

# 1 Estimation of cell lineage trees by 2 maximum-likelihood phylogenetics

3 **Jean Feng**<sup>1,2\*</sup>, **William S DeWitt III**<sup>2,3</sup>, **Aaron McKenna**<sup>3</sup>, **Noah Simon**<sup>1</sup>, **Amy Willis**<sup>1</sup>,  
4 **Frederick A Matsen IV**<sup>2,3\*</sup>

**\*For correspondence:**

[jeanfeng@uw.edu](mailto:jeanfeng@uw.edu) (JF);  
[matsen@fredhutch.org](mailto:matsen@fredhutch.org) (FAM)

**Present address:** Seattle,  
Washington

5 <sup>1</sup>Department of Biostatistics, University of Washington, Seattle, United States;  
6 <sup>2</sup>Computational Biology Program, Fred Hutchinson Cancer Research Center, Seattle,  
7 United States; <sup>3</sup>Department of Genome Sciences, University of Washington, Seattle,  
8 United States

## 9 Abstract

10 CRISPR technology has enabled large-scale cell lineage tracing for complex multicellular organisms  
11 by mutating synthetic genomic barcodes during organismal development. However, these sophisti-  
12 cated biological tools currently use ad-hoc and outmoded computational methods to reconstruct  
13 the cell lineage tree from the mutated barcodes. Because these methods are agnostic to the  
14 biological mechanism, they are unable to take full advantage of the data's structure. We propose a  
15 statistical model for the mutation process and develop a procedure to estimate the tree topology,  
16 branch lengths, and mutation parameters by iteratively applying penalized maximum likelihood  
17 estimation. In contrast to existing techniques, our method estimates time along each branch, rather  
18 than number of mutation events, thus providing a detailed account of tissue-type differentiation.  
19 Via simulations, we demonstrate that our method is substantially more accurate than existing ap-  
20 proaches. Our reconstructed trees also better recapitulate known aspects of zebrafish development  
21 and reproduce similar results across fish replicates.

## 22 Introduction

23 Recent advancements in genome editing with CRISPR (clustered regularly interspaced short palin-  
24 dromic repeats) have renewed interest in the construction of large-scale cell lineage trees for  
25 complex organisms [*McKenna et al., 2016, Woodworth et al., 2017, Spanjaard et al., 2018, Schmidt*  
26 *et al., 2017*]. These lineage-tracing technologies, such as the GESTALT method [*McKenna et al.,*  
27 *2016*] that we focus on here<sup>1</sup>, inject Cas9 and single-guide RNA (sgRNA) into the embryo of a  
28 transgenic organism harboring an array of CRISPR/Cas9 targets separated by short linker sequences  
29 (barcodes). These barcodes accumulate mutations because Cas9 cuts are imperfectly repaired  
30 by non-homologous end joining (NHEJ) during development while Cas9 and sgRNA are available.  
31 The resulting mutations are passed from parent cell to daughter cell, which thereby encodes the  
32 ontogeny. Mutated barcodes are later sequenced from the organism, and computational phyloge-  
33 netic methods are then used to estimate the cell lineage tree. Because these barcodes have great  
34 diversity, GESTALT provides researchers with rich data with the potential to reveal organism and  
35 disease development in high resolution.

36 The most common phylogenetic methods used to analyze GESTALT data are Camin-Sokal (C-S)  
37 parsimony [*Camin and Sokal, 1965*] and the neighbor-joining distance-based method [*Saitou and*  
38 *Nei, 1987*]. However these methods are blind to the operation of the GESTALT mutation process, so  
39 the accuracy of the estimated trees are poor [*Salvador-Martínez et al., 2018*]. In addition, existing  
40 methods supply branch length estimates in terms of an abstract notion of distance rather than time,  
41 limiting their interpretability. Therefore, these estimated trees only provide ordering information  
42 between nodes on the same lineage, but not for nodes on parallel lineages. In addition, C-S

<sup>1</sup>Genome Editing of Synthetic Target Arrays for Lineage Tracing

43 parsimony is unable to distinguish between equally parsimonious trees, so obtaining a single tree  
44 estimate is difficult in practice: We find over ten thousand parsimony-optimal trees for existing  
45 datasets. To address these challenges, we set out to develop a statistical model of the mutation  
46 process, allowing us to estimate branch lengths that correspond to time as well as the mutation  
47 parameters.

48 No appropriate likelihood model is currently available for GESTALT because CRISPR arrays  
49 violate many classical statistical phylogenetic assumptions. First, Cas9 enzymes may cut two targets  
50 simultaneously with the entire intervening sequence deleted during NHEJ. In addition, once the  
51 nucleotide sequence for a target is modified, Cas9 is no longer able to cut the target. Thus sites  
52 are not independent, the mutation process is irreversible, and cuts can introduce long insertion  
53 and/or deletions. In contrast, the classical phylogenetic assumptions are that individual nucleotide  
54 positions are independent and that the mutation process is reversible and only introduces point  
55 mutations [Felsenstein, 2004, Yang, 2014]. Finally, these types of methods assume that there are  
56 many independent observations — their estimates are unstable when the effective sample size  
57 is much smaller than the number of parameters to estimate [Goolsby, 2016, Adams and Collyer,  
58 2018, Julien et al., 2018].

59 In this paper, we introduce GAPML (GESTALT analysis using penalized Maximum Likelihood), a  
60 statistical model for GESTALT and tree-estimation method (including topology and branch lengths)  
61 by an iterative procedure based on maximum likelihood estimation. We model barcode mutations  
62 as a two-step process: Targets are cut according to a continuous time Markov chain, immediately  
63 followed by random insertions or deletions of nucleotides (indels). Our method does not rely on  
64 the aforementioned assumptions. Instead we introduce the following assumptions tailored to the  
65 GESTALT setting:

- 66 • (*outermost-cut*) an indel is introduced by cuts at the outermost cut sites
- 67 • (*target-rate*) the cut rates only depend on which targets are active (i.e. able to be cut)
- 68 • (*indel-probability*) the conditional probability that an indel is introduced only depends on which  
69 targets were cut.

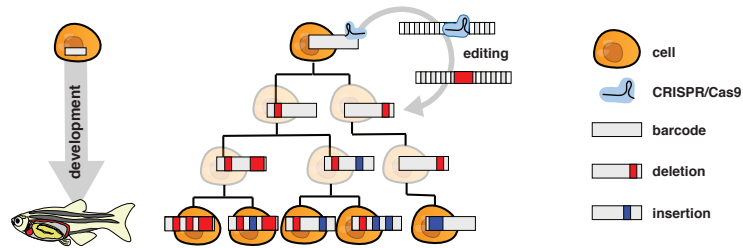
70 From these assumptions, we show that the Markov process is “lumpable” and the aggregated  
71 process is compatible with Felsenstein's pruning algorithm, thereby enabling efficient computation  
72 of the likelihood [Kemeny and Laurie Snell, 1976, Felsenstein, 1981]. Since only a small number of  
73 barcodes are usually available in practice, we propose a regularization method on the branch length  
74 and mutation parameters to stabilize and improve estimates. Our method extends maximum-  
75 likelihood phylogenetic methods with branch-length penalties [Kim and Sanderson, 2008] to the  
76 setting where the tree topology is unknown.

77 We validate our method on simulated and empirical data. In simulations, our method is  
78 more accurate than current tree-estimation methods. In addition, we reconstruct cell lineage  
79 trees of transgenic zebrafish from McKenna et al. [2016] and show that our trees better reflect  
80 the known biology of zebrafish development. Based on these results, we conclude that with  
81 appropriate statistical techniques it is possible to reconstruct an accurate cell lineage tree with  
82 current GESTALT technology, which addresses some concerns raised in Salvador-Martínez et al.  
83 [2018]. Our simulation engine and estimation method are available on Github ([https://github.com/  
84 matsengrp/gestaltmania](https://github.com/matsengrp/gestaltmania)).

## 85 Results

### 86 Brief description of our probabilistic GESTALT evolution model

87 We model the GESTALT barcode (see Figure 1) as a continuous time Markov chain where the state  
88 space is the set of all nucleotide sequences. A state transition is an instantaneous event where  
89 either (1) an unmodified target is cut then the repair process inserts/deletes nucleotides around the  
90 cut site, or (2) two unmodified targets are cut, the intervening sequence is removed, and the repair  
91 process inserts/deletes nucleotides around the cut sites. The transition rate between barcode



**Figure 1.** An unmodified array of CRISPR/Cas9 target sites (i.e., a GESTALT barcode) is engineered into an organism's genome. CRISPR/Cas9 enzyme complex with corresponding guide sequences are directed to make double-stranded breaks in the barcode. These breaks are repaired in an error-prone fashion resulting in insertions and deletions at target sites. These insertions and deletions will accumulate in a lineage specific fashion, passed from mother to daughter cell, and further insertions and deletions can add additional information. These integrated barcodes can then be recovered by DNA sequencing at the timepoint of interest.

92 sequences depends on the entire sequence and each target is associated with a separate cut rate.  
93 If multiple copies of the barcode are used, we assume the barcodes are on separate chromosomes  
94 or are sufficiently far apart that they act in an independent and identically distributed (iid) manner.

95 We use this Markov model for GESTALT barcodes evolving along a cell lineage tree where the  
96 vertices represent cell divisions and the edge lengths represent time between cell divisions. The  
97 full cell lineage tree describes the relationships of all cells in the organism. Since we only collect a  
98 small sample of all the cells, our goal is to recover the subtree describing the development of the  
99 observed sequences.

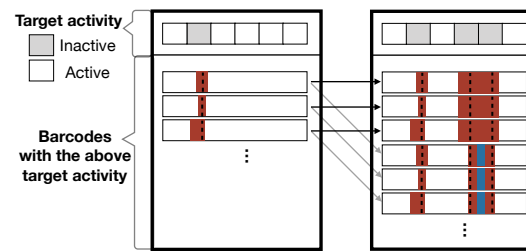
100 To estimate this subtree, our method needs to calculate the likelihood of possible trees and  
101 model parameters, which requires an enumeration of the possible barcodes at each internal node.  
102 However a full enumeration is infeasible. For example, a double cut (transition (2) described above)  
103 could remove one or more targets, which could have themselves been modified in an infinite  
104 number of possible ways before the double cut erased this history.

105 We have carefully chosen our assumptions to make this likelihood tractable yet maintain  
106 biological realism. Briefly, under the *target-rate* and *indel-probability* assumptions, we can group  
107 states together if they share the same set of unmodified targets to calculate the likelihood more  
108 efficiently, a property known more formally known as “lumpability” (Figure 2). Since the number  
109 of targets in a barcode is typically small (e.g. 10 targets per barcode in *McKenna et al. [2016]*,  
110 *Schmidt et al. [2017]*), calculating the likelihood becomes computationally feasible. In addition, the  
111 *outermost-cut* assumption allows us to exclude many groups from the likelihood computation so  
112 that the number of enumerated groups at most internal nodes is typically linear in the number of  
113 unique indels.

## 114 **A maximum-likelihood tree estimation procedure**

115 We follow current best practice for maximum-likelihood phylogenetics by optimizing the tree and  
116 mutation parameters of our model using a hill-climbing iterative search over tree space. First, we  
117 initialize the tree topology by selecting a random parsimony-optimal tree from C-S parsimony. At  
118 each subtree prune and regraft (SPR) iteration, we select a random subtree and regraft where the  
119 penalized log likelihood is highest (Figure 3c). The method stops when the tree no longer changes.  
120 At each iteration, we only consider SPR moves that preserve the parsimony score as we have  
121 found that the parsimony-optimal trees tend to have the highest likelihoods (Figure 18). The entire  
122 algorithm is presented in Algorithm 1. We discuss some important details of our method below.

123 We maximize a penalized log likelihood as opposed to the unpenalized version since the latter  
124 tends to give unstable and inaccurate estimates when the dataset is generated by a small number  
125 of barcodes. In particular, the length of the leaf branches and the variance of the target rates  
126 tend to be overestimated in such settings. Thus we use a penalty function that discourages large



**Figure 2.** 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 the same for all barcodes in the left box.

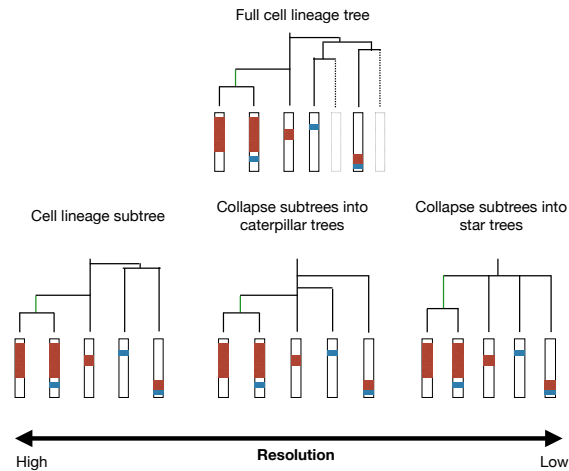
127 differences in branch lengths and target cut rates. Penalization introduces a slight complication  
128 since certain candidate SPR moves have naturally larger penalties. In order to make the penalty  
129 comparable between candidate SPR moves, we randomly select a leaf in the subtree and apply the  
130 candidate SPR moves only to that single leaf. When scoring the SPR moves, the penalty is calculated  
131 for the shared subtree, i.e. the tree where we ignore the random leaf. Finally, we regraft the entire  
132 subtree where the penalized log likelihood is highest.

133 Our method is able to estimate the tree at a finer resolution than existing methods (Figure 3a).  
134 The most commonly used method, C-S parsimony, produces estimates at the coarsest resolution:  
135 For nodes where the ordering is ambiguous, the method simply groups them under a single parent  
136 node. This commonly results in tree estimates with many multifurcating nodes (nodes with 3+  
137 children) that have ten or more children. Our method uses the estimated model parameters to  
138 estimate the order and time of ambiguous nodes by projecting the subtree onto the space of  
139 caterpillar trees (Figure 3b). By producing tree estimates at a finer resolution, our method allows  
140 researchers to learn more about the structure of the true cell lineage tree. In addition, taking  
141 advantage of the irreversibility property, we efficiently estimate the branch ordering within the  
142 caterpillar trees by solving a single optimization problem, rather than considering each possible  
143 ordering separately.

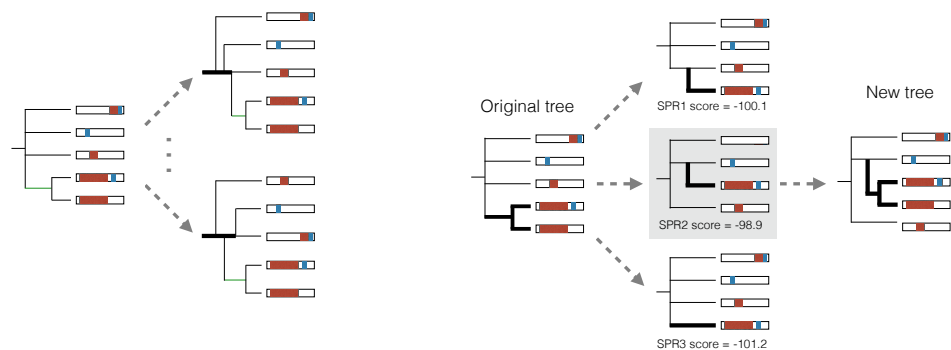
## 144 **Simulation engine and results**

145 We built a simulation engine of the GESTALT mutation process during embryonic development.  
146 Since cell divisions during embryonic development begin in a fast metasynchronous fashion and  
147 gradually become more asynchronous [Moody, 1998], the simulation engine generates a cell lineage  
148 tree by performing a sequence of synchronous cell divisions followed by a birth-death process  
149 where the birth rate decays with time. We mutate the barcode along this cell lineage tree according  
150 to our model of the GESTALT mutation process. The simulation engine can generate data that  
151 closely resembles the data collected from zebrafish embryos in McKenna et al. [2016] (Figure 4a).  
152 We can input different barcode designs into the simulation engine to understand how they affect  
153 our ability to reconstruct the cell lineage tree.

154 We used our simulation engine to assess the validity and accuracy of the estimated model  
155 parameters and tree. Because our method infers branch lengths, we evaluate the accuracy using  
156 two metrics that include branch length information: BHV distance [Billera et al., 2001] and internal  
157 node height correlation (see Figure 15). We compare our method to a simpler model-free approach:  
158 estimating the tree topology using C-S parsimony [Camin and Sokal, 1965] or neighbor-joining  
159 (NJ) [Saitou and Nei, 1987] and then applying semiparametric rate smoothing (chronos in the R



**(a)** We show the subtree of a full cell lineage tree (top) at different resolutions. The highest resolution preserves the bifurcating tree structure (left). The lowest resolution preserves very coarse order information by collapsing a subtree into a multifurcating node (right). In between these two resolutions, we can project the tree onto the space of caterpillar trees and preserve the ordering information between nodes (middle).



**(b)** We resolve each multifurcation in the tree into a caterpillar tree, which places all the children nodes along a central path. This central path, called a caterpillar spine, is indicated by the bold lines in the trees on the right. There are many possible orderings in a caterpillar tree. Here we show two such orderings. Our method chooses the ordering that maximizes the penalized log likelihood.

**(c)** 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 score (right).

**Figure 3.** Overview of our tree estimation method.

---

**Algorithm 1** Cell lineage tree reconstruction for penalty parameter  $\kappa$

---

Initialize tree  $\mathbb{T}$ . Let the sequenced GESTALT barcodes be denoted  $D$ .

**for** Iteration  $k$  **do**

Pick a random subtree from  $\mathbb{T}$ . Select one of the leaves  $C$  of the subtree.

**for** each possible SPR move involving the subtree that doesn't change the parsimony score (including the no-op) **do**

Construct  $\mathbb{T}'$  by applying the SPR to leaf  $C$ ; let  $\mathbb{T}'_{\text{shared}}$  be the subtree of  $\mathbb{T}'$  when excluding  $C$

Set the score of the SPR move as the penalized log likelihood maximized with respect to the branch length parameters  $\ell$  and deactivation and indel process parameters  $\theta$  and  $\beta$ , respectively:

$$\max_{\ell, \theta, \beta} \log \Pr(D, \text{the barcode is constant along all caterpillar spines}; \mathbb{T}', \ell, \theta, \beta)$$

Approximation to the likelihood

$$+ \underbrace{\text{Pen}_{\kappa}(\mathbb{T}'_{\text{shared}}, \ell, \theta, \beta)}_{\text{Penalty on branch lengths and mutation parameters}}$$

**end for**

Update the tree  $\mathbb{T}$  by performing the SPR move on the subtree that maximizes the score

**end for**

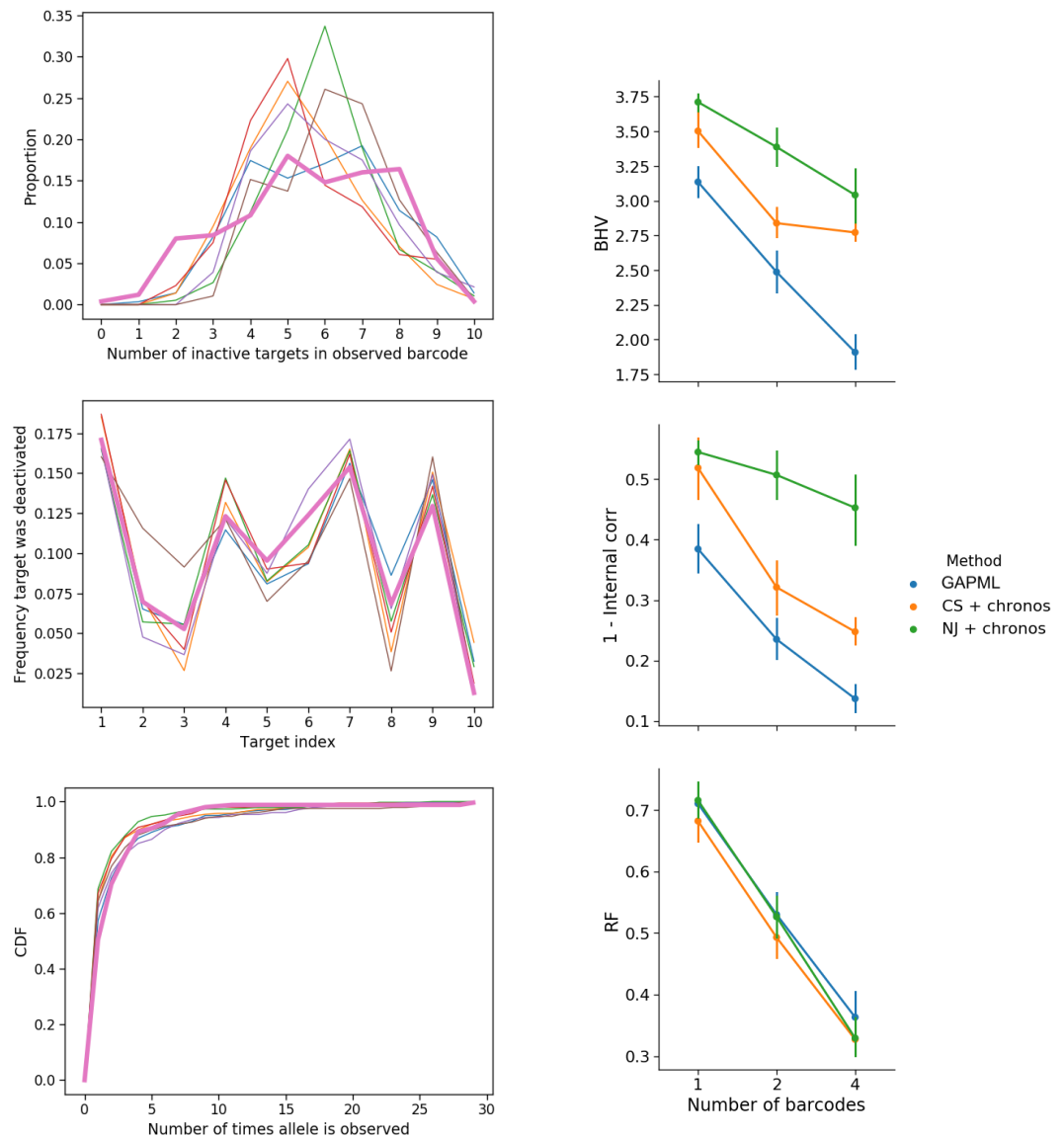
---

160 package ape) to estimate branch lengths [Sanderson, 2002]. We will refer to these two approaches  
 161 as “CS+chronos” and “NJ+chronos.” We do not compare against the original tree estimates from  
 162 C-S parsimony and neighbor-joining since those branch lengths correspond to edit distance and  
 163 have very poor performance according to our two metrics. Our method consistently outperforms  
 164 these alternative methods (Figure 4c). We note that previous *in silico* analyses of GESTALT measure  
 165 accuracy in terms of the Robinson-Foulds (R-F) distance, which only depends on the tree topology  
 166 [Salvador-Martínez et al., 2018]. However the R-F distance does not recognize that different tree  
 167 topologies can be very similar depending on their branch lengths, and is therefore too coarse as a  
 168 performance metric.

169 We find, based on the simulations, that our likelihood-based method improves in performance  
 170 as the number of independent barcodes increases (Figures 4b). In a simulation with a six-target  
 171 barcode, the estimated tree from a single barcode has internal node height correlation of 0.5 with  
 172 the true tree whereas using four barcodes increases the correlation to 0.9. Even though other  
 173 analyses of GESTALT have recommended increasing the number of targets in a single barcode  
 174 to improve tree estimation [Salvador-Martínez et al., 2018], it is more effective to increase the  
 175 number of targets by introducing independent (and identical) barcodes (Figure 16).

176 **Improved zebrafish lineage reconstruction**

177 To validate our method, we reconstructed cell lineages using our method and other tree-building  
 178 methods on GESTALT data from zebrafish [McKenna et al., 2016]. As the true cell lineage tree is  
 179 not known for zebrafish, we employed more indirect measures of validity. For each method, we  
 180 asked (1) if similar conclusions could be made across different biological replicates and (2) if the tree  
 181 estimates aligned with the known biology of zebrafish development. The dataset includes two adult  
 182 zebrafish where cells were sampled from dissected organs. The organs were chosen to represent  
 183 all germ layers: the brain and both eyes (ectodermal), the intestinal bulb and posterior intestine  
 184 (endodermal), the heart and blood (mesodermal), and the gills (neural crest, with contributions  
 185 from other germ layers). The heart was further divided into four samples— a piece of heart tissue,  
 186 dissociated unsorted cells (DHCs), FACS- sorted GFP+ cardiomyocytes, and non-cardiomyocyte heart  
 187 cells (NCs). In addition, datasets were collected from embryos before gastrulation (dome stage, 4.3  
 188 hours post-fertilization (hpf)), at pharyngula stage (30 hpf), and from early larvae (72 hpf), where



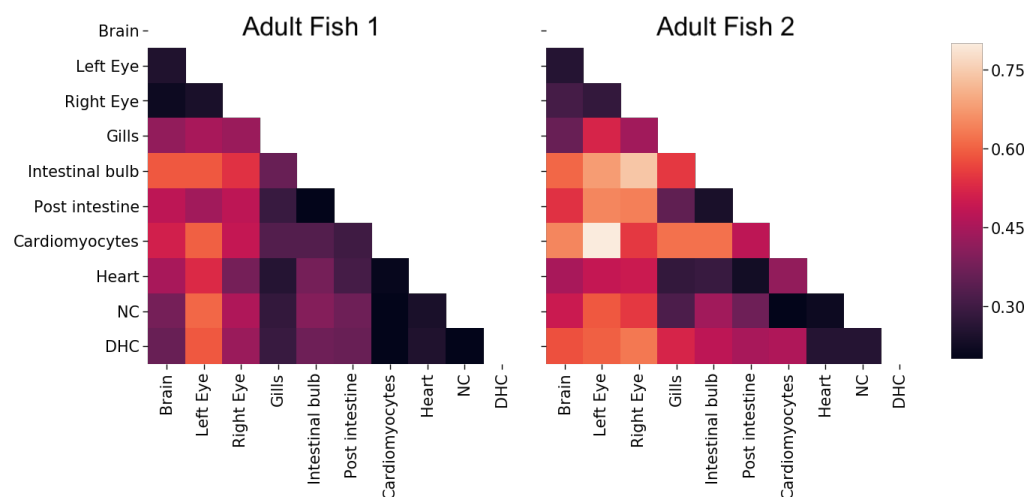
**(a)** A comparison of summary statistics on the simulated data (each thin line is a replicate; color used only to aid in distinguishing between replicates) vs. 250 randomly selected alleles from the first dome fish (bolded line). We generated data from our simulation engine and randomly sampled leaves to obtain around 250 unique alleles. The distribution of inactive targets and allele abundances (the number of times an allele is observed) are similar.

**(b)** Results for data simulated from a barcode with six targets and randomly sampled to obtain roughly 100 unique alleles. The performance of GAPML improves with the number of barcodes. GAPML performs significantly better than the other methods in terms of BHV (top) and the internal node height correlation metrics (middle). The methods are hard to distinguish with respect to the Robinson-Foulds (RF) metric (bottom).

Method	BHV	1 - Internal node correlation
GAPML	5.68 (5.51, 5.85)	0.45 (0.42, 0.48)
CS + chronos	6.39 (6.20, 6.58)	0.58 (0.52, 0.64)
NJ + chronos	8.48 (8.38, 8.58)	0.66 (0.64, 0.68)

**(c)** 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.

**Figure 4.** Simulation results. We denote Camin-Sokal parsimony and neighbor-joining with nonparametric rate smoothing as CS+chronos and NJ+chronos, respectively.



**Figure 5.** The 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.

**Figure 5-Figure supplement 1.** Distance matrices for the estimated trees from Camin-Sokal parsimony and *chronos*

**Figure 5-Figure supplement 2.** Distance matrices for the estimated trees from neighbor-joining and *chronos*

189 the cell type assignments are unknown.

190 GAPML captures more similar developmental relationships between tissue types across  
191 the two adult fish replicates.

192 For each estimated tree, we calculated the *distance between tissues* — the average tree distance  
193 between a leaf of one tissue to the closest internal node leading to a leaf from the other tissue,  
194 weighted by the allele abundance (Figure 5). (All alleles that were found in the blood were removed  
195 since blood is found in all dissected organs and can confound the relationship between organs  
196 *McKenna et al. [2016]*.) Recall that all of the fitting procedures are completely agnostic to any tissue  
197 source or cell abundance information. For a good method, we expect the correlation between  
198 tissue distances from the two fish to be close to one. We tested if the correlations were significant  
199 by permuting the cell types and abundances in the estimated trees. The correlation was 0.770  
200 ( $p < 0.001$ ) using our method whereas ‘CS+*chronos*’ and ‘NJ+*chronos*’ had correlations of 0.306  
201 ( $p = 0.21$ ) and  $-0.325$  ( $p = 0.22$ ), respectively. One might be concerned that our method is consistent  
202 across fish replicates because it returns very similar trees regardless of the data. However, this is  
203 not the case: When we re-run our method with randomly permuted cell types and abundances, the  
204 average correlation between the tissue distances drops to zero.

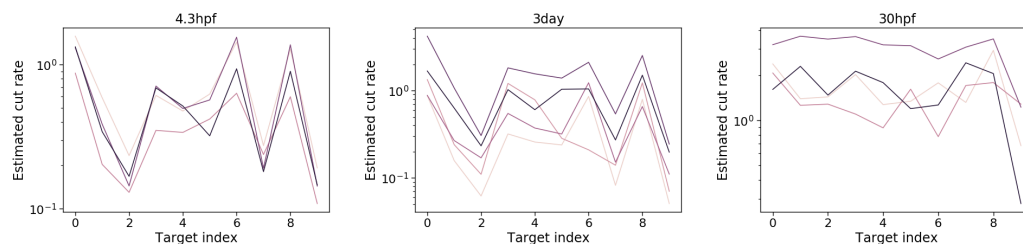
205 GAPML estimates similar mutation parameters across fish replicates.

206 For each time point, the fish replicates were traced using the same GESTALT barcode and processed  
207 using the same experimental protocol (Table 6a). We compared the estimated target rates from our  
208 method to those estimated using a model-free empirical average approach where the estimated  
209 target cut rate is the proportion of times a cut was observed in that target in the set of unique  
210 observed indels. The average correlation between the estimated target rates from our method were  
211 much higher than that for the alternate approach (Figure 6a). In fact, we can also compare target  
212 cut rates between fish of different ages that share the same barcode, even if the experimental  
213 protocols are slightly different. The 4.3hpf and 3day fish share the same barcode version and we  
214 find that the target rate estimates are indeed similar (Figure 6b). Again, a possible concern is that



Fish age	$n$	Barcode version	GAPML Correlation	Empirical average correlation
4 months	2	7	0.891	0.685
3 days	5	7	0.881 (0.839, 0.982)	0.688 (0.610, 0.923)
30 hpf	4	6	0.309 (0.309, 0.794)	0.052 (0.052, 0.727)
4.3 hpf	4	7	0.931 (0.931, 0.982)	0.743 (0.717, 0.976)

(a) Mean Spearman correlation between target lambda rates across fish replicates. 95% confidence intervals (via bootstrap) shown in parentheses.



(b) Fitted target lambda rates for fish sampled at 4.3hpf (left), 3 days (middle), and 30 hpf (right), where each colored line corresponds to the estimates for a single fish. The fish sampled at 4.3hpf and 3 days had the same barcode and share similar target rates. The 30hpf fish used a different barcode and have different estimated target rates.

**Figure 6.** Target cut rate estimates are consistent across fish replicates.

215 our method may have high correlation because it outputs very similar values regardless of the  
 216 data. However, the estimated target cut rates were different for fish with different barcode versions.  
 217 More specifically, the 30hpf fish used version 6 of the GESTALT barcode whereas the other fish used  
 218 version 7. Visually, the target rates look quite different between those in the 30hpf fish and the  
 219 other fish with the version 7 barcode (Figure 6b). Calculating the pairwise correlations between  
 220 the estimated rates in 30hpf versus 3day fish, the average correlation, 0.416, is quite low and the  
 221 bootstrap 95% confidence interval, (0.046, 0.655), is very wide and nearly covers zero.

222 GAPML recovers both cell-type and germ-layer restriction.

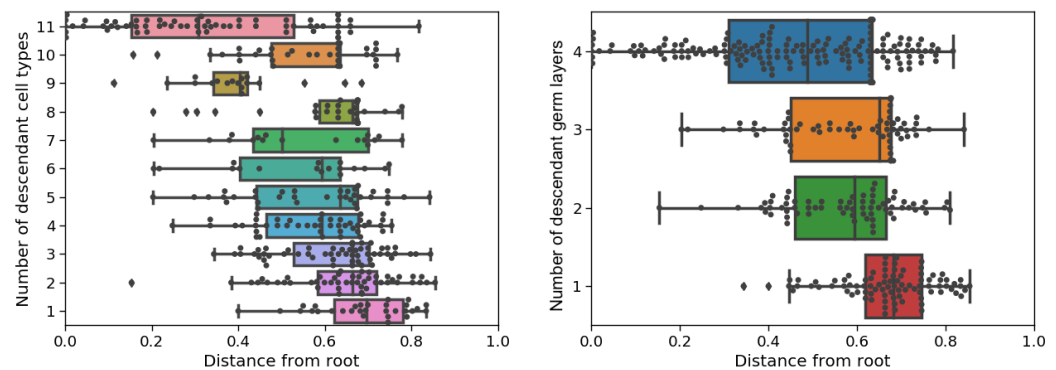
223 It is well known that cells are pluripotent initially and specialize during development. To evaluate  
 224 recovery of specialization by tissue type, we calculated the correlation between the estimated  
 225 time of internal tree nodes and the number of descendant tissue types; to evaluate recovery of  
 226 specialization by germ layer, we calculated the correlation between the estimated time of internal  
 227 nodes and the number of germ layers represented at the leaves. (As before, all the estimation  
 228 methods do not use the tissue origin and germ layer labels.) Since any tree should generally  
 229 show a trend where parent nodes tend to have more descendant cell types than their children,  
 230 we compared our tree estimate to the same tree but with random branch length assignments  
 231 and randomly permuted tissue types. Our method estimated much higher correlations compared  
 232 to these random trees (Table 7a). We show an example of the node times versus the number of  
 233 descendant cell types and germ layers in Figure 7b. The estimated correlations from the other  
 234 methods tended to be closer to zero compared to those in GAPML in all cases, except when using  
 235 'NJ + chronos' to analyze the second adult fish. However upon inspection, the correlation is high for  
 236 'NJ + chronos' because it estimates that cells are pluripotent for over 90% of the fish's life cycle and  
 237 specialize during a small time slice at the very end.

### 238 Analysis of the zebrafish GESTALT data

239 In this section, we analyze the fitted trees of the adult zebrafish in more detail. Our primary goals are  
 240 to (1) check if summaries concord with known zebrafish biology, (2) generate new hypotheses about

Adult Fish	Estimation Method	# tissue types vs time			# germ layers vs time		
		Corr	Random corr	p-value	Corr	Random corr	p-value
1	GAPML	-0.492	-0.168	< 0.001	-0.421	-0.124	< 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	-0.493	-0.220	<0.001	-0.410	-0.151	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

(a) Estimated correlations between the number of descendant cell types/germ layers vs. the time of internal nodes in the tree. Since some tree topologies naturally have higher correlations, we also show the correlation when cell types are shuffled and branch lengths are randomly assigned. The p-value for each tree is calculated with respect to their respective randomly shuffled trees.



(b) Box plots of the internal node times in the estimated tree for the first adult fish using GAPML, where nodes are grouped by the number of descendant cell types (left column) and the number of descendant germ layers (right column).

**Figure 7.** Estimated relationships between node times and number of descendant cell types and germ layers in the two adult fish for the different methods.

**Figure 7-Figure supplement 1.** Internal node times versus number of descendant cell types in the estimated tree for the second adult fish using GAPML.

**Figure 7-Figure supplement 2.** Internal node times versus number of descendant germ layers in the estimated tree for the second adult fish using GAPML.

**Figure 7-Figure supplement 3.** Internal node times versus number of descendant cell types in the estimated tree for the first adult fish using Camin-Sokal parsimony and *chronos*.

**Figure 7-Figure supplement 4.** Internal node times versus number of descendant germ layers in the estimated tree for the first adult fish using Camin-Sokal parsimony and *chronos*.

**Figure 7-Figure supplement 5.** Internal node times versus number of descendant cell types in the estimated tree for the second adult fish using Camin-Sokal parsimony and *chronos*.

**Figure 7-Figure supplement 6.** Internal node times versus number of descendant germ layers in the estimated tree for the second adult fish using Camin-Sokal parsimony and *chronos*.

**Figure 7-Figure supplement 7.** Internal node times versus number of descendant cell types in the estimated tree for the first adult fish using neighbor-joining and *chronos*.

**Figure 7-Figure supplement 8.** Internal node times versus number of descendant germ layers in the estimated tree for the first adult fish using neighbor-joining and *chronos*.

**Figure 7-Figure supplement 9.** Internal node times versus number of descendant cell types in the estimated tree for the second adult fish using neighbor-joining and *chronos*.

**Figure 7-Figure supplement 10.** Internal node times versus number of descendant germ layers in the estimated tree for the second adult fish using neighbor-joining and *chronos*.

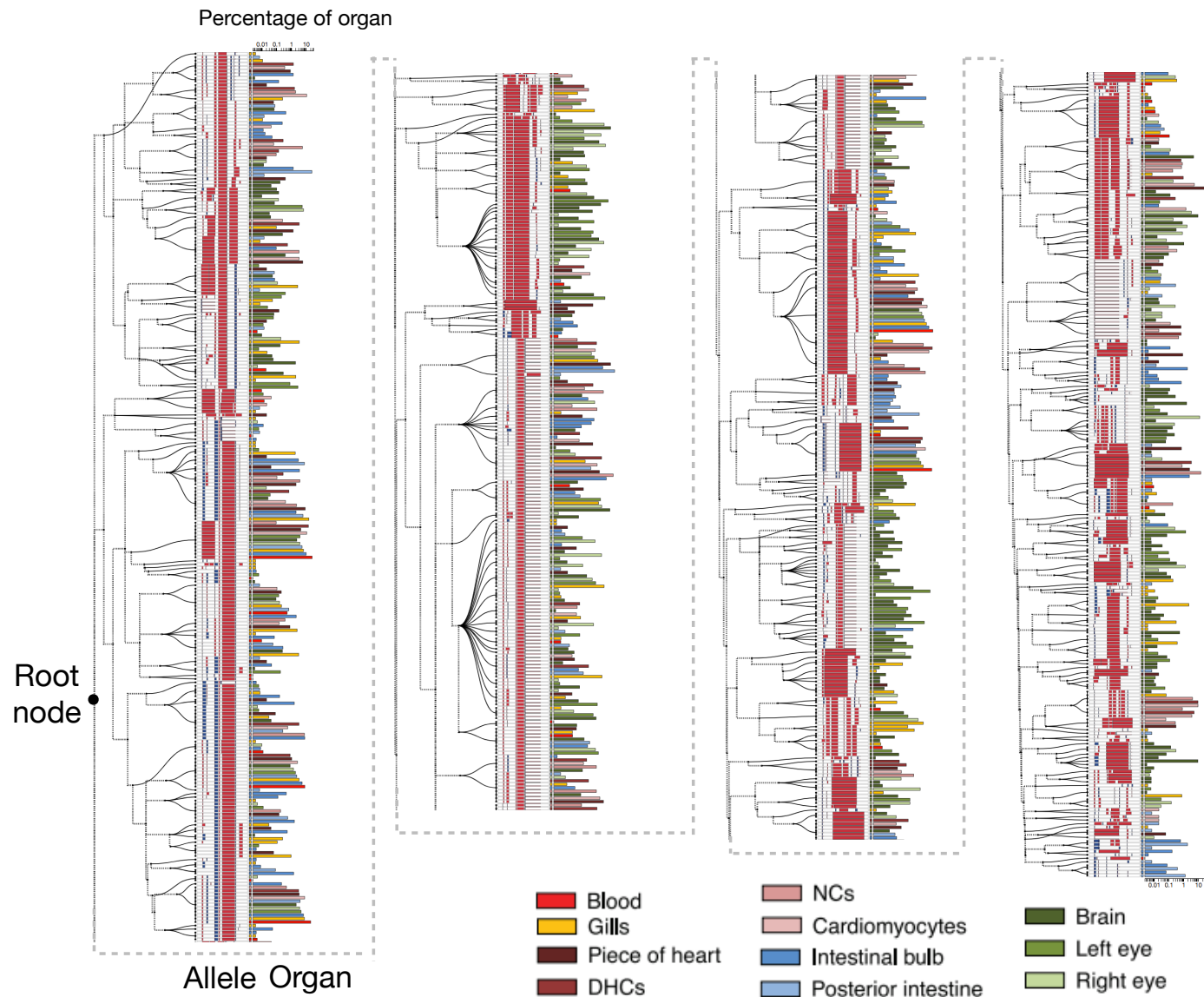
241 zebrafish development, and (3) generate new hypotheses on how to improve the experimental  
242 procedure. Again, as our method is agnostic to the tissue types, our trees have no prior assumptions  
243 or particular biases regarding the relationships between cell types.

244 Here we focus on the ordering and relative length of events. We ignore absolute estimated  
245 times since our fitting procedure for a single barcode heavily penalizes large differences between  
246 branch lengths. Though this procedure aids estimation accuracy, it also heavily biases the absolute  
247 time estimates. Thus in the figures, we scale time to be between 0 to  $T = 1$  to draw focus away from  
248 the absolute times. (We anticipate the branch length estimates to be more accurate to improve  
249 with the GESTALT technology. According to our simulations, the absolute branch length estimates  
250 are much more reliable when the several barcodes are inserted into the organism.)

251 We begin with a coarse summary of the cell lineage tree: We plot the average distance between  
252 a leaf node of one tissue type to the most recent ancestor of each different tissue type (Figure 5).  
253 This matrix recapitulates some well-established facts about zebrafish development. For example,  
254 we estimate that tissue types from the endoderm and mesoderm tended to have shorter shared  
255 lineage distances; these tissue types tended to separate from the ectodermal tissues earliest. This  
256 signal potentially captures the migration of the mesoderm and endoderm through the blastopore,  
257 isolating them from the ectoderm [*Solnica-Krezel, 2005*]. In addition, previous studies have found  
258 that gills are formed when the anterior part of the intestine grows toward and fuses with the body  
259 integument [*Shadrin and Ozernyuk, 2002*]. The distance matrix shows a large proportion of gill  
260 cells dividing late from the other endoderm and mesoderm layers.

261 The distance matrix also shows that the GFP+ cardiomyocytes tend to be farthest away from  
262 other tissue types, which could be either a developmental signal or an artifact of the experimental  
263 protocol. GFP+ cardiomyocytes were sorted using fluorescence-activated cell (FACS) and this  
264 purity could drive their separation from the other more heterogeneous organ populations. An  
265 interesting biological speculation would be that the heart is the first organ to form during vertebrate  
266 embryo development and, in particular, the myocardial cells are the first to develop, driving this  
267 observed signal [*Keegan et al., 2004*]. These observations show GAPML's improved lineage distance  
268 estimation provide a more refined measure of the developmental process, and as our simulations  
269 show, will only improve as experimental approaches becomes more sophisticated.

270 The full cell lineage tree estimated using GAPML for the first adult zebrafish provides significantly  
271 more detail than the Camin-Sokal parsimony tree inferred for the original *McKenna et al. [2016]*  
272 publication (Figure 8). Our tree has estimated branch lengths whereas the branches were all  
273 unit-length in *McKenna et al. [2016]*. In addition, the dashed lines in our tree correspond to the  
274 caterpillar spines where we have estimated the ordering between children of multifurcating nodes.  
275 Since the original maximum parsimony tree estimate in *McKenna et al. [2016]* contained many  
276 multifurcating nodes and our method converts any multifurcating node to a caterpillar tree, our  
277 final tree contains many caterpillar trees. The longest caterpillar spine in our estimated tree starts  
278 from the root node and connects all the major subtrees that share no indel tracts. As the zebrafish  
279 embryo rapidly divides from the single-cell stage, these initial CRISPR editing events establish the  
280 founding cell in each subtree. GAPML estimates the target cut rates to order the events along  
281 the caterpillar trees, an impossible task in the original Camin-Sokal multifurcating trees. Lastly,  
282 we observe that the last three subtrees at the end of this spine (farthest away from the root) are  
283 primarily composed of alleles only observed in the intestinal bulb and the posterior intestine. This  
284 concords with our understanding of zebrafish development: Of the dissected organs, the digestive  
285 tract is the last to fully differentiate at day four [*Moody, 1998*]. In aggregate, these examples again  
286 show the power of a refined lineage tree to establish new interesting biological questions and a  
287 refined map in which to answer them.



**Figure 8.** 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 (log<sub>10</sub> scale). The dashed lines correspond to the caterpillar spines.

**Figure 8-source data 1.** Raw data for estimated tree in Figure 8

**Figure 8-source data 2.** Raw data for estimated tree from Camin-sokal parsimony and `chronos` for the same alleles in Figure 8

**Figure 8-source data 3.** Raw data for estimated tree from neighbor-joining and `chronos` for the same alleles in Figure 8

**Figure 8-Figure supplement 1.** Estimated cell lineage tree for all alleles from the second adult zebrafish using GAPML.

**Figure 8-source data 4.** Raw data for estimated tree from GAMPL for alleles from the second adult fish

**Figure 8-source data 5.** Raw data for estimated tree from Camin-sokal parsimony and `chronos` for alleles from the second adult fish

**Figure 8-source data 6.** Raw data for estimated tree from neighbor-joining and `chronos` for alleles from the second adult fish

## 288 **Analysis of GESTALT barcode mutation parameters**

289 Finally, our model's estimated target cut rate parameters (Table 1) provide an interesting resource  
290 when considering redesigns of the GESTALT barcode. Here we focus on the GESTALT barcode in the  
291 adult fish. The estimated target cut rates were very similar across the two fish replicates.

292 We estimated very different cut rates across the ten targets. Target 1 and 9 had the highest cut  
293 rates; target 3 had the lowest cut rate. The ratio between the highest and lowest cut rates is at least  
294 10 in both fish, i.e. a deletion at target 1 is at least 10 times more likely to occur than at target 3. In  
295 terms of the tree estimation, the targets with high cut rates mainly help capture early cell divisions  
296 whereas targets with low cut rates tend to capture late cell divisions. Having a broad spectrum of  
297 target cut rates is useful for capturing cell divisions throughout the tree, though the specific details  
298 depends on the true tree. Our simulation engine may be useful for understanding how variation in  
299 the target rates affects estimation accuracy under various conditions.

300 The double cut rate is similar across the fish. The rate is quite high: For the first adult fish, the  
301 double cut rate of 0.076 means that the probability of introducing a single-target indel as opposed  
302 to an inter-target indel in an unmodified barcode is 59%. To counter this, we can decrease the  
303 number of long inter-target deletions (and the number of masked events) by placing high cut-rate  
304 targets closer together in the barcode. One potentially helpful barcode design is to place the highest  
305 cut-rate targets in the center and the lowest cut-rate targets on the outside. Alternatively, designers  
306 could arrange the targets from highest to lowest cut rate. Table 1 shows that the current barcode  
307 design is counter to our suggestion, as the two targets with the highest cut rates in the two adult  
308 fish are targets 1 and 9.

309 The characterization of target efficiencies in a compact multi-target barcode is challenging  
310 problem. Our method can help steer the next generation of CRISPR-based lineage recording  
311 technologies to have increased recording capacity.

## 312 **Discussion**

313 In this manuscript, we have proposed a statistical model for the mutation process for GESTALT, a new  
314 cell lineage tracing technology that inserts a synthetic barcode composed of CRISPR/Cas9 targets  
315 into the embryo. Our method, GAPML, estimates the cell lineage tree and the mutation parameters  
316 from the sequenced modified barcode. Unlike existing methods, our method estimates branch  
317 lengths and the ordering between children nodes that share the same parent. We demonstrate  
318 that our method outperforms existing methods on simulated data, provides more consistent  
319 results across biological replicates, and outputs trees that better concord with our understanding  
320 of developmental biology. We have answered the question "Is it possible to reconstruct an accurate  
321 cell lineage tree using CRISPR barcodes?" in *Salvador-Martínez et al. [2018]* in the affirmative: The  
322 cell lineage tree can be estimated to a high degree of accuracy as long as appropriate methods are  
323 used.

324 Our method provides a number of technical contributions to the phylogenetics literature. The  
325 GESTALT mutation process violates many of the classical phylogenetic assumptions that existing  
326 methods rely upon for computational tractability. Thus we determined the most appropriate  
327 assumptions that are most suitable in this new setting and developed new methods so that the  
328 likelihood is computationally tractable and the estimated trees are accurate. We believe these  
329 techniques could be useful for other phylogenetic problems where the independent-site assumption  
330 does not hold. In addition, our methods may be useful as a jumping off point for analyzing other  
331 CRISPR-based cell lineage tracing technologies, such as that using homing CRISPR barcodes [*Kalhor*  
332 *et al., 2017, 2018*]. There are still many areas of improvements for the current method, such as  
333 quantifying the uncertainty of our estimates, estimating meta-properties about the cell lineage tree  
334 for organisms of the same species, and utilizing data sampled at multiple time points.

335 Finally, the biological results were relatively limited since the goal of this manuscript was  
336 primarily on methods development and the data in *McKenna et al. [2016]* only provide tissue

	Adult fish #1	Adult fish #2	3 day #1	30 hpf #5	4.3 hpf #1
Target 1	3.053	1.320	0.410	1.595	1.301
Target 2	1.232	0.317	0.155	0.697	0.474
Target 3	0.063	0.108	0.129	0.618	0.154
Target 4	1.234	0.821	0.223	0.561	0.399
Target 5	0.619	0.542	0.179	0.510	0.276
Target 6	1.329	0.652	0.182	1.155	0.385
Target 7	0.761	0.470	0.344	0.544	1.088
Target 8	0.090	0.136	0.141	1.171	0.151
Target 9	2.422	1.529	0.404	1.176	1.146
Target 10	0.285	0.371	0.132	1.150	0.155
Double cut rate	0.076	0.084	0.052	0.090	0.065
Left trim zero prob	0.015	0.028	0.015	0.258	0.025
Left trim length mean	6.330	6.634	5.165	12.000	6.405
Left trim length SD	4.956	5.184	4.245	6.633	4.998
Right trim zero prob	0.906	0.834	0.890	0.238	0.818
Right trim length mean	4.945	3.759	3.716	4.800	3.478
Right trim length SD	6.173	5.534	5.529	4.011	5.360
Insertion zero prob	0.400	0.411	0.401	0.520	0.419
Insertion length mean	5.085	4.540	5.786	5.446	4.589
Insertion length SD	5.798	5.533	5.219	7.291	4.708

**Table 1.** 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.

337 source information. In future work, we plan to apply our method to analyze data where each allele  
338 is paired with much richer information, such as single-cell gene expression data [*Raj et al., 2018*].

## 339 **Materials and Methods**

### 340 **Data**

341 The data processed in this paper are all from *McKenna et al. [2016]* and are available at the Gene  
342 Expression Omnibus under GSE81713. We use the aligned data where each allele was described  
343 with the observed insertion/deletions (indels) at each target. Each CRISPR target can only be  
344 modified once and indels can only be introduced via a double-stranded break at a target cut site,  
345 so we further processed the aligned data accordingly: merging indels if there were more than one  
346 associated with a given target, and extending the deletion lengths and insertion sequences so that  
347 a target cut site was nested within each indel. For this paper, we assume that the processed data is  
348 correct and do not attempt to model the effects of alignment error.

### 349 **Methodological contributions**

350 Here we highlight the key methodological contributions that we needed to develop in order to  
351 analyze GESTALT. We needed to develop new methodology because the GESTALT mutation process  
352 violates many classical assumptions in phylogenetics.

353 Since GESTALT is a new technology, we introduce new mathematical abstractions for the biologi-  
354 cal process. We then consider all statistical models that satisfy our proposed set of assumptions. We  
355 carefully designed these assumptions to balance biological realism and computationally feasibility.  
356 We next show that such models are “lumpable” and use this to efficiently calculate the likelihood.  
357 Even though lumpability has been used to reduce computation time for general Markov chains, this  
358 idea is rarely used in phylogenetics. We show how lumpability can be combined with Felsenstein’s  
359 algorithm if the mutation process is irreversible.

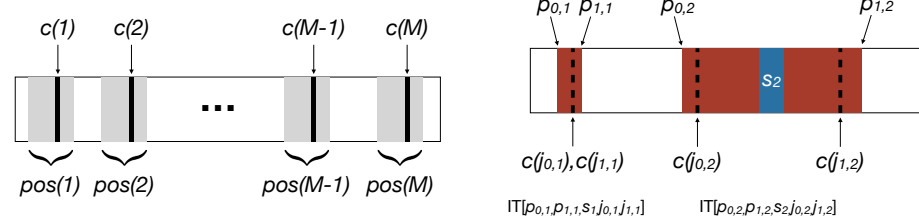
360 In addition, our method estimates trees at a finer resolution compared to other methods, which  
361 leads to better tree estimates. In particular, we resolve the multifurcations as a caterpillar tree and  
362 show how to efficiently search over caterpillar tree topologies. As far as we know, this is one of the  
363 few methods that tunes the tree topology, which is typically treated as a combinatorial optimization  
364 problem, by solving a continuous optimization problem. The closest equivalent appears in the  
365 Bayesian phylogenetics literature where local changes to the topology may be introduced via a  
366 Subtree-Slide operator [*Hohna et al., 2008*].

367 To handle the small number of barcodes, we improve the stability of the method by maximizing  
368 the *penalized* log likelihood. Previous phylogenetic methods that penalize branch lengths assume  
369 that the tree topology is known [*Kim and Sanderson, 2008, Zhang et al., 2018*]. However, the tree  
370 topology is unknown in GESTALT. We show how penalization can be combined with tree topology  
371 search methods. Combining the two is not trivial since a naïve approach will bias the search towards  
372 incorrect topologies.

373 Finally, we use an automatic differentiation framework to optimize the phylogenetic likelihood.  
374 Automatic differentiation software has accelerated deep learning research since they allow re-  
375 searchers to quickly iterate on their models [*Baydin et al., 2018*]. Likewise, we found that this tool  
376 greatly accelerated our progress and, through this experiment, we believe that this tool may greatly  
377 accelerate the development of maximum likelihood estimation methods in phylogenetics.

### 378 **GESTALT framework and definitions**

379 In this section, we present mathematical definitions for the many components in GESTALT, though  
380 begin in this paragraph by giving an overview in words. We begin with defining the *barcode* and  
381 the individual *targets* within it. A barcode is mutated when nucleotides are inserted and/or deleted,  
382 which is referred to as an *indel tract*. We then define a possibly-mutated barcode or *allele* as a  
383 collection of the observed indel tracts. Finally, we use all these abstractions to define the barcode



(a) A barcode with  $M$  targets. The cut site of the targets,  $c(j)$  for  $j = 1, \dots, M$ , are shown by the bolded black lines. The positions associated with each target are highlighted in the gray boxes.

(b) An example allele with two indel tracts. The left indel tract was introduced by a cut at a single target and does not have an insertion, i.e.  $s_1 = \emptyset$ . The right indel tract was introduced by cuts at two targets. Red indicates deletion and blue indicates insertion.

**Figure 9.** Illustration of GESTALT definitions

384 mutation process, which is framed as stochastic process where the state space corresponds to all  
 385 possible alleles and the transitions correspond to indel tracts. To aid the reader, Table 2 briefly  
 386 summarizes the definitions used in the paper.

387 The unmodified barcode is a nucleotide sequence where  $M$  disjoint subsequences are des-  
 388 ignated as targets (Figure 9a). The targets are numbered from 1 to  $M$  from left to right and the  
 389 positions spanned by target  $j$  are specified by the set  $\text{pos}(j)$ . Each target  $j$  is associated with a single  
 390 cut site  $c(j) \in \text{pos}(j)$ .

391 A barcode can be modified by the introduction of an indel tract. An indel tract, denoted by  
 392  $\text{IT}[p_0, p_1, s, j_0, j_1]$ , is a mutation event in which targets  $j_0$  and  $j_1$  are cut ( $j_0 \leq j_1$ ), positions  $p_0, p_0 +$   
 393  $1, \dots, p_1 - 1$  in the unmodified barcode are deleted, and a nucleotide sequence  $s$  is inserted. If  $j_0 = j_1$ ,  
 394 only a single target is cut. When  $p_0 = p_1$ , then no positions are deleted. If  $s$  is of length zero, then no  
 395 nucleotides are inserted. We only consider indel tracts that modify the sequence, i.e. either  $p_0 < p_1$   
 396 or  $s$  has positive length, and nest at least one target cut site between positions  $p_0$  and  $p_1$ .

A possibly-modified barcode, also referred to as an allele, is a sequence of disjoint indel tracts associated with a single barcode (Figure 9b)

$$a \equiv \{ \text{IT}[p_{0,k}, p_{1,k}, s_k, j_{0,k}, j_{1,k}] : k \in \{1, \dots, m\} \} \quad (1)$$

397 where  $m \geq 0$  and  $p_{1,k} < p_{0,k+1}$  for  $k = 1, \dots, m - 1$ . The positions  $p_{0,k}, p_{1,k}$  in the indel tracts always refer  
 398 to the positions in the original unmodified barcode: deletions and insertions do not change the  
 399 indexing. Let  $\Omega$  be the set of all possible alleles.

A target  $j$  is active in allele  $a$  if none of the nucleotides in  $\text{pos}(j)$  are modified. That is, the status  
 of target  $j$  in allele  $a$  is

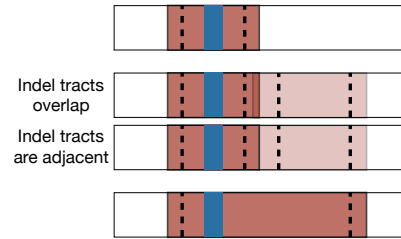
$$\text{TargStat}(j; a) = \mathbb{1} \{ \exists \text{IT}[p_0, p_1, s, j'_0, j'_1] \in a \text{ and } \exists k \in \text{pos}(j) \text{ s.t. } p_0 \leq k \leq p_1 \}.$$

So  $\text{TargStat}(j; a)$  is 0 if target  $j$  is active and 1 if it is inactive. For convenience, denote the target  
 status of allele  $a$  as

$$\text{TargStat}(a) = (\text{TargStat}(1; a), \dots, \text{TargStat}(M; a)). \quad (2)$$

400 We now introduce the rules governing how alleles change through the introduction of indel  
 401 tracts. First, transitions between possible allele states  $\Omega$  are constrained by the status of the targets.  
 402 For a given allele  $a$ , we can introduce the indel tract  $d = \text{IT}[p_0, p_1, s, j_0, j_1]$  if and only if targets  $j_0$  and





**Figure 10.** Possible transitions from the top allele and the bottom allele: either an indel tract is introduced that overlaps with an existing indel tract (middle top) or an adjacent indel tract is introduced (middle bottom).

403  $j_1$  are active. Note that the set of transitions allowed under this rule is a superset of biologically-  
 404 plausible transitions. For example, even if position  $p_0$  is deleted, introducing indel tract  $d$  is formally  
 405 allowed in our scheme. However in order to exclude biologically-implausible transitions, our models  
 406 assign near-zero probability to such transitions.

407 Let  $\text{Apply}(a, d)$  be the resulting allele from introducing indel tract  $d$  into allele  $a$ . If indel tract  $d$   
 408 does not overlap any other indel tract in  $a$ , then  $\text{Apply}(a, d)$  is simply the union  $a \cup \{d\}$ . If  $d$  completely  
 409 masks other indel tracts, i.e. all indel tracts in  $a$  are either completely within the range of  $p_0$  to  $p_1 - 1$   
 410 or are completely outside of the range, then  $\text{Apply}(a, d)$  is the resulting allele after removing the  
 411 masked indel tracts and adding  $d$ . The last possibility is that  $d$  is adjacent or overlaps, but does  
 412 not fully mask, other indel tract(s) in  $a$ ; then  $\text{Apply}(a, d)$  is the resulting allele after properly merging  
 413 overlapping indel tracts (Figure 10). From a biological perspective, it is impossible to introduce  
 414 overlapping but non-masking indel tracts. However the likelihood model is much simpler if we  
 415 allow such events to happen. Since deletion lengths tend to be short in the empirical data (75%  
 416 quantile = 10, *McKenna et al. [2016]*), we will estimate that long deletion lengths occur with small  
 417 probability, which means that these overlapping but non-masking indel tracts have very small  
 418 probabilities. Thus we believe our model closely approximates the GESTALT mutation process. We  
 419 will also discuss an assumption that removes these overlapping indel tracts from the likelihood  
 420 calculation entirely in more detail in the following section.

## 421 **GESTALT Model and Assumptions**

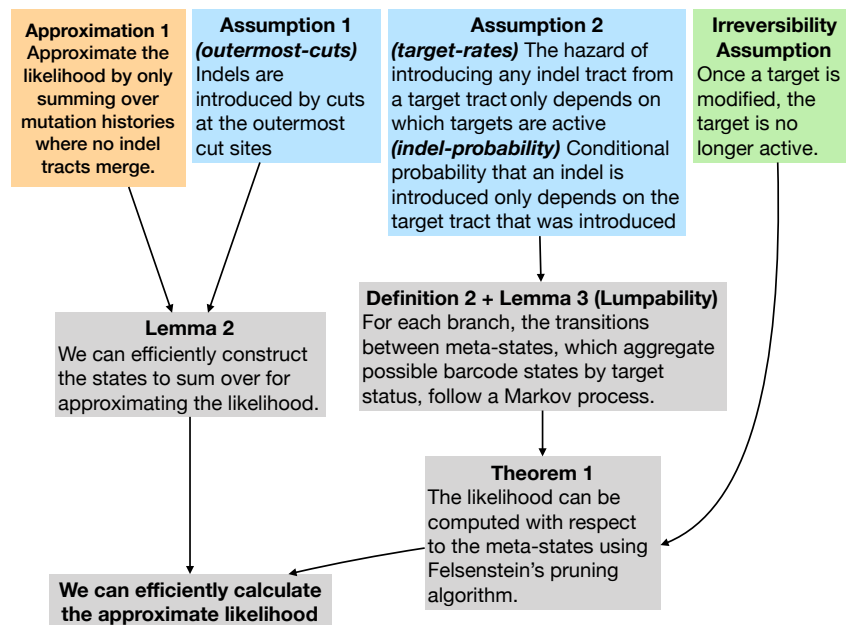
422 Here we define the GESTALT model more concretely and formalize the model assumptions pre-  
 423 sented in the Introduction. Recall the three model assumptions. First, as we rarely observe very  
 424 long deletion lengths in the GESTALT data, we assume that indel tracts are introduced by cuts  
 425 at the outermost cut sites (assumption A). In addition, based on our biological understanding  
 426 of the CRISPR/Cas9 mutation process, we assume that mutations occur in two stages: first the  
 427 targets are cut according to the rates of the active targets (assumption B) and then nucleotides are  
 428 deleted/inserted according to a process that only depends on which targets were cut (assumption  
 429 C). Figure 11 presents a flowchart of how the assumptions are used to derive later results.

430 The barcode mutation process up to time  $T$  is formulated as a continuous time Markov chain  
 431  $\{X(t) : 0 \leq t \leq T\}$  with state space  $\Omega$ . As mentioned before, the choice of using state space  $\Omega$   
 432 implicitly assumes that indel tracts are introduced instantaneously, i.e. nucleotides are inserted  
 433 and/or deleted immediately after target(s) are cut.

434 For a given tree  $\mathbb{T}$ , let  $t_{\mathbb{N}}$  be the length of the branch ending with node  $\mathbb{N}$  and let  $\{X_{\mathbb{N}}(t) : 0 \leq t \leq t_{\mathbb{N}}\}$   
 435 be the Markov process along the branch. In addition, let  $a_{\mathbb{N}}$  be the allele observed at that leaf, and  
 436 let  $\text{Leaves}(\mathbb{N})$  denote all leaves with ancestral node  $\mathbb{N}$ . The set of leaves in the entire tree  $\mathbb{T}$  is denoted  
 437  $\text{Leaves}(\mathbb{T})$ . As notational shorthand, we denote the Markov process for the branches with end nodes  
 438 in the set  $S$  as  $X_S$ . In addition, the observed alleles in the leaf set  $S$  are denoted  $a_S$ .

Symbol	Description	Eq.
$X_N(t)$	Markov process along branch ending with node $N$	
$X_S(t)$	Markov process along branches ending with nodes in set $S$	
$a_N$	Observed allele at leaf $N$	
$a_S$	Observed alleles at leaves in set $S$	
$\text{pos}(j)$	Positions of target $j$ in the unmodified barcode	
$c(j)$	Cut site of target $j$	
$\text{Leaves}(N)$	Leaves of node $N$	
$\text{Desc}(N)$	Descendants of node $N$	
$\text{TargStat}(a)$	Status of targets in allele $a$ . 1 in position $j$ indicates target $j$ is inactive	(2)
$\text{IT}[p_0, p_1, s, j_0, j_1]$	Indel tract that cuts targets $j_0$ and $j_1$ , deletes positions $p_0$ to $p_1 - 1$ , inserts $s$	
$\text{TT}[j'_0, j_0, j_1, j'_1]$	Target tract: all indel tracts that cut targets $j_0$ and $j_1$ and deactivate targets $j'_0$ to $j'_1$	(5)
$\text{TT}(d)$	The target tract that indel tract $d$ belongs to	
$\text{WC}[j_0, j_1]$	Wildcard: any indel tract that only deactivates targets with indices $j_0$ to $j_1$	(8)
$\text{SGWC}[p_0, p_1, s, j_0, j_1]$	Singleton-wildcard: union of an indel tract and its inner wildcard	(9)
$\text{AncState}(N)$	The set of likely ancestral states for node $N$	(7)

**Table 2.** Notation used in this paper



**Figure 11.** A guide for how assumptions, approximations, and derived results connect and lead to our final algorithm for approximating the likelihood. The flowchart also maps the formal assumptions here to Assumptions A, B, and C in the introduction.

439 If there are multiple barcodes, we use the notation  $X_N^{(i)}(\cdot)$  to represent the process for the  $i$ th  
 440 barcode and  $a_N^{(i)}$  to represent the allele observed on the  $i$ th barcode. In the manuscript, we assume  
 441 the barcodes are independently and identically distributed. Therefore we will typically discuss the  
 442 model and assumptions in the context of a single barcode and omit the index of the barcode.

443 Unfortunately, calculating the likelihood of the tree for a general model where the mutation  
 444 rate depends on the entire sequence is computationally intractable: We would need to compute  
 445 the transition rates between an infinite number of barcode states. Instead we introduce a model  
 446 assumption so that we can aggregate the possible barcode states into lumped states indexed by  
 447 target statuses. Then we can compute the likelihood using transition matrices of dimension at  
 448 most  $2^M \times 2^M$ . As  $M$  is typically small ( $M = 10$  in *McKenna et al. [2016]*), the assumption makes the  
 449 likelihood computationally feasible.

450 First we formalize the *outermost-cuts* assumption, which states that the cuts for an indel tract  
 451 occur at the outermost cut sites. We define this mathematically by requiring that for any indel tract  
 452 where targets  $j_0$  and  $j_1$  are cut, the deletions to the left and right cannot extend past the cut site of  
 453 the neighboring targets  $j_0 - 1$  and  $j_1 + 1$ .

**Assumption 1** (*outermost-cuts*). All indel tracts are of the form  $IT[p_0, p_1, s, j_0, j_1]$  where

$$\begin{aligned} c(j_0 - 1) < p_0 \leq c(j_0) & \quad \text{if } j_0 \geq 1 \\ c(j_1) \leq p_1 < c(j_1 + 1) & \quad \text{if } j_1 \leq M. \end{aligned}$$

454

This assumption limits the possible mutation histories of the alleles. Note that Assumption 1 still  
 allows indel tracts to deactivate targets immediately neighboring the cut site. That is, an indel tract  
 $d = IT[p_0, p_1, s, j_0, j_1]$  can either have a *short* deletion to the left so that target  $j_0 - 1$  is unaffected or a  
*long* deletion to the left such that target  $j_0 - 1$  is deactivated, i.e.

$$d \text{ has a short left deletion if } p_0 \in \text{pos}(j_0) \tag{3}$$

$$d \text{ has a long left deletion if } p_0 \in \text{pos}(j_0 - 1) \tag{4}$$

455 We can have similar short and long deletions to the right.

For the second assumption, let us introduce the concept of a target tract, which is a set of  
 indel tracts that cut the same target(s) and deactivate the same target(s). A target tract, denoted  
 $TT[j'_0, j_0, j_1, j'_1]$ , 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.

$$TT[j'_0, j_0, j_1, j'_1] = \{IT[p_0, p_1, s, j_0, j_1] : p_0 \in \text{pos}(j'_0), p_1 \in \text{pos}(j'_1)\}. \tag{5}$$

456 Note that we always have that  $j'_0 \leq j_0 \leq j_1 \leq j'_1$ . Each indel tract  $d$  belongs to a single target tract; we  
 457 denote its associated target tract by  $TT(d)$ .

458 This second assumption decomposes the mutation process into a two-step process where  
 459 targets are cut and then indels are introduced; and combines the *target-rate* and *indel-probability*  
 460 assumptions. In particular, the assumption states that the instantaneous rate of introducing an  
 461 indel tract can be factorized into the rate of introducing any indel tract from a target tract, which  
 462 depends on the target status of the current allele, and the conditional probability of introducing an  
 463 indel tract which only depends on the target tract.

**Assumption 2** (*rate-rate, indel-probability*). Let  $a$  be any allele,  $d$  be any indel tract that can be  
 introduced into  $a$ , and  $\tau$  be the target tract  $TT(d)$ . The instantaneous rate of introducing indel tract  $d$  in  
 allele  $X(t) = a$  can be factorized into two terms: first, a function  $h$  that only depends on  $\tau$ ,  $\text{TargStat}(a)$ ,  
 and time  $t$ , then second, the conditional probability of introducing  $d$  given  $\tau$ :

$$\begin{aligned} q(a, \text{Apply}(a, d), t) & := \lim_{\Delta \rightarrow 0} \frac{\Pr(X(t + \Delta) = \text{Apply}(a, d) | X(t) = a)}{\Delta} \\ & = h(\tau, \text{TargStat}(a), t) \Pr(d | \tau). \end{aligned}$$

464

465 Thus the instantaneous rate of introducing  $d$  only depends the allele  $a$  through its target status.  
 466 Using this assumption, we will show that the mutation process is equivalent to a continuous time  
 467 Markov chain where we lump together possible allele states that share the same target status.

### 468 Likelihood approximation: summing over likely ancestral states

469 The likelihood of a given tree and mutation parameters is the sum of the probability of the data  
 470 over all possible mutation histories. There are many possible ancestral states since inter-target  
 471 deletions can mask previously-introduced indel tracts. In this section, we present an approximation  
 472 of the likelihood that only sums over the likely ancestral states and ignores those with very small  
 473 probabilities. We also provide a simple algorithm that efficiently specifies the set of these likely  
 474 states, which is useful when we actually implement the (approximate) likelihood calculation.

475 We first address the problem that inter-target indel tracts have too many possible histories  
 476 for brute-force computation. Not only can inter-target deletions mask previously-introduced  
 477 indel tracts, but they can also result from merging overlapping multiple indel tracts. To simplify  
 478 the likelihood calculation, we ignore any mutation history where indel tracts merge. We are  
 479 motivated to do so by observing that indel tracts rarely merge – merging only occurs when many  
 480 nucleotides are deleted whereas we mostly observe short deletions in the experimental data. Thus,  
 481 mutation histories involving merging indel tracts probably contribute a negligible fraction to the  
 482 total likelihood. The approximation is therefore as follows:

**Approximation 1.** *We approximate the likelihood with the probability of the data over all possible mutation histories where no two indel tracts are ever merged:*

$$\Pr (X_{\text{Leaves}(\mathbb{N})}(T) = a_{\text{Leaves}(\mathbb{N})}) \approx \Pr (X_{\text{Leaves}(\mathbb{N})}(T) = a_{\text{Leaves}(\mathbb{N})}, \text{no indel tracts merged}). \quad (6)$$

483

484 Note that this approximation strictly lower bounds the full likelihood since we sum over a subset of  
 485 the possible histories.

Based on Approximation 1, we define a partial ordering between alleles. Given two alleles  $a, a' \in \Omega$ , we use  $a' \geq a$  to indicate that  $a$  can transition to the allele  $a'$  without merging indel tracts, i.e. there is a sequence of indel tracts  $\{d_i\}_{i=1}^m$  such that

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

486 where no indel tracts merge.

Now we present an algorithm to concisely express the set of alleles that are summed over at each internal node under Approximation 1. For the tree  $\mathbb{T}$  and observed alleles  $a_{\text{Leaves}(\mathbb{T})}$ , define the set of likely ancestral states at each internal node  $\mathbb{N}$  as

$$\text{AncState}(\mathbb{N}; \mathbb{T}, a_{\text{Leaves}(\mathbb{T})}) = \{a \in \Omega : a \leq a_L \forall L \in \text{Leaves}(\mathbb{N})\}. \quad (7)$$

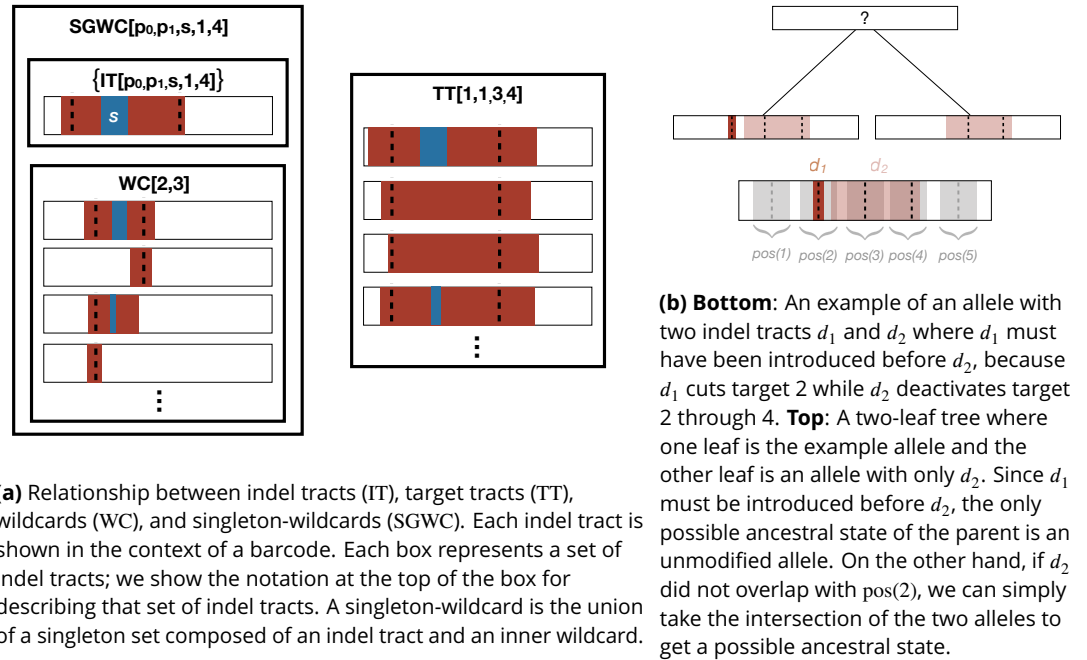
487 We define  $\text{AncState}(\cdot)$  over leaf nodes in the same way as (7), but we interpret this set as the  
 488 alleles that likely preceded the observed allele. Henceforth, we use the shorthand notation  
 489  $\text{AncState}(\mathbb{N}) \equiv \text{AncState}(\mathbb{N}; \mathbb{T}, a_{\text{Leaves}(\mathbb{T})})$  whenever the context is clear. The approximate likelihood  
 490 from Approximation 1 is equal to summing over  $\text{AncState}(\mathbb{N})$  at each internal node  $\mathbb{N}$ .

491 To construct the sets  $\text{AncState}(\mathbb{N})$ , we also need sets of indel tracts of the following forms  
 492 (Figure 12a):

- wildcard<sup>2</sup>  $\text{WC}[j_0, j_1]$ : the set of all indel tracts that only deactivate targets within the range  $j_0$  to  $j_1$

$$\text{WC}[j_0, j_1] = \{\text{IT}[p'_0, p'_1, s', j'_0, j'_1] : \text{pos}(j_0 - 1) < p'_0, p'_1 < \text{pos}(j_1 + 1)\} \quad (8)$$

<sup>2</sup>In software systems, a wildcard is a symbol used to represent one or more characters (e.g. “\*”). Similarly, we define wildcard here as all indel tracts that only deactivate targets within a specified range.



**Figure 12.** Visual dictionary of indel tract sets (a) and example of ordering between indel tracts in an allele (b).

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

$$SGWC[p_0, p_1, s, j_0, j_1] = \begin{cases} \{IT[p_0, p_1, s, j_0, j_1]\} \cup WC[j_0 + 1, j_1 - 1] & \text{if } j_0 + 1 \leq j_1 - 1 \\ \{IT[p_0, p_1, s, j_0, j_1]\} & \text{otherwise} \end{cases} \quad (9)$$

493 The singleton of singleton-wildcard (9) refers to  $\{IT[p_0, p_1, s, j_0, j_1]\}$  and the inner wildcard of a  
494 singleton-wildcard refers to  $WC[j_0 + 1, j_1 - 1]$  if it exists and  $\emptyset$  otherwise.

495 Two singleton-wildcards are disjoint if the range of their positions don't overlap. A wildcard is  
496 disjoint from a wildcard/singleton-wildcard if the range of their targets don't 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  $D$ , i.e.  $Alleles(D)$  is

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

We are interested in wildcards and singleton-wildcards because for any leaf node  $L$  with indel tracts  $IT[p_{0,m}, p_{1,m}, s_m, j_{0,m}, j_{1,m}]$  for  $m = 1, \dots, M_L$ , a superset of  $AncState(L)$  is the alleles generated by the union of their corresponding singleton-wildcards, i.e.

$$AncState(L) \subseteq Alleles \left( \bigcup_{m=1, \dots, M_L} SGWC[p_{0,m}, p_{1,m}, s_m, j_{0,m}, j_{1,m}] \right). \quad (10)$$

497 In fact, the following lemma states that we can use a recursive algorithm to compute supersets  
498 of  $AncState(N)$ . The algorithm starts at the leaves and proceeds up towards the root. We first use  
499 (10) to compute supersets of  $AncState(L)$  for all leaf nodes  $L$ . Now consider any internal node  $N$   
500 with two (direct) children nodes  $C_1$  and  $C_2$ . For mathematical induction, suppose that we have  
501 already computed supersets of  $AncState(C_i)$  for  $i = 1$  and  $2$  that are the alleles generated by unions  
502 of wildcards/singleton-wildcards. We compute the superset of  $AncState(N)$  as the alleles generated

503 by the nonempty pairwise intersections of wildcards and/or singleton-wildcards corresponding  
 504 to  $\text{AncState}(C_1)$  and  $\text{AncState}(C_2)$ . Since the intersection of wildcards and/or singleton-wildcards  
 505 is always a wildcard or singleton-wildcard, we can also write this superset as the alleles gener-  
 506 ated by the union of disjoint wildcards and/or singleton-wildcards. We can repeatedly intersect  
 507 wildcards/singleton-wildcards for internal nodes with multiple children nodes. The proof for the  
 508 lemma is straightforward so we omit it here.

**Lemma 1.** Consider any internal node  $N$  with children nodes  $C_1, \dots, C_K$ . Suppose for each child  $C_k$ , we have that

$$\text{AncState}(C_k) \subseteq \text{Alleles} \left( \bigcup_{m=1}^{M_{C_k}} D_{C_k,m} \right) \quad (11)$$

where  $\{D_{C_k,m}\}_{m=1}^{M_{C_k}}$  are pairwise disjoint wildcards and/or singleton-wildcards. Then, we can also write  $\text{AncState}(N)$  in the form of (11) where  $\{D_{N,m}\}_{m=1}^{M_N}$  are the non-empty intersections of  $D_{C_1,m_1} \cap \dots \cap D_{C_K,m_K}$ , i.e.

$$\{D_{N,m}\}_{m=1}^{M_N} = \{D' = D_{C_1,i_1} \cap \dots \cap D_{C_K,m_K} : D' \neq \emptyset, m_1 \in \{1, \dots, M_{C_1}\}, \dots, m_K \in \{1, \dots, M_{C_K}\}\}. \quad (12)$$

For efficiency reasons, we are not satisfied with computing supersets of  $\text{AncState}(\cdot)$ ; rather, we would like to concisely express the set of alleles that is exactly equal to  $\text{AncState}(\cdot)$ . The only case in which the algorithm computes a strict superset of  $\text{AncState}(N)$  is when the alleles observed at the leaves of  $N$  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  $d_1$  must be introduced before  $d_2$  if  $d_1$  cuts target  $j$  and  $d_2$  deactivates target  $j$  (Figure 12b). Due to this ordering, we may find that the same indel tract observed in two alleles must have been introduced independently (also known as homoplasmy in the phylogenetics literature). 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

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

509 Per this definition,  $\text{Order}(d \Rightarrow d')$  contains all alleles that do not include  $d$ .

510 The following lemma builds on Lemma 1 and computes the sets exactly equal to  $\text{AncState}(\cdot)$ . The  
 511 algorithm is similar as before, but the parent nodes also adopt ordering requirements from their  
 512 children nodes. Note that ordering requirements only ever involve observed indel tracts. Again, the  
 513 proof for the lemma is straightforward so we omit it here.

**Lemma 2.** For any leaf node  $L$ , suppose its observed allele is  $\{d_m : m = 1, \dots, M_L\}$  for some  $M_L \geq 0$ , where  $d_m = \text{IT}[p_{0,m}, p_{1,m}, s, j_{0,m}, j_{1,m}]$ . Denote its list of ordering requirements as

$$\text{Orderlist}_L = \left\{ \text{Order}(d_m \Rightarrow d_{m'}) : m, m' \in \{1, \dots, M_L\}, m \neq m', p_{1,m} \in \text{pos}(j_{0,m'}) \text{ or } p_{0,m} \in \text{pos}(j_{1,m'}) \right\}.$$

Then

$$\text{AncState}(L) = \text{Alleles} \left( \bigcup_{m=1}^{M_L} D_{L,m} \right) \cap \left[ \bigcap_{\text{Order}(d \Rightarrow d') \in \text{Orderlist}_L} \text{Order}(d \Rightarrow d') \right] \quad (13)$$

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

Similarly, for any internal node  $N$ , we can also write  $\text{AncState}(N)$  in the form of (13). If node  $N$  has children nodes  $C_1, \dots, C_K$ ,  $\{D_{N,m}\}_{m=1}^{M_N}$  are pairwise disjoint wildcards and/or singleton-wildcards satisfying (12) and

$$\text{Orderlist}_N = \left\{ \text{Order}(d \Rightarrow d') \in \left[ \bigcup_{k=1}^K \text{Orderlist}_{C_k} \right] : d \in \left[ \bigcup_{m=1}^{M_N} D_{N,m} \right] \right\}.$$

515 Now that we've shown that  $\text{AncState}(N)$  can be written in terms of disjoint wildcards and singleton-  
 516 wildcards, we introduce one more notation that will be useful later. Define  $\text{SG}(N)$  to be the singletons  
 517 from the singleton-wildcards in  $\text{AncState}(N)$ .

518 **Likelihood calculation: aggregating states**

519 Here we show how to use the concept of “lumpability” to calculate the approximate likelihood (6),  
 520 even when there are an infinite number of ancestral states. Recall that we previously proposed  
 521 calculating (6), which sums over a subset of all possible ancestral states. Though this has decreased  
 522 the set of states to sum over, the number of ancestral states under consideration at each tree node  
 523 is still infinite. Even if we ignore the insertion sequences, the number of possible ancestral states  
 524 still grows at a rate of  $O(p^2)$  where  $p$  is the number of positions in the unmodified barcode and the  
 525 likelihood calculation has complexity  $O(p^6)$  since we need to construct transition matrices. Since  
 526 the barcode is 300 nucleotides long in *McKenna et al. [2016]*, we cannot calculate the likelihood by  
 527 considering all possible states separately.

528 To handle Markov processes with infinite state spaces, one solution is to partition the states into  
 529 lumped states and show that the behavior of the original Markov process is equivalent to that of an  
 530 aggregate Markov process over the lumped states [*Kemeny and Laurie Snell, 1976, Hillston, 1995*].  
 531 This property, called “lumpability,” is defined as follows.

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

$$\Pr(X(t) \in A_i) = \Pr(Y(t) = A_i) \quad \forall i = 1, \dots, M,$$

532 then  $X$  is lumpable.

533 Although lumpability is a well-established technique for Markov chains, the practical difficulty is  
 534 typically in constructing the appropriate partition [*Ganguly et al., 2014*].

There is relatively little work on applying these ideas of lumpability in phylogenetics. (The one  
 application in *Davydov et al. [2017]* calculates the likelihood of a codon model approximately by  
 assuming states are lumpable, even though this is not necessarily true in their model; here we will  
 show that the states are indeed lumpable.) Here we extend lumpability to the phylogenetics setting  
 where we have different partitions of the state space at each tree node. In particular, for some  
 indexing set  $B$ , define a partition  $\{g(b; N) : b \in B\}$  of  $\Omega$  at every node  $N$ . Lumpability is only useful for  
 efficient phylogenetic likelihood computation if these partitions are compatible with Felsenstein’s  
 pruning algorithm [*Felsenstein, 1981*]. For any  $b \in B$  and node  $N$ , let  $p_N(b)$  be the component of the  
 likelihood for the subtree below  $N$  for states in partition  $b$ :

$$p_N(b) = \Pr(X_{\text{Leaves}(N)}(T) = a_{\text{Leaves}(N)} \mid X_N(t_N) \in g(b; N)) \quad (14)$$

By Felsenstein’s algorithm, we have

$$p_N(b) = \prod_{C \in \text{children}(N)} \left\{ \sum_{b' \in B} p_C(b') \Pr(X_C(t_C) \in g(b'; C) \mid X_C(0) \in g(b; N)) \right\}. \quad (15)$$

For lumpability to be useful, we must be able to show that there exists an easy-to-compute weight  
 function  $w(b, N, b', C)$  such that

$$p_N(b) = \prod_{C \in \text{children}(N)} \left\{ \sum_{b' \in B} p_C(b') w(b, N, b', C) \Pr(X_C(t_C) \in g(b'; C) \mid X_C(0) \in g(b; C)) \right\} \quad (16)$$

535 and that for each tree node  $C$ ,  $X_C(\cdot)$  is indeed lumpable over the partition  $g(\cdot; C)$ . Obviously if the  
 536 partitions are the same across all tree nodes, then we can just set all weights to one. However we  
 537 will need to construct a different partition for each tree node for the GESTALT likelihood.

538 We propose partitioning  $\Omega$  at tree node  $N$  based on whether or not the allele is a likely ancestral  
 539 state (i.e. is it in  $\text{AncState}(N)$ ) and its target status (Figure 2):

540 **Definition 2.** Define the indexing set  $B$  to be  $\{0, 1\}^M \cup \{\text{other}\}$ .

For internal tree node  $N$ , partition the state space  $\Omega$  into

$$\begin{cases} g(b; N) = \{a \in \text{AncState}(N) : \text{TargStat}(a) = b\} \quad \forall b \in \{0, 1\}^M \\ g(\text{other}; N) = \Omega - \text{AncState}(N). \end{cases} \quad (17)$$

For leaf node  $N$ , partition the state space  $\Omega$  into

$$\begin{cases} g(b; N) = \{a_N\} & \text{if } b = \text{TargStat}(a_N) \\ g(b; N) = \emptyset & \text{if } b \in \{0, 1\}^M \text{ and } b \neq \text{TargStat}(a_N) \\ g(\text{other}; N) = \Omega - \{a_N\}. \end{cases} \quad (18)$$

541

542 To prove that the Markov process over the branch with end node  $N$  is lumpable with respect to  
 543 the proposed partition, we show that the instantaneous transition rate from any allele in  $g(b; N)$  to  
 544 the set  $g(b'; N)$  is the same. Therefore we use  $q_{\text{lump}}$  to denote the transition rates between the lumped  
 545 states  $\{g(b; N)\}$ . The results show that there are two types of transitions between the lumped states,  
 546 which determines the appropriate formula for calculating  $q_{\text{lump}}$ . Either the transition corresponds to  
 547 an observed indel tract and there is only one indel tract that is a valid for transitioning between the  
 548 lumped states; or the transition corresponds to a masked indel tract, in which case all indel tracts  
 549 from the possible target tracts are valid transitions between the lumped states.

**Lemma 3.** *Suppose Assumption 2 holds. Consider any branch with child node  $C$ , and target statuses  $b, b' \in \{0, 1\}^M$  where the sets  $g(b; C)$  and  $g(b'; C)$  are nonempty. For any alleles  $a, a' \in g(b; C)$ , we have*

$$\begin{aligned} & q_{\text{lump}}(g(b; C), g(b'; C), t) \\ &= \lim_{\Delta \rightarrow 0} \frac{\Pr(X_C(t + \Delta) \in g(b'; C) | X_C(t) = a)}{\Delta} \\ &= \lim_{\Delta \rightarrow 0} \frac{\Pr(X_C(t + \Delta) \in g(b'; C) | X_C(t) = a')}{\Delta}. \end{aligned} \quad (19)$$

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

$$q_{\text{lump}}(g(b; C), g(b'; C), t) = h(\text{TT}(d), b, t) \Pr(d | \text{TT}(d))$$

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

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

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

$$\begin{aligned} \lim_{\Delta \rightarrow 0} \frac{\Pr(X(t + \Delta) \in g(b'; C) | X(t) = a)}{\Delta} &= \sum_{a' \in g(b'; C)} q(a, a', t) \\ &= \sum_{d: \text{Apply}(d, a) \in g(b'; N)} h(\tau, b, t) \Pr(d | \tau). \end{aligned}$$

550 If  $d$  is an indel tract that can be introduced to the allele  $a \in g(b; C)$  and  $\text{Apply}(a, d)$  has target status  
 551  $b'$ , then we can introduce the same indel tract to any other allele  $a' \in g(b; C)$  and  $\text{Apply}(a', d)$  will also  
 552 have the target status  $b'$ . Therefore we have proven that (19) must hold for all  $a, a' \in g(b; C)$ .

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

$$q_{\text{lump}}(g(b; C), g(b'; C), t) = \sum_{\substack{\tau: \exists d \in \tau = \text{TT}(d) \\ \text{s.t. } \text{Apply}(d, a) \in g(b'; N)}} \left\{ \sum_{d \in \tau: \text{Apply}(d, a) \in g(b'; N)} h(\tau, b, t) \Pr(d | \tau) \right\}. \quad (20)$$

553 One of the following two cases must be true:



- 554 1. From the decomposition (13) of  $\text{AncState}(\mathcal{C})$ , there is only one indel tract  $d$  in the sets  $D_{\mathcal{C},m}$  for  
 555  $m = 1, \dots, M_{\mathcal{C}}$  such that  $\text{Apply}(a, d) \in g(b', \mathcal{C})$  for all  $a \in g(b, \mathcal{C})$ .  $d$  cannot be from a wildcard or  
 556 the inner wildcard of a singleton-wildcard since this would contradict the fact that  $d$  is the only  
 557 indel tract in  $\{D_{\mathcal{C},m}\}$  such that  $\text{Apply}(a, d) \in g(b', \mathcal{C})$  for all  $a \in g(b, \mathcal{C})$ . Therefore  $d$  must be the  
 558 singleton for some singleton-wildcard  $D_{\mathcal{C},m}$ . In other words, the only possible transition from  
 559  $g(b; \mathcal{C})$  to  $g(b'; \mathcal{C})$  is via the indel tract  $d$ .
- 560 2. Otherwise, for some target tract  $\tau$ , there are at least two indel tracts in  $d, d' \in \tau$  in the  
 561 sets  $D_{\mathcal{C},m}$  for  $m = 1, \dots, M_{\mathcal{C}}$  that deactivate targets  $b' \setminus b$  such that  $\text{Apply}(a, d) \in g(b', \mathcal{C})$  and  
 562  $\text{Apply}(a, d') \in g(b', \mathcal{C})$  for all  $a \in g(b, \mathcal{C})$ . In this case,  $d$  and  $d'$  must be from a wildcard or  
 563 the inner wildcard of a singleton-wildcard ( $d$  and  $d'$  cannot both be from a singleton of a  
 564 singleton-wildcard since  $d \neq d'$ ). Therefore every indel tract  $d$  in  $\tau$  satisfies  $\text{Apply}(a, d) \in g(b', \mathcal{C})$   
 565 for all  $a \in g(b, \mathcal{C})$ .

Therefore (20) simplifies to

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

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

570 We are finally ready to combine lumpability with Felsenstein's pruning algorithm. The following  
 571 theorem provides a recursive algorithm for calculating (6), using results from above.

**Theorem 1.** *Suppose the above model assumptions hold. Consider any tree node  $N$ , target status  $b$ , and nonempty allele group  $g(b; \mathbb{N})$ . Denote*

$$p_N(b) = \Pr(X_{\text{Leaves}(N)}(T) = a_{\text{Leaves}(N)} \mid \{X_{N'}(t) : 0 \leq t \leq t_{N'}\} \subseteq \text{AncState}(N') : N' \in \text{Desc}(N) \mid X_N(t_N) \in g(b; \mathbb{N})). \quad (21)$$

If  $N$  is an internal node, then

$$p_N(b) = \prod_{\mathcal{C} \in \text{children}(N)} \left\{ \sum_{\substack{b' \in \{0,1\}^M \\ g(b'; \mathcal{C}) \neq \emptyset}} p_{\mathcal{C}}(b') \Pr(X_{\mathcal{C}}(t_{\mathcal{C}}) \in g(b'; \mathcal{C}) \mid X_{\mathcal{C}}(0) \in g(b; \mathcal{C})) \right\}. \quad (22)$$

572 where  $\Pr(X_{\mathcal{C}}(t_{\mathcal{C}}) \in g(b'; \mathcal{C}) \mid X_{\mathcal{C}}(0) \in g(b; \mathcal{C}))$  is calculated using the instantaneous transition rates given in  
 573 Lemma 3.

*Proof.* For any internal node, we know that

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

(We do not need to sum over the partition  $g(\text{other}; \mathbb{N})$  since it contributes zero probability.) By irreversibility of the mutation process,  $g(b; \mathbb{N}) \subseteq g(b; \mathcal{C})$  if  $\mathcal{C}$  is a child of  $N$ . By (19) in Lemma 3,

$$\Pr(\mathcal{C}(t_{\mathcal{C}}) \in g(b'; \mathcal{C}) \mid \mathcal{C}(0) \in g(b; \mathbb{N})) = \Pr(\mathcal{C}(t_{\mathcal{C}}) \in g(b'; \mathcal{C}) \mid \mathcal{C}(0) \in g(b; \mathcal{C})),$$

574 which means (22) also holds for node  $N$ .  $\square$

575 Note that (22) requires enumerating the possible target statuses. We can do this quickly using  
576 Lemma 2.

577 Since each node may have a different partition of the state space, we compute a separate  
578 instantaneous transition rate matrix for each branch. If we have multiple barcodes, we need to  
579 compute a separate matrix for each branch and for each barcode. Calculating the likelihood and its  
580 gradient can therefore become memory-intensive when there are many branches and/or barcodes.  
581 One way to reduce the amount of memory is to sum over subsets of  $\text{AncState}(\cdot)$  instead. This is  
582 often reasonable since it is unlikely for the barcode to have many hidden events. For the analyses  
583 of the zebrafish data, we only sum over states that can be reached when at most one masked indel  
584 tract occurs along each branch. If there are more than 20 such states at a node, we only sum over  
585 the possible states when no masked indel tracts occur along that branch.

### 586 Caterpillar trees

587 As discussed in the main manuscript, we resolve the multifurcations in the tree as caterpillar trees  
588 to estimate the ordering of events. Recall that a caterpillar tree is a tree where all the leaves branch  
589 off of a central path, which we call the “caterpillar spine.” Thus for each multifurcating node  $N$  with  
590 children nodes  $C_1, \dots, C_K$ , we resolve the multifurcation as a tree where the children nodes branch off  
591 of the caterpillar spine (Figure 3b). We do not consider all possible resolutions of the multifurcations  
592 since there are a super-exponential number of them and we likely do not have enough information  
593 to choose between all the possible trees (recall that they are parsimony-equivalent).

594 We need an efficient method to select the best ordering in each caterpillar tree since the  
595 number of possible orderings for  $K$  children nodes is  $K!$ , which is also super-exponential. Since it  
596 is computationally intractable to calculate the likelihood for each ordering, we take an alternate  
597 approach where we introduce another approximation of the likelihood. This approximate likelihood  
598 can be computed using the same mathematical expression regardless of the ordering of the children  
599 nodes, which means we can tune over all possible orderings in the caterpillar trees by solving a  
600 single continuous optimization problem.

**Approximation 2.** *We approximate the likelihood by considering only the mutation histories that have a constant allele along the caterpillar spines:*

$$\Pr(X_{\text{Leaves}(\mathbb{T})}(T) = a_{\text{Leaves}(\mathbb{T})}) \approx \Pr(X_{\text{Leaves}(\mathbb{T})}(T) = a_{\text{Leaves}(\mathbb{T})}, \text{alleles are constant on all spines}). \quad (23)$$

601

602 To construct a mathematical expression for (23) that is independent of the ordering along  
603 caterpillar trees, we first re-parameterize the branch lengths for children of multifurcating nodes.  
604 For each child  $C$  of a multifurcating node, let  $\ell_C$  indicate the distance between the child node and  
605 the multifurcating node and  $\beta_C \in (0, 1)$  indicate the proportion of distance  $\ell_C$  that is located on the  
606 caterpillar spine (Figure 13). We can capture all possible orderings in a caterpillar tree by varying  
607 the values of these two sets of parameters across the children of a multifurcating node.

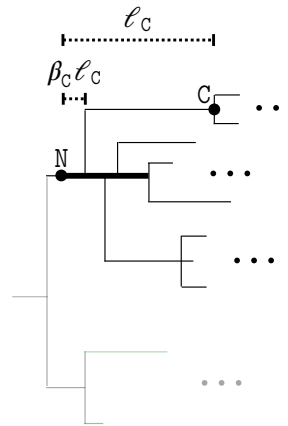
Next we extend the likelihood calculation in Theorem 1 where the multifurcations are resolved as caterpillar trees and we want to calculate the approximate likelihood (23). We do this by recursing on the analogous quantity

$$\tilde{p}_N(a) = \Pr(X_N(T), \text{alleles are constant on all spines} | N(0) = a). \quad (24)$$

To calculate (24), we use the recursive relation that  $\tilde{p}_N(a)$  is equal to

$$\begin{cases} \Pr(X_N(t_{\text{spine}}) = a | X_N(0) = a) \prod_{C \in \text{children}(N)} \{ \sum_{a' \in \Omega} \Pr(X_C(\ell_C(1 - \beta_C)) = a' | X_C(0) = a) \tilde{p}_C(a') \} & \text{if } |\text{children}(N)| > 2 \\ \prod_{C \in \text{children}(N)} \{ \sum_{a' \in \Omega} \Pr(X_C(t_C) = a' | X_C(0) = a) \tilde{p}_C(a') \} & \text{otherwise} \end{cases} \quad (25)$$

608 where  $t_{\text{spine}} = \max\{\ell_C \beta_C : C \in \text{children}(N)\}$ . Using the same machinery in Lemma 3 and Theorem 1,  
609 we can then calculate (25) efficiently by grouping ancestral allele states into lumped states.



**Figure 13.** Parameterization of branch lengths in a caterpillar tree within the context of the entire tree  $\mathbb{T}$ . The rest of the tree is greyed out to draw focus to the caterpillar tree. The bolded path is the caterpillar spine. Each child node  $C$  of the caterpillar tree is associated with parameters  $\ell_C$  and  $\beta_C$ .  $\ell_C$  is the length of the path from the start of the caterpillar spine to  $C$ .  $\beta_C$  is the proportion of this path is along the caterpillar spine. The length of the caterpillar spine,  $t_{\text{spine}}$ , is the maximum value of  $\beta_C \ell_C$  over all children nodes  $C$ .

## 610 Penalization

611 Our algorithm fits the tree and mutation parameters by maximizing the penalized log likelihood.  
 612 Penalization improves accuracy when the number of observations is small relative to the number  
 613 of parameters; GESTALT exhibits this problem because the number of parameters is large and the  
 614 number of independent barcodes is small (McKenna *et al.* [2016] only has one barcode).

We propose a tree-based penalty that discourages large differences in the branch lengths  $\ell$  and the target rates  $\lambda$ . For multifurcating nodes, the branch lengths include the length of the spine as well as the lengths of branches off of the spine, i.e.  $\ell_C(1 - \beta_C)$ . Let  $L$  be the number of branch lengths in  $\ell$ . The penalty is then

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

615 where  $\kappa_1, \kappa_2 > 0$  are penalty parameters.

616 Our penalty on the branch length was also considered in the penalized likelihood framework  
 617 in Kim and Sanderson [2008]. However the focus of Kim and Sanderson [2008] was to encourage  
 618 development of methods that were more flexible and had fewer assumptions, rather than to  
 619 improve estimation in high-dimensional settings. In particular, their work focused on the standard  
 620 phylogenetic setting in which there are multiple independent sites. However, in the GESTALT setting,  
 621 the current available datasets were generated using a single barcode and we cannot tune the  
 622 penalty parameters using the common approach of cross-validation [Hastie *et al.*, 2009, Arlot and  
 623 Celisse, 2010]. In addition, Kim and Sanderson [2008] only discussed penalized likelihood in the  
 624 context of a fixed topology. In our setting the true tree is unknown and we must consider various  
 625 tree topologies.

626 We found that a major hurdle in applying this penalized likelihood framework is that some  
 627 topologies will naturally have larger penalties. Therefore we cannot simply choose the tree with  
 628 the highest penalized log likelihood. Our solution is to perform a hill-climbing iterative search and  
 629 score topology updates based on the penalized log likelihood where the penalty is restricted to the  
 630 shared subtree. We found that our method tends to choose topology updates that improve the  
 631 tree estimate (see Figure 17b).

632 Alternatively, we could have applied regularization methods tailored for the setting where the  
 633 topology is unknown. These methods typically regularize the tree towards a prespecified tree

634 [Wu et al., 2013, Dinh et al., 2018]. However we would like to place minimal assumptions on the  
635 developmental process and we have little to no knowledge about the true tree.

### 636 Tuning penalty parameters

637 By varying the value of the penalty parameters  $\kappa_1$  and  $\kappa_2$ , we can control the trade-off between  
638 minimizing the penalty versus maximizing the log likelihood. Choosing appropriate values is  
639 crucial for estimation accuracy. A common approach for tuning penalty parameters is to use  
640 cross-validation [Arlot and Celisse, 2010]; we use this procedure whenever possible. Note that we  
641 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 validation sets  $T$  and  $V$ , respectively. Next we fit tree and mutation parameters  $\hat{\ell}_\kappa$  and  $\hat{\theta}_\kappa$ , respectively, for each  $\kappa$  using only the training data. We choose the  $\kappa$  with the highest validation log likelihood

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

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

644 Unfortunately cross-validation cannot be utilized when there is a single barcode since we  
645 cannot split the dataset by barcodes. Instead we propose a variant of cross-validation described  
646 in Algorithm 2. The main differences are that we partition the leaves instead of the barcodes into  
647 training and validation sets  $S$  and  $S^c$ , respectively; and we select the best penalty parameter that  
648 maximizes the conditional probability of the observed alleles at  $S^c$  given the observed alleles at  $S$ .

649 To partition the leaves, we randomly select a subset of leaf children of each multifurcating node  
650 to put in the validation set  $S^c$ . We partition leaves in this manner, rather than simply dividing the  
651 leaves in half, because we must be able to evaluate (or closely approximate) (27) at the end of  
652 Algorithm 2 using the fitted branch length and mutation parameters. That is, we must be able  
653 to regraft the leaves in the set  $S^c$  onto the fitted tree. Regrafting is easy for the leaves in our  
654 specially-constructed set: The parent node of each leaf in  $S^c$  must be located somewhere along  
655 the caterpillar spine corresponding to its original multifurcating parent. In our implementation, we  
656 chose to regraft the leaves to the midpoints of their corresponding caterpillar spines. The regrafting  
657 procedure is illustrated in Figure 14. Note that we do not tune the branch lengths of these validation  
658 leaves since it amounts to peeking at the validation data. In our simulations, we found that when  
659 tuning the branch lengths to maximize the unpenalized (or penalized) log likelihood, we nearly  
660 almost always choose the smallest penalty parameter since it prioritizes maximizing the likelihood  
661 and, therefore, (27).

To assess each candidate penalty parameter  $\kappa$ , 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  $\ell^*$  and  $\theta^*$ , we know that

$$E [\log \Pr(X_{S^c}(T) | X_S(T); \ell^*, \theta^*); \ell^*, \theta^*] \geq E [\log \Pr(X_{S^c}(T) | X_S(T); \ell, \theta); \ell^*, \theta^*] \quad \forall \ell, \theta \quad (26)$$

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

666 From a theoretical standpoint, using (26) to select penalty parameters makes the most sense if  
667 we have an unbiased estimate of the expected conditional probability. Unfortunately, in our setting,  
668 the conditional probability in (27) is actually a biased estimate since the fitted parameters depended  
669 on the observed alleles at leaves  $S$ . Nonetheless, in simulations (where the truth is known), this  
670 biased estimate seemed to work well, as the selected penalty parameter was typically close to the  
671 best penalty parameter.

---

**Algorithm 2** Cross validation for a single barcode

---

Initialize  $S$  to be all the leaves in tree  $\mathbb{T}$ . Throughout, let  $S^c$  denote all the leaves in  $\mathbb{T}$  not in  $S$ .

**for** each multifurcating node  $\mathbb{N}$  where at least one children is a leaf **do**

Let  $C_1, \dots, C_m$  be the children nodes of  $\mathbb{N}$  that are leaves. Randomly select  $m' \geq 1$  of them and remove these from  $S$ .

**end for**

Let  $\mathbb{T}_S$  be the subtree over the leaves  $S$ .

**for** each candidate penalty parameter  $\kappa$  **do**

Maximize the penalized log likelihood of the tree  $\mathbb{T}_S$  with respect to its branch lengths  $\ell$  and mutation parameters  $\theta$

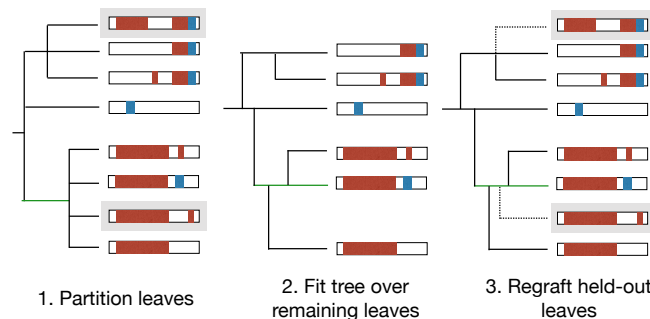
$$\hat{\ell}_\kappa, \hat{\theta}_\kappa = \arg \max_{\ell, \theta} \log \Pr (X_S(T) = a_S; \ell, \theta) + \text{Pen}_\kappa (\ell, \theta).$$

**end for**

Return the penalty parameter that maximizes the conditional probability:

$$\hat{\kappa} = \arg \max_{\kappa} \Pr (X_{S^c}(T) = a_{S^c} \mid X_S(T) = a_S; \hat{\ell}_\kappa, \hat{\theta}_\kappa). \quad (27)$$


---



**Figure 14.** 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).

672 Finally, we must simultaneously tune the penalty parameter and the topology of the tree (from  
 673 Algorithm 1). Our full algorithm alternates between tuning the penalty parameters for a fixed tree  
 674 topology and running a single iteration of Algorithm 1 for a fixed penalty parameter. After the  
 675 penalty parameters are stable, we keep them fixed and only run Algorithm 1.

### 676 Specific model implementation

677 Here we discuss the specific implementation we use to model the GESTALT data. We suppose the  
 678 mutation process is homogeneous and does not depend on  $t$ . Therefore we will drop the time  
 679 index  $t$  in the model. Recall that there are two major components of the mutation model: the rates  
 680 at which target tracts are introduced and the conditional probability of an indel tract given the  
 681 selected target tract.

To parameterize the rate at which a target tract  $\tau = \text{TT}[j'_0, j_0, j_1, j'_1]$  is introduced, we further  
 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 (3)  
 and (4)):

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

682 where  $\gamma_0$  and  $\gamma_1$  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 exponential  
 distribution with rate of  $\lambda_j > 0$ . For focal target cuts where  $j_0 = j_1$ , we define

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

683 For double cuts at targets  $j_0$  and  $j_1$ , we suppose the cut time follows an exponential distribution  
 684 with rate  $\omega \cdot (\lambda_{j_0} + \lambda_{j_1})$ , where  $\omega$  is an additional model parameter that we estimate and does not  
 685 depend on the targets.

Our parameterization of the double-cut rate is based on the assumption that an inter-target  
 deletion is introduced when the cuts at both targets occur within a small time window of length  
 $\epsilon$ . For random cut times  $X_{j_0}$  and  $X_{j_1}$  for targets  $j_0$  and  $j_1$ , we approximate that the cut times occur  
 within a small window  $\epsilon$  with the distribution

$$p\left(|X_{j_0} - X_{j_1}| \leq \epsilon, \frac{X_{j_0} + X_{j_1}}{2} = t\right) \approx \Pr(|X_{j_0} - X_{j_1}| \leq \epsilon) p(X_{j_0} = X_{j_1} = t | X_{j_0} = X_{j_1}). \quad (28)$$

686 The values on the left and right hand sides approach each other as  $\epsilon \rightarrow 0$ . The first component on  
 687 the right-hand-side of (28) approaches zero as  $\epsilon \rightarrow 0$  and does not vary much for different values  
 688 of  $\lambda_{j_0}, \lambda_{j_1}$  if  $\epsilon$  is sufficiently small. Hence we use the same value of  $\omega$  for all targets. The second  
 689 component on the right-hand-side of (28) corresponds to an exponential distribution with the rate  
 690  $\lambda_{j_0} + \lambda_{j_1}$ .

691 We can interpret  $\omega$  in two ways. First,  $\omega$  controls how often a double cut is introduced. In  
 692 an unmodified barcode, the relative rate that a double cut is introduced versus a single cut is  
 693  $\omega \sum_{j_1 < j_2} (\lambda_{j_1} + \lambda_{j_2})$  versus  $\sum_{j=1}^M \lambda_j$ . The second interpretation, based on (28), is that  $\omega$  serves as a proxy  
 694 for  $\epsilon$ : Larger  $\omega$  indicates that an inter-target deletion can be introduced by two cuts spaced farther  
 695 apart in time.

696 The second major component of the GESTALT mutation model specifies the conditional proba-  
 697 bility of introducing a particular indel tract given target tract  $\tau = \text{TT}[j'_0, j_0, j_1, j'_1]$ . An indel tract can  
 698 be represented by its deletion lengths to the left and right and the insertion sequence. We will  
 699 suppose that the probability of a single insertion sequence is uniform over all possible nucleotide  
 700 sequences of that length. Let  $X_0, X_1, X_2$  be the random variables parameterizing the lengths of the  
 701 left deletion, right deletion, and insertion, respectively. Let  $x_{\tau, \min, i}$  and  $x_{\tau, \max, i}$  for  $i = 0$  and 1 specify  
 702 the minimum and maximum deletion lengths to the left and right, respectively, for target tract  
 703  $\tau$ . (For example, if  $j_0 = j'_0$ , the left deletions must be short so  $x_{\tau, \min, 0} = 0$  and  $x_{\tau, \max, 0}$  is the longest

704 deletion without deactivating target  $j_0 - 1$ . As another example, if  $j_0 = j'_0 + 1$ , the left deletion is  
 705 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  
 706 without deactivating target  $j'_0 - 1$ .) For insertions,  $x_{\tau, \min, 2} = 0$  and  $x_{\tau, \max, i} = \infty$  regardless of the target  
 707 tract.

We parameterize the conditional probability of indel  $d$  with left deletion, right deletion, and insertion lengths  $x_0, x_1, x_2$  given target tract  $\tau$  as

$$\Pr(X_0 = x_0, X_1 = x_1, X_2 = x_2 | \tau) = p(x_0, x_1, x_2 | \tau; x_{\tau, \min, 0}, x_{\tau, \min, 1}, x_{\tau, \min, 2})$$

where

$$p(x_0, x_1, x_2 | \tau; x_{\min, 0}, x_{\min, 1}, x_{\min, 2}) = \begin{cases} p_{0, \text{boost}} p(x_0, x_1, x_2 | \tau; x_{\min, 0} + 1, x_{\min, 1}, x_{\min, 2}) \\ + p_{1, \text{boost}} p(x_0, x_1, x_2 | \tau; x_{\min, 0}, x_{\min, 1} + 1, x_{\min, 2}) \\ + p_{2, \text{boost}} p(x_0, x_1, x_2 | \tau; x_{\min, 0}, x_{\min, 1}, x_{\min, 2} + 1) & \text{if } x_{\min, 0} = x_{\min, 1} = x_{\min, 2} = 0 \\ \Pr(X_0 = x_0 | \tau; x_{\min, 0}) \Pr(X_1 = x_1 | \tau; x_{\min, 1}) \Pr(X_2 = x_2 | \tau; x_{\min, 2}) & \text{otherwise} \end{cases}$$

708 where  $p_{0, \text{boost}} + p_{1, \text{boost}} + p_{2, \text{boost}} = 1$ . The probabilities  $p_{i, \text{boost}}$  ensure that we can never introduce an  
 709 indel tract that deletes and inserts nothing. When the minimum insertion and deletion lengths are  
 710 zero (in the case of focal target tracts), we use the probabilities  $p_{i, \text{boost}}$  to randomly pick whether to  
 711 boost the minimum left deletion, right deletion, or insertion length by one.

We assume that the deletion lengths follow a zero-inflated, truncated negative binomial distribution; and the insertion lengths follow a zero-inflated negative binomial distribution. Let  $\text{NB}(m, q)$  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. The zero-inflation factor for deletion lengths for focal indel tracts is  $p_{i, 0}$  and inter-target indel tracts is  $p_{i, 1}$ , where left and right are indicated by  $i = 0$  and  $i = 1$ , respectively. The zero-inflation factor for insertion lengths is  $p_{2, 0} = p_{2, 1}$ . Then for  $i = 0, 1, 2$ , we define

$$\Pr(X_i = x_i | \tau; x_{\min, i}) = \begin{cases} p_{i, \mathbb{1}\{j_0=j_1\}} & \text{if } x_i = x_{\min, i} = 0 \\ (1 - p_{i, \mathbb{1}\{j_0=j_1\}}) \left[ \Pr(X = x - x_{\min, i}; \text{NB}(m_i, q_i)) + \frac{\Pr(X > x_{\max, i} - x_{\min, i}; \text{NB}(m_i, q_i))}{x_{\max, i} - x_{\min, i}} \right] & \text{if } x_i > x_{\min, i}. \end{cases}$$

## 712 Implementation

713 The code is implemented in Python using Tensorflow. We maximize the penalized log likelihood  
 714 using Adam [Kingma and Ba, 2014].

## 715 Comparison Methods

716 We use PHYLIP version 3.697 [Felsenstein, 1995], the neighbor-joining algorithm in Bio.Phylo  
 717 (Biopython version 1.72) [Talevich et al., 2012], and the chronos function in R package ape version  
 718 5.2 [Paradis and Schliep, 2018].

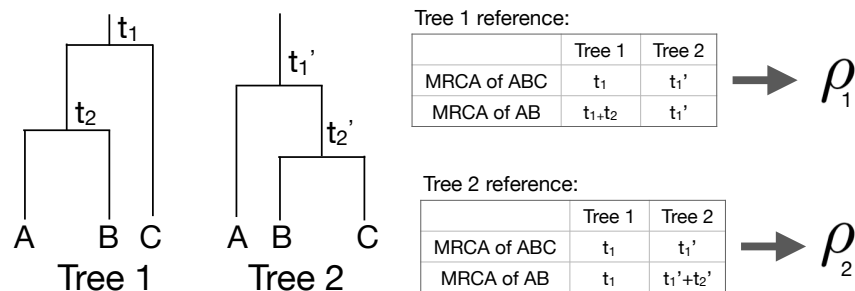
## 719 Evaluation metrics

720 Given ultrametric trees 1 and 2 with the same set of leaves, the internal node height correlation  
 721 between the two trees is calculated as follows (Figure 15):

- 722 1. For each internal node in tree 1, find the matching node in tree 2 that is the most recent  
 723 common ancestor of the same set of leaves.
- 724 2. Calculate the Pearson correlation of the heights of matched nodes.
- 725 3. Do the same swapping tree 1 and 2.

726 4. Average the two correlation values.

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



**Figure 15.** Example calculation of the internal node height correlation. For each tree, define the groups of leaves based on its internal nodes and calculate the correlation of the time of the most recent common ancestors (MRCAs) of the leaf groups. The internal node height correlation is the average of the two correlation values.

728

### 729 Simulation setup and additional results

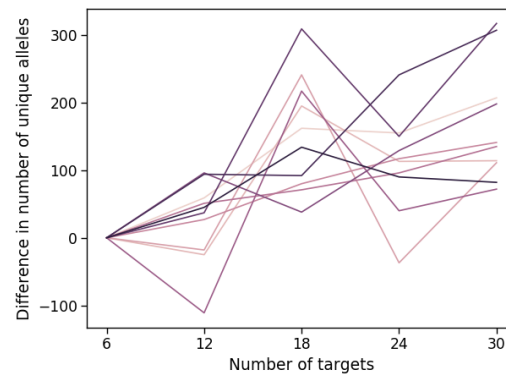
730 For the results in Figure 4b, the data was simulated with 5 synchronous cell division cycles followed  
 731 by a birth-death process where the birth rate decayed at a rate of  $\exp(-18t)$ . The barcode was  
 732 composed of six targets with  $\lambda = 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6$ . The weight  $\omega$  was set to 0.06 so that  
 733 20% of the unique observed indel tracts were due to double cuts. We sampled 8% of the leaves  
 734 so that the average number of unique observed alleles was around 100 leaves. We refer to this  
 735 simulation setup as Simulation A. We ran 20 replicates of Simulation A.

736 The results in Figure 4c are from a larger simulation, which we will refer to as Simulation B, that  
 737 is closer to the data collected in *McKenna et al. [2016]*. Since zebrafish data undergo around 11  
 738 synchronous cell division cycles, this larger simulation entailed 9 synchronous cell division cycles  
 739 followed by a birth-death process. We simulated with a barcode composed of ten targets. The  
 740 resulting tree had on average around 200 leaves. We ran GAPML for 8 topology tuning iterations; at  
 741 each iteration, we consider at most 15 SPR moves. The displayed results are from 20 replicates.

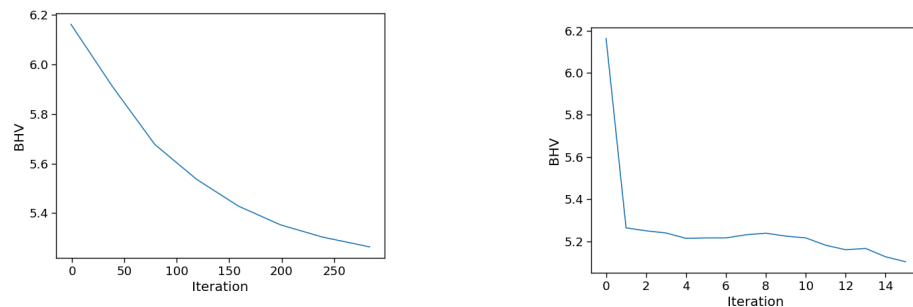
742 For this larger simulation, we also compared the runtimes of the methods on a server with an  
 743 Intel Xeon 2x8 core processor at 3.20GHz and 256 GB RAM. Obtaining tree topologies from C-S  
 744 parsimony and neighbor-joining runs on the order of minutes. Branch length estimation using  
 745 *chronos* runs on the order of seconds. In contrast, GAPML required up to three hours. Though the  
 746 runtime of our method is much longer, it is still reasonable compared to the amount of time spent  
 747 on data collection, which includes waiting until the organism is a certain age.

748 Using our simulation engine, we compare two very simple barcode design ideas: a single  
 749 barcode with many targets, recommended in *Salvador-Martínez et al. [2018]*, or many identical  
 750 barcodes. However we believe the latter is more effective since spreading the targets over separate  
 751 barcodes tends to create more unique alleles. In particular, the inter-target deletions tend to be  
 752 shorter, which means fewer existing mutations are deleted and fewer targets are deactivated. To  
 753 test this idea, we compared to the two design options in a simulation setup where we iteratively  
 754 increased the number of targets by six, i.e. add six targets to the existing barcode or add a new  
 755 barcode with six targets. Here we observe all 1024 leaves of a full binary tree with 10 levels. All  
 756 targets had the same single-cut rate. We calibrated the double-cut weight  $\omega$  to be around 18%  
 757 for both barcode designs – this slightly favors the single-barcode design since it would have a  
 758 higher rate of double cuts *in vivo* compared to a multiple-barcode design. Nevertheless, we find in  
 759 our simulations that splitting the targets over separate barcodes tends to result in a much larger





**Figure 16.** We compare the number of unique alleles obtained 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.



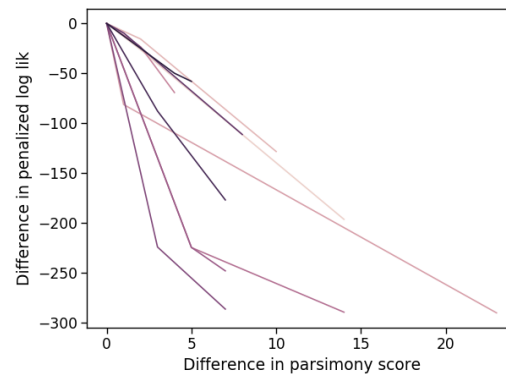
**(a)** Example of how the BHV distance changes as the branch lengths and mutation parameters are updated using gradient descent to maximize the penalized likelihood. **(b)** Example of how the BHV distance changes at each SPR iteration, where we select the SPR with the highest likelihood with penalization over only the shared tree.

**Figure 17.** Examples of how the BHV distance changes as the algorithm proceeds for one simulation replicate from the ten-target setting.

760 number of unique alleles than using a single barcode (Figure 16). At 30 targets, the multiple-barcode  
761 design has roughly 200 more unique alleles on average than the single-barcode design. Another  
762 reason we prefer the multiple-barcode design is that our model and tree estimates improve as the  
763 number of independent and identical barcodes increases, as illustrated in Figure 4b.

764 Next, to better understand our algorithm GAPML, we show in-depth simulation results from a  
765 single replicate (Figure 17). Here we use the settings from Simulation B. Starting from the initial  
766 tree topology, the algorithm tunes the branch lengths and mutation parameters to maximize the  
767 penalized likelihood. During the gradient descent algorithm, the BHV distance of the tree estimate  
768 decreases (Figure 17a). In addition, we see that the BHV distance of the tree estimate decreases as  
769 Algorithm 1 iteratively performs SPR moves to update the tree topology.

770 Our method searches over the maximally parsimonious trees since they tend to have the highest  
771 penalized log likelihood. To justify this restricted search, we compared the penalized log likelihood  
772 for tree topology candidates of different parsimony scores, where the data was generated using  
773 Simulation A. To generate tree topologies with different parsimony scores, we started with the  
774 maximally parsimonious tree fit from Camin-Sokal and iteratively applied random SPR moves. For



**Figure 18.** 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.

775 each of tree rearrangement, we fit a model by maximizing the penalized log likelihood. The penalty  
776 parameter is the same across all rearrangements. As seen in Figure 18, the most parsimonious  
777 trees have the highest penalized log likelihoods. Since our method aims to select a tree topology  
778 that maximizes the penalized log likelihood, it would not benefit from considering SPR moves that  
779 make the tree less parsimonious; instead, considering these additional moves would make the  
780 method much slower.

### 781 Zebrafish data analysis

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

786 To test if the fitted trees are recovering similar developmental relationships across fish rather  
787 than random noise, we ran a permutation test comparing the correlation between tissue distances  
788 from the estimated trees to that from randomly-estimated trees over randomly-shuffled data.  
789 More specifically, for a given tree topology, we randomly permute the observed alleles at the  
790 leaves. Each allele is associated with the number of times it is observed in each tissue type; we  
791 randomly shuffle these abundances over the possible tissue types within each allele. Finally, we  
792 randomly assign branch lengths along the tree by drawing samples from a uniform distribution and  
793 using the  $t$ -parameterization of *Gavryushkin and Drummond [2016]* to assign branch lengths. The  
794 correlation between tissue distances in these random trees is close to zero. All permutation tests  
795 were performed using 2000 replicates.

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

799 We conclude by noting that the random trees are generated using the estimated tree topology  
800 from each method. Thus the null distributions are different and the p-values are not directly  
801 comparable between methods. Though this slightly complicates interpretation, we prefer this  
802 approach since the estimated tree topology may naturally induce correlation between tissue  
803 distances. For most validation tests, the mean of the null distribution was similar across the  
804 different methods, and therefore the p-values are somewhat comparable. The major exception was  
805 the tests that checked recovery of cell-type and germ-layer restriction: here the mean of the null

806 distribution were very different and we abstain from comparing p-values across methods.  
807 For Figure 6a, we bootstrapped fish replicates to estimate confidence intervals for the average  
808 correlation between estimated target cut rates.

## 809 References

- 810 **Adams DC**, Collyer ML. Multivariate Phylogenetic Comparative Methods: Evaluations, Comparisons, and  
811 Recommendations. *Syst Biol.* 2018 Jan; 67(1):14–31.
- 812 **Arlot S**, Celisse A. A survey of cross-validation procedures for model selection. *Stat Surv.* 2010; 4:40–79.
- 813 **Baydin AG**, Pearlmutter BA, Radul AA, Siskind JM. Automatic Differentiation in Machine Learning: a Survey. *J*  
814 *Mach Learn Res.* 2018; 18(153):1–43.
- 815 **Billera LJ**, Holmes SP, Vogtmann K. Geometry of the Space of Phylogenetic Trees. *Adv Appl Math.* 2001 Nov;  
816 27(4):733–767.
- 817 **Camin JH**, Sokal RR. A Method for Deducing Branching Sequences in Phylogeny. *Evolution.* 1965; 19(3):311–326.
- 818 **Davydov II**, Robinson-Rechavi M, Salamin N. State aggregation for fast likelihood computations in molecular  
819 evolution. *Bioinformatics.* 2017 Feb; 33(3):354–362.
- 820 **Dinh V**, Tung Ho LS, Suchard MA, Matsen FA 4th. Consistency and convergence rate of phylogenetic inference  
821 via regularization. *Ann Stat.* 2018 Aug; 46(4):1481–1512.
- 822 **Felsenstein J**. Evolutionary trees from DNA sequences: a maximum likelihood approach. *J Mol Evol.* 1981;  
823 17(6):368–376. <http://www.ncbi.nlm.nih.gov/pubmed/7288891>.
- 824 **Felsenstein J**. PHYLIP (phylogeny inference package), version 3.5 c; 1995.
- 825 **Felsenstein J**. *Inferring phylogenies*, vol. 2. Sinauer associates Sunderland, MA; 2004.
- 826 **Ganguly A**, Petrov T, Koepl H. Markov chain aggregation and its applications to combinatorial reaction  
827 networks. *J Math Biol.* 2014 Sep; 69(3):767–797.
- 828 **Gavryushkin A**, Drummond AJ. The space of ultrametric phylogenetic trees. *J Theor Biol.* 2016 Aug; 403:197–  
829 208.
- 830 **Goolsby EW**. Likelihood-Based Parameter Estimation for High-Dimensional Phylogenetic Comparative Models:  
831 Overcoming the Limitations of “Distance-Based” Methods. *Syst Biol.* 2016 Sep; 65(5):852–870.
- 832 **Hastie T**, Tibshirani R, Friedman J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*,  
833 Second Edition. 2 ed. Springer Series in Statistics, Springer-Verlag New York; 2009.
- 834 **Hillston J**. Compositional Markovian Modelling Using a Process Algebra. In: *Computations with Markov Chains*  
835 Springer US; 1995. p. 177–196.
- 836 **Hohna S**, Defoin-Platel M, Drummond AJ. Clock-constrained tree proposal operators in Bayesian phylogenetic  
837 inference. In: *2008 8th IEEE International Conference on Bioinformatics and BioEngineering*; 2008. p. 1–7.
- 838 **Julien C**, Leandro A, Hélène M. A Penalized Likelihood Framework For High-Dimensional Phylogenetic Compara-  
839 tive Methods And An Application To New-World Monkeys Brain Evolution. *Syst Biol.* 2018 Jun; .
- 840 **Kalhor R**, Kalhor K, Mejia L, Leeper K, Graveline A, Mali P, Church GM. Developmental barcoding of whole mouse  
841 via homing CRISPR. *Science.* 2018 Aug; 361(6405).
- 842 **Kalhor R**, Mali P, Church GM. Rapidly evolving homing CRISPR barcodes. *Nat Methods.* 2017 Feb; 14(2):195–200.
- 843 **Keegan BR**, Meyer D, Yelon D. Organization of cardiac chamber progenitors in the zebrafish blastula. *Develop-*  
844 *ment.* 2004 Jul; 131(13):3081–3091.
- 845 **Kemeny JG**, Laurie Snell J. *Finite Markov Chains: With a New Appendix “Generalization of a Fundamental Matrix”*.  
846 1 ed. Undergraduate Texts in Mathematics, Springer-Verlag New York; 1976.
- 847 **Kim J**, Sanderson MJ. Penalized likelihood phylogenetic inference: bridging the parsimony-likelihood gap. *Syst*  
848 *Biol.* 2008 Oct; 57(5):665–674.
- 849 **Kingma DP**, Ba J. Adam: A Method for Stochastic Optimization. . 2014 Dec; .

- 850 **McKenna A**, Findlay GM, Gagnon JA, Horwitz MS, Schier AF, Shendure J. Whole-organism lineage tracing by  
851 combinatorial and cumulative genome editing. *Science*. 2016; 353(6298).
- 852 **Moody SA**. *Cell Lineage and Fate Determination*. Elsevier; 1998.
- 853 **Paradis E**, Schliep K. ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R.  
854 *Bioinformatics*. 2018; xx:xxx-xxx.
- 855 **Raj B**, Wagner DE, McKenna A, Pandey S, Klein AM, Shendure J, Gagnon JA, Schier AF. Simultaneous single-cell  
856 profiling of lineages and cell types in the vertebrate brain. *Nat Biotechnol*. 2018 Jun; 36(5):442–450.
- 857 **Saitou N**, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol*  
858 *Evol*. 1987 Jul; 4(4):406–425.
- 859 **Salvador-Martínez I**, Grillo M, Averof M, Telford MJ. Is it possible to reconstruct an accurate cell lineage using  
860 CRISPR recorders?; 2018.
- 861 **Sanderson MJ**. Estimating absolute rates of molecular evolution and divergence times: a penalized likelihood  
862 approach. *Mol Biol Evol*. 2002 Jan; 19(1):101–109.
- 863 **Schmidt ST**, Zimmerman SM, Wang J, Kim SK, Quake SR. Quantitative Analysis of Synthetic Cell Lineage Tracing  
864 Using Nuclease Barcoding. *ACS Synth Biol*. 2017 Jun; 6(6):936–942.
- 865 **Shadrin AM**, Ozernyuk ND. Development of the Gill System in Early Ontogenesis of the Zebrafish and Ninespine  
866 Stickleback. *Russ J Dev Biol*. 2002 Mar; 33(2):91–96.
- 867 **Solnica-Krezel L**. Conserved Patterns of Cell Movements during Vertebrate Gastrulation. *Current Biol*-  
868 *ogy*. 2005; 15(6):R213 – R228. <http://www.sciencedirect.com/science/article/pii/S0960982205002770>, doi:  
869 <https://doi.org/10.1016/j.cub.2005.03.016>.
- 870 **Spanjaard B**, Hu B, Mitic N, Olivares-Chauvet P, Janjuha S, Ninov N, Junker JP. Simultaneous lineage tracing and  
871 cell-type identification using CRISPR-Cas9-induced genetic scars. *Nat Biotechnol*. 2018 Jun; 36(5):469–473.
- 872 **Talevich E**, Invergo BM, Cock PJA, Chapman BA. Bio.Phylo: a unified toolkit for processing, analyzing and  
873 visualizing phylogenetic trees in Biopython. *BMC Bioinformatics*. 2012 Aug; 13:209.
- 874 **Woodworth MB**, Girskis KM, Walsh CA. Building a lineage from single cells: genetic techniques for cell lineage  
875 tracking. *Nat Rev Genet*. 2017 Apr; 18(4):230–244.
- 876 **Wu YC**, Rasmussen MD, Bansal MS, Kellis M. TreeFix: statistically informed gene tree error correction using  
877 species trees. *Syst Biol*. 2013 Jan; 62(1):110–120.
- 878 **Yang Z**. *Molecular Evolution: A Statistical Approach*. Oxford University Press; 2014.
- 879 **Zhang C**, Dinh V, Matsen FA IV. Non-bifurcating phylogenetic tree inference via the adaptive LASSO. . 2018 May;  
880 .  
881

882 **Appendix 1**

	30 hpf #4	30 hpf #5	30 hpf #6	30 hpf #8	4.3 hpf #1	4.3 hpf #3	4.3 hpf #8	4.3 hpf #10
Target 1	2.386	2.075	3.212	1.613	1.578	0.874	1.310	1.330
Target 2	1.397	1.262	3.659	2.301	0.590	0.204	0.387	0.343
Target 3	1.437	1.285	3.499	1.477	0.234	0.130	0.144	0.168
Target 4	2.026	1.103	3.637	2.134	0.613	0.351	0.711	0.691
Target 5	1.276	0.892	3.203	1.791	0.477	0.339	0.497	0.518
Target 6	1.344	1.618	3.161	1.200	0.626	0.420	0.570	0.321
Target 7	1.784	0.779	2.577	1.269	1.438	0.634	1.553	0.937
Target 8	1.320	1.712	3.091	2.430	0.275	0.239	0.191	0.181
Target 9	2.944	1.792	3.509	2.058	1.282	0.600	1.375	0.901
Target 10	0.678	1.285	1.227	0.278	0.192	0.109	0.144	0.147
Double cut rate	0.051	0.049	0.039	0.028	0.044	0.063	0.052	0.052
Left long trim length mean	24.291	23.514	25.000	25.000	24.304	23.571	23.650	23.926
Left long trim length SD	1.476	1.124	1.414	1.414	1.477	1.169	1.226	1.375
Left short trim length mean	6.253	3.730	5.749	5.225	6.394	6.074	6.301	5.824
Left short trim length SD	5.594	5.735	4.648	6.650	5.008	5.083	5.107	4.362
Left short trim zero prob	0.264	0.258	0.256	0.280	0.026	0.025	0.024	0.031
Long factor left	0.063	0.058	0.039	0.024	0.059	0.059	0.057	0.056
Long factor right	0.032	0.025	0.025	0.015	0.094	0.181	0.129	0.130
Right long trim length mean	22.647	22.392	24.500	24.500	23.199	22.854	22.759	23.021
Right long trim length SD	1.388	1.127	1.708	1.708	1.707	1.538	1.474	1.631
Right short trim length mean	5.109	4.801	4.677	4.736	3.708	4.145	3.946	4.012
Right short trim length SD	4.141	4.013	3.770	3.845	5.536	5.849	5.719	5.764
Right short trim zero prob	0.147	0.238	0.323	0.188	0.821	0.877	0.860	0.805
Insertion length mean	6.192	5.446	9.796	6.614	4.587	5.972	5.536	5.273
Insertion length sd	7.519	7.291	11.123	7.729	4.723	7.415	5.292	5.432
Insertion zero prob	0.564	0.520	0.535	0.543	0.419	0.438	0.510	0.489

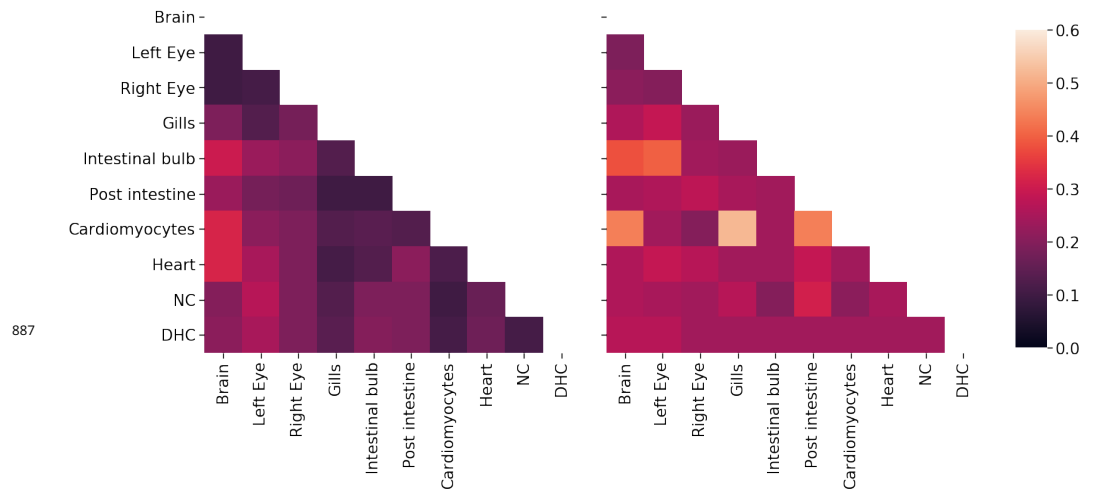
883

	3 day #1	3 day #2	3 day #3	3 day #4	3 day #5
Target 1	0.874	1.318	0.881	4.192	1.675
Target 2	0.158	0.239	0.268	1.071	0.631
Target 3	0.062	0.109	0.170	0.306	0.233
Target 4	0.319	1.198	0.548	1.820	1.026
Target 5	0.258	0.780	0.373	1.556	0.605
Target 6	0.239	0.282	0.319	1.390	1.037
Target 7	0.858	0.208	1.230	2.108	1.045
Target 8	0.082	0.138	0.152	0.541	0.273
Target 9	0.793	1.216	0.657	2.524	1.501
Target 10	0.051	0.070	0.111	0.245	0.198
Double cut rate	0.041	0.025	0.038	0.031	0.056
Left long trim length mean	24.307	24.312	23.847	23.920	23.988
Left long trim length SD	1.471	1.464	1.335	1.371	1.395
Left short trim length mean	5.246	5.119	5.446	6.069	6.073
Left short trim length SD	4.220	4.394	4.727	5.211	5.008
Left short trim zero prob	0.014	0.019	0.024	0.021	0.016
Long factor left	0.050	0.029	0.047	0.037	0.034
Long factor right	0.132	0.084	0.108	0.085	0.171
Right long trim length mean	22.834	23.189	22.810	22.908	22.883
Right long trim length SD	1.518	1.684	1.502	1.569	1.548
Right short trim length mean	3.854	4.755	4.362	4.417	4.738
Right short trim length SD	5.596	6.069	5.939	5.997	6.123
Right short trim zero prob	0.899	0.904	0.858	0.794	0.903
Insertion length mean	5.448	5.932	6.596	5.831	5.110
Insertion length sd	5.155	6.224	7.358	6.390	5.463
Insertion zero prob	0.403	0.385	0.391	0.383	0.371

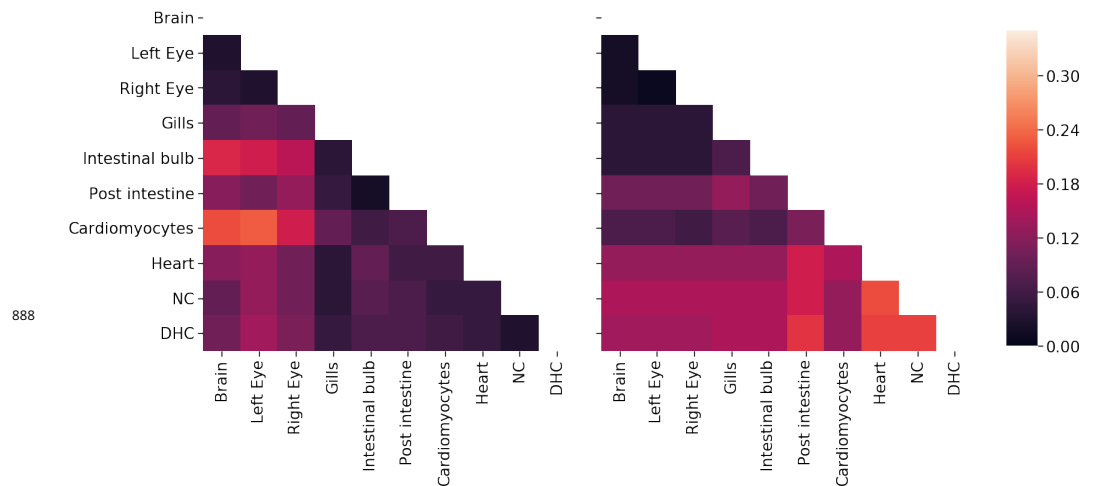
**Appendix 1 Table 1.** Fitted mutation parameters for zebrafish

884

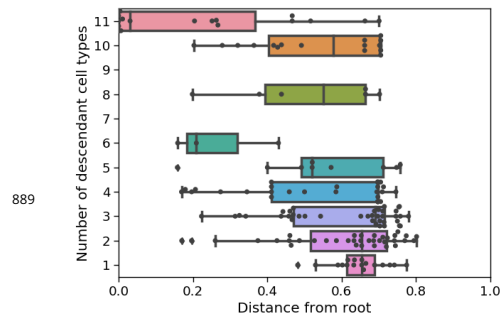
886



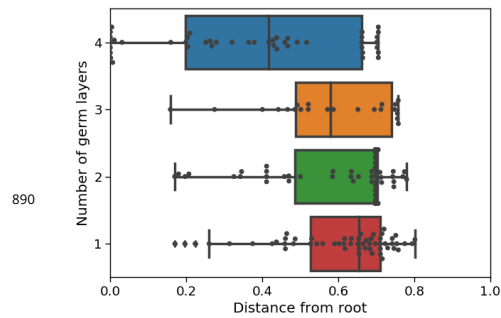
**Figure 5-Figure supplement 1.** The average distance between tissue sources from adult fish 1 (left) and 2 (right) for the tree estimates from Camin-Sokal parsimony and *CHRONOS*. The distance between tissues is the average time from a leaf of one tissue to the closest internal node with a descendant of the other tissue. The shading reflects distance, where bright means far and dark means close.



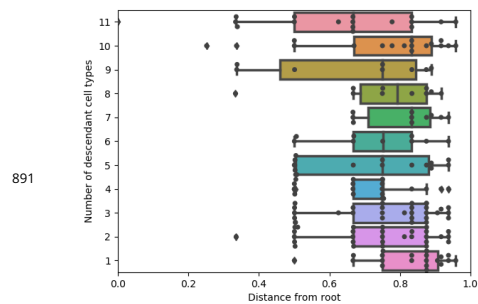
**Figure 5-Figure supplement 2.** The average distance between tissue sources from adult fish 1 (left) and 2 (right) for the tree estimates from neighbor-joining and *CHRONOS*. The distance between tissues is the average time from a leaf of one tissue to the closest internal node with a descendant of the other tissue. The shading reflects distance, where bright means far and dark means close.



**Figure 7-Figure supplement 1.** Internal node times versus number of descendant cell types in the estimated tree for the second adult fish using GAPML.

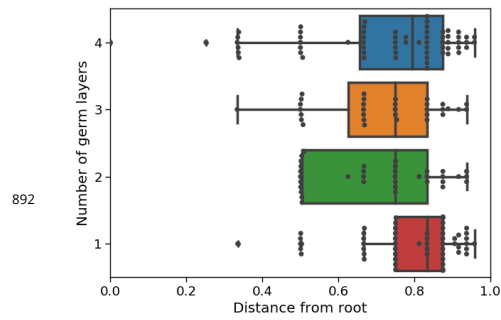


**Figure 7-Figure supplement 2.** Internal node times versus number of descendant germ layers in the estimated tree for the second adult fish using GAPML.

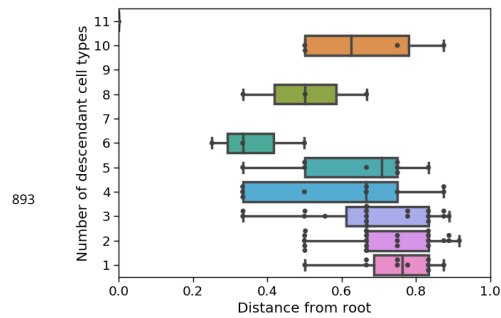


**Figure 7-Figure supplement 3.** Internal node times versus number of descendant cell types in the estimated tree for the first adult fish using Camin-Sokal parsimony and  $\chi_{\theta 00000}$ .

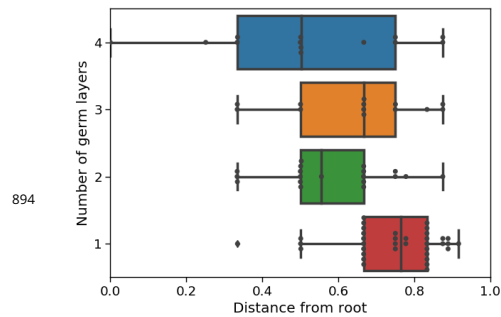




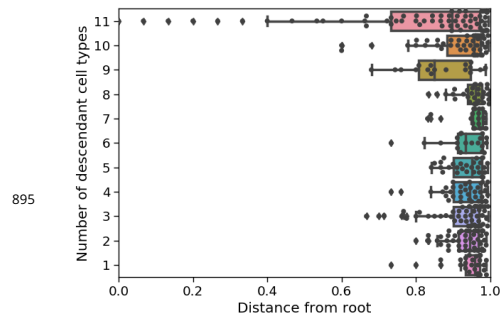
**Figure 7-Figure supplement 4.** Internal node times versus number of descendant germ layers in the estimated tree for the first adult fish using Camin-Sokal parsimony and *chronos*.



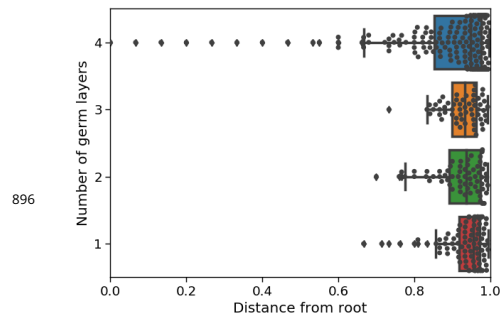
**Figure 7-Figure supplement 5.** Internal node times versus number of descendant cell types in the estimated tree for the second adult fish using Camin-Sokal parsimony and *chronos*.



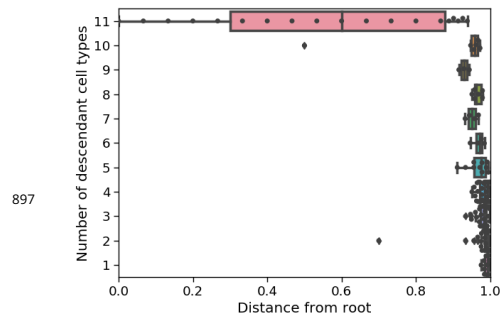
**Figure 7-Figure supplement 6.** Internal node times versus number of descendant germ layers in the estimated tree for the second adult fish using Camin-Sokal parsimony and *chronos*.



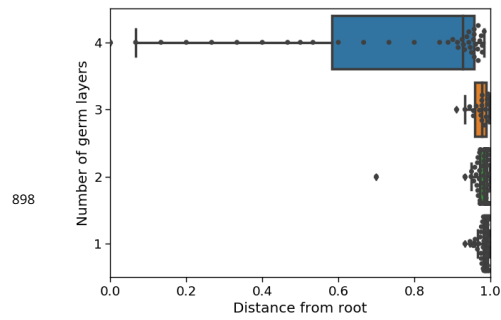
895  
**Figure 7-Figure supplement 7.** Internal node times versus number of descendant cell types in the estimated tree for the first adult fish using neighbor-joining and *chronos*.



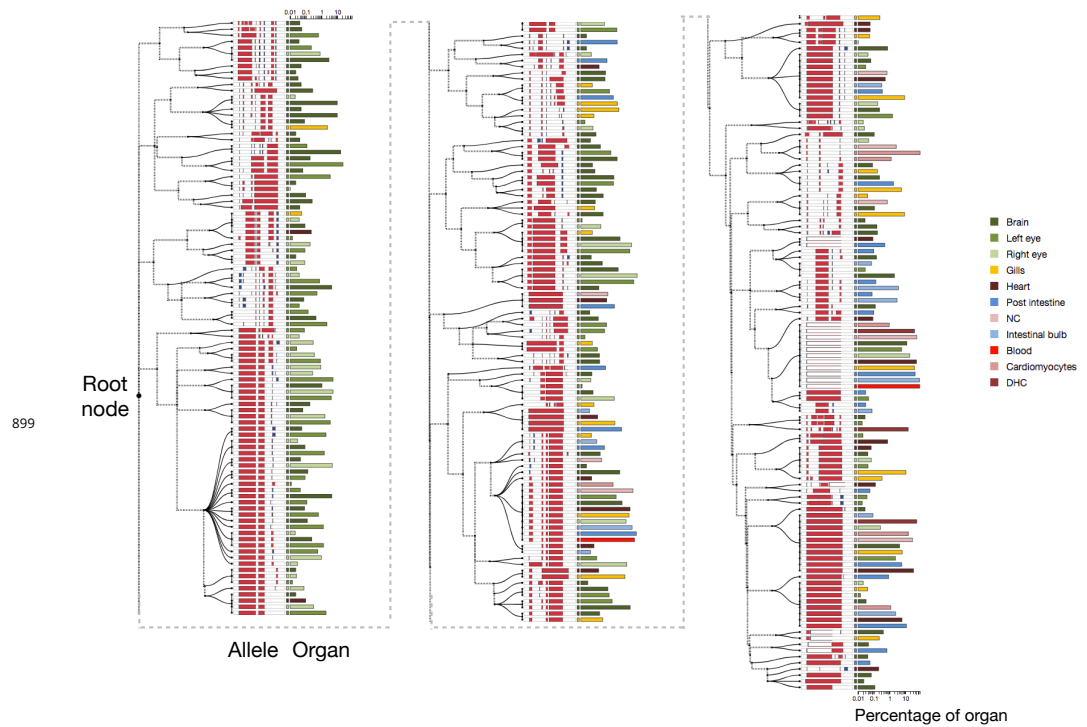
896  
**Figure 7-Figure supplement 8.** Internal node times versus number of descendant germ layers in the estimated tree for the first adult fish using neighbor-joining and *chronos*.



897  
**Figure 7-Figure supplement 9.** Internal node times versus number of descendant cell types in the estimated tree for the second adult fish using neighbor-joining and *chronos*.



**Figure 7-Figure supplement 10.** Internal node times versus number of descendant germ layers in the estimated tree for the second adult fish using neighbor-joining and *chronos*.



**Figure 8-Figure supplement 1.** Estimated cell lineage tree for all alleles from the second adult zebrafish using GAPML.