Reconstruction of evolving gene variants and fitness from short sequencing reads

Max W. Shen[1,2,3,4], Kevin T. Zhao[2,3,4], David R. Liu[2,3,4]

SUPPLEMENTARY INFORMATION

Supplementary Table 1

Supplementary Notes

1. Theoretical results
2. Utility of unnormalized skew regularizer
3. Practical usage guidelines
4. Comparisons to related approaches

Supplementary References

Supplementary Table 1

Comparison of DNA sequencing strategies for diverse samples of genotypes at timepoints from directed evolution campaigns

	Typical accessibility*	Cost	Maximum post-alignment read length	Error (s.d.) without consensus correction	Relative Evoracle reconstruction performance for gene-length molecules†
Pooled Sanger	High	$10/sample	1 nt	~3%	80-89%
Illumina	Moderate	$1,000/run	600 nt	0.1%	88-96%
PacBio	Low	$1,000/run	10 - 100 kb	15%	-
Nanopore	Low	$100-$1,000/run	10 - 100 kb	20%	-

* Accessibility represents typical extent of adoption across labs, which includes typical ease of access to sequencing facilities, ease of use in lab, breadth of uses, and other factors

† Comparison of R^2 between inferred and observed full-length genotype frequencies across timepoints for *Cry1ac* and *TadA* campaigns, generalized to typical gene lengths.

Supplementary Note 1 | Theoretical results

Here, we describe some theoretical analysis on the genotype reconstruction problem and various aspects of Evoracle's design.

Lemmas 1 and 2 together motivate Evoracle's skew regularizer. Lemma 1 states that natural selection sorts genotypes by fitness in logarithmic time, while lemma 2 states that when genotypes are nearly sorted by fitness, the skew statistic monotonically increases over time under natural selection recurrence relationship. Together, lemmas 1 and 2 build a perspective where genotypes become unsorted by fitness when new mutants with high fitness are discovered by random mutation, but genotypes are quickly re-sorted (in logarithmic time) and re-enter time spans where the frequency skew increases monotonically. Natural selection thus often yields populations with highly skewed genotype frequencies. We leverage this natural sparsity structure with our regularizer.

Lemmas 3, 4, and 5 explore which parameters are identifiable under our lossy measurement process. In general, full-length genotype frequencies are not identifiable from our lossy linear measurement process. However, our work suggests that adding domain knowledge in the form of enforcing inferred frequencies to be consistent with natural selection produces a method capable of accurately inferring full-length genotypes in practice. This suggests a possible conjecture that identifiability may be restored with knowledge of underlying dynamics. We view lemmas 3, 4, and 5 as initial steps exploring this possibility. Lemma 3 states that the single highest fitness full-length genotype is identifiable under a surprisingly lenient condition. This result shows that a specific object can always be recovered despite lossy measurement. The fact that this object is the highest fitness genotype is of practical relevance. Lemma 4 states that the true fitness vector is identifiable from the full-length genotype frequency matrix, which is a result that does not involve the lossy measurement process. Lemma 4 thus provides a formal proof that unidentifiability is a direct result of the lossy measurement process, and not the natural selection recurrence relation. Finally, lemma 5 investigates a toy example of three full-length genotypes, and shows that the true genotype frequencies are identifiable using natural selection dynamics knowledge despite lossy measurements using one timepoint of full-length genotype frequencies and two or more timepoints of lossy measurements. It would be interesting to extend lemma 5's result to the case of no true genotype frequency knowledge, and larger number of genotypes, in future work.

Lemma 6 shows that our gradient-matching loss term that encourages fidelity of inferred states to the state transition process implicitly regularizes the L_1-norm of $\boldsymbol{s}$, the frequencies of genotypes when first introduced to the population. This aligns with the typically low mutation rates of 1×10^{-5} used in directed evolution.

Lemma 7 shows that our gradient-matching loss term that encourages fidelity of inferred states to the state transition process implicitly ensures that genotypes cannot rise to appreciable frequencies

in the population more than once, which is a constraint imposed by our state transition process on latent states, and enable us to bypass the challenging problem of inferring $\boldsymbol{s, z}$ and instead directly infer latent states. Lemma 7 also addresses potential concerns on model misspecification, by showing that a state transition process with at most one entrance is qualitatively similar to state transition processes where genotypes can enter multiple times.

Definitions and setup

Here, we set up notation describing our assumed data generative process, and state general properties of certain mathematical objects that are useful in later proofs. Evoracle operates in a non-linear dynamical system describing natural selection and mutation that introduces new genotypes to the population. In general, we use bold symbols to refer to vectors and matrices, and use bracketing indexing notation such that v[i] to denote the i-th element of a vector $\boldsymbol{v}$.

$\boldsymbol{x_t}$ is a vector of genotype frequencies at time t that represents a state updated by the non-linear dynamical system. We assume that there are G finite genotypes. We use $\boldsymbol{X}$ to refer to the matrix concatenating $\boldsymbol{x_1, x_2, \ldots, x_T}$ over T total timepoints.

The non-linear dynamical system includes an asexual natural selection recurrence relation commonly used in the literature:

$$\boldsymbol{x_{t+1}} = \frac{\boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{x_t}} \odot \boldsymbol{x_t} \tag{1}$$

where $\odot$ is element-wise multiplication of two vectors, and $\boldsymbol{w}$ is fitness vector with non-negative values. Without loss of generality, we assume that $\boldsymbol{w}$ is ordered such that $w[1] > w[2] > \ldots > w[G]$.

The corresponding differential equation is:

$$\boldsymbol{x_{t+1}} - \boldsymbol{x_t} = \frac{\boldsymbol{w} - \boldsymbol{w}^T \boldsymbol{x_t}}{\boldsymbol{w}^T \boldsymbol{x_t}} \odot \boldsymbol{x_t} \tag{2}$$

We observe data $\boldsymbol{y_t}$ through a lossy observation process $\boldsymbol{y_t} = \boldsymbol{B x_t}$ at time t where B is a rank-deficient matrix since the dimension of $\boldsymbol{y_t}$ is less than the dimension of $\boldsymbol{x_t}$. We use $\boldsymbol{Y}$ to refer to the matrix concatenating $\boldsymbol{y_1, y_2, \ldots, y_T}$ over T timepoints.

Let there be M distinct measurements, so that $\boldsymbol{B}$ has shape (M, G). We use the notation $\boldsymbol{B}_m \subset \{1, \ldots, G\}$ for an event m to denote the subset of genotypes that contain event m. For example, m could be the event that position 1 has a mutation to 'A', or that positions 1-4 are equal to 'ACGT'.

Theoretical properties

Lemma 1 (Natural selection exponentially enriches for the maximum fitness genotype) *If* $w[i] < w[j]$ *and* $x[i] \geq x[j]$*, then we will have* $x[i] < x[j]$ *after*

$$\tau > \log_{\frac{w[j]}{w[i]}}(\frac{x[i]}{x[j]}) \tag{3}$$

time steps.

Proof. We achieve $x[i] < x[j]$ at τ time steps when:

$$\begin{aligned} x[i]\prod_{t=1}^{\tau}\frac{w[i]}{\boldsymbol{w}^T\boldsymbol{x_t}} &< x[j]\prod_{t=1}^{\tau}\frac{w[j]}{\boldsymbol{w}^T\boldsymbol{x_t}} \\ x[i](\frac{w[i]}{w[j]})^{\tau} &< x[j] \\ \tau &> \log_{\frac{w[j]}{w[i]}}(\frac{x[i]}{x[j]}) \end{aligned} \tag{4}$$

The sign flips because $\log(\frac{w[i]}{w[j]}) < 0$.

Remarks. This result shows that, if the fitness difference between $w[1]$ and $w[2]$ is 10-fold, then $x[1]$ can overcome $x[2]$ that is 100-fold higher in two time steps.

Lemma 2 (Natural selection induces genotype frequency distributions with increasing skew) *For any* $\boldsymbol{w}^T\boldsymbol{x_t}$*, let there be an index* d *where* $w[d] > \boldsymbol{w}^T\boldsymbol{x_t}$ *and* $w[d+1] \leq \boldsymbol{w}^T\boldsymbol{x_t}$*. If* $x_t[i] > x_t[j]$ *for all* i *in* $\{1,\ldots,d\}$ *and all* j *in* $\{d+1,\ldots,G\}$*, and* $min(\{x_t[i]|i \in \{1,\ldots,d\}\}) > 2/G$*, then*

$$skew(\boldsymbol{x_{t+1}}) > skew(\boldsymbol{x_t}) \tag{5}$$

where we define the skew of a genotype frequency vector as $skew(\boldsymbol{x_t}) = \| \boldsymbol{x_t} - 1/G \|_3^3$. This term corresponds to the unnormalized statistical skew around the mean frequency 1/G.

Remarks. The initial condition is a relaxation of requiring p to be exactly sorted in the same order as $\boldsymbol{w}$: we only require that the pivot point defined by $\boldsymbol{w}^T\boldsymbol{x_t}$ in fitness space also defines a pivot point in genotype frequency space that splits genotypes into two groups: one group with relatively higher initial frequency that will element-wise increase, and the other group with relatively lower initial frequency that will element-wise decrease. By theorem 1, sorting occurs in logarithmic time for all mismatched pairs. Moreover, sorting across all pairs occurs simultaneously. In practice, high selection stringency and rapid

generation time in directed evolution experiments suggest that these conditions are mild and achievable in practice.

Also, note that once the initial conditions are satisfied at some t, they will continue to be satisfied for any $t' > t$ under the natural selection recurrence relation. The conditions will be broken if, for example, a new rare genotype with fitness greater than the whole population is introduced by spontaneous mutation.

Proof. We begin by considering the skew function for a single term, $skew(x_t[d]) = (x_t[d] - \frac{1}{G})^3$ and note that it is a convex function when $x_t[d] \geq 1/G$ and concave when $x_t[d] \leq 1/G$. We will use the following convexity bounds.

Observation. (Convexity bounds). Let $f(x)$ be a convex function and $y < x$. Then,

$$(x - y)f'(x) \geq f(x) - f(y) \tag{6}$$

and

$$(x - y)f'(y) \leq f(x) - f(y) \tag{7}$$

Proof.

By the definition of convexity, we have for $0 < \lambda < 1$:

$$\begin{array}{ll} f((1-\lambda)x + \lambda y) & \leq (1-\lambda)f(x) + \lambda f(y) \\ f((1-\lambda)x + \lambda y) - f(y) & \leq (1-\lambda)f(x) + \lambda f(y) - f(y) \\ & \leq (1-\lambda)(f(x) - f(y)) \end{array} \tag{8}$$

We divide by $(1-\lambda)(x-y)$ since it is not equal to zero.

$$\frac{f((1-\lambda)x+\lambda y)-f(y)}{(1-\lambda)(x-y)} \leq \frac{(1-\lambda)(f(x)-f(y))}{(1-\lambda)(x-y)} \tag{9}$$

As $\lambda \to 1$, the left side approaches $f'(y)$, and we obtain:

$$f'(y) \leq \frac{f(x)-f(y)}{(x-y)}. \tag{10}$$

This shows equation 7. An analogous approach can be used to obtain equation 6.

We now continue to proving theorem 2. Comparing between time t and $t+1$, some terms $skew(x[d])$ may increase and other terms may decrease, subject to $\sum_d x_t[d] = 1$ for all t. Let $A = \{1, \ldots, d\}$ and $B = \{d+1, \ldots, D\}$. Then:

$$\delta = \sum_{d \in A} x_{t+1}[d] - x_t[d] \tag{11}$$

where each term $x_{t+1}[d] - x_t[d] > 0$ for all $d \in A$. The overall increase δ is balanced by:

$$-\delta = \sum_{d \in B} x_{t+1}[d] - x_t[d] \tag{12}$$

where each term $x_{t+1}[d] - x_t[d] < 0$ for all $d \in B$.

The plan of the proof is to show that the increase in skew from increasing $x_t[d]$ for all $d \in A$ by a total of δ is greater than the decrease in skew from any possible combination of changes in $x_t[d]$ for all $d \in B$ that sum to $-\delta$.

Note that for each term, in general, the change in skew will be:

$$skew(x_{t+1}[d]) - skew(x_t[d]) = (x_t[d] + \delta z_d - 1/G)^3 - (x_t[d] - 1/G)^3 \tag{13}$$

where $z_d < 0$ for all $d \in B$ and $\sum_{d \in B} z_d = -1$. Analogously, we have $z_d > 0$ for all $d \in A$ and $\sum_{d \in A} z_d = 1$.

Let a be the index $\min_d\{x_t[d] | d \in A\}$ and let b be the index $\max[d]\{x_t[d] | d \in B\}$. Note that we have $x_t[a] > x_t[b]$.

We proceed by case analysis with one case for when $skew(x_t[d])$ is convex and another case for when it is concave. For reasons that will become clear in a moment, however, we will tweak these cases to consider when $x_t[b] \leq 2/G$ and $x_t[b] > 2/G$. Recall that by our initial conditions, $x_t[a] > 2/G$.

Case 1: $x_t[b] \leq 2/G$

Consider when $x_t[b] \leq 1/G$ such that $skew(x_t[b])$ is concave. For any $d \in B$, we can bound the magnitude of its decrease in skew using convexity bounds from lemma ??:

$$\begin{aligned} |skew(x_{t+1}[d]) - skew(x_t[d])| &= |(x_t[d] + \delta z_d - 1/G)^3 - (x_t[d] - 1/G)^3| \\ &\leq |\delta z_d skew'(x_t[d] + \delta z_d - 1/G)| \\ &\leq |\delta z_d skew'(-1/G)| \end{aligned} \tag{14}$$

Then,

$$\begin{aligned} |\sum_{d \in B} skew(x_{t+1}[d]) - skew(x_t[d])| &\leq |\sum_{d \in B} \delta z_d skew'(-1/G)| \\ &\leq \delta skew'(-1/G) \end{aligned} \tag{15}$$

Now, consider an element $1/G < x_t[d] \leq 2/G$ that is decreasing; that is, $d \in B$. In this regime, skew is a convex function. To bound its decrease in skew, we have:

$$\begin{aligned} |skew(x_{t+1}[d]) - skew(x_t[d])| &= |(x_t[d] + \delta z_d - 1/G)^3 - (x_t[d] - 1/G)^3| \\ &\leq |\delta z_d skew'(x_t[d] - 1/G)| \\ &\leq |\delta z_d skew'(2/D - 1/G)| \\ &\leq |\delta z_d skew'(1/G)| \end{aligned} \tag{16}$$

Since we have $|\delta z_d skew'(1/G)| = |\delta z_d skew'(-1/G)|$, we have the bound:

$$|\sum_{d \in B} skew(x_{t+1}[d]) - skew(x_t[d])| \leq \delta skew'(1/G) \tag{17}$$

for all $d \in B$ when $x_t[b] \leq 2/G$.

Now, we show that a lower bound on the increase in skew from X is greater than $\delta skew'(1/G)$. For any $d \in A$ such that $x_t[d] > 2/G$, we have:

$$\begin{aligned} |skew(x_{t+1}[d]) - skew(x_t[d])| &= |(x_t[d] + \delta z_d - 1/G)^3 - (x_t[d] - 1/G)^3| \\ &> |\delta z_d skew'(x_t[d] - 1/G)| \\ &> |\delta z_d skew'(2/D - 1/G)| \\ &> |\delta z_d skew'(1/G)| \end{aligned} \tag{18}$$

Thus:

$$|\sum_{d \in B} skew(x_{t+1}[d]) - skew(x_t[d])| < |\sum_{d \in A} skew(x_{t+1}[d]) - skew(x_t[d])| \tag{19}$$

Case 2: $x_t[b] > 2/G$

By the same argument as above, we can bound the decrease in skew for any $d \in B$ where $x_t[d] < x_t[b]$ as:

$$\begin{aligned} |skew(x_{t+1}[d]) - skew(x_t[d])| &= |(x_t[d] + \delta z_d - 1/G)^3 - (x_t[d] - 1/G)^3| \\ &\leq |\delta z_d skew'(x_t[d] - 1/G)| \\ &\leq |\delta z_d skew'(x_t[b] - 1/G)| \end{aligned} \tag{20}$$

which is greater than $|\delta z_d skew'(-1/G)|$, so the bound works for all $x_t[d]$ between 0 and $x_t[b]$.

Similarly, we lower bound the increase in skew for any $d \in A$ where $x_t[d] > x_t[a]$ as:

$$|skew(x_{t+1}[d]) - skew(x_t[d])| \geq |\delta z_d skew'(x_t[a] - 1/G)| \tag{21}$$

Applying the bound across all of A and B eliminates z_d, so we conclude:

$$|\delta skew'(x_t[b] - 1/G)| \leq |\delta skew'(x_t[a] - 1/G)| \tag{22}$$

which is true by our initial conditions where $x_t[a] > x_t[b]$ and $x_t[a] > 2/G$.

Thus, the magnitude of the increase in skew is greater than the magnitude of the decrease in skew from one step of natural selection, so

$$skew(\boldsymbol{x_{t+1}}) > skew(\boldsymbol{x_t}) \tag{23}$$

Lemma 3 (The highest fitness full-length genotype is identifiable from lossy measurements) *When* $w[2] < \boldsymbol{w}^T\boldsymbol{x_t}$ *or equivalently* $x_t[1] > \frac{w[2]}{w[1]}$, *then* g_1 *(the full-length genotype with the highest fitness) is identifiable from two timepoints from* y *alone.*

Remarks. The condition $w[2] < \boldsymbol{w}^T\boldsymbol{x_t}$ expresses that $x_{t+1}[i] - x_t[i] > 0$ for $i = 1$ only. This condition is satisfied when $x_t[1] > \frac{w[2]}{w[1]}$ by expressing $x_t[1] = \frac{w[2]}{w[1]} + k$ for $k > 0$ and noting:

$$\begin{aligned} w[2] &\leq \boldsymbol{w}^T\boldsymbol{x_t} \\ w[2] &\leq w[1]x_t[1] + c \\ w[2] &\leq w[1](\tfrac{w[2]}{w[1]} + k) + c \\ w[2] &\leq w[2] + w[1]k + c \end{aligned} \tag{24}$$

is true. Note that $c > 0$ because $w[d] > 0$ and $x[d] \geq 0$ for all d.

Proof.

We show that when $w[2] < \boldsymbol{w}^T\boldsymbol{x_t}$, we have that $y_{t+1}[m] - y_t[m]$ is positive if and only if the genotype element 1 is in $\boldsymbol{B}_m$. First, consider:

$$y_{t+1}[m] - y_t[m] = \textstyle\sum_{d \in B_m} x_{t+1}[d] - x_t[d] = \sum_{d \in B_m} \frac{w[d] - \boldsymbol{w}^T\boldsymbol{x_t}}{w[d]} \tag{25}$$

Since $\boldsymbol{w}^T\boldsymbol{x_t} > w[2] > w[3] > \ldots > w[d] > 0$, we have $x_{t+1}[d] - x_t[d] \leq 0$ for $d \in \{2, \ldots, D\}$ and $x_{t+1}[1] - x_t[1] > 0$ for only the highest fitness genotype g_1.

We will use δ to refer to $x_{t+1}[1] - x_t[1]$ and note that $\delta > 0$. Since $\sum_d x_t[d] = 1$ for all t,

$$\textstyle\sum_{d \neq 1} x_{t+1}[d] - x_t[d] = \sum_{d \neq 1} \frac{w[d] - \boldsymbol{w}^T\boldsymbol{x_t}}{w[d]} = -\delta \tag{26}$$

where each term $\frac{w[d]-\boldsymbol{w}^T\boldsymbol{x_t}}{w[d]} < 0$.

Since $\boldsymbol{B}_m \subset \{1,2,\ldots,D\}$ is a strict subset; that is, $|\boldsymbol{B}_m| < G$ by construction, we have for each m:

$$0 > \sum_{\substack{d\neq 1,\\ d\in \boldsymbol{B}_m}} x_{t+1}[d] - x_t[d] = \sum_{\substack{d\neq 1,\\ d\in \boldsymbol{B}_m}} \frac{w[d]-\boldsymbol{w}^T\boldsymbol{x_t}}{w[d]} > -\delta \tag{27}$$

For m where $1 \in \boldsymbol{B}_m$, we then have:

$$y_{t+1}[m] - y_t[m] = \delta + \sum_{\substack{d\neq 1,\\ d\in B_m}} x_{t+1}[d] - x_t[d] > 0 \tag{28}$$

And for m where $1 \notin \boldsymbol{B}_m$, we have:

$$y_{t+1}[m] - y_t[m] = 0 + \sum_{\substack{d\neq 1,\\ d\in B_m}} x_{t+1}[d] - x_t[d] < 0 \tag{29}$$

Thus, $y_{t+1}[m] - y_t[m] > 0$ if and only if $1 \in \boldsymbol{B}_m$ for all events m.

Lemma 4. *Given $\boldsymbol{X}$ with at least two timepoints, we can infer a fitness vector $\boldsymbol{w}$ that is equal to the true fitness vector up to a scaling factor.*

Remarks. In our inference problem, we generally assume that $\boldsymbol{X}$ is hidden and that we aim to learn it from $\boldsymbol{Y}$. This theorem establishes that the function $\boldsymbol{w}, \boldsymbol{x_0} \rightarrow \boldsymbol{X}$ is injective, and thus the key source of non-identifiability in the general inference problem is from the lossy measurements through $\boldsymbol{B}$. The general task of recovering $\boldsymbol{X}$ from $\boldsymbol{Y}$ has $O(GT)$ degrees of freedom. Under the hard constraint of requiring $\boldsymbol{X}$ to be consistent with trajectories from natural selection, however, the degrees of freedom is reduced to $O(G)$. Thus, while the problem remains non-identifiable in the general case, the selection regularizer adds structure to the problem and eliminates spurious solutions.

Our method also uses the shortest distance between sampling times as the generation time for the natural selection recurrence relation. These "generation" times may be as long as 24 h, which may appear at first glance to be a modeling error when generation times in directed evolution are known to be much faster. This theorem shows that "misspecifying" generation time does not actually cause any issues when modeling a population under natural selection. In particular, we show that $\boldsymbol{w}$ learned from any generation time yields exactly identical trajectories as $\boldsymbol{w}$ calculated from the true generation time up to a rescaling of the time axis.

Proof. Without loss of generality, let the true generation time be 1, such that $p(1)$ occurs one generation after the initial $\boldsymbol{x_0}$, and let the sampling time be an integer $\tau > 1$. Let $\boldsymbol{\psi}$ be the fitness vector that we will calculate from $\boldsymbol{x_\tau}$ and $\boldsymbol{x_0}$ and compare to the true fitness vector $\boldsymbol{w}$.

$$\begin{aligned} \frac{x_T[d]}{x_0[d]} = \frac{\prod_{t=0}^{\tau-1}(\frac{w[d]}{\boldsymbol{w}^T\boldsymbol{x_t}})x_0[d]}{x_0[d]} &= \frac{\psi[d]}{\boldsymbol{\psi}^T\boldsymbol{x_0}} \\ \prod_{t=0}^{\tau-1}(\frac{w[d]}{\boldsymbol{w}^T\boldsymbol{x_t}}) &= \frac{\psi[d]}{\boldsymbol{\psi}^T\boldsymbol{x_0}} \end{aligned} \tag{30}$$

Consider $\boldsymbol{w}^T\boldsymbol{x_t}$ and recall that $x_{t+1}[d] = \frac{w[d]}{\boldsymbol{w}^T\boldsymbol{x_t}}x_t[d]$. Thus:

$$\begin{aligned} \boldsymbol{w}^T\boldsymbol{x_t} &= \sum_d w[d]x_t[d] \\ &= \sum_d w[d]\frac{w[d]x_{t-1}[d]}{\boldsymbol{w}^T\boldsymbol{x_{t-1}}} \\ &= \frac{1}{\boldsymbol{w}^T\boldsymbol{x_{t-1}}}\sum_d w[d]^2x_{t-1}[d] \end{aligned} \tag{31}$$

By recursing t from $\tau - 1$ to 0, we get:

$$\boldsymbol{w}^T\boldsymbol{x_t} = \prod_{t=0}^{\tau-2}\frac{1}{\boldsymbol{w}^T\boldsymbol{x_t}}\sum_d w[d]^Tx_0[d] \tag{32}$$

Using this in equation 30:

$$\begin{aligned} \prod_{t=0}^{T-1}(\frac{w[d]}{\boldsymbol{w}^T\boldsymbol{x_t}}) &= \frac{\psi[d]}{\boldsymbol{\psi}^T\boldsymbol{x_0}} \\ \frac{w[d]^T}{(\prod_{t=0}^{T-2}\boldsymbol{w}^T\boldsymbol{x_t})(\prod_{t=0}^{T-2}\frac{1}{\boldsymbol{w}^T\boldsymbol{x_t}})\sum_d w[d]^Tx_0[d]} &= \frac{\psi[d]}{\boldsymbol{\psi}^T\boldsymbol{x_0}} \\ \frac{w[d]^T}{\sum_d w[d]^Tx_0[d]} &= \frac{\psi[d]}{\boldsymbol{\psi}^T\boldsymbol{x_0}} \end{aligned} \tag{33}$$

We have this equation for each d. The solution to the system of equations is thus $\psi[d] = w[d]^\tau$. If we know τ, we can recover $\boldsymbol{w}$ up to a constant scaling factor. Even if we don't know τ, however, the natural selection trajectories using ψ will be the same as using $\boldsymbol{w}$ on an appropriately scaled time axis, and we can obtain trajectories to arbitrary time resolution by increasing τ.

Lemma 5 (Identifiability from observations in a simple example given $\boldsymbol{x_0}$) *Consider two independent mutation events ($M = 2$) and a population of two single mutants (*10 *and* 01*) and a*

*double mutant (*11*), where we can only observe single mutation frequencies. If genotype frequencies are known at time* $t-1$ *as* $\boldsymbol{x_{t-1}}$*, then fitness* $\boldsymbol{w}$ *is identifiable from lossy measurements* $\boldsymbol{y_t}$ *and* $\boldsymbol{t_{t+1}}$*.*

Remarks. This theorem establishes that full-length genotype trajectories (equivalent to, and computable from, $\boldsymbol{w}$) can be recovered exactly from lossy measurements of only position-wise mutation frequencies representing mixtures of genotypes, when $\boldsymbol{x_0}$ is known. In practice, $\boldsymbol{x_0}$ cannot easily be measured. This theorem, however, establishes that lossy measurements do not completely prohibit identifiability.

Proof. Denote $\boldsymbol{x_t^*}$ as the full-length genotype frequency vector at time t following $\boldsymbol{w}^*$, and $\boldsymbol{x_t}$ as following $\boldsymbol{w}$. The same notation shall be used for $y_t^*[m]$ and $y_t[m]$.

Since $\boldsymbol{x_{t-1}^* = x_{t-1}}$, we also have $y_{t-1}^*[m] = y_{t-1}[m]$ for all m. It is possible for observed marginals to be identical at another timepoint; without loss of generality, denote this time interval as 1 so we have $y_t^*[m] = y_t[m]$ for all m. Recall that by 4, we know that if $\boldsymbol{w}^* \neq \boldsymbol{w}$, then $\boldsymbol{x_t^* = x_t}$ is true for at most 1 value of t. Thus, if $y_t^*[m] = y_t[m]$, we can express:

$$x_t[d] = x_t^*[d] + \delta_d \tag{34}$$

where $\sum_{d \in \boldsymbol{B}_m} \delta_d = 0$ for each m and $x_t[d] \neq x_t^*[d]$; that is, δ is not the zero vector ($\delta \neq 0$).

We will derive $\boldsymbol{x_{t+1}^*}$ and $\boldsymbol{x_{t+1}}$ from fitnesses calculated from $x_t[d] \neq x_t^*[d]$ relative to $x_{t-1}^*[d] = x_{t-1}[d]$. From there, we will obtain a system of equations $y_{t+1}^*[m] = y_{t+1}[m]$ for each m as a function of free variables δ_d and show by a computer-assisted proof that the only solution to the system of equations is $\delta = 0$. Thus will allow us to conclude that:

$$\begin{cases} \boldsymbol{x_{t-1}^* = x_{t-1}}, \\ \forall m, y_t^*[m] = y_t[m] \\ \forall m, y_{t+1}^*[m] = y_{t+1}[m] \end{cases} \tag{35}$$

holds only when $\boldsymbol{w}^* = \boldsymbol{w}$; thus, $\boldsymbol{w}$ is identifiable from these observations.

By definition, we have:

$$\begin{aligned} x_{t+1}^*[d] &= \frac{w^*[d]}{\boldsymbol{w}^{*T}\boldsymbol{x_t}} x_t^*[d] \\ \frac{w^*[d]}{\boldsymbol{w}^{*T}\boldsymbol{x_t}} &= \frac{x_{t+1}^*[d]}{x_t^*[d]} \end{aligned} \tag{36}$$

Since fitness is identical up to scale, without loss of generality, let $\boldsymbol{w^{*T} x^*_{t-1}} = 1$. We use this and the previous equation:

$$
\begin{aligned}
x^*_{t+1}[d] &= \left(\frac{w^*[d]}{w^*[d]\cdot \boldsymbol{x^*_t}}\right)\left(\frac{w^*[d]}{w^*[d]\cdot \boldsymbol{x^*_{t-1}}}\right)x^*_{t-1}[d] \\
x^*_{t+1}[d] &= \left(\frac{\frac{x^*_t[d]}{x^*_{t-1}[d]}}{\sum_i \left(\frac{p^*_i(t)}{p^*_i(t-1)}\right)p^*_i(t)}\right)\left(\frac{x^*_t[d]}{x^*_{t-1}[d]}\right)x^*_{t-1}[d] \\
x^*_{t+1}[d] &= \left(\frac{\frac{x^*_t[d]}{x^*_{t-1}[d]}}{\sum_i \left(\frac{p^*_i(t)}{p^*_i(t-1)}\right)p^*_i(t)}\right)x^*_t[d] \\
x^*_{t+1}[d] &= \left(\frac{\frac{x^*_t[d]^2}{x^*_{t-1}[d]}}{\sum_i \left(\frac{p^*_i(t)^2}{p^*_i(t-1)}\right)}\right)
\end{aligned}
\tag{37}
$$

We express $x_{t+1}[d]$ analogously, using $x_t[d] = x^*_t[d] + \delta_d$:

$$
x_{t+1}[d] = \left(\frac{\frac{(x^*_t[d]+\delta_d)^2}{x^*_{t-1}[d]}}{\sum_i \left(\frac{(p^*_i(t)+\delta_i)^2}{p^*_i(t-1)}\right)}\right) \tag{38}
$$

For a particular marginal m, we have:

$$
\begin{aligned}
0 &= y^*_{t+1}[m] - y_{t+1}[m] \\
0 &= \left(\sum_i \frac{p^*_i(t)^2}{p^*_i(t-1)}\right)\left(\sum_{d\in\mathbb{B}_m} \frac{(x^*_t[d]+\delta_d)^2}{x^*_{t-1}[d]}\right) - \left(\sum_i \frac{(p^*_i(t)+\delta_i)^2}{p^*_i(t-1)}\right)\left(\sum_{d\in\boldsymbol{B}_m} \frac{x^*_t[d]^2}{x^*_{t-1}[d]}\right)
\end{aligned}
\tag{39}
$$

We write out this system of equations $G = 3$ for two single mutants (10 and 01) and one double mutant (11) at two marginal positions ($M = 2$) and solve using a symbolic math Python library:

```
import sympy
from sympy import *

# x, y, z are deltas; x and y are single mutants and z is the double mutant
x, y, z = symbols('x y z')

# pTD; T in [0, 1], D in [0, 1, 2]
p00, p01, p02, p10, p11, p12 = symbols('p00 p01 p02 p10 p11 p12')

# Define marginal observations: mixtures of x+z, and y+z
sum_star_all = p10**2/p00 + p11**2/p01 + p12**2/p02
```

```
sum_star_02 = p10**2/p00 + p12**2/p02
sum_star_12 = p11**2/p01 + p12**2/p02

sum_params_all = (p10+x)**2/p00 + (p11+y)**2/p01 + (p12+z)**2/p02
sum_params_02 = (p10+x)**2/p00 + (p12+z)**2/p02
sum_params_12 = (p11+y)**2/p01 + (p12+z)**2/p02

eq02 = Eq(sum_star_all * sum_params_02 - sum_params_all * sum_star_02, 0)
eq12 = Eq(sum_star_all * sum_params_12 - sum_params_all * sum_star_12, 0)
eq_xz = Eq(x+z, 0)
eq_yz = Eq(y+z, 0)

solve([eq02, eq12, eq_xz, eq_yz], [x, y, z])

# Runtime: <1 second
>>> [0, 0, 0]
```

The solution is $\delta = 0$. Thus, $\boldsymbol{w}$ is identifiable under the conditions of the proof.

Lemma 6 (Gradient-matching implicitly encourages smaller $\left|\left|s\right|\right|_1$) *Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be distributions over N discrete elements, and $\boldsymbol{s_1}, \boldsymbol{s_2}$ be vectors over M discrete elements with non-negative values whose sum is at most 1. Then, we have $\| \boldsymbol{s_1} \|_1 < \| \boldsymbol{s_2} \|_1$ if and only if $D_{KL}(\boldsymbol{p} \| (1-\| \boldsymbol{s_1} \|_1)\boldsymbol{q} + \boldsymbol{s_1}) < D_{KL}(\boldsymbol{p} \| (1-\| \boldsymbol{s_2} \|_1)\boldsymbol{q} + \boldsymbol{s_2})$ over $N + M$ elements.*

Proof. Note that $D_{KL}(\boldsymbol{p} \| \boldsymbol{q}) = H(\boldsymbol{p}) - \sum_x p(x)\log q(x)$, where $H(\boldsymbol{p})$ is the non-negative entropy of $\boldsymbol{p}$ and independent of $\boldsymbol{q}$. Using the monotonicity of $\log(x)$, we have that scalars $y < z$ if and only if $\log(y) < \log(z)$ if and only if $H(\boldsymbol{p}) - \sum_x p(x)\log(y) > H(\boldsymbol{p}) - \sum_x p(x)\log(z)$.

(=>) Suppose that $\| \boldsymbol{s_1} \|_1 < \| \boldsymbol{s_2} \|_1$. Then, $(1-\| \boldsymbol{s_1} \|_1)q(x) > (1-\| \boldsymbol{s_2} \|_1)q(x)$ for all x. Then, by our observations above, $D_{KL}(\boldsymbol{p} \| (1-\| \boldsymbol{s_1} \|_1)\boldsymbol{q} + \boldsymbol{s_1}) < D_{KL}(\boldsymbol{p} \| (1-\| \boldsymbol{s_2} \|_1)\boldsymbol{q} + \boldsymbol{s_2})$.

(<=) Now, suppose that $D_{KL}(\boldsymbol{p} \| (1-\| \boldsymbol{s_1} \|_1)\boldsymbol{q} + \boldsymbol{s_1}) < D_{KL}(\boldsymbol{p} \| (1-\| \boldsymbol{s_2} \|_1)\boldsymbol{q} + \boldsymbol{s_2})$. As $p(x) = 0$ for all x in the support of $\boldsymbol{s_1}, \boldsymbol{s_2}$, this reduces to $D_{KL}(\boldsymbol{p} \| (1-\| \boldsymbol{s_1} \|_1)\boldsymbol{q}) < D_{KL}(\boldsymbol{p} \| (1-\| \boldsymbol{s_2} \|_1)\boldsymbol{q})$. By our previous observations, this implies that $\| \boldsymbol{s_1} \|_1 < \| \boldsymbol{s_2} \|_1$.

Remarks. This lemma maps to our problem by considering the N discrete elements to be present genotypes and the M discrete elements to be absent genotypes at some time t. The result states that the higher the total frequency of genotypes introduced to the population at some time t, the larger the D_{KL} loss term is, no matter what $\boldsymbol{p}$ or $\boldsymbol{q}$ are.

Lemma 7 (Even if genotypes can enter the population multiple times, genotype frequencies under natural selection can rise to appreciable frequency at most once) *Let $\boldsymbol{x_t}$ be states evolving under a state transition process $\boldsymbol{x_{t+1}} = (1-\| \boldsymbol{p_t} \|_1)(\frac{\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{x_t}}) \odot \boldsymbol{x_t} + \boldsymbol{p_t}$ where $\boldsymbol{p_t}$ is a vector whose i-th element is non-negative $s_t[i]$ if $z_t[i] = t$ otherwise 0, with parameters $\boldsymbol{w}, \boldsymbol{s_t}, \boldsymbol{z_t}$, such that $z_t[i] \neq t$ for all i where $x_t[i] > 0$ (genotypes can only be introduced when they were at 0 frequency). Let there be a threshold $\epsilon < \lambda$ such that if $x_t[i] \leq \epsilon$, then $x_{t+1}[i] = 0$. Let $0 < \lambda << 1$ be an upper bound such that for any genotype i, if $x_t[i] = 0$, then $x_{t+1}[i] < \lambda$ for any t. If $\boldsymbol{w}^T\boldsymbol{x_t}$ increases monotonically over time and there exists a time τ such that for some genotype i, for all $t < \tau$, the rate of change according to fitness-based natural selection $\frac{w[i]-\boldsymbol{w}^T\boldsymbol{x_t}}{\boldsymbol{w}^T\boldsymbol{x_t}} > 0$, and for all $t' > \tau$, $\frac{w[i]-\boldsymbol{w}^T\boldsymbol{x_t}}{\boldsymbol{w}^T\boldsymbol{x_t}} < 0$, then genotype i cannot have a local maxima above λ at any time $t' > \tau$.*

Remarks. The state transition process used here is closely related to Evoracle's, with the key difference that $\boldsymbol{s}$ and $\boldsymbol{z}$ now depend on time, allowing genotypes to enter the population multiple times. The condition that $\boldsymbol{w}^T\boldsymbol{x_t}$ increases monotonically over time is satisfied by a well-known theoretical result in the literature when genotypes do not enter or leave the population under our deterministic asexual natural selection model. Here, however, genotypes entering the population with high frequency and low fitness could lower the mean fitness over time. The condition can be still be satisfied if λ is low enough such that new genotypes enter at very low frequencies, which can be achieved during inference by setting ϵ low.

This state transition process captures the situation during inference, where by directly inferring states, we risk the possibility of inferring multiple entries into the population which is incompatible with Evoracle's assumed model. The lemma's result states that this inference risk is not a concern when present genotype trajectories fit the natural selection model, because a genotype cannot rise above λ frequency more than once. By taking λ to be low, genotypes cannot rise to appreciable frequency in the population more than once.

This lemma also states that even if we changed Evoracle's assumed model to allow for multiple entries, it would not lead to qualitatively different trajectories than a single-entry model.

Proof. Suppose that for some genotype i at some time $\gamma > \tau$, $x_{\gamma-1}[i] > 0$. Then, if $0 < x_{\gamma-1}[i] \leq \epsilon$, $x_\gamma[i] = 0$ by construction. If $x_{\gamma-1}[i] > \epsilon$, then $x_\gamma[i] = (1-\| \boldsymbol{p_{\gamma-1}} \|_1)(\frac{\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{x_{\gamma-1}}}) \odot \boldsymbol{x_{\gamma-1}}$. As $\gamma > \tau$, the rate of change is negative, so $(\frac{\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{x_{\gamma-1}}}) < 1$. Also, $(1-\| \boldsymbol{p_{\gamma-1}} \|_1) \leq 1$ by construction as its elements are non-negative. Thus, $x_\gamma[i] < x_{\gamma-1}[i]$ in all cases.

The only way for $x_\gamma[i]$ to increase is if it is re-introduced to the population, so that $x_{\gamma-1}[i] = 0$ and $x_\gamma[i] > 0$. By the definition of λ, this non-zero introduction frequency is $x_\gamma[i] < \lambda$. However, by the above argument, it must then shrink every following timestep back down to 0. Therefore, any genotype i cannot have a local maxima above λ at any time $t' > \tau$.

Supplementary Note 2 | Utility of unnormalized skew regularization.

We evaluated model performance on Cry1Ac and TadA data while varying hyperparameters (Extended Data Fig. 1) and observed that varying the weight of the skew regularizer, or even removing it completely (with weight set to zero) had no effect on performance. While these results may suggest that the categorical skew regularizer contributes nothing in general, we conservatively conclude that the regularizer has no effect on these two directed evolution datasets, and further discuss here why we consider it an integral part of the model and recommend using it in practice.

Demonstration of utility on synthetic data

In Extended Data Fig. 1, we present results of an experiment with synthetic data where a wild-type genotype and a double mutant undergo natural selection, such that the double mutant slowly sweeps the population. We provided point mutation frequencies for the two mutations to Evoracle and evaluated its ability to reconstruct the held-out ground-truth genotype trajectories. This is non-trivial for the model because it can disentangle the point mutation frequencies which rise in near synchrony into the contributions of two separate variants each with a point mutation, or a single double mutant. In this case, we observe that not using the skew regularizer yields inaccurate reconstructions where the point mutants peak at 4 h at ~15% population frequency. In contrast, using the skew regularizer with a stronger weight yields a reconstruction where the model suppresses the point mutants to essentially 0% frequency throughout the entire time range.

Theoretical motivation

We theoretically prove that natural selection yields high-skew genotype frequencies (see Lemma 2) and observe in our work that directed evolution often empirically generates skewed genotype frequencies in practice. Sparsity is, in general, a powerful property for many modeling applications, and the regularizer more explicitly encourages sparsity.

Encoding prior knowledge of sparsity

In practice, it is common for experimentalists to assess the activity in a directed evolution campaign through not just short-read sequencing data but also using low-throughput Sanger sequencing of individually isolated clones and other low-cost techniques. Such information can be useful for improving the accuracy of reconstruction, but there is no direct way to incorporate such information into Evoracle. There are indirect ways to do so, however—if the pool is known to be particularly sparse or skewed, this prior knowledge can be communicated to Evoracle by tuning the beta weight on the skew regularizer.

Supplementary Note 3 | Practical usage guidelines

Our model is available as a light-weight python package at https://github.com/maxwshen/evo-tft. For details on installing and running the model in python, refer to the write-up in the GitHub repository. Here, we mention some general and practical usage guidelines.

DNA sequencing strategies

To use Evoracle with short-read sequencing data to reconstruct long genotypes, DNA sequencing should be designed to maximize coverage across the full length of the gene. Badran *et al.* performed Illumina sequencing on randomly sheared segments[1]. This approach could risk uneven coverage from non-uniform PCR amplification of segments due to, for instance, uneven GC content, or length. In principle, DNA sequencing could also be performed from sets of designed primers which could improve coverage uniformity. In this case, it is important to ensure that all nucleotides are observed by some pair of primers, so consecutive primer pairs should be designed to overlap each other.

As an example, consider a 3,100-nt gene that will be covered by 300-nt Illumina sequencing reads. This gene could be covered by 3100/300 rounded up = 11 tiled primer pairs. In general, we recommend at least 10,000 reads per nucleotide. This recommendation yields a total sequencing depth of 10,000 reads per nucleotide * 11 tiled primer pairs = 110,000 reads per timepoint.

Frequency of timepoints

In practice for PACE, PANCE, and OrthoRep, we strongly encourage 6-h or 12-h timepoints, and no longer than 24 h between timepoints for directed evolution campaigns with high selection stringency. We note that in general, the best timepoint frequency is impossible to know before the campaign.

As an illustrative example, consider the case where within 24 h, a single genotype rises to dominate the pool, and then depletes as it is taken over by a new genotype. If timepoints are taken 24 h apart before this genotype has risen and after it is depleted, we would see no change. Since Evoracle only uses timepoint sequencing data as input, events like these are invisible to Evoracle. Fast rises and falls are more likely to occur as selection stringency is increased, so finer timepoint resolution is particularly important towards the end of a campaign.

Evoracle's performance benefits from finer timepoint resolution, because by providing more timepoint samples, Evoracle has more data to work with for detecting subtle covariations between mutation frequencies which are the primary signal that Evoracle uses to reconstruct full-length genotypes and their fitnesses.

We anticipate that a common application of Evoracle is proposing high-fitness variants from a directed evolution campaign. This use case will benefit substantially from finer timepoint resolution

near the final timepoint, not only to compensate for higher selection stringencies from common directed evolution campaign design, but also because the variants with the best fitness are present primarily at the final timepoint. The more data Evoracle has near the final timepoint, the better Evoracle can decipher the data and propose high-fitness variants with higher confidence.

We recommend using a timepoint schedule that has a large common denominator, such as 6 h, 12 h, and 24 h. When the number of total timepoints that can be taken is limited, we recommend using coarser timepoint resolution near the beginning of the campaign and finer timepoint resolution near the end of the campaign.

Manual inspection of proposed genotypes

Our model uses a heuristic (see Online Methods) to propose full-length genotypes that could exist in the population given an input table of mutation frequencies in read segments across timepoints (Online Methods). Evoracle attempts to explain observed data given this proposed list of full-length genotypes, so any full-length genotypes not in the proposed list are assumed to not exist in the population. We strongly recommend manually checking the list of proposed full-length genotypes to ensure they are consistent with expectations. It is straightforward to modify the proposed list to incorporate prior domain knowledge, for example by adding full-length genotypes that are expected to exist in the population.

Our work evaluates model performance while varying the proposal strategy (Extended Data Fig. 2). We find that our baseline heuristic of proposing full-length genotypes based on mutations that rise and fall together is strong. Evoracle is robust to proposing too many full-length genotypes. In Extended Data Fig. 2, we explored adding full-length genotypes with combinations of mutations that could exist in the population but lack strong evidence for existing beyond low frequencies. We observed that Evoracle performance does not decrease substantially in the presence of extra full-length genotypes.

Evaluation of model fit

To what extent can Evoracle's inferences be trusted? A crucial step in calibrating trust is evaluating how well the model was able to fit the provided mutation frequency table, also referred to as the marginal observations. This table serves as the primary input the model and is the only component of the optimized loss function that relates to observed data. Do not trust model inferences if the marginal observations fit poorly, which can be evaluated by plotting them against each together or calculating summary statistics such as Pearson correlation across timepoints. If the fit is poor, we recommend tuning hyperparameters. For example, it can be helpful to increase the alpha weight in the loss function. Poor model fit can also occur if the proposed full-length genotypes is incomplete or otherwise incapable of explaining the observed table of mutation frequencies from short read data.

Uncertainty estimates for fitness inferences

It is possible to empirically evaluate the uncertainty for a fitness estimate given observed or predicted full-length genotype frequencies. Each consecutive pair of timepoints in which the full-length genotype is present can be used to obtain a single-point estimate of the genotype's fitness and can be summarized by their variance as a metric of empirical uncertainty.

Expected performance of Evoracle in selective sweep, clonal interference, and neutral drift conditions.

The directed evolution campaigns considered in our work all have high selection stringency, short generation time, and high mutation rates. These properties are characteristic of continuous directed evolution platforms such as PACE[1,2] and OrthoRep[3]. These conditions tend to produce highly skewed populations where just a few variants comprise most of the pool at any given timepoint, as observed empirically, as well as in theory (see **Supplementary Note 1**). We speculate that this skewness property provides a strong statistical signal, enables robust inference, and helps explain why Evoracle generally performs well across these datasets.

Clonal interference describes an evolving population comprised of a diverse mixture of lower frequency genotypes[4]. Clonal interference can occur when mutation rates are high and selection is less stringent, such that it is difficult for any single variant to take over the population before new variants with higher fitness arise by mutation. We speculate that Evoracle would perform worse in conditions like these, since the marginal covariation between allele frequencies across all evolving genotypes would sum over a larger number of single genotypes. In these conditions, we think that increasing the density of timepoints can improve performance.

Neutral drift is often used to diversify a population before directed evolution and can be induced by applying high mutation rates and little to no selection. Evoracle's core mathematical model assumes that genotype frequencies change according to their fitness, but fitness is ill-defined or non-existent when there is no competition. We do not expect Evoracle to be useful on populations that only undergo neutral drift. For directed evolution campaigns that include a neutral drift warmup phase followed by high selection stringency, we recommend running Evoracle only on timepoints from the high selection stringency regime.

Supplementary Note 4 | Comparisons to related approaches

Comparison of Evoracle's time complexity to a standard expectation maximization approach

One common approach to solve inference problems in latent variable models is expectation maximization (EM). Here, we describe an EM approach to our inference problem, and show that Evoracle achieves improved asymptotic time complexity, while requiring fewer explicit assumptions from the user, than the EM approach.

The inference task is: Given $\boldsymbol{y_1, y_2, \ldots, y_T}$, infer the parameters $(\boldsymbol{w, s, z})$ and the unobserved genotype frequency states $\boldsymbol{x_1, x_2, \ldots, x_T}$. Let $\boldsymbol{Y} = [\boldsymbol{y_1, y_2, \ldots, y_T}]$ be a matrix and define $\boldsymbol{X}$ analogously. EM infers the MLE estimates of $\boldsymbol{X}$, $(\boldsymbol{w, s, z})$ given $\boldsymbol{Y}$ under specified conditional probabilities $p(\boldsymbol{Y}|\boldsymbol{X})$, which can be understood as an observational noise model, and $p(\boldsymbol{X}|\boldsymbol{s, z, w})$, which would compare a genotype frequency to the expected frequency given its spawning time, initial frequency, and fitness.

The EM algorithm would proceed in two steps:

1. Update $\boldsymbol{X}$ to maximize $p(\boldsymbol{Y, X}|\boldsymbol{s, z, w}) = p(\boldsymbol{Y}|\boldsymbol{X})p(\boldsymbol{X}|\boldsymbol{s, z, w})$ given fixed $\boldsymbol{Y}, (\boldsymbol{s, z, w})$

   Evoracle essentially only performs this step, but without relying on known fixed $(\boldsymbol{s, z})$, which are instead encoded in $\boldsymbol{X}$.

2. Update $(\boldsymbol{s, z, w})$ to maximize $p(\boldsymbol{Y, X}|\boldsymbol{s, z, w}) = p(\boldsymbol{Y}|\boldsymbol{X})p(\boldsymbol{X}|\boldsymbol{s, z, w})$ given fixed $\boldsymbol{X, Y}$

   Here, we proceed by updating each individual parameter while holding the other parameters fixed.
   - $\boldsymbol{w}$ is updated based on empirical fitness based on when genotypes are present according to a fixed $\boldsymbol{z}$.
   - $\boldsymbol{s}$ is updated towards the initial genotype frequencies in $\boldsymbol{X}$ based on a fixed $\boldsymbol{z}$
   - $\boldsymbol{z}$ is updated using fixed $\boldsymbol{w, s, X}$. It is discrete, so we check all possible values. We check $O(GT)$ values for the whole dataset, and each likelihood evaluation takes $O(T)$ time, for an overall complexity of $O(GT^2)$

By avoiding an explicit representation of discrete $\boldsymbol{z}$, Evoracle avoids an $O(GT^2)$ term every optimization step. This gives Evoracle an improved time complexity of $O(GT)$, which is the minimum time required to update $\boldsymbol{X}$. While EM alternates between updating $\boldsymbol{X}$ and $\boldsymbol{w}$, Evoracle optimizes them simultaneously, which we find to work well in practice.

Comparisons to other DNA sequencing strategies.

Sanger sequencing with Surveyor deconvolution costs $10/timepoint. To evaluate the price per timepoint that can be achieved with high-throughput DNA sequencing by pooling timepoints into the same run, consider as an example a 2x150-bp Illumina MiSeq run may cost ~$1,000 and yield 40 million reads for a 1,000-nt gene that can be sequenced in full with four sets of primers that each yield up to 300-nt read segments. We thus have an effective 10 million reads to split among however many timepoints we would like to pool. In an extreme case, it can be sufficient to obtain 5,000 reads per timepoint, which would enable 2,000 timepoints to be sequenced in one run for $0.50 per timepoint. Of course, it is unusual to sequence thousands of timepoints. While it is possible for the cost per timepoint of high-throughput sequencing to be competitive with Sanger sequencing, Sanger sequencing remains a cheaper and more accessible option for many labs in practice.

An interesting aspect for comparison is time. Cost-effective high-throughput sequencing with a single run requires all samples to be collected before sequencing or requires coordination with a sequencing facility to sequence collected samples as a small component of other planned runs. In contrast, Sanger sequencing can be performed directly on a single sample once it has been collected for $10, and results are typically returned within a day. This workflow raises the possibility of near real-time monitoring of genetic dynamics during directed evolution campaigns. As Surveyor deconvolution and Evoracle reconstruction are computer programs that run within minutes, it is plausible to imagine a directed evolution experiment using these tools where the experimenter can make decisions regarding selection stringency, selection circuit, flow rate, or other parameters with near real-time reconstructions of full-length genotype trajectories and fitness in the population.

Methods comparison to BHap and SGML

We briefly discuss our choice of methods to compare against. While many methods have been developed for haplotype reconstruction[5–10], the vast majority require extensive overlaps between reads[8], or additional input such as long-read[10] or gene expression data[9]. Methods using only short-read data with incomplete linkage are rare, and can be categorized into model-free and model-based methods. Among model-free methods, BHap[7] and EVORhA[11] are dominant and conceptually similar, but BHap has been shown to outperform EVORhA, so we chose to compare Evoracle to BHap. As Evoracle is a model-based method, however, we reserve most of our focus on comparing to other model-based methods. In the literature, model-based methods for this problem have been presented by Illingworth, Sobel-Leonard, and co-workers[6,12], which are conceptually similar to Evoracle, but with some key differences in the genotype proposal strategy and the mutation model.

Bacterial haplotype reconstruction (BHap)

We describe BHap[7] using Evoracle's notation, though since BHap ignores time information, we drop all time-related notation. BHap's methodology is motivated by the idea that if a genotype i is present with frequency $x[i]$, and is observed through the same lossy measurement process $E[\boldsymbol{y}] = \boldsymbol{Bx}$ used by Evoracle, then genotype i may induce observed mutations $y[j]$ with frequency similar to $x[i]$ for all j where $B[i][j] = 1$. However, critically, this may only be true if $x[i]$ is the lowest frequency genotype in the population, otherwise, the observed frequency of some of its mutations may include other genotypes.

BHap uses a Poisson noise model for the observational process, so $y[j]$ are read counts. BHap's algorithm begins with expectation maximization to cluster mutation frequencies (or read counts) $y[j]$ for all mutations j using a mixture of G Poisson distributions with parameters $\lambda_1, \ldots, \lambda_G$. At each optimization step, the affinity $\alpha[i][j]$ between each mutation $y[j]$ and each Poisson cluster λ_i is

calculated, then each Poisson cluster parameter λ_i is updated to be closer to the "center" of the mutation data $y[j]$ as weighted by their affinities $\alpha[i][j]$.

After EM converges, BHap attempts to propose a genotype using λ_{min} by finding the minimum cost path through a graph where nodes are mutations at specific positions, and edges connect all mutations in consecutive positions. A path represents a genotype that includes an allele at every position, and its cost is the sum over node costs. The node cost for a mutation j is $1 - max_j Poisson\big(y[j] - \lambda_{min}; \lambda_j\big)$. A node has low cost if its observed read count $y[j]$ is explained well by a simple mixture of two clusters, λ_j and λ_{min}. After this step, BHap subtracts λ_{min} from all observed read counts $y[j]$, then iterates starting from the EM step. BHap terminates when the graph becomes disconnected or when a path cost exceeds a certain threshold.

We chose to compare Evoracle to BHap because it is conceptually representative while comparing favorably to other model-free methods[7,11].

We note that BHap[7], EVORhA[11], and related model-free methods have poor theoretical guarantees, largely because reconstruction from lossy information cannot have any performance guarantees in general without additional data or structure provided by a model. Specifically, BHap and EVORhA theoretically and empirically cannot reconstruct simple mixtures of genotypes present in equal proportion.

BHap was evaluated on small synthetic mixtures of less than 10 genotypes. In contrast, evolutionary datasets often feature hundreds of relevant genotypes, where a large mixture of genotypes have low but comparable frequency. A critical aspect of BHap's algorithm is the assumption that a genotype can be reconstructed from all mutations around the lowest observed frequency. When a large mixture of genotypes have low but comparable frequency, BHap is likely to perform poorly.

The BHap code package from *Li et al., 2019*[7] takes as input a dataset of paired-end reads and a reference genome. First, BHap aligns reads to the reference genome and identifies distinct linkage groups. To perform a fair and focused comparison, we ignored this alignment part of BHap and instead isolated the most relevant logic of BHap's algorithm. We converted the Python 2.7 code related to the EM algorithm and genotype proposal algorithm to Python 3.7 and configured it to run on the same input data used by Evoracle (frequencies of mutations in distinct linkage groups). We encountered errors in the EM step from the default initialization strategy in BHap, as some mutation frequencies were too far from any cluster at initialization. To solve this, we modified the initialization strategy such that all observed mutation read counts were within three standard deviations of some initial Poisson distribution, which improved the stability of BHap and enabled successful execution of the algorithm.

Single-gene multi-locus model (SGML)

At a high-level, SGML has many similarities to Evoracle alongside some crucial differences. We use the most recently published description of SGML in *Sobel-Leonard et al., 2017*[12] which builds on *Illingworth, et al. 2015*[6]. Where descriptions of various aspects of SGML differed, we use the most up-to-date description of that aspect. SGML's high-level algorithm is similar to Evoracle: first, genotypes that could exist in the population are proposed using short-read data. Then, fitness and genotype frequencies are inferred from short-read data under a model of natural selection and mutation by maximum likelihood estimation using gradient descent. The key differences are 1) SGML's maximal genotype proposal strategy, which does not scalable to real datasets with many linkage groups unlike Evoracle's more minimalistic genotype proposal strategy, and 2) unlike Evoracle which assumes a flexible mutation model where any genotype can enter the population at any time, SGML only introduces new genotypes that are a single mutation away from some present genotype. When we compared SGML to Evoracle on real datasets, factor 1 caused SGML to encounter out-of-memory errors. When we ran SGML using genotypes proposed by Evoracle, we found that Evoracle substantially outperforms SGML on our directed evolution datasets due to factor 2 (**Fig. 6A, Extended Data Fig. 7**).

The papers describing SGML only provide an archival record of code that was used for experiments in their papers, and do not provide a public-facing code package that is convenient to use by other researchers[6,12]. For this reason, we implemented our own version of SGML based on reported mathematical descriptions of the method[6,12–14], which we describe here.

The most recent version of SGML uses a maximal genotype proposal strategy[12]: instead of minimizing the number of genotypes required to explain the data, SGML constructs the full set of possible genotypes that could exist in the population, with no concern for the frequencies at which those genotypes may or may nor exist in the population. While this is guaranteed to be sufficient to explain the data, in the case of pooled Sanger sequencing with complete loss of linkage between alleles, the number of proposed genotypes is exponential in the number of alleles considered. At runtime, the maximal genotype proposal strategy resulted in excessive time and memory requirements. For the Cry1ac dataset, SGML proposed $2^{19} \approx 500{,}000$ genotypes which caused the optimization algorithm to fail on our server with 16 GB RAM. We note that SGML requires a mutation matrix describing the 1-mutation neighbors of every genotype; implemented naively, this matrix for 500,00 genotypes requires 250 GB of RAM. We used a sparse matrix implementation, but this still surpassed our server's RAM. SGML proposed $2^{27} \approx 100$ million genotypes for the TadA dataset. While the maximal genotype proposal strategy does not scale well, SGML has other features that we sought to compare to Evoracle. To enable a more fair comparison, we ran SGML using the same set of genotypes proposed by Evoracle, combined with a random subset of all possible genotypes containing 100x the number of genotypes proposed by Evoracle (around 5,000 to 20,000 genotypes) in order to be consistent with SGML's maximal design philosophy. This workaround helps focus our

comparison on differences in modeling approaches. We note that we ran Evoracle using a similar maximal set of genotypes comprising 100x the standard number of genotypes and found robust performance (**Extended Data Fig. 2**) that outperforms SGML.

SGML uses the following state transition process of mutation and selection[6]. Using the same notation as Evoracle: at each time t, genotype frequencies $\boldsymbol{x_t}$ first undergo the same standard asexual natural selection model used by Evoracle: $\frac{\boldsymbol{w}}{\boldsymbol{w^T x_t}} \odot \boldsymbol{x_t}$. Then, genotypes undergo mutation as $x_{t+1}[a] = x_t[a] + \mu \sum_b (x_t[b] - x_t[a])$, where the mutation rate μ = 10^{-5}, and the sum is conducted over genotypes b that differ from some genotype a by a single nucleotide or residue. While this mutation model enables inferring the introduction of new genotypes to the population by mutation, it is restrictive as new genotypes are constrained to enter the population at low frequencies that depend on their single-mutant neighbors. In contrast, Evoracle uses a flexible mutation model that places no constraints on when any genotype may enter the population, which is advantageous when data fitting and genotype reconstruction are considered more important than recovering evolutionary lineages under a specific mutation model. For directed evolution applications, fitness inference and genotype reconstruction can lead to the design of higher-fitness genotypes, which is typically of primary interest, while evolutionary lineages are typically of lesser interest. When evolutionary data features large jumps in sequence space, SGML's restrictive mutation model can yield poor data fit, in contrast to Evoracle's flexible mutation model.

SGML infers fitness and initial frequency parameters for each proposed genotype using maximum likelihood estimation. Observed allele frequencies are modeled as generated by a Dirichlet-multinomial likelihood function from genotype frequencies, which is motivated as an overdispersed version of a multinomial likelihood. SGML uses a conservative estimate of noise. We implemented the Dirichlet-multinomial likelihood using the univariate beta-binomial likelihood to compare individual predicted marginal frequencies to observations. Overdispersion increases variance relative to an observed sample size. We take this process to the limit of the smallest possible sample size of 1 for maximum variance, resulting in a Bernoulli likelihood, noting that this does not change the MLE solution. We implemented SGML in PyTorch[15] and used gradient descent for maximum likelihood estimation.

We conclude this section by reviewing the differences between our implementation of SGML and its original description. Since we primarily sought to compare SGML and Evoracle on genotype reconstruction, we did not implement the meta-modeling approach for inferring alleles under selection and epistatic interactions, which used BIC to select from a hierarchical set of models a model of fitness which contained varying single-loci and multi-loci fitness terms that summed to determine the fitness of a genotype. We note that by allowing unconstrained inference of any fitness value for each genotype, our implementation of SGML should fit the data better than the original description, and

enables a more fair comparison to Evoracle which also infers fitness without constraints on which alleles are under selection or in epistatic interactions.

Comparison to clonal Sanger sequencing

Full-length genotype frequency trajectories can be measured with multiple clonal Sanger sequencing samples from timepoints, and rising stars can also be proposed using such data. We compared this approach to Evoracle by simulating clonal Sanger sequencing data at various depths in the Cry1Ac and TadA datasets by random subsampling of full-gene data at each timepoint.

The consistency between sampled and observed full-length genotype frequencies across timepoints increased with sequencing depth, as expected (**Extended Data Fig. 7**). In the Cry1Ac dataset, Evoracle's performance on a single pooled Sanger sequencing sample (no linkage, 3% noise) with $R^2 = 0.90$ was matched by 5-8x clonal sequencing depth. In the TadA dataset, Evoracle's performance with $R^2 = 0.80$ was matched with 15-20x clonal sequencing depth. Collectively, these results suggest that clonal Sanger sequencing requires roughly an order of magnitude more depth than Evoracle to achieve similar performance in reconstructing full-length genotype frequency trajectories. We note that this represents an approximate cost of $50-200 per timepoint. For many directed evolution experiments with more than a handful of timepoints, this cost quickly compares unfavorably to Illumina or long-read NGS.

We also compared the clonal Sanger sequencing approach to Evoracle at identifying rising stars. This task is more challenging than reconstructing full-length genotype frequencies, because accurate identification of rising stars requires identifying low-frequency genotypes across multiple timepoints and correctly identifying their trend of increasing in frequency. For Cry1Ac, the clonal Sanger approach matched Evoracle's performance with 15x depth when evaluating accuracy and number of true rising stars found. For TadA, the clonal Sanger approach was unable to match Evoracle's performance with coverage up to 30x. We reason that this is because many true rising stars in the TadA dataset never rise above 10% frequency, and have very gently positive slopes.

Taken together, these results suggest that clonal Sanger sequencing is not competitive with Evoracle in terms of cost at two important tasks. Evoracle reduces resource requirements by an order of magnitude or more compared to clonal Sanger sequencing.

Supplementary References

1. Badran, A. H. *et al.* Continuous evolution of Bacillus thuringiensis toxins overcomes insect resistance. *Nature* **533**, 58 (2016).
2. Esvelt, K. M., Carlson, J. C. & Liu, D. R. A system for the continuous directed evolution of biomolecules. *Nature* **472**, 499–503 (2011).
3. Ravikumar, A., Arzumanyan, G. A., Obadi, M. K. A., Javanpour, A. A. & Liu, C. C. Scalable, Continuous Evolution of Genes at Mutation Rates above Genomic Error Thresholds. *Cell* **175**, 1946-1957.e13 (2018).
4. Lang, G. I. *et al.* Pervasive genetic hitchhiking and clonal interference in forty evolving yeast populations. *Nature* **500**, 571–574 (2013).
5. Cleary, B. *et al.* Detection of low-abundance bacterial strains in metagenomic datasets by eigengenome partitioning. *Nat. Biotechnol.* **33**, 1053–1060 (2015).
6. Illingworth, C. J. R. Fitness Inference from Short-Read Data: Within-Host Evolution of a Reassortant H5N1 Influenza Virus. *Mol. Biol. Evol.* **32**, 3012–3026 (2015).
7. Li, X., Saadat, S., Hu, H. & Li, X. BHap: a novel approach for bacterial haplotype reconstruction. *Bioinformatics* **35**, 4624–4631 (2019).
8. Aguiar, D. & Istrail, S. HapCompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. *J. Comput. Biol. J. Comput. Mol. Cell Biol.* **19**, 577–590 (2012).
9. Berger, E. *et al.* Improved haplotype inference by exploiting long-range linking and allelic imbalance in RNA-seq datasets. *Nat. Commun.* **11**, 4662 (2020).
10. Kuleshov, V. *et al.* Whole-genome haplotyping using long reads and statistical methods. *Nat. Biotechnol.* **32**, 261–266 (2014).
11. Pulido-Tamayo, S. *et al.* Frequency-based haplotype reconstruction from deep sequencing data of bacterial populations. *Nucleic Acids Res.* **43**, e105–e105 (2015).
12. Sobel Leonard, A. *et al.* The effective rate of influenza reassortment is limited during human infection. *PLOS Pathog.* **13**, e1006203 (2017).
13. Illingworth, C. J. R., Fischer, A. & Mustonen, V. Identifying Selection in the Within-Host Evolution of Influenza Using Viral Sequence Data. *PLOS Comput. Biol.* **10**, e1003755 (2014).

14. Illingworth, C. J. R. SAMFIRE: multi-locus variant calling for time-resolved sequence data. *Bioinformatics* **32**, 2208–2209 (2016).

15. Paszke, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Adv. Neural Inf. Process Syst. 32* 8024–8035 (2019).