of mutating to nucleotide $\mathtt{N}$ at position j at time t is modeled as

$$h_{j,\to\mathtt{N}}(t) = 1\{X_j(t) \neq \mathtt{N}\} h_0(t) \exp\left(\left(\boldsymbol{\theta} + \boldsymbol{\theta}_{\mathtt{N}}\right)^\top \psi_j(X(t))\right). \tag{1.4}$$

As $\mathtt{N} \to \mathtt{N}$ is not considered a mutation, we include the indicator function $1\{\cdot\}$ in (1.4) to specify that a position cannot mutate to the nucleotide that currently appears there.

2.3 Maximum likelihood via MCEM

We are now ready to present a maximum likelihood estimation method for our model. We assume that the hazard rate follows (1.3). The per-target model in (1.4) is a straightforward extension of this simpler case. Let the observed data, namely the single pair of naïve and mutated nucleotide sequences, be denoted $\mathbf{S}_{\text{obs}}$, where we suppose that n positions have mutated.

When $h_0(t)$ is an arbitrary unspecified baseline hazard function, only the order of the mutations carries information about $\boldsymbol{\theta}$, even if the mutation times are observed [30]. Explained intuitively, time can be transformed by an arbitrary increasing function and the form of the hazard function would still be of the form (1.3). [For more details, see Chapter

4 in 30]. Consequently, estimating $\boldsymbol{\theta}$ involves only maximizing the likelihood of observing the mutation order.

For now, suppose we observe the order that the mutations occurred in. Let π_j be the position of the jth mutation for $j = 1, ..., n$. Let $\boldsymbol{\pi}_{1:j}$ denote the positions of the first through jth mutation, where $\boldsymbol{\pi}_{1:0}$ is defined to be the empty set. Define $S(\boldsymbol{\pi}_{1:j})$ to be the nucleotide sequence after positions $\boldsymbol{\pi}_{1:j}$ mutate. Thus the observed data is $\mathbf{S}_{\text{obs}} = \{S(\pi_{1:0}), S(\pi_{1:n})\}$. The set $R(\boldsymbol{\pi}_{1:j}) \equiv \{1, ..., p\} \setminus \boldsymbol{\pi}_{1:j}$ is the set of positions at risk of mutating, commonly referred to as the risk group in the survival analysis literature. Then the marginal likelihood of $\boldsymbol{\theta}$ is

$$\mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) = \prod_{j=1}^{n} \frac{\exp(\boldsymbol{\theta}^\top \psi_{\pi_j}(S(\boldsymbol{\pi}_{1:j-1})))}{\sum_{k \in R(\boldsymbol{\pi}_{1:j-1})} \exp(\boldsymbol{\theta}^\top \psi_k(S(\boldsymbol{\pi}_{1:j-1})))}. \tag{1.5}$$

Our result looks like the marginal likelihood derived in equation 4.47 in [30] except that it is derived under a more general set of assumptions – whereas they assume the covariates are fixed, we assume the covariates to be fixed between events. The derivation of (1.5) is given in the Appendix [18].

The marginal likelihood in (1.5) implies that the mutation order can be simulated by drawing positions from successive multinomial distributions. To simulate mutation at the jth position, we draw a position from the risk group $R(\boldsymbol{\pi}_{1:j-1})$. In fact, Gupta, Vander Heiden, Uduman, Gadala-Maria, Yaari, and Kleinstein [20] use this procedure to simulate the somatic hypermutation process, though they do not provide a statistical justification.

Unfortunately the mutation order $\boldsymbol{\pi}$ is not observed in our problem. We instead maximize the observed data likelihood, which is the complete data likelihood marginalized over all admissible mutation orders $\mathcal{A}(\mathbf{S}_{\text{obs}})$:

$$\mathcal{L}(\mathbf{S}_{\text{obs}}; \boldsymbol{\theta}) = \sum_{\boldsymbol{\pi} \in \mathcal{A}(\mathbf{S}_{\text{obs}})} \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}). \tag{1.6}$$

Assuming positions mutate at most once, $\mathcal{A}(\mathbf{S}_{\text{obs}})$ is a set of $n!$ possible mutation orders.

When the number of mutated positions n is small, we can enumerate all possible mutation orders and maximize (1.6) using a nonlinear optimization algorithm such as EM [10]. However, in most data sets, n is much too large for direct enumeration to be computationally tractable, so we maximize (1.6) using MCEM.

MCEM extends the traditional EM algorithm by approximating the expectation in the E-step using a Monte Carlo sampling method. Let $\boldsymbol{\pi} = \boldsymbol{\pi}_{1:n}$ be a full mutation order. We use the Gibbs sampler in Algorithm 1 to sample $\boldsymbol{\pi} \mid \{\mathbf{S}_{\text{obs}}, \boldsymbol{\theta}\}$. Given a full mutation order $\boldsymbol{\pi}$, let $\boldsymbol{\pi}_{(-j)}$ be the partial mutation order where the jth mutation is removed from $\boldsymbol{\pi}$; a full mutation order $\boldsymbol{\pi}'$ is consistent with $\boldsymbol{\pi}_{(-j)}$ if there is some $j' \in \{1, ..., n\}$ such that $\boldsymbol{\pi}'_{(-j')} = \boldsymbol{\pi}_{(-j)}$. For instance, if $\boldsymbol{\pi} = [1, 3, 2]$ then the partial mutation order $\boldsymbol{\pi}_{(-2)}$ is $[1, 2]$ and $\boldsymbol{\pi}' = [3, 1, 2]$ is consistent with $\boldsymbol{\pi}_{(-2)}$ since $\boldsymbol{\pi}_{(-2)} = \boldsymbol{\pi}'_{(-1)}$. For each Gibbs sweep, the index j cycles through $\{1, ..., n\}$ in some random order. For Gibbs step k, we sample a full mutation order $\boldsymbol{\pi}^{(k)}$ that is consistent with the partial mutation order $\boldsymbol{\pi}^{(k-1)}_{(-j)}$. The proof that this sampler converges to the desired probability distribution is standard and similar to that of Goggins, Finkelstein, Schoenfeld, and Zaslavsky [19].

We efficiently calculate the probability of a full mutation order given a partial mutation order by reusing previous computations. In particular, for partial mutation order $\boldsymbol{\pi}_{(-j)}$, we calculate the probabilities of each consistent full mutation order starting from the full mutation order where position π_j mutates first to that where position π_j mutates last. By ordering consistent full mutation orders in this way, the j'th consistent full mutation order $\boldsymbol{\pi}'$ and $(j'+1)$th consistent full mutation order $\boldsymbol{\pi}''$ are the same except that the j' and $(j'+1)$th mutations are swapped. The ratio of the conditional probabilities of $\boldsymbol{\pi}'$ and $\boldsymbol{\pi}''$

given $\boldsymbol{\pi}_{(-j)}$ is

$$\frac{\Pr(\boldsymbol{\pi}'|\boldsymbol{\pi}_{(-j)})}{\Pr(\boldsymbol{\pi}''|\boldsymbol{\pi}_{(-j)})} = \frac{\exp\left(\boldsymbol{\theta}^\top\left(\psi_{\pi'_{j'}}\left(S(\boldsymbol{\pi}'_{1:j'-1})\right) + \psi_{\pi'_{j'+1}}\left(S(\boldsymbol{\pi}'_{1:j'})\right)\right)\right)}{\exp\left(\boldsymbol{\theta}^\top\left(\psi_{\pi''_{j'}}\left(S(\boldsymbol{\pi}''_{1:j'-1})\right) + \psi_{\pi''_{j'+1}}\left(S(\boldsymbol{\pi}''_{1:j'})\right)\right)\right)} \times \frac{\sum_{i\in R(\boldsymbol{\pi}''_{1:j'})}\exp\left(\boldsymbol{\theta}^\top\psi_i\left(S(\boldsymbol{\pi}''_{1:j'})\right)\right)}{\sum_{i\in R(\boldsymbol{\pi}'_{1:j'})}\exp\left(\boldsymbol{\theta}^\top\psi_i\left(S(\boldsymbol{\pi}'_{1:j'})\right)\right)}. \tag{1.7}$$

So if we already have $\Pr(\boldsymbol{\pi}''|\boldsymbol{\pi}_{(-j)})$, we can divide it by (1.7) to quickly obtain $\Pr(\boldsymbol{\pi}'|\boldsymbol{\pi}_{(-j)})$. Moreover, we can efficiently calculate (1.7) by storing previous computational results: for instance, the summation over the risk group $R(\boldsymbol{\pi}'_{1:j'})$ shares many elements with the summation over the risk group $R(\boldsymbol{\pi}''_{1:j'})$. Similar ideas can be used to speed up other calculations required for MCEM.

Algorithm 1 Gibbs sampler for mutation orders

Initialize Gibbs step index $k = 1$ and mutation order $\boldsymbol{\pi}^{(0)}$.
for Gibbs sweep index $i = 1, 2, ...$ **do**
  for $j \in \{1, ..., n\}$ **do**
    $\boldsymbol{\pi}_{(-j)} \coloneqq \boldsymbol{\pi}^{(k-1)}_{(-j)}$
    Sample $\boldsymbol{\pi}^{(k)}$ from the distribution $\Pr\left(\boldsymbol{\pi}\middle|\boldsymbol{\pi}^{(k-1)}_{(-j)}\right)$.
    $k \coloneqq k + 1$
  end for
end for

Given the Monte Carlo samples from the E-step, the M-step maximizes the mean log-likelihood of the complete data. Suppose the E-step generates Monte Carlo samples $\boldsymbol{\pi}^{(1)}, ..., \boldsymbol{\pi}^{(E)}$. Then during the M-step, we solve

$$\max_{\boldsymbol{\theta}} \frac{1}{E}\sum_{i=1}^{E}\log\mathcal{L}_c\left(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}^{(i)}; \boldsymbol{\theta}\right) \tag{1.8}$$

using iterative procedures such as gradient ascent.

We use ascent-based MCEM [3] to maintain the monotonicity property of the EM algorithm. Briefly, ascent-based MCEM gives a rule for deciding if the proposed MCEM

estimate at each iteration should be accepted or if the Monte Carlo sample size should be increased. As the number of Monte Carlo samples increases, the standard error of the estimated expected log likelihood decreases. So for a sufficiently large number of Monte Carlo samples, we can ensure that the observed data likelihood increases with high probability.

2.4 Regularization and variable selection

In many cases, it is desirable to model the effects of many features. For instance, Yaari et al. [54] estimate a 5-mer model with 1024 parameters. Estimating the parameters for a per-target model increases the number of parameters by an additional factor of four. If the number of sequences in the dataset is small compared to the number of features, the optimization problem in (1.6) can be ill-posed. For such high-dimensional settings, it is common to use regularization to stabilize our estimates and encourage model structure.

In particular, we may believe that only a small subset of the features affects the mutation rate. Yaari et al. [54] assume that the nucleotides closest to a position have the most significant effect on its mutation rate: for 5-mer motifs with a small number of observations, they estimate its mutation rate using an offset 3-mer motif. In our method, we use the lasso [46] to perform variable selection.

To incorporate the lasso, our estimation procedure requires two steps. The first step maximizes the observed log-likelihood with a lasso penalty and thereby performs variable selection. The second step aims to quantify the uncertainty of our model parameter estimates: we refit the model parameters by maximizing the unpenalized objective and use the confidence intervals for the unpenalized model as an assessment of uncertainty.

In the first step, we split the data into training and validation sets denoted $\mathbf{S}_{\text{obs,train}}$ and $\mathbf{S}_{\text{obs,val}}$, respectively, and maximize the penalized log-likelihood of the training data

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \left\{ \log \mathcal{L}(\mathbf{S}_{\text{obs,train}}; \boldsymbol{\theta}) - \lambda \|\boldsymbol{\theta}\|_1 \right\}, \tag{1.9}$$

where $\lambda > 0$ is a penalty parameter. To solve (1.9), we use a variant of MCEM: the E-step

is the same as before, but we maximize the penalized EM surrogate function during the M-step. The penalized EM surrogate function is simply (1.8) with a lasso penalty:

$$\frac{1}{E}\sum_{i=1}^{E}\log\mathcal{L}_c\left(\mathbf{S}_{\text{obs,train}},\boldsymbol{\pi}^{(i)};\boldsymbol{\theta}\right)-\lambda\|\boldsymbol{\theta}\|_1. \tag{1.10}$$

This can be maximized using the generalized gradient ascent algorithm given in Algorithm 2 [2, 37].

Algorithm 2 M-step via generalized gradient ascent

Initialize $\boldsymbol{\theta}$. Choose a step size $\alpha > 0$.
for iteration $k = 1, 2, ...$ until convergence **do**

$$\boldsymbol{\theta} := \boldsymbol{\theta} + \alpha\nabla_\theta\frac{1}{E}\sum_{i=1}^{E}\mathcal{L}_c(\mathbf{S}_{\text{obs,train}},\boldsymbol{\pi}^{(i)};\boldsymbol{\theta})$$

  for parameter index $j = 1, .., p$ **do**
    $\theta_j := \text{sign}(\theta_j)\max(|\theta_j - \lambda|, 0)$
  end for
end for

We tune the penalty parameter λ in (1.9) by training-validation split. In the typical ideal case, we choose the penalty parameter that maximizes the likelihood of the observed validation data. Unfortunately the likelihood of observed data is computationally intractable. Instead we use the property that, for any $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, the difference between the log-likelihoods of the observed data is bounded below by the difference between the expected log-likelihoods of the complete data

$$\begin{aligned}\log\mathcal{L}(\mathbf{S}_{\text{obs}};\boldsymbol{\theta})-\log\mathcal{L}(\mathbf{S}_{\text{obs}};\boldsymbol{\theta}')\geq\\ \mathbb{E}\left[\log\mathcal{L}_c(\mathbf{S}_{\text{obs}},\boldsymbol{\pi};\boldsymbol{\theta})-\log\mathcal{L}_c(\mathbf{S}_{\text{obs}},\boldsymbol{\pi};\boldsymbol{\theta}')\mid\mathbf{S}_{\text{obs}};\boldsymbol{\theta}'\right],\end{aligned} \tag{1.11}$$

which follows directly from Jensen's inequality. The expectation above is taken with respect to the conditional distribution of the mutation orders $\boldsymbol{\pi}$ given the observed data $\mathbf{S}_{\text{obs}}$ and model parameter $\boldsymbol{\theta}'$. Thus the right-hand side can be estimated by sampling mutation

orders from the Gibbs sampler in Algorithm 1. If the right-hand side of (1.11) is positive, then $\boldsymbol{\theta}$ has a higher log-likelihood than $\boldsymbol{\theta}'$ on the validation set. However, if the right-hand side is negative, we do not know how the two parameters compare.

Our proposal for tuning the penalty parameter, Algorithm 3, is based on (1.11). The algorithm searches across a one-dimensional grid of penalty parameters, from largest to smallest. For consecutive penalty parameters, we estimate the right-hand side of (1.11) to determine if the smaller penalty parameter has a higher observed log-likelihood. We keep shrinking the penalty parameter until the estimate for the right-hand side of (1.11) is negative. Since the check based on (1.11) is conservative, we may end up choosing a penalty parameter that is slightly larger than desired. Nonetheless, our simulation results suggest that this procedure works well in practice.

Algorithm 3 can be easily extended to incorporate multiple training-validation splits such as in k-fold cross-validation: we average the estimates of the right-hand side of (1.11) across the training-validation splits and stop shrinking the penalty parameter if the average is negative. After selecting a penalty parameter, we obtain the final parameter support from the k-fold procedure by refitting the penalized model on the whole training set.

Algorithm 3 Tuning penalty parameters via training-validation split

Consider a grid of penalty parameters $\lambda_1 > ... > \lambda_K \geq 0$.
Initialize $\lambda_{\text{best}} \coloneqq \lambda_1$. Fit λ_1 to get $\hat{\boldsymbol{\theta}}_{(1)}$.
for iteration $i = 2, ..., K$ **do**
  Solve (1.9) with $\lambda \equiv \lambda_i$ to get $\hat{\boldsymbol{\theta}}_{(i)}$.
  Estimate via Monte Carlo

$$\eta = \mathbb{E}\left[\log \mathcal{L}_c\left(\mathbf{S}_{\text{obs,val}}, \boldsymbol{\pi}; \hat{\boldsymbol{\theta}}_{(i)}\right) - \log \mathcal{L}_c\left(\mathbf{S}_{\text{obs,val}}, \boldsymbol{\pi}; \hat{\boldsymbol{\theta}}_{(i-1)}\right) \mid \mathbf{S}_{\text{obs,val}}; \hat{\boldsymbol{\theta}}_{(i-1)}\right]. \quad (1.12)$$

  if $\eta > 0$ **then**
    $\lambda_{\text{best}} \coloneqq \lambda_i$
  else
    break
  end if
end for

Now we move on to the second step where our goal is to quantify the uncertainty of our estimated model parameters. Unfortunately, estimating confidence intervals after model selection is a difficult problem, even in the much simpler case of linear models [12]. Hence some papers use the approach of fitting a penalized model, refitting an unpenalized model based on the selected variables, and then using the confidence intervals generated using traditional inference procedures for unpenalized models [31, 24]. We proceed in the same manner: we refit the model by maximizing the unpenalized observed log-likelihood (1.6) of the entire dataset with respect to the selected variables and constraining the others to zero; then we construct confidence intervals for the unpenalized model, ignoring the fact that we have already peeked at the data in the first step. Though these confidence intervals are asymptotically valid only under very restrictive conditions, they provide some measure of the uncertainty of our fitted parameters; we show via simulation in Section 3 that the coverage of these intervals is close to nominal. To highlight that these intervals are not truly confidence intervals, we refer to them as uncertainty intervals, where 100(1 - α)% uncertainty intervals are constructed using intervals with nominal 100(1 - α)% coverage.

To obtain these uncertainty intervals, we calculate the standard error of our estimates using an estimate of the observed information matrix. Louis [34] shows that the observed information matrix is related to the complete data likelihood via the following identity:

$$\begin{aligned} I\left[\boldsymbol{\theta} \mid \mathbf{S}_{\text{obs}}\right] = &-\mathbb{E}\left[\nabla^2_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) \mid \mathbf{S}_{\text{obs}}; \boldsymbol{\theta}\right] \\ &- \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) \left(\nabla_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta})\right)^\top \mid \mathbf{S}_{\text{obs}}; \boldsymbol{\theta}\right] \\ &+ \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) \mid \mathbf{S}_{\text{obs}}; \boldsymbol{\theta}\right] \mathbb{E}^\top \left[\nabla_{\boldsymbol{\theta}} \log \mathcal{L}_c(\mathbf{S}_{\text{obs}}, \boldsymbol{\pi}; \boldsymbol{\theta}) \mid \mathbf{S}_{\text{obs}}; \boldsymbol{\theta}\right]. \end{aligned}$$

Therefore we can estimate the observed information matrix using samples from the final MCEM iteration and then invert it to obtain uncertainty intervals.

Finally, one caveat of our method is that the two-step procedure is not guaranteed to give estimates of standard errors/uncertainty intervals: The first step of our procedure may choose a penalty parameter such that the estimated information matrix in the second step

is not positive definite. We see this behavior in a small number of simulations in Section 3, though we do not observe such behavior in our data analysis. To avoid this issue, we suggest combining k-fold cross-validation with Algorithm 3 and use the average estimate of the lower bound (1.12) from each of the k folds to tune the penalty parameter.

Our GPLv3-licensed Python implementation of `samm` is available at http://github.com/matsengrp/samm. The repository includes code used for generating plots in this manuscript, as well as a tutorial for how to run `samm`. All output from Sections 3 and 4 is available on http://zenodo.org/record/1321330 with DOI `10.5281/zenodo.1321330`.

2.5 Examples

By varying the motif dictionary $\mathcal{M}$, our procedure can fit different models of the mutation process. In this section, we list some example models that can be fit using our procedure and discuss the interplay between the motifs included in $\mathcal{M}$ and our feature-selection step. In the simplest case, analogous to existing work [54, 8], we can estimate a "k-mer model" (where k is odd) by letting

$$\mathcal{M} = \left\{(m, (k+1)/2) : m \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^k\right\}. \tag{1.13}$$

The lasso would encourage setting elements in $\boldsymbol{\theta}$ to zero, which means that these k-mer motifs would have the same baseline risk of experiencing a mutation.

In practice, instead of modeling only the effects of k-mers for a fixed k, we may believe that the hazard rate for a position is affected more by positions closer to it. In this case, we can model the effect of z-mers of varying length, e.g., $1, 3, ..., k$-mer motifs, with

$$\mathcal{M} = \{(m, (z+1)/2) : m \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^z, z \in 1, 3, \ldots k\}. \tag{1.14}$$

We refer to this model as "hierarchical", as the elements in $\mathcal{M}$ relate to each other in a nested fashion. By including motifs in a hierarchical fashion, the lasso penalty encourages z-mers

with the same inner $(z-2)$-mer to share the same mutation rate. This model formalizes the intuition used by Yaari et al. [54]: they try to estimate the mutation rates of 5-mers but fall back to using a 3-mer sub-motif if that 5-mer does not appear enough times in the data.

As mentioned before, we can add offset motifs to our motif dictionary as previous work suggests the mutation rates depend on upstream or downstream motifs [42, 38, 54]. For instance, one can include all the offset motifs that overlap the mutating position in the motif dictionary. We refer to such models as offset k-mer models.

Finally, we can model the hazard rate of motifs mutating to different target nucleotides as in (1.4). We parameterize the model using $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{\mathtt{N}}$ for $\mathtt{N} \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ since the penalized per-target model

$$\begin{aligned} \arg \max_{\boldsymbol{\theta}, \boldsymbol{\theta}_{\mathtt{N}}: \mathtt{N} \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} & \log \mathcal{L}\left(\mathbf{S}_{\text{obs,train}};\ \boldsymbol{\theta}, \{\boldsymbol{\theta}_{\mathtt{N}} : \mathtt{N} \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}\}\right) \\ & - \lambda \left(\|\boldsymbol{\theta}\|_1 + \sum_{\mathtt{N} \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} \|\boldsymbol{\theta}_{\mathtt{N}}\|_1 \right) \end{aligned} \tag{1.15}$$

will encourage hazard rates for the different target nucleotides to be the same if they share the same motif.

Many of these example models are overparameterized in order to obtain some desired sparsity pattern. Such overparameterized models may have singular information matrices during the refitting procedure. However this is not a problem since we are truly interested in the confidence intervals for the parameters $\boldsymbol{\theta}_{\text{agg}} = \boldsymbol{A}\boldsymbol{\theta}$ associated with the simple k-mer model, where $\boldsymbol{A}$ is a matrix that aggregates hierarchical motifs into a single k-mer. Since this aggregate k-mer model is identifiable, we can get uncertainty intervals for $\boldsymbol{\theta}_{\text{agg}}$: we calculate the pseudo-inverse $\boldsymbol{I}^{-}$ of the (estimated) information matrix and then use $\boldsymbol{A}\boldsymbol{I}^{-}\boldsymbol{A}^{\top}$ to get an estimate of the covariance matrix of $\boldsymbol{\theta}_{\text{agg}}$.

3 *Simulation results*

We now present a simulation study of our procedure, including a comparison to the current state-of-the-art method `SHazaM` [54, version 0.1.8] and the logistic regression approach in Section 2.1.

3.1 Understanding the effect of various models and settings

We fit the following three models to simulated data:

- 3-mer model: the hazard rate modeled by (1.3) with motif dictionary (1.13) where $k = 3$,
- 3-mer per-target model: the hazard rate modeled by (1.4) with motif dictionary (1.13) where $k = 3$,
- 2,3-mer model: the hazard rate modeled by (1.3) with motif dictionary

$$\mathcal{M} = \left\{(m, j') : m \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^2, j' \in \{1, 2\}\right\} \cup \left\{(m, 2) : m \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}^3\right\}.$$

To understand how dataset composition affects the performance of our procedure, we simulate different datasets by varying the sample sizes, sparsity levels, and effect sizes.

We generate the true $\boldsymbol{\theta}^*$ according to the same hierarchical structure as each model we consider. Let the model parameters corresponding to the motif m be θ^*_m and corresponding to motif m with target nucleotide `N` be $\theta^*_{m\to\mathtt{N}}$. To obtain the desired sparsity level, we randomly select a portion of the parameters to zero out. For per-target parameters, instead of setting the probability of mutating to `N` to zero, we set $\theta^*_{m\to\mathtt{N}}$ to $\log 1/3$ for all possible values of `N`, indicating no mutation preference. We scale the model parameters appropriately to control the effect size.

Our goal with these simulations is to obtain synthetic data that reflects different possible settings one may encounter when analyzing experimental data. We use the experimental

data in Cui, Di Niro, Vander Heiden, Briggs, Adams, Gilbert, O'Connor, Vigneault, Shlomchik, and Kleinstein [8] analyzed in Section 4 as a template and alter various underlying properties of this dataset to simulate data that replicates what typical real-world datasets look like. We first generate naïve sequences using `partis`[2] [39, 40] by drawing a set of genes from the IMGT database [33] and simulating an observation frequency for each. Antibodies are composed of two units, a heavy and a light chain. Further, light chains can be classed as either κ or λ depending on where the encoded sequence came from in the genome. Both mice and humans have antibodies structured in this way. We select only κ-light chain mouse BCRs for our simulation, as this reflects our experimental data in Section 4. To generate the true $\boldsymbol{\theta}^*$ parameters, we randomly draw values from the mouse somatic hypermutation targeting model `MK_RS5NF` of Cui et al. [8]; we refer to these parameters as $\boldsymbol{\theta}^*_{\texttt{MK}}$. The `MK_RS5NF` model is a collection of mutabilities and substitution probabilities from a 5-mer fit to κ-light chain mouse BCR data.

The average length of the naïve sequences is around 290 nucleotides. We use the survival model to mutate between 1% and 5% of the positions of each naïve sequence, obtaining a collection of simulated BCR sequences. Conditional on their naïve sequences, BCR sequences mutate independently.

We vary sparsity, effect size, and sample size as follows. We generate the true $\boldsymbol{\theta}^*$ parameters with 25%, 50%, and 100% non-zero elements. We also consider different effect sizes by scaling $\boldsymbol{\theta}^*$ such that its variance is 50%, 100%, and 200% of the variance of the values in $\boldsymbol{\theta}^*_{\texttt{MK}}$. Finally, we fit the model using 100, 200, and 400 mutated BCR sequences. For the main manuscript, we report the simulation settings where we vary one simulation setting and fix the other settings to the middle value (e.g., we vary number of samples but keep the effect size at 100% and the number of non-zero elements at 50%); we report the result from running one hundred replicates for each setting. For the remaining possible settings, as each separate model fit takes on average an hour to complete, we run only ten replicates

[2]Version 0.12.0: http://git.io/fNvOx

and report the results in the Appendix Section A.3 [18].

To determine the optimal penalty parameter for `samm`, we split the data by gene subgroups, an externally-defined categorization that groups genes that share at least 75% identity at the nucleotide level [32], reserving 20% of subgroups for validation and the remainder for training. Splitting by gene subgroup ensures that the training and validation sets look sufficiently different; otherwise the sequences in the validation set look nearly identical to those in the training set, and we select a penalty parameter that is too small. We then apply Algorithm 3 over a decreasing sequence of penalty parameter-values $10^{-j}, 10^{-(j+0.5)}, 10^{-(j+1)}, \ldots$. The starting value for the sequence of penalty parameter values was pre-tuned so that we use a smaller j for smaller effect sizes and sample sizes. In particular, we chose $j = 1$ if effect size is 50% or sample size is 100; $j = 2$ if the effect size is 200% or sample size is 400; and $j = 1.5$ otherwise.

For each penalty parameter-value, we run a maximum of ten MCEM iterations. Mutation orders are sampled from each Gibbs sampler run every eight sweeps, after an initial burn-in period of 16 Gibbs sweeps. For each E-step, we sample four mutation orders and continue to double the number of sampled mutation orders if the proposed estimate is not accepted by ascent-based MCEM. Once we have an estimate of the support of our model, we refit an unpenalized model to obtain uncertainty estimates. We run MCEM until the model has converged and the variance estimates of the estimated model parameters are all nonnegative.

We assess the performance of our procedure using three measures. These performance metrics are all calculated with respect to the aggregate model since our complete model is overparameterized by design. We calculate the relative $\boldsymbol{\theta}$ error, defined as $\|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|_2 / \|\boldsymbol{\theta}^*\|_2$, to see how close the estimated parameters are to the true parameter $\boldsymbol{\theta}^*$. We also calculate Kendall's tau coefficient to see how well our procedure ranks the motifs in terms of their mutabilities. Finally we calculate the coverage of our approximate 95% uncertainty intervals. We define the average coverage as the proportion of aggregate model parameters where

the uncertainty intervals covered the true value. The coverage calculations only involve aggregate parameters not zeroed out by our models.

These simulations demonstrate that our estimation procedure performs as expected (Figure 1.3). As the sample size and effect size increase, the relative $\boldsymbol{\theta}$ error decreases and the rank correlation increases. On the other hand, as the percent of non-zero elements increases, both the relative $\boldsymbol{\theta}$ error and rank correlation increase. The error increases because there are more parameters to estimate. The increase in rank correlation is likely an artifact of how the metric is calculated, as Kendall's tau removes ties from the calculations. In particular, as the percent of non-zero elements increases, the number of ties in the data decreases, so the rank correlation seems to increase. In all the plots, we see that the 3-mer per-target model tends to be the most difficult to estimate. This is expected as it contains 256 parameters whereas the 3-mer model only has 64 parameters.

Our simulations show that the coverages for the 3-mer and the 2,3-mer models are close to 95%, which is surprising as our uncertainty intervals ignore the double-peeking issue (Figure 1.3). Zhao, Shojaie, and Witten [57] explain why this procedure might work: under certain assumptions, the variables selected by the lasso are deterministic with high probability, so using the lasso to select variables does not really constitute as peeking at the data twice.

However, the coverage of the 3-mer per-target is much lower, dropping below 80% in certain settings (Figure 1.3). We suspect that the low coverage is mainly due to a lack of data, as the coverage improves with the number of samples. When there is a small number of samples compared to the number of parameters, our method may only provide a reasonable ranking of how mutable the motifs are but may not provide good estimates and uncertainty intervals.

Across the 2700 simulation runs, there were twenty where the estimated information matrices were not positive definite and therefore uncertainty intervals cannot be calculated (Table 1.5). We believe that this occurs when the selected penalty parameter is too small;

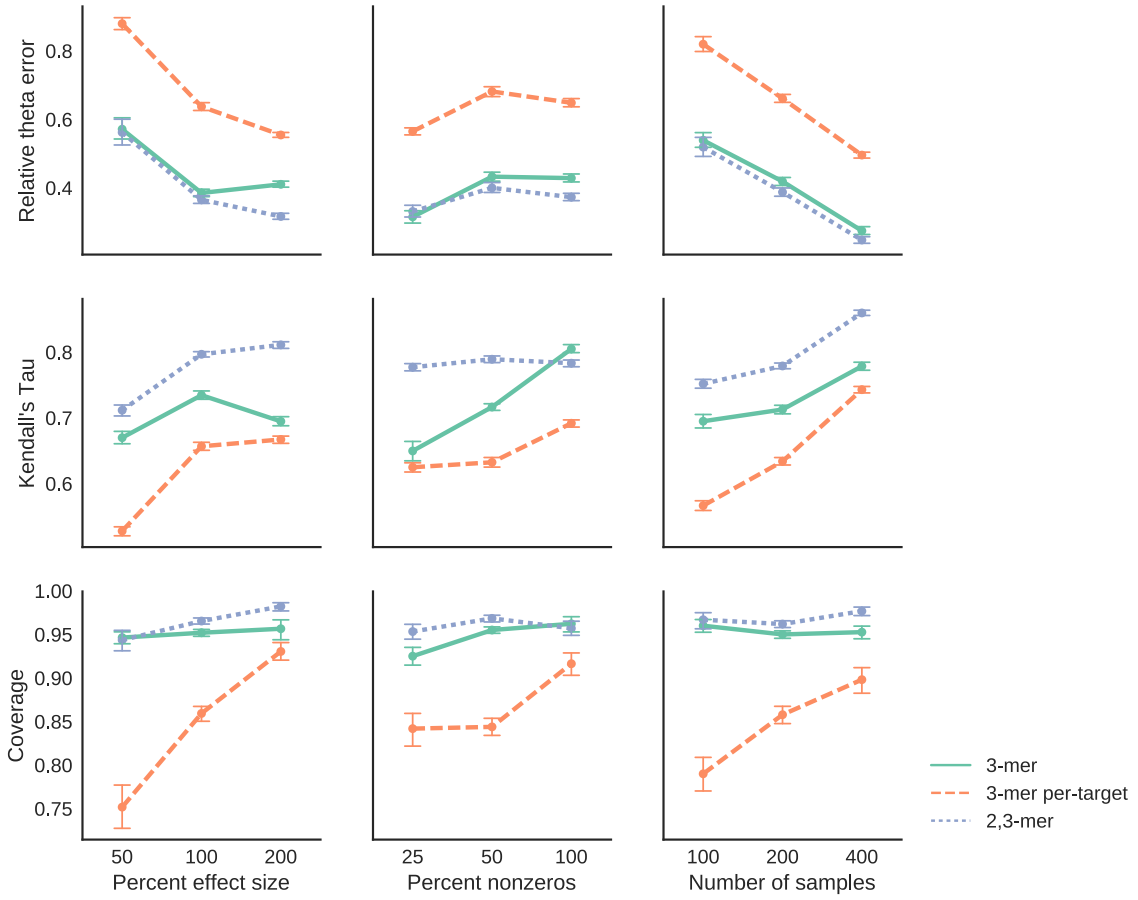


Figure 1.3: Relative error, correlation and coverage under different simulations settings for 3-mer, 3-mer per-target and 2,3-mer models.

for small penalty parameters, the support of the fitted model becomes too large. In this case, when we refit the model with no penalty parameter the problem is ill-posed and therefore the estimated information matrix is not positive definite. To avoid this issue, we recommend using k-fold cross-validation in practice, rather than just a training/validation split. (We use 5-fold cross-validation for the real data analysis and do not run into this issue.)

3.2 Method comparisons

In this section, we compare the performance of `samm` to `SHazaM` and penalized logistic regression on simulated data. Since `SHazaM` only estimates the effect of 5-mer motifs, we simulate data such that the mutation rate at a specific site depends on the 5-mer centered at that position and the target nucleotide. We simulate 2000 BCR sequences from 4 mice. For each mouse, we generate a separate set of naïve sequences using the same procedure as in Section 3.1. From these naïve sequences, we simulate the mutation process independently to generate BCR sequences. We use two methods to simulate the mutation process:

- **Survival Simulation**: We generate model parameters $\boldsymbol{\theta}$ by resampling the values from $\boldsymbol{\theta}_{\texttt{MK}}$ into a 3,5-mer per-target model structure. We then mutate the naïve sequences according to the survival model.

- `SHMulate` **Simulation**: We use $\boldsymbol{\theta}_{\texttt{MK}}$ and mutate the naïve sequences using the `SHMulate` function in the `SHazaM` package [54, 20]. `SHMulate` simulates the mutation process using a procedure that is similar to a survival model. However the exact calculations differ somewhat (e.g. it does not allow the mutation process to create stop codons).

`SHazaM` should have an advantage in the `SHMulate` simulations since the $\boldsymbol{\theta}_{\texttt{MK}}$ was estimated using `SHazaM` on a separate BCR dataset and `SHazaM` uses some prior assumptions about the model structure. In particular, `SHazaM` assumes that 5-mer motifs that share certain upstream/downstream nucleotides have similar mutabilities. The simulations are run until 1–5% of the sequence is mutated. This mutation rate is on the low end for affinity-matured BCR sequences [compare the 3× higher rate in 22], giving `SHazaM` and logistic regression a slight edge since the mutation rates will not change for most positions with accumulation of BCR mutations.

We fit a 3,5-mer per-target `samm` model using the same procedure as in Section 3.1. Using the same motif dictionary, we also fit a 3,5-mer per-target logistic regression model using

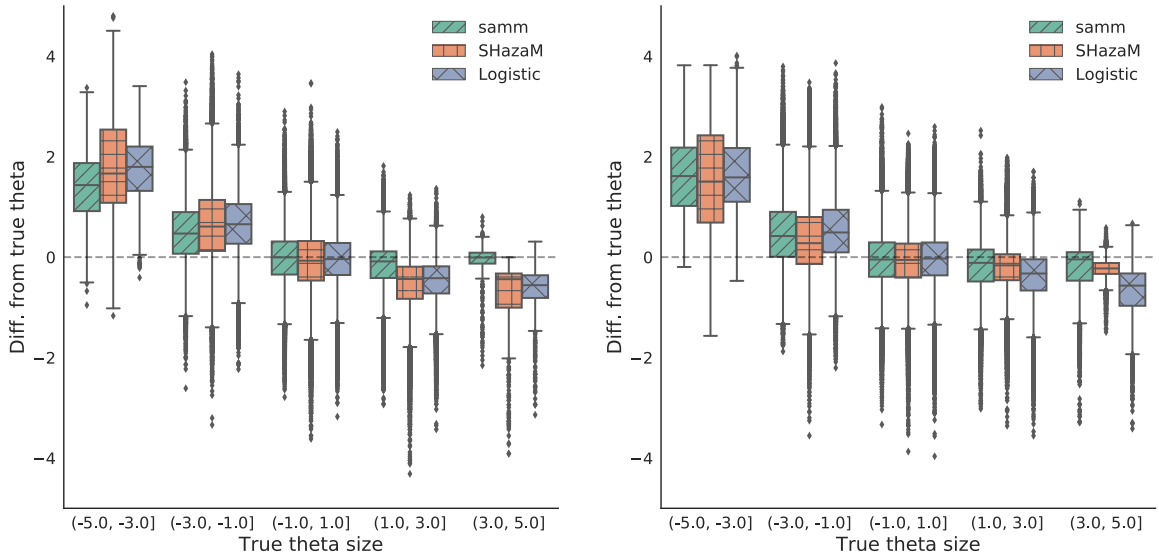
Table 1.1: Comparison of `samm`, `SHazaM`, and penalized logistic regression given 2000 simulated B-cell receptor sequences from 4 mice. Relative $\boldsymbol{\theta}$ error and Kendall's tau computed separately for each of the 100 replicates. Monte Carlo standard errors calculated over these 100 estimates are given in parentheses.

Simulator	Model	Relative $\boldsymbol{\theta}$ error	Kendall's tau
survival model	`samm`	0.571 (0.002)	0.630 (0.001)
	`SHazaM`	0.731 (0.002)	0.507 (0.002)
	logistic	0.611 (0.002)	0.596 (0.001)
`SHMulate`	`samm`	0.478 (0.001)	0.689 (0.001)
	`SHazaM`	0.489 (0.001)	0.690 (0.001)
	logistic	0.499 (0.002)	0.677 (0.001)

logistic regression with a lasso penalty. We measure model performance by the relative $\boldsymbol{\theta}$ error and rank correlation over 100 simulation replicates.

Our method implemented in `samm` significantly outperforms logistic regression and `SHazaM` in both scenarios (Table 1.1), even though `SHazaM` should have an advantage when we simulate data using a dense model from `SHMulate`. Logistic regression and `SHazaM` tended to produce similar estimates, though logistic regression tended to do better when we simulated using the survival model and `SHazaM` tended to do better when we used the `SHMulate` model.

We present the results of model fitting in more detail in Figure 1.4. For negative $\boldsymbol{\theta}$ values, all the methods are biased towards zero, though `SHazaM` and logistic regression tend to be more so. For positive $\boldsymbol{\theta}$ values, `samm` is nearly unbiased while `SHazaM` and logistic regression are somewhat biased towards zero. The methods probably have trouble estimating negative values since we only observe a small number of mutations per sequence and the data is more informative for finding motifs with high mutation rates rather than those with low mutation rates. Based on results from Section 3.1, we expect the bias of `samm` to shrink as the number of training observations increases.



(a) Survival simulation

(b) `SHMulate` simulation

Figure 1.4: Boxplots of the differences between median-centered fitted and true $\boldsymbol{\theta}$ values for samm (left), `SHazaM` (middle), and logistic regression (right).

4 Data analysis

We fit models to the BCR sequence data obtained from a vaccination study of four transgenic mice published in [8]. In this experimental setting, the substitutions present in the κ-light chain sequences are unlikely to be affected by natural selection on BCR function. Thus we restrict our analysis to only κ-light chain data in order to estimate somatic hypermutation rates, rather than a combination of somatic hypermutation and selection [53, 35, 55]. A single naïve sequence can give rise to many different B-cell receptors by somatic hypermutation, forming a so-called "clonal family" which may have varying levels of shared evolutionary history. We use `partis` [39] to assign mutated sequences to clonal families and infer the most likely naïve sequence in each family. In both the sequencing and the clonal family inference there is the possibility of error propagation; we begin our analysis by assuming BCRs are accurately sequenced and assigned to clonal families. The resulting data has the composition shown in Table 1.2. To mitigate double-counting mutations, we sample a

single mutated sequence from each clonal family. Though this discards a lot of the data, we believe this gives more accurate estimates than other approaches that try to use all the data or estimate mutation history; we analyze this issue in more depth in Section A.3 in the Appendix [18].

Table 1.2: Statistics of processed κ-light chain data from Cui et al. [8]. `SHazaM` uses all sequences while `samm` samples a single sequence from each clonal family. We filter sequences with indels in all analyses. There are fewer clonal families in the sampled sequences as `samm` filters out sequences with no mutations.

	All sequences	Sampled sequences
Number of mutated sequences	15,025	2,429
Number of clonal families	2,565	2,429
Median mutated sequence length	282	282
Average mutation frequency (%)	2.32	2.17
Number of 5-mers in naïve sequences	1,014	967

We fit a 3,5-mer model using `samm` using the same settings as before (Figure 1.5), though with 5-fold cross-validation to determine the optimal parameter support. The $\boldsymbol{\theta}$ estimate has a block-like and 4-fold-repetitive pattern because many 5-mer motifs were zeroed out during the lasso step. The 95% uncertainty intervals suggest that many motifs have a marked nonzero effect.

Our model recovers many of the well-known "hot" (more mutable) and "cold" spots (less mutable k-mers), which are denoted by the red, blue, and green bars in Figure 1.5. Hot/cold motifs are typically denoted with an underline indicating which position is mutating and represented by degenerate bases $\mathtt{W} = \{\mathtt{A}, \mathtt{T}\}$, $\mathtt{R} = \{\mathtt{A}, \mathtt{G}\}$, $\mathtt{Y} = \{\mathtt{C}, \mathtt{T}\}$, $\mathtt{S} = \{\mathtt{C}, \mathtt{G}\}$, $\mathtt{N} = \{\mathtt{A}, \mathtt{G}, \mathtt{C}, \mathtt{T}\}$. We confirm that many highly mutable 5-mer motifs match the classical hot spot motif $\mathtt{WR}\underline{\mathtt{C}}$ and its reverse complement $\underline{\mathtt{G}}\mathtt{YW}$ (since the mutation process can happen on either DNA strand) [41]. We also confirm that many less mutable 5-mer motifs match the canonical cold spot $\mathtt{SY}\underline{\mathtt{C}}/\underline{\mathtt{G}}\mathtt{RS}$ [54]. For example, one of the 5-mers we estimate to have high mutability ($\theta = 1.688$) is $\mathtt{AA}\underline{\mathtt{G}}\mathtt{CT}$, which is of the form $\mathtt{NN}\underline{\mathtt{G}}\mathtt{YW}$ and ends with the 3-mer $\underline{\mathtt{G}}\mathtt{YW}$.

As C is an example of a Y nucleotide and T is an example of a W, A A G C T is an example of the hot spot motif GYW.

Our model also reveals shortcomings with the current hot and cold spot definitions. Our estimates show significant variability in the mutabilities of motifs, even if they contain the same hot or cold spot motif. For instance, in the established literature the A TGG C motif is considered to be a cold spot since it is of the form GRS. We estimate its θ value to be very large ($\theta = 2.206$) relative to the other θ values, suggesting that it is actually a hot spot. We also see SHazaM estimates all motifs of the form C C CNN to have negative mutability, and these are examples of the known cold spot SYC. Estimates from samm show C C CG N has a positive mutability even though it is also of the form SYC, indicating the inner 3-mer C CG may increase mutation rate more than the two C nucleotides to the left of the mutating position. In addition, the classic hot spots with a central T nucleotide actually had very low mutability estimates; this suggests that using the well-known WA/TW to identify hot spots may not be appropriate.

Finally, our model suggests that samm can be used to discover new hot and cold spots. For example, consider motifs with the central base C mutating. We find that the mutabilities of the 5-mer C A CG C and of the 3-mers GCG, GCT, A CT, and A CG are all higher than any motif of the form W R C. As each of these motifs are of the form N R C, this indicates the R nucleotide immediately preceding the mutating C may affect mutation rate more than the W nucleotide two bases away. A well-defined inferential procedure to determine significant collections of hot and cold spots with ample support from the data will require additional future work.

For comparison, we fit SHazaM on the same data without sampling a single sequence from each clonal family, as was done by Yaari et al. [54]. We also fit the logistic model on the same data as samm. All models use the data to determine the degrees of freedom to use in fitting $\boldsymbol{\theta}$, resulting in the number of unique $\boldsymbol{\theta}$ values fit to be less than the saturated model size of 1024 for a 5-mer model. SHazaM estimated 1015 unique θ values out of a maximum of 1024 while samm only estimated 137 unique θ values and logistic estimated

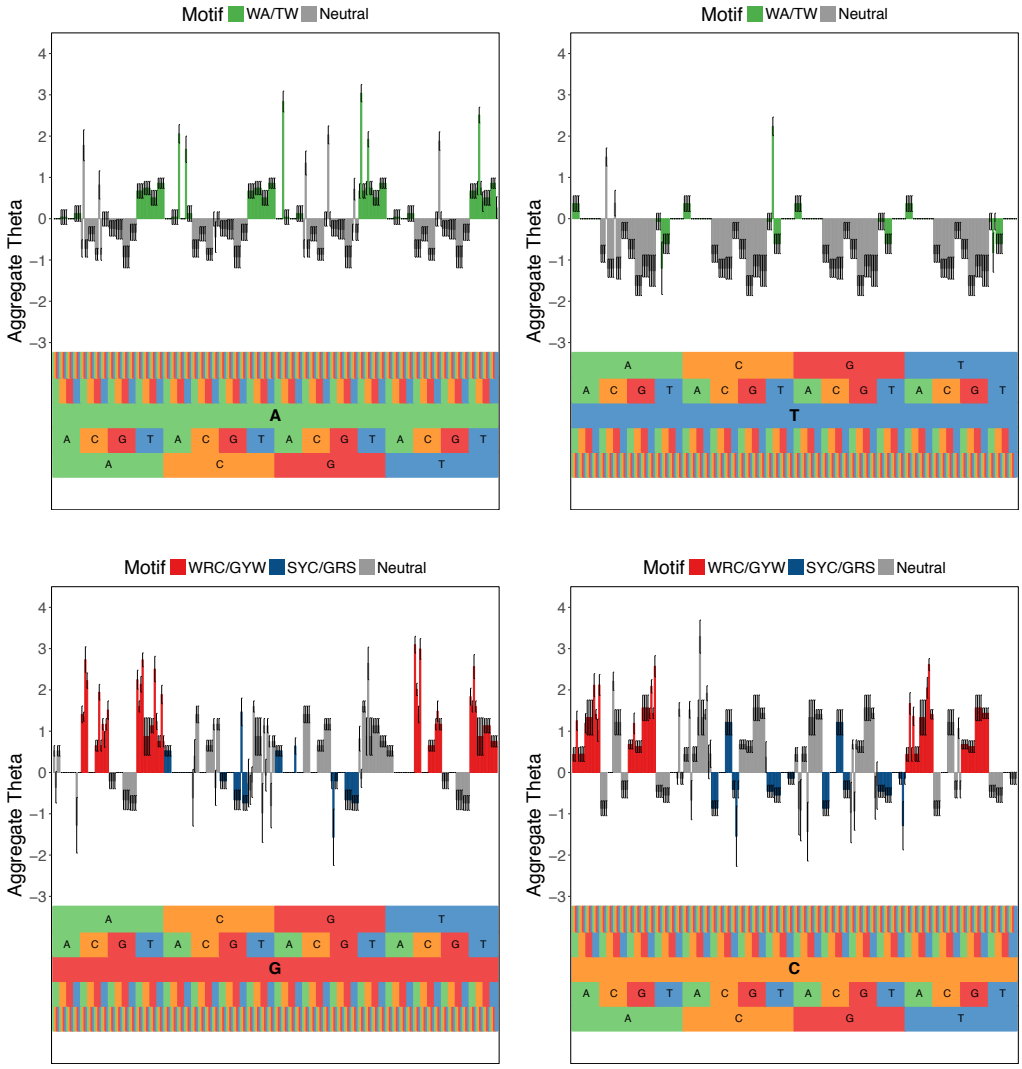
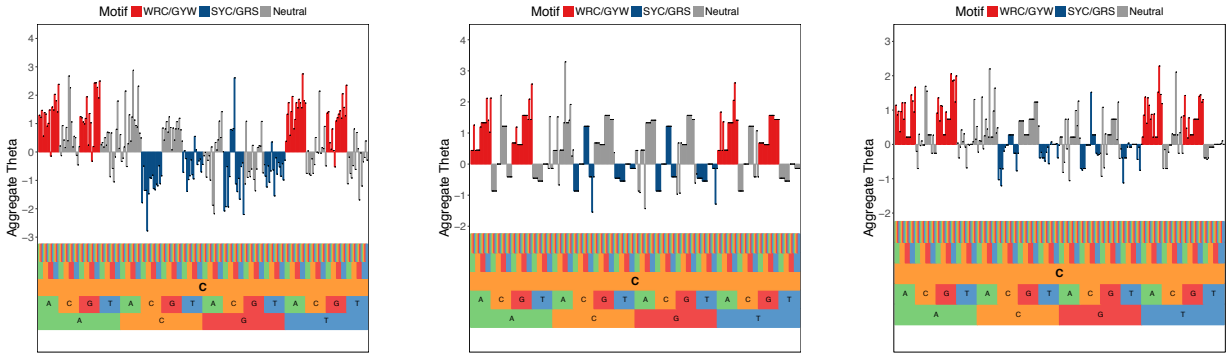


Figure 1.5: Estimated somatic hypermutation model for mouse light chains using `samm` for 5-mer motifs centered on the bases `A` (top left), `T` (top right), `G` (bottom left), and `C` (bottom right). The motif corresponding to an x-axis position can be read from bottom to top. Plots depict the estimated aggregate θ of 5-mer motifs after estimating the model for a 3,5-mer model and aggregating estimates using the procedure outlined in Section 2.5. A negative value means a reduced mutation rate relative to the baseline hazard, whereas a positive means an enhancement. Well-known hot spots, `WRC`/`GYW` and `WA`/`TW`, are colored red and green, respectively. The well-known cold spot `SYC`/`GRS` is colored blue. All other motifs are colored grey. The 95% uncertainty intervals for the estimates are depicted by black lines in the center of each bar.



(a) `SHazaM`: 1015 unique θ values (b) `samm`: 137 unique θ values (c) logistic: 485 unique θ values

Figure 1.6: A comparison of fitted aggregate θ values from `SHazaM` (left), `samm` (middle), and logistic regression (right) for 5-mer motifs with central base `C`. The `samm` fit is the same as in Figure 1.5. Both `samm` and logistic are 3,5-mer fits aggregated into 5-mer models. `samm` and logistic tend to fit more parsimonious models compared to `SHazaM`, so the left plot looks more "spiky" than the middle and right ones. `samm` produces the most parsimonious fits among the three methods.

485. Visually, estimates from the three models look similar, with similar hot- and cold-spots, though `SHazaM` is more "spiky" than `samm` and logistic (Figure 1.6). In terms of model interpretability, `samm` or logistic regression seem to be preferable to `SHazaM` as they produce much more parsimonious models. The logistic model seems to fit a model that is intermediate to `samm` and `SHazaM` in terms of parameter support.

Ideally, we would be able to compare the different methods in terms of their observed data likelihood on a test set. However due to methodological difficulties and incompatibilities of the methods, we were unable to come up with a concrete way to compare the methods. In particular, `SHazaM` is not a likelihood-based method. In addition, the observed data likelihood for `samm` is computationally intractable, which makes it difficult to compare to other likelihood-based methods. We hope to come up with a good solution for assessing `samm` on real-world data in the future.

5 Discussion

We have modeled somatic hypermutation of BCR sequences using Cox proportional hazards. Due to the context-dependence of mutation rates, we must take into account the unknown mutation order to compute the full likelihood. To deal with this missing data, we used MCEM, where we marginalize over the possible mutation orders using Markov chain Monte Carlo. Unlike current methods, our regression framework can model the effect of arbitrary features, such as varying motif lengths and sequence positions. We use the lasso to perform feature selection and stabilize our estimates in high-dimensional settings. One can easily extend this approach to use other sparsity-inducing penalties to reflect other prior beliefs about the model structure. We show that `samm` achieves better performance than the state-of-the-art method under a variety of simulation settings.

There are a few limitations with our current method. We currently subsample our data significantly to ensure our training set is composed of independent observations. This would not be necessary if we were able to perform accurate phylogenetic ancestral sequence estimation using context-sensitive models. In addition, our method returns "uncertainty" intervals rather than confidence intervals since there are no guarantees on their nominal coverage. Simulations show that our uncertainty intervals are close to their nominal coverage levels if there is a sufficient amount of data (Figure 1.3), but better methods may be available.

While the present analysis only considers sequence context, other biologically-motivated features may be just as informative: nucleotide position, proximity to other contexts, etc. By incorporating other types of features into the model, we may be able to help verify or find problems with the currently accepted model of somatic hypermutation [36].

Finally, our model can be used in other contexts to model other biological processes. For instance, our method could be used to model the rate of single-nucleotide polymorphisms [1] and transcription-factor binding [58].

A Appendix

A.1 Proof of marginal likelihood

We now prove the statement in Section 2 that the marginal likelihood of $\boldsymbol{\theta}$ is given by (1.5) and only depends on the mutation order $\boldsymbol{\pi}_{1:n}$.

Proof. Suppose the mutation times are observed, so u_i is the time of the ith mutation. Then the conditional probability of observing a mutation order $\boldsymbol{\pi}_{1:n}$ given mutation times $\boldsymbol{u} = (u_1, ..., u_n)$ can be written as

$$\Pr\left(\boldsymbol{\pi}_{1:n}|\boldsymbol{u};\boldsymbol{\theta},h_0\right) = \prod_{i=1}^{n} \Pr\left(\pi_i|\boldsymbol{\pi}_{1:i-1},u_{i-1},u_i;\boldsymbol{\theta},h_0\right) \tag{1.16}$$

$$= \prod_{i=1}^{n} \frac{\Pr\left(\pi_i,u_i|\boldsymbol{\pi}_{1:i-1},u_{i-1};\boldsymbol{\theta},h_0\right)}{\sum_{q\in R(\boldsymbol{\pi}_{1:i-1})}\Pr\left(q,u_i|\boldsymbol{\pi}_{1:i-1},u_{i-1};\boldsymbol{\theta},h_0\right),} \tag{1.17}$$

where the conditional probability of observing a mutation in position q at time u_i is defined as

$$\Pr\left(q,u_i|\boldsymbol{\pi}_{1:i-1},u_{i-1};\boldsymbol{\theta},h_0\right) \tag{1.18}$$

$$= h_0(u_i)\exp\left(\boldsymbol{\theta}^\top\psi_q(S(\boldsymbol{\pi}_{1:i-1}))\right)\times \tag{1.19}$$

$$\exp\left(-\sum_{q'\in R(\boldsymbol{\pi}_{1:i-1})}\exp\left(\boldsymbol{\theta}^\top\psi_{q'}(S(\boldsymbol{\pi}_{1:i-1}))\right)\int_{u_{i-1}}^{u_i}h_0(t)dt\right). \tag{1.20}$$

Notice that in (1.18), the terms $h_0(u_i)$ and (1.20) do not depend on q. So plugging (1.18) into (1.17), these two terms cancel and we get

$$\Pr\left(\boldsymbol{\pi}_{1:n}|\boldsymbol{u};\boldsymbol{\theta},h_0\right) = \prod_{i=1}^{n}\frac{\exp(\boldsymbol{\theta}^\top\psi_{\pi_i}(S(\boldsymbol{\pi}_{1:i-1})))}{\sum_{q\in R(\boldsymbol{\pi}_{1:i-1})}\exp(\boldsymbol{\theta}^\top\psi_q(S(\boldsymbol{\pi}_{1:i-1})))}. \tag{1.21}$$

Since the conditional probability of the mutation order does not depend on mutation times $\boldsymbol{u}$, then the marginal probability of the mutation order $\Pr(\pi_{1:n};\boldsymbol{\theta},h_0)$ is also equal to (1.21).

□

A.2 Pre-processing data

If we are interested in modeling the effect of k-mer motifs on the hazard rate where $k > 1$, then the positions at the ends of the B-cell receptor sequences must be properly handled. The issue is that the positions at the ends might not have enough neighboring nucleotides to fully construct a k-mer motif.

In order to deal with this issue, we first preprocess our data by trimming the two ends of the BCR sequences until the ends of the naïve and mutated sequences are the same. We then assume that these end positions are fixed and not part of the mutation process.

For example, if we are interested in modeling how 3-mer motifs affect the mutation rate of the center position, we need to handle the special case of the two positions at the ends of the sequence. Given a naïve BCR sequence and its associated mutated sequence, we trim away the positions at the ends of both sequences until the first and last positions are the same. If our trimmed sequence is of length p', we suppose that only positions 2 through $p' - 1$ can undergo mutation. We can now apply our estimation method since all positions use the same feature vector mapping.

A.3 Other simulations

Reconstructing mutation history

Since most clonal families contain multiple sequences, including all sequences without reconstructing the shared mutation history within each family can introduce bias by considering some mutations more than once. To overcome this bias, we consider two approaches: we can either attempt to estimate this history using standard methods, or we can randomly sample a single sequence from each clonal family. For the former case, to date, there are no methods that incorporate context-specific mutation models; we introduce one standard and useful approach to consider for a single clonal family.

Assume we have a collection of nucleotide sequences that have mutated away from a known naïve sequence. In time, as we mutate away from this naïve sequence, a series

Table 1.3: Statistics on reconstructing $\boldsymbol{\theta}$ using various data preprocessing methods.

Data processing	Model	Relative $\boldsymbol{\theta}$ error	Kendall's tau
All data	`SHazaM`	0.677	0.595
	`samm`	0.515	0.657
Imputation	`SHazaM`	0.721	0.580
	`samm`	0.537	0.639
Sampling	`SHazaM`	0.721	0.515
	`samm`	0.500	0.647

of intermediate nucleotide sequences are introduced on the way to obtaining the mutated sequences. These intermediate sequences, known as "ancestral states," are related to one another and to our observed sequences by an unknown phylogeny: a tree of dependencies that ties all sequences together by common ancestry. For a comprehensive treatment of phylogenetics, see Felsenstein [16].

Unfortunately we do not observe these ancestral states. A simple approach to estimate them is to use parsimony imputation [15], a method that minimizes the total number of mutations that occur on the tree. Reconstructing ancestral states using parsimony with `dnapars` [17] involves searching through a number of candidate trees and computing the minimum number of changes necessary to obtain each tree. Among the equally parsimonious trees returned by `dnapars`, we choose the first one to compute mutation contexts.

To determine the optimal data processing strategy between sampling, imputing ancestral states, and including all sequences without imputation, we simulate 3000 clonal families with realistic sizes. The composition for each of these clonal families is determined by sampling at random a cluster size and an inferred naïve sequence from the `partis`-processed Cui et al. [8] dataset. Cluster sizes range from 1–109. The median cluster size is two, and about 42% of all clusters are singletons. Sequences are 2.5% mutated on average. We take $\boldsymbol{\theta}$ to be a random resampling of $\boldsymbol{\theta}_{\texttt{MK}}$ parameters [8].

In Table 1.3 we see imputing ancestors using parsimony does not provide any improve-

ments in the model fit in most cases. Given that mutations in the simulation above occur based on the sequence context, the relatively poor performance of imputing ancestors may be due to the heterogeneity of mutation rates among sites [25]. Sampling a random descendant from each clonal family decreases the relative error for `samm`. For `SHazaM`, using all of the data results in the lowest relative error, most likely due to the fact that `SHazaM` fits mutabilities differently when not enough observations are present, and this case has more data than in the case of sampling. In Section 4, we sample from each clonal family to estimate the fit for `samm` while using all of the data for `SHazaM`.

Model misspecification: mutating with replacement

Throughout this manuscript, we have assumed that the positions in a BCR sequence mutate at most once. This assumption is for computational simplicity: if a position can mutate more than once, our estimation procedure must consider every single possible nucleotide sequence. However, this may not be realistic biologically. In this section, we present a simulation study to see how `samm`'s accuracy changes when positions are allowed to mutate multiple times.

Much of the simulation settings are similar to before. For the somatic hypermutation model, we resample from $\boldsymbol{\theta}_{\texttt{MK}}$ – defined in Section 2.5 – for each 3-mer motif, then randomly set half of them to zero. Each dataset consists of 300 simulated BCR sequences from a single mouse. Mutations are simulated using a survival model where each position can mutate multiple times versus at most one time. This simulation study is run twenty times.

For low mutation rates of 1–5%, we have similar accuracy when the model is misspecified (Table 1.4). The accuracies are similar since a position is very unlikely to mutate more than once in a low mutation rate setting.

We also try higher mutation rates as it is common to see mutation rates of 5–15% in humans, especially in individuals with chronic viral infections [22]. Even in this scenario with higher mutation rates, the accuracies are still similar. These results suggest that

Table 1.4: Results on twenty replicates of simulated data with standard errors (SE).

Mutation Rate (%)	True Model	Relative $\boldsymbol{\theta}$ error (SE)	Kendall's tau (SE)
1–5	Mutate at most once	0.364 (0.015)	0.722 (0.008)
	Mutate multiple times	0.348 (0.014)	0.723 (0.008)
5–15	Mutate at most once	0.194 (0.010)	0.791 (0.008)
	Mutate multiple times	0.190 (0.010)	0.781 (0.007)

our simplifying assumption gives up very little accuracy for a huge gain in computational efficiency.

Simulation results for the 27 settings

We report the results from the full set of possible simulation settings from Section 3.1 for the unpenalized (Tables 1.6 and 1.7) and penalized (Table 1.8) fits. Settings reported in the main manuscript were run 100 times; the others were run ten times. Across all 2700 simulation runs, a total of twenty replicates fail to obtain confidence intervals after eighty MCEM iterations, given in Table 1.5.

In most cases, the penalized fits and unpenalized fits obtain similar relative errors and rank correlations. In roughly half of cases, the penalized fits obtain smaller relative errors than the unpenalized fits; this may be an effect of the shrinkage present in the penalized $\boldsymbol{\theta}$. In all cases but four, the unpenalized fits have higher correlation. We prefer unpenalized fits as they are the only way to obtain reliable uncertainty estimates, though if reconstructing $\boldsymbol{\theta}$ is the primary goal then penalized fits provide a quicker solution.

For the unpenalized fits, the average number of false positives is less than one in the majority of settings, indicating our procedure has good support recovery. Our model has the most false positives with hierarchical fits on large numbers of samples. Moreover, we see expected trends in the output – relative error decreases and correlation increases as sample size increases, and per-target models are more difficult to fit than same-target ones. Varying effect size and sparsity levels does not seem to affect our method's performance

Table 1.5: Number of failed replicates, i.e. replicates where variance estimates are negative, out of total number of failed replicates for the simulations

% effect size	% nonzeros	# of samples	Model: failed reps/total reps
50	50	200	2,3-mer: 1/100
100	25	200	2,3-mer: 2/10
	50	100	2,3-mer: 1/100
			3-mer per-target: 1/100
		200	2,3-mer: 1/100
		400	2,3-mer: 7/100
	100	200	2,3-mer: 2/100
200	50	200	2,3-mer: 1/100
			3-mer per-target: 1/100
		400	3-mer per-target: 1/10
	100	100	3-mer per-target: 1/10
		200	3-mer: 1/10

significantly.

Table 1.6: Full simulation results using the unpenalized refit $\boldsymbol{\theta}$.

			Relative $\boldsymbol{\theta}$ error (SE)			Kendall's tau (SE)		
% effect size	% nonzeros	# samples	2,3-mer	3-mer	3-mer per-target	2,3-mer	3-mer	3-mer per-target
50	25	100	0.725 (0.156)	0.836 (0.205)	1.027 (0.197)	0.573 (0.061)	0.504 (0.061)	0.393 (0.040)
		200	0.474 (0.068)	0.558 (0.098)	0.846 (0.062)	0.709 (0.018)	0.575 (0.063)	0.477 (0.057)
		400	0.380 (0.091)	0.377 (0.110)	0.656 (0.052)	0.777 (0.030)	0.612 (0.036)	0.564 (0.035)
	50	100	0.682 (0.140)	0.697 (0.103)	1.017 (0.106)	0.625 (0.042)	0.606 (0.056)	0.433 (0.033)
		200	0.562 (0.196)	0.572 (0.153)	0.880 (0.089)	0.712 (0.044)	0.670 (0.046)	0.528 (0.035)
		400	0.393 (0.175)	0.435 (0.081)	0.679 (0.026)	0.805 (0.026)	0.720 (0.041)	0.647 (0.029)
	100	100	0.710 (0.208)	0.840 (0.194)	1.217 (0.236)	0.614 (0.065)	0.624 (0.023)	0.458 (0.041)
		200	0.547 (0.100)	0.603 (0.088)	0.880 (0.097)	0.708 (0.026)	0.708 (0.026)	0.552 (0.045)
		400	0.370 (0.047)	0.468 (0.057)	0.718 (0.080)	0.777 (0.028)	0.769 (0.017)	0.648 (0.021)
100	25	100	0.443 (0.143)	0.424 (0.119)	0.680 (0.067)	0.725 (0.059)	0.614 (0.047)	0.535 (0.051)
		200	0.331 (0.089)	0.316 (0.091)	0.565 (0.053)	0.777 (0.029)	0.650 (0.078)	0.625 (0.036)
		400	0.246 (0.046)	0.245 (0.092)	0.406 (0.044)	0.807 (0.021)	0.649 (0.025)	0.704 (0.025)
	50	100	0.519 (0.148)	0.539 (0.106)	0.820 (0.115)	0.752 (0.033)	0.695 (0.050)	0.567 (0.039)
		200	0.355 (0.090)	0.369 (0.073)	0.658 (0.051)	0.812 (0.028)	0.745 (0.039)	0.658 (0.035)
		400	0.248 (0.049)	0.275 (0.060)	0.496 (0.046)	0.860 (0.020)	0.779 (0.032)	0.743 (0.026)
	100	100	0.496 (0.063)	0.631 (0.101)	0.852 (0.062)	0.712 (0.033)	0.742 (0.027)	0.568 (0.034)
		200	0.373 (0.057)	0.429 (0.062)	0.649 (0.061)	0.783 (0.026)	0.805 (0.031)	0.692 (0.026)
		400	0.270 (0.031)	0.318 (0.064)	0.483 (0.035)	0.842 (0.017)	0.864 (0.020)	0.770 (0.017)
200	25	100	0.483 (0.124)	0.499 (0.069)	0.639 (0.050)	0.689 (0.025)	0.610 (0.030)	0.559 (0.049)
		200	0.357 (0.047)	0.389 (0.046)	0.534 (0.034)	0.737 (0.043)	0.637 (0.037)	0.619 (0.042)
		400	0.272 (0.033)	0.304 (0.024)	0.467 (0.035)	0.792 (0.024)	0.708 (0.055)	0.690 (0.022)
	50	100	0.420 (0.065)	0.494 (0.044)	0.633 (0.033)	0.755 (0.029)	0.648 (0.044)	0.594 (0.029)
		200	0.317 (0.042)	0.411 (0.046)	0.555 (0.036)	0.811 (0.025)	0.695 (0.034)	0.667 (0.028)
		400	0.257 (0.028)	0.316 (0.021)	0.501 (0.043)	0.857 (0.017)	0.760 (0.029)	0.714 (0.029)
	100	100	0.459 (0.039)	0.532 (0.056)	0.684 (0.081)	0.680 (0.038)	0.747 (0.025)	0.618 (0.024)
		200	0.378 (0.053)	0.388 (0.035)	0.533 (0.041)	0.741 (0.040)	0.808 (0.020)	0.716 (0.017)
		400	0.312 (0.029)	0.323 (0.033)	0.454 (0.030)	0.800 (0.033)	0.852 (0.016)	0.771 (0.013)

Table 1.7: Full simulation results reporting coverage and false discovery statistics using the unpenalized refit $\boldsymbol{\theta}$.

			Coverage (SE)			Num False Positive; Num Discovered (SE; SE)		
% effect size	% nonzeros	# samples	2,3-mer	3-mer	3-mer per-target	2,3-mer	3-mer	3-mer per-target
50	25	100	85.4 (10.5)	89.7 (10.2)	72.2 (11.9)	1.2; 6.9 (0.7; 1.5)	1.5; 8.9 (0.8; 2.0)	2.0; 13.0 (1.6; 4.5)
		200	93.8 (5.6)	87.9 (8.8)	67.4 (13.9)	0.9; 9.0 (0.8; 2.9)	2.5; 12.7 (1.7; 1.6)	4.9; 24.0 (1.7; 2.6)
		400	96.4 (1.7)	93.6 (4.8)	81.6 (10.6)	1.4; 13.2 (0.8; 2.0)	2.5; 15.0 (1.8; 1.7)	4.0; 32.7 (2.7; 4.3)
	50	100	85.8 (9.5)	93.7 (3.6)	71.5 (14.5)	0.9; 8.7 (0.8; 2.7)	1.3; 15.1 (1.0; 2.8)	0.3; 15.7 (0.5; 4.7)
		200	94.4 (6.0)	94.7 (3.5)	75.2 (12.8)	1.6; 11.1 (1.4; 4.0)	1.1; 18.3 (1.0; 2.1)	1.6; 26.1 (1.3; 4.8)
		400	98.9 (1.6)	93.6 (4.2)	87.3 (7.9)	2.2; 13.5 (2.6; 6.7)	1.5; 23.8 (1.3; 2.0)	1.6; 43.6 (1.2; 4.8)
	100	100	89.7 (9.7)	92.7 (4.0)	80.0 (9.8)	0.0; 7.1 (0.0; 1.9)	0.0; 18.0 (0.0; 1.7)	0.0; 23.3 (0.0; 5.3)
		200	96.1 (3.7)	95.7 (1.9)	81.1 (9.3)	0.0; 6.3 (0.0; 2.5)	0.0; 24.0 (0.0; 2.9)	0.0; 32.5 (0.0; 4.9)
		400	96.7 (3.1)	96.2 (3.2)	92.2 (6.2)	0.0; 10.5 (0.0; 4.3)	0.0; 30.3 (0.0; 2.7)	0.0; 55.3 (0.0; 6.1)
100	25	100	96.7 (2.3)	92.2 (9.3)	76.1 (11.2)	0.3; 8.5 (0.6; 3.1)	1.5; 10.6 (1.3; 2.0)	2.3; 21.2 (1.5; 4.4)
		200	95.3 (4.4)	92.5 (5.2)	84.2 (9.6)	1.1; 12.5 (1.4; 4.0)	1.8; 13.3 (1.3; 2.3)	2.5; 30.3 (1.9; 5.4)
		400	97.4 (1.7)	93.4 (4.6)	87.5 (9.8)	1.0; 14.6 (0.7; 2.6)	2.2; 15.5 (1.8; 2.1)	4.9; 46.7 (2.0; 4.2)
	50	100	96.7 (4.4)	96.0 (3.7)	79.0 (9.4)	1.3; 10.9 (1.3; 4.7)	0.8; 16.7 (0.9; 2.6)	1.1; 27.9 (1.3; 6.2)
		200	97.3 (3.3)	96.0 (4.0)	85.0 (9.8)	2.3; 16.0 (2.1; 6.2)	0.9; 21.5 (0.9; 2.5)	0.9; 40.0 (0.9; 5.8)
		400	97.7 (2.5)	95.3 (4.1)	89.8 (7.6)	4.5; 21.2 (2.5; 7.2)	1.1; 24.2 (1.0; 4.6)	1.9; 59.1 (1.3; 6.9)
	100	100	95.6 (6.7)	96.0 (4.1)	84.0 (8.7)	0.0; 6.7 (0.0; 5.1)	0.0; 22.3 (0.0; 6.4)	0.0; 39.4 (0.0; 5.3)
		200	95.7 (4.1)	96.2 (4.4)	91.7 (6.5)	0.0; 13.0 (0.0; 6.2)	0.0; 31.6 (0.0; 8.1)	0.0; 63.1 (0.0; 8.0)
		400	95.8 (2.7)	97.1 (3.3)	94.6 (3.4)	0.0; 24.8 (0.0; 3.8)	0.0; 34.2 (0.0; 15.5)	0.0; 86.8 (0.0; 14.3)
200	25	100	99.5 (1.0)	93.4 (2.2)	83.9 (8.0)	0.2; 4.7 (0.4; 2.4)	2.1; 10.4 (1.0; 1.7)	1.3; 23.7 (1.1; 2.3)
		200	98.4 (2.1)	88.5 (6.6)	83.5 (5.6)	0.8; 9.8 (1.0; 4.0)	2.3; 12.6 (1.6; 1.8)	1.5; 29.7 (1.0; 2.6)
		400	95.9 (6.0)	88.9 (7.0)	87.4 (6.5)	3.7; 16.3 (3.0; 6.1)	1.7; 14.4 (1.6; 1.9)	2.2; 36.0 (1.9; 5.2)
	50	100	99.2 (1.1)	96.5 (2.8)	92.3 (4.8)	0.5; 10.7 (0.9; 3.5)	0.7; 15.0 (0.8; 1.7)	0.3; 30.3 (0.6; 6.3)
		200	98.2 (2.5)	95.7 (5.9)	93.1 (5.2)	3.1; 17.5 (2.1; 6.1)	0.6; 14.9 (0.9; 3.9)	0.2; 39.6 (0.6; 6.7)
		400	95.0 (5.3)	95.6 (5.5)	89.9 (9.2)	6.7; 25.8 (2.9; 5.7)	0.2; 18.1 (0.4; 3.1)	0.6; 48.6 (0.7; 9.9)
	100	100	98.0 (2.0)	98.5 (3.4)	91.3 (4.1)	0.0; 8.2 (0.0; 4.2)	0.0; 16.0 (0.0; 7.8)	0.0; 46.2 (0.0; 2.9)
		200	96.4 (7.8)	93.1 (7.1)	95.4 (4.7)	0.0; 11.9 (0.0; 5.5)	0.0; 24.8 (0.0; 6.6)	0.0; 56.4 (0.0; 12.4)
		400	93.6 (7.5)	98.5 (2.2)	96.6 (1.9)	0.0; 25.1 (0.0; 6.4)	0.0; 28.9 (0.0; 12.3)	0.0; 80.3 (0.0; 12.1)

Table 1.8: Full simulation results using penalized $\boldsymbol{\theta}$.

			Relative $\boldsymbol{\theta}$ error (SE)			Kendall's tau (SE)		
% effect size	% nonzeros	# samples	2,3-mer	3-mer	3-mer per-target	2,3-mer	3-mer	3-mer per-target
50	25	100	0.539 (0.064)	0.671 (0.083)	0.809 (0.058)	0.555 (0.059)	0.503 (0.063)	0.396 (0.036)
		200	0.438 (0.052)	0.521 (0.069)	0.698 (0.060)	0.660 (0.049)	0.561 (0.060)	0.454 (0.059)
		400	0.332 (0.041)	0.391 (0.086)	0.572 (0.041)	0.754 (0.036)	0.586 (0.036)	0.543 (0.036)
	50	100	0.558 (0.056)	0.553 (0.037)	0.780 (0.041)	0.608 (0.046)	0.595 (0.062)	0.427 (0.031)
		200	0.464 (0.090)	0.508 (0.100)	0.701 (0.028)	0.702 (0.048)	0.660 (0.046)	0.523 (0.030)
		400	0.361 (0.079)	0.393 (0.057)	0.599 (0.030)	0.788 (0.025)	0.711 (0.042)	0.615 (0.034)
	100	100	0.555 (0.066)	0.692 (0.087)	0.824 (0.048)	0.617 (0.055)	0.608 (0.031)	0.460 (0.050)
		200	0.466 (0.031)	0.562 (0.034)	0.722 (0.044)	0.700 (0.022)	0.681 (0.039)	0.546 (0.031)
		400	0.368 (0.027)	0.451 (0.048)	0.622 (0.050)	0.770 (0.019)	0.744 (0.027)	0.631 (0.030)
100	25	100	0.399 (0.112)	0.425 (0.092)	0.706 (0.063)	0.690 (0.063)	0.608 (0.053)	0.502 (0.047)
		200	0.305 (0.067)	0.383 (0.090)	0.550 (0.066)	0.757 (0.038)	0.646 (0.076)	0.600 (0.038)
		400	0.211 (0.031)	0.296 (0.078)	0.439 (0.034)	0.800 (0.026)	0.643 (0.026)	0.678 (0.024)
	50	100	0.413 (0.102)	0.465 (0.055)	0.685 (0.038)	0.740 (0.040)	0.681 (0.045)	0.548 (0.040)
		200	0.315 (0.060)	0.360 (0.067)	0.592 (0.043)	0.805 (0.031)	0.735 (0.039)	0.633 (0.038)
		400	0.240 (0.036)	0.281 (0.058)	0.503 (0.044)	0.848 (0.023)	0.768 (0.032)	0.713 (0.034)
	100	100	0.430 (0.038)	0.540 (0.033)	0.696 (0.051)	0.712 (0.041)	0.724 (0.027)	0.579 (0.036)
		200	0.362 (0.052)	0.421 (0.053)	0.580 (0.035)	0.763 (0.035)	0.783 (0.032)	0.670 (0.028)
		400	0.272 (0.025)	0.332 (0.038)	0.490 (0.041)	0.826 (0.008)	0.839 (0.023)	0.742 (0.025)
200	25	100	0.448 (0.102)	0.518 (0.018)	0.674 (0.035)	0.637 (0.034)	0.593 (0.044)	0.495 (0.069)
		200	0.359 (0.036)	0.487 (0.040)	0.642 (0.017)	0.701 (0.052)	0.611 (0.050)	0.567 (0.057)
		400	0.290 (0.032)	0.453 (0.040)	0.567 (0.046)	0.758 (0.039)	0.707 (0.049)	0.644 (0.031)
	50	100	0.375 (0.025)	0.462 (0.034)	0.649 (0.034)	0.747 (0.023)	0.633 (0.041)	0.540 (0.040)
		200	0.315 (0.037)	0.422 (0.038)	0.590 (0.035)	0.796 (0.031)	0.678 (0.034)	0.624 (0.036)
		400	0.279 (0.032)	0.368 (0.027)	0.545 (0.043)	0.838 (0.016)	0.737 (0.035)	0.684 (0.031)
	100	100	0.466 (0.022)	0.518 (0.029)	0.647 (0.033)	0.652 (0.040)	0.717 (0.017)	0.597 (0.024)
		200	0.403 (0.049)	0.433 (0.049)	0.567 (0.024)	0.709 (0.042)	0.777 (0.024)	0.678 (0.020)
		400	0.348 (0.035)	0.364 (0.044)	0.503 (0.027)	0.777 (0.034)	0.826 (0.026)	0.735 (0.015)

A.4 Computing the survival process likelihood on a tree with ancestral sequences at internal nodes

Sequences evolve along a tree with a shared mutation history. Often, given a set of sequence data, many candidate trees optimize the maximum parsimony objective function [15], and thus phylogenetic algorithms can return multiple solutions. We have found it to be useful to rank a set of equally-parsimonious phylogenetic trees in terms of an additional objective function [9, 11]. To do such ranking with a motif mutability model requires taking into account mutation order – though the naïve and mutated sequences are the same across trees, the pairs of parent and descendant sequences on each branch are not. Though tempting, we cannot use the surrogate function (1.11) on different trees separately and compare them as the observed data differs between different trees. Instead we are interested in

$$\log \mathcal{L}(\mathbf{S}_{\text{obs}}; \boldsymbol{\theta}) - \log \mathcal{L}(\mathbf{S}'_{\text{obs}}; \boldsymbol{\theta}) \tag{1.22}$$

which requires estimating the observed likelihood.

To obtain the observed likelihood of data given a tree with inferred ancestral sequences at internal nodes, we use Chib's method to integrate out mutation order along each branch [5]. This gives us an estimate of the observed likelihood and allows us to compare multiple trees fit to the same data.

BIBLIOGRAPHY

[1] Varun Aggarwala and Benjamin F Voight. An expanded sequence context model broadly explains variability in polymorphism levels across the human genome. Nature Genetics, 48(4):349–355, April 2016. ISSN 1061-4036, 1546-1718. doi: 10.1038/ng.3511. URL http://dx.doi.org/10.1038/ng.3511.

[2] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal on Imaging Sciences, 2(1):183–202, 2009.

[3] Brian S. Caffo, Wolfgang Jank, and Galin L. Jones. Ascent-based monte carlo expectation–maximization. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2):235–251, 2005. ISSN 1467-9868. doi: 10.1111/j.1467-9868.2005.00499.x. URL http://dx.doi.org/10.1111/j.1467-9868.2005.00499.x.

[4] Richard Chahwan, Winfried Edelmann, Matthew D Scharff, and Sergio Roa. AIDing antibody diversity by error-prone mismatch repair. Seminars in Immunology, 24(4): 293–300, August 2012. ISSN 1044-5323, 1096-3618. doi: 10.1016/j.smim.2012.05.005. URL http://dx.doi.org/10.1016/j.smim.2012.05.005.

[5] Siddhartha Chib. Marginal likelihood from the gibbs output. Journal of the American Statistical Association, 90(432):1313–1321, 1995.

[6] Reuma Magori Cohen, Steven H Kleinstein, and Yoram Louzoun. Somatic hypermutation targeting is influenced by location within the immunoglobulin V region. Molecular Immunology, 48(12-13):1477–1483, July 2011. ISSN 0161-5890, 1872-9142. doi: 10.1016/j.molimm.2011.04.002. URL http://dx.doi.org/10.1016/j.molimm.2011.04.002.

[7] Lindsay G Cowell and Thomas B Kepler. The nucleotide-replacement spectrum under somatic hypermutation exhibits microsequence dependence that is strand-symmetric and distinct from that under germline mutation. The Journal of Immunology, 164(4): 1971–1976, February 2000. ISSN 0022-1767, 1550-6606. doi: 10.4049/jimmunol.164.4.1971. URL http://www.jimmunol.org/content/164/4/1971.short.

[8] Ang Cui, Roberto Di Niro, Jason A Vander Heiden, Adrian W Briggs, Kris Adams, Tamara Gilbert, Kevin C O'Connor, Francois Vigneault, Mark J Shlomchik, and Steven H Kleinstein. A model of somatic hypermutation targeting in mice based on high-throughput Ig sequencing data. The Journal of Immunology, 197(9):3566–3574, 1 November 2016. ISSN 0022-1767, 1550-6606. doi: 10.4049/jimmunol.1502263. URL http://dx.doi.org/10.4049/jimmunol.1502263.

[9] Kristian Davidsen and Frederick Matsen. Benchmarking tree and ancestral sequence inference for B cell receptor sequences. April 2018. URL https://www.biorxiv.org/content/early/2018/04/25/307736.

[10] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society: Series B (Statistical Methodology), pages 1–38, 1977.

[11] William S DeWitt, 3rd, Luka Mesin, Gabriel D Victora, Vladimir N Minin, and Frederick A Matsen, 4th. Using genotype abundance to improve phylogenetic inference. Molecular Biology and Evolution, 35(5):1253–1265, May 2018. ISSN 0737-4038, 1537-1719. doi: 10.1093/molbev/msy020. URL http://dx.doi.org/10.1093/molbev/msy020.

[12] Ruben Dezeure, Peter Bühlmann, Lukas Meier, Nicolai Meinshausen, et al. High-dimensional inference: confidence intervals, p-values and r-software hdi. Statistical science, 30(4):533–558, 2015.

[13] D K Dunn-Walters, A Dogan, L Boursier, C M MacDonald, and J Spencer. Base-specific sequences that bias somatic hypermutation deduced by analysis of out-of-frame human IgVH genes. The Journal of Immunology, 160(5):2360–2364, 1 March 1998. ISSN 0022-1767. URL https://www.ncbi.nlm.nih.gov/pubmed/9498777.

[14] Yuval Elhanati, Zachary Sethna, Quentin Marcou, Curtis G Callan, Jr, Thierry Mora, and Aleksandra M Walczak. Inferring processes underlying B-cell repertoire diversity. Philosophical Transactions of the Royal Society B: Biological Sciences, 370(1676), 5 September 2015. ISSN 0962-8436, 1471-2970. doi: 10.1098/rstb.2014.0243. URL http://dx.doi.org/10.1098/rstb.2014.0243.

[15] James S. Farris. Methods for computing wagner trees. Systematic Zoology, 19(1): 83–92, 1970.

[16] J. Felsenstein. Inferring phylogenies. Sinauer Associates, Sunderland, MA, 2003.

[17] J. Felsenstein. PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author, 2005. Department of Genome Sciences, University of Washington, Seattle.

[18] Jean Feng, David A. Shaw, Vladimir N. Minin, Noah Simon, and Frederick A. Matsen IV. Supplement to: "survival analysis of dna mutation motifs with penalized proportional hazards". Annals of Applied Statistics, 2019.

[19] William B Goggins, Dianne M Finkelstein, David A Schoenfeld, and Alan M Zaslavsky. A markov chain monte carlo em algorithm for analyzing interval-censored data under the cox proportional hazards model. Biometrics, pages 1498–1507, 1998.

[20] Namita T Gupta, Jason A Vander Heiden, Mohamed Uduman, Daniel Gadala-Maria, Gur Yaari, and Steven H Kleinstein. Change-O: a toolkit for analyzing large-scale B cell immunoglobulin repertoire sequencing data. Bioinformatics, 31(20):3356–3358, 2015.

[21] Barton F Haynes, Garnett Kelsoe, Stephen C Harrison, and Thomas B Kepler. B-cell-lineage immunogen design in vaccine development with HIV-1 as a case study. Nature Biotechnology, 30(5):423–433, May 2012. ISSN 1087-0156, 1546-1696. doi: 10.1038/nbt.2197. URL http://dx.doi.org/10.1038/nbt.2197.

[22] Linling He, Devin Sok, Parisa Azadnia, Jessica Hsueh, Elise Landais, Melissa Simek, Wayne C Koff, Pascal Poignard, Dennis R Burton, and Jiang Zhu. Toward a more accurate view of human B-cell repertoire by next-generation sequencing, unbiased repertoire capture and single-molecule barcoding. Scientific Reports, 4:6778, 27 October 2014. ISSN 2045-2322. doi: 10.1038/srep06778. URL http://dx.doi.org/10.1038/srep06778.

[23] Uri Hershberg, Mohamed Uduman, Mark J Shlomchik, and Steven H Kleinstein. Improved methods for detecting selection by mutation analysis of Ig V region sequences. International Immunology, 20(5):683–694, 7 April 2008. ISSN 0953-8178. doi: 10.1093/intimm/dxn026. URL http://dx.doi.org/10.1093/intimm/dxn026.

[24] Tim Hesterberg, Nam Hee Choi, Lukas Meier, Chris Fraley, et al. Least angle and ℓ_1 penalized regression: A review. Statistics Surveys, 2:61–93, 2008.

[25] S.Y.W. Ho and L.S. Jermiin. Tracing the decay of the historical signal in biological sequence data. Systematic Biology, 53:628–637, 2004.

[26] Asger Hobolth. A Markov chain Monte Carlo expectation maximization algorithm for statistical analysis of DNA sequence evolution with neighbor-dependent substitution rates. Journal of Computational and Graphical Statistics, 17(1):138–162, 2008. ISSN 1061-8600. doi: 10.1198/106186008X289010. URL http://dx.doi.org/10.1198/106186008X289010.

[27] Kenneth B Hoehn, Gerton Lunter, and Oliver G Pybus. A phylogenetic codon substitution model for antibody lineages. Genetics, 206(1):417–427, May 2017. ISSN

0016-6731, 1943-2631. doi: 10.1534/genetics.116.196303. URL http://dx.doi.org/10.1534/genetics.116.196303.

[28] Dick G Hwang and Phil Green. Bayesian markov chain monte carlo sequence analysis reveals varying neutral substitution patterns in mammalian evolution. Proceedings of the National Academy of Sciences USA, 101(39):13994–14001, 28 September 2004. ISSN 0027-8424. doi: 10.1073/pnas.0404142101. URL http://dx.doi.org/10.1073/pnas.0404142101.

[29] Joyce K Hwang, Chong Wang, Zhou Du, Robin M Meyers, Thomas B Kepler, Donna Neuberg, Peter D Kwong, John R Mascola, M Gordon Joyce, Mattia Bonsignori, Barton F Haynes, Leng-Siew Yeap, and Frederick W Alt. Sequence intrinsic somatic mutation mechanisms contribute to affinity maturation of VRC01-class HIV-1 broadly neutralizing antibodies. Proceedings of the National Academy of Sciences USA, July 2017. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1709203114. URL http://dx.doi.org/10.1073/pnas.1709203114.

[30] John D Kalbfleisch and Ross L Prentice. The statistical analysis of failure time data, volume 360. John Wiley & Sons, 2011.

[31] Hannes Leeb, Benedikt M Pötscher, Karl Ewald, et al. On various confidence intervals post-model-selection. Statistical Science, 30(2):216–227, 2015.

[32] Marie-Paule Lefranc. Immunoglobulins: 25 years of immunoinformatics and IMGT-ONTOLOGY. Biomolecules, 4(4):1102–1139, December 2014. ISSN 2218-273X. doi: 10.3390/biom4041102. URL http://dx.doi.org/10.3390/biom4041102.

[33] Marie-Paule Lefranc, Véronique Giudicelli, Chantal Ginestoux, Julia Bodmer, Werner Müller, Ronald Bontrop, Marc Lemaitre, Ansar Malik, Valérie Barbié, and Denys Chaume. Imgt, the international immunogenetics database. Nucleic Acids Research, 27(1):209–212, 1999.

[34] Thomas A Louis. Finding the observed information matrix when using the em algorithm. Journal of the Royal Statistical Society: Series B (Statistical Methodology), pages 226–233, 1982.

[35] Connor O McCoy, Trevor Bedford, Vladimir N Minin, Philip Bradley, Harlan Robins, and Frederick A Matsen, IV. Quantifying evolutionary constraints on B-cell affinity maturation. Philosophical Transactions of the Royal Society B: Biological Sciences, 370 (1676), 5 September 2015. ISSN 0962-8436, 1471-2970. doi: 10.1098/rstb.2014.0244. URL http://dx.doi.org/10.1098/rstb.2014.0244.

[36] S P Methot and J M Di Noia. Chapter two - molecular mechanisms of somatic hypermutation and class switch recombination. In Frederick W. Alt, editor, Advances in Immunology, volume 133, pages 37–87. Academic Press, 2017. URL http://www.sciencedirect.com/science/article/pii/S0065277616300530.

[37] Y. Nesterov. Gradient methods for minimizing composite objective functions. Mathematical Programming, 140(1):125–161, 2013.

[38] Phuong Pham, Ronda Bransteitter, John Petruska, and Myron F Goodman. Processive AID-catalysed cytosine deamination on single-stranded DNA simulates somatic hypermutation. Nature, 424(6944):103–107, 3 July 2003. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature01760. URL http://dx.doi.org/10.1038/nature01760.

[39] Duncan K. Ralph and Frederick A. Matsen IV. Consistency of vdj rearrangement and substitution parameters enables accurate b cell receptor sequence annotation. PLOS Computational Biology, 12(1):1–25, 01 2016. doi: 10.1371/journal.pcbi.1004409. URL https://doi.org/10.1371/journal.pcbi.1004409.

[40] Duncan K Ralph and Frederick A Matsen IV. Likelihood-based inference of b cell clonal families. PLOS Computational Biology, 12(10):e1005086, 2016.

[41] I B Rogozin and M Diaz. Cutting edge: DGYW/WRCH is a better predictor of mutability at G: C bases in Ig hypermutation than the widely accepted RGYW/WRCY motif and probably reflects a two-step Activation-Induced Cytidine Deaminase-triggered process. The Journal of Immunology, 2004. URL http://www.jimmunol.org/content/172/6/3382.short.

[42] I B Rogozin and N A Kolchanov. Somatic hypermutagenesis in immunoglobulin genes. II. Influence of neighbouring base sequences on mutagenesis. Biochimica et Biophysica Acta, 1171(1):11–18, 15 November 1992. ISSN 0006-3002. URL http://www.ncbi.nlm.nih.gov/pubmed/1420357.

[43] Igor B Rogozin, Youri I Pavlov, Katarzyna Bebenek, Toshiro Matsuda, and Thomas A Kunkel. Somatic mutation hotspots correlate with DNA polymerase η error spectrum. Nature Immunology, 2(6):530–536, 1 June 2001. ISSN 1529-2908. doi: 10.1038/88732. URL http://dx.doi.org/10.1038/88732.

[44] David G Schatz and Yanhong Ji. Recombination centres and the orchestration of V (d) J recombination. Nature Reviews Immunology, 11(4):251–263, 2011. ISSN 1474-1733. URL https://www.nature.com/nri/journal/v11/n4/abs/nri2941.html.

[45] Zizhang Sheng, Chaim A Schramm, Rui Kong, NISC Comparative Sequencing Program, James C Mullikin, John R Mascola, Peter D Kwong, and Lawrence Shapiro. Gene-specific substitution profiles describe the types and frequencies of amino acid changes during antibody somatic hypermutation. Frontiers in Immunology, 8:537, May 2017. ISSN 1664-3224. doi: 10.3389/fimmu.2017.00537. URL http://dx.doi.org/10.3389/fimmu.2017.00537.

[46] Robert Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Statistical Methodology), pages 267–288, 1996.

[47] Robert Tibshirani et al. The lasso method for variable selection in the cox model. Statistics in Medicine, 16(4):385–395, 1997.

[48] S Tonegawa. Somatic generation of antibody diversity. Nature, 302(5909):575–581, 14 April 1983. ISSN 0028-0836. doi: 10.1038/302575a0. URL http://www.ncbi.nlm.nih.gov/pubmed/6300689.

[49] Mohamed Uduman, Gur Yaari, Uri Hershberg, Jacob A Stern, Mark J Shlomchik, and Steven H Kleinstein. Detecting selection in immunoglobulin sequences. Nucleic Acids Research, 39(Web Server issue):W499–504, 10 June 2011. ISSN 0305-1048. doi: 10.1093/nar/gkr413. URL http://dx.doi.org/10.1093/nar/gkr413.

[50] Greg CG Wei and Martin A Tanner. A monte carlo implementation of the em algorithm and the poor man's data augmentation algorithms. Journal of the American Statistical Association, 85(411):699–704, 1990.

[51] Kevin Wiehe, Todd Bradley, R Ryan Meyerhoff, Connor Hart, Wilton B Williams, David Easterhoff, William J Faison, Thomas B Kepler, Kevin O Saunders, S Munir Alam, Mattia Bonsignori, and Barton F Haynes. Functional relevance of improbable antibody mutations for HIV broadly neutralizing antibody development. Cell Host Microbe, 0(0), May 2018. ISSN 1931-3128. doi: 10.1016/j.chom.2018.04.018. URL http://www.cell.com/article/S1931312818302191/abstract.

[52] Gur Yaari and Steven H Kleinstein. Practical guidelines for B-cell receptor repertoire sequencing analysis. Genome Medicine, 7(1):121, 2015. ISSN 1756-994X. doi: 10.1186/s13073-015-0243-2. URL http://dx.doi.org/10.1186/s13073-015-0243-2.

[53] Gur Yaari, Mohamed Uduman, and Steven H Kleinstein. Quantifying selection in high-throughput Immunoglobulin sequencing data sets. Nucleic Acids Research, 40(17):e134, 27 May 2012. ISSN 0305-1048. doi: 10.1093/nar/gks457. URL http://dx.doi.org/10.1093/nar/gks457.

[54] Gur Yaari, Jason A Vander Heiden, Mohamed Uduman, Daniel Gadala-Maria, Namita Gupta, Joel N H Stern, Kevin C O'Connor, David A Hafler, Uri Laserson, Francois Vigneault, and Steven H Kleinstein. Models of somatic hypermutation targeting and substitution based on synonymous mutations from high-throughput immunoglobulin sequencing data. Frontiers in Immunology, 4:358, 15 November 2013. ISSN 1664-3224. doi: 10.3389/fimmu.2013.00358. URL http://dx.doi.org/10.3389/fimmu.2013.00358.

[55] Gur Yaari, Jennifer I C Benichou, Jason A Vander Heiden, Steven H Kleinstein, and Yoram Louzoun. The mutation patterns in B-cell immunoglobulin receptors reflect the influence of selection acting at multiple time-scales. Philosophical Transactions of the Royal Society B: Biological Sciences, 370(1676), 5 September 2015. ISSN 0962-8436, 1471-2970. doi: 10.1098/rstb.2014.0242. URL http://dx.doi.org/10.1098/rstb.2014.0242.

[56] Leng-Siew Yeap, Joyce K Hwang, Zhou Du, Robin M Meyers, Fei-Long Meng, Agnė Jakubauskaitė, Mengyuan Liu, Vinidhra Mani, Donna Neuberg, Thomas B Kepler, Jing H Wang, and Frederick W Alt. Sequence-intrinsic mechanisms that target AID mutational outcomes on antibody genes. Cell, 2015. ISSN 0092-8674. doi: 10.1016/j.cell.2015.10.042. URL http://www.sciencedirect.com/science/article/pii/S0092867415013975.

[57] Sem Zhao, Ali Shojaie, and Daniela Witten. In defense of the indefensible: a very naive approach to high-dimensional inference. arXiv preprint arXiv:1705.05543, 2017.

[58] Qing Zhou and Jun S Liu. Modeling within-motif dependence for transcription factor binding site predictions. Bioinformatics, 20(6):909–916, 12 April 2004. ISSN 1367-4803. doi: 10.1093/bioinformatics/bth006. URL http://dx.doi.org/10.1093/bioinformatics/bth006.

Chapter 2

ESTIMATION OF CELL LINEAGE TREES BY MAXIMUM-LIKELIHOOD PHYLOGENETICS

Summary

CRISPR technology has enabled cell lineage tracing for complex multicellular organisms through insertion-deletion mutations of synthetic genomic barcodes during organismal development. To reconstruct the cell lineage tree from the mutated barcodes, current approaches apply general-purpose computational tools that are agnostic to the mutation process and are unable to take full advantage of the data's structure. We propose a statistical model for the CRISPR mutation process and develop a procedure to estimate the resulting tree topology, branch lengths, and mutation parameters by iteratively applying penalized maximum likelihood estimation. By treating the barcode as a molecular clock, our method infers relative ordering across parallel lineages, whereas existing techniques only infer ordering for nodes along the same lineage. When analyzing transgenic zebrafish data from **?**], we find that our method recapitulates known aspects of zebrafish development and the results are consistent across samples.

1 Introduction

Recent advancements in genome editing with CRISPR[1] has enabled the construction of large-scale cell lineage trees for complex organisms [**? ? ? ? ?**]. One of the pioneering methods — and the focus of this chapter — is Genome Editing of Synthetic Target Arrays for Lineage Tracing (GESTALT) [**?**]. GESTALT integrates an array of CRISPR/Cas9 targets, referred to as a barcode, into the genome of an embryo. Cas9 enzymes injected into the embryo are directed by single guide RNAs (sgRNAs) to bind and cleave the bar-

[1] Clustered Regularly Interspaced Short Palindromic Repeats

code. A mutation is introduced when nucleotides are deleted and/or inserted during DNA repair. As the organism develops, the barcode accumulates these random mutations and the mutated barcode is passed from parent cell to daughter cell, which thereby encodes the ontogeny. These mutated barcodes are sequenced from the organism at some timepoint, and computational phylogenetic methods are used to estimate the cell lineage tree. Due to the high diversity of the mutated barcodes, GESTALT has the potential to reveal organism development in high resolution. Other CRISPR-based lineage-tracing methods are similar but can vary in which genomic regions they target and how Cas9 is expressed. See **?**] for a comprehensive review of current CRISPR-based lineage-tracing technologies.

Current computational phylogenetic tools to analyze GESTALT data are insufficient. The most common methods are Camin-Sokal (C-S) parsimony [**?**] and the neighbor-joining distance-based method [**?**]. Tree estimates from these methods have limited interpretability since the branch lengths are in terms of an abstract notion of distance rather than time. Thus, they can only order nodes on the same lineage but not on parallel lineages. In addition, these general-purpose methods are blind to the mutation mechanism in GESTALT, so their accuracy can be poor [**?**]. Finally, because parsimony is a coarse scoring metric, C-S parsimony often generates many parsimony-optimal trees — over ten thousand in some existing datasets — requiring the user to choose one of them.

We set out to develop a statistical model and estimation method to address these challenges. No appropriate probabilistic model is currently available for GESTALT because the mutation process violates the classical statistical phylogenetic assumptions. For example, long tracts of DNA can be deleted from the barcode during GESTALT. So, the usual assumptions that mutations occur pointwise and that individual positions are independent are not satisfied [**?** **?**]. Moreover, the GESTALT mutation process is irreversible unlike most models in phylogenetics.

We introduce a statistical model for GESTALT and an iterative penalized maximum likelihood procedure to estimate the tree topology, branch lengths, and mutation parameters.

Our method, called GAPML (GESTALT analysis using penalized Maximum Likelihood), models the mutation process as a two-step procedure: Targets are cut according to a continuous time Markov chain, immediately followed by random insertions or deletions of nucleotides (indels). We have carefully tailored a new set of assumptions and approximations for GESTALT that makes the likelihood tractable yet maintains biological realism. We show that the Markov process can be modeled using a higher-level Markov process with many fewer "lumped" states [**?**]. We then combine lumpability with Felsenstein's pruning algorithm to efficiently compute the likelihood [**?**]. Throughout, we treat the GESTALT barcode as a molecular clock and obtain time estimates with respect to this clock.

We have designed GAPML for datasets generated by a small number of barcodes because inserting many barcodes is currently a technical challenge. In fact, existing GESTALT datasets were generated using only a single barcode. Maximum-likelihood phylogenetic methods are known to be unstable when the number of parameters is large but the number of independent observations (barcodes) is small [**?** **?** **?**]. Based on the success of penalization techniques in the high-dimensional statistics literature [**?**], we augment the objective with a penalty on the branch lengths and mutation parameters, and design an iterative tree search procedure compatible with this penalty. We note that penalties on the distance between the tree estimate and a pre-specified tree [**?** **?**] are not applicable here because we have little to no knowledge about the true tree.

Finally, our method estimates trees at a finer resolution compared to other methods. Whereas C-S parsimony estimates trees with many multifurcations (nodes with 3+ children), GAPML resolves multifurcations as caterpillar trees to infer additional ordering information. We efficiently tune the caterpillar tree orderings by solving a single continuous optimization problem, rather than a combinatorial one. This is noteworthy since there are very few situations in phylogenetics in which a topology search can be formulated as a continuous optimization problem.

The chapter is organized as follows. Sections 2 and 3 present the probabilistic model

and estimation method, respectively. We validate our method on simulated data in Section 4 and empirical data in Section 5. Compared to existing tree-estimation methods, our method is more accurate in simulations and better recapitulates the known biology of zebrafish development given data from **?**]. Our simulation engine and estimation method are available at https://github.com/matsengrp/gapml.

2 GESTALT model

Our goal is to reconstruct the cell lineage tree using data from **?**], which is generated using a barcode with 10 contiguous CRISPR/Cas9 target sites. Nodes in the tree represent cell divisions and branch lengths represent time between cell divisions. The full tree describes the relationships of all cells in the organism. Our goal is to recover the subtree for the observed sequences.

The experimental protocol in **?**] is as follows. At the single-cell zygote stage, a single barcode is integrated into the genome and Cas9 enzyme and sgRNAs are injected (Fig 2.1). Each target in the barcode is 23 nucleotides long, including the required protospacer adjacent motif, and are separated by a 4 base spacer. Individual sgRNAs matching the nucleotide sequence of a single unmodified target guide Cas9 enzymes to make double-stranded breaks at a specific cut site within each target. Mutations are introduced when a break is repaired in an error-prone fashion, and nucleotides are inserted or deleted around the cut site. Sometimes, two targets are cut, the intervening sequence is removed, and nucleotides are inserted/deleted during repair. Once a target is modified, the sgRNA no longer matches and the target can no longer be cut.

Since barcodes are inherited from mother to daughter cells, mutations accumulate along the barcodes in a lineage-specific fashion. These mutated barcodes, which we refer to as alleles, are recovered by DNA sequencing at the timepoint of interest. The number of unique sampled alleles are typically on the order of hundreds or thousands. Future experiments will likely include multiple barcodes to increase the number of unique alleles.

We model the GESTALT barcode as a continuous time Markov chain (CTMC). Calcu-

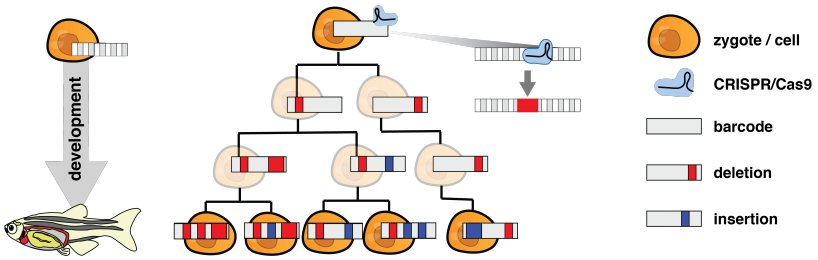


Figure 2.1: Overview of the GESTALT experimental setup. A barcode composed of CRISPR/Cas9 target sites is embedded into the genome of a zygote. During development, the barcode is inherited from mother to daughter cells. Mutations accumulate along the barcode when the Cas9 enzyme cuts target(s) and an error-prone repair process deletes and/or inserts nucleotides.

lating the likelihood of the tree for a general CTMC is computationally intractable for two reasons: First, the mutation rate can depend on the entire barcode sequence and second, because long deletion tracts mask previous mutation events, we must marginalize over an infinite number of possible ancestral states. To simplify the calculations, we propose the following assumptions, which are formalized mathematically later:

1 An indel is introduced by cuts at the outermost cut sites.

2A The cut rates only depend on which targets are unmodified.

2B The conditional probability that an indel is introduced only depends on which targets were cut.

2C The mutation process is irreversible.

In addition, we introduce approximations of the likelihood that significantly speeds up computation. Fig 2.15 in the Appendix summarizes how the main results are derived from the assumptions and approximations.

2.1 Definitions and notation

We begin with presenting mathematical abstractions for GESTALT. Table 2.4 in the Appendix is provided as a reference for the main definitions used in this chapter.

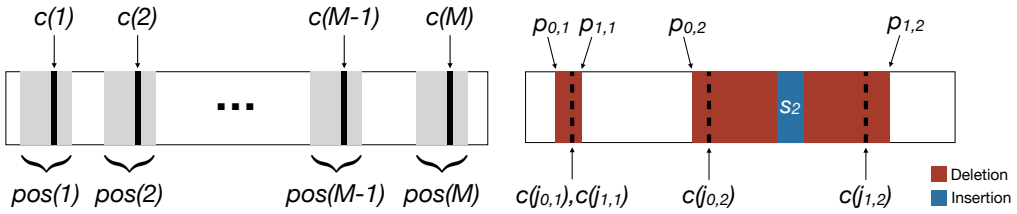


Figure 2.2: Left: A barcode with M targets. The cut site of the targets $c(\cdot)$ are indicated by bold lines. The positions associated with each target are highlighted using gray boxes. Right: Example allele with two indel tracts $\mathrm{IT}[p_{0,i}, p_{1,i}, s_i, j_{0,i}, j_{1,i}]$ for $i = 1, 2$. The first one was introduced by a cut at a single target and inserted nothing. The second one was introduced by cuts at two targets and the insertion of s_2.

Barcode

The unmodified barcode is a nucleotide sequence composed of M disjoint subsequences called targets (Fig 2.2 left). The targets are numbered from 1 to M from left to right, and the positions spanned by target j are specified by the set $\mathrm{pos}(j)$. Each target j is associated with a single cut site $c(j) \in \mathrm{pos}(j)$. For convenience, define $\mathrm{pos}(0) = \{0\}$ and $\mathrm{pos}(M+1) = \{l+1\}\}$ where l is the length of the barcode.

A barcode can be modified by the introduction of an indel tract. An indel tract, denoted by $\mathrm{IT}[p_0, p_1, s, j_0, j_1]$, is a mutation event in which targets j_0 and j_1 are cut ($j_0 \leq j_1$), positions $p_0, p_0+1, \ldots, p_1-1$ in the unmodified barcode are deleted, and a nucleotide sequence s is inserted. If $j_0 = j_1$, only a single target is cut. When $p_0 = p_1$, no positions are deleted. A valid indel tract must modify the sequence ($p_0 < p_1$ or s has positive length) and have cut sites for its targets nested within positions p_0 and p_1.

An allele is a sequence of $m \geq 0$ disjoint indel tracts (Fig 2.2 right):

$$a \equiv \{\mathrm{IT}[p_{0,k}, p_{1,k}, s_k, j_{0,k}, j_{1,k}] : k \in \{1, \ldots, m\}\} \tag{2.1}$$

where $p_{1,k} < p_{0,k+1}$ and $j_{1,k} < j_{0,k+1}$ for $k = 1, \ldots, m-1$. Note that the indices are always defined with respect to the original unmodified barcode. Let Ω be the set of all possible alleles.

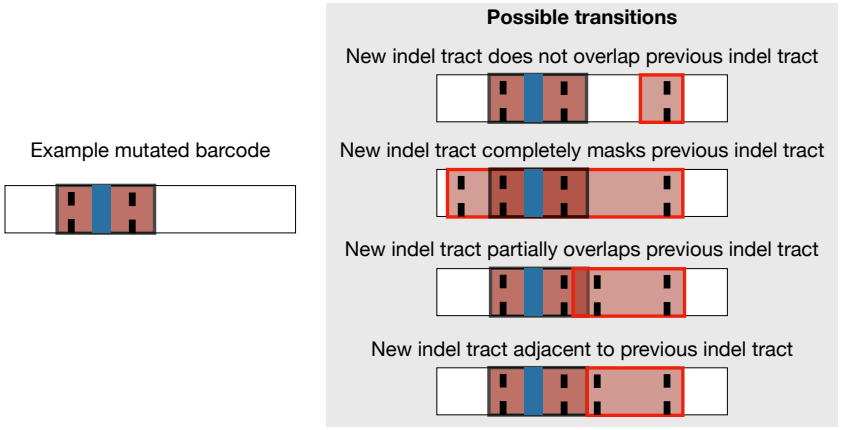


Figure 2.3: Possible transitions from the left allele are shown on the right. From top to bottom, the mutation process can introduce a new indel tract that does not overlap, completely masks, partially overlaps, or is adjacent to the previous indel tract.

Target j is active in allele a if no nucleotides in $\text{pos}(j)$ are modified. We denote the target's status as $\text{TargStat}(j; a)$, where zero means the target is active and one otherwise:

$$\text{TargStat}(j; a) = \mathbb{1}\{\exists\, \text{IT}[p_0, p_1, s, j_0, j_1] \in a \text{ and } \exists p' \in \text{pos}(j) \text{ s.t. } p_0 \leq p' \leq p_1\}.$$

For convenience, denote the target status of allele a as

$$\text{TargStat}(a) = (\text{TargStat}(1; a), \ldots, \text{TargStat}(M; a)). \tag{2.2}$$

The mutation process can introduce indel tract $d = \text{IT}[p_0, p_1, s, j_0, j_1]$ into an allele if and only if (i) targets j_0 and j_1 are active and (ii) p_0 and p_1 have not been deleted. Let $\text{Apply}(a, d)$ be the resulting allele from introducing indel tract d into allele a. A new indel tract either does not overlap existing indel tracts, completely masks other indel tracts, or merges with other indel tracts by partially overlapping or being adjacent to them (Fig 2.3).

Mutation process

The mutation process up to time T is formulated as a continuous time Markov chain $\{X(t) : 0 \leq t \leq T\}$ with state space Ω. Since Ω is defined as the set of possible alleles, we have implicitly assumed that indel tracts are introduced instantaneously, i.e. nucleotides are inserted and/or deleted immediately after target(s) are cut.

For tree $\mathbb{T}$, denote the leaves for node $\mathtt{N}$ as Leaves($\mathtt{N}$); use Leaves($\mathbb{T}$) to denote the set of all leaves. Let $a_{\mathtt{L}}$ be the allele observed at leaf node $\mathtt{L}$. For the branch ending with node $\mathtt{N}$, denote its length as $t_{\mathtt{N}}$ and the Markov process along it as $\{X_{\mathtt{N}}(t) : 0 \leq t \leq t_{\mathtt{N}}\}$. For simplicity, we present the model in the context of a single barcode. If there are multiple barcodes, we assume in this chapter that they are sufficiently far apart that they act in an independent and identically distributed (iid) manner.

2.2 Assumptions

We now formalize the assumptions presented before. Assumption 1 states that for any indel tract that cuts targets j_0 and j_1, its deletions cannot extend past the cut site of neighboring targets $j_0 - 1$ and $j_1 + 1$. Note that it can still deactivate neighboring targets by mutating nucleotides at the edge of these targets. We use this assumption to limit the set of possible mutation histories.

Assumption 1. *Each indel tract* $\mathrm{IT}[p_0, p_1, s, j_0, j_1]$ *satisfies* $c(j_0 - 1) < p_0 \leq c(j_0)$ *and* $c(j_1) \leq p_1 < c(j_1 + 1)$.

To formalize Assumptions 2A-C, define a target tract as a set of indel tracts that cut and deactivate the same target(s). A target tract, denoted $\mathrm{TT}[j_0', j_0, j_1, j_1']$ $(j_0' \leq j_0 \leq j_1 \leq j_1')$, is the set of all indel tracts that cut targets j_0 and j_1 and delete nucleotides such that targets j_0' through j_1' are inactive, i.e.

$$\mathrm{TT}[j_0', j_0, j_1, j_1'] = \left\{\mathrm{IT}[p_0, p_1, s, j_0, j_1] : p_0 \in \mathrm{pos}(j_0'), p_1 \in \mathrm{pos}(j_1')\right\}. \tag{2.3}$$

For instance, $\mathrm{TT}[2,2,3,4]$ is the set of indel tracts that cut targets 2 and 3, introduce deletions rightward that deactivate target 4 but not beyond, and introduce short deletions leftward so that target 1 is unaffected. Every indel tract d belongs to a single target tract, which we denote $\mathrm{TT}(d)$.

The second assumption states that the instantaneous rate of introducing indel tract d into allele a is the product of the rate of introducing any element from $\mathrm{TT}(d)$, which only depends on the target status of a, and the conditional probability of introducing d given $\mathrm{TT}(d)$. It also states that the mutation process is irreversible and homogeneous. As such, we treat the GESTALT barcode as a molecular clock. Note that the total mutation rate of a barcode varies over time based on which targets are active, but the model for the transition rates is stationary.

Assumption 2. *Let a be an allele, d be an indel tract that can be introduced into a, and $\tau = \mathrm{TT}(d)$. The instantaneous rate of introducing d in a at time t can be factored into two terms: first, a function that only depends on the triple $(\tau, \mathrm{TargStat}(a), t)$, and second, the conditional probability of introducing d given τ:*

$$\begin{aligned} q(a, \mathrm{Apply}(a,d)) &:= \lim_{\Delta \to 0} \frac{\Pr\left(X(\Delta) = \mathrm{Apply}(a,d) \mid X(0) = a\right)}{\Delta} \\ &= h\left(\tau, \mathrm{TargStat}(a)\right) \Pr\left(d \mid \tau\right). \end{aligned}$$

Moreover, $h(\tau, \mathrm{TargStat}(a)) = 0$ if τ cuts a target that is inactive in a.

Using Assumptions 1 and 2, we can calculate the (approximate) likelihood efficiently as described below. Assume the topology is fixed for now, which we denote as $\mathbb{T}$.

2.3 Summing over likely ancestral states

The first step to calculating the likelihood is to characterize the possible ancestral states. In this section, we provide a recursive algorithm for characterizing a *subset* of the ancestral states, which should capture all the likely ancestral states and only exclude those with very small probability.

Our approximation of the likelihood excludes mutation histories where overlapping indel tracts merged but did not fully mask one another:

Approximation 1. *The probability of indel tracts merging is approximately zero, i.e.*

$$\begin{aligned} &\Pr\left(X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in \text{Leaves}(\mathbb{T})\right) \\ \approx &\Pr\left(X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in \text{Leaves}(\mathbb{T}), \textit{no indel tracts merged}\right). \end{aligned} \tag{2.4}$$

We will refer to the right-hand probability as the approximate likelihood. We believe merge events are rare since they occur when deletion lengths are long, whereas most deletions are short in **?**]. By excluding merge events, we show that the set of ancestral states in Approximation 1 can be expressed compactly.

Now, let us define a partial ordering among alleles using Approximation 1 and Assumption 1. Given two alleles $a, a' \in \Omega$, $a \preceq a'$ means that a can transition to a' *without merging indel tracts*, i.e. there is a sequence of indel tracts $\{d_i\}_{i=1}^m$ for some $m \geq 0$ such that

$$a' = \text{Apply}\left(d_m, \text{Apply}\left(d_{m-1}, \ldots \text{Apply}(d_1, a)\right)\right)$$

where no indel tracts merge. Then, the set of "likely" ancestral states at internal node $\mathtt{N}$ in tree $\mathbb{T}$ is defined as

$$\text{AncState}\left(\mathtt{N}\right) = \left\{a \in \Omega : a \preceq a_{\mathtt{L}}\ \forall\, \mathtt{L} \in \text{Leaves}(\mathtt{N})\right\}. \tag{2.5}$$

(Note that AncState$(\cdot)$ is also defined for leaf nodes, in which case it is the set of alleles that likely preceded the observed allele.) To calculate the approximate likelihood in (2.4), we marginalize over AncState$(\mathtt{N})$ at each internal node $\mathtt{N}$.

We can characterize AncState$(\mathtt{N})$ using only two building blocks (Fig 2.4): wildcards and singleton-wildcards. A wildcard[2] WC$[j_0, j_1]$ is the set of all indel tracts that only deactivate

[2]In software systems, a wildcard is a symbol used to represent one or more characters (e.g. "*"). Similarly,

targets within the range j_0 to j_1, inclusive:

$$\mathrm{WC}[j_0, j_1] = \left\{\mathrm{IT}[p_0', p_1', s', j_0', j_1'] : \mathrm{pos}(j_0 - 1) < p_0', p_1' < \mathrm{pos}(j_1 + 1)\right\}. \tag{2.6}$$

A singleton-wildcard SGWC$[p_0, p_1, s, j_0, j_1]$ is the union of the singleton set $\{\mathrm{IT}[p_0, p_1, s, j_0, j_1]\}$ and its inner wildcard WC$[j_0 + 1, j_1 - 1]$, if it exists:

$$\mathrm{SGWC}[p_0, p_1, s, j_0, j_1] = \begin{cases} \{\mathrm{IT}[p_0, p_1, s, j_0, j_1]\} \cup \mathrm{WC}[j_0 + 1, j_1 - 1] & \text{if } j_0 + 1 \leq j_1 - 1 \\ \{\mathrm{IT}[p_0, p_1, s, j_0, j_1]\} & \text{otherwise.} \end{cases} \tag{2.7}$$

Two or more wildcards (WCs) and/or singleton-wildcards (SGWCs) are disjoint if the maximum ranges of targets deactivated by indel tracts in these sets do not overlap.

Given a set of indel tracts D, let the alleles generated by D, denoted Alleles(D), be the set of alleles that can be created using subsets of D:

$$\{\{\mathrm{IT}[p_{0,k}, p_{1,k}, s_j, j_{0,k}, j_{1,k}]\}_{k=1}^{m} \subseteq D : \\ m \in \mathbb{N}, p_{1,k} < p_{0,k+1}, j_{1,k} < j_{0,k+1}\ \forall k = 1, \ldots, m-1\}.$$

Then for leaf L with allele $\{\mathrm{IT}[p_{0,k}, p_{1,k}, s_k, j_{0,k}, j_{1,k}]\}_{k=1}^{m}$, AncState(L) is any subset of the alleles generated by its corresponding singleton-wildcards, i.e.

$$\mathrm{AncState}(\mathtt{L}) = \mathrm{Alleles}\left(\bigcup_{k=1,\ldots,m} \mathrm{SGWC}[p_{0,k}, p_{1,k}, s_k, j_{0,k}, j_{1,k}]\right).$$

We now define a recursive procedure to characterize AncState($\cdot$) for all nodes in the tree. We have already established that AncState for a leaf node is characterized by a union of disjoint SGWCs. To recur up the tree, Lemma 1 states that AncState(N) for node N is *also* characterized by a union of disjoint WC/SGWCs.

---

we define wildcard here as all indel tracts that only deactivate targets within a specified range.

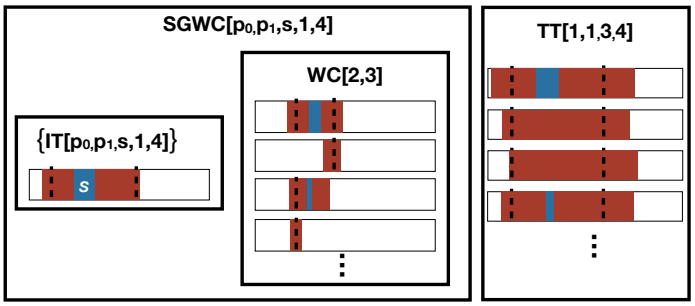


Figure 2.4: Relationship between indel tracts (IT), target tracts (TT), wildcards (WC), and singleton-wildcards (SGWC). Each IT is shown in the context of a barcode. Each box represents a set of ITs. For example, the singleton set $\{\text{IT}[p_0, p_1, s, 1, 4]\}$ is the indel tract that cuts targets 1 and 4, deletes positions p_0 to p_1, and inserts sequence s. Wildcard $\text{WC}[2, 3]$ contains all indel tracts that only deactivate targets 2 and/or 3. $\text{SGWC}[p_0, p_1, s, 1, 4]$ is the union of the singleton set $\{\text{IT}[p_0, p_1, s, 1, 4]\}$ and the internal wildcard $\text{WC}[2, 3]$. $\text{TT}[1, 1, 3, 4]$ is the set of indel tracts that cut targets 1 and 3 and deactivate 1 to 4.

Lemma 1. *Consider any internal node* $\mathtt{N}$ *with children nodes* $\mathtt{C}_1, \ldots, \mathtt{C}_K$. *For each child* $\mathtt{C}_k$, *suppose*

$$\text{AncState}(\mathtt{C}_k) \subseteq \text{Alleles}\left(\bigcup_{m=1}^{M_{\mathtt{C}_k}} D_{\mathtt{C}_k,m}\right) \tag{2.8}$$

where $\{D_{\mathtt{C}_k,m}\}_{m=1}^{M_{\mathtt{C}_k}}$ *are pairwise disjoint wildcards and/or singleton-wildcards. Then,* $\text{AncState}(\mathtt{N})$ *can be written in the form of* (2.8) *where* $\{D_{\mathtt{N},m}\}_{m=1}^{M_{\mathtt{N}}}$ *are disjoint wildcards and/or singleton-wildcards and is equal to the non-empty intersections of* $D_{\mathtt{C}_1,m_1} \cap \ldots \cap D_{\mathtt{C}_K,m_K}$, *i.e.*

$$\{D_{\mathtt{C}_1,i_1} \cap \ldots \cap D_{\mathtt{C}_K,m_K} : m_1 = 1, \ldots, M_{\mathtt{C}_1}, \cdots, m_K = 1, \ldots, M_{\mathtt{C}_K}\} \setminus \emptyset. \tag{2.9}$$

In practice, we use the recursive algorithm in Section A.2 of the Appendix to compute $\text{AncState}(\cdot)$ *exactly* for additional computational efficiency.

2.4 Lumpability

The previous section discussed approximating the likelihood by summing over likely ancestral states. Nevertheless, there are still an infinite number of these likely ancestral states.

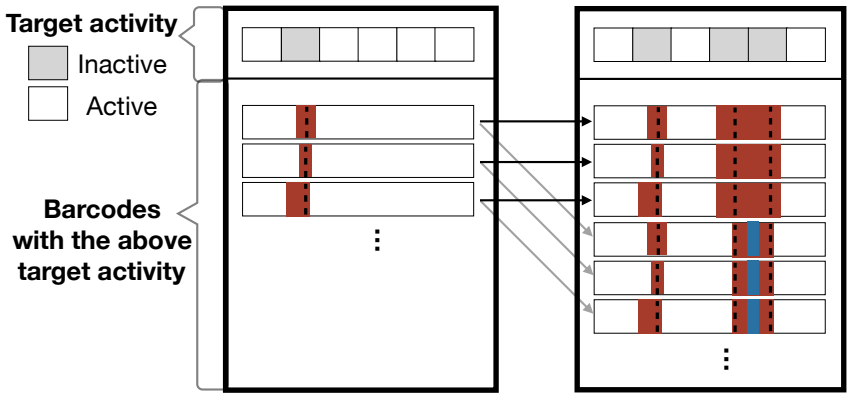


Figure 2.5: An example of lumping together barcodes that share the same target activity. The two outer boxes correspond to two of the lumped states. The left box is the grouped state for possible ancestral barcode states where the second target is no longer active, while the right box represents when the second, fourth, and fifth targets are no longer active. The arrows represent possible transitions and the color represents the transition rates. Notice that each barcode in the left box has the same set of outgoing arrows. To show that the states are lumpable, we show that the total transition rate out of a barcode in the left box to the right box is identical for all barcodes in the left box.

Next, we use Assumption 2 and efficiently compute the approximate likelihood by marginalizing over a small number of "lumped" states.

Lumpability, a well-studied property for Markov chains, states that the behavior of a Markov process can be described by a Markov process over the lumped states [**?** **?**] (Fig 2.5):

Definition 1. *Let $X(t)$ be a continuous time Markov chain with state space Ω. If there exists a partition $\{A_1, \ldots, A_M\}$ of Ω and a continuous time Markov chain $Y(t)$ with state space $\{A_1, \ldots, A_M\}$ such that*

$$\Pr(X(t) \in A_i) = \Pr(Y(t) = A_i) \quad \forall i = 1, \ldots, M, \tag{2.10}$$

then X is lumpable.

If we can find a partition that satisfies (2.10), then we can calculate the likelihood over the lumped states instead. The main practical hurdle in using lumpability is finding such a

partition [**?**].

There is relatively little work on using lumpability in phylogenetics. The one application in **?**] calculates the likelihood of a codon model approximately by assuming states are lumpable, even though this is not satisfied. Here we show that lumpability is satisfied exactly in our setting. Since our solution partitions the state space differently at each node, we must extend Felsenstein's pruning algorithm [**?**] to calculate the approximate likelihood (2.4).

We will define a partition of Ω at node $\mathtt{N}$ denoted $\{g(b;\mathtt{N}) : b \in B\}$ for some index set B. We partition the states based on their target status and whether or not they are likely ancestral states (Fig 2.5), as defined below.

Definition 2. *Define index set B to be $\{0,1\}^M \cup \{\mathsf{other}\}$.*

For internal tree node $\mathtt{N}$, partition the state space Ω into

$$\begin{cases} g(b;\mathtt{N}) = \{a \in \text{AncState}(\mathtt{N}) : \text{TargStat}(a) = b\} & \forall\, b \in \{0,1\}^M \\ g(\mathsf{other};\mathtt{N}) = \Omega - \text{AncState}(\mathtt{N}). \end{cases} \tag{2.11}$$

For leaf node $\mathtt{N}$, partition the state space Ω into

$$\begin{cases} g(b;\mathtt{N}) = \{a_{\mathtt{N}}\} & \text{if } b = \text{TargStat}(a_{\mathtt{N}}) \\ g(b;\mathtt{N}) = \emptyset & \text{if } b \in \{0,1\}^M \text{ and } b \neq \text{TargStat}(a_{\mathtt{N}}) \\ g(\mathsf{other};\mathtt{N}) = \Omega - \{a_{\mathtt{N}}\}. \end{cases} \tag{2.12}$$

Using Assumption 2, we prove in Lemma 4 (see Appendix) that for any $b, b' \in \{0,1\}^M$, the instantaneous transition rate from any allele a in $g(b;\mathtt{N})$ to $g(b';\mathtt{N})$ is the same. Therefore we can construct a Markov process over the lumped states $\{g(b;\mathtt{N}) : b \in B\}$, calculate its instantaneous transition rate matrix $Q_{\text{lump},\mathtt{N}}$ as defined in Lemma 4, and exponentiate this

matrix to calculate the transition probability

$$\Pr\left(X_{\mathtt{N}}(t) \in g(b';\mathtt{N}) | X_{\mathtt{N}}(0) \in g(b;\mathtt{N})\right) = \left\{e^{Q_{\text{lump},\mathtt{N}}t}\right\}_{b,b'} \quad \forall b, b' \in B. \tag{2.13}$$

The following theorem extends Felsenstein's pruning algorithm to calculate the phylogenetic likelihood by marginalizing over at most 2^M lumped states. For $b \in B$, let the probability of observing the data (marginalizing over likely ancestral states) given that the allele at node $\mathtt{N}$ is in partition $g(b;\mathtt{N})$ be denoted

$$p_{\mathtt{N}}(b) = \Pr\left(X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in \text{Leaves}(\mathtt{N}) | X_{\mathtt{N}}(t_{\mathtt{N}}) \in g(b;\mathtt{N})\right). \tag{2.14}$$

Theorem 1. *Suppose Assumptions 1 and 2 and Approximation 1 hold. For any internal tree node* $\mathtt{N}$*, target status* b*, and nonempty allele group* $g(b;\mathtt{N})$*, we have*

$$p_{\mathtt{N}}(b) = \prod_{\mathtt{C} \in \text{children}(\mathtt{N})} \left\{ \sum_{\substack{b' \in \{0,1\}^M \\ g(b';\mathtt{C}) \neq \emptyset}} p_{\mathtt{C}}(b') \Pr\left(X_{\mathtt{C}}(t_{\mathtt{C}}) \in g(b';\mathtt{C}) | X_{\mathtt{C}}(0) \in g(b;\mathtt{C})\right) \right\}. \tag{2.15}$$

where $\Pr\left(X_{\mathtt{C}}(t_{\mathtt{C}}) \in g(b';\mathtt{C}) | X_{\mathtt{C}}(0) \in g(b;\mathtt{C})\right)$ *is defined in* (2.13).

2.5 Caterpillar trees

We would like to estimate trees at the finest resolution possible. C-S parsimony produces estimates at a coarse resolution: If the ordering between nodes is ambiguous, they are all grouped under a single parent node. We propose estimating trees resolving multifurcations at the finer resolution of caterpillar trees (Fig 2.6a). A caterpillar tree is one where all subtrees branch off of a central path called the spine. We do not assume that the true tree is a caterpillar tree. Rather, we use the caterpillar tree to uncover the order in which indel tracts were introduced.

Calculating the likelihood for all possible branch orderings in a caterpillar tree is compu-

tationally intractable because there are $K!$ such orderings for K children nodes. We sidestep this issue by approximating the likelihood using another lower bound: we only marginalize over mutation histories where the alleles are constant along caterpillar spines. To see why this a reasonable approximation, consider the example in Fig 2.6b. Because the GESTALT mutation process is irreversible, the only possible ancestral state at many internal nodes along the spine is the unmutated barcode. In other words, the allele was constant along most of the spine. Thus, we propose the following approximation of the likelihood.

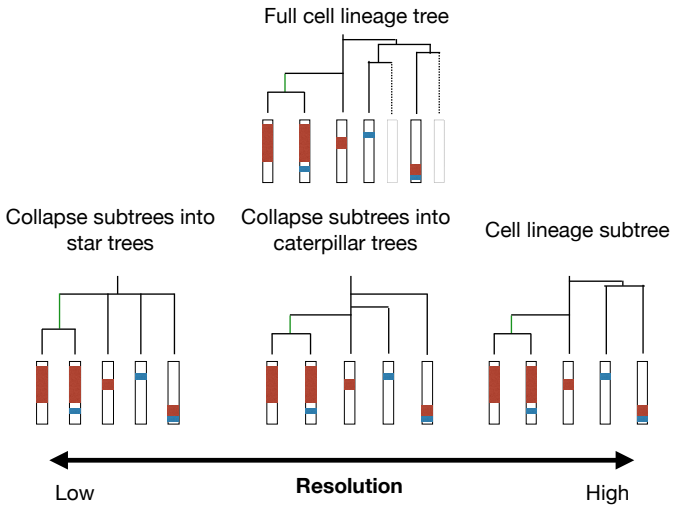
Approximation 2. *We approximate* $\Pr\left(X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in \text{Leaves}(\mathbb{T})\right)$ *by considering only the mutation histories that have a constant allele along the caterpillar spines:*

$$\Pr\left(X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in \text{Leaves}(\mathbb{T}), \textit{alleles are constant on all spines}\right). \tag{2.16}$$

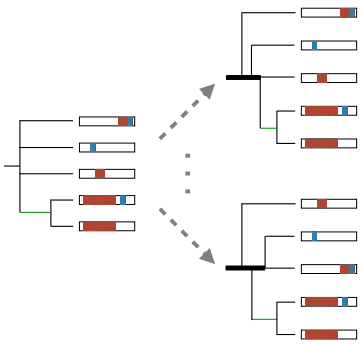
This approximation is particularly attractive because it can be computed using the same mathematical expression regardless of the ordering of the children nodes. This allows us to tune the ordering in the caterpillar tree by solving a single continuous optimization problem (Fig 2.6b).

We re-parameterize the branch lengths for caterpillar branches (Fig 2.6c). Consider a caterpillar tree with root node $\mathtt{N}$ and child node $\mathtt{C}$. Let $\ell_{\mathtt{C}}$ indicate the distance between $\mathtt{C}$ and $\mathtt{N}$. For $\beta_{\mathtt{C}} \in (0, 1)$, let $\beta_{\mathtt{C}}\ell_{\mathtt{C}}$ be branch length of $\mathtt{C}$. We can capture all possible orderings for the caterpillar tree rooted at $\mathtt{N}$ by varying the values of $(\ell_{\mathtt{C}}, \beta_{\mathtt{C}})$ for each child node $\mathtt{C}$.

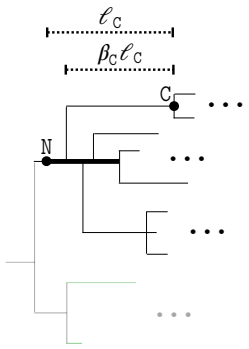
With this parameterization, we now extend Theorem 1 to calculate (2.16). For allele a and node $\mathtt{N}$, define $\tilde{p}_{\mathtt{N}}(a)$ the same as (2.14) but now assuming both Approximations 1 and 2. Again, apply Felsenstein's pruning algorithm to recursively compute $\tilde{p}_{\mathtt{N}}$ for each node $\mathtt{N}$.



(a) We show the subtree of a full cell lineage tree (top) from low (left) to high (right) resolutions. The lowest resolution collapses ambiguous orderings as multifurcating nodes. To infer ordering information, we increase the resolution of the tree by projecting onto the space of caterpillar trees (middle).



(b) Our method resolves multifurcating nodes as caterpillar trees. There are many possible orderings in a caterpillar tree, two of which are shown above. We tune the ordering by maximizing the penalized log likelihood.



(c) We parameterize the branch lengths in a caterpillar tree by associating each child node C with parameters $\ell_C > 0$ and $\beta_C \in [0, 1]$. The length of the caterpillar spine, highlighted in bold, is the maximum value of $\ell_C(1 - \beta_C)$ over all children nodes C.

Figure 2.6: Caterpillar tree goals and parameterization

However, if $\mathtt{N}$ is the root of a caterpillar tree, then $\tilde{p}_{\mathtt{N}}(a)$ is equal to

$$\Pr\left(X_{\mathtt{N}}(t_{\text{spine}}) = a | X_{\mathtt{N}}(0) = a\right) \prod_{\mathtt{C} \in \text{children}(\mathtt{N})} \left\{ \sum_{a' \in \Omega} \Pr\left(X_{\mathtt{C}}(\ell_{\mathtt{C}}\beta_{\mathtt{C}}) = a' | X_{\mathtt{C}}(0) = a\right) \tilde{p}_{\mathtt{C}}(a') \right\} \tag{2.17}$$

where $t_{\text{spine}} = \max\{\ell_{\mathtt{C}}(1 - \beta_{\mathtt{C}}) : \mathtt{C} \in \text{children}(\mathtt{N})\}$. To efficiently calculate the likelihood, we marginalize over the corresponding lumped states instead.

In summary, we have shown how to tune caterpillar trees by solving a single continuous optimization problem. Compared to considering each tree topology separately, this approach is more computationally efficient and performs a more comprehensive search over tree space in practice.

2.6 Model implementation

We briefly describe our specific model implementation here and leave details to Section A.5. Each target is associated with a different cut rate λ_j. If targets j_0 and j_1 are active, the rate for cutting target j_0 is λ_{j_0} and the rate for simultaneously cutting targets j_0 and j_1 is $\omega(\lambda_{j_0} + \lambda_{j_1})$ for some $\omega > 0$. We model the distribution of deletion lengths using zero-inflated truncated negative binomial random variables (RVs) and insertion lengths using zero-inflated negative binomial RVs. Finally, Section A.7 discusses our actual code implementation, which includes an additional approximation used to limit memory usage.

3 Estimation method

Now that the approximate likelihood is computationally tractable, we are ready to estimate the cell lineage tree and mutation model parameters.

3.1 A simple approach

Consider the following estimation procedure: Given a pool of candidate tree topologies, select the one with the highest likelihood after optimizing over its corresponding parameters.

Unfortunately, this procedure can be highly inaccurate for existing GESTALT datasets, where we must estimate thousands of parameters given data generated by a single barcode. Because this problem is high-dimensional, we found in simulations that the maximum likelihood estimate tends to overestimate the length of the leaf branches and the variance of the target rates.

3.2 Penalization

To improve the estimation accuracy, we propose performing penalized maximum likelihood estimation instead. We penalize large differences in the branch lengths $\boldsymbol{\ell}$ and target rates $\boldsymbol{\lambda}$ using

$$\text{Pen}_{\boldsymbol{\kappa}}(\theta) = \kappa_1 \left\| \log \boldsymbol{\lambda} - \frac{1}{M} \sum_{i=1}^{M} \log(\lambda_i) \right\|_2^2 + \kappa_2 \left\| \log \boldsymbol{\ell} - \frac{1}{L} \sum_{i=1}^{L} \log(\ell_i) \right\|_2^2,$$

where $\kappa_1, \kappa_2 > 0$ are penalty parameters and L is dimension of $\boldsymbol{\ell}$. A similar branch-length penalty was considered in **?**], but they assume the topology is known.

We cannot directly combine this penalty with the simple approach in Section 3.1 because different topologies may naturally have larger branch length penalties. To ensure that the penalized log likelihood (PLL) is comparable between different topologies, we use an iterative search procedure instead.

For fixed penalty parameters, our estimation procedure follows Algorithm 4. We initialize the topology by selecting a random parsimony-optimal tree from C-S parsimony. At each iteration, we select a random subtree and consider candidate subtree-prune-regraft (SPR) moves that preserve the parsimony score, since parsimony-optimal trees tend to have the highest likelihoods (Fig 2.21). To make sure the PLLs are comparable, we choose a random leaf from the subtree, calculate the likelihood for the tree from regrafting only this random leaf, and calculate the penalty with respect to the *shared* subtree (Fig 2.7).[3] We select the

[3]We chose to regraft a random leaf from the subtree to make the penalty as comparable as possible across candidate SPR moves. Using this approach, the resulting trees from the SPR move differ by only a single

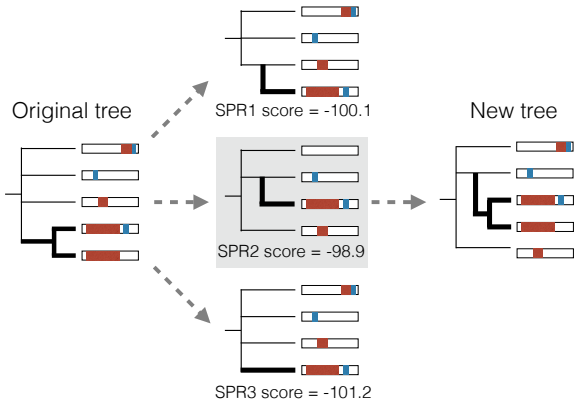


Figure 2.7: To tune the tree topology, we select a random subtree (left) and score possible SPR moves that preserve the parsimony score by selecting a random subleaf and calculating the maximized penalized log likelihood of the resulting tree (middle). We then update the tree by applying the SPR move with the highest penalized log likelihood (right).

SPR move with the highest PLL. In simulations, we found that this procedure progressively improves the tree estimate (Fig 2.20b). See Section A.4 for a discussion on tuning penalty parameters.

4 Simulation engine and results

We built a simulation engine of the GESTALT mutation process during embryonic development. Since cell divisions during embryonic development begin in a fast metasynchronous fashion and gradually become more asynchronous [**?**], the simulation engine generates a cell lineage tree by performing a sequence of synchronous cell divisions followed by a birth-death process where the birth rate decays with time. We mutate the barcode along this cell lineage tree according to our model of the GESTALT mutation process. The simulation engine generates data that closely resembles the data collected from zebrafish embryos in **?**] (Fig 2.8). We can input different barcode designs into the simulation engine to understand how they affect our ability to reconstruct the cell lineage tree.

We used two evaluation metrics to evaluate tree estimates: BHV distance [**?**] and a

leaf; Whereas, if we regrafted an entire subtree, the resulting trees will differ significantly.

Algorithm 4 Cell lineage tree reconstruction for penalty parameter $\boldsymbol{\kappa}$

1: Initialize tree $\mathbb{T}$. Let the sequenced GESTALT barcodes be denoted D.
2: **for** Iteration k **do**
3: Pick a random subtree from $\mathbb{T}$. Select one of the leaves `C` of the subtree.
4: **for** each possible SPR move involving the subtree that doesn't change the parsimony score (including the no-op) **do**
5: Construct $\mathbb{T}'$ by applying the SPR to leaf `C`; let $\mathbb{T}'_{\text{shared}}$ be the subtree of $\mathbb{T}'$ when excluding `C`
6: Evaluate the penalized log likelihood for the SPR move, maximized with respect to parameters θ:

$$\max_{\theta} \log \underbrace{\Pr(D, \text{alleles are constant on caterpillar spines, no merging events}; \mathbb{T}', \theta)}_{\text{Approximation to the likelihood}} - \underbrace{\text{Pen}_{\boldsymbol{\kappa}}(\mathbb{T}'_{\text{shared}}, \theta)}_{\text{Penalty on branch lengths and mutation parameters}}$$

7: **end for**
8: Update the tree $\mathbb{T}$ by performing the SPR move on the subtree with the highest penalized log likelihood
9: **end for**

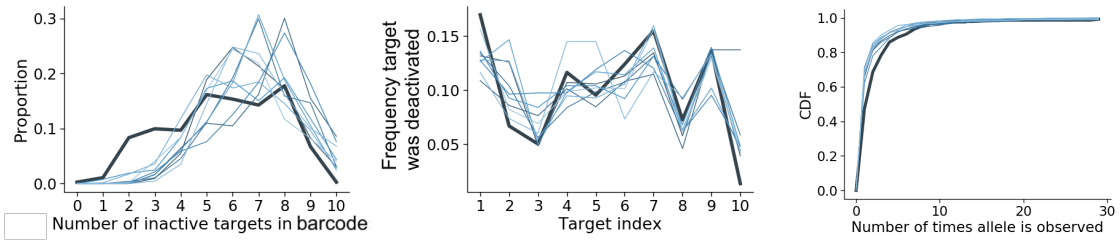


Figure 2.8: Comparison of simulated data (each thin line is a replicate) versus observed alleles from a fish at 4.3 hours post-fertilization (bold line). The distribution of inactive targets and the number of times an allele is observed are similar.

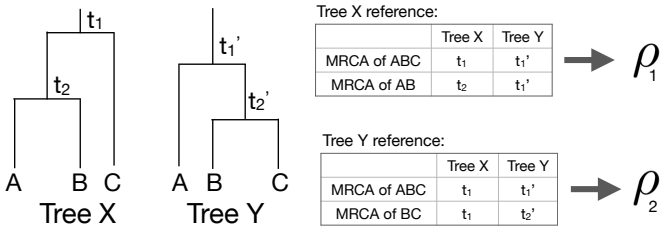


Figure 2.9: Example calculation of the internal node time correlation. For each tree, calculate the heights for each internal node (e.g. t_1, t_2, t_1', t_2') and calculate the correlation between the times of the most recent common ancestors (MRCAs) of corresponding leaf groups. The internal node time correlation is the average of correlations ρ_1 and ρ_2.

new metric we call internal node time correlation. Intuitively, the BHV distance between two trees is the smallest total change in branch lengths to transform one tree into the other (so its minimum value is zero). It is a formal distance metric and therefore enjoys many nice mathematical properties; However, our collaborators found internal node time correlation easier to interpret, particularly when BHV distance is large. Given ultrametric trees X and Y for the same set of leaves, the internal node time correlation is calculated as follows (Fig 2.9):

1. For each internal node in tree X, find the matching node in tree Y that is the most recent common ancestor of the same set of leaves.

2. Calculate the Pearson correlation of the heights of matched nodes.

3. Repeat steps 1 and 2 but swap trees X and Y.

4. Average the two correlation values.

A correlation of 1 means that the trees are exactly the same; the smaller the correlation is, the less similar the trees are.

We compare our method to estimating the tree topology using C-S parsimony [**?**] or neighbor-joining (NJ) [**?**] and then applying semiparametric rate smoothing (`chronos`

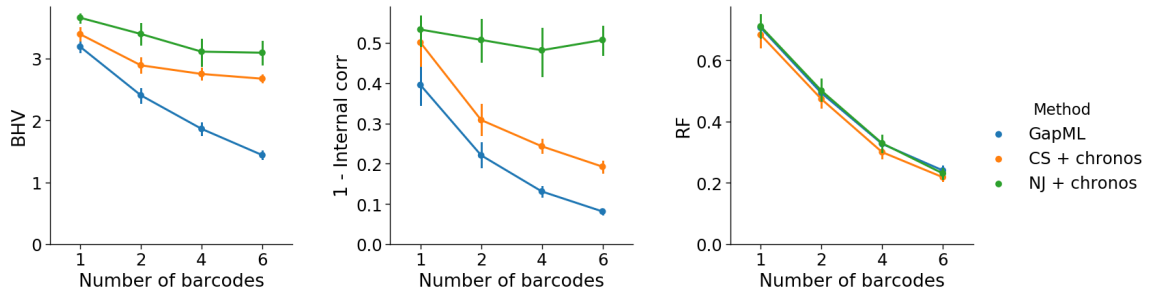


Figure 2.10: Error of trees estimates from GAPML as well as Camin-Sokal parsimony (CS) and neighbor-joining (NJ) with node time estimation by semiparametric rate smoothing (`chronos`). Simulated trees has ≈100 leaves, where the barcode is composed of six targets and the number of barcodes is varied from one to six. GAPML outperforms other methods in terms of BHV (left) and the internal node time correlation metrics (middle). The methods are similar in terms of the Robinson-Foulds (RF) metric (right).

in the `R` package `ape`) to estimate node times [**?**]. We refer to these approaches as "CS+chronos" and "NJ+chronos," respectively. Previous in silico analyses measured accuracy in terms of the Robinson-Foulds (R-F) distance, which only evaluates differences in tree topology [**?**]. However, this is a very coarse metric and fails to recognize that trees with different topologies can still have very similar internal node times. As such, we also evaluate tree estimates using BHV distance and internal node time correlation; We find that GAPML consistently outperforms the other methods with respect to these two metrics (Fig 2.10 and Table 2.1).

Figure 2.10 also shows that GAPML improves in performance as the number of independent barcodes increases. In this simulation, the estimated tree from a single barcode has internal node time correlation of 0.5 with the true tree whereas using six barcodes increases the correlation to 0.9. Even though other analyses have recommended increasing the number of targets in a single barcode to improve tree estimation [**?**], we found that adding independent barcodes is more effective (Fig 2.19).

Section A.9 is a larger simulation study of the method's asymptotic properties. We show that the method continues to improve, even when there are hundreds of barcodes. Section A.9 shows that tree estimates from GAPML are robust to errors resolving ambiguous indels.

Method	BHV	1 - Internal node correlation
GAPML	5.33 (5.06, 5.60)	0.43 (0.39, 0.46)
CS + chronos	6.57 (6.45, 6.9)	0.57 (0.54, 0.60)
NJ + chronos	8.55 (8.51, 8.59)	0.67 (0.66, 0.68)

Table 2.1: Comparison of methods on simulated data using a single barcode with ten targets and around 200 leaves. The 95% confidence intervals are given in parentheses.

5 Real data analysis of a zebrafish

To validate our method, we reconstructed cell lineages using our method and other tree-building methods on GESTALT data from zebrafish [**?**]. As the true cell lineage tree is not known for zebrafish, we employed indirect measures of validity. For each method, we asked (1) if similar conclusions could be made across different biological replicates and (2) if the tree estimates aligned with the known biology of zebrafish development.

The dataset includes two adult zebrafish where cells were sampled from dissected organs. The organs were chosen to represent all germ layers: the brain and both eyes (ectodermal), the intestinal bulb and posterior intestine (endodermal), the heart and blood (mesodermal), and the gills (neural crest, with contributions from other germ layers). The heart was further divided into four samples— a piece of heart tissue, dissociated unsorted cells (DHCs), FACS-sorted GFP+ cardiomyocytes, and non-cardiomyocyte heart cells (NCs). In addition, datasets were collected from embryos at the dome stage (4.3 hours post-fertilization (hpf)), pharyngula stage (30 hpf), and from early larvae (72 hpf), where the cell types are unknown. We set the total height of each tree estimate to $T = 1$.

5.1 Replication of developmental relationships between tissue types

Here, we check if the estimated developmental relationships between tissue types is replicated in the two adult fish samples. For each estimated tree, we calculated the distance between tissues, which we define as the average tree distance between a leaf of one tissue to the closest internal node leading to a leaf from the other tissue, weighted by the allele abundance (Fig 2.11). (All alleles that were found in the blood were removed since blood

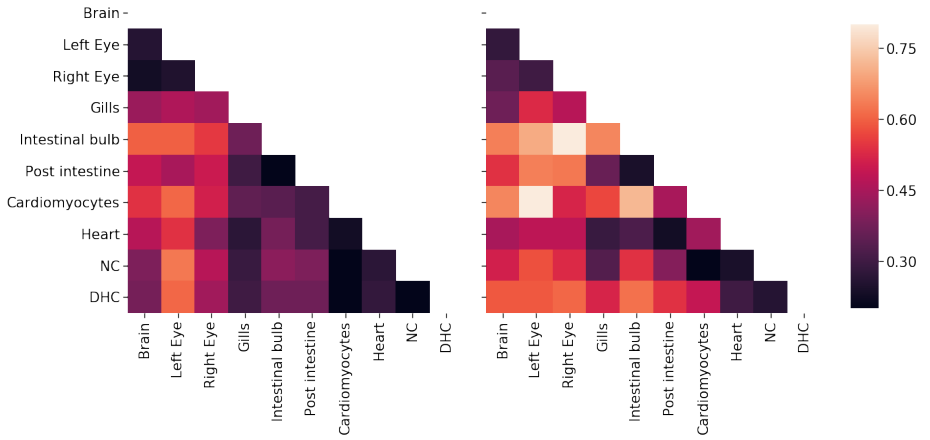


Figure 2.11: Average distance between tissue sources from adult fish 1 (left) and 2 (right) for tree estimates from GAPML. The distance between tissues is the average time from a leaf of one tissue to the closest internal node with a descendant of the other tissue. The shading reflects distance, where bright means far and dark means close. The tissue distances share similar trends between the two fish. For example, the top (brain and eyes) and lower right (heart-related organs) tend to be the darker regions in both distance matrices.

is found in all dissected organs and can confound the relationship between organs [**?**].) Recall that all of the fitting procedures are completely agnostic to any tissue source or cell abundance information. We tested if the correlations were significant by permuting the cell types and abundances in the estimated trees. The correlation was 0.730 ($p < 0.001$) using our method, whereas 'CS+chronos' and 'NJ+chronos' had correlations of 0.306 ($p = 0.21$) and -0.325 ($p = 0.22$), respectively.[4]

5.2 Replication of mutation parameters

Here, we check if the mutation parameters replicate across fish samples. For each time point, the fish replicates were traced using the same GESTALT barcode and processed using the same experimental protocol (Table 2.2). We compared the estimated target rates from our method to those estimated using a model-free empirical average approach where

[4] One might be concerned that our method is consistent across fish replicates because it returns very similar trees regardless of the data. However, this is not the case: When we re-run our method with randomly permuted cell types and abundances, the average correlation between the tissue distances drops to zero.

Fish age	n	Barcode version	GAPML Correlation	Empirical average correlation
4 months	2	7	0.891	0.685
72 hpf	6	7	0.897 (0.855, 0.973)	0.648 (0.616, 0.842)
30 hpf	4	6	0.287(0.287, 0.788)	0.052 (0.052, 0.727)
4.3 hpf	5	7	0.942 (0.942, 0.976)	0.749 (0.714, 0.918)

Table 2.2: Mean Spearman correlation between the estimated target lambda rates across fish replicates (hpf is short for hours post fertilization). 95% confidence intervals via bootstrap are shown in parentheses. The correlation is higher for GAPML estimates, compared to setting rate estimates as the proportion of times each target was cut.

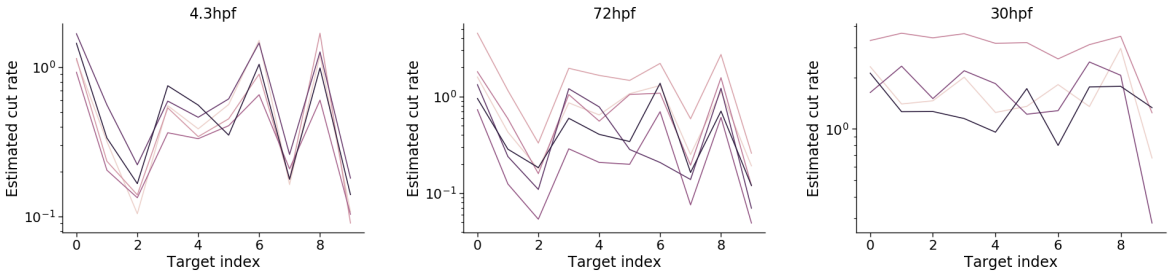


Figure 2.12: Fitted target lambda rates for fish sampled at different time points; Each line corresponds to estimates for a single fish. Fish sampled at 4.3hpf (left) and 72 hpf (middle) used the same barcode and have similar rate estimates. The 30hpf fish (right) used a different barcode and has different rate estimates.

the estimated target cut rate is the proportion of times a cut was observed in that target in the set of unique observed indels. The average correlation between the estimated target rates from our method were much higher than that for the alternate approach (Table 2.2). In fact, we can also compare target cut rates between fish of different ages that share the same barcode, even if the experimental protocols are slightly different. The 4.3hpf and 72hpf fish share the same barcode version, and we find that the target rate estimates are indeed similar (Fig 2.12).

5.3 Recovery of cell-type and germ-layer restriction

It is well known that cells are pluripotent initially and specialize during development. To evaluate recovery of specialization by tissue type, we calculated the correlation between the estimated time of internal tree nodes and the number of descendant tissue types; to evaluate recovery of specialization by germ layer, we calculated the correlation between the estimated time of internal nodes and the number of germ layers represented at the leaves. (As before, all the estimation methods do not use the tissue origin and germ layer labels.) Since any tree should generally show a trend where parent nodes tend to have more descendant cell types than their children, we compared our tree estimate to the same tree but with random branch length assignments and randomly permuted tissue types. Our method estimated much higher correlations compared to these random trees (Table 2.3). We show an example of the node times versus the number of descendant cell types and germ layers in Figure 2.13. The other methods have lower correlation compared to GAPML in all cases, except for 'NJ + chronos' in the second adult fish. However, upon inspection, the correlation is high for 'NJ + chronos' because it estimates that cells are pluripotent for over 90% of the fish's life cycle and specialize during a small time slice at the very end.

Adult Fish	Estimation Method	# tissue types vs time			# germ layers vs time		
		Corr	Random corr	p-value	Corr	Random corr	p-value
1	GAPML	-0.476	-0.176	< 0.001	-0.404	-0.125	< 0.001
	CS+chronos	-0.182	0.037	0.002	-0.142	0.032	0.044
	NJ+chronos	-0.271	-0.126	0.003	-0.179	-0.094	0.084
2	GAPML	-0.547	-0.243	<0.001	-0.437	-0.184	0.002
	CS+chronos	-0.389	0.070	0.001	-0.397	0.090	< 0.001
	NJ+chronos	-0.621	-0.236	<0.001	-0.475	-0.183	0.001

Table 2.3: Correlations between the number of descendant cell types/germ layers vs. the time of internal nodes for different estimation methods. For each tree estimate, we create random trees by fixing the topology but shuffling cell types and assigning random branch lengths. The correlation for these random trees are shown. The p-value is calculated with respect to these random trees.

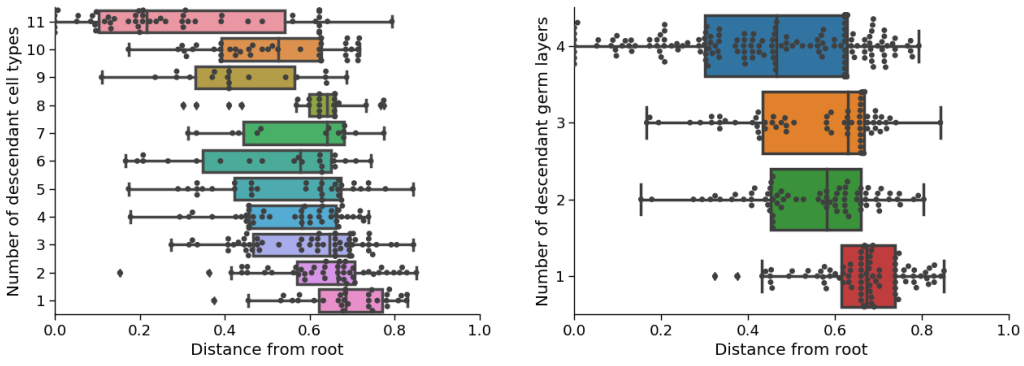


Figure 2.13: Visualization of internal node times for the first adult fish using GAPML, stratified by the number of descendant cell types (left) and germ layers (right). The estimated node times recover the known phenomenon of cell type and germ layer restriction during organism development.

5.4 Analysis of the zebrafish GESTALT data

Now we analyze the fitted trees of the adult zebrafish in more detail to check if summaries concord with known zebrafish biology; and generate new hypotheses about zebrafish development and the experimental procedure.

The estimated tissue distance matrices (Fig 2.11) present a coarse summary of the developmental process. We observe that they recapitulate some well-established facts about zebrafish development. For example, we estimate that tissue types from the endoderm and mesoderm tended to be closer. This signal potentially captures the migration of the endoderm and mesoderm through the blastopore, isolating them from the ectoderm [**?**]. In addition, previous studies established that gills form when the anterior part of the intestine grows toward and fuses with the body integument [**?**]. Likewise, our method estimates that gill cells are closer to tissues from the endoderm and mesoderm.

Fig 2.11 also shows that the GFP+ cardiomyocytes tend to be farthest away from other tissue types, which could be either a developmental signal or an artifact of the experimental protocol. GFP+ cardiomyocytes were sorted using fluorescence-activated cell and this purity could drive their separation from the other more heterogeneous organ populations. An interesting biological speculation would be that the myocardial cells are one of the first cells to differentiate during vertebrate embryo development, which could be driving this observed signal [**?**].

The cell lineage tree estimated using GAPML provides significantly more detail than the C-S parsimony tree inferred in **?**] (Fig 2.14). Unlike the C-S tree, the GAPML tree infers relative timing of events across parallel subtrees and infers the order of events.

The full tree estimated using GAPML for the first adult zebrafish is displayed in Figure 2.24. The raw tree data and tools for visualizing the tree are available at `https://github.com/matsengrp/gapml`. Its longest caterpillar spine starts from the root node and connects all the major subtrees that share no indel tracts. As the zebrafish embryo rapidly divides from the single-cell stage, these initial CRISPR editing events establish the

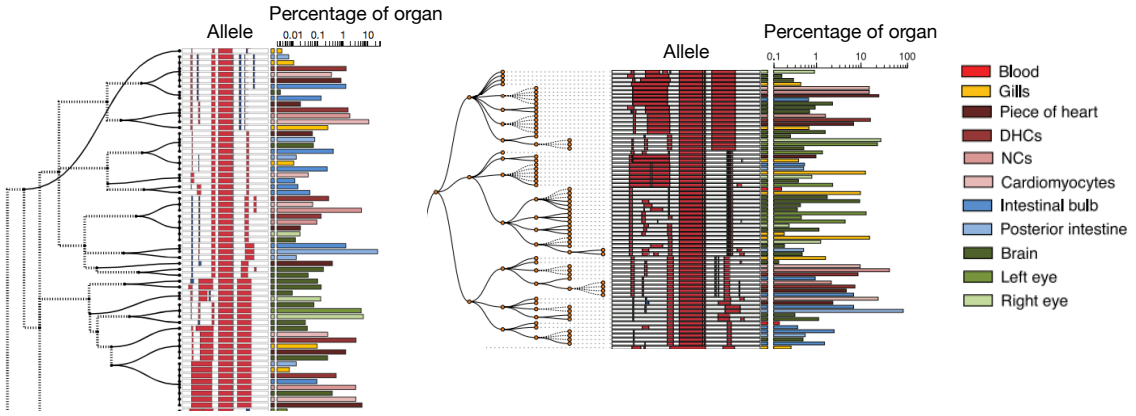


Figure 2.14: Subtree from trees estimated using GAPML (left) and Camin-Sokal parsimony (right). Red and blue bars in the allele indicate deletions and insertions, respectively. Alleles observed in multiple organs are plotted on separate lines per organ. The barchart on the right of each subfigure indicates the proportion of cells in that organ represented by each allele. The dashed lines in the GAPML tree correspond to the caterpillar spines.

founding cell in each subtree. We see that the last three subtrees at the end of this spine (farthest away from the root) were observed primarily in the intestinal bulb and the posterior intestine. This concords with our understanding of zebrafish development: of the dissected organs, the digestive tract is the last to fully differentiate [**?**]. These examples show that this more refined lineage tree can inspire new and interesting biological questions and provide a means to answer them.

5.5 Analysis of mutation parameters

Finally, our estimated mutation parameters (Table 2.5) can guide redesigns of the GESTALT barcode. For example, we estimate that Targets 1 and 9 in the barcode from **?**] have the highest cut rates. To decrease the frequency of inter-target deletions, one suggestion is therefore to move those targets to the center of the barcode and targets with lower cut rates to the outside.

6 Discussion

We have proposed a statistical model for the mutation process of GESTALT, a lineage-tracing technology that leverages a synthetic barcode of CRISPR/Cas9 targets to record development. Our method, GAPML, estimates the cell lineage tree and the mutation parameters of this system. GAPML outperforms existing methods on simulated data, provides more consistent results across biological replicates, and outputs trees that better concord with our understanding of developmental biology. Its performance will continue to improve with our ability to integrate more barcodes.

Our method provides a number of technical contributions to the phylogenetics literature. Because the GESTALT mutation process violates many of the classical phylogenetic assumptions, we have introduced new assumptions and methods to make the problem computationally tractable. We believe these techniques could be useful for other phylogenetic problems where the common assumptions do not hold.

A limitation of our method is that it treats the barcode as a molecular clock, even though its mutation rates can actually vary due to cell state, e.g. chromatin state and transcriptional activity. Future work includes relaxing the molecular clock assumption, as well as quantifying the uncertainty of our estimates and merging data across sample replicates.

Finally, our methods may be useful for analyzing other CRISPR-based cell lineage tracing technologies. GAPML is most readily applied to techniques that insert arrays of Cas9 targets [**?** **?**] or those that mutate transgenes in organisms [**?** **?**]. More work needs to be done to adapt it to homing CRISPR guide RNA systems [**?** **?**].

A Appendix

A.1 Useful guides for reading the proofs

Symbol	Description	Eq.
$X_{\texttt{N}}(t)$	Markov process along branch ending with node N	
$a_{\texttt{N}}$	Observed allele at leaf N	
pos(j)	Positions of target j in the unmodified barcode	
$c(j)$	Cut site of target j	
Leaves(N)	Leaves of node N	
Desc(N)	Descendants of node N	
TargStat(a)	Status of targets in allele a. 1 in position j indicates target j is inactive	(2.2)
IT$[p_0, p_1, s, j_0, j_1]$	Indel tract that cuts targets j_0 and j_1, deletes positions p_0 to $p_1 - 1$, inserts s	
TT$[j_0', j_0, j_1, j_1']$	Target tract: all indel tracts that cut targets j_0 and j_1 and deactivate targets j_0' to j_1'	(2.3)
TT(d)	The target tract that indel tract d belongs to	
WC$[j_0, j_1]$	Wildcard: any indel tract that only deactivates targets with indices j_0 to j_1	(2.6)
SGWC$[p_0, p_1, s, j_0, j_1]$	Singleton-wildcard: union of an indel tract and its inner wildcard	(2.7)
AncState(N)	The set of likely ancestral states for node N	(2.5)

Table 2.4: Notation used in this chapter

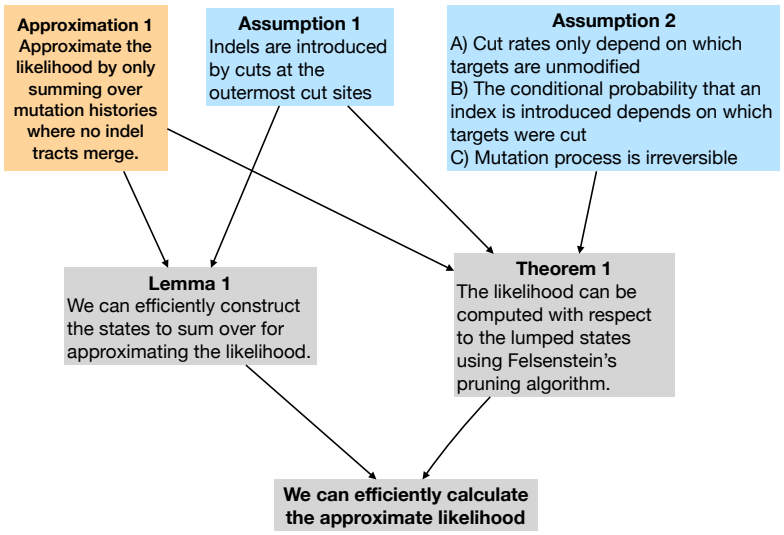


Figure 2.15: A guide for how assumptions, approximations, and derived results connect and lead to our final algorithm for approximating the likelihood.

A.2 Calculating possible ancestral states

Proof for Lemma 1

First, we present the following lemma.

Lemma 2. *The intersection of any two wildcard/singleton-wildcards must also be a wildcard, singleton-wildcard, or the empty set.*

Proof. Consider the case where we intersect two distinct two singleton-wildcards $D = \text{SGWC}[p_0, p_1, s, j_0, j_1]$ and $D' = \text{SGWC}[p'_0, p'_1, s', j'_0, j'_1]$. Since $\text{IT}[p_0, p_1, s, j_0, j_1] \neq \text{IT}[p'_0, p'_1, s', j'_0, j'_1]$, then $D \cap D'$ does not contain either indel tract and $D \cap D'$ is equal to the intersection of their inner wildcards $\text{WC}[j_0 + 1, j_1 - 1]$ and $\text{WC}[j'_0 + 1, j'_1 - 1]$. It is staightforward to show that the intersection of a wildcard with another wildcard or with a singleton-wildcard satisfies the lemma. □

Next, we introduce some more notation. For indel tract set D, let $\text{targ}_{\text{start}}(D)$ be the leftmost target that can be deactivated some $d \in D$ and $\text{targ}_{\text{end}}(D)$ be the rightmost target

that can be deactivated some $d \in D$.

We now present the proof for Lemma 1.

Proof. We prove the case where internal node $\mathtt{N}$ has two children nodes $\mathtt{C}_1$ and $\mathtt{C}_2$. It is easy to prove the case where $\mathtt{N}$ has 3+ children by induction.

For each pair $D_{\mathtt{C}_1,m_1}$ and $D_{\mathtt{C}_2,m_2}$, define its intersection as $D_{\mathtt{N},m_1,m_2}$. Then

$$\begin{aligned}
\text{AncState}(\mathtt{N}) &= \text{AncState}(\mathtt{C}_1) \cap \text{AncState}(\mathtt{C}_2) && (2.18)\\
&\subseteq \text{Alleles}\left(\bigcup_{m_1=1}^{M_{\mathtt{C}_1}} D_{\mathtt{C}_1,m_1}\right) \cap \text{Alleles}\left(\bigcup_{m_2=1}^{M_{\mathtt{C}_2}} D_{\mathtt{C}_2,m_2}\right) && (2.19)\\
&= \text{Alleles}\left(\bigcup_{m_1=1}^{M_{\mathtt{C}_1}} \bigcup_{m_2=1}^{M_{\mathtt{C}_2}} D_{\mathtt{N},m_1,m_2}\right) && (2.20)
\end{aligned}$$

Consider any m_1, m_2. Per Lemma 2, if $D_{\mathtt{N},m_1,m_2}$ is not the empty set, it is a wildcard or a singleton-wildcard that satisfies:

$$\begin{aligned}
\text{targ}_{\text{start}}(D_{\mathtt{N},m_1,m_2}) &\geq \max\left(\text{targ}_{\text{start}}(D_{\mathtt{C}_1,m_1}), \text{targ}_{\text{start}}(D_{\mathtt{C}_2,m_2})\right) && (2.21)\\
\text{targ}_{\text{end}}(D_{\mathtt{N},m_1,m_2}) &\leq \min\left(\text{targ}_{\text{end}}(D_{\mathtt{C}_1,m_1}), \text{targ}_{\text{end}}(D_{\mathtt{C}_2,m_2})\right). && (2.22)
\end{aligned}$$

Since the associated wildcard/singleton-wildcards $\{D_{\mathtt{C},m} : m = 1, ..., M_{\mathtt{C}}\}$ for each child node $\mathtt{C}$ are pairwise disjoint, then the set of intersections $\{D_{\mathtt{N},m_1,m_2}\}$ must also be pairwise disjoint wildcards/singleton-wildcards. □

Calculating AncState(·) *exactly*

For efficiency reasons, we are not satisfied with computing supersets of AncState(·); rather, we would like to concisely express the set of alleles that is exactly equal to AncState(·). The only case in which the algorithm computes a strict superset of AncState($\mathtt{N}$) is when the alelles observed at the leaves of $\mathtt{N}$ imply that the observed indel tracts must be introduced in a particular order. For example, if an allele has indel tracts d_1 and d_2, we know that d_1

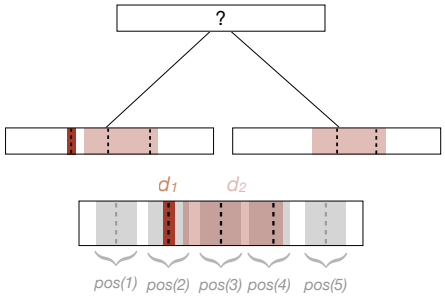


Figure 2.16: **Bottom**: An example of an allele with two indel tracts d_1 and d_2 where d_1 must have been introduced before d_2, because d_1 cuts target 2 while d_2 deactivates target 2 through 4. **Top**: A two-leaf tree where one leaf is the example allele and the other leaf is an allele with only d_2. Since d_1 must be introduced before d_2, the only possible ancestral state of the parent is an unmodified allele. On the other hand, if d_2 did not overlap with pos(2), we can simply take the intersection of the two alleles to get a possible ancestral state.

must be introduced before d_2 if d_1 cuts target j and d_2 deactivates target j (Figure 2.16). To indicate such orderings, we use the notation $d \in a \Rightarrow d' \in a$ to denote that "if indel tract d is in allele a, then indel tract d' must also be in a." The set of alleles respecting this ordering constraint is denoted

$$\text{Order}\left(d \Rightarrow d'\right) = \left\{a \in \Omega : d \in a \implies d' \in a\right\}.$$

Note that $\text{Order}(d \Rightarrow d')$ contains all alleles that do not include d.

The following lemma builds on Lemma 1. Whereas Lemma 1 constructed supersets of AncState(·), the following lemma constructs sets *exactly* equal to AncState(·). Using the following lemma, the recursive algorithm now also requires parent nodes to adopt ordering requirements from their children. Note that ordering requirements only involve observed indel tracts. The proof for the lemma is straightforward so we omit it here.

Lemma 3. *For any leaf node* $\mathtt{L}$*, suppose its observed allele is* $\{d_m : m = 1, ..., M_{\mathtt{L}}\}$ *for some* $M_{\mathtt{L}} \geq 0$*, where* $d_m = \text{IT}[p_{0,m}, p_{1,m}, s, j_{0,m}, j_{1,m}]$*. Its set of ordering requirements, denoted*

$\text{Orderlist}_{\mathtt{L}}$, *is*

$$\left\{\text{Order}\left(d_m \Rightarrow d_{m'}\right) : m, m' \in \{1, ..., M_{\mathtt{L}}\}, m \neq m', p_{1,m} \in \text{pos}(j_{0,m'}) \text{ or } p_{0,m} \in \text{pos}(j_{1,m'})\right\}.$$

Then

$$\text{AncState}(\mathtt{L}) = \text{Alleles}\left(\bigcup_{m=1}^{M_{\mathtt{L}}} D_{\mathtt{L},m}\right) \cap \left[\bigcap_{\text{Order}(d\Rightarrow d')\in\text{Orderlist}_{\mathtt{L}}} \text{Order}(d \Rightarrow d')\right] \tag{2.23}$$

where $D_{\mathtt{L},m} = \text{SGWC}[p_{0,m}, p_{1,m}, s_j, j_{0,m}, j_{1,m}]$.

Similarly, for any internal node $\mathtt{N}$*, we can also write* $\text{AncState}(\mathtt{N})$ *in the form of* (2.23)*. If node* $\mathtt{N}$ *has children nodes* $\mathtt{C}_1$*, ...,* $\mathtt{C}_K$*,* $\{D_{\mathtt{N},m}\}_{m=1}^{M_{\mathtt{N}}}$ *are pairwise disjoint wildcards and/or singleton-wildcards satisfying* (2.9) *and*

$$\underset{\mathtt{N}}{\text{Orderlist}} = \left\{\text{Order}(d \Rightarrow d') \in \left[\bigcup_{k=1}^{K} \underset{\mathtt{C}_k}{\text{Orderlist}}\right] : d \in \left[\bigcup_{m=1}^{M_{\mathtt{N}}} D_{\mathtt{N},m}\right]\right\}.$$

Proof. We will prove the lemma for any internal node with children nodes $\mathtt{C}_1$ and $\mathtt{C}_2$. We can easily prove the case for nodes with more than two children by applying the result iteratively.

We give a proof by induction. Recall the definition of AncState, which is given in (2.5). Then it is easy to see that the base case at the leaf nodes, specified by (2.23), holds. Any allele $a \preceq a_{\mathtt{L}}$ must be composed of observed indel tracts in $a_{\mathtt{L}}$ and/or indel tracts masked by the the observed indel tracts in $a_{\mathtt{L}}$. It also must respect all the ordering requirements specified in $\text{Orderlist}_{\mathtt{L}}$, since an indel tract cannot be introduced if its target is deactivated already. Therefore the left hand side is a subset of the right hand side. Next, consider any allele a from the right hand side. To see that $a \preceq a_{\mathtt{L}}$, note that we can apply the remaining indel tracts from $a_{\mathtt{L}}$ (i.e. those not in a) in the order they were introduced into $a_{\mathtt{L}}$.

The rest of the induction is simple. We defined $\text{AncState}(\mathtt{N})$ is defined as the intersection of $\text{AncState}(\mathtt{C}_1)$ and $\text{AncState}(\mathtt{C}_2)$. Likewise, the right hand side for characterizing

AncState(N) is simply an intersection of the right hand sides characterizing the children nodes. □

A.3 Lumpability

Proof of lumpability

Under Assumption 2, we can show that the instantaneous transition rate from any allele in $g(b;\mathtt{N})$ to the set $g(b';\mathtt{N})$ is the same. Assuming the transition rate between lumped states $g(b;\mathtt{N})$ and $g(b';\mathtt{N})$ is nonzero, we show that there are two types of transitions. Either the only transition between the two lumped states is through an indel tract d observed at the leaf nodes; or all indel tracts from a set of target tracts are valid transitions between the lumped states.

Lemma 4. *Suppose Assumptions 1 and 2 and Approximation 1 hold. For node* N*, let* SG(N) *to be the singletons from the singleton-wildcards in* AncState(N)*. Consider any branch with child node* C*, and target statuses* $b, b' \in \{0,1\}^M$ *where the sets* $g(b;\mathtt{C})$ *and* $g(b';\mathtt{C})$ *are nonempty. For any alleles* $a, a' \in g(b;\mathtt{C})$*, we have*

$$\begin{aligned} q_{\text{lump}}\left(g(b;\mathtt{C}), g(b';\mathtt{C})\right) &= \lim_{\Delta\to 0} \frac{\Pr\left(X_{\mathtt{C}}(\Delta) \in g(b';\mathtt{C}) | X_{\mathtt{C}}(0) = a\right)}{\Delta} \\ &= \lim_{\Delta\to 0} \frac{\Pr\left(X_{\mathtt{C}}(\Delta) \in g(b';\mathtt{C}) | X_{\mathtt{C}}(9) = a'\right)}{\Delta}. \end{aligned} \tag{2.24}$$

If the only transition from an element in $g(b;\mathtt{C})$ *to* $g(b';\mathtt{C})$ *is via the unique indel* $d \in \text{SG}(\mathtt{C})$ *that deactivates the targets* $b' \setminus b$*, then*

$$q_{\text{lump}}\left(g(b;\mathtt{C}), g(b';\mathtt{C})\right) = h(\text{TT}(d), b)\Pr(d|\,\text{TT}(d))$$

where h *is defined in Assumption 2. Otherwise, we have*

$$q_{\text{lump}}\left(g(b;\mathtt{C}), g(b';\mathtt{C})\right) = \sum_{\tau:\exists d\in\tau=\text{TT}(d)\ s.t.\ \text{Apply}(d,a)\in g(b';\mathtt{N})} h(\tau, b).$$

Proof. The instantaneous transition rates for an allele $a \in g(b;\mathtt{C})$ to the set $g(b';\mathtt{C})$ is

$$\lim_{\Delta\to 0} \frac{\Pr\left(X(\Delta) \in g(b';\mathtt{C}) | X(0) = a\right)}{\Delta} = \sum_{a' \in g(b';\mathtt{C})} q(a,a') = \sum_{d:\mathrm{Apply}(d,a)\in g(b';\mathtt{N})} h(\tau,b)\Pr(d|\tau).$$

If d is an indel tract that can be introduced to the allele $a \in g(b;\mathtt{C})$ and $\mathrm{Apply}(a,d)$ has target status b', then we can introduce the same indel tract to any other allele $a' \in g(b;\mathtt{C})$ and $\mathrm{Apply}(a',d)$ will also have the target status b'. Therefore we have proven that (2.24) must hold for all $a, a' \in g(b;\mathtt{C})$.

To calculate the hazard rate between these lumped states, we rewrite the summation by grouping indel tracts with the same target tract:

$$q_{\mathrm{lump}}\left(g(b;\mathtt{C}), g(b';\mathtt{C})\right) = \sum_{\substack{\tau:\exists d\in\tau=\mathrm{TT}(d) \\ \text{s.t. } \mathrm{Apply}(d,a)\in g(b';\mathtt{N})}} \left\{ \sum_{d\in\tau:\mathrm{Apply}(d,a)\in g(b';\mathtt{N})} h(\tau,b)\Pr(d|\tau) \right\}. \quad (2.25)$$

One of the following two cases must be true:

1. From the decomposition (2.23) of $\mathrm{AncState}(\mathtt{C})$, there is only one indel tract d in the sets $D_{\mathtt{C},m}$ for $m = 1, ..., M_{\mathtt{C}}$ such that $\mathrm{Apply}(a,d) \in g(b',\mathtt{C})$ for all $a \in g(b,\mathtt{C})$. d cannot be from a wildcard or the inner wildcard of a singleton-wildcard since this would contradict the fact that d is the only indel tract in $\{D_{\mathtt{C},m}\}$ such that $\mathrm{Apply}(a,d) \in g(b',\mathtt{C})$ for all $a \in g(b,\mathtt{C})$. Therefore d must be the singleton for some singleton-wildcard $D_{\mathtt{C},m}$. In other words, the only possible transition from $g(b;\mathtt{C})$ to $g(b';\mathtt{C})$ is via the indel tract d.

2. Otherwise, for some target tract τ, there are at least two indel tracts in $d, d' \in \tau$ in the sets $D_{\mathtt{C},m}$ for $m = 1, ..., M_{\mathtt{C}}$ that deactivate targets $b' \backslash b$ such that $\mathrm{Apply}(a,d) \in g(b',\mathtt{C})$ and $\mathrm{Apply}(a,d') \in g(b',\mathtt{C})$ for all $a \in g(b,\mathtt{C})$. In this case, d and d' must be from a wildcard or the inner wildcard of a singleton-wildcard (d and d' cannot both be from

a singleton of a singleton-wildcard since $d \neq d'$). Therefore every indel tract d in τ satisfies $\text{Apply}(a, d) \in g(b', \mathtt{C})$ for all $a \in g(b, \mathtt{C})$.

Therefore (2.25) simplifies to

$$q_{\text{lump}}\left(g(b;\mathtt{C}), g(b';\mathtt{C})\right) = \begin{cases} h(\tau, b)\Pr(d|\tau) & \text{if case (1)} \\ \sum_{\tau:\exists d\in\tau=\text{TT}(d) \text{ s.t. } \text{Apply}(d,a)\in g(b';\mathtt{N})} h(\tau, b) & \text{if case (2).} \end{cases}$$

Note that to construct the entire instantaneous transition rate matrix of the aggregated process, we can easily calculate the total transition rate away from a target status and then calculate the transition rate to sink state $g(\mathsf{other};\mathtt{C})$ using the fact that each row sums to zero. The transition rate away from $g(\mathsf{other};\mathtt{C})$ is zero. □

Proof for Theorem 1

Proof. For any internal node, we know that

$$p_{\mathtt{N}}(b) = \prod_{\mathtt{C}\in\text{children}(\mathtt{N})} \left\{ \sum_{\substack{b'\in\{0,1\}^M \\ g(b';\mathtt{C})\neq\emptyset}} p_{\mathtt{C}}(b') \Pr\left(X_{\mathtt{C}}(t_{\mathtt{C}}) \in g(b';\mathtt{C}) | X_{\mathtt{C}}(0) \in g(b;\mathtt{N})\right) \right\}.$$

(We do not need to sum over the partition $g(\mathsf{other};\mathtt{N})$ since it contributes zero probability.) By irreversibility of the mutation process, $g(b;\mathtt{N}) \subseteq g(b;\mathtt{C})$ if $\mathtt{C}$ is a child of $\mathtt{N}$. By (2.24) in Lemma 4,

$$\Pr\left(\mathtt{C}(t_{\mathtt{C}}) \in g(b';\mathtt{C}) |\, \mathtt{C}(0) \in g(b;\mathtt{N})\right) = \Pr\left(\mathtt{C}(t_{\mathtt{C}}) \in g(b';\mathtt{C}) |\, \mathtt{C}(0) \in g(b;\mathtt{C})\right),$$

which means (2.15) also holds for node $\mathtt{N}$. □

A.4 Tuning penalty parameters

By varying the value of the penalty parameters κ_1 and κ_2, we can control the trade-off between minimizing the penalty versus maximizing the log likelihood. Choosing appropriate values is crucial for estimation accuracy. A common approach for tuning penalty parameters is to use cross-validation [**?**]; we use this procedure whenever possible. Note that we keep the tree topology fixed when tuning the penalty parameter.

We can perform cross-validation when there are multiple barcodes. First we partition the barcodes into training and valdation sets T and V, respectively. Next we fit tree and mutation parameters $\hat{\ell}_{\boldsymbol{\kappa}}$ and $\hat{\theta}_{\boldsymbol{\kappa}}$, respectively, for each $\boldsymbol{\kappa}$ using only the training data. We choose the $\boldsymbol{\kappa}$ with the highest validation log likelihood

$$\frac{1}{|V|}\sum_{i\in V}\log \Pr\left(X_{\mathtt{L}}^{(i)}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in \text{Leaves}(\mathbb{T}); \hat{\ell}_{\boldsymbol{\kappa}}, \hat{\theta}_{\boldsymbol{\kappa}}\right).$$

For our simulation studies with two and four barcodes, we used half of the barcodes for the validation set and half for training.

Unfortunately cross-validation cannot be utilized when there is a single barcode since we cannot split the dataset by barcodes. Instead we propose a variant of cross-validation described in Algorithm 5. The main differences are that we partition the leaves instead of the barcodes into training and validation sets S and S^c, respectively; and we select the best penalty parameter that maximizes the conditional probability of the observed alleles at S^c given the observed alleles at S.

To partition the leaves, we randomly select a subset of leaf children of each multifurcating node to put in the validation set S^c. We partition leaves in this manner, rather than simply dividing the leaves in half, because we must be able to evaluate (or closely approximate) (2.27) at the end of Algorithm 5 using the fitted branch length and mutation parameters. That is, we must be able to regraft the leaves in the set S^c onto the fitted tree. Regrafting is easy for the leaves in our specially-constructed set: The parent node of each leaf in S^c must

be located somewhere along the caterpillar spine corresponding to its original multifurcating parent. In our implementation, we chose to regraft the leaves to the midpoints of their corresponding caterpillar spines. The regrafting procedure is illustrated in Figure 2.17. Note that we do not tune the branch lengths of these validation leaves since it amounts to peeking at the validation data. In our simulations, we found that when tuning the branch lengths to maximize the unpenalized (or penalized) log likelihood, we nearly almost always choose the smallest penalty parameter since it prioritizes maximizing the likelihood and, therefore, (2.27).

To assess each candidate penalty parameter $\boldsymbol{\kappa}$, we compare the conditional probability of the observed alleles at S^c given the observed alleles at S. Our motivation is similar to that in cross-validation: If the alleles are observed from the tree with branch and mutation parameters ℓ^* and θ^*, we know that

$$\begin{aligned} &E\left[\log \Pr(X_{\mathtt{L}}(t_{\mathtt{L}}) \forall\, \mathtt{L} \in S^c \mid X_{\mathtt{L}}(t_{\mathtt{L}} \forall\, \mathtt{L} \in S); \ell^*, \theta^*); \ell^*, \theta^*\right] \\ &\geq E\left[\log \Pr(X_{\mathtt{L}}(t_{\mathtt{L}}) \forall\, \mathtt{L} \in S^c \mid X_{\mathtt{L}}(t_{\mathtt{L}} \forall\, \mathtt{L} \in S); \ell, \theta); \ell^*, \theta^*\right] \qquad \forall \ell, \theta \end{aligned} \tag{2.26}$$

by Jensen's inequality. (Note that this conditional probability is high only for if we have good estimates of both the mutation parameters and branch lengths of leaves S^c. It is not sufficient to only have an accurate estimate of the mutation parameters.) Recall cross-validation is motivated by a similar inequality but uses $\Pr(X; \ell, \theta)$ rather than a conditional probability.

From a theoretical standpoint, using (2.26) to select penalty parameters makes the most sense if we have an unbiased estimate of the expected conditional probability. Unfortunately, in our setting, the conditional probability in (2.27) is actually a biased estimate since the fitted parameters depended on the observed alleles at leaves S. Nonetheless, in simulations (where the truth is known), this biased estimate seemed to work well, as the selected penalty parameter was typically close to the best penalty parameter.

To perform the entire GAPML estimation procedure, we must tune the penalty parame-

Algorithm 5 Cross validation for a single barcode

1: Initialize S to be all the leaves in tree $\mathbb{T}$. Throughout, let S^c denote all the leaves in $\mathbb{T}$ not in S.
2: **for** each multifurcating node $\mathtt{N}$ where at least one children is a leaf **do**
3: Let $\mathtt{C}_1, ..., \mathtt{C}_m$ be the children nodes of $\mathtt{N}$ that are leaves. Randomly select $m' \geq 1$ of them and remove these from S.
4: **end for**
5: Let $\mathbb{T}_S$ be the subtree over the leaves S.
6: **for** each candidate penalty parameter $\boldsymbol{\kappa}$ **do**
7: Maximize the penalized log likelihood of the tree $\mathbb{T}_S$ with respect to its branch lengths $\boldsymbol{\ell}$ and mutation parameters $\boldsymbol{\theta}$

$$\hat{\boldsymbol{\ell}}_{\boldsymbol{\kappa}}, \hat{\boldsymbol{\theta}}_{\boldsymbol{\kappa}} = \arg\max_{\boldsymbol{\ell}, \boldsymbol{\theta}} \log \Pr\left(X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in S; \boldsymbol{\ell}, \boldsymbol{\theta}\right) - \mathrm{Pen}_{\boldsymbol{\kappa}}\left(\boldsymbol{\ell}, \boldsymbol{\theta}\right).$$

8: **end for**
9: Return the penalty parameter that maximizes the conditional probability:

$$\hat{\boldsymbol{\kappa}} = \arg\max_{\boldsymbol{\kappa}} \Pr\left(X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in S^c \mid X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in S; \hat{\boldsymbol{\ell}}_{\boldsymbol{\kappa}}, \hat{\boldsymbol{\theta}}_{\boldsymbol{\kappa}}\right). \tag{2.27}$$

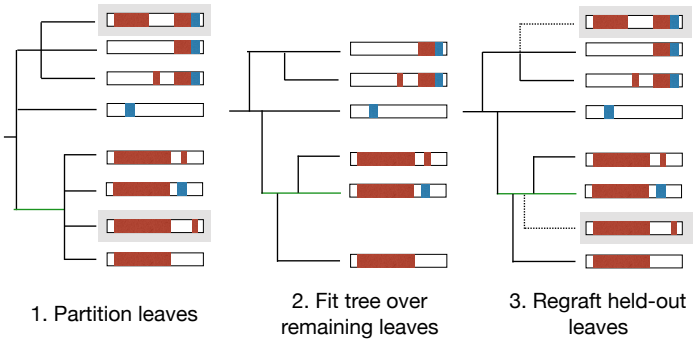


Figure 2.17: Cross-validation to tune penalty parameters with only one barcode. We split leaves into training and validation sets S and S^c, respectively as follows (left): For each multifurcating node, randomly select a subset of its children that are leaves to put in the "validation" set, denoted by the gray boxes. Fit branch lengths and mutation parameters on the subtree over the remaining leaves (middle). Regraft the leaves in the "validation" set back onto the fitted tree (right).

ters and the topology of the tree. Our full algorithm alternates between tuning the penalty parameters for a fixed tree topology and running a single iteration of Algorithm 4 for a fixed penalty parameter. After the penalty parameters are stable, we keep them fixed and only run Algorithm 4.

A.5 Specific model implementation

First, let us define short and long deletions. The deletion to the left in indel tract $d = \text{IT}[p_0, p_1, s, j_0, j_1]$ is *short* if target $j_0 - 1$ is unaffected or *long* if target $j_0 - 1$ is deactivated, i.e.

$$d \text{ has a short left deletion if } p_0 \in \text{pos}(j_0) \tag{2.28}$$

$$d \text{ has a long left deletion if } p_0 \in \text{pos}(j_0 - 1). \tag{2.29}$$

We define short and long deletions to the right similarly.

Rate parameterization

To parameterize the rate at which a target tract $\tau = \text{TT}[j_0', j_0, j_1, j_1']$ is introduced, we decompose the rate into a rate h_0 that represents the rate at which the targets j_0 and j_1 are cut and various scaling factors that control how often deletions are short or long (recall the definition in (2.28) and (2.29)):

$$h\left(\tau, \text{TargStat}(a)\right) = h_0\left(j_0, j_1, \text{TargStat}(a)\right) \prod_{i=0}^{1} \left[\gamma_i \mathbb{1}\{j_i \neq j_i'\} + \mathbb{1}\{j_i = j_i'\}\right],$$

where γ_0 and γ_1 parameterize how often long deletions occur to the left and right, respectively.

We specify h_0 using the assumption that the cutting time for target j follows an expo-

nential distribution with rate of $\lambda_j > 0$. For focal target cuts where $j_0 = j_1$, we define

$$h_0\left(j_0, j_0, \text{TargStat}(a)\right) = \lambda_{j_0} \mathbb{1}\{\text{TargStat}(j_0, a) = 0\}.$$

For double cuts at targets j_0 and j_1, we suppose the cut time follows an exponential distribution with rate $\omega \cdot (\lambda_{j_0} + \lambda_{j_1})$, where ω is an additional model parameter that we estimate and does not depend on the targets. Our parameterization of the double-cut rate is based on i) the assumption that an inter-target deletion is introduced when the cuts at both targets occur within a small time window of length ϵ and ii) approximating the probability of such an event as follows. The probability that random cut times X_{j_0} and X_{j_1} for targets j_0 and j_1 occur within a small window ϵ is approximately

$$p\left(|X_{j_0} - X_{j_1}| \leq \epsilon, \frac{X_{j_0} + X_{j_1}}{2} = t\right) \tag{2.30}$$

$$\approx \Pr(|X_{j_0} - X_{j_1}| \leq \epsilon)\, p(X_{j_0} = X_{j_1} = t | X_{j_0} = X_{j_1}) \tag{2.31}$$

$$\approx \omega(\lambda_{j_0} + \lambda_{j_1}). \tag{2.32}$$

Our approximation for the first line using the second improves as $\epsilon \to 0$. Next, the first component in (2.31) approaches zero as $\epsilon \to 0$ and does not vary much for different values of $\lambda_{j_0}, \lambda_{j_1}$ if ϵ is sufficiently small. Thus we approximate it using the same value of ω for all targets. The second component in (2.31) corresponds to an exponential distribution with the rate $\lambda_{j_0} + \lambda_{j_1}$.

We can interpret ω in two ways. First, ω controls how often a double cut is introduced. In an unmodified barcode, the relative rate that a double cut is introduced versus a single cut is $\omega \sum_{j_1 < j_2} (\lambda_{j_1} + \lambda_{j_2})$ versus $\sum_{j=1}^{M} \lambda_j$. The second interpretation, based on (2.32), is that ω serves as a proxy for ϵ: Larger ω indicates that an inter-target deletion can be introduced by two cuts spaced farther apart in time.

Deletion/insertion sequence parameterization

The second major component of the GESTALT mutation model specifies the conditional probability of introducing a particular indel tract given target tract $\tau = \mathrm{TT}[j_0', j_0, j_1, j_1']$. An indel tract can be represented by its deletion lengths to the left and right and the insertion sequence. We will suppose that the probability of a single insertion sequence is uniform over all possible nucleotide sequences of that length.

Let X_0, X_1, X_2 be the random variables (RVs) that parameterize the lengths of the left deletion, right deletion, and insertion, respectively. Let $x_{\tau,\min,i}$ and $x_{\tau,\max,i}$ for $i = 0$ and 1 specify the minimum and maximum deletion lengths to the left and right, respectively, for target tract τ. (For example, if $j_0 = j_0'$, the left deletions must be short so $x_{\tau,\min,i} = 0$ and $x_{\tau,\max,i}$ is the longest deletion without deactivating target $j_0 - 1$. As another example, if $j_0 = j_0' + 1$, the left deletion is long so $x_{\tau,\min,i}$ is the minimum deletion length to deactivate target j_0' and $x_{\tau,\max,i}$ is the longest length without deactivating target $j_0' - 1$.) For insertions, $x_{\tau,\min,2} = 0$ and $x_{\tau,\max,i} = \infty$ regardless of the target tract.

In our implementation, X_0 and X_1 are zero-inflated truncated negative binomial RVs for short trims and zero-inflated truncated poisson RVs for long trims. X_2 is a zero-inflated negative binomial RV. Let $\mathrm{NB}(m, q)$ denote the negative binomial distribution, which is the distribution for the number of successes until m failures are observed and q is the probability of success. Let $\mathrm{Pois}(r)$ be a poisson distribution with mean r. For $i = 0, 1$, define $\Pr(X_i = x_i|\tau)$ as

$$\begin{cases} p_{i,\mathbb{1}\{j_0=j_1\}} & \text{if } x_i = 0 \\ (1 - p_{i,\mathbb{1}\{j_0=j_1\}}) \Pr\left(X_i = x_i - x_{\tau,\min,i} - 1 | X_i < x_{\tau,\max,i} - 1; \mathrm{NB}(m_i, q_i)\right) & \text{if } x_i > 0, j_0' = j_0 \\ (1 - p_{i,\mathbb{1}\{j_0=j_1\}}) \Pr\left(X_i = x_i - x_{\tau,\min,i} - 1 | X_i < x_{\tau,\max,i} - 1; \mathrm{Pois}(r_i)\right) & \text{if } x_i > 0, j_0' \neq j_0 \end{cases} \cdot \tag{2.33}$$

Here, $p_{0,1}$ and $p_{1,1}$ are the probabilities of deleting zero nucleotides to the left and right,

respectively, in single-target indels; Likewise, $p_{0,0}$ and $p_{1,0}$ are the probabilities of deleting zero nucleotides to the left and right, respectively, in inter-target indels. The distribution of the insertion length is defined as

$$\Pr(X_2 = x_2|\tau) = \begin{cases} p_2 & \text{if } x_2 = 0 \\ (1-p_2)\Pr\left(X_2 = x_2 - 1; \mathrm{NB}(m_2, q_2)\right) & \text{if } x_2 > 0 \end{cases}, \tag{2.34}$$

where p_2 is the probability of inserting zero nucleotides.

Using the building blocks from above, we model the conditional probability of indel d with left deletion, right deletion, and insertion lengths x_0, x_1, x_2 given target tract τ as

$$\begin{aligned} &\Pr\left(X_0 = x_0, X_1 = x_1, X_2 = x_2 \mid \tau\right) \\ &= \begin{cases} \frac{\Pr(X_0=x_0|\tau)\Pr(X_1=x_1|\tau)\Pr(X_2=x_2|\tau)}{\Pr(X_0+X_1+X_2>0|\tau)} & \text{if } j_0' = j_0 = j_1 = j_1' \\ \Pr(X_0 = x_0|\tau)\Pr(X_1 = x_1|\tau)\Pr(X_2 = x_2|\tau) & \text{otherwise} \end{cases}. \end{aligned} \tag{2.35}$$

The case where $j_0' = j_0 = j_1 = j_1'$ corresponds to single-target indels with short left and right trims. We needed to separate out this special case because we only observe indels in this category if there is some deletion or insertion, i.e. $X_0 + X_1 + X_2 \neq 0$.

We chose to model the deletion/insertion lengths using these distributions primarily for their simplicity. In practice, the choice to use this model should depend on the probability that the same indel will be introduced independently in parallel lineages, also known as homoplasy. When homoplasy is rare, the tree estimate (topology and branch lengths) is not very sensitive to the model used for deletion/insertion lengths, because Assumption 2 assumes the mutation rates factorize into the target cut rates and deletion/insertion probabilities. However, if homoplasy is likely to occur, we suggest modeling the insertion and deletion process more carefully.

A.6 Data pre-processing

The data in this chapter are all from **?**] and are available at the Gene Expression Omnibus under GSE81713. We use the aligned data where each allele was described with the observed insertion/deletions (indels) at each target. Each CRISPR target can only be modified once and indels can only be introduced via a double-stranded break at a target cut site, so we further processed the aligned data accordingly: merging indels if there were more than one associated with a given target, and extending the deletion lengths and insertion sequences so that a target cut site was nested within each indel. We assume that the processed data is correct and do not attempt to model the effects of alignment error.

A.7 Code implementation

The code is implemented in Python using Tensorflow [**?**]. We maximize the penalized log likelihood using Adam [**?**]. We chose to use an automatic differentiation software since it has enables us to quickly iterate on the specific GESTALT model, without needing to recode the gradients. Just as automatic differentiation has greatly accelerated deep learning research [**?**], we believe that it can also accelerate the development of maximum likelihood estimation methods in phylogenetics.

Calculating the likelihood and its gradient can therefore become memory-intensive when there are many branches and/or barcodes. For large trees, we reduce memory usage by adding one more approximation. In particular, we only sum over states in AncState(·) where at most one masked indel tract occured along the branch. If there are still over 20 states, we only sum over states where no masked indel tracts occur along that branch. This is reasonable since it is unlikely for many hidden events to occur.

A.8 Comparison Methods

We use PHYLIP version 3.697 [**?**], the neighbor-joining algorithm in Bio.Phylo (Biopython version 1.72) [**?**], and the chronos function in R package `ape` version 5.2 [**?**].

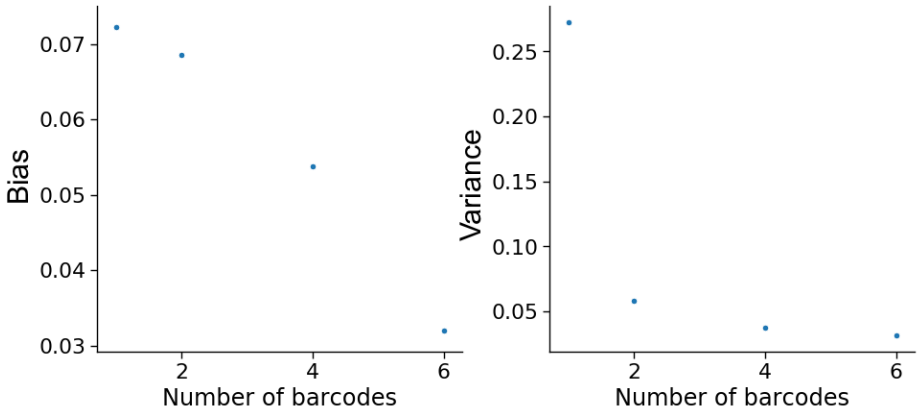


Figure 2.18: Plot of bias and variance of the mutation parameter estimates versus number of barcodes.

A.9 Simulation setup and additional results

For the results in Figure 2.10, the data was simulated with 5 synchronous cell division cycles followed by a birth-death process where the birth rate decayed at a rate of $\exp(-18t)$. The barcode was composed of six targets with $\boldsymbol{\lambda} = 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6$. The weight ω was set to 0.06 so that 20% of the unique observed indel tracts were due to double cuts. We sampled 8% of the leaves so that the average number of unique observed alleles was around 100 leaves. We refer to this simulation setup as Simulation A. We ran 20 replicates of Simulation A. In addition, plots of the bias and variance of the mutation parameter estimates are given in Figure 2.18.

The results in Figure 2.1 are from a larger simulation, which we will refer to as Simulation B, that is closer to the data collected in **?**]. Since zebrafish undergo around 11 synchronous cell division cycles at the beginning of development, this larger simulation entailed 9 synchronous cell division cycles followed by a birth-death process. We simulated with a barcode composed of ten targets. The resulting tree had on average around 200 leaves. We ran GAPML for 8 topology tuning iterations; at each iteration, we consider at most 15 SPR moves. The displayed results are from 20 replicates.

For this larger simulation, we also compared the runtimes of the methods on a server with

an Intel Xeon 2x8 core processor at 3.20GHz and 256 GB RAM. Obtaining tree topologies from C-S parsimony and neighbor-joining runs on the order of minutes. Branch length estimation using `chronos` runs on the order of seconds. In contrast, GAPML required up to three hours. Though the runtime of our method is much longer, it is still reasonable compared to the amount of time spent on data collection, which includes waiting until the organism is a certain age.

Using our simulation engine, we compare two very simple barcode design ideas: a single barcode with many targets, recommended in **?**], or many identical barcodes. However we believe the latter is more effective since spreading the targets over separate barcodes tends to create more unique alleles. In particular, the inter-target deletions tend to be shorter, which means fewer existing mutations are deleted and fewer targets are deactivated. To test this idea, we compared to the two design options in a simulation setup where we iteratively increased the number of targets by six, i.e. add six targets to the existing barcode or add a new barcode with six targets. Here we observe all 1024 leaves of a full binary tree with 10 levels. All targets had the same single-cut rate. We calibrated the double-cut weight ω to be around 18% for both barcode designs – this slightly favors the single-barcode design since it would have a higher rate of double cuts *in vivo* compared to a multiple-barcode design. Nevertheless, we find in our simulations that splitting the targets over separate barcodes tends to result in a much larger number of unique alleles than using a single barcode (Figure 2.19). At 30 targets, the multiple-barcode design has roughly 200 more unique alleles on average than the single-barcode design. Another reason we prefer the multiple-barcode design is that our model and tree estimates improve as the number of independent and identical barcodes increases, as illustrated in Figure 2.10.

Next, to better understand our algorithm GAPML, we show in-depth simulation results from a single replicate (Figure 2.20). Here we use the settings from Simulation B. Starting from the initial tree topology, the algorithm tunes the branch lengths and mutation parameters to maximize the penalized likelihood. During the gradient descent algorithm, the BHV

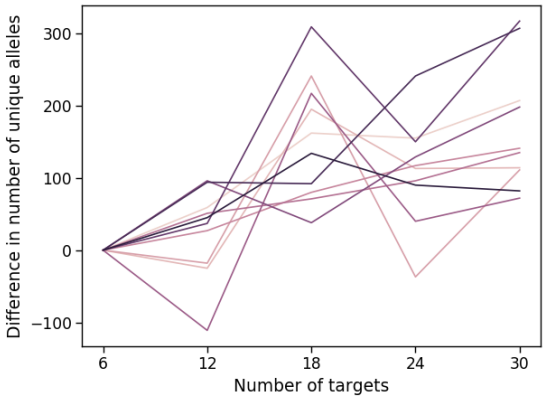
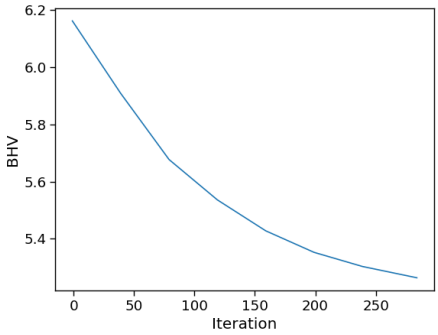


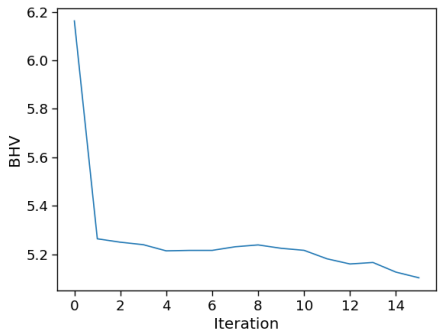
Figure 2.19: We compare the number of unique alleles obtained in GESTALT using a single barcode with many targets versus splitting the targets over multiple independent barcodes. The alleles are simulated on a full binary tree with 1024 leaves. Each line corresponds to a simulation where we iteratively add six targets, either by extending the single barcode or adding another barcode with six targets. A positive difference that the multiple-barcode design has more unique alleles, and vice versa.

distance of the tree estimate decreases (Figure 2.20a). In addition, we see that the BHV distance of the tree estimate decreases as Algorithm 4 iteratively performs SPR moves to update the tree topology.

Our method searches over the maximally parsimonious trees since they tend to have the highest penalized log likelihood. To justify this restricted search, we compared the penalized log likelihood for tree topology candidates of different parsimony scores, where the data was generated using Simulation A. To generate tree topologies with different parsimony scores, we started with the maximally parsimonious tree fit from Camin-Sokal and iteratively applied random SPR moves. For each of tree rearrangement, we fit a model by maximizing the penalized log likelihood. The penalty parameter is the same across all rearrangements. As seen in Figure 2.21, the most parsimonious trees have the highest penalized log likelihoods. Since our method aims to select a tree topology that maximizes the penalized log likelihood, it would not benefit from considering SPR moves that make the tree less parsi-



(a) Example of how the BHV distance changes as the branch lengths and mutation parameters are updated using gradient descent to maximize the penalized likelihood.



(b) Example of how the BHV distance changes at each SPR iteration, where we select the SPR with the highest likelihood with penalizaton over only the shared tree.

Figure 2.20: Examples of how the BHV distance changes as the algorithm proceeds for one simulation replicate from the ten-target setting.

monious; instead, considering these additional moves would make the method much slower.

Consistency analysis

Here we evaluate the consistency of our method through a simulation study. We consider barcodes with 6 targets and insert 16, 64, and 256 barcodes into the embryo cell. We simulated trees with roughly 110 leaves and sample 8% of the leaves. We then evaluated the BHV distance between the estimated tree and the true tree, as well as the mean squared error between the estimated and true mutation parameters as defined in Section A.5. As seen Figure 2.22, the tree and the mutation parameter estimates improve with the number of barcodes.

These simulation results complement the results in Section 4 of the main manuscript. There, we considered a small number of barcodes, which is more realistic with the current GESTALT technology. Here, we establish that adding more barcodes will continue to

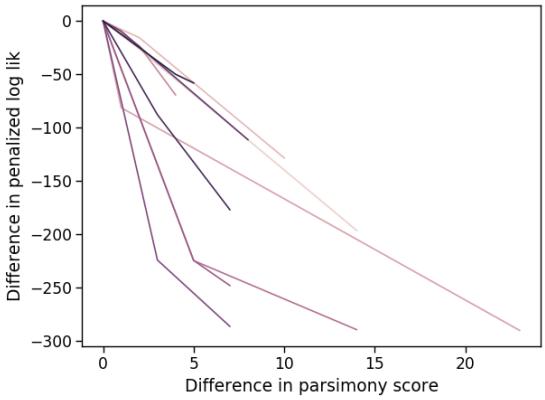


Figure 2.21: We compare the maximized penalized log likelihood of maximally parsimonious trees to less parsimonious trees. Each simulation replicate, represented by each line, shows four candidate tree topologies, starting from the most parsimonious one ($x = 0$) to increasingly less parsimonious ones (large differences in parsimony score). The y-value is the maximized penalized log likelihood of the candidate tree topology minus that of the maximally parsimonious tree.

improve estimates, even for much larger numbers of barcodes.

Sensitivity analysis

We now evaluate the sensitivity of our algorithm to errors when sequencing the mutated barcodes or ambiguities when resolving indels. Recall that the main manuscript assumed that all indels were sequenced and resolved accurately. This can be difficult to do in practice because there might be mixtures of insertions and deletions that cannot be resolved unambiguously.

If the primary goal is to recover tree topology and branch lengths, we find that the GAPML algorithm is relatively robust to errors when resolving indels. We simply require the method for resolving indels to map each unique indel d to the same indel d' from the same target tract (at least most of the time). This robustness property is a natural consequence of Assumption 2, which states that the mutation rate factors into the target cut rate and the insertion/deletion probabilities. As such, our ability to estimate the tree topology and

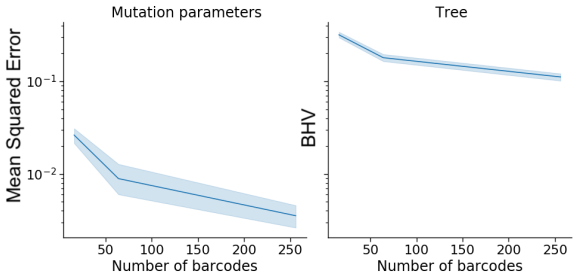


Figure 2.22: Mean squared error of estimated mutation parameters (left) and BHV distance between the estimated and true trees (right). Data is generated using 16, 64, and 256 barcodes. Plot reflects 20 simulation replicates.

branch lengths mostly depend on our ability to accurately estimate the target cut rates.

We illustrate the robustness of our algorithm in the following simulation study. For each unique indel, we introduced small perturbations to the resolved left deletion lengths, right deletion lengths, and the inserted sequence, each with some probability p. Using this mapping, each sampled allele is resolved as a collection of these incorrectly observed indels. We varied this error rate p from 0% to 20%. The barcode here is composed of six targets; four barcodes are inserted into the embryo. We simulated trees with 400 leaves on average and sample 5% of the leaves. As expected, the BHV distance and the internal node time correlation are relatively constant across the different error rates (Figure 2.23).

Finally, we note that this robustness property does not hold if the goal is to accurately estimate the insertion/deletion parameters (e.g. those defined in Section A.5). In that case, the indels need to be resolved correctly.

Zebrafish data analysis

For the zebrafish analyses, we estimated the tree over at most 400 randomly selected alleles (without replacement). 50% of the fish in this dataset had fewer than 400 alleles and the median number of unique alleles over the zebrafish datasets was 443. 25% of the fish in this dataset had more than 1000 alleles. We limit the number of alleles analyzed due to runtime restrictions.

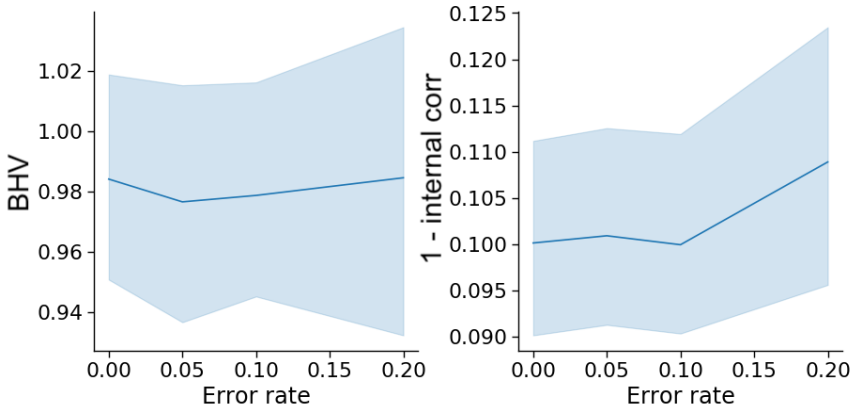


Figure 2.23: BHV distance and internal node time correlation between estimated and true trees when the indels are resolved with error rates 0%, 5%, 10%, and 20%. Plots reflect 50 simulation replicates.

To test if the fitted trees are recovering similar developmental relationships across fish rather than random noise, we ran a permutation test comparing the correlation between tissue distances from the estimated trees to that from randomly-estimated trees over randomly-shuffled data. More specifically, for a given tree topology, we randomly permute the observed alleles at the leaves. Each allele is associated with the number of times it is observed in each tissue type; we randomly shuffle these abundances over the possible tissue types within each allele. Finally, we randomly assign branch lengths along the tree by drawing samples from a uniform distribution and using the t-parameterization of **?**] to assign branch lengths. The correlation between tissue distances in these random trees is close to zero. All permutation tests were performed using 2000 replicates.

We also tested if the Pearson correlation between the number of tissue types/cell types and the internal node times is different from that of random trees. The random trees were generated using the same procedure as above.

We conclude by noting that the random trees are generated using the estimated tree topology from each method. Thus the null distributions are different and the p-values are not directly comparable between methods. Though this slightly complicates interpretation,

we prefer this approach since the estimated tree topology may naturally induce correlation between tissue distances. For most validation tests, the mean of the null distribution was similar across the different methods, and therefore the p-values are somewhat comparable. The major exception was the tests that checked recovery of cell-type and germ-layer restriction: here the mean of the null distribution were very different and we abstain from comparing p-values across methods.

For Figure 2.2, we bootstrapped fish replicates to estimate confidence intervals for the average correlation between estimated target cut rates.

	Adult fish #1	Adult fish #2	72 hpf #1	30 hpf #5	4.3 hpf #1
Target 1, λ_1	3.142	1.449	1.675	2.122	0.730
Target 2, λ_2	1.591	0.393	0.560	1.262	0.125
Target 3, λ_3	0.172	0.156	0.222	1.265	0.054
Target 4, λ_4	1.555	1.228	0.593	1.151	0.288
Target 5, λ_5	1.099	0.821	0.462	0.958	0.208
Target 6, λ_6	1.854	0.969	0.613	1.723	0.199
Target 7, λ_7	1.015	0.740	1.452	0.801	0.696
Target 8, λ_8	0.183	0.270	0.261	1.763	0.076
Target 9, λ_9	2.508	1.594	1.265	1.777	0.607
Target 10, λ_{10}	0.367	0.428	0.181	1.333	0.049
Long factor left γ_0	0.040	0.067	0.058	0.058	0.051
Long factor right γ_1	0.102	0.122	0.092	0.025	0.133
Double cut rate ω	0.047	0.062	0.045	0.049	0.041
Left short trim zero prob	0.048	0.093	0.081	0.490	0.053
Left short trim length mean	6.142	6.381	6.210	2.244	5.123
Left short trim length sd	4.533	4.657	4.525	3.625	4.013
Left long trim length mean	25.302	25.210	25.617	24.592	25.335
Left long trim length sd	1.183	1.190	1.136	1.150	1.180
Right short trim zero prob	0.541	0.530	0.520	0.169	0.530
Right short trim length mean	2.141	2.058	2.105	4.721	2.028
Right short trim length sd	3.401	3.286	3.351	3.831	3.242
Right long trim length mean	25.254	25.563	26.173	25.012	25.939
Right long trim length sd	1.323	1.247	0.987	1.359	1.107
Insertion zero prob	0.548	0.614	0.618	0.629	0.622
Insertion length mean	5.161	4.504	4.575	5.446	5.635
Insertion length sd	5.898	5.479	4.711	7.291	5.209

Table 2.5: Fitted parameters in the adult fish as well as some other fish embryos. The parameters above the line are related to target cut rates and the ones below the line are related to the nucleotide deletion and insertion process. Short versus long deletions are defined in (2.28) and (2.29), respectively.

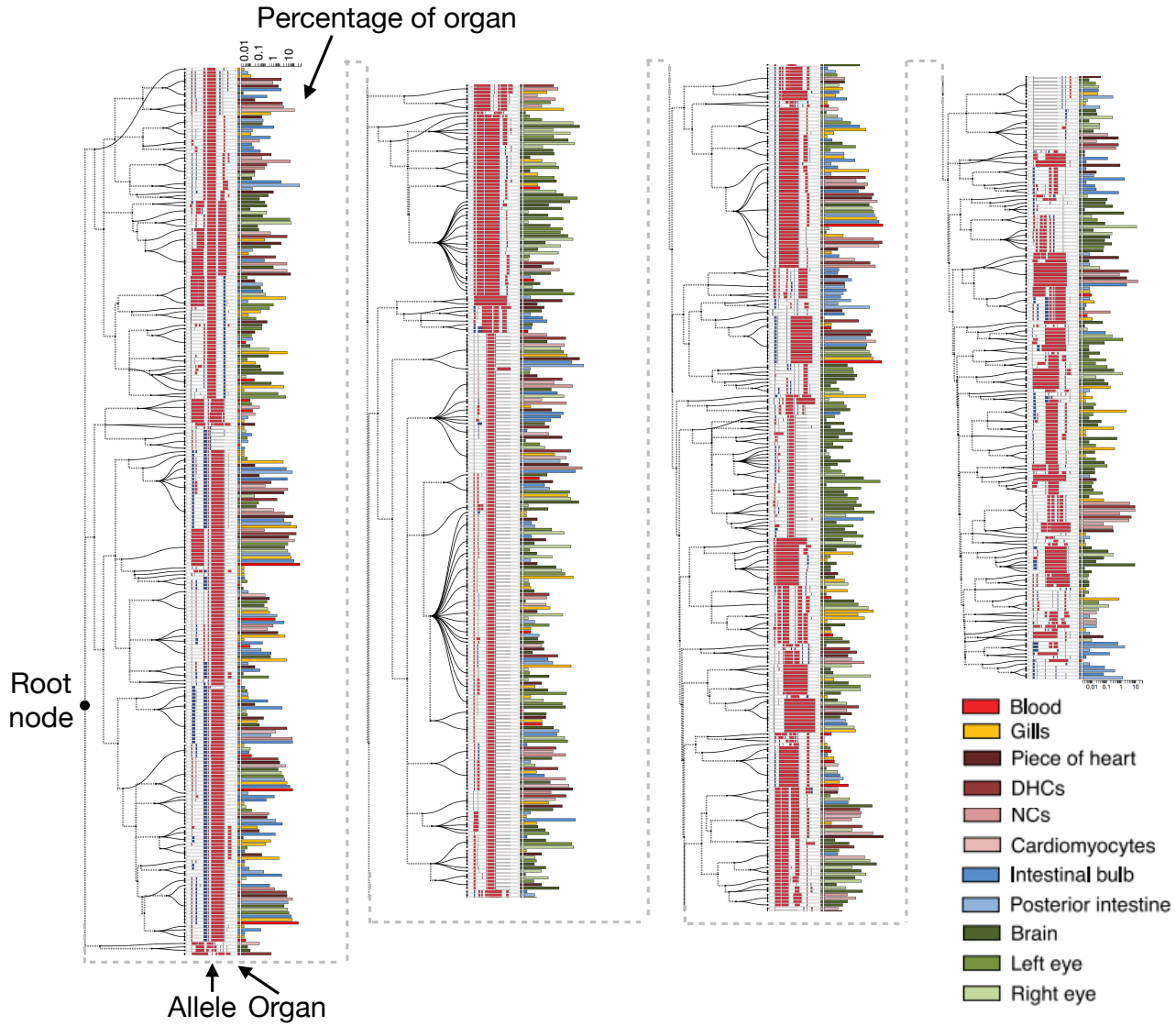


Figure 2.24: Estimated cell lineage tree for 400 randomly selected alleles from the first adult zebrafish using GAPML. Editing patterns in individual alleles are represented as shown previously. Alleles observed in multiple organs are plotted on separate lines per organ and are connected with stippled branches. Two sets of bars outside the alleles identify the organ in which the allele was observed and the proportion of cells in that organ represented by that allele (log10 scale). The dashed lines correspond to the caterpillar spines.

Part II

PREDICTION MODELS FOR MEDICINE AND HEALTHCARE

The second part of this thesis investigates the accuracy and reliability non-parametric machine learning algorithms and their use in precision medicine. In Chapter 3, we take the view of a model developer and study the efficacy of neural networks on datasets commonly found in biomedicine, which tend to have a large number of covariates, a limited number of observations, and a low signal-to-noise ratio. Neural networks are typically dismissed in this setting because they tend to overfit and are computationally expensive. This can be surprising, given that neural networks are also state-of-the-art in many complex problem domains like computer vision and natural language processing. To resolve this large gap in performance, we show that simple modifications to neural networks can make them a practical alternative to existing off-the-shelf machine learning algorithms. In Chapter 4, we then take the view of a regulatory agency that aims to ensure the safety and accuracy of machine learning algorithms used for medical decision making. Our goal is to design policies that can approve good modifications to these algorithms both efficiently and judiciously. Moreover, given the popularity of black-box prediction models, the policies must be agnostic to the specific estimation and modification procedure used.

Chapter 3

ENSEMBLED SPARSE-INPUT HIERARCHICAL NETWORKS FOR HIGH-DIMENSIONAL DATASETS

Summary

Neural networks have seen limited use in prediction for high-dimensional data with small sample sizes, because they tend to overfit and require tuning many more hyperparameters than existing off-the-shelf machine learning methods. With small modifications to the network architecture and training procedure, we show that dense neural networks can be a practical data analysis tool in these settings. The proposed method, **E**nsemble by **A**veraging **S**parse-**I**nput hi**ER**archical networks (EASIER-net), appropriately prunes the network structure by tuning only two L_1-penalty parameters, one that controls the input sparsity and another that controls the number of hidden layers and nodes. The method selects variables from the true support if the irrelevant covariates are only weakly correlated with the response; otherwise, it exhibits a grouping effect, where strongly correlated covariates are selected at similar rates. On a collection of real-world datasets with different sizes, EASIER-net selected network architectures in a data-adaptive manner and achieved higher prediction accuracy than off-the-shelf methods on average.

1 Introduction

Deep neural networks are highly modular and expressive nonparametric models that can learn complicated relationships between their inputs and outputs. Although they are state-of-the-art in many complex prediction problems where large amounts of data are available, their utility for analyzing datasets with few observations has been limited, particularly when the number of dimensions is high and the signal to noise ratio is low. There are two reasons why data analysts usually dismiss deep neural networks in these settings. First, they easily

overfit to the training data. Second, the data analyst needs to tune many hyperparameters, including the number of hidden layers, the number of hidden nodes per layer, regularization parameters, learning rates for the optimization algorithm, among others [4].

Nevertheless, the assumption that deep learning is too brittle or burdensome in these settings is worth revisiting. Many methods have been developed to reduce the generalization error of deep neural networks [13], such as the use of rectified linear units [12], stochastic gradient descent, regularization, and ensemble methods. In fact, Zhang et al. [53] has urged the research community to rethink generalization error for neural networks, since these models were able to both achieve low generalization error *and* memorize the training data. In addition, neural networks are easier than ever to implement and train. Many powerful automatic differentiation packages are freely available (e.g. `Tensorflow` [1] and `PyTorch` [36]) and computing power has grown exponentially in the past few decades.

Although most analyses of deep learning have been performed in large image datasets, there is now a growing body of evidence that neural networks can also be used for smaller sample sizes. In an analysis of over a hundred UCI classification datasets, Olson et al. [34] showed that ensemble neural networks with zero hyperparameter tuning do only slightly worse than random forests on average. An alternate approach pruned input weights using L_1 and L_2 penalties, resulting in sparse-input neural networks (SPINN), and outperformed random forests and the Lasso in certain cases [9]. Finally, Bayesian neural networks have been used for variable selection in high-dimensional datasets [32, 15, 25].

While these methods are promising, they are still insufficient. We found that the performance of network ensembles in Olson et al. [34] heavily depends on the dataset and can be quite poor without any network structure learning. SPINN requires tuning five or more hyperparameters, which is time consuming even with the help of modern hyperparameter optimization methods (see e.g., Snoek et al. [44]). Bayesian networks even more computationally expensive than their non-Bayesian counterparts.

Our goal is to design a procedure that is competitive with off-the-shelf machine learning

algorithms and requires only a "reasonable" amount of hyperparameter tuning. As many popular statistical procedures require tuning one or two hyperparameters (see e.g. Zou and Hastie [54] and Simon et al. [42]), we define tuning two hyperparameters as "reasonable." Training neural networks typically involves an outer search over hyperparameters such as the entire network structure and an inner training procedure for each candidate hyperparameter set. Here, we design the outer search to tune two *summary* features of the network structure that correspond to major sources of variation for generalization error: input sparsity and network size. The inner procedure is then responsible for both selecting the specific network structure and fitting model parameters.

We propose a method for training a single network, **S**parse-**I**nput hi**ER**archical network (SIER-net), and its ensemble version, **E**nsemble by **A**veraging **S**parse-**I**nput hi**ER**archical networks (EASIER-net). In SIER-net, the network is initialized with a fixed architecture with many hidden layers (deep), many hidden nodes per layer (wide), skip-connections from each layer to the output, and an input filter layer. Network structure learning is performed solely by tuning two L_1 penalty parameters, one to modulate the number of selected inputs and another for the number of hidden nodes and layers. Although many nodes and layers are included in the initial topology, these are perhaps better thought of as "candidate" nodes and layers: The combination of penalties and skip-connections allows the network to data-adaptively determine appropriate widths and depth (up to the maximum network size indicated).

By deriving probability bounds on the variable selection accuracy, we show that SIER-net is likely to select covariates in the true support that have high predictive value relative to the remaining covariates. On the other hand, if the covariates are highly correlated, we show that it is impossible for *any* procedure to accurately recover the true support. Since support recovery is unrealistic in these settings, we show that ensembling these networks using EASIER-net lets us quantify the uncertainty of the variable selection procedure. Moreover, in cases where correlated variables belong to a meaningful group (e.g. gene expression values

from the same pathway), EASIER-net exhibits a grouping effect, where covariates in the same group are selected at similar rates.

We evaluate EASIER-net on a collection of regression and classification tasks from the UCI Machine Learning Repository, which were chosen to represent a variety of dataset sizes. EASIER-net consistently achieves higher prediction accuracy than other neural network estimators. In addition, EASIER-net achieves higher prediction accuracy on average compared to off-the-shelf machine learning methods and is competitive in computation time.

The chapter is organized as follows. Section 2 discusses related work. In Section 3, we define SIER-net and analyze its variable screening accuracy. We then introduce our ensembled estimator and discuss the effects of ensembling using a Bayesian perspective. Finally, we present empirical analyses of EASIER-net on both simulated and real data in Section 4. Code for fitting EASIER-net is available at `https://github.com/jjfeng/easier_net`.

2 Related Work

There is a growing body of literature on applying neural networks to high-dimensional datasets with low sample sizes. Existing methods typically rely on some form of regularization to mitigate the neural network's tendency to overfit. Sparsity-inducing penalties, like the lasso and group lasso, can be applied to the network weights to encourage feature selection and drastically improve prediction accuracy [41, 50, 9]. A similar idea in the Bayesian literature is to apply a prior over the model weights, such as through an "Automatic Relevance Determination" (ARD) prior [29, 32, 33] or a Bernoulli prior for the probability that a weight is included [25]. An alternative idea inspired by the success of random forests is to regularize neural networks by ensembling [34]. Even though these methods are promising, neural networks still have limited utility in these settings. Many of these methods either require substantially more hyperparameter tuning or offer negligible gains in prediction accuracy compared to existing off-the-shelf machine learning methods. Here, we propose using three simple ideas — sparsity, skip-connections, and ensembling — to improve prediction

accuracy and decrease hyperparameter tuning requirements.

Applying sparsity-inducing penalties to perform network structure learning is a well-accepted practice [17, 24, 51, 16, 48]. In addition to lower computational and memory requirements, sparser networks often achieve lower generalization error [26, 50, 9]. However, these penalization-based methods typically require tuning many penalty parameters and/or network structure hyperparameters. In contrast, EASIER-net requires tuning only two L_1 penalty parameters and select an appropriate network structure for each candidate penalty parameter set.

Skip-connections are a classical technique for expanding the neural network model class to include simpler models like linear and logistic regression (e.g. Chapter 5 of Ripley and Hjort [40]). This technique often improves prediction accuracy on high-dimensional datasets with small sample sizes by leveraging the performance of simple parametric models. As such, Liang et al. [25] used skip-connections to combine Bayesian linear regression with Bayesian neural networks with a single hidden layer. Our method EASIER-net can fit even deeper networks and adaptively chooses the depth using a Lasso penalty. Although skip-connections have also been used to improve gradient propagation in ultra-deep neural networks [18, 20], our architecture is designed for shallower networks with five or so hidden layers, which tend to perform well on smaller datasets.

Ensembling has been shown to improve the generalization error [34] and uncertainty quantification of neural networks [23]. The benefit of ensembling has been explained from many perspectives, including its ability to reduce model variance [6] and its similarity to Bayesian posterior inference and model averaging [2, 27, 37, 49]. However, none of these works have discussed the interaction between variable selection procedures and ensembling. In this work, we show that ensembling sparse-input neural networks better reflects the uncertainty of the true support and induces a "grouping effect" for correlated predictors, a feature commonly associated with the elastic net [54].

3 Method

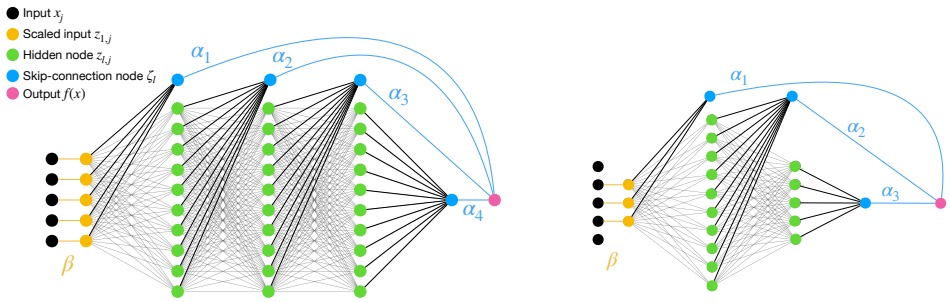
3.1 Notation

Consider the usual prediction setup with covariates $\boldsymbol{x} = (x_1, ..., x_d) \in \mathcal{X} \subseteq \mathbb{R}^d$ and response $y \in \mathcal{Y}$. $\mathcal{Y}$ may be real-valued in the case of regression and categorical in the case of classification. Suppose we observe n independent and identically distributed (IID) observations, denoted $(\boldsymbol{x}_i, y_i)$ for $i = 1, ..., n$.

The typical dense neural network with L layers is defined as follows. Let d_l be the number of nodes in layer l and $\boldsymbol{z}_l \in \mathbb{R}^{1 \times d_l}$ be the node values, where $l = 1$ denotes the input layer (i.e. $\boldsymbol{z}_1 = \boldsymbol{x}$) and $l = L$ denotes the output. For simplicity, we suppose all hidden nodes use the same nonlinear activation function $\phi : \mathbb{R} \mapsto \mathbb{R}$. Here, we set ϕ as the rectified linear unit $a \mapsto a_+$, though one could use alternative activation functions like the sigmoid and hyperbolic tangent. Each non-output layer l is associated with a weight matrix $W_l \in \mathbb{R}^{d_l \times d_{l+1}}$ and bias vector $b_l \in \mathbb{R}^{1 \times d_{l+1}}$ that is used to linearly combine its inputs. The specified activation function is then applied to create the input for the subsequent layer. For $l = 1, ..., L-2$, we define $z_{l+1} = \phi\left(z_l W_l + b_l\right)$ where ϕ is applied element-wise. The final model output is defined as $\boldsymbol{z}_L = \phi_{\text{out}}(\boldsymbol{z}_{L-1} W_{L-1} + \boldsymbol{b}_{L-1})$. For regression problems, $d_L = 1$ and $\phi_{\text{out}} : \mathbb{R}^{1 \times 1} \mapsto \mathbb{R}$ is typically the identity function; For classification problems, d_L is the number of classes and $\phi_{\text{out}} : \mathbb{R}^{1 \times d_L} \mapsto [0, 1]^{d_L}$ is typically the softmax function.

3.2 Sparse-input hierarchical networks

Our method, **S**parse-**I**nput hi**ER**archical networks (SIER-net), modifies a dense neural network by adding an input filter layer and skip-connections from each intermediate layer to the output (Figure 3.1). The structure of the network and the estimation procedure in SIER-net are specially designed to facilitate network structure learning in a data-adaptive manner: the network is first initialized per some maximum allowable network structure and then network nodes and layers are pruned by minimizing an L_1-penalized empirical loss. The network is parameterized by weights W and biases b in the base network, input



(a) Initial network structure (b) Fitted network

Figure 3.1: A sparse-input hierarchical network with three hidden layers. The network augments a dense neural network by adding i) an input filter layer (yellow edges) that scales the inputs by parameter β and ii) skip-connections from each layer to the output. The output is the weighted average of the blue nodes with weights $|\alpha_l|$ for $l = 1, .., 4$. The network is initialized with non-zero weights along all edges. During training, many of these edges are set to zero by lasso penalties in the objective, which may prune away inputs, nodes, and entire layers.

scaling factors $\beta \in \mathbb{R}^d$, skip-connection factors $\alpha \in \mathbb{R}^{L-1}$, as well as additional weights W' and biases b' associated with the skip-connections. We denote the network as $f_{W,b,\beta,\alpha}$ and provide the full set of equations defining its output in Section A.1 in the Appendix. For regression problems, training involves minimizing the following objective with loss function $\ell : \mathbb{R} \times \mathbb{R} \mapsto \mathbb{R}$:

$$\min_{W,b,\beta,\alpha} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f_{W,b,\beta,\alpha}(x_i), y_i\right) + \lambda_1 \left(\|\beta\|_1 + \|W_1'\|_1\right) + \lambda_2 \left(\sum_{l=1}^{L-2} \|W_l\|_1 + \sum_{l=2}^{L-1} \|W_l'\|_1\right) \quad (3.1)$$

where $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ are penalty parameters. For classification problems, the L_1 penalties are applied to the bias parameters as well to regularize the marginal probability estimates for each class. Broadly speaking, the first set of L_1 penalties scaled by λ_1 controls the input sparsity and the second set of L_1 penalties scaled by λ_2 controls the number of active layers and hidden nodes.

The input filter layer replaces the first layer with scaled inputs $z_{1,i} = \beta_i x_i$ for $i = 1, ..., d$, where $\boldsymbol{\beta}$ is a learnable parameter. Since the lasso penalty encourages parameters to be

exactly equal to zero, minimizing (3.1) will encourage the model to depend on a smaller set of covariates. So as λ_1 increases, the size of the fitted model's support will decrease.

To add skip-connections, we add d_L additional nodes per layer, denoted by $\boldsymbol{\zeta}_l \in \mathbb{R}^{1\times d_L}$, to layers $l = 1, \cdots, L-1$, where $\boldsymbol{\zeta}_l = \boldsymbol{z}_l W_l' + \boldsymbol{b}_l'$ for $W_l' \in \mathbb{R}^{d_l \times d_L}$ and $\boldsymbol{b}_l' \in \mathbb{R}^{1\times d_L}$. (Recall that d_L is the number of output nodes from the network, so $d_L = 1$ in the regression setting.) For each $l = 1, ..., L-1$, nodes $\boldsymbol{\zeta}_l$ are associated with scalar weight α_l. We then set the network's output as a function of the weighted average of these additional nodes, i.e.

$$f_{W,b,\beta,\alpha}(\boldsymbol{x}) := \boldsymbol{z}_L = \phi_{\text{out}}\left(\sum_{l=1}^{L-1} \frac{|\alpha_l|}{\sum_{l'=1}^{L-1}|\alpha_{l'}|}\boldsymbol{\zeta}_l\right). \tag{3.2}$$

Normalizing the skip-connection weights is useful in a few ways. First, normalizing the skip-connection weights serves as a form of regularization. Second, we can modulate the normalized skip-connection weights by varying λ_2. In particular, as λ_2 decreases, the number of active layers and nodes increases and, consequently, the model tends to assign larger skip-connection weights to layers closer to the output. Finally, the normalized weights can be roughly interpreted as the conditional importance of each layer. For a formal comparison, we evaluate the proportion of variance contributed by layer $l = 1, ..., L-1$ to the final network output, i.e.

$$\text{Var}\left(|\alpha_l|\zeta_l\right) / \text{Var}\left(\sum_{l'=1}^{L-1}|\alpha_{l'}|\zeta_{l'}\right). \tag{3.3}$$

With the aforementioned modifications, the neural network is naturally hierarchical, where simpler models can be represented with fewer nonzero weights. Consider a few special cases in the regression setting. If all network parameters are set to zero except for $\boldsymbol{\beta}$, α_1, W_1', and $\boldsymbol{b}_1'$, then the fitted network is a linear model. More generally, if all of the parameters associated with layers $l \geq L'$ are zero for some $L' < L$, then $f_{W,b,\beta,\alpha}$ is a neural network with only L' hidden layers. Finally, if many entries of $\boldsymbol{\beta}$ are zero, the fitted network is sparse with respect to the number of inputs. By design, the ordering in the hierarchy corresponds

to the axes defined by penalty parameters λ_1 and λ_2.

To minimize (3.1) (or find a local minimum), the simplest approach is to run the popular stochastic gradient-based optimization algorithm `Adam` [21]. However, `Adam` does not shrink network weights exactly to zero. So to obtain a network that is actually sparse, we recommend running a few more steps of (batch) proximal gradient descent [35] after `Adam` converges. Recall that proximal gradient descent is a form of projected gradient descent used to solve non-differentiable convex optimization problems; therefore, it should converge within a few iterations, as (3.1) should be locally quadratic after `Adam` converges. This two-step optimization procedure is shown in Algorithm 6. At each iteration of proximal gradient descent, we perform a batch gradient update with respect to the empirical loss, followed by the proximal operator on the two L_1 penalties. The proximal operator, in this case, is very efficient, since it is the soft-thresholding function

$$S_\lambda(\theta) = \begin{cases} \theta - \lambda & \text{if } \theta > \lambda \\ 0 & \text{if } |\theta| \leq \lambda \\ \theta + \lambda & \text{if } \theta < -\lambda \end{cases} . \tag{3.4}$$

Algorithm 6 SIER-net

Run Adam until convergence
for $k = 1, 2, 3...$ **do**
  $\eta^{(k+1)} := \eta^{(k)} - t_k \nabla_\eta \frac{1}{n} \sum_{i=1}^n \ell(f_\eta(x_i), y_i)$
  for each parameter η_i with an L_1 penalty with penalty parameter λ_i **do**
    $\eta_i^{(k+1)} := S_{\lambda_i t_k}(\eta_i^{(k+1)})$
  end for
end for

Variable screening

In this section, we derive probability bounds to understand the variable screening properties of SIER-net. We show that the expected number of false negatives is intimately tied to the

predictive value of each variable in the true support, relative to the remaining covariates. We formalize this idea by defining a predictive value function γ in Definition 3 and then show how it appears in the probability bounds on the number of false negatives. Our results show that SIER-net is effective at support screening when the predictive value of the true support is high, which is maximized when the covariates are independent. We then show that no estimation procedures can accurately perform support screening or recovery if the predictive value is low, which occurs when the covariates are highly correlated. Given this impossibility result, we will introduce an ensembling procedure in Section 3.3 that quantifies the uncertainty of the true support instead. The proofs for results in this section are given in Section A.2 of the Appendix.

We start by introducing some notation and definitions. Let $\mathcal{F}$ be a class of functions that map from $\mathcal{X}$ to $\mathcal{Y}$. For any function $f \in \mathcal{F}$, define its support $s(f)$ as the set $\tilde{s} \subseteq \{1, ..., d\}$ such that $f(x) = f(x')$ for all $x, x' \in \mathcal{X}$ where $x_i = x'_i$ for all $i \in \tilde{s}$ (If there are multiple possible supports, use a random rule to select one among those with the smallest cardinality). Also, for any set $\tilde{s} \subseteq \{1, ..., d\}$, let $\mathcal{F}_{\tilde{s}}$ be the set of functions $f \in \mathcal{F}$ with support $s(f) \subseteq \tilde{s}$ and let $|\tilde{s}|$ denote its cardinality. Suppose we observe IID observations (X_i, Y_i) for $i = 1, ..., n$ drawn from joint distribution P. Let its conditional mean be denoted $\mu_P(x) = \mathbb{E}_P[Y|X = x]$ and, for convenience, let the true support be denoted $s^*_P = s(\mu_P)$.

Let Θ_{L,B_0,B_1} be the set of all sparse-input hierarchical network parameters with up to L layers, infinity norm up to B_0, and total variation norm up to B_1. Here, we use the total variation norm of a neural network as defined in Barron and Klusowski [3], which scales with the size of the support and number of hidden nodes. Given a neural network estimator $f_{\hat{\theta}_n}$, our goal is to characterize the probability of missing at least m relevant covariates, i.e.

$$P\left(\left|s^*_P \setminus s(f_{\hat{\theta}_n})\right| \geq m\right). \tag{3.5}$$

For simplicity, we suppose the inputs are scaled such that $\mathcal{X} = [-1, 1]^d$, consider only the regression setting, and let ℓ be the squared error loss. Moreover, suppose that $\epsilon = Y - \mu(X)$

is a Gaussian random variable with variance σ^2 and the true regression function is a neural network, i.e. $\mu_P = f_{\theta^*}$ for some $\theta^* \in \Theta_{L,B_0,B_1}$. The results here can be generalized to sub-Gaussian noise and alternative loss functions.

We quantify the predictive value of individual variables and groups of variables using the following definition.

Definition 3. *For set $\tilde{s} \subseteq \{1, ..., d\}$, let $f^*_{\tilde{s}}$ be the population risk minimizer with support $\tilde{s}$, i.e.*

$$f^*_{\tilde{s}} = \underset{f \in \mathcal{F}_{\tilde{s}}}{\arg\min}\, \mathbb{E}_P\, \ell(f(X), Y). \tag{3.6}$$

For integers $m = 1, ..., d$, denote the minimum achievable excess risk when omitting m variables from the true support as

$$\gamma(m; P) = \min_{\tilde{s} \subseteq \{1,...,d\} : |s^*_P \setminus \tilde{s}| = m} \mathbb{E}_P \left\{ \ell(f^*_{\tilde{s}}(X), Y) - \ell(\mu_P(X), Y) \right\}. \tag{3.7}$$

Specifically, a large value for $\gamma(m; P)$ means that any model that fails to include m variables from the true support will have significantly worse prediction accuracy. Put another way, $\gamma(m; P)$ is large if the variables outside the true support are not highly predictive of the response Y. Next, we show that our ability to perform variable screening is fundamentally tied to γ.

We derive the following upper bound of (3.5) by combining our definition of γ with standard techniques for bounding the excess risk of the constrained empirical risk minimizer. Although the previous section described fitting the network by minimizing the penalized empirical risk in (3.1), we analyze the constrained minimizer here since we can directly apply results from Barron and Klusowski [3]. Moreover, by using the method of Lagrangian multipliers, the constrained problem can be transformed to a similar structure as the penalized form. For appropriately chosen penalty parameters, the two estimation procedures

can give similar estimates and, in fact, are equivalent in convex settings.

Theorem 2. *Given n observations, let $\hat{\theta}_n$ be a global minimizer of the empirical risk*

$$\hat{\theta}_n \in \underset{\theta \in \Theta_{L,B_0,B_1}}{\arg\min} \frac{1}{n} \sum_{i=1}^{n} \ell\left(f_\theta(x_i), y_i\right). \tag{3.8}$$

Then for any $m = 1, \ldots, d$ and $\delta \geq 0$, we have

$$P\left(\left|s_P^* \setminus s(f_{\hat{\theta}_n})\right| \geq m\right) \leq \mathbb{1}\left\{B_1(4B_0 + \sigma)L\sqrt{\frac{2(L\log(2) + \log(2d))}{n}} + 2\delta \geq \gamma(m; P)\right\} + \exp\left(-\frac{n\delta^2}{2B_0^4}\right) + 2\exp\left(-\frac{n\delta^2}{8B_0^2\sigma^2}\right). \tag{3.9}$$

Using asymptotic notation, the upper bound in (3.9) is nontrivial (i.e. smaller than one) only if the number of observations $n \gtrsim 1/\gamma^2(m)$. Thus, the upper bound is small when the predictive value of the true support relative to the remaining covariates is high. These conditions are nonparametric generalizations of those used for support screening and variable selection in linear models, i.e. incoherence and beta-min conditions and variants thereof [7, 47].

In addition, Theorem 2 reveals how variable screening performance and prediction accuracy depend on the model capacity. In particular, we see that the upper bound grows quickly with respect to L, B_0, and B_1. So, one should therefore constrain the network structure search to networks with depth, infinity norm, and total variation norm no more than that of the true model. Indeed, we designed SIER-net based on this theoretical result: the second set of L_1 penalties in (3.1) helps us control network depth; the sum of all the penalties controls the infinity norm; and because the first set of penalties controls the input sparsity and the second controls the number of hidden nodes, their sum controls the total variation norm.

Next, we characterize how the best achievable performance by any support screening

procedure depends on γ. Consider any non-negative monotonically non-decreasing function γ. For any $k \in \{1, ..., d\}$, let $\mathcal{P}_{k,\gamma}$ be the set of distributions P with support size up to k where the predictive value of the true support is lower bounded by γ, i.e.

$$\mathcal{P}_{k,\gamma} := \{P : |s(\mu_P)| \leq k, \gamma(m; P) \geq \gamma(m) \ \forall m = 1, ..., k\}.$$

Thus our goal is to lower bound the minimax probability

$$\inf_{\hat{f}_n : |s(\hat{f}_n)| \leq k} \sup_{P \in \mathcal{P}_{k,\gamma}} P\left(\left|s(\mu_P) \setminus s(\hat{f}_n)\right| \geq m\right) \tag{3.10}$$

for $m = 1, ..., \min(k, \lfloor d/2 \rfloor)$.[1] For this minimax probability to be meaningful, note that we have removed trivial estimation procedures that select an excessive number of predictors. Instead, (3.10) is restricted to estimators that select at most k predictors. The following lower bound is a straightforward application of Le Cam's method (see Wainwright [47]). The proof relates the difficulty of the estimation procedure to the distance between probability models, which in turn can be bounded by γ.

Theorem 3. *For any positive integer $m \leq \min(k, \lfloor d/2 \rfloor)$, we have that*

$$\inf_{\hat{f}_n : |s(\hat{f}_n)| \leq k} \sup_{P \in \mathcal{P}_{k,\gamma}} P\left(\left|s(\mu_P) \setminus s(\hat{f}_n)\right| \geq m\right) \geq \frac{1}{2}\left(1 - \sqrt{\frac{n\sigma^2}{2}\gamma(2m)}\right). \tag{3.11}$$

Assuming γ is strictly positive, the minimax probability is smaller than $q \in (0, 1/2)$ only if the number of observations $n \gtrsim (1/2 - q)^2/\gamma(2m)$.

From Theorems 2 and 3, we see that the upper and lower bounds on the probability of having m false positives both depend on γ. For this probability to be small, the upper bound states that $n \gtrsim 1/\gamma^2(m)$ while the lower bound states that $n \gtrsim 1/\gamma(2m)$. Admit-

[1] We ignore the case where $m > \lfloor d/2 \rfloor$ since this probability is only relevant if $k > \lfloor d/2 \rfloor$. However, the probability must be zero in this case by the pigeonhole principle.

tedly, the bounds differ by a polynomial factor, which we believe could be tightened with additional assumptions about the neural network estimation procedure. Nevertheless, the results clearly show that our ability to perform support screening is fundamentally connected to the predictive value of the relevant covariates relative to the irrelevant ones.

Although the proof techniques used in this section are not novel, this connection between support screening and γ has important practical implications that, to our knowledge, have not been sufficiently emphasized in the literature. In particular, we have established that good support recovery is an unrealistic goal if covariates in the true support can be predicted accurately using covariates outside the support. This issue cannot be ignored since covariates are likely to be correlated in high-dimensional data. We will address this issue in the following section by constructing an *ensemble* of sparse-input hierarchical networks.

3.3 Ensemble by averaging sparse-input hierarchical networks

We now propose ensemble by averaging sparse-input hierarchical networks, which we refer to as EASIER-net. Ensembling addresses two issues with training a single sparse-input hierarchical network. First, as shown in Section 3.2, accurate variable screening is unrealistic when covariates are highly correlated because there is not enough information to distinguish relevant versus irrelevant variables. Instead, the fitted model should reflect the uncertainty of the true support. Second, in many high-dimensional datasets, strongly-correlated covariates actually belong to a group, such as gene expression values from the same pathway. In this case, we would like the variable selection method to select variables from the same group at similar rates. We outline the ensembling procedure below and engage a Bayesian perspective to help us understand how ensembling addresses these two issues.

In EASIER-net, the ensemble is composed of B independently-trained sparse-input hierarchical networks, denoted $\{\hat{f}_{(1)}, ..., \hat{f}_{(B)}\}$. We inject noise into the training procedure to reduce correlation across the fitted models and, thereby, the variance of the ensemble [6]. We follow nearly the same recipe as in Lakshminarayanan et al. [23]: for each network, we

randomly initialize the parameters and minimize (3.1) by running `Adam` with shuffled mini-batches and then batch proximal gradient descent. To predict the response for a given input x, we average the predictions across networks, i.e. $\hat{f}_{\text{ensemble}} = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_{(b)}(x)$. Note that in classification problems, this involves averaging the categorical distributions. Also, unlike some ensembling procedures, e.g. random forests, we *do not* bootstrap or subsample our data. In empirical experiments, we found that random network initializations and random mini-batch ordering without data re/sub-sampling achieved stronger performance.

Ensembling can be viewed as an approximation of Bayesian model averaging (BMA), a connection that has been noted in previous works [37, 49]. BMA places a prior over all possible model classes and averages over the posterior distributions to make a prediction [30]. Ensembling uses a similar approach: since stochastic gradient descent approximates the Bayesian posterior using a single mode [31], the aforementioned ensembling procedure randomly samples maximum a posteriori (MAP) estimates and averages their predictions. By sampling multiple modes, ensembling often produces better approximations of the posterior distribution than procedures that sample from a single mode [49, 37]. Since BMA is computationally intractable in many settings, we can instead use ensembling as a computationally efficient approximation.

We use this Bayesian perspective to understand why ensembling can help in settings where covariates are highly correlated. In variable selection problems, BMA quantifies the uncertainty of the true support via the posterior distribution [38, 19, 46]. Since EASIER-net is an approximate BMA procedure, the probability that a member in EASIER-net selects support $\tilde{s}$ is approximately equal to the posterior probability that the true support is $\tilde{s}$, i.e. for any $b = 1, .., B$, we have

$$\Pr\left(s(\hat{f}_{(b)}) = \tilde{s}; \{(x_i, y_i) : i = 1, \ldots, n\}\right) \approx P\left(s(f) = \tilde{s} \mid \{(x_i, y_i) : i = 1, \ldots, n\}\right). \quad (3.12)$$

Thus, the variety of the fitted supports in EASIER-net reflects the uncertainty of the variable selection procedure. Moreover, (3.12) implies that the selection rate of variable i in EASIER-

net is approximately the posterior probability of variable i belonging in the true support. So if all variables in a group are strongly correlated, their selection rates should be similar.

3.4 Hyperparameter tuning

We tune the two penalty parameters in EASIER-net using K-fold cross-validation and fix the remaining hyperparameters to their default values (see the Section A.3 in the Appendix for reasonable defaults). We recommend performing either a grid search or a random search [5] over candidate penalty parameter values since these procedures are easily parallelizable. For each candidate value, we apply EASIER-net to each of the K folds and obtain the average validation loss. We then apply EASIER-net on all of the data using the penalty parameters that minimize the average validation loss.

4 Empirical analyses

For all analyses, we initialize SIER-net with 5 hidden layers and 100 hidden nodes per layer. In all cases, EASIER-net was trained with $B = 20$ members, unless specified otherwise. To handle class imbalance issues in (multi-)classification problems, we weight the observation's loss by the inverse of the empirical frequency of its class. All continuous covariates and outcomes were centered and scaled to have mean zero and variance one. Penalty parameters were tuned via 4-fold cross-validation unless specified otherwise. The mini-batch size for Adam was one-third of the training dataset size.

4.1 Simulation study: Deconstructing EASIER-net

The purpose of this simulation is to study the contribution from each of the proposed modifications in EASIER-net. We do this by deconstructing EASIER-net, starting from a dense neural network and adding modifications progressively.

We consider three model-fitting procedures and ensemble versions of each one. The baseline model (`DropoutNet`) is a typical dense neural network that is regularized using a dropout rate of 15%; its ensemble version is similar to that in Olson et al. [34]. For

Model	Test loss	# layers	Avg # hidden nodes	# support
20 relevant, 600 observations				
DropoutNet-Single	0.497423	5	100.0	100.0
DropoutNet-Ensemble	0.355795	3.0	100.0	100.0
SparseNet-Single	0.241641	1	100.0	39.0
SparseNet-Ensemble	0.226045	3.0	100.0	100.0
SIER-net	0.223311	3	100.0	17.0
EASIER-net	0.226281	3.0	100.0	25.6
20 relevant, 3000 observations				
DropoutNet-Single	0.473178	5	100.0	100.0
DropoutNet-Ensemble	0.278896	5.0	100.0	100.0
SparseNet-Single	0.076208	1	65.0	25.0
SparseNet-Ensemble	0.060577	1.0	100.0	23.6
SIER-net	0.087473	5	100.0	43.0
EASIER-net	0.069780	5.0	100.0	43.2
100 relevant, 600 observations				
DropoutNet-Single	0.511054	3	100.0	100.0
DropoutNet-Ensemble	0.404107	3.0	100.0	100.0
SparseNet-Single	0.382923	1	71.0	58.0
SparseNet-Ensemble	0.399566	3.0	100.0	100.0
SIER-net	0.382481	3	100.0	64.0
EASIER-net	0.385061	5.0	100.0	92.0
100 relevant, 3000 observations				
DropoutNet-Single	0.513196	3	100.0	100.0
DropoutNet-Ensemble	0.304877	3.0	100.0	100.0
SparseNet-Single	0.271722	1	16.5	100.0
SparseNet-Ensemble	0.269778	1.0	17.7	100.0
SIER-net	0.264340	2	32.0	100.0
EASIER-net	0.256367	2.0	86.4	100.0

Table 3.1: We deconstruct EASIER-net to understand how each of the three modifications — sparsity, ensembling, and skip-connections — affect the test loss and the selected network structure. The baseline method `DropoutNet` only regularizes the model using dropout and requires tuning one hyperparameter. Next, the `SparseNet` method adds lasso penalties and increases the number of hyperparameters to three. Then, `SIER-net` adds skip-connections and reduces the number of hyperparameters to two; its ensemble version is `EASIER-net`. For `DropoutNet` and `SparseNet`, we denote the two single and ensemble versions using `-Single` and `-Ensemble`.

`DropoutNet` and its ensemble version, we only tune the number of hidden layers. Next, `SparseNet` adds an input filter layer and regularizes the network weights using two L_1 penalties, one to encourage sparsity in the inputs and another for the network weights. For `SparseNet` and its ensemble version, we tune the number of hidden layers and two L_1 penalty parameters. Finally, `SIER-net` adds skip-connections to `SparseNet`, keeps the initial number of hidden layers fixed, and reduces the number of hyperparameters to two. All networks were initialized with 100 hidden nodes per layer.

We consider four different simulation setups to reflect a variety of probable settings. Covariates $(X_1, ..., X_{100})$ are sampled independently from Unif$(0, 1)$. Response Y is defined as

$$Y = \sum_{i=0}^{m} (\sin(2x_{4i+1} + 2x_{4i+2}) + 5x_{4i+3} \, |x_{4i+4} - 0.25|) + \epsilon$$

where ϵ is a Gaussian random variable with variance chosen so that the signal to noise ratio is 2. To vary the sparsity of the true data-generating mechanism, we consider $m = 4$ and 19, which correspond to 20 and 100 relevant covariates, respectively. We also vary the amount of training data by considering 600 versus 3000 observations. We tune the hyperparameters using a single training validation split, where the number of validation samples is one-fourth of that for training.

As shown in Table 3.1, sparsity and ensembling improve the test loss in all four simulation settings. Sparsity offers the most improvement when the true model truly depends on a small subset of the covariates, but is still helpful when the true model depends on all the covariates. Ensembling offers the most improvement to the dense neural network because sparse models tend to be smaller and have more similar structures. The improvement in prediction accuracy by ensembling sparse-input hierarchical networks appears to depend on the data. Although ensembling seemed to improve accuracy only slightly in these simulations, we find that it provides significant gains on certain datasets in Section 4.3.

Ensemble sparse neural networks and EASIER-net have very similar predictive accuracy in all simulation settings, which is exactly as desired. Recall that the former requires

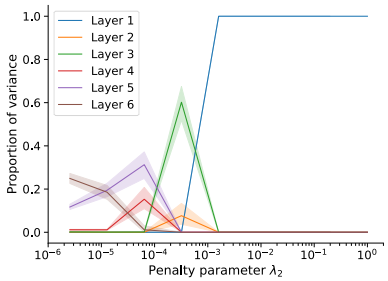


Figure 3.2: Illustration of how varying the penalty parameter λ_2 controls the proportion of variance contributed by each layer to the final prediction, which thereby controls the size of the fitted network. When $\lambda_2 = 1$, the fitted network only depends on the first layer, which means it is linear in the inputs. As λ_2 decreases, the higher levels contribute more to the final prediction and the lower levels contribute less.

tuning three hyperparameters whereas the latter requires tuning only two. Thus, the skip-connections in EASIER-net help reduce the amount of computation time spent on network structure learning without sacrificing predictive accuracy.

Finally, we give an example to illustrate how varying penalty parameter λ_2 modulates the dependence on each of the network layers. Figure 3.2 plots the proportion of variance contributed by each layer, as defined in 3.3. For large values of λ_2, the proportion of variance contributed by the scaled inputs (layer $l = 1$) is one, which means the neural network is simply a linear model. As λ_2 decreases, the contribution from scaled inputs decreases, and the contribution by the hidden layers increase in the order of their depth; the first and second hidden layers begin contributing earliest, then the third and fourth, and finally the fifth. As expected, the fifth hidden layer contributes the most as λ_2 approaches zero, since the last hidden layer has the highest model capacity.

4.2 Simulation study: Variable selection

In this section, we study the two claims in this chapter regarding variable selection by EASIER-net: i) the number of false negatives should be small if the truly relevant covariates contain predictive information that the other covariates do not and ii) highly correlated variables should be selected with similar proportions across the ensemble.

We run a simulation study where we vary the correlation between the covariates. For different values of $\rho \in [0, 1]$, we generate eight covariates $\boldsymbol{X}$ by sampling $\tilde{X}_i$ from Unif$(0, 1)$ for $i = 1, ..., 8$ and defining $X_i = \tilde{X}_i$ and $X_{i+4} = \rho\tilde{X}_i + (1 - \rho)\tilde{X}_{i+4}$ for $i = 1, ..., 4$. The response only depends on the first four through the equation $Y = X_1 * X_2 + \sin(X_3 + X_4) + \epsilon$, where ϵ is a Gaussian random variable with variance such that the signal to noise ratio is 2. We generated a total of 500 observations. To get accurate estimates of the variable selection rates, we fit EASIER-net with $B = 100$ members rather than 20.

In this simulation, EASIER-net was highly effective at support screening for ρ from 0 to 0.8 (Table 3.2). In fact, we had perfect support recovery in most cases, which is a more difficult task than support screening. For $\rho = 0.8$, only one member in the ensemble omitted the first covariate and selected the fifth one instead. In addition, these results show that most member networks will select the same support in settings where support screening is effective using a single network. This means that ensembling does not make sparse-input hierarchical networks less interpretable when support screening is feasible.

The simulation results also show a clear grouping effect from ensembling. As ρ increases, the selection probabilities of the correlated covariates X_i and X_{i+4} get closer, showing evidence of the grouping effect. In the extreme case where covariates $(X_1, ..., X_4)$ are identical to $(X_5, ..., X_8)$ (i.e. $\rho = 1$), the selection probabilities for X_i and X_{i+4} are similar for all $i = 1, ..., 4$. This is unsurprising since no procedure can differentiate which covariates are truly relevant in this setting.

4.3 Evaluation on UCI datasets

We study the performance of EASIER-net on five classification problems and six regression problems from the UCI repository [8], whose details are shown in Table 3.4 of the Appendix. To represent a variety of problem settings, the selected datasets vary in sample size, number of features, and number of classes. We compare against three other classifiers: logistic or linear regression with the lasso, random forests, and gradient boosted trees. Based on Olson

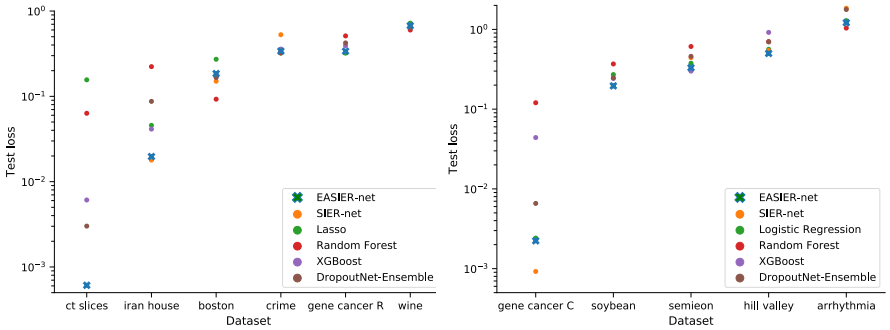
	Covariate index							
ρ	1	2	3	4	5	6	7	8
0.00	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000
0.50	1.000	1.000	1.000	1.000	0.000	0.000	0.000	0.000
0.80	0.980	1.000	1.000	1.000	0.020	0.000	0.000	0.000
0.90	0.860	0.890	0.870	0.990	0.160	0.120	0.280	0.010
0.95	0.630	0.730	0.540	0.940	0.380	0.300	0.570	0.270
1.00	0.550	0.480	0.470	0.570	0.460	0.540	0.600	0.640

Table 3.2: The proportion of networks in EASIER-net that select each of the covariates. Only the first four covariates are in the true support. We simulate covariates such that the correlation between X_i and X_{i+4} is ρ for $i = 1, ..., 4$. For small ρ, the model recovers the true support; for big ρ, the variable selection rates reflect the uncertainty of the true support.

et al. [34], we also fit an ensemble of dense neural networks with dropout with ten hidden layers and 100 hidden nodes per layer. We used the negative log likelihood and squared error as the loss functions for the classification and regression tasks, respectively.

EASIER-net attains the smallest test loss across the regression and classification datasets on average. In particular, they achieve the best performance on two of the five classification problems and two of the six regression problems (Figure 3.3). We found that EASIER-net typically has similar or better performance than SIER-net, presumably because they are able to quantify model uncertainty. Among the remaining datasets, SIER-net, Lasso with linear or logistic regression, random forests, and XGBoost attained top performance in at least one dataset. On the other hand, deep neural networks with dropout never achieved top performance on any dataset, because they can almost always benefit from network structure pruning.

The computation time for EASIER-net was on the same order as the other methods. For most of the datasets, the time to fit a single sparse-input hierarchical network took only a few minutes; for the largest dataset CT slices with 53500 observations, this took around forty minutes. By parallelizing both the ensembling and cross-validation procedures, the total time for running EASIER-net should take around double the time to fit a single sparse-



Method	Average test loss Regression	Classification
EASIER-net	**0.259**	**0.448**
SIER-net	0.286	0.605
Lasso	0.306	0.500
RF	0.302	0.568
XGBoost	0.273	0.541
DropoutNet	0.276	0.637

Figure 3.3: Comparison of test loss of different estimation methods across regression (left) and classification (right) datasets. The average test loss for each method is shown in the table.

input hierarchical network. Thus, given the prediction accuracy and computation time of EASIER-net, we conclude that they are a useful addition to the data analyst's toolbox.

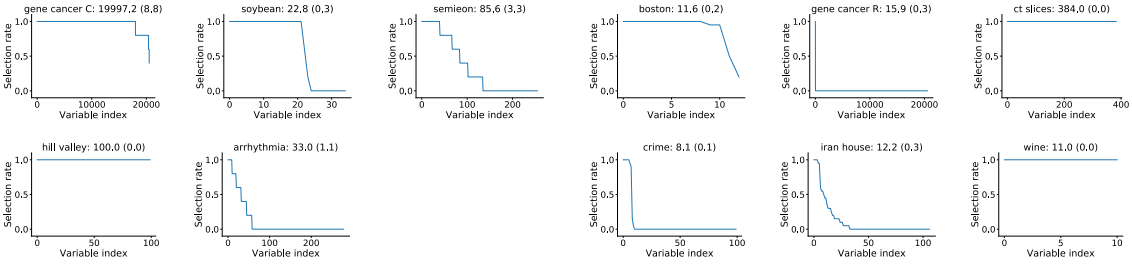
Across the eleven datasets, EASIER-net selected a variety of network structures via cross-validation (Table 3.3). In three cases, the selected model was linear in the input. In the remaining datasets, the network used at least one hidden layer, and the layer that contributed the most variance varied. We also show the variable selection rates for each dataset in Figure 3.4. The agreement across member networks in the ensemble depends on the dataset. For instance, the agreement was high on the soybean dataset and was much lower on the arrhythmia, semieon, and Iran house datasets. This is consistent with our analysis: the soybean dataset contains a small number of high-level features that likely measure different pieces of information whereas many features in the arrhythmia dataset were extracted from the same channel [14].

5 Discussion

We have shown that an ensemble of sparse-input hierarchical networks is a useful modeling technique for datasets with a small number of observations. By employing skip-connections and sparsity-inducing penalties, this methodology can data-adaptively determine the appropriate model complexity: they can fit simple linear models, shallow neural networks, or

Dataset	Avg # hidden nodes	Proportion of variance contributed by layer 1	2	3	4	5	6
		Classification tasks					
gene cancer C	0.0 (0.0)	**1.00** (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
soybean	0.0 (0.0)	**1.00** (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
semieon	100.0 (0.0)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	**0.99** (0.00)
hill valley	100.0 (0.0)	0.00 (0.00)	0.00 (0.00)	0.01 (0.00)	0.01 (0.00)	0.02 (0.01)	**0.61** (0.03)
arrhythmia	60.0 (6.1)	0.00 (0.00)	0.00 (0.00)	**0.61** (0.10)	0.32 (0.09)	0.07 (0.03)	0.00 (0.00)
		Regression tasks					
crime	6.5 (0.6)	0.00 (0.00)	0.00 (0.00)	**0.95** (0.05)	0.05 (0.05)	0.00 (0.00)	0.00 (0.00)
ct slices	100.0 (0.0)	0.02 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.03 (0.01)	**0.46** (0.02)
boston	100.0 (0.0)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.10 (0.02)	**0.53** (0.04)
iran house	100.0 (0.0)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.01 (0.00)	0.08 (0.00)	**0.32** (0.01)
wine	9.9 (4.9)	0.01 (0.01)	**0.71** (0.11)	0.30 (0.11)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
gene cancer	0.0 (0.0)	**1.00** (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)

Table 3.3: Summaries of the structure of member networks from EASIER-net: the average number of hidden nodes per active layer and the proportion of variance contributed by each layer. Standard errors are shown in parentheses. We highlight the layer that contributes the most to the network prediction.



(a) Classification (b) Regression

Figure 3.4: For each dataset, the proportion of member networks from EASIER-net that selected each variable, where each ensemble consists of 20 independently-trained networks. Variables are sorted along the x-axis according to their selection rates. The plot titles show the average support size and the standard error in parentheses. The variable selection rates reflect the uncertainty of the true support.

deep networks. Empirically, we find that EASIER-net can significantly outperform popular machine learning algorithms in terms of prediction accuracy. Moreover, the model can do variable screening when the mutual information between the relevant and irrelevant covariates is low, and appears to induce a grouping effect when the mutual information is high.

The theoretical analyses of SIER-net provide probability bounds on the number of false negatives. We have not bounded the number of false positives and leave this to future work. Nevertheless, the empirical experiments demonstrate that both SIER-net and EASIER-net are able to perform support recovery under certain conditions.

Because neural networks are highly modular, our method can be easily extended. For example, by modifying the output layer and objective function, we can output prediction sets/intervals to better quantify model uncertainty [45, 11]. For specific problem domains, we can also tailor the network structure and sparsity pattern to reflect our prior knowledge, such as applying a group lasso to known groups of covariates [52].

Finally, interpretability of SIER-net and EASIER-net remains an issue, even with the help of sparsity. To better understand the model's inner-workings, one may try using techniques like variable importance measures [39, 28, 10], saliency maps [43], and influence functions [22].

A Appendix

A.1 Sparse-input hierarchical network definition

For the reader's convenience, the full set of equations that define a sparse-input hierarchical network $f_{W,b,\beta,\alpha}$ is given below:

$$z_{1,i} = \beta_i x_i \qquad \forall i = 1, ..., d \tag{3.13}$$

$$\boldsymbol{z}_l = \phi(\boldsymbol{z}_{l-1} W_{l-1} + \boldsymbol{b}_{l-1}) \qquad \forall l = 2, ..., L-1 \tag{3.14}$$

$$\boldsymbol{\zeta}_l = \boldsymbol{z}_l W_l' + \boldsymbol{b}_l' \qquad \forall l = 1, ..., L-1 \tag{3.15}$$

$$f_{W,b,\beta,\alpha}(\boldsymbol{x}) = \boldsymbol{z}_L = \phi_{\text{out}} \left(\sum_{l=1}^{L-1} \frac{|\alpha_l|}{\sum_{l'=1}^{L-1} |\alpha_{l'}|} \boldsymbol{\zeta}_l \right) \tag{3.16}$$

A.2 Proofs

For convenience, we overload the notation as follows. Consider a probability space $(\Omega, \mathcal{A}, P)$ with sample space $\Omega = \mathcal{X} \times \mathcal{Y}$, σ-algebra $\mathcal{A}$, and probability measure P. For a real-valued function $f : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, let $Pf(X, Y)$ denote the expected value of $f(X, Y)$. For a set-valued function $g : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{A}$, let $Pg(X, Y)$ denote the probability measure of $g(X, Y)$.

The proof for Theorem 2 uses standard empirical process techniques. Recall that for a given distribution μ, the μ-complexity of function class $\mathcal{F}$ for sample $(X_1, ..., X_n)$ is defined as

$$E \left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \xi_i (f(X_i) - f^*(X_i)) \mid X_1, ..., X_n \right] \tag{3.17}$$

where $\xi \sim \mu$ are independent and identically distributed (IID). The Gaussian complexity of $\mathcal{F}$ refers to the case when $\mu = N(0, 1)$. Let $\mathcal{G}_n(\mathcal{F})$ denote the uniform bound of the Gaussian complexity of $\mathcal{F}$ over all samples from $\mathcal{X}$. The Rademacher complexity of $\mathcal{F}$ refers to the case when μ is the Rademacher distribution. Analogously, let $\mathcal{R}_n(\mathcal{F})$ denote the uniform bound of the Rademacher complexity of $\mathcal{F}$ over all samples from $\mathcal{X}$.

We will use the μ-complexity bounds derived in Barron and Klusowski [3]. We note that their bounds technically pertain to dense ReLU neural networks, not sparse-input hierarchical networks. Nevertheless, it is easy to show that any sparse-input hierarchical network can be expressed as a dense ReLU neural network, albeit with additional hidden nodes per layer. Thus their derived complexity bounds are also valid for sparse-input hierarchical networks.

Proof for Theorem 2. Let P_n be the empirical distribution of the training data. Because $\hat{\theta}$ is a global empirical risk minimizer, we have that

$$\begin{aligned} P\left\{(f_{\hat{\theta}_n}(X)-Y)^2-(f_{\theta^*}(X)-Y)^2\right\} &\leq (P-P_n)\left((f_{\hat{\theta}_n}(X)-Y)^2-(f_{\theta^*}(X)-Y)^2\right) \\ &= (P-P_n)\left((f_{\hat{\theta}_n}(X)-f_{\theta^*}(X))^2-\epsilon(f_{\theta^*}(X)-Y)\right). \end{aligned}$$

Then for any $j=1,\cdots,d$, we have that

$$P\left(|s-\hat{s}|\geq j\right) \tag{3.18}$$

$$\leq P\left(P\left\{(f_{\hat{\theta}_n}(X)-Y)^2-(f_{\theta^*}(X)-Y)^2\right\}\geq \gamma(j;P)\right) \tag{3.19}$$

$$\leq P\left((P-P_n)\left((f_{\hat{\theta}_n}(X)-f_{\theta^*}(X))^2-\epsilon(f_{\hat{\theta}_n}(X)-f_{\theta^*}(X))\right)\geq \gamma(j;P)\right) \tag{3.20}$$

$$\leq P\left(\left[\sup_{\theta\in\Theta_{L,B_0,B_1}}(P-P_n)\left(f(X)-f_{\theta^*}(X)\right)^2\right]+\left[\sup_{\theta\in\Theta_{L,B_0,B_1}}(P-P_n)\,\epsilon(f(X)-f_{\theta^*}(X))\right]\geq\gamma(j;P)\right). \tag{3.21}$$

For convenience, let $\mathcal{F}_{L,B_0,B_1}=\{f_\theta:\theta\in\Theta_{L,B_0,B_1}\}$. We can bound the above probability

by the union bound

$$P\left(|s-\hat{s}| \geq j\right) \leq P\left(4B_0\mathcal{R}_n(\mathcal{F}_{L,B_0,B_1}) + \delta + \sigma\mathcal{G}_n(\mathcal{F}_{L,B_0,B_1}) + \delta \geq \gamma(j;P)\right) \tag{3.22}$$

$$+ P\left(\sup_{f\in\mathcal{F}_{L,B_0,B_1}} (P-P_n)\left(f(X)-f_{\theta^*}(X)\right)^2 - 4B_0\mathcal{R}_n(\mathcal{F}_{L,B_0,B_1}) \geq \delta\right) \tag{3.23}$$

$$+ P\left(\sup_{f\in\mathcal{F}_{L,B_0,B_1}} (P-P_n)\,\epsilon(f(X)-f_{\theta^*}(X)) - \sigma\mathcal{G}_n(\mathcal{F}_{L,B_0,B_1}) \geq \delta\right) \tag{3.24}$$

So with minor modifications of the Rademacher complexity to handle squared of functions with infinity-norm bounded by B_0, we have that (3.23) is bounded by $\exp\left(-\frac{n\delta^2}{2B_0^4}\right)$. Likewise, since ϵ is a mean-zero Gaussian RV with variance σ^2, then (3.24) is bounded by $2\exp\left(-\frac{n\delta^2}{8B_0^2\sigma^2}\right)$. Combining the above inequalities, we have for all $j = 1, \cdots, d$ that

$$P\left(|s-\hat{s}| \geq j\right) \leq \mathbb{1}\left\{4B_0\mathcal{R}_n(\mathcal{F}_{L,B_0,B_1}) + \sigma\mathcal{G}_n(\mathcal{F}_{L,B_0,B_1}) + 2\delta \geq \gamma(j;P)\right\} \tag{3.25}$$

$$+ \exp\left(-\frac{n\delta^2}{2B_0^4}\right) \tag{3.26}$$

$$+ 2\exp\left(-\frac{n\delta^2}{8B_0^2\sigma^2}\right). \tag{3.27}$$

Finally, since Barron and Klusowski [3] proved that $\mathcal{G}_n(\mathcal{F}_{L,B_0,B_1})$ and $\mathcal{R}_n(\mathcal{F}_{L,B_0,B_1})$ are both bounded by $B_1\sqrt{2(L\log(2)+\log(2d))/n}$, we can plug this into (3.25) and obtain our desired result. □

Our proof for Theorem 3 depends on the following lemma, which is a straightforward adaptation of Proposition 15.1 in Wainwright [47]. For any positive integer $m \leq \min(k, \lfloor d/2 \rfloor)$, select two distributions $P_{(1)}, P_{(2)} \in \mathcal{P}_{k,\gamma}$ such that

$$\Delta(P_{(1)}, P_{(2)}) = \min(|s(\mu_{P_{(1)}}) \setminus s(\mu_{P_{(2)}})|, |s(\mu_{P_{(2)}}) \setminus s(\mu_{P_{(1)}})|) \geq 2m. \tag{3.28}$$

Let Q denote the joint distribution over the pair of random variables (Z, Y) generated using

the following procedure:

1. Sample J from $\{1, 2\}$ uniformly at random.

2. Given $J = j$, sample $X_1, ..., X_n$ iid from $P_{(j)}$.

Define a testing function ψ as a mapping from $\mathcal{X}^n \mapsto \{1, 2\}$.

Lemma 5. *For any positive integer $m \leq \min(k, d/2)$, we have that*

$$\inf_{\hat{f}_n : |s(\hat{f}_n)| \leq k} \sup_{P \in \mathcal{P}_{k,\gamma}} P\left(\left|s(\mu_P) \setminus s(\hat{f}_n)\right| \geq m\right) \geq \inf_{\psi} Q\left(\psi(X_1, ..., X_n) \neq J\right) \tag{3.29}$$

where the infimum ranges over all test functions.

Proof. For any M, we have that

$$\sup_{P \in \mathcal{P}_{k,\gamma}} P\left(\left|s(\mu_P) \setminus s(\hat{f}_n)\right| \geq m\right) \geq \frac{1}{2} \sum_{j=1}^{2} P_{(j)}\left(\left|s(\mu_{P_{(j)}}) \setminus s(\hat{f}_n)\right| \geq m\right). \tag{3.30}$$

Define the testing function $\psi(X_1, ..., X_n) = \arg\min_{j \in \{1,2\}} \left|s(\mu_{P_{(j)}}) \setminus s(\hat{f}_n)\right|$. We next show that if $\mu_{P_{(j)}}$ is the true distribution, the event $\left[\left|s(\mu_{P_{(j)}}) \setminus s(\hat{f}_n)\right| \leq m\right]$ implies that $\psi(X_1, ..., X_n) = j$. Because $\hat{f}_n$ can choose at most k elements in the support and the assumption that $\mu_{P_{(1)}}$ and $\mu_{P_{(2)}}$ differ by at least $2m$ elements in their support, we have that if $\left|s(\mu_{P_{(j)}}) \setminus s(\hat{f}_n)\right| \leq m$ for some $j = 1, 2$, then $\left|s(\mu_{P_{(j')}}) \setminus s(\hat{f}_n)\right| \geq m$ for $j' \neq j$. Moreover, this implies that $\psi(X_1, ..., X_n) = j$. Thus, we have established that

$$\frac{1}{2} \sum_{j=1}^{2} P_{(j)}\left(\left|s(\mu_{P_{(j)}}) \setminus s(\hat{f}_n)\right| \geq m\right) \geq Q\left(\psi(X_1, ..., X_n) \neq J\right). \tag{3.31}$$

Finally, take the infimum with respect to all estimators on the left hand side and the infimum over all induced tests on the right hand side. Since the full infimum can only be smaller, we have established the desired result. □

Proof for Theorem 3. From Le Cam's inequality, for any $P_{(1)}, P_{(2)} \in \mathcal{P}_{k,\gamma}$ that satisfy (3.28), we have that

$$\inf_{\psi} Q\left(\psi(X_1, ..., X_n) \neq J\right) \geq \frac{1}{2}\left(1 - \left\|P_{(1)}^n - P_{(2)}^n\right\|_{TV}\right), \tag{3.32}$$

where $\|\cdot\|_{TV}$ is the total variation norm for probability distributions. We then lower bound the right hand side by relating the KL-divergence to the total variation norm as follows:

$$\left\|P_{(1)}^n - P_{(2)}^n\right\|_{TV} \leq \sqrt{\frac{1}{2}D(P_{(1)}^n||P_{(2)}^n)} = \sqrt{\frac{n}{2}D(P_{(1)}||P_{(2)})}. \tag{3.33}$$

Since we assumed that $\epsilon \sim N(0, \sigma^2)$, the squared error loss ℓ is equal to the negative log likelihood scaled by σ^2. Thus, $\gamma(2m)/\sigma^2$ is the minimum KL-divergence between two functions in $\mathcal{P}_{k,\gamma}$ with support differing by $2m$, i.e.

$$\inf_{P_{(1)}, P_{(2)} \in \mathcal{P}_{k,\gamma}: \Delta(P_{(1)}, P_{(2)}) \geq 2m} D(P_{(1)}||P_{(2)}) = \gamma(2m)/\sigma^2. \tag{3.34}$$

Combining the above results with Lemma 5, we have that

$$\inf_{\hat{f}_n: |s(\hat{f}_n)| \leq k} \sup_{P \in \mathcal{P}_{k,\gamma}} P\left(\left|s(\mu_P) \setminus s(\hat{f}_n)\right| \geq m\right) \geq \sup_{P_{(1)}, P_{(2)} \in \mathcal{P}_{k,\gamma}: \Delta(P_{(1)}, P_{(2)}) \geq 2m} \frac{1}{2}\left(1 - \left\|P_{(1)}^n - P_{(2)}^n\right\|_{TV}\right) \tag{3.35}$$

$$\geq \frac{1}{2}\left(1 - \sqrt{\frac{n}{2}\gamma(2m)/\sigma^2}\right), \tag{3.36}$$

where the first inequality follows from taking the supremum over the right hand side of (3.32). □

A.3 Hyperparameter default values

We recommend selecting the size of the ensemble B to be sufficiently large for its predicted value to plateau. We found that $B = 20$ worked well in all our experiments. We initialize the

Dataset	# features	# observations	# classes	Held-out proportion
		Classification		
soybean	35	307	19	1/4
arrythmia	279	452	13	1/4
gene cancer C	20531	801	5	1/3
hill valley	100	606	2	1/3
semeion	256	1593	10	1/3
		Regression		
boston	13	506	–	1/3
gene cancer R	20530	801	–	1/3
ct slices	384	53500	–	1/3
crime	122	1994	–	1/3
iran house	103	372	–	1/3
wine	11	4898	–	1/3

Table 3.4: Summary statistics for the selected datasets from the UCI Machine Learning Repository. The datasets were chosen to represent varying dataset shapes and sizes. One third of the data was held out for testing unless a random split resulted in no samples from a particular class. The gene cancer regression (gene cancer R) task is a derivative of the gene cancer classification (gene cancer C) task, where we try to predict the expression value for the first gene in the dataset instead of the type of cancer.

sparse-input hierarchical networks to be sufficiently wide and deep to obtain a small training loss. In the empirical analyses, we use 5 hidden layers and 100 hidden nodes per layer. To perform penalized empirical minimization, we run Adam with the default learning rates and parameters until convergence and then batch proximal gradient descent until convergence. We use a mini-batch size that is one third of the total size of the data.

To speed up K-fold cross-validation, one could try to fit a smaller ensemble for each candidate penalty parameter set. In our work, we used $B = 10$ for tuning the penalty parameters.

BIBLIOGRAPHY

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale machine learning on heterogeneous distributed systems. March 2016.

[2] Johnathan M Bardsley, Antti Solonen, Heikki Haario, and Marko Laine. Randomize-Then-Optimize: A method for sampling from posterior distributions in nonlinear inverse problems. SIAM J. Sci. Comput., 36(4):A1895–A1910, January 2014. ISSN 1064-8275. doi: 10.1137/140964023.

[3] Andrew R Barron and Jason M Klusowski. Complexity, statistical risk, and metric entropy of deep nets using total path variation. arXiv, February 2019.

[4] Yoshua Bengio. Practical recommendations for Gradient-Based training of deep architectures. In Grégoire Montavon, Geneviève B Orr, and Klaus-Robert Müller, editors, Neural Networks: Tricks of the Trade: Second Edition, pages 437–478. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 9783642352898. doi: 10.1007/978-3-642-35289-8_26.

[5] James Bergstra and Yoshua Bengio. Random search for Hyper-Parameter optimization. J. Mach. Learn. Res., 13(Feb):281–305, 2012. ISSN 1532-4435, 1533-7928.

[6] Leo Breiman. Bagging predictors. Mach. Learn., 24(2):123–140, August 1996. ISSN 0885-6125, 1573-0565. doi: 10.1023/A:1018054314350.

[7] Peter Bühlmann and Sara van de Geer. Statistics for High-Dimensional Data: Methods, Theory and Applications. Springer, Berlin, Heidelberg, 2011. ISBN 9783642201912. doi: 10.1007/978-3-642-20192-9.

[8] Dheeru Dua and Casey Graff. UCI machine learning repository. arXiv, 2017. URL http://archive.ics.uci.edu/ml.

[9] Jean Feng and Noah Simon. Sparse-Input neural networks for high-dimensional nonparametric regression and classification. arXiv, 2019.

[10] Jean Feng, Brian Williamson, Noah Simon, and Marco Carone. Nonparametric variable importance using an augmented neural network with multi-task learning. ICML, 80: 1496–1505, 2018.

[11] Jean Feng, Arjun Sondhi, Jessica Perry, and Noah Simon. Selective prediction-set models with coverage guarantees. arXiv, June 2019.

[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. International Conference on Artificial Intelligence and Statistics, 2011.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, November 2016. ISBN 9780262337373.

[14] H A Guvenir, B Acar, G Demiroz, and A Cekin. A supervised machine learning algorithm for arrhythmia analysis. In Computers in Cardiology 1997, pages 433–436. ieeexplore.ieee.org, September 1997. doi: 10.1109/CIC.1997.647926.

[15] Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the NIPS 2003 feature selection challenge. In L K Saul, Y Weiss, and L Bottou, editors, Advances in Neural Information Processing Systems 17, pages 545–552. MIT Press, 2005.

[16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, Advances in Neural Information Processing Systems 28, pages 1135–1143. Curran Associates, Inc., 2015.

[17] Stephen Jose Hanson and Lorien Y Pratt. Comparing biases for minimal network construction with Back-Propagation. In D S Touretzky, editor, Advances in Neural Information Processing Systems 1, pages 177–185. Morgan-Kaufmann, 1989.

[18] K He, X Zhang, S Ren, and J Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

[19] Jennifer A Hoeting, David Madigan, Adrian E Raftery, and Chris T Volinsky. Bayesian model averaging: a tutorial. Stat. Sci., 14(4):382–417, November 1999. ISSN 0883-4237, 2168-8745. doi: 10.1214/ss/1009212519.

[20] G Huang, Z Liu, L v d. Maaten, and K Q Weinberger. Densely connected convolutional networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2261–2269, July 2017. doi: 10.1109/CVPR.2017.243.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. International Conference for Learning Representations, 2015.

[22] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. Proceedings of the 34th International Conference on Machine Learning, 2017.

[23] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, Advances in

Neural Information Processing Systems 30, pages 6402–6413. Curran Associates, Inc., 2017.

[24] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In D S Touretzky, editor, Advances in Neural Information Processing Systems 2, pages 598–605. Morgan-Kaufmann, 1990.

[25] Faming Liang, Qizhai Li, and Lei Zhou. Bayesian neural networks for selection of drug sensitive genes. J. Am. Stat. Assoc., 113(523):955–972, July 2018. ISSN 0162-1459. doi: 10.1080/01621459.2017.1409122.

[26] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. International Conference on Learning Representations, February 2018.

[27] Xiuyuan Lu and Benjamin Van Roy. Ensemble sampling. Advances in Neural Information Processing Systems, pages 3258–3266, 2017.

[28] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. Advances in Neural Information Processing Systems 30, pages 4765–4774, 2017.

[29] David J C MacKay. Bayesian Non-Linear modeling for the prediction competition. In Glenn R Heidbreder, editor, Maximum Entropy and Bayesian Methods: Santa Barbara, California, U.S.A., 1993, pages 221–234. Springer Netherlands, Dordrecht, 1996. ISBN 9789401587297. doi: 10.1007/978-94-015-8729-7_18.

[30] David Madigan and Adrian E Raftery. Model selection and accounting for model uncertainty in graphical models using occam's window. J. Am. Stat. Assoc., 89(428): 1535–1546, 1994. ISSN 0162-1459. doi: 10.2307/2291017.

[31] Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. J. Mach. Learn. Res., January 2017. ISSN 1532-4435.

[32] Radford M Neal. Bayesian Learning for Neural Networks. Lecture Notes in Statistics. 1996. ISBN 9780387947242. doi: 10.1007/978-1-4612-0745-0.

[33] Radford M Neal and Jianguo Zhang. High dimensional classification with bayesian neural networks and dirichlet diffusion trees. In Isabelle Guyon, Masoud Nikravesh, Steve Gunn, and Lotfi A Zadeh, editors, Feature Extraction: Foundations and Applications, pages 265–296. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 9783540354888. doi: 10.1007/978-3-540-35488-8_11.

[34] Matthew Olson, Abraham Wyner, and Richard Berk. Modern neural networks generalize on small data sets. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, Advances in Neural Information Processing Systems 31, pages 3619–3628. Curran Associates, Inc., 2018.

[35] Neal Parikh and Stephen Boyd. Proximal algorithms. Foundations and Trends® in Optimization, 1(3):127–239, 2014. ISSN 2167-3888. doi: 10.1561/2400000003.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, High-Performance deep learning library. In Advances in Neural Information Processing Systems 32, pages 8026–8037. Curran Associates, Inc., 2019.

[37] Tim Pearce, Felix Leibfried, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. Uncertainty in neural networks: Approximately bayesian ensembling. International Conference on Artificial Intelligence and Statistics, 2020.

[38] Adrian E Raftery, David Madigan, and Jennifer A Hoeting. Bayesian model averaging

for linear regression models. J. Am. Stat. Assoc., 92(437):179–191, March 1997. ISSN 0162-1459. doi: 10.1080/01621459.1997.10473615.

[39] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pages 1135–1144. dl.acm.org, 2016.

[40] Brian D Ripley and N L Hjort. Pattern Recognition and Neural Networks. Cambridge University Press, January 1996. ISBN 9780521460866.

[41] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. Neurocomputing, 241:81–89, June 2017. ISSN 0925-2312. doi: 10.1016/j.neucom.2017.02.029.

[42] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A Sparse-Group lasso. J. Comput. Graph. Stat., 22(2):231–245, April 2013. ISSN 1061-8600. doi: 10.1080/10618600.2012.681250.

[43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for Large-Scale image recognition. International Conference on Learning Representations, 2015.

[44] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 2951–2959. Curran Associates, Inc., 2012.

[45] James W Taylor. A quantile regression neural network approach to estimating the conditional density of multiperiod returns. J. Forecast., 19(4):299–311, 2000. ISSN 0277-6693.

[46] V Viallefont, A E Raftery, and S Richardson. Variable selection and bayesian model

averaging in case-control studies. Stat. Med., 20(21):3215–3230, November 2001. ISSN 0277-6715. doi: 10.1002/sim.976.

[47] Martin J Wainwright. High-Dimensional Statistics: A Non-Asymptotic Viewpoint. Cambridge University Press, February 2019. ISBN 9781108498029.

[48] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett, editors, Advances in Neural Information Processing Systems 29, pages 2074–2082. Curran Associates, Inc., 2016.

[49] Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. arXiv, February 2020.

[50] Jaehong Yoon and Sung Ju Hwang. Combined group and exclusive sparsity for deep neural networks. In Doina Precup and Yee Whye Teh, editors, Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pages 3958–3966, International Convention Centre, Sydney, Australia, 2017. PMLR.

[51] D Yu, F Seide, G Li, and L Deng. Exploiting sparseness in deep neural networks for large vocabulary speech recognition. In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4409–4412, March 2012. doi: 10.1109/ICASSP.2012.6288897.

[52] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. J. R. Stat. Soc. Series B Stat. Methodol., 68(1):49–67, 2006. ISSN 1369-7412.

[53] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. International Conference on Learning Representations, 2017.

[54] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. J. R. Stat. Soc. Series B Stat. Methodol., 67(2):301–320, 2005. ISSN 1369-7412, 1467-9868.

Chapter 4

APPROVAL POLICIES FOR MODIFICATIONS TO MACHINE LEARNING-BASED SOFTWARE AS A MEDICAL DEVICE: A STUDY OF BIO-CREEP

Summary

Successful deployment of machine learning algorithms in healthcare requires careful assessments of their performance and safety. To date, the FDA approves locked algorithms prior to marketing and requires future updates to undergo separate premarket reviews. However, this negates a key feature of machine learning–the ability to learn from a growing dataset and improve over time. This chapter frames the design of an approval policy, which we refer to as an automatic algorithmic change protocol (aACP), as an online hypothesis testing problem. As this process has obvious analogy with noninferiority testing of new drugs, we investigate how repeated testing and adoption of modifications might lead to gradual deterioration in prediction accuracy, also known as "bio-creep" in the drug development literature. We consider simple policies that one might consider but do not necessarily offer any error-rate guarantees, as well as policies that do provide error-rate control. For the latter, we define two online error-rates appropriate for this context: Bad Approval Count (BAC) and Bad Approval and Benchmark Ratios (BABR). We control these rates in the simple setting of a constant population and data source using policies aACP-BAC and aACP-BABR, which combine alpha-investing, group-sequential, and gate-keeping methods. In simulation studies, bio-creep regularly occurred when using policies with no error-rate guarantees, whereas aACP-BAC and -BABR controlled the rate of bio-creep without substantially impacting our ability to approve beneficial modifications.

1 Introduction

Due to the rapid development of artificial intelligence (AI) and machine learning (ML), the use of AI/ML-based algorithms has expanded in the medical field. As such, an increasing number of AI/ML-based Software as a Medical Device (SaMD) are seeking approval from the Center of Diagnostics and Radiologic Health (CDRH) at the US Food and Drug Administration (FDA). In the past couple years, the FDA has approved algorithms for computer-aided diagnosis [2], medical triage [27], consumer-facing health devices [1], among others.

ML algorithms are attractive for their ability to improve over time by training over a growing body of data. Thus, rather than using a locked algorithm trained on a limited dataset, developers might like to train it further on a much more representative sample of the patient population that can only be obtained after deployment. To collect input on this regulatory problem, the FDA recently outlined a proposed regulatory framework for modifications to AI/ML-based SaMDs in a discussion paper [15].

Regulating evolving algorithms presents new challenges because the CDRH has historically only approved "locked" algorithms, i.e. algorithms that do not change after they are approved. This is a new regulatory problem because updating traditional medical devices and drugs is often logistically difficult whereas updating software is both fast and easy.

The FDA [15] proposes companies stipulate SaMD Pre-specifications (SPS) and an Algorithm Change Protocol (ACP). When listing the anticipated modifications in the SPS, it behooves the company to cast as wide a net as possible within FDA-imposed constraints. The ACP specifies how the company will ensure that their modifications are acceptable for deployment. Once the FDA approves the SPS and ACP, the company follows these pre-specified procedures to deploy changes without further intervention. As such, we refer to the ACP in this thesis as an "automatic ACP" (aACP). The aACP is the FDA's primary tool for ensuring safety and efficacy of the modifications. However, specific aACP designs or requirements are noticeably absent from FDA [15]. We aim to address this gap in this

chapter.

A manufacturer has two potential motivations for changing an AI/ML-based SaMD: to advance public health and to increase their financial wealth. Modifications that improve performance and usability are encouraged. On the other hand, changes that do not and are deployed only for the sake of change itself have been used in the past to advance a manufacturer's financial interest and are contrary to public interest. Historically, such modifications have been used to 1) decrease competition because it is difficult for competitors to compare against an ever-changing benchmark; 2) file for a patent extension and keep prices artificially high; and 3) increase sales for a supposedly new and improved product [20, 23, 19]. To prevent this type of behavior with drugs and biologics, the FDA regulates modifications through various types of bridging studies ([24]). Likewise, an aACP should only grant approval to modifications to AI/ML-based SaMD after ensuring safety and efficacy.

Here, we provide a framework for designing and evaluating an aACP, consider a variety of aACP designs, and investigate their operating characteristics. We assume the manufacturer is allowed to propose arbitrary (and possibly deleterious) modifications, which include changes to model parameters, structure, and input features. For this manuscript, we focus on the setting of a constant population and data source, rather than more complicated settings with significant time trends. Throughout, we evaluate modifications solely in terms of their operating characteristics. Thus, the aACPs treat simple models and complex black-box estimators, such as neural networks and boosted gradient trees, the same. This parallels the drug approval process, which primarily evaluates drugs on their efficacy and safety with respect to some endpoints, even if the biological mechanism is not completely understood.

To our knowledge, there is no prior work that directly addresses the problem of regulating modifications to AI/ML-based SaMD, though many have studied related problems. In online hypothesis testing, alpha-investing procedures are used to control the online false discovery rate (FDR) [18, 25, 30, 31, 35], which is important for companies that test many hypotheses over a long period of time (Tang et al. [34]). We will consider aACPs that use alpha-investing

to control online error rates; However, we will need to significantly adapt these ideas for use in our context. In addition, differential privacy methods [8, 12] have been used to tackle the problem of ranking model submissions to a ML competition, where the submissions are evaluated using the same test data and models are submitted in a sequential and adaptive manner. Though that problem is related, those approaches cannot evaluate modifications that add previously-unmeasured covariates. Finally, online learning methods are a major motivation for studying this regulatory problem and can be used to automatically update the model [32]. However, rather than designing bespoke aACPs for online learning methods, we will consider approval policies for arbitrary modifications as a first step.

We evaluate the rates at which different policies make bad approvals as well as their rates of approving beneficial modifications. Due to the analogy between this problem and noninferiority testing of new drugs, we investigate how repeated testing of proposed modifications might lead to gradual deterioration in model performance, also known as "bio-creep" [17]. We compare simple aACPs that one might consider, but do not necessarily have error-rate guarantees, to policies that *do* provide error rate control. For the latter, we define two online error rates appropriate for this context—the expected Bad Approval Count (BAC) and Bad Approval and Benchmark Ratios (BABR)—and control them using policies aACP-BAC and aACP-BABR, respectively. In simulation studies, bio-creep frequently occurred when using the simple aACPs. By using aACP-BAC or -BABR instead, we significantly reduce the risk of bio-creep without substantially reducing the rate of model improvement. Based on these findings, we conclude that 1) bio-creep is a major concern when designing an aACP and 2) there are promising solutions for mitigating it without substantially hindering model improvements.

2 Motivating examples

We present examples of actual AI/ML-based medical devices and discuss possible modifications that manufacturers might consider. The examples are ordered by increasing regulatory complexity and risk. Throughout, we only discuss regulating modifications to the software

and assume the intended use of the device remains constant.

2.1 Blood tests using computer vision

Sight Diagnostics has developed a device that collects and images blood samples to estimate complete blood count (CBC) parameters. They are evaluating the device in a clinical trial (ClinicalTrials.gov ID NCT03595501) where the endpoints are the estimated linear regression parameters (slope and intercept) between their CBC parameter estimates and gold standard.

The FDA requires locking the entire procedure, which includes blood collection, imaging, and the ML algorithm, prior to marketing. Nonetheless, the company might want to improve the accuracy of their test after obtaining regulatory approval. For instance, they can train more complex models that capture nonlinearities and interactions (between covariates and/or outcomes) or use a different FDA-approved device to image the blood sample. All these changes have the potential to improve prediction accuracy, though it is not guaranteed.

To regulate such modifications, we will need to define acceptable changes to endpoint values. This is not straightforward when multiple endpoints are involved: Do all endpoints have to improve? What if the model has near-perfect performance with respect to some endpoints and room for improvement for others? To tackle these questions, we must run both superiority and non-inferiority (NI) tests. Moreover, introducing NI tests prompts even more questions, such as how to choose an appropriate NI margin.

2.2 Detecting large vessel occlusion from CT angiogram images of the brain

ContaCT is a SaMD that identifies whether CT angiogram images of the brain contain a suspected large vessel occlusion. If so, it notifies a medical specialist to intervene. The manufacturer evaluated ContaCT using images analyzed by neuro-radiologists. The primary endpoints were estimated sensitivity and specificity. The secondary endpoint was the

difference in notification time between ContaCT and standard-of-care. ContaCT achieved 87% sensitivity and 89% specificity and significantly shortened notification time.

Having obtained FDA approval [14], the company might want to improve ContaCT by, say, training on more images, extracting a different set of image features, or utilizing clinical covariates from electronic health records. This last modification type requires special consideration since the distribution of clinical covariates and their missingness distribution are susceptible to time trends.

2.3 Blood test for cancer risk prediction

GRAIL is designing a blood test that sequences cell-free nucleic acids (cfNAs) circulating in the blood to detect cancer early. They are currently evaluating this test in an observational study (ClinicalTrials.gov ID NCT02889978) where the gold standard is a cancer diagnosis from the doctor within 30 months. For time-varying outcomes, one may consider evaluating performance using time-dependent endpoints, such as those in Heagerty and Zheng [21].

After the blood test is approved, GRAIL might still want to change their prediction algorithm. For example, they could collect additional omics measurements, sequence the cfNAs at a different depth (e.g. lower to decrease costs, higher to improve accuracy), or train the model on more data. Regulating modifications to this blood test is particularly difficult because the gold standard might not be observable in all patients, its definition can vary between doctors, and it cannot be measured instantaneously. In fact, the gold standard might not be measurable at all because test results will likely affect patient and doctor behavior.

3 Problem setup

In this section, we provide a general framework and abstractions to understand the approval process for modifications to AI/ML-based SaMD. We begin with reviewing the approval process for a single AI/ML-based SaMD since it forms the basis of our understanding and is a prerequisite to getting modifications approved.

3.1 AI/ML-based SaMD

Formally, the FDA defines SaMD as software intended to be used for one or more medical purposes without being part of a hardware medical device. An AI/ML-system is software that learns to perform tasks by tracking performance measures. A SaMD must be approved for a specific indication, which describes the population, disease, and intended use. Here we only consider SaMDs whose predictions do not change the observed outcome; We leave SaMDs that affect the observed outcome (e.g. by recommending treatment) to future work.

Predictive accuracy is typically characterized by multiple endpoints, or co-primary endpoints [26, 13]. The most common endpoints for binary classifiers are sensitivity and specificity because they tend to be independent of disease prevalence, which can vary across subpopulations [28]. Additionally, we can evaluate endpoints over different subgroups to guarantee a minimum level of accuracy for each one.

We now define a model developer (the manufacturer) in mathematical terms. Let $\mathcal{X}$ be the support of the targeted patient population, where a patient is represented by their covariate measurements. Let $\mathcal{Y}$ be output range (possibly multivariate). Let $\mathcal{Q}$ be a family of prediction models $f : \mathcal{X} \mapsto \mathcal{Y}$. Each model f defines the entire pipeline for calculating the SaMD output, including feature extraction, pre-processing steps, and how missing data is handled. The model developer is a functional g that maps the training data $(X_T, Y_T) \in \mathcal{X}^n \times \mathcal{Y}^n$ to a function in $\mathcal{Q}$. Let $\mathcal{P}$ be the family of distributions for $X \times Y$. The performance of a model f on population $\mathbb{P} \in \mathcal{P}$ is quantified by the K-dimensional endpoint $m : \mathcal{Q} \times \mathcal{P} \mapsto \mathbb{R}^K$. For each endpoint m_k, we assume that a larger value indicates better performance.

3.2 Modifications to AI/ML-based SaMD

The proposed workflow in FDA [15] for modifying an AI/ML-based SaMD iterates between three stages. First, the manufacturer proposes a modification by training on monitoring and/or external data and adds this to a pool of proposed modifications. Second, the aACP evaluates each candidate modification and grants approval to those satisfying some criteria.

The most recently approved version is then recommended to doctors and patients. Finally, a new batch of monitoring data is collected, which can be used to evaluate and train future models. For simplicity, suppose these three stages are executed in the above order over a fixed grid of time points $t = 1, 2, ...$

The model developer is allowed to propose arbitrary modifications in a sequential and possibly adaptive manner. For example, the modification can depend on all previously collected monitoring data as well as the set of approvals up to that time point. For generality, we represent each modification as an entirely separate model. Let filtration $\mathcal{F}_t$ be the sigma algebra representing the information up to time t, which includes observed monitoring data, proposed models, and aACP outputs up to time t. We now redefine the model developer in this online setting as a sequence of functionals $\{g_t : t = 1, 2, ...\}$, where g_t is a $\mathcal{F}_t$-measurable functional mapping to $\mathcal{Q}$. Let $\hat{f}_t$ be the realized model proposal at time t. In addition, suppose that each proposed model $\hat{f}_t$ has a maximum wait time Δ_t that specifies how long the manufacturer will wait for approval of this model, i.e. the model is no longer considered for approval after time $t + \Delta_t$.

Time trends are likely to occur in long-running processes, as found in long-running clinical trials and non-inferiority trials [3, 17]. This includes changes to any component of the joint distribution between the patient population and the outcome, such as the marginal distributions of the covariates, their correlation structure, their prognostic values, and the prevalence of the condition. As such, let the joint distribution at time t of patients X_t and outcomes Y_t be denoted $\mathbb{P}_t$. For endpoint m, its value for model f at time t is then $m(f, \mathbb{P}_t)$. More generally, we might characterize a model by the average endpoint value over time points t to $t + D - 1$ for some $D \geq 1$. We denote the average endpoint using $m(f, \mathbb{P}_{t:t+D-1})$, where $\mathbb{P}_{t:t'}$ indicates a uniform mixture of $\mathbb{P}_t, ..., \mathbb{P}_{t'}$. Here D acts as a smoothing parameter; Larger D increases the smoothness of endpoint values.

Finally, we assume that monitoring data collected at time t are representative of the current population $\mathbb{P}_t$. Of course, satisfying this criteria is itself a complex issue. We will

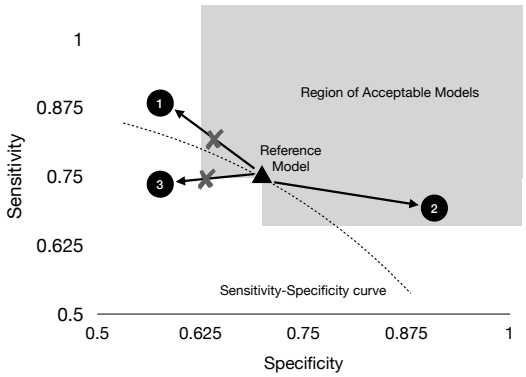


Figure 4.1: Example of an acceptability graph for binary classifiers evaluated on sensitivity and specificity. Given a reference model (triangle) and NI margin ϵ, a candidate model is acceptable if one endpoint is non-inferior and the other is superior compared to the reference model. The NI margin can be chosen to encourage approval of updates to a better ROC curve. Models in the shaded area are acceptable updates to the reference model. Model 3 is not acceptable since it is on a strictly inferior ROC curve. Model 1 and 2 are likely on better ROC curves, but 1 is not within the NI margin and is therefore not acceptable either.

not discuss the challenges here and instead refer the reader to Pepe [28] for more details, such as selecting an appropriate sampling scheme, measuring positive versus negative examples, and obtaining gold standard versus noisy labels.

Defining acceptable modifications

A fundamental building block for designing an aACP is defining when a modification is acceptable to a reference model. Our solution is to represent which modifications are acceptable using a directed graph between models in $\mathcal{Q}$, which we refer to as an "acceptability graph." That is, if there is a directed edge from model f to model f', then it is acceptable to update f to f'. Here, "acceptability" is defined for a pre-defined vector of non-inferiority (NI) margins $\epsilon \in \mathbb{R}_+^K$. An update from f to f' is acceptable if it demonstrates non-inferiority with respect to all endpoints and superiority in at least one [6, 7]. So for a binary classifier where the endpoints are sensitivity and specificity, one may select the NI margins to encour-

age modifications that shift the model to a better ROC curve (Figure 4.1). An acceptability graph is formally defined below:

Definition 4. *For a fixed evaluation window $D \in \mathbb{Z}^+$ and NI margin $\epsilon \in \mathbb{R}_+^K$, the acceptability graph at time t over $\mathcal{Q}$ contains the edge from f to f' if $m_k(f, \mathbb{P}_{t:t+D-1}) - \epsilon_k \leq m_k(f', \mathbb{P}_{t:t+D-1})$ for all $k = 1, ..., K$ and there is some $k = 1, ..., K$ such that $m_k(f', \mathbb{P}_{t:t+D-1}) > m_k(f, \mathbb{P}_{t:t+D-1})$. The existence of this edge is denoted $f \to_{\epsilon,D,t} f'$ and $f \not\to_{\epsilon,D,t} f'$ otherwise.*

Here, we assume D is fixed and use the notation $f \to_{\epsilon,t} f'$. For simplicity, Definition 4 uses the same NI margin across all models. In practice, it may be useful to let the margin depend on the reference model or the previously established limits of its predictive accuracy.

We obtain different graphs for different choices of ϵ. For instance, $\epsilon = 0$ means that a model is only acceptable if it is superior with respect to all endpoints, though this can be overly strict in some scenarios. Setting $\epsilon \neq 0$ is useful for approving modifications that maintain the value of some endpoints or have very small improvements with respect to some endpoints.

Finally, we define hypothesis tests based on the acceptability graph. In an ϵ-acceptability test, we test the null hypothesis that a model f' is not an ϵ-acceptable update to model f at time t, i.e. $H_0 : f \not\to_{\epsilon,t} f'$. A superiority test is simply an ϵ-acceptability test where $\epsilon = 0$.

4 An online hypothesis testing framework

At each time point, we suppose an aACP evaluates which candidates to approve by running a battery of hypothesis tests. Since AI/ML-based SaMDs can be modified more easily and frequently compared to drugs, the aACP may run a large number of tests. To account for the multiplicity of tests, we will frame aACPs as online hypothesis testing procedures where the goal is to control the error rate over a sequence of tests.

Each aACP specifies a sequence of approval functions A_t for times $t = 1, 2, ...$ (Fig-

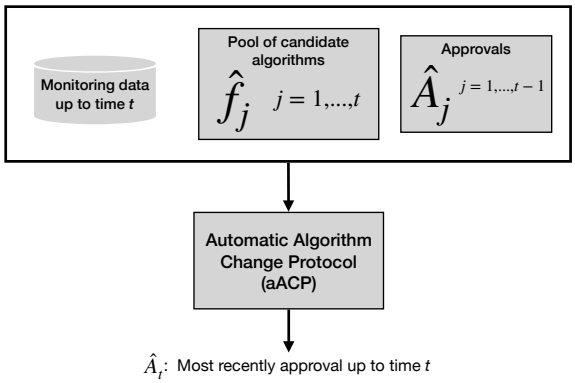


Figure 4.2: An automatic Algorithm Change Protocol (aACP) outputs the index of the most recently approved model $\hat{A}_t$ at each time t. To do so, it evaluates the pool of candidate models against the pool of previously approved models using monitoring data collected up to that time.

ure 4.2), where A_t is a $\tilde{\mathcal{F}}_t$-measurable function that outputs the index of the most recently approved model at time t (some value in $\{0, ..., t-1\}$). Filtration $\tilde{\mathcal{F}}_t$ is the sigma-algebra for monitoring data up to time t and proposed models and aACP outputs up to time $t-1$. The index of the latest approved model at time t is denoted $\hat{A}_t$. A model was approved at time t if $\hat{A}_t \neq \hat{A}_{t-1}$. Assuming companies are not interested in approving older models, we require $\hat{A}_t \geq \hat{A}_{t-1}$.

Different approval functions lead to different aACPs. We only consider aACPs that evaluate candidate modifications using prospectively-collected monitoring data, i.e. data collected *after* the candidate modification has been proposed, as candidate modifications can train on all previously-collected monitoring data. The following are two simple aACPs that one may plausibly consider but do not provide error-rate guarantees:

aACP-Baseline approves any modification that demonstrates ϵ-acceptability to the initially approved model at a fixed level α. This can be useful when the initial model has high predictive accuracy. The manufacturer may also argue this is reasonable policy because the current laws only require a model to perform better than placebo, i.e. the standard of care without utilizing AI/ML-based SaMDs.

aACP-Reset approves any modification that demonstrates ϵ-acceptability to the currently approved model at some fixed level α. As opposed to aACP-Baseline, this policy encourages the model to improve over time.

5 Online error rates for aACPs

We define two online Type I error rates for this setting and describe aACPs that uniformly control these error rates over time. Manufacturers and regulators should select the error rate definition and aACP most suitable for their purposes. These aACPs achieve error rate control as long as their individual hypothesis tests are controlled at their nominal levels.

For both definitions, the error rate at time T is evaluated over the window $1 \vee (T - W)$ to T for some width $W \geq 1$. The hyperparameter W must be pre-specified and specifies different trade-offs between error control and speed: $W = \infty$ requires the strongest error rate control, but is overly strict in most cases, and $W = 1$ requires the weakest error control, but can lead to bad long-running behavior. The desired trade-off is typically in between these extremes.

5.1 Bad approval count

We define a bad approval as one where the modification is unacceptable with regards to *any* of the previously approved models. The first error rate is defined as the expected Bad Approval Count (BAC) within the current window of width W:

Definition 5. *The expected bad approval count within the W-window at time T is*

$$\mathrm{BAC}_W(T) = E\left[\sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\exists t' = 1, ..., t-1 \ \textit{s.t.}\ \hat{f}_{\hat{A}_{t'}} \nrightarrow_{\epsilon,t} \hat{f}_{\hat{A}_t}\right\}\right].$$

This error rate captures two important ways errors can accumulate over time: bio-creep and the multiplicity of hypotheses. We discuss these two issues below.

When a sequence of NI trials is performed and the reference in each trial is the latest model that demonstrated NI, the performance of the approved models will gradually degrade over time; This phenomenon has been called bio-creep in previous work [17]. Bio-creep can also happen in our setting: Even if each approved model demonstrates superiority with respect to some endpoints and NI with respect to others, repeated applications of ϵ-acceptability tests can still lead to approval of strictly inferior models. The risk of bio-creep is particularly pronounced because the model developer can perform unblinded adaptations. To protect against bio-creep, Definition 5 counts it as a type of bad approval.

Second, when a long sequence of hypothesis tests is performed, the probability of a false rejection is inflated due to the multiplicity of hypotheses. Definition 5 accounts for multiplicity by summing the probabilities of bad approvals across the window. It is an upper bound for the probability of making any bad approval within the window, which is similar to the definition of family-wise error rate (FWER). In fact, we use the connection between FWER and BAC in the following section to design an aACP that controls this error rate.

aACP to control bad approval counts

We now present aACP-BAC, which uniformly controls $\text{BAC}_W(\cdot)$. An aACP is defined by its skeletal structure, which specifies the sequence of hypothesis tests run, and a procedure that selects the levels to perform the hypothesis tests. To build up to aACP-BAC, we i) first describe a simple aACP skeleton that launches a fixed sequence of group sequential tests (GSTs), ii) add gate-keeping to increase its flexibility, and iii) finally pair it with a sequence of $\tilde{\mathcal{F}}_t$-measurable functions $\{\alpha_t : t = 1, 2, ...\}$ for choosing the hypothesis test levels. The full algorithm is given in Algorithm 7 in the Appendix. For now, we assume the distributions are constant and simply use the notation $\to_\epsilon$ in place of $\to_{\epsilon,t}$. We discuss robustness to time trends in a later section.

Let us first consider a simple aACP skeleton that compares each proposed model to previously approved models using a single hypothesis test (Figure 4.3). More specifically,

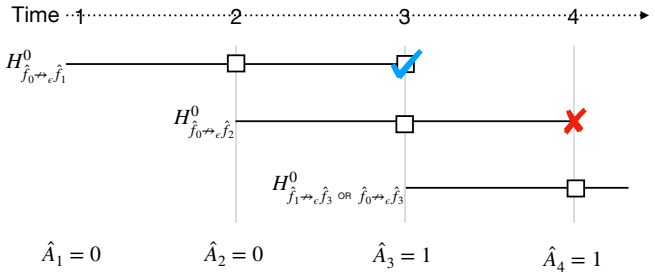


Figure 4.3: At each time point, this simple aACP launches a single group sequential test (GST) comparing the newly proposed model to previously approved models. Here, each model has a maximum wait time of $\Delta = 2$ and each interim analysis is represented by a square. A checkmark indicates that the null hypothesis is rejected and an "X" indicates that the interim analysis is not performed. The final interim analysis for $\hat{f}_2$ is not performed because its GST only compares $\hat{f}_2$ to $\hat{f}_0$ and not the newly approved model $\hat{f}_1$. Thus, $\hat{f}_2$ has no chance of being approved.

at time t, it launches a group sequential ϵ-acceptability test with the null hypothesis

$$H_0 : \exists t' = 1, ..., t \text{ s.t. } \hat{f}_{\hat{A}_{t'}} \not\rightarrow_\epsilon \hat{f}_t. \tag{4.1}$$

The number of interim analyses is the maximum wait time Δ_t and the critical values are chosen according to an alpha-spending function specified prior to launch [10]. At each time point, we also perform interim analyses for all active hypothesis tests (i.e. those not past their maximum wait time). The aACP approves $\hat{f}_j$ at time t if it demonstrates acceptability to $\hat{f}_{\hat{A}_1}, ..., \hat{f}_{\hat{A}_{t-1}}$. If multiple models are acceptable, it selects the latest one.

A drawback of this simple aACP skeleton is that it fails to adapt to new model approvals that occur in the middle of a group sequential test (GST). Consider the example in Figure 4.3, where a GST with null hypothesis $H^0_{\hat{f}_0 \not\rightarrow_\epsilon \hat{f}_1}$ is launched at time $t = 1$ and a second GST with null hypothesis $H^0_{\hat{f}_0 \not\rightarrow_\epsilon \hat{f}_2}$ is launched at time $t = 2$. If $\hat{f}_1$ is approved at time $t = 3$, this aACP cannot approve $\hat{f}_2$ since its GST only compares $\hat{f}_2$ to $\hat{f}_0$. Ideally, it could adapt to the new approval and add a test comparing $\hat{f}_2$ to $\hat{f}_1$.

aACP-BAC addresses this issue by evaluating proposed model $\hat{f}_t$ using a *family* of ac-

ceptability tests instead (Figure 4.4). In addition to the aforementioned test for the null hypothesis (4.1), this family includes acceptability tests to test each of the null hypotheses

$$H_{0,j} : \hat{f}_j \nrightarrow_{\epsilon} \hat{f}_t \text{ for } j = \hat{A}_t + 1, ..., t - 1. \tag{4.2}$$

As before, a model is approved at time t only if it demonstrates acceptability compared to all approved models up to time t. To control the online error rate, aACP-BAC controls the FWER for each family of tests using a serial gate-keeping procedure. Recall that gate-keeping tests hypotheses in a pre-specified order and stops once it fails to reject a null hypothesis [11]. No alpha adjustment is needed in gate-keeping; It controls FWER at α by performing all tests at level α. Here, the tests are naturally ordered by the index of the reference models, from oldest to latest. Moreover, this ordering maximizes the probability of approval, assuming the proposed models improve in predictive accuracy. We use the overall hierarchical rule to perform GSTs with gate-keeping Tamhane et al. [33].

To uniformly control $\text{BAC}_W(\cdot)$ at α, aACP-BAC computes an over-estimate of $\text{BAC}_W(t)$ at each time t and selects level $\hat{\alpha}_t$ such that the over-estimate is bounded by α. Using a union bound like that in Bonferroni correction, it uses the over-estimate

$$\widehat{\text{BAC}}_W(t) = \sum_{t'=1}^{t} \hat{\alpha}_{t'} \mathbb{1}\left\{t - W \leq t' + \Delta_{t'} \leq t\right\} \tag{4.3}$$

and selects $\hat{\alpha}_t$ such that

$$\widehat{\text{BAC}}_W(t) \leq \alpha. \tag{4.4}$$

See supporting information for a proof that aACP-BAC achieves the nominal rate.

Alternatively, we can think of aACP-BAC as an alpha-investing procedure [18] that begins with an alpha-wealth of α, spends it when a family of tests is launched, and earns it back when the family leaves the current window. Thus, aACPs that control BAC over

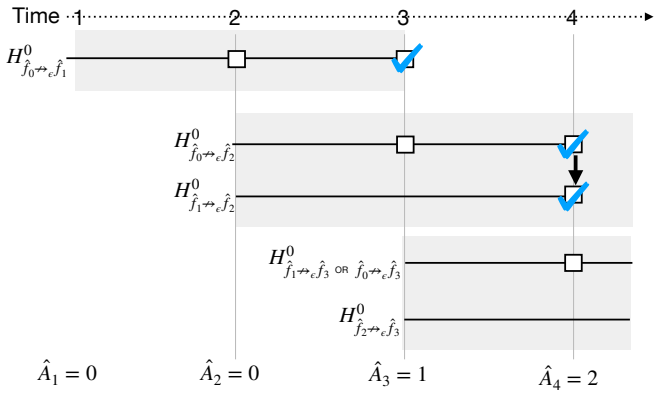


Figure 4.4: At each time point, this aACP launches a family of group sequential tests (shaded gray boxes) comparing the newly proposed model to previously approved models as well as other models that might be approved in the interim. Within each family, we test the hypotheses using a gatekeeping procedure, which provides a mechanism for comparing a candidate model to newly approved models in the interim. We use the same notation in Figure 4.3. An arrow between squares indicates that we rejected a null hypothesis and proceeded to the next test in the gatekeeping sequence.

window size $W = \infty$ inevitably have low power to approve later modifications because they only spend but never earn alpha-wealth. This is analogous to the so-called "alpha-death" issue that occurs in procedures that control online FWER [30]. We sidestep the issue of alpha-death by selecting a reasonable value for W.

5.2 Bad approval and benchmark ratios

If the goal is to ensure that the SaMD improves on average and occasional drops in performance are tolerated, the approval policies for controlling BAC_W can be overly strict and unnecessarily conservative. There are two solutions to this problem. One approach (*reward-approach*) is to reward the company for each superior model by resetting the level alpha. The FDA essentially uses this procedure right now, as each clinical trial resets the alpha-spending clock. Another idea (*FDR-approach*) is to draw on the false discovery rate (FDR) literature: These procedures control the expected proportion of false rejections rather than the FWER, which has higher power when some of the null hypotheses are false [5]. This

section defines a second online error rate based on these ideas.

The *reward-approach* punishes bad approvals and rewards the approval of superior models. To signify that a recent set of modifications has led to the creation of a superior model, we now define aACPs with an additional function that can label models as "benchmarks." By labeling a model as a benchmark, the aACP is claiming that it is *superior* to the previous benchmark. More formally, we define a benchmark function B_t as a $\tilde{F}_t$ measurable function that outputs the index of the latest benchmark model at time t. For $t = 0$, we have $B_0 \equiv 0$. We require benchmarks to be a previously approved model since superiority implies acceptability. Again, we use the hat notation to indicate the realized benchmark index. A bad benchmark is one in which $\hat{f}_{\hat{B}_{t-1}} \nrightarrow_{0,t} \hat{f}_{\hat{B}_t}$. We do not compare against all previous benchmarks since $\nrightarrow_{0,t}$ is a transitive property when the superiority graph is constant.

Based on the *FDR-approach*, we now introduce bad approval and benchmark ratios. An aACP needs to control both ratios to control the frequency of bad approvals and benchmarks.

Definition 6. *For NI margin ϵ, the bad approval ratio within W-window at time T is*

$$\mathrm{BAR}_W(T) = \frac{\sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\exists t' = 1, ..., t-1 \ s.t. \ \hat{f}_{\hat{A}_{t'}} \nrightarrow_{\epsilon,t} \hat{f}_{\hat{A}_t}\right\}}{1 + \sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\hat{B}_t \neq \hat{B}_{t-1}\right\}}. \tag{4.5}$$

The bad benchmark ratio within W-window at time T is

$$\mathrm{BBR}_W(T) = \frac{\sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\hat{f}_{\hat{B}_{t-1}} \nrightarrow_{0,t} \hat{f}_{\hat{B}_t}\right\}}{1 + \sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\hat{B}_t \neq \hat{B}_{t-1}\right\}}. \tag{4.6}$$

Since only approved models can be designated as benchmarks, BAR_W is an upper bound for the proportion of bad approvals (this is approximate because the denominator is off by one).

The denominator in (4.5) was deliberately chosen to be the number of unique benchmarks rather than the number of approvals because the latter is easy to inflate artificially. We

can simply propose models by alternating between two models that are ϵ-acceptable to each other. This strategy does not work for benchmarks because they require demonstrating superiority.

aACP to control bad approval and benchmark ratios

Instead of controlling the expectations of (4.5) and (4.6), we describe aACP-BABR for controlling the modified expected bad approval and benchmark ratios. These modified ratios are based on a similar quantity in the online FDR literature known as modified online FDR [18]. We chose to control the modified versions because they can be controlled under less restrictive conditions and using relatively intuitive techniques [30]. Moreover, Foster and Stine [18] found that modified online FDR has similar long-running behavior to online FDR. We define modified expected bad approval and benchmark ratios below.

Definition 7. *For NI margin ϵ, the modified expected bad approval ratio within W-window at time T is*

$$\text{meBAR}_W(T) = \frac{E\left[\sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\exists t' = 1, ..., t-1 \ s.t. \ \hat{f}_{\hat{A}_{t'}} \nrightarrow_{\epsilon,t} \hat{f}_{\hat{A}_t}\right\}\right]}{E\left[1 + \sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\hat{B}_t \neq \hat{B}_{t-1}\right\}\right]}. \tag{4.7}$$

The modified expected bad benchmark ratio within W-window at time T is

$$\text{meBBR}_W(T) = \frac{E\left[\sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\hat{f}_{\hat{B}_{t-1}} \nrightarrow_{0,t} \hat{f}_{\hat{B}_t}\right\}\right]}{E\left[1 + \sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\hat{B}_t \neq \hat{B}_{t-1}\right\}\right]}. \tag{4.8}$$

Next, we describe how aACP-BABR uniformly controls $\text{meBAR}_W(\cdot)$ and $\text{meBBR}_W(\cdot)$ at levels α and α', respectively (Algorithm 8). We begin with its skeleton and then discuss the alpha-investing procedure. Again, we assume the distributions $\mathbb{P}_t$ are constant.

aACP-BABR uses the acceptability tests from aACP-BAC to approve modifications and superiority tests to discover benchmarks. So at time t, in addition to launching a family of acceptability tests to evaluate model $\hat{f}_t$ for approval, aACP-BABR also launches a family

of group-sequential superiority tests comparing $\hat{f}_t$ to models with indices $\left\{\hat{B}_{t-1}, ..., t-1\right\}$, which are executed in a gate-keeping fashion from oldest to latest. Let Δ'_t be the maximum wait time for the superiority tests, which can differ from the maximum wait time for acceptability tests. A model $\hat{f}_j$ is designated as a new benchmark at time t if it demonstrates superiority to models $\hat{f}_{\hat{B}_{j-1}}, ..., \hat{f}_{\hat{B}_{t-1}}$. If multiple benchmarks are discovered at the same time, the aACP can choose any of them (we choose the oldest one in our implementation).

aACP-BABR uses an alpha-investing procedure based on Ramdas et al. [30] to control the error rates. Let the $\tilde{F}_t$-measurable function α'_t specify the level to perform superiority tests launched at time t. At time t, aACP-BABR constructs over-estimates of the error rates $\mathrm{BAR}_W(t)$ and $\mathrm{BBR}_W(t)$ and selects $\hat{\alpha}_t$ and $\hat{\alpha}_{t'}$ such that the over-estimates are no larger than the nominal levels. The over-estimates are

$$\widehat{\mathrm{BAR}}_W(t) = \frac{\sum_{t'=1}^{t} \hat{\alpha}_{t'} \mathbb{1}\left\{t - W \leq t' + \Delta_{t'} \leq t\right\}}{1 + \sum_{t'=1 \vee (t-W)}^{t} \mathbb{1}\{\hat{B}_{t'} \neq \hat{B}_{t'-1}\}} \tag{4.9}$$

$$\widehat{\mathrm{BBR}}_W(t) = \frac{\sum_{t'=1}^{t} \hat{\alpha}'_{t'} \mathbb{1}\left\{t - W \leq t' + \Delta'_{t'} \leq t\right\}}{1 + \sum_{t'=1 \vee (t-W)}^{t} \mathbb{1}\{\hat{B}_{t'} \neq \hat{B}_{t'-1}\}}. \tag{4.10}$$

It selects $\hat{\alpha}_t$ and $\hat{\alpha}'_t$ such that

$$\widehat{\mathrm{BAR}}_j(t) \leq \alpha \quad \forall j = 1, ..., W \tag{4.11}$$

$$\widehat{\mathrm{BBR}}_j(t) \leq \alpha' \quad \forall j = 1, ..., W. \tag{4.12}$$

(We consider all window sizes since we also need to over-estimate future errors $\mathrm{BAR}_W(t')$ and $\mathrm{BBR}_W(t')$ for $t' > t$.) So, aACP-BABR earns alpha-wealth when new benchmarks are discovered, which unites ideas from *FDR-approach* and *reward-approach*. See the supporting information for a proof that aACP-BABR provides the desired error control.

5.3 Effect of time trends

Time trends are likely to occur when an aACP is run for a long time. We now discuss how robust aACP-BAC and -BABR are to time trends. We consider levels of increasing severity: the distributions are relatively constant over time (*no-trend*), the distributions are variable but the acceptability graphs are relatively constant (*graph-constant*), and the acceptability graphs change frequently (*graph-changing*).

When the distributions are relatively constant over time, aACP-BAC and -BABR should approximately achieve their nominal error rates. Recall that the two aACPs perform paired T-tests by approximating the distribution $\mathbb{P}_{t:t+D-1}$ with monitoring data sampled from $\mathbb{P}_{j\vee j':t-1}$, which is reasonable when the distributions are relatively constant over time. When there are multiple endpoints, one can either choose a GST that rejects the null hypothesis when all endpoints surpass the significance threshold at the same interim time point or when the endpoints surpass their respective thresholds at any interim timepoint. The former approach is more robust to time trends with only modest differences in power [4].

When the distributions are not constant but the acceptability graphs are, the GSTs have inflated error rates since they only guarantee Type I error control under the strong null. To handle heterogeneity in distributions over time, we can instead use combination tests, such as Fisher's product test and the inverse normal combination test, to aggregate results across time points [16, 22]. Since we assumed that the acceptability graphs are constant, this tests the null hypothesis that the shared acceptability graph does not have a particular edge (i.e. $H_0 : f \nrightarrow_{\epsilon,\cdot} f'$); The alternative hypothesis is that the edge exists. Thus, we can replace GSTs with combination tests to achieve the desired error control.

The most severe time trend is where the acceptability graphs change frequently. Controlling error rates in this setting is difficult because previous data is not informative for future time points. In fact, even bad approvals are not well-defined since the relative performance of models changes over time, e.g. an approval at time t that looks bad at time $t+1$ might turn out to be a very good at time $t+2$. As such, we recommend checking

that the acceptability graphs are reasonably constant to ensure proper use of aACP-BAC and -BABR. For example, one could track a moving average for the evaluation metrics of all previously proposed modifications and check that their values and/or their relative orderings are stable over time.

6 Cumulative utility of an aACP

Just as hypothesis tests are judged by their Type I error and power, aACPs should be judged by their rates for approving bad and good modifications. Taking a decision-theoretic approach, we characterize the rate of good approvals as the cumulative mean of an endpoint, which we refer to as "cumulative utility." This quantity is similar to "regret" in the online learning literature [32].

Definition 8. *The cumulative utility of an aACP with respect to endpoint m is*

$$E\left[\frac{1}{T}\sum_{t=1}^{T} m\left(\hat{f}_{\hat{A}_t}, \mathbb{P}_t\right)\right]. \tag{4.13}$$

There is no single aACP that maximizes (4.13) for all possible model developers since we allow arbitrary unblinded adaptations. Instead, we suggest running simulation studies under probable model improvement rates to understand the cumulative utility under different aACP settings, such as window size W, NI margin ϵ, and monitoring data batch size.

7 Simulations

Through simulation studies, we evaluate the operating characteristics of the following aACPs:

1. Blind: Approve all model updates
2. Reset: Perform an acceptability test at level 0.05 against the last-approved model
3. Baseline: Perform an acceptability test at level 0.05 against the initial model
4. aACP-BAC at level $\alpha = 0.2$ with window $W = 15$

5. aACP-BABR at level $\alpha = \alpha' = 0.2$ with window $W = 15$. The ratio of maximum wait times between the benchmark and approval was fixed at $\Delta'/\Delta = 2$.

6. Fixed: Only approve the first model

The first three aACPs have no error rate guarantees but are policies one may consider; The others provide error rate control. In the first two simulations, we try to inflate the error rates of the aACPs. The next two study the cumulative utility of the aACPs when proposed models are improving on average. The last simulation explores the effects of time trends.

In the simulations below, we consider a binary classification task with sensitivity and specificity as metrics. We compare aACPs by plotting the metrics of the approved model over time. We test for acceptability/superiority using repeated confidence intervals [9] and Pocock alpha-spending functions [29], where $\epsilon = 0.05$ for both endpoints. The Appendix contains summary statistics of aACP performance (Table 4.1 and 4.2), simulation details (Section A.2), and sensitivity analyses to hyperparameters W and ϵ (Section A.3).

7.1 Incremental model deterioration

In this simulation, the proposed models deteriorate gradually. This can occur in practice for a number of reasons. For instance, a manufacturer might try to make their SaMD simpler, cheaper, and/or more interpretable by using fewer input variables, collecting measurements through other means, or training a less complex model. Even if their modifications are well-intended, the sponsor might end up submitting inferior models. A model developer can also inadvertently propose adverse modifications if they repeatedly overfit to the training data. Finally, a properly trained model can be inferior if the training data is not representative of future time points if, say, the biomarkers lose their prognostic value over time.

This simulation setup tries to induce bio-creep by submitting models that are acceptable to the currently approved model but gradually deteriorate over time. Each proposed model is worse by $\epsilon/2$ in one endpoint and better by $\epsilon/4$ in the other. By alternating be-

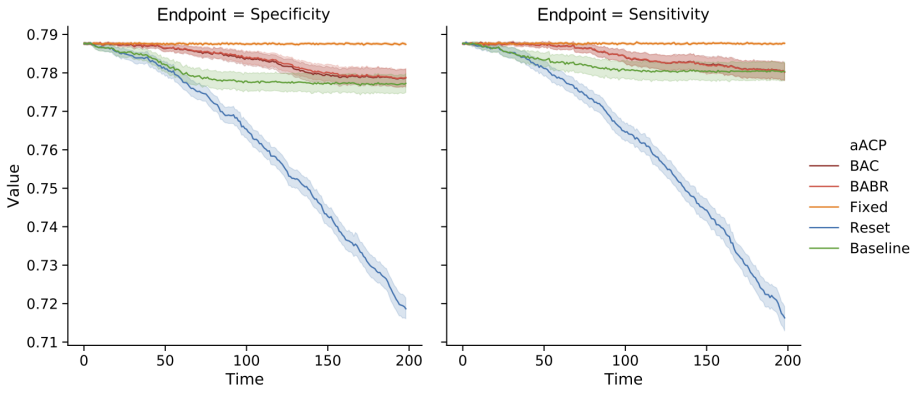


Figure 4.5: Comparison of the sensitivity and specificity of models approved by different aACPs when the proposed models are gradually deteriorating. (We omit aACP-Blind from this plot since it would obviously perform the worst.)

tween deteriorating the two endpoints, the manufacturer eventually submits strictly inferior models.

Bio-creep occurs consistently when using the aACP-Reset since it only compares against the most recently approved model (Figure 4.5). Both the sensitivity and specificity for the approved model at the final time point are significantly worse than the initial model. aACP-BAC and aACP-BABR properly controlled the occurence of bio-creep since they require modifications to demonstrate acceptability with respect to *all* previously approved models.

7.2 Periodic model deterioration and improvement

Next we consider a simulation in which the proposed modifications periodically decline and improve in performance. This scenario is more realistic than the previous section since a manufacturer is unlikely to only submit bad modifications. More specifically, the proposed models monotonically improve in performance over the first fifteen time points and, thereafter, alternate between deteriorating and improving monotonically every ten time points.

As expected, aACP-Baseline had the worst error and cumulative utility. It performed like aACP-Blind and the performances of the approved models were highly variable over

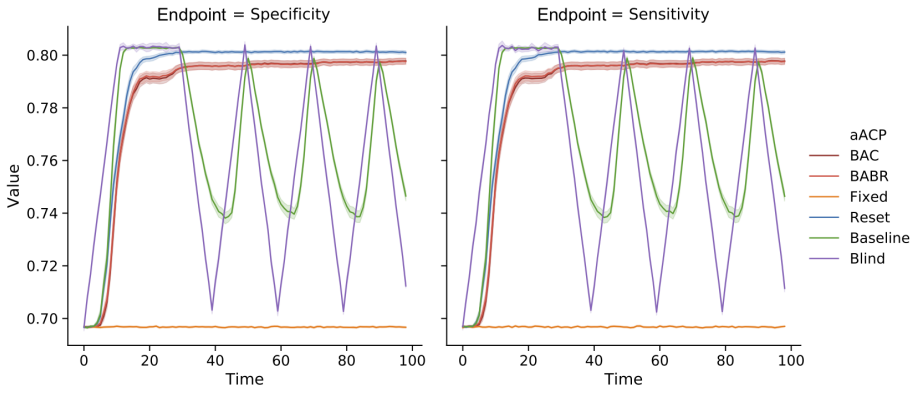


Figure 4.6: Comparison of the sensitivity and specificity of models approved by different aACPs when the proposed models periodically deteriorate and improve in performance.

time (Figure 4.6). In contrast, the other aACPs displayed much less variability and the performances were generally monotonically increasing. aACP-Reset had the highest utility here because it performs hypothesis tests at a higher level alpha than aACP-BAC and aACP-BABR.

7.3 Accumulating data model updates

We now suppose the manufacturer automatically generates modifications by training the same model on accumulating monitoring data. In this simulation, the developer iteratively performs penalized logistic regression. Since model parameters are estimated with increasing precision, the expected improvement decreases over time and performance eventually plateaus. As such, we investigate aACP behavior over a shorter time period.

aACP-Blind approved good modifications the fastest (Figure 4.7). We find that the remaining aACPs, excluding aACP-fixed, are close in cumulative utility because less efficient aACPs can "catch up": Even if an aACP fails to approve a small improvement, there will eventually be a proposal with sufficiently large improvement that is easy to discern. aACP-BABR often discovered one or no new benchmarks within a window and was unable to earn alpha-wealth much of the time because the models improved at a slow pace and performance

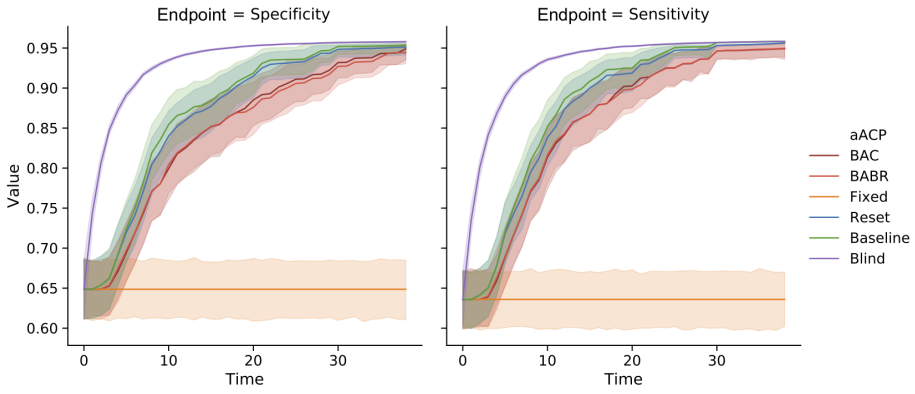


Figure 4.7: Comparison of the sensitivity and specificity of models approved by different aACPs when the model developer trains a logistic model on accumulating monitoring data.

plateaued over time. As such, aACP-BAC and -BABR behaved similarly in this simulation.

7.4 Significant model improvements

Next, we simulate a manufacturer that proposes models with large improvements in performance at each time point. Large improvements usually occur when the modifications significantly change the model, such as adding a highly informative biomarker or replacing a simple linear model with a complex one that accounts for non-linearities and interactions.

Since large improvements are relatively rare, we used a short total time. We designed the simulation to be less favorable for aACP-BAC and -BABR. The model developer proposes a modification that improves both endpoints by 4% compared to the most recently approved model. Therefore an aACP cannot catch up by simply waiting for large improvements.

As expected, Blind-aACP is the most efficient, followed by aACP-Baseline and aACP-Reset (Figure 4.8). The aACPs with error rate control are less efficient. For example, the performance of the final models approved by aACP-BABR and aACP-Reset differed by 4% on average. Unlike in previous simulations, there is a clear improvement in efficiency from using aACP-BABR over aACP-BAP. Since the models here improve at a fast pace, aACP-BABR earns enough alpha-wealth to discover new benchmarks with high probability.

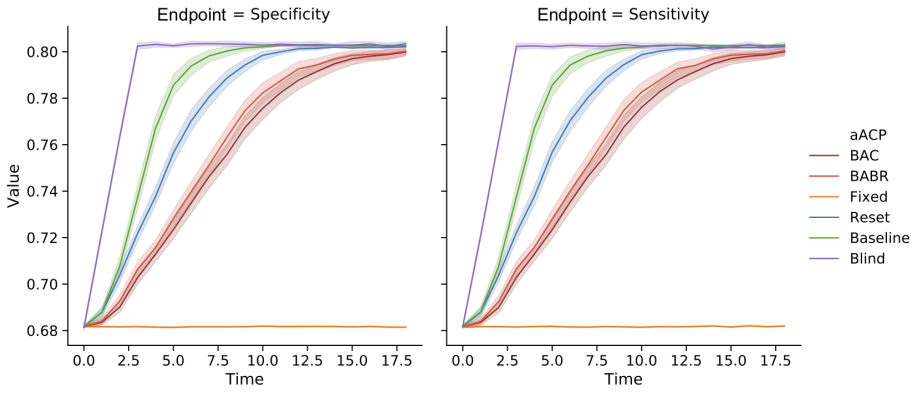


Figure 4.8: Comparison of the sensitivity and specificity of models approved by different aACPs when the model developer adaptively proposes a significantly better model than the currently approved model. We evaluate three different settings for aACP-BABR, where a larger index means that alpha-wealth is spent more greedily.

7.5 Robustness to time trends

Finally, we evaluate the robustness of aACP-BAC and -BABR to time trends by simulating the three time trend severity levels from Section 5.3. We simulate the endpoints of the proposed models to follow a sinusoidal curve. In addition, the proposed model is always strictly inferior to the currently approved model on average. For the *graph-constant* setting, the sinusoids are aligned so that the proposed model is unacceptable at all time points. For the *graph-changing* setting, the sinusoids are offset by exactly half the period so that the proposed model is superior to the currently approved model at certain time points.

The error rates in the *graph-constant* and *no-trend* settings were similar (Table 4.2 in the Appendix), which implies that aACP-BAC and -BABR still control error rates if the acceptability graphs stay relatively constant. However, they performed poorly in the *graph-changing* setting since bad modifications appeared superior at particular time points.

8 Discussion

In this work, we have presented and evaluated different policies for regulating modifications to AI/ML-based SaMDs. One of our motivations was to investigate the possibility of bio-

creep, due to the parallels between this problem and noninferiority testing of new drugs. We found that the risk of bio-creep is heightened in this regulatory problem compared to the traditional drug development setting because software modifications are easy and fast to deploy. Nonetheless, we show that aACPs with appropriate online error-rate guarantees can sufficiently reduce the possibility of bio-creep without substantial sacrifices in our ability to approve beneficial modifications, at least in the specific settings discussed in this chapter.

We have only considered a limited scope of problems and there are still many interesting directions for future work. One direction is to develop more efficient aACPs, perhaps by spending alpha-wealth more judiciously, discovering benchmarks using a different procedure, sequestering monitoring data for repeated testing, or considering the special case with pre-specified modifications. Also, we have not considered aACPs that regulate modifications to SaMDs that are intended to treat and are evaluated based on patient outcomes.

Our results raise the interesting question regarding the general structure of the regulatory policy framework. Although aACP-BAC and -BABR mitigate the effect of bio-creep, they cannot provide indefinite error rate control without large sacrifices in cumulative utility. So if one desires both indefinite error rate control and fast approval of good modifications, perhaps the solution is not to use a fully automated approach. For example, human experts could perform comprehensive analyses every couple of years and the manufacturer could use an aACP in between to quickly deploy modifications.

Finally, we highlight that regulating modifications to AI/ML-based SaMDs is a highly complex problem. We have primarily focused on the idealized setting of a constant diagnostic environment. Our findings suggest that problems with bio-creep is more pervasive when modifications are designed to accommodate time trends in the patient population, available measurements, and bioclinical practice. It is crucial that we thoroughly understand the safety risks before allowing modifications in these more complex settings.

A Appendix

A.1 Proofs

Theorem 4. *aACP-BAC achieves uniform control of* $\mathrm{BAC}_W(\cdot)$ *at level* α*, i.e.*

$$\mathrm{BAC}_W(T) \leq \alpha \qquad T = 1, 2, ... \tag{4.14}$$

Proof. At each time point, aACP-BAC launches a set of hypothesis tests comparing $\hat{f}_t$ to models with indices $\hat{M}_t = \{\hat{A}_1, ..., \hat{A}_t, \hat{A}_t + 1, ..., t-1\}$. Let the $\tilde{\mathcal{F}}_t$-measurable random variable G_t indicate the indices of the true null hypotheses, i.e.

$$\hat{G}_t = \left\{ j \in \hat{M}_t : \hat{f}_j \nrightarrow_\epsilon \hat{f}_t \right\}.$$

It is easy to see that the number of bad approvals is upper bounded by the number of incorrect rejections of the launched null hypotheses, i.e.

$$\begin{aligned} &\sum_{1 \vee (T-W)}^{T} \mathbb{1}\left\{ \exists t' = 1, \cdots, t-1 \text{ s.t. } \hat{f}_{\hat{A}_{t'}} \nrightarrow_\epsilon \hat{f}_{\hat{A}_t} \right\} \\ \leq &\sum_{1 \vee (T-W)}^{T} \mathbb{1}\left\{ \exists j \in \hat{G}_t, \exists t' = 1, \cdots, \Delta_t, \text{ s.t. reject } \hat{f}_j \nrightarrow_\epsilon \hat{f}_t \text{ at time } t + t' \right\}. \end{aligned} \tag{4.15}$$

Taking the expectations on both sides, $\mathrm{BAC}_W(T)$ is upper-bounded by

$$\sum_{1 \vee (T-W)}^{T} \Pr\left(\exists j \in \hat{G}_t, \exists t' = 1, \cdots, \Delta_t, \text{ s.t. reject } \hat{f}_j \nrightarrow_\epsilon \hat{f}_t \text{ at time } t + t' \right). \tag{4.16}$$

Since the hypothesis tests are tested using a gatekeeping procedure, each probability in (4.16) is equal to the probability of rejecting the first true null hypothesis in the gatekeeping

sequence. Thus,

$$\Pr\left(\exists j \in \hat{G}_t, \exists t' = 1, \cdots, \Delta_t, \text{ s.t. reject } \hat{f}_j \nrightarrow_\epsilon \hat{f}_t \text{ at time } t+t'\right) \tag{4.17}$$

$$= \Pr\left(\hat{G}_t \neq \emptyset, \exists t' = 1, \cdots, \Delta_t, \text{ s.t. reject } \hat{f}_{\min \hat{G}_t} \nrightarrow_\epsilon \hat{f}_t \text{ at time } t+t'\right) \tag{4.18}$$

$$\leq E\left[\Pr\left(\hat{G}_t \neq \emptyset, \exists t' = 1, \cdots, \Delta_t, \text{ s.t. reject } \hat{f}_{\min \hat{G}_t} \nrightarrow_\epsilon \hat{f}_t \text{ at time } t+t' \mid \tilde{\mathcal{F}}_t\right)\right] \tag{4.19}$$

$$\leq E\left[\hat{\alpha}_t \mathbb{1}\left\{\hat{G}_t \neq \emptyset\right\}\right]. \tag{4.20}$$

Summing together the probabilities within the window, we have

$$\mathrm{BAC}_W(T) \leq E\left[\sum_{1 \vee (T-W)}^{T} \hat{\alpha}_t\right] \leq \alpha, \tag{4.21}$$

where the last inequality follows from the fact that $\hat{\alpha}_t$ is always selected such that

$$\sum_{1 \vee (T-W)}^{T} \hat{\alpha}_t \leq \alpha.$$

□

Theorem 5. *aACP-BABR achieves uniform control of* $\mathrm{meBAR}_W(\cdot)$ *and* $\mathrm{meBBR}_W(\cdot)$ *at levels* α *and* α'*, respectively, i.e.*

$$\mathrm{meBAR}_W(T) \leq \alpha \qquad \forall T = 1, 2, \cdots \tag{4.22}$$

$$\mathrm{meBBR}_W(T) \leq \alpha' \qquad \forall T = 1, 2, \cdots \tag{4.23}$$

Proof. For all T, $\hat{\alpha}_T$ is selected such that

$$\hat{\mathrm{BAR}}_{W'}(T) = \frac{\sum_{t=1}^{T} \hat{\alpha}_t \mathbb{1}\left\{t - W \leq t' + \Delta_{t'} \leq t\right\}}{1 + \sum_{t=1 \vee (T-W')}^{T-1} \mathbb{1}\left\{\hat{B}_{t-1} \neq \hat{B}_t\right\}} \leq \alpha \qquad \forall W' = 1, ..., W. \tag{4.24}$$

Note that we can always set $\hat{\alpha}_T = 0$ to satisfy these constraints, assuming that (4.11) was satisfied at times $t = 1, ..., T-1$. Using the result in the proof of Theorem 4, we then bound the numerator of $\text{BAR}_W(T)$ as follows

$$E\left[\sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\exists t' = 1, ..., t-1 \text{ s.t. } \hat{f}_{\hat{A}_{t'}} \nrightarrow_{\epsilon,t} \hat{f}_{\hat{A}_t}\right\}\right] \tag{4.25}$$

$$\leq E\left[\sum_{t=1}^{T} \hat{\alpha}_t \mathbb{1}\left\{t - W \leq t' + \Delta_{t'} \leq t\right\}\right] \tag{4.26}$$

$$\leq E\left[\alpha\left(1 + \sum_{t=1\vee(T-W)}^{T} \mathbb{1}\left\{\hat{B}_{t-1} \neq \hat{B}_t\right\}\right)\right], \tag{4.27}$$

where the last line follows from (4.24). Rearranging, we get that $\text{meBAR}_W(T) \leq \alpha$. The proof for uniform control of $\text{meBBR}_W(\cdot)$ is essentially the same, where we replace the alpha-spending function with α'_t and the threshold with α'. □

Algorithm 7 aACP-BAC

for $t = 1, 2, ...$ **do**
  $\hat{A}_t = \hat{A}_{t-1}$ {Determine if there are new approvals}
  for $j = \hat{A}_{t-1} + 1, ..., t-1$ **do**
    if $t \leq j + \Delta_j$ **then**
      Run ϵ-acceptability tests: Test null hypotheses $\hat{f}_{j'} \nrightarrow_{\epsilon} \hat{f}_j$ for $j' = \hat{A}_1, ..., \hat{A}_{j-1}, \hat{A}_{j-1} + 1, ..., \hat{A}_{t-1}$ with critical value $c_j(t)$ in gatekeeping style
    end if
    if All ϵ-acceptability tests pass **then**
      $\hat{A}_t = j$
    end if
  end for{Launch new hypothesis tests for new model proposal}
  Launch family of ϵ-acceptability tests with null hypotheses $\hat{f}_j \nrightarrow_{\epsilon} \hat{f}_t$ for $j = \hat{A}_1, ...\hat{A}_t, \hat{A}_t + 1, ..., t-1$ Choose $\hat{\alpha}_t$ such that (4.4) is satisfied Select alpha-spending function for $\hat{\alpha}_t$ and its critical value function $c_t(\cdot)$ over the next Δ_t time points.
end for

Algorithm 8 aACP-BABR

for $t = 1, 2, ...$ **do**
  $\hat{A}_t = \hat{A}_{t-1}$ {Determine if there are new approvals}
  for $j = \hat{A}_{t-1} + 1, ..., t-1$ **do**
    if $t \leq j + \Delta_j$ **then**
      Run ϵ-acceptability tests: Test null hypotheses $\hat{f}_{j'} \nrightarrow_{\epsilon} \hat{f}_j$ for $j' = \hat{A}_j, ..., \hat{A}_{t-1}$ with critical value $c_j(t)$ in gatekeeping style
    end if
    if All ϵ-acceptability tests pass **then**
      $\hat{A}_t = j$
    end if
  end for
  $\hat{B}_t = \hat{B}_{t-1}$ {Determine if there are new benchmarks}
  for $j = \hat{A}_1, ..., \hat{A}_{t-1}$ **do**
    if $j > \hat{B}_{t-1}$ and $t \leq j + \Delta_j$ **then**
      Run superiority tests: Test null hypotheses $\hat{f}_{j'} \nrightarrow_{0} \hat{f}_j$ for $j' = \hat{B}_j, ..., \hat{B}_{t-1}$ with critical value $c'_j(t)$ in gatekeeping style
    end if
    if All superiority tests pass **then**
      $\hat{A}_t = j$
    end if
  end for
  {Launch new hypothesis tests for new model proposal}
  Launch family of ϵ-acceptability tests with null hypotheses $\hat{f}_j \nrightarrow_{\epsilon} \hat{f}_t$ for $j = \hat{A}_1, ...\hat{A}_t, \hat{A}_t+1, ..., t-1$ Launch family of superiority tests with null hypotheses $\hat{f}_j \nrightarrow_{\epsilon} \hat{f}_t$ for $j = \hat{B}_t, ..., t-1$ Choose $\hat{\alpha}_t, \hat{\alpha}'_t$ such that (4.11) and (4.12) are satisfied Select alpha-spending function for $\hat{\alpha}_t$ and $\hat{\alpha}'_t$ and their critical value functions $c_t(\cdot), c'_t(\cdot)$ over the next Δ_t time points.
end for

A.2 Simulation settings

We ran 200 replicates for each simulation.

Hypothesis testing procedure for acceptability

All the aACPs tested for acceptability of a modification from f to f' with null hypothesis $H_0 : f \to_\epsilon f'$ in the following manner. Let the true difference in sensitivities and specificities be denoted (θ_1, θ_2). At each stage of the group sequential test, we construct rectangular confidence regions for the evaluation metrics using confidence intervals for each metric [9]. At any stage, if the rectangular confidence region is completely within the region of acceptable modifications as defined by the NI margin ϵ, then we reject the null hypothesis. For simplicity, we use Pocock's alpha-spending function to determine the confidence levels at each stage [29]. Because our estimates for θ_1 and θ_2 are independent conditional on the number of negative and positive samples, we can control the Type I error of testing $H_0 : f \to_\epsilon f'$ at level α by using the significance thresholds from level $1 - \sqrt{1-\alpha}$ group sequential tests for the individual metrics. It is easy to see why this works:

$$\Pr(\text{Confidence region fails to cover } (\theta_1, \theta_2) \text{ at some stage}) \tag{4.28}$$

$$= 1 - \Pr(\text{Confidence region covers } (\theta_1, \theta_2) \text{ at all stages}) \tag{4.29}$$

$$= 1 - \Pr(\text{CI covers } \theta_1 \text{ at all stages}) \Pr(\text{CI covers } \theta_2 \text{ at all stages}) \tag{4.30}$$

$$= \alpha \tag{4.31}$$

We test for superiority by setting $\epsilon = 0$.

Incremental deterioration

We set total time $T = 200$ and the maximum wait time $\Delta = 5$ for all models. The number of new monitoring observations at each time point increments by ten to estimate the true performance difference with increasing precision over time, starting with 200 observations.

aACP	BAC_W	# approved	Final Specificity	Final Sensitivity	Cumulative Utility Specificity	Cumulative Utility Sensitivity	meBAR_W	meBBR_W
Incremental model deterioration, Section 7.1								
BABR	0.000	1.940	0.782	0.777	0.784	0.784	0.000	0.000
BAC	0.000	1.960	0.781	0.778	0.783	0.784	0.000	0.000
Baseline	0.060	2.220	0.778	0.779	0.781	0.781	0.060	0.000
Fixed	0.000	1.000	0.787	0.788	0.787	0.788	0.000	0.000
Reset	1.180	9.860	0.719	0.715	0.763	0.761	0.541	0.541
Periodic model deterioration/improvement, Section 7.2								
BABR	0.000	2.920	0.799	0.799	0.786	0.787	0.000	0.000
BAC	0.000	2.940	0.799	0.799	0.786	0.787	0.000	0.000
Baseline	6.300	43.360	0.749	0.749	0.765	0.766	6.300	0.000
Blind	15.000	99.000	0.712	0.711	0.762	0.762	15.000	0.000
Fixed	0.000	1.000	0.697	0.697	0.697	0.697	0.000	0.000
Reset	0.020	3.740	0.801	0.801	0.790	0.791	0.020	0.020
Training on accumulating data, Section 7.3								
BABR	0.015	3.495	0.945	0.950	0.842	0.854	0.010	0.000
BAC	0.010	3.430	0.949	0.949	0.844	0.854	0.010	0.000
Baseline	0.105	24.015	0.953	0.958	0.872	0.876	0.105	0.000
Blind	3.070	39.000	0.958	0.958	0.926	0.925	3.070	0.000
Fixed	0.000	1.000	0.649	0.636	0.649	0.636	0.000	0.000
Reset	0.055	4.810	0.951	0.956	0.866	0.871	0.018	0.139
Significant model improvement, Section 7.4								
BABR	0.000	4.020	0.802	0.802	0.757	0.757	0.000	0.000
BAC	0.000	3.980	0.801	0.801	0.753	0.753	0.000	0.000
Baseline	0.000	17.163	0.803	0.803	0.778	0.779	0.000	0.000
Blind	0.000	19.000	0.803	0.803	0.790	0.790	0.000	0.000
Fixed	0.000	1.000	0.681	0.682	0.682	0.682	0.000	0.000
Reset	0.000	4.429	0.802	0.802	0.770	0.771	0.000	0.007

Table 4.1: Comparison of automatic Algorithm Change Protocols in different simulation settings for IID data. Columns $\text{BAC}_W, \text{meBAR}_W, \text{meBBR}_W$ display the maximum error rate over all time points. In the incremental model deterioration, we omit aACP-Blind since it always converges to completely uninformative classifier.

aACP	BAC_W	# approved	Final		Cumulative Utility		meBAR$_W$	meBBR$_W$
			Specificity	Sensitivity	Specificity	Sensitivity		
No time trend								
BABR	0.010	1.025	0.712	0.712	0.712	0.712	0.010	0.000
BAC	0.010	1.025	0.712	0.712	0.712	0.712	0.010	0.000
Fixed	0.000	1.000	0.712	0.712	0.712	0.712	0.000	0.000
Acceptability graph constant								
BABR	0.010	1.035	0.651	0.651	0.712	0.712	0.010	0.000
BAC	0.015	1.035	0.651	0.651	0.712	0.712	0.015	0.000
Fixed	0.000	1.000	0.651	0.652	0.712	0.712	0.000	0.000
Acceptability graph changing								
BABR	—	1.704	0.652	0.652	0.689	0.689	—	—
BAC	—	1.720	0.649	0.650	0.689	0.689	—	—
Fixed	—	1.000	0.772	0.773	0.712	0.712	—	—

Table 4.2: Comparison of automatic Algorithm Change Protocols for different time trend scenarios (Section 7.5). Columns BAC$_W$, meBAR$_W$, meBBR$_W$ display the maximum error rate over all time points. Their values are omitted for the case where the acceptability graph is changing over time since the error rates are not well-defined.

Periodic model deterioration and improvement

We set total time $T = 100$ and maximum wait time $\Delta = 5$ for all models. We accumulate 200 new observations at each time point.

Accumulating data

Each patient is represented by 30 covariates and the true outcome is generated using a logistic model. The developer performs logistic regression with a lasso penalty and tunes the penalty parameter using cross-validation. To increase the margin of model improvement at later time points and the ability to detect small improvements, we increase the number of training observations at each time point by five, starting with size 20, and use a larger wait time of $\Delta = 10$. The total time is $T = 40$ since the model performance plateaus quickly.

Significant model improvements

In order to make the model improvements significant with high probability, we accumulate 650 observations at each time point, which is more than the other simulation settings. Since large improvements are relatively rare, we used a short total time of $T = 20$. Since a company is likely more confident in these improvements, the maximum wait time is set to $\Delta = 3$.

Time trends

The total time is $T = 100$ and the wait time is $\Delta = 5$. We accumulate 300 new observations at each time point.

A.3 Sensitivity to choice of hyper-parameters

The definition of the error rates depends on two hyper-parameters: the window size W and the non-inferiority margin ϵ. To study how sensitive the aACPs are to these hyper-parameter, we run the same set of simulations from Section 7 but vary either W or ϵ.

First, we vary ϵ over the values 0, 0.01, 0.05, and 0.1 while keeping the window $W = 15$ fixed (Figure 4.9). (For $\epsilon = 0$, note that approval requires demonstrating superiority.) As the NI margin increases, all the aACPs approve more modifications, both bad and good ones. Compared to aACP-BAC and -BABR, the error rates of aACP-Reset and -Baseline increase more quickly with respect to ϵ. As such, the relative ordering between the aACPs is similar across different values of ϵ.

In Figure 4.10, we vary the window size W over the values 1, 15, 25, and 50 while keeping the NI margin $\epsilon = 0.05$ fixed. Only aACP-BAC and -BABR depend on W; The other aACPs are agnostic to the choice of W. As W increases, aACP-BAC and -BABR become more conservative and approve fewer modifications. Since their error rates are already quite low, the error rates are not very sensitive to changes in W. On the other hand, choosing an excessively large value of W, such as our example with $W = 50$, leads to significantly slower approval rates for proposed model improvements. As such, we suggest selecting a value for W that corresponds to the minimal time period one would like to control error rates for.

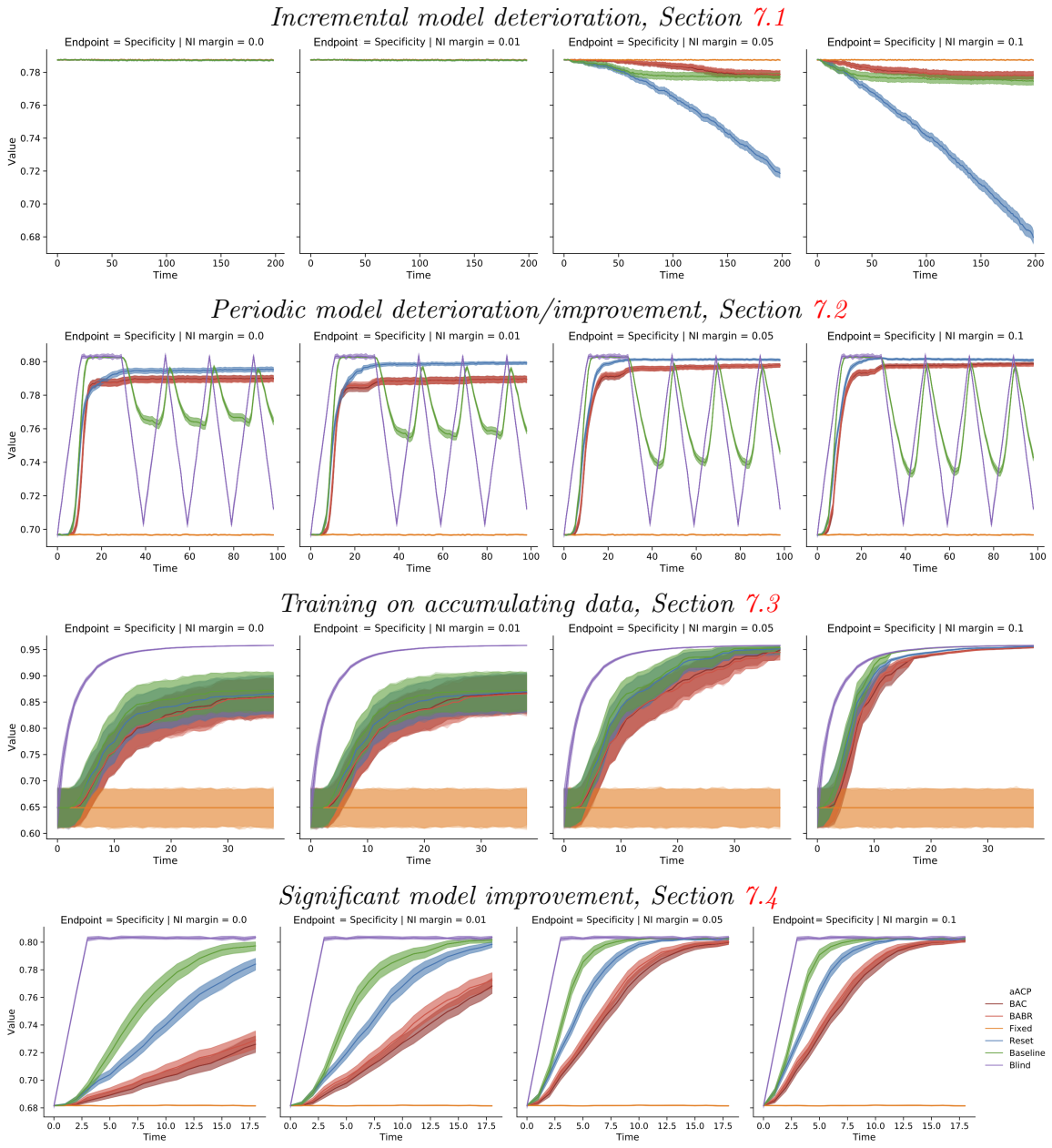


Figure 4.9: Specificity of the currently approved model over time for different simulation settings (rows) and non-inferiority margin values ϵ (columns). Sensitivity plots are very similar and, thus, have been omitted.

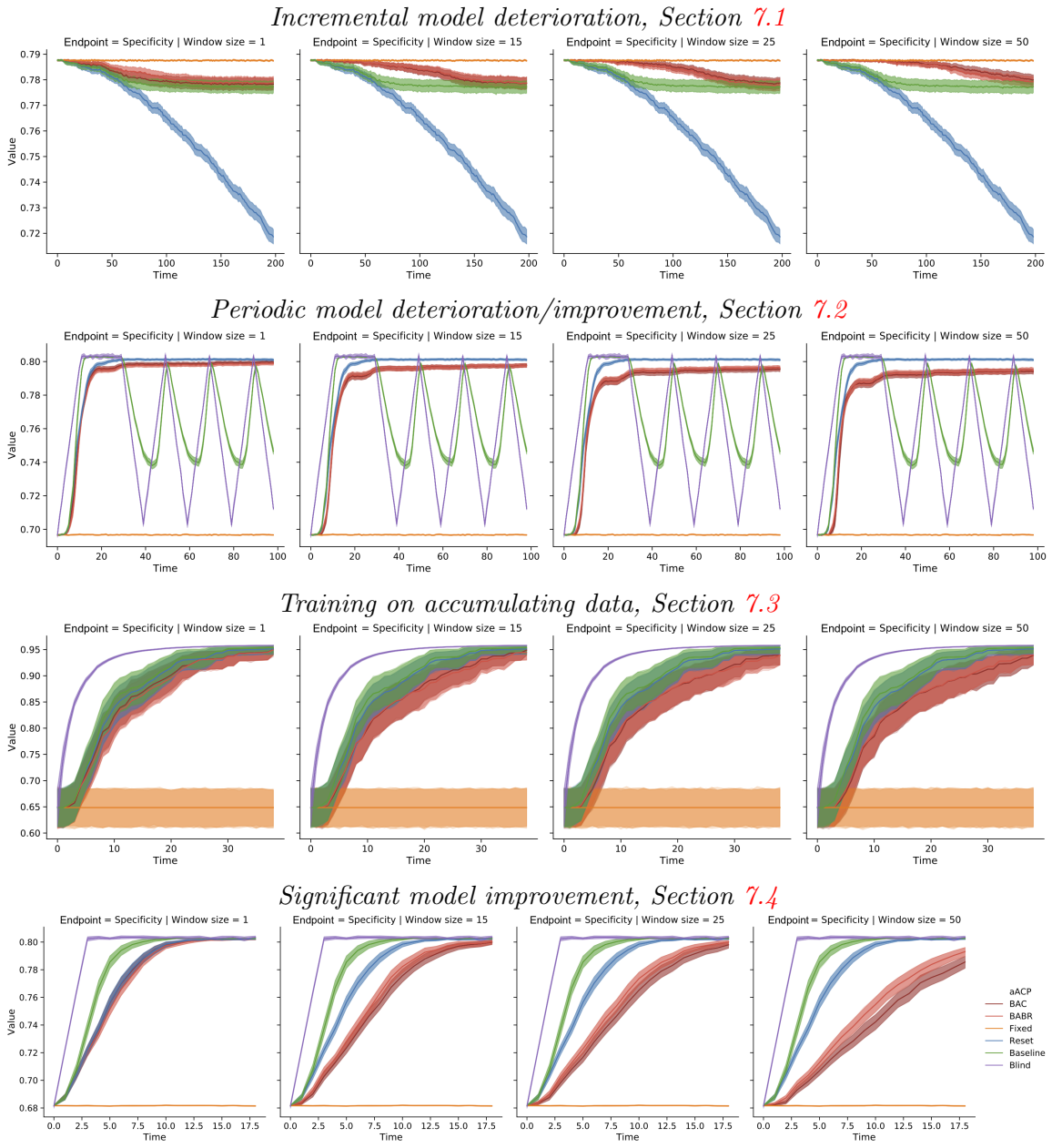


Figure 4.10: Specificity of the currently approved model over time for different simulation settings (rows) and window sizes W (columns). Sensitivity plots are very similar and, thus, have been omitted.

BIBLIOGRAPHY

[1] Apple DEN 180044 reclassification order. https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfpmn/denovo.cfm?ID=DEN180044. Accessed: 2020-5-10.

[2] QuantX DEN 170022 reclassification order. https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfpmn/denovo.cfm?ID=DEN170022. Accessed: 2020-5-10.

[3] D G Altman and J P Royston. The hidden effect of time. Stat. Med., 7(6):629–637, June 1988.

[4] Koko Asakura, Toshimitsu Hamasaki, Tomoyuki Sugimoto, Kenichi Hayashi, Scott R Evans, and Takashi Sozu. Sample size determination in group-sequential clinical trials with two co-primary endpoints. Stat. Med., 33(17):2897–2913, July 2014.

[5] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. J. R. Stat. Soc. Series B Stat. Methodol., 57(1):289–300, 1995.

[6] D A Bloch, T L Lai, and P Tubert-Bitter. One-sided tests in clinical trials with multiple endpoints. Biometrics, 57(4):1039–1047, December 2001.

[7] Daniel A Bloch, Tze Leung Lai, Zheng Su, and Pascale Tubert-Bitter. A combined superiority and non-inferiority approach to multiple endpoints in clinical trials. Stat. Med., 26(6):1193–1207, March 2007.

[8] Avrim Blum and Moritz Hardt. The ladder: A reliable leaderboard for machine learning competitions. International Conference on Machine Learning, 37:1006–1014, 2015.

[9] R J Cook. Interim monitoring of bivariate responses using repeated confidence intervals. Control. Clin. Trials, 15(3):187–200, June 1994.

[10] D L DeMets and K K Lan. Interim analysis: the alpha spending function approach. Stat. Med., 13(13-14):1341–52; discussion 1353–6, 1994.

[11] Alex Dmitrienko and Ajit C Tamhane. Gatekeeping procedures with clinical trial applications. Pharm. Stat., 6(3):171–180, July 2007.

[12] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. Science, 349(6248):636–638, August 2015.

[13] FDA. Multiple endpoints in clinical trials guidance for industry. Technical report, 2017.

[14] FDA. FDA permits marketing of clinical decision support software for alerting providers of a potential stroke in patients, 2018.

[15] FDA. US FDA artificial intelligence and machine learning discussion paper. Technical report, April 2019.

[16] R A Fisher. Statistical Methods for Research Workers. Oliver and Boyd, Edinburgh, 4th edition, 1932.

[17] Thomas R Fleming. Current issues in non-inferiority trials. Stat. Med., 27(3):317–332, February 2008.

[18] Dean P Foster and Robert A Stine. α-investing: a procedure for sequential control of expected false discoveries. J. R. Stat. Soc. Series B Stat. Methodol., 70(2):429–444, April 2008.

[19] Scott Gottlieb. FDA working to lift barriers to generic drug competition, September 2019. Accessed: 2019-8-8.

[20] Himanshu Gupta, Suresh Kumar, Saroj Kumar Roy, and R S Gaud. Patent protection strategies. J. Pharm. Bioallied Sci., 2(1):2–7, January 2010.

[21] Patrick J Heagerty and Yingye Zheng. Survival model predictive accuracy and ROC curves. Biometrics, 61(1):92–105, March 2005.

[22] Larry V Hedges and Ingram Olkin. Statistical Methods for Meta-Analysis. Academic Press, 1985.

[23] Andrew W Hitchings, Emma H Baker, and Teck K Khong. Making medicines evergreen. BMJ, 345:e7941, November 2012.

[24] International Conference on Harmonisation. Ethnic factors in the acceptability of foreign clinical data (ICH E5). 1998.

[25] Adel Javanmard and Andrea Montanari. On online control of false discovery rate. February 2015.

[26] Walter Offen, Christy Chuang-Stein, Alex Dmitrienko, Gary Littman, Jeff Maca, Laura Meyerson, Robb Muirhead, Paul Stryszak, Alex Baddy, Kun Chen, Kati Copley-Merriman, Willard Dere, Sam Givens, David Hall, David Henry, Jose Jackson, Alok Krishen, Thomas Liu, Steve Ryder, A J Sankoh, Julia Wang, and Chyon-Hwa Yeh. Multiple co-primary endpoints: Medical and statistical solutions: A report from the multiple endpoints expert team of the pharmaceutical research and manufacturers of america. Drug Inf. J., 41(1):31–46, January 2007.

[27] Office of the Commissioner. FDA permits marketing of clinical decision support software for alerting providers of a potential stroke in patients. http://www.fda.gov/news-events/press-announcements/fda-permits-marketing-clinical-decision-support-software-alerting-providers-potential-s 2018. Accessed: 2019-8-8.

[28] Margaret Sullivan Pepe. The Statistical Evaluation of Medical Tests for Classification and Prediction. Oxford University Press, 2003.

[29] Stuart J Pocock. Group sequential methods in the design and analysis of clinical trials. Biometrika, 64(2):191–199, August 1977.

[30] Aaditya Ramdas, Fanny Yang, Martin J Wainwright, and Michael I Jordan. Online control of the false discovery rate with decaying memory. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5650–5659. Curran Associates, Inc., 2017.

[31] Aaditya Ramdas, Tijana Zrnic, Martin Wainwright, and Michael Jordan. SAFFRON: an adaptive algorithm for online control of the false discovery rate. International Conference on Machine Learning, 2018.

[32] Shai Shalev-Shwartz. Online learning and online convex optimization. Foundations and Trends® in Machine Learning, 4(2):107–194, 2012.

[33] Ajit C Tamhane, Jiangtao Gou, Christopher Jennison, Cyrus R Mehta, and Teresa Curto. A gatekeeping procedure to test a primary and a secondary endpoint in a group sequential design with multiple interim looks. Biometrics, 74(1):40–48, March 2018.

[34] Diane Tang, Ashish Agarwal, Deirdre O'Brien, and Mike Meyer. Overlapping experiment infrastructure: More, better, faster experimentation. In Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '10, pages 17–26, New York, NY, USA, 2010. ACM.

[35] Tijana Zrnic, Aaditya Ramdas, and Michael I Jordan. Asynchronous online testing of multiple hypotheses. December 2018.

FUTURE DIRECTIONS

In this thesis, we have presented methods for building interpretable and reliable models for biomedicine. Broadly speaking, the models from Part I are carefully designed based on prior knowledge whereas those from Part II are black-box algorithms that require no prior knowledge. In Part I, the models are interpretable but require a large upfront cost; In Part II, the models are quick to build but are difficult to interpret. There is clearly a need for methods that span these two extremes. Though this dissertation does not address this gap, our results serve as a useful step towards this direction.