Submitted to the Annals of Applied Statistics

SUPPLEMENT TO: "ESTIMATION OF CELL LINEAGE TREES BY MAXIMUM-LIKELIHOOD PHYLOGENETICS"

By Jean Feng, William S. DeWitt III, Aaron McKenna, Noah Simon, Amy Willis, Frederick A. Matsen IV

APPENDIX A: USEFUL GUIDES FOR READING THE PROOFS

Symbol	Description	Eq.
$X_{\mathtt{N}}(t)$	Markov process along branch ending with node `N`	
$a_{\mathtt{N}}$	Observed allele at leaf `N`	
pos(j)	Positions of target j in the unmodified barcode	
$c(j)$	Cut site of target j	
Leaves(`N`)	Leaves of node `N`	
Desc(`N`)	Descendants of node `N`	
TargStat(a)	Status of targets in allele a. 1 in position j indicates target j is inactive	(2)
IT$[p_0, p_1, s, j_0, j_1]$	Indel tract that cuts targets j_0 and j_1, deletes positions p_0 to $p_1 - 1$, inserts s	
TT$[j_0', j_0, j_1, j_1']$	Target tract: all indel tracts that cut targets j_0 and j_1 and deactivate targets j_0' to j_1'	(3)
TT(d)	The target tract that indel tract d belongs to	
WC$[j_0, j_1]$	Wildcard: any indel tract that only deactivates targets with indices j_0 to j_1	(6)
SGWC$[p_0, p_1, s, j_0, j_1]$	Singleton-wildcard: union of an indel tract and its inner wildcard	(7)
AncState(`N`)	The set of likely ancestral states for node `N`	(5)

Table S1
Notation used in this paper

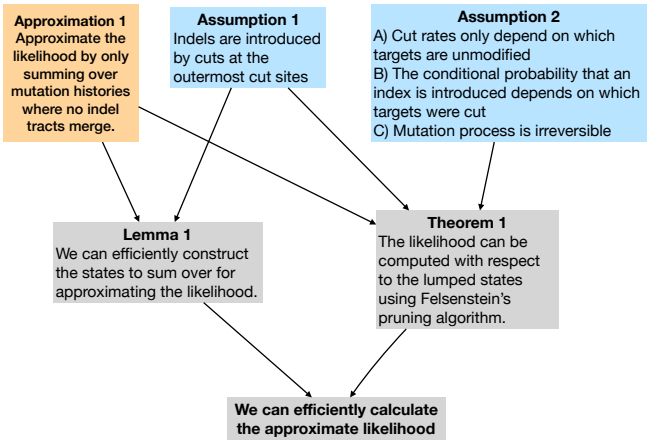


Fig S1: A guide for how assumptions, approximations, and derived results connect and lead to our final algorithm for approximating the likelihood.

APPENDIX B: CALCULATING POSSIBLE ANCESTRAL STATES

B.1. Proof for Lemma 1. First, we present the following lemma.

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

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

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

We now present the proof for Lemma 1.

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

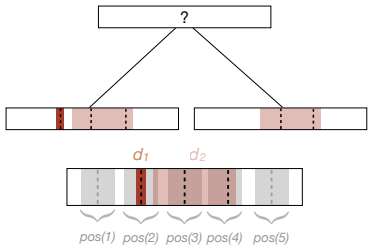


Fig S2: **Bottom**: An example of an allele with two indel tracts d_1 and d_2 where d_1 must have been introduced before d_2, because d_1 cuts target 2 while d_2 deactivates target 2 through 4. **Top**: A two-leaf tree where one leaf is the example allele and the other leaf is an allele with only d_2. Since d_1 must be introduced before d_2, the only possible ancestral state of the parent is an unmodified allele. On the other hand, if d_2 did not overlap with pos(2), we can simply take the intersection of the two alleles to get a possible ancestral state.

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

$$\text{(S1)} \quad \text{AncState}(\mathtt{N}) = \text{AncState}(\mathtt{C}_1) \cap \text{AncState}(\mathtt{C}_2)$$

$$\text{(S2)} \quad \subseteq \text{Alleles}\left(\bigcup_{m_1=1}^{M_{\mathtt{C}_1}} D_{\mathtt{C}_1,m_1}\right) \cap \text{Alleles}\left(\bigcup_{m_2=1}^{M_{\mathtt{C}_2}} D_{\mathtt{C}_2,m_2}\right)$$

$$\text{(S3)} \quad = \text{Alleles}\left(\bigcup_{m_1=1}^{M_{\mathtt{C}_1}} \bigcup_{m_2=1}^{M_{\mathtt{C}_2}} D_{\mathtt{N},m_1,m_2}\right)$$

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

$$\text{(S4)} \quad \text{targ}_{\text{start}}(D_{\mathtt{N},m_1,m_2}) \geq \max\left(\text{targ}_{\text{start}}(D_{\mathtt{C}_1,m_1}), \text{targ}_{\text{start}}(D_{\mathtt{C}_2,m_2})\right)$$

$$\text{(S5)} \quad \text{targ}_{\text{end}}(D_{\mathtt{N},m_1,m_2}) \leq \min\left(\text{targ}_{\text{end}}(D_{\mathtt{C}_1,m_1}), \text{targ}_{\text{end}}(D_{\mathtt{C}_2,m_2})\right).$$

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

B.2. Calculating AncState(·) exactly. For efficiency reasons, we are not satisfied with computing supersets of AncState(·); rather, we would like to concisely express the set of alleles that is exactly equal to AncState(·). The only case in which the algorithm computes a strict superset of AncState(N)

is when the alelles observed at the leaves of $\mathtt{N}$ imply that the observed indel tracts must be introduced in a particular order. For example, if an allele has indel tracts d_1 and d_2, we know that d_1 must be introduced before d_2 if d_1 cuts target j and d_2 deactivates target j (Figure S2). To indicate such orderings, we use the notation $d \in a \Rightarrow d' \in a$ to denote that "if indel tract d is in allele a, then indel tract d' must also be in a." The set of alleles respecting this ordering constraint is denoted

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

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

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

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

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

Then

(S6)

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

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

Similarly, for any internal node $\mathtt{N}$*, we can also write* $\text{AncState}(\mathtt{N})$ *in the form of* (S6)*. If node* $\mathtt{N}$ *has children nodes* $\mathtt{C}_1$*, ...,* $\mathtt{C}_K$*,* $\{D_{\mathtt{N},m}\}_{m=1}^{M_{\mathtt{N}}}$ *are pairwise disjoint wildcards and/or singleton-wildcards satisfying* (9) *and*

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

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

We give a proof by induction. Recall the definition of AncState, which is given in (5). Then it is easy to see that the base case at the leaf nodes, specified by (S6), holds. Any allele $a \preceq a_{\mathtt{L}}$ must be composed of observed indel tracts in $a_{\mathtt{L}}$ and/or indel tracts masked by the the observed indel tracts in $a_{\mathtt{L}}$. It also must respect all the ordering requirements specified in $\text{Orderlist}_{\mathtt{L}}$, since an indel tract cannot be introduced if its target is deactivated already. Therefore the left hand side is a subset of the right hand side. Next, consider any allele a from the right hand side. To see that $a \preceq a_{\mathtt{L}}$, note that we can apply the remaining indel tracts from $a_{\mathtt{L}}$ (i.e. those not in a) in the order they were introduced into $a_{\mathtt{L}}$.

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

APPENDIX C: LUMPABILITY

C.1. Proof of lumpability. Under Assumption 2, we can show that the instantaneous transition rate from any allele in $g(b;\mathtt{N})$ to the set $g(b';\mathtt{N})$ is the same. Assuming the transition rate between lumped states $g(b;\mathtt{N})$ and $g(b';\mathtt{N})$ is nonzero, we show that there are two types of transitions. Either the only transition between the two lumped states is through an indel tract d observed at the leaf nodes; or all indel tracts from a set of target tracts are valid transitions between the lumped states.

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

$$
\begin{aligned}
q_{\text{lump}}\left(g(b;\mathtt{C}), g(b';\mathtt{C})\right) &= \lim_{\Delta\to 0} \frac{\Pr\left(X_{\mathtt{C}}(\Delta) \in g(b';\mathtt{C}) | X_{\mathtt{C}}(0) = a\right)}{\Delta} \\
&= \lim_{\Delta\to 0} \frac{\Pr\left(X_{\mathtt{C}}(\Delta) \in g(b';\mathtt{C}) | X_{\mathtt{C}}(9) = a'\right)}{\Delta}.
\end{aligned} \tag{S7}
$$

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

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

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

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

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

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

If d is an indel tract that can be introduced to the allele $a \in g(b;\mathtt{C})$ and $\text{Apply}(a,d)$ has target status b', then we can introduce the same indel tract to any other allele $a' \in g(b;\mathtt{C})$ and $\text{Apply}(a',d)$ will also have the target status b'. Therefore we have proven that (S7) must hold for all $a, a' \in g(b;\mathtt{C})$.

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

(S8)

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

One of the following two cases must be true:

1. From the decomposition (S6) of AncState($\mathtt{C}$), there is only one indel tract d in the sets $D_{\mathtt{C},m}$ for $m = 1, ..., M_{\mathtt{C}}$ such that $\text{Apply}(a,d) \in g(b',\mathtt{C})$ for all $a \in g(b,\mathtt{C})$. d cannot be from a wildcard or the inner wildcard of a singleton-wildcard since this would contradict the fact that d is the only indel tract in $\{D_{\mathtt{C},m}\}$ such that $\text{Apply}(a,d) \in g(b',\mathtt{C})$ for all $a \in g(b,\mathtt{C})$. Therefore d must be the singleton for some singleton-wildcard $D_{\mathtt{C},m}$. In other words, the only possible transition from $g(b;\mathtt{C})$ to $g(b';\mathtt{C})$ is via the indel tract d.
2. Otherwise, for some target tract τ, there are at least two indel tracts in $d, d' \in \tau$ in the sets $D_{\mathtt{C},m}$ for $m = 1, ..., M_{\mathtt{C}}$ that deactivate targets $b' \setminus b$ such that $\text{Apply}(a,d) \in g(b',\mathtt{C})$ and $\text{Apply}(a,d') \in g(b',\mathtt{C})$ for all $a \in g(b,\mathtt{C})$. In this case, d and d' must be from a wildcard or the inner wildcard of a singleton-wildcard (d and d' cannot both be from a singleton of a singleton-wildcard since $d \neq d'$). Therefore every indel tract d in τ satisfies $\text{Apply}(a,d) \in g(b',\mathtt{C})$ for all $a \in g(b,\mathtt{C})$.

Therefore (S8) simplifies to

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

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

C.2. Proof for Theorem 1.

PROOF. For any internal node, we know that

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

(We do not need to sum over the partition $g(\mathsf{other};\mathtt{N})$ since it contributes zero probability.) By irreversibility of the mutation process, $g(b;\mathtt{N}) \subseteq g(b;\mathtt{C})$ if $\mathtt{C}$ is a child of $\mathtt{N}$. By (S7) in Lemma 4,

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

which means (15) also holds for node $\mathtt{N}$. □

APPENDIX D: TUNING PENALTY PARAMETERS

By varying the value of the penalty parameters κ_1 and κ_2, we can control the trade-off between minimizing the penalty versus maximizing the log likelihood. Choosing appropriate values is crucial for estimation accuracy. A common approach for tuning penalty parameters is to use cross-validation (Arlot and Celisse, 2010); we use this procedure whenever possible. Note that we keep the tree topology fixed when tuning the penalty parameter.

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

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

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

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

To partition the leaves, we randomly select a subset of leaf children of each multifurcating node to put in the validation set S^c. We partition leaves in this manner, rather than simply dividing the leaves in half, because we must be able to evaluate (or closely approximate) (S10) at the end of Algorithm 2 using the fitted branch length and mutation parameters. That is, we must be able to regraft the leaves in the set S^c onto the fitted tree. Regrafting is easy for the leaves in our specially-constructed set: The parent node of each leaf in S^c must be located somewhere along the caterpillar spine corresponding to its original multifurcating parent. In our implementation, we chose to regraft the leaves to the midpoints of their corresponding caterpillar spines. The regrafting procedure is illustrated in Figure S3. Note that we do not tune the branch lengths of these validation leaves since it amounts to peeking at the validation data. In our simulations, we found that when tuning the branch lengths to maximize the unpenalized (or penalized) log likelihood, we nearly almost always choose the smallest penalty parameter since it prioritizes maximizing the likelihood and, therefore, (S10).

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

$$\begin{aligned} &E\left[\log \Pr(X_{\mathtt{L}}(t_{\mathtt{L}}) \forall\, \mathtt{L} \in S^c \mid X_{\mathtt{L}}(t_{\mathtt{L}} \forall\, \mathtt{L} \in S); \ell^*, \theta^*); \ell^*, \theta^*\right] \\ &\geq E\left[\log \Pr(X_{\mathtt{L}}(t_{\mathtt{L}}) \forall\, \mathtt{L} \in S^c \mid X_{\mathtt{L}}(t_{\mathtt{L}} \forall\, \mathtt{L} \in S); \ell, \theta); \ell^*, \theta^*\right] \qquad \forall \ell, \theta \end{aligned} \tag{S9}$$

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

From a theoretical standpoint, using (S9) to select penalty parameters makes the most sense if we have an unbiased estimate of the expected conditional probability. Unfortunately, in our setting, the conditional probability

in (S10) is actually a biased estimate since the fitted parameters depended on the observed alleles at leaves S. Nonetheless, in simulations (where the truth is known), this biased estimate seemed to work well, as the selected penalty parameter was typically close to the best penalty parameter.

Algorithm 2 Cross validation for a single barcode

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

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

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

$$\hat{\boldsymbol{\kappa}} = \arg\max_{\boldsymbol{\kappa}} \Pr\left(X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in S^c \mid X_{\mathtt{L}}(t_{\mathtt{L}}) = a_{\mathtt{L}} \forall\, \mathtt{L} \in S; \hat{\boldsymbol{\ell}}_{\boldsymbol{\kappa}}, \hat{\boldsymbol{\theta}}_{\boldsymbol{\kappa}}\right). \tag{S10}$$

To perform the entire GAPML estimation procedure, we must tune the penalty parameters and the topology of the tree. Our full algorithm alternates between tuning the penalty parameters for a fixed tree topology and running a single iteration of Algorithm 1 for a fixed penalty parameter. After the penalty parameters are stable, we keep them fixed and only run Algorithm 1.

APPENDIX E: SPECIFIC MODEL IMPLEMENTATION

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

$$d \text{ has a short left deletion if } p_0 \in \mathrm{pos}(j_0) \tag{S11}$$

$$d \text{ has a long left deletion if } p_0 \in \mathrm{pos}(j_0 - 1). \tag{S12}$$

We define short and long deletions to the right similarly.

E.1. Rate parameterization. To parameterize the rate at which a target tract $\tau = \mathrm{TT}[j_0', j_0, j_1, j_1']$ is introduced, we decompose the rate into

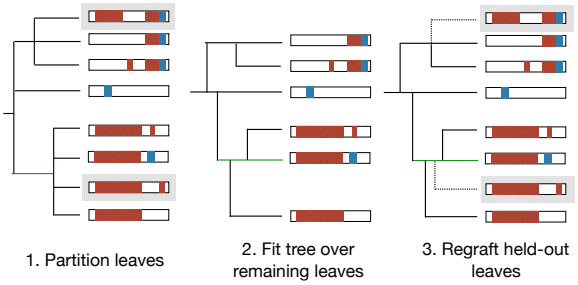


Fig S3: Cross-validation to tune penalty parameters with only one barcode. We split leaves into training and validation sets S and S^c, respectively as follows (left): For each multifurcating node, randomly select a subset of its children that are leaves to put in the "validation" set, denoted by the 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).

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 (S11) and (S12)):

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

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

We specify h_0 using the assumption that the cutting time for target j follows an exponential distribution with rate of $\lambda_j > 0$. For focal target cuts where $j_0 = j_1$, we define

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

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

X_{j_1} for targets j_0 and j_1 occur within a small window ϵ is approximately

$$p\left(|X_{j_0} - X_{j_1}| \leq \epsilon, \frac{X_{j_0} + X_{j_1}}{2} = t\right) \tag{S13}$$

$$\approx \Pr(|X_{j_0} - X_{j_1}| \leq \epsilon)\, p(X_{j_0} = X_{j_1} = t | X_{j_0} = X_{j_1}) \tag{S14}$$

$$\approx \omega(\lambda_{j_0} + \lambda_{j_1}). \tag{S15}$$

Our approximation for the first line using the second improves as $\epsilon \to 0$. Next, the first component in (S14) approaches zero as $\epsilon \to 0$ and does not vary much for different values of $\lambda_{j_0}, \lambda_{j_1}$ if ϵ is sufficiently small. Thus we approximate it using the same value of ω for all targets. The second component in (S14) corresponds to an exponential distribution with the rate $\lambda_{j_0} + \lambda_{j_1}$.

We can interpret ω in two ways. First, ω controls how often a double cut is introduced. In an unmodified barcode, the relative rate that a double cut is introduced versus a single cut is $\omega \sum_{j_1<j_2}(\lambda_{j_1} + \lambda_{j_2})$ versus $\sum_{j=1}^{M} \lambda_j$. The second interpretation, based on (S15), is that ω serves as a proxy for ϵ: Larger ω indicates that an inter-target deletion can be introduced by two cuts spaced farther apart in time.

E.2. Deletion/insertion sequence parameterization. The second major component of the GESTALT mutation model specifies the conditional probability of introducing a particular indel tract given target tract $\tau = \mathrm{TT}[j_0', j_0, j_1, j_1']$. An indel tract can be represented by its deletion lengths to the left and right and the insertion sequence. We will suppose that the probability of a single insertion sequence is uniform over all possible nucleotide sequences of that length.

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

In our implementation, X_0 and X_1 are zero-inflated truncated negative binomial RVs for short trims and zero-inflated truncated poisson RVs for long trims. X_2 is a zero-inflated negative binomial RV. Let $\mathrm{NB}(m, q)$ denote the negative binomial distribution, which is the distribution for the number

of successes until m failures are observed and q is the probability of success. Let $\text{Pois}(r)$ be a poisson distribution with mean r. For $i = 0, 1$, define $\Pr(X_i = x_i|\tau)$ as

(S16)
$$\begin{cases} p_{i,\mathbb{1}\{j_0=j_1\}} & \text{if } x_i = 0 \\ (1 - p_{i,\mathbb{1}\{j_0=j_1\}}) \Pr\left(X_i = x_i - x_{\tau,\min,i} - 1 | X_i < x_{\tau,\max,i} - 1; \text{NB}(m_i, q_i)\right) & \text{if } x_i > 0, j_0' = j_0 \\ (1 - p_{i,\mathbb{1}\{j_0=j_1\}}) \Pr\left(X_i = x_i - x_{\tau,\min,i} - 1 | X_i < x_{\tau,\max,i} - 1; \text{Pois}(r_i)\right) & \text{if } x_i > 0, j_0' \neq j_0 \end{cases}.$$

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

(S17)
$$\Pr(X_2 = x_2|\tau) = \begin{cases} p_2 & \text{if } x_2 = 0 \\ (1 - p_2) \Pr\left(X_2 = x_2 - 1; \text{NB}(m_2, q_2)\right) & \text{if } x_2 > 0 \end{cases},$$

where p_2 is the probability of inserting zero nucleotides.

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

(S18)
$$\begin{aligned} &\Pr\left(X_0 = x_0, X_1 = x_1, X_2 = x_2 \mid \tau\right) \\ &= \begin{cases} \frac{\Pr(X_0=x_0|\tau) \Pr(X_1=x_1|\tau) \Pr(X_2=x_2|\tau)}{\Pr(X_0+X_1+X_2>0|\tau)} & \text{if } j_0' = j_0 = j_1 = j_1' \\ \Pr(X_0 = x_0|\tau) \Pr(X_1 = x_1|\tau) \Pr(X_2 = x_2|\tau) & \text{otherwise} \end{cases}. \end{aligned}$$

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

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

APPENDIX F: EXPERIMENTAL SETTINGS

F.1. Data pre-processing. The data in this paper are all from McKenna, Findlay and Gagnon et al. (2016) and are available at the Gene Expression Omnibus under GSE81713. We use the aligned data where each allele was described with the observed insertion/deletions (indels) at each target. Each CRISPR target can only be modified once and indels can only be introduced via a double-stranded break at a target cut site, so we further processed the aligned data accordingly: merging indels if there were more than one associated with a given target, and extending the deletion lengths and insertion sequences so that a target cut site was nested within each indel. For this paper, we assume that the processed data is correct and do not attempt to model the effects of alignment error.

F.2. Code implementation. The code is implemented in Python using Tensorflow (Abadi, Agarwal and Barham et al., 2015). We maximize the penalized log likelihood using Adam (Kingma and Ba, 2014). We chose to use an automatic differentiation software since it has enables us to quickly iterate on the specific GESTALT model, without needing to recode the gradients. Just as automatic differentiation has greatly accelerated deep learning research (Baydin, Pearlmutter and Radul et al., 2018), we believe that it can also accelerate the development of maximum likelihood estimation methods in phylogenetics.

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

F.3. Comparison Methods. We use PHYLIP version 3.697 (Felsenstein, 1995), the neighbor-joining algorithm in Bio.Phylo (Biopython version 1.72) (Talevich, Invergo and Cock et al., 2012), and the chronos function in R package `ape` version 5.2 (Paradis and Schliep, 2018).

F.4. Simulation setup and additional results. For the results in Figure 10, the data was simulated with 5 synchronous cell division cycles followed by a birth-death process where the birth rate decayed at a rate of $\exp(-18t)$. The barcode was composed of six targets with $\boldsymbol{\lambda} = 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6$. The weight ω was set to 0.06 so that 20%

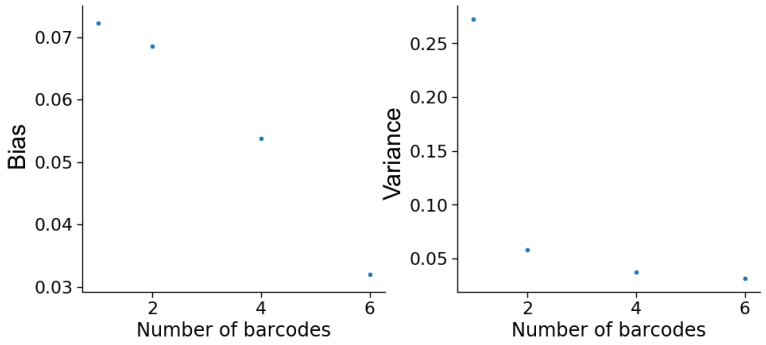


Fig S4: Plot of bias and variance of the mutation parameter estimates versus number of barcodes.

of the unique observed indel tracts were due to double cuts. We sampled 8% of the leaves so that the average number of unique observed alleles was around 100 leaves. We refer to this simulation setup as Simulation A. We ran 20 replicates of Simulation A. In addition, plots of the bias and variance of the mutation parameter estimates are given in Figure S4.

The results in Figure 1 are from a larger simulation, which we will refer to as Simulation B, that is closer to the data collected in McKenna et al. (2016). Since zebrafish undergo around 11 synchronous cell division cycles at the beginning of development, this larger simulation entailed 9 synchronous cell division cycles followed by a birth-death process. We simulated with a barcode composed of ten targets. The resulting tree had on average around 200 leaves. We ran GAPML for 8 topology tuning iterations; at each iteration, we consider at most 15 SPR moves. The displayed results are from 20 replicates.

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

Using our simulation engine, we compare two very simple barcode design ideas: a single barcode with many targets, recommended in Salvador-Martínez, Grillo and Averof et al. (2018), or many identical barcodes. However we believe the latter is more effective since spreading the targets over

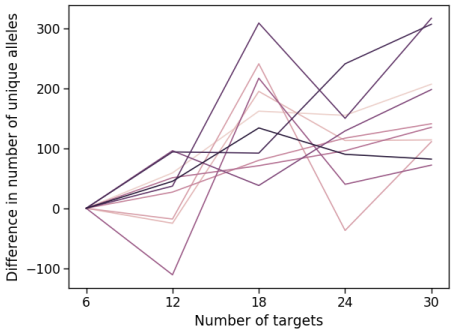
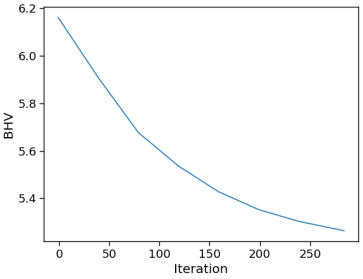


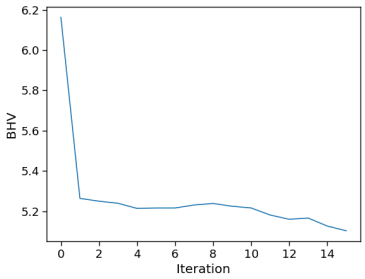
Fig S5: We compare the number of unique alleles obtained in GESTALT using a single barcode with many targets versus splitting the targets over multiple independent barcodes. The alleles are simulated on a full binary tree with 1024 leaves. Each line corresponds to a simulation where we iteratively add six targets, either by extending the single barcode or adding another barcode with six targets. A positive difference that the multiple-barcode design has more unique alleles, and vice versa.

separate barcodes tends to create more unique alleles. In particular, the inter-target deletions tend to be shorter, which means fewer existing mutations are deleted and fewer targets are deactivated. To test this idea, we compared to the two design options in a simulation setup where we iteratively increased the number of targets by six, i.e. add six targets to the existing barcode or add a new barcode with six targets. Here we observe all 1024 leaves of a full binary tree with 10 levels. All targets had the same single-cut rate. We calibrated the double-cut weight ω to be around 18% for both barcode designs – this slightly favors the single-barcode design since it would have a higher rate of double cuts *in vivo* compared to a multiple-barcode design. Nevertheless, we find in our simulations that splitting the targets over separate barcodes tends to result in a much larger number of unique alleles than using a single barcode (Figure S5). At 30 targets, the multiple-barcode design has roughly 200 more unique alleles on average than the single-barcode design. Another reason we prefer the multiple-barcode design is that our model and tree estimates improve as the number of independent and identical barcodes increases, as illustrated in Figure 10.

Next, to better understand our algorithm GAPML, we show in-depth simulation results from a single replicate (Figure S6). Here we use the settings from Simulation B. Starting from the initial tree topology, the algorithm



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



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

Fig S6: Examples of how the BHV distance changes as the algorithm proceeds for one simulation replicate from the ten-target setting.

tunes the branch lengths and mutation parameters to maximize the penalized likelihood. During the gradient descent algorithm, the BHV distance of the tree estimate decreases (Figure S6a). In addition, we see that the BHV distance of the tree estimate decreases as Algorithm 1 iteratively performs SPR moves to update the tree topology.

Our method searches over the maximally parsimonious trees since they tend to have the highest penalized log likelihood. To justify this restricted search, we compared the penalized log likelihood for tree topology candidates of different parsimony scores, where the data was generated using Simulation A. To generate tree topologies with different parsimony scores, we started with the maximally parsimonious tree fit from Camin-Sokal and iteratively applied random SPR moves. For each of tree rearrangement, we fit a model by maximizing the penalized log likelihood. The penalty parameter is the same across all rearrangements. As seen in Figure S7, the most parsimonious trees have the highest penalized log likelihoods. Since our method aims to select a tree topology that maximizes the penalized log likelihood, it would not benefit from considering SPR moves that make the tree less parsimonious; instead, considering these additional moves would make the method much slower.

F.4.1. *Consistency.* Here we evaluate the consistency of our method through a simulation study. We consider barcodes with 6 targets and in-

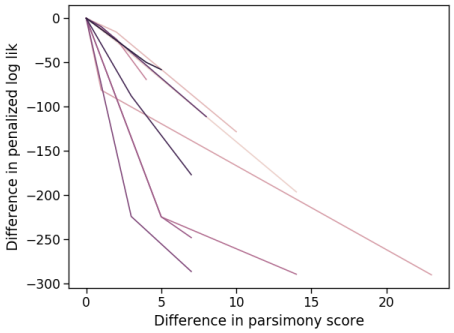


Fig S7: We compare the maximized penalized log likelihood of maximally parsimonious trees to less parsimonious trees. Each simulation replicate, represented by each line, shows four candidate tree topologies, starting from the most parsimonious one ($x = 0$) to increasingly less parsimonious ones (large 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.

sert 16, 64, and 256 barcodes into the embryo cell. We simulated trees with roughly 110 leaves and sample 8% of the leaves. We then evaluated the BHV distance between the estimated tree and the true tree, as well as the mean squared error between the estimated and true mutation parameters as defined in Section E. As seen Figure S8, the tree and the mutation parameter estimates improve with the number of barcodes.

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

F.4.2. *Sensitivity analysis.* We now evaluate the sensitivity of our algorithm to errors when sequencing the mutated barcodes or ambiguities when resolving indels. Recall that the main manuscript assumed that all indels were sequenced and resolved accurately. This can be difficult to do in practice because there might be mixtures of insertions and deletions that cannot be resolved unambiguously.

If the primary goal is to recover tree topology and branch lengths, we find that the GAPML algorithm is relatively robust to errors when resolving indels. We simply require the method for resolving indels to map each unique

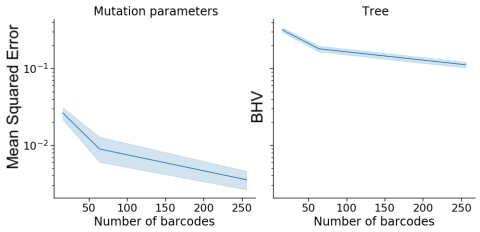


Fig S8: Mean squared error of estimated mutation parameters (left) and BHV distance between the estimated and true trees (right). Data is generated using 16, 64, and 256 barcodes. Plot reflects 20 simulation replicates.

indel d to the same indel d' from the same target tract (at least most of the time). This robustness property is a natural consequence of Assumption 2, which states that the mutation rate factors into the target cut rate and the insertion/deletion probabilities. As such, our ability to estimate the tree topology and branch lengths mostly depend on our ability to accurately estimate the target cut rates.

We illustrate the robustness of our algorithm in the following simulation study. For each unique indel, we introduced small perturbations to the resolved left deletion lengths, right deletion lengths, and the inserted sequence, each with some probability p. Using this mapping, each sampled allele is resolved as a collection of these incorrectly observed indels. We varied this error rate p from 0% to 20%. The barcode here is composed of six targets; four barcodes are inserted into the embryo. We simulated trees with 400 leaves on average and sample 5% of the leaves. As expected, the BHV distance and the internal node time correlation are relatively constant across the different error rates (Figure S9).

Finally, we note that this robustness property does not hold if the goal is to accurately estimate the insertion/deletion parameters (e.g. those defined in Section E.2). In that case, the indels need to be resolved correctly.

F.5. Zebrafish data analysis. For the zebrafish analyses, we estimated the tree over at most 400 randomly selected alleles (without replacement). 50% of the fish in this dataset had fewer than 400 alleles and the median number of unique alleles over the zebrafish datasets was 443. 25% of the fish in this dataset had more than 1000 alleles. We limit the number of alleles analyzed due to runtime restrictions.

To test if the fitted trees are recovering similar developmental relationships across fish rather than random noise, we ran a permutation test comparing the correlation between tissue distances from the estimated trees

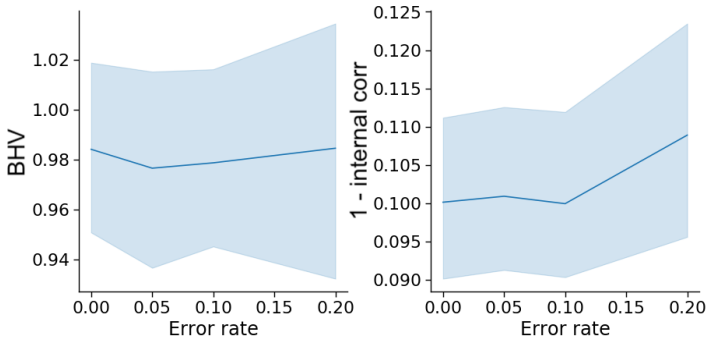


Fig S9: BHV distance and internal node time correlation between estimated and true trees when the indels are resolved with error rates 0%, 5%, 10%, and 20%. Plots reflect 50 simulation replicates.

to that from randomly-estimated trees over randomly-shuffled data. More specifically, for a given tree topology, we randomly permute the observed alleles at the leaves. Each allele is associated with the number of times it is observed in each tissue type; we randomly shuffle these abundances over the possible tissue types within each allele. Finally, we randomly assign branch lengths along the tree by drawing samples from a uniform distribution and using the t-parameterization of Gavryushkin and Drummond (2016) to assign branch lengths. The correlation between tissue distances in these random trees is close to zero. All permutation tests were performed using 2000 replicates.

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

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

For Figure 2, we bootstrapped fish replicates to estimate confidence intervals for the average correlation between estimated target cut rates.

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

TABLE S2

Fitted parameters in the adult fish as well as some other fish embryos. The parameters above the line are related to target cut rates and the ones below the line are related to the nucleotide deletion and insertion process. Short versus long deletions are defined in (S11) *and* (S12)*, respectively.*

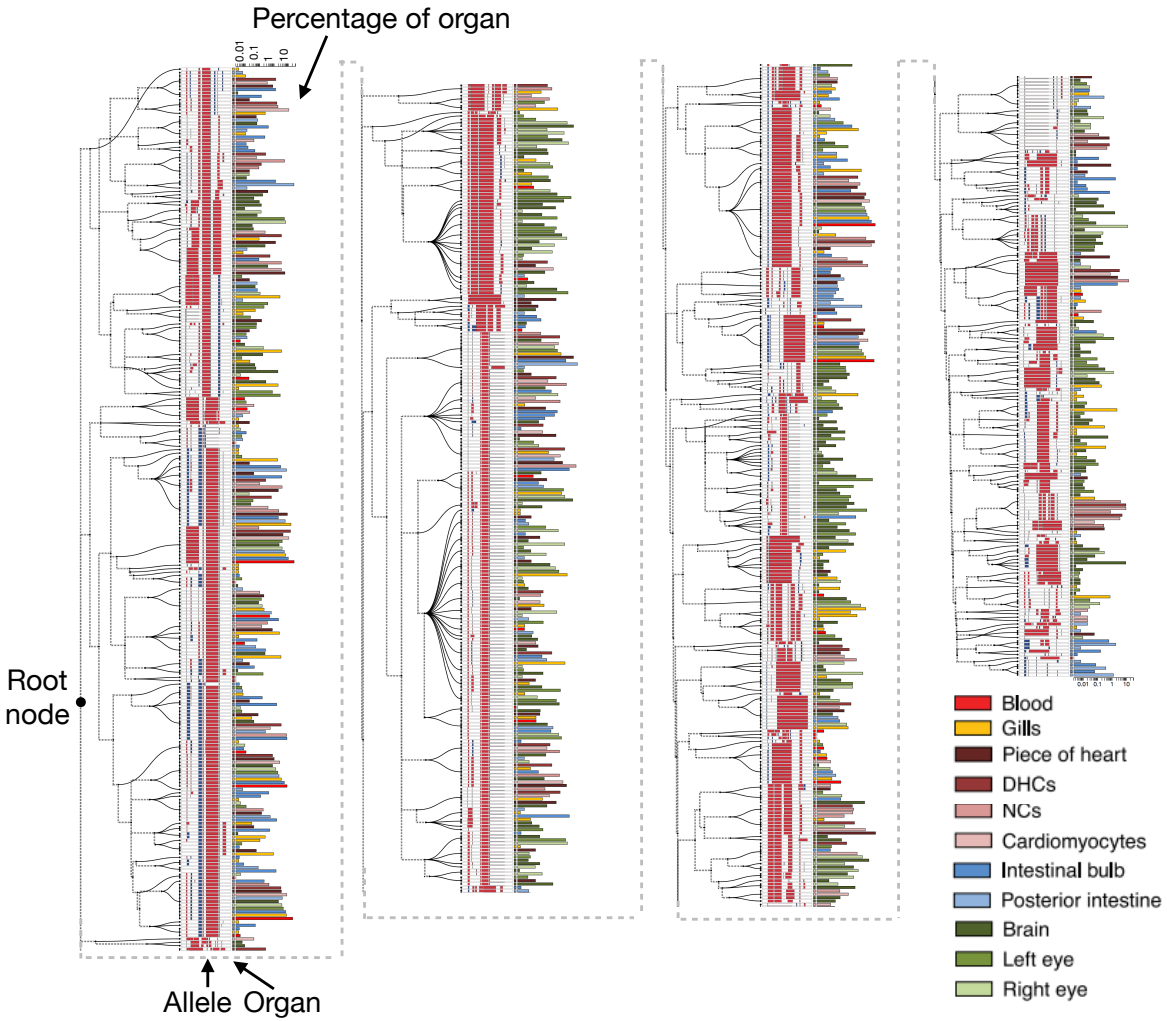


Fig S10: Estimated cell lineage tree for 400 randomly selected alleles from the first adult zebrafish using GAPML. Editing patterns in individual alleles are represented as shown previously. Alleles observed in multiple organs are plotted on separate lines per organ and are connected with stippled branches. Two sets of bars outside the alleles identify the organ in which the allele was observed and the proportion of cells in that organ represented by that allele (log10 scale). The dashed lines correspond to the caterpillar spines.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Stat. Surv.* **4** 40–79.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A. and Siskind, J. M. (2018). Automatic Differentiation in Machine Learning: a Survey. *J. Mach. Learn. Res.* **18** 1–43.

Felsenstein, J. (1995). *PHYLIP (phylogeny inference package), version 3.5 c.*

Gavryushkin, A. and Drummond, A. J. (2016). The space of ultrametric phylogenetic trees. *J. Theor. Biol.* **403** 197–208.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization.

McKenna, A., Findlay, G. M., Gagnon, J. A., Horwitz, M. S., Schier, A. F. and Shendure, J. (2016). Whole-organism lineage tracing by combinatorial and cumulative genome editing. *Science* **353**.

Paradis, E. and Schliep, K. (2018). ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics* **xx** xxx-xxx.

Salvador-Martínez, I., Grillo, M., Averof, M. and Telford, M. J. (2018). Is it possible to reconstruct an accurate cell lineage using CRISPR recorders?

Talevich, E., Invergo, B. M., Cock, P. J. A. and Chapman, B. A. (2012). Bio.Phylo: a unified toolkit for processing, analyzing and visualizing phylogenetic trees in Biopython. *BMC Bioinformatics* **13** 209.