Supplemental Material – Modeling and predicting cancer clonal evolution with reinforcement learning

Stefan Ivanovic[1] and Mohammed El-Kebir[1,2,†]

[1]Department of Computer Science, University of Illinois at Urbana-Champaign, IL 61801, USA
[2]Cancer Center at Illinois, University of Illinois Urbana-Champaign, Urbana, IL 61801, USA
[†]Correspondence: melkebir@illinois.edu

Contents

List of Figures

List of Tables

A Supplemental methods

Supplementary Material A.1 provides details on CloMu's model and its training. We describe additional prediction tasks in Supplementary Material A.2 Finally, in Supplementary Material A.3, we discuss bootstrapping.

A.1 CloMu's model

We begin by providing details on CloMu's neural network in Supplementary Material A.1.1. We then discuss how we used reinforcement learning to identify model parameters f_θ for the neural network. Specifically, Supplementary Material A.1.2 gives a brief introduction to policy gradients. In Supplementary Material A.1.3 we discuss how policy gradients can be applied to CloMu's model. Finally, in Supplementary Material A.1.4 we discuss a speedup for the typical case of a small number of trees per patient.

A.1.1 Neural network

CloMu uses a two layer neural network with a small number L of hidden neurons for the function f_θ. Specifically, the input to the neural network is the vector $\mathbf{c}$ and the output is a log rate $f_\theta(\mathbf{c}, s)$ for each mutation s. This model has $2Lm + m + L = O(Lm)$ total parameters. We use a tanh nonlinearity rather than a ReLU nonlinearity to reduce the risk of latent variables being stuck at zero during training. To optionally enforce the infinite sites assumption, we set $f_\theta(\mathbf{c}, s)$ to a very large negative number if $c_s = 1$ to ensure a mutation is not added twice to the same clone.

To enable easier regularization, which is used primarily for the sake of interpretability but also used to avoid overfitting on some extremely small datasets, we made an two additional modifications to the neural network. First, the neural network also contains a single scalar value for each input mutation, which is added

to all variables in the output equally. This allows one variable per mutation to influence the general fitness of the clone, allowing for a mutation to affect clonal fitness while only modifying one parameter (rather than modifying one parameter per output mutation). Specifically, this is equivalent to adding a function independent of s, $h_\theta(\mathbf{c})$, to $f_\theta(\mathbf{c}, s)$ like the below equation.

$$f_\theta(\mathbf{c}, s) := f_\theta(\mathbf{c}, s) + h_\theta(\mathbf{c}). \tag{1}$$

Second, a one-hot encoded variable representing the number of mutations in the clone is included. This allows the model to take into account the number of mutations on a clone without modifying the parameter for every individual input mutation. This therefore allows the model to easily know if the clone is the non-mutated root clone with only one parameter.

These two modifications are essentially useless and give no empirical advantage when no regularization is used, since these variables can automatically be computed and used from the already existing data. However, once L1 regularization is included during training, these variables help the model achieve sparsity. On very small datasets sparsity can be important to avoid overfitting; however, on larger datasets, sparsity is simply useful because it helps with interpretability. Having all variables that do not need to be nonzero set to zero helps with creating good plots and thus interpretability. This also likely helps with any downstream tasks that would use the latent representation of mutations or clones generated by the model.

A.1.2 Basic introduction to policy gradients

The type of reinforcement learning used in this paper is policy gradients. Define A as a sequence of actions the model may take, $R(A)$ as the reward for those actions, and $\Pr(A \mid f_\theta)$ as the probability our model takes those actions. As an example, if the environment was a video game, A could be a sequence of actions taken in the game, and $R(A)$ could be the score achieved by taking those actions. In our application, A will be the phylogenetic generation processes sampled from our model, which is demonstrated in Fig. S1. Define S_A as the set of all possible sequences of actions our model could take. In our application, S_A would correspond to the set of all tree generating processes. To optimize the reward that the model gets, one wants to increase the probability of high reward actions. Therefore, one wants to identify model parameters θ that maximize

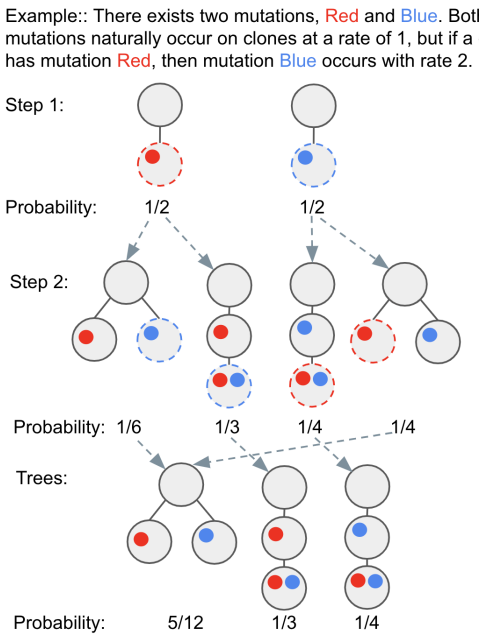


Figure S1: **Demonstration of tree generation.** This is a toy example to demonstrate how causal relationships affect the probability of phylogeny trees in a simple system. It also demonstrates how there are multiple ways of generating the same tree; specifically, trees 1 and 4 in step 2 are the same tree.

the below equation.

$$\sum_{A \in S_A} R(A) \Pr(A \mid f_\theta). \tag{2}$$

The ideal gradient for optimizing this would theoretically be the below equation.

$$\frac{d}{d\theta} \sum_{A \in S_A} R(A) \Pr(A \mid f_\theta) = \sum_{A \in S_A} R(A) \frac{d}{d\theta} \Pr(A \mid f_\theta). \tag{3}$$

In practice, it is infeasible to sum over all $A \in S_A$ since S_A may be a very large set. However, one can estimate (3) by sampling from $\Pr(A \mid f_\theta)$:

$$\sum_{A \in S_A} R(A) \frac{d}{d\theta} \Pr(A \mid f_\theta) = \sum_{A \in S_A} R(A) \Pr(A \mid f_\theta) \frac{d}{d\theta} \log(\Pr(A \mid f_\theta)) = \mathbb{E}_A[R(A) \frac{d}{d\theta} \log(\Pr(A \mid f_\theta))] \tag{4}$$

Therefore one uses the standard REINFORCE (Williams, 1992) gradient estimate

$$\mathbb{E}_A[R(A) \frac{d}{d\theta} \log(\Pr(A \mid f_\theta))]. \tag{5}$$

Note, this section is not intended to give a complete description of policy gradients. Rather, it explains the necessary content for an understanding of CloMu.

A.1.3 Policy gradients for CloMu

In our application, a sequence A of actions will be the phylogenetic generation processes G sampled from our model. Moreover, the set S_G of all possible sequences of actions corresponds to the set of all tree generating processes on m mutations. To use reinforcement learning, we need to meet two criteria. First, we need to be able to sample from our distribution $\Pr(G \mid f_\theta)$. For our model, this is very easy since G is generated by a simple probabilistic process (Section Independent clonal evolution problem). Second, we need to be able to formulate our objective as the below equation, where $R(G)$ is the reward function.

$$\sum_{G \in S_G} R(G) \Pr(G \mid f_\theta). \tag{6}$$

An important aspect of $R(G)$ is that the gradient with respect to θ that is used for optimization does not affect $R(G)$. To make this explicit and clear, instead of formulating our objective function as that objective function, we formulate the gradient of our objective function to fit the reinforcement learning objective function gradient. Specifically, to apply policy gradients, the gradient of our objective function must be formulated as below.

$$\sum_{G \in S_G} R(G) \frac{d}{d\theta} \Pr(G \mid f_\theta). \tag{7}$$

We can do so through the below manipulations of the gradient of the log likelihood.

$$\frac{d}{d\theta} \log \Pr(\mathcal{T}_1, \ldots, \mathcal{T}_n \mid f_\theta) = \frac{d}{d\theta} \sum_{p=1}^{n} \log\Big(\sum_{T \in \mathcal{T}_p} \sum_{G \in \mathcal{G}(T)} \Pr(G \mid f_\theta)\Big) \tag{8}$$

$$= \sum_{p=1}^{n} \left(\sum_{T \in \mathcal{T}_p} \sum_{G \in \mathcal{G}(T)} \Pr(G \mid f_\theta) \right)^{-1} \frac{d}{d\theta} \sum_{T \in \mathcal{T}_p} \sum_{G \in \mathcal{G}(T)} \Pr(G \mid f_\theta) \tag{9}$$

$$= \sum_{p=1}^{n} \left(\sum_{T \in \mathcal{T}_p} \sum_{G \in \mathcal{G}(T)} \Pr(G \mid f_\theta) \right)^{-1} \sum_{T \in \mathcal{T}_p} \sum_{G \in \mathcal{G}(T)} \frac{d}{d\theta} \Pr(G \mid f_\theta) \tag{10}$$

$$= \sum_{p=1}^{n} \sum_{T \in \mathcal{T}_p} \sum_{G \in \mathcal{G}(T)} \left(\sum_{T' \in \mathcal{T}_p} \sum_{G' \in \mathcal{G}(T')} \Pr(G \mid f_\theta) \right)^{-1} \frac{d}{d\theta} \Pr(G \mid f_\theta) \tag{11}$$

$$= \sum_{G \in S_G} \sum_{p=1}^{n} \frac{\delta(p, G)}{\sum_{T \in \mathcal{T}_p} \sum_{G' \in \mathcal{G}(T)} \Pr(G' \mid f_\theta)} \cdot \frac{d}{d\theta} \Pr(G \mid f_\theta) \tag{12}$$

In the above derivation, we define $\delta(p, G)$ as 1 if $G \in \mathcal{G}(T)$ for some $T \in \mathcal{T}_p$, and $\delta(p, G) = 0$ otherwise. In other words $\delta(p, G)$ indicates whether G corresponds to an observed tree of patient p. Equation (12) corresponds to the reinforcement learning gradient if we set the reward function as below.

$$R(G) = \sum_{p=1}^{n} \frac{\delta(p, G)}{\sum_{T \in \mathcal{T}_p} \sum_{G' \in \mathcal{G}(T)} \Pr(G' \mid f_\theta)}. \tag{13}$$

In reinforcement learning, one is sampling from $\Pr(G \mid f_\theta)$, so we can easily estimate $\sum_{T \in \mathcal{T}_p} \sum_{G \in \mathcal{G}(T)} \Pr(G \mid f_\theta)$ by simply observing what fraction of generated G are in $\mathcal{G}(T)$ with $T \in \mathcal{T}_p$. In other words, we can observe what fraction of generated trees are possible for patient p. This fraction changes as we train the model, and therefore must constantly be updated. However, assuming the model does not change much in a small number of iterations, it does not need to be calculated from scratch in every gradient update. The reinforcement learning gradient is now the below expression

$$\mathbb{E}_G[R(G) \frac{d}{d\theta} \log(\Pr(G \mid f_\theta))]. \tag{14}$$

A.1.4 Policy gradients with few trees per patient for CloMu

When the set of possible trees per patient is small enough to be reasonably explicitly enumerated, it can be beneficial to directly model the probability of each individual tree, rather than indirectly modeling the probability of each tree through reinforcement learning. In particular, when there are very few possible trees per patient, the reinforcement learning method introduced in the previous section may struggle to find those trees and would perform much better if the probability of each tree was explicitly modeled. However, one cannot just avoid using reinforcement learning in those cases because there could still be a very large number of generation processes per tree. To solve both of these problems, we use a modified reinforcement learning system where the probability of each possible tree is explicitly modeled, but the probability of each generation process is not explicitly modeled and only learned through reinforcement learning. This version of CloMu was implemented and is used for all experiments as well as made publicly available.

More specifically, for each possible tree T in the dataset (the union of $\mathcal{T}_p$ for all patients $p \in [n]$), we only sample generation processes $G \in \mathcal{G}(T)$. Of course, this requires a modification of the objective

function to compensate for the change in the sampling procedure. Define $P_T(G \mid f_\theta)$ as the probability of generating G with our procedure restricted such that it always generates processes which give the tree T. Define $R'(G) = R(G)\Pr(G \mid f_\theta)/P_T(G \mid f_\theta)$. $R'(G)$ is the new reward function for this adjusted sampling procedure as shown in the below calculation.

$$\mathbb{E}_{G\sim P_T}[R(G)\log(\Pr(G \mid f_\theta))] = \mathbb{E}_{G\sim P_T}[\Pr(G \mid f_\theta)/P_T(G \mid f_\theta)R(G)\log(\Pr(G \mid f_\theta))] \tag{15}$$

$$= \mathbb{E}_{G\sim P_T}[R'(G)\log(\Pr(G \mid f_\theta))]. \tag{16}$$

A.2 Additional CloMu prediction tasks

This section provides details on determining evolutionary pathways (Supplementary Material A.2.1), multi-mutation causality (Supplementary Material A.2.2) and consensus trees (Supplementary Material A.2.3).

A.2.1 Determining evolutionary pathways

For the sake of interpretability, it is useful to represent patterns in cancer evolution as evolutionary pathways that frequently occur. We define an evolutionary pathway of interchangeable mutations as a list of sets of mutations such that the cancer obtained one mutation from each set in order. More precisely, we define an *evolutionary pathway* of length ℓ as sets of mutations $S_1, \ldots, S_\ell$. A clone with mutations $s_1, \ldots, s_\ell$ in temporal order fits the evolutionary pathway $S_1, \ldots, S_\ell$ if $s_k \in S_k$ for all $k \in [\ell]$.

The probability of an evolutionary pathway $S_1, \ldots, S_\ell$ is defined as the probability that a clone with ℓ mutations fits the evolutionary pathway. Define $\mathbf{c}(s_1, \ldots, s_k)$ as the clone with mutations $s_1, \ldots, s_k$. The probability of the evolutionary pathway $S_1, \ldots, S_\ell$ is given by the below expression.

$$\Pr(S_1, \ldots, S_\ell \mid f_\theta) = \sum_{(s_1,\ldots,s_\ell)\in S_1\times\ldots\times S_\ell} \Pr((\mathbf{c}_0, s_1), (\mathbf{c}(s_1), s_2), \ldots, (\mathbf{c}(s_1, \ldots, s_{\ell-1}), s_\ell) \mid f_\theta). \tag{17}$$

Fig. S2 gives an example pathway and set of trees for this calculation. As a null model, we consider the probability $\Pr_0(s \mid f_\theta, \ell)$ of a mutation s occurring in a pathway of length ℓ.

$$\Pr_0(s \mid f_\theta, \ell) = \sum_{\substack{\{s_1,\ldots,s_\ell\}\subseteq[m] \\ \text{with } s\in\{s_1,\ldots,s_\ell\}}} \Pr(\{s_1\}, \ldots \{s_\ell\} \mid f_\theta), \tag{18}$$

where $\Pr(\{s_1\}, \ldots, \{s_\ell\} \mid f_\theta)$ is the probability of the pathway $\{s_1\}, \ldots \{s_\ell\}$.

The null probability, ignoring causal relationship, of a mutation s occurring at a particular step for a particular clone would be $\Pr_0(s \mid f_\theta, \ell)/\ell$. The null probability for a pathway would then be the null probability of generating every clone that fits the pathway based on the default probability in every step of generating the clone. Specifically, the null probability of a pathway $S_1, \ldots, S_\ell$ is defined by the below equation.

$$\Pr_0(S_1, \ldots, S_\ell \mid f_\theta) = \sum_{(s_1,\ldots,s_\ell) \in S_1 \times \ldots \times S_\ell} \prod_{k=1}^{\ell} \Pr_0(s_k \mid f_\theta, \ell)/\ell = \prod_{k=1}^{\ell} \sum_{s_k \in S_k} P_0(s_k \mid f_\theta, \ell)/\ell. \tag{19}$$

The expression $\Pr(S_1, \ldots, S_\ell \mid f_\theta)/\Pr_0(S_1, \ldots S_\ell \mid f_\theta)$ defines how much more likely a pathway is to occur than what one would expect if there were no causal relationships. For a pathway to be meaningful, one wants $\Pr(S_1, \ldots, S_\ell \mid f_\theta)/\Pr_0(S_1, \ldots, S_\ell \mid f_\theta)$ to be high in order to indicate that the pathway demonstrates meaningful causal relationships. Additionally, one wants $\Pr(S_1, \ldots, S_\ell \mid f_\theta)$ itself to be high so that the pathway occurs frequently enough to be meaningful. Define the score of a pathway with the below equation, where $a > 0$ is a nonnegative constant.

$$\text{score}(S_1, \ldots, S_L, a) = \Pr(S_1, \ldots, S_L \mid f_\theta)^a \frac{\Pr(S_1, \ldots, S_L \mid f_\theta)}{\Pr_0(S_1, \ldots, S_L \mid f_\theta)}. \tag{20}$$

The value of a can be adjusted based on the specific application. When implementing this we operate in logarithmic space.

In order to determine the most meaningful pathways, one can optimize for the score of the pathway. Specifically, given a length $\ell \leq m$ the procedure used in this paper starts with an evolutionary pathway $S_1, \ldots, S_\ell$ where each $S_k = [m]$. Then, in a greedy fashion, we add or remove a single mutation that maximizes $\text{score}(S_1, \ldots, S_\ell, a)$. We repeat this until a locally optimal pathway is found. Once an individual optimal pathway is found, we set the probability $\Pr((\mathbf{c}_0, s_1), \ldots, (\mathbf{c}(s_1, \ldots, s_{\ell-1}), s_\ell) \mid f_\theta)$ used in (17) to 0 in a lookup table so as to discourage repeatedly finding the same or similar pathways. Then, the optimization procedure is run again to generate a new pathway. If the score of a pathway reaches below $\Pr([m], \ldots, [m] \mid f_\theta)^a$, it is not at all meaningful, and the process of generating optimal pathways ends. This cut off is used since the pathway consisting of only the set of all mutations at every step has that score.

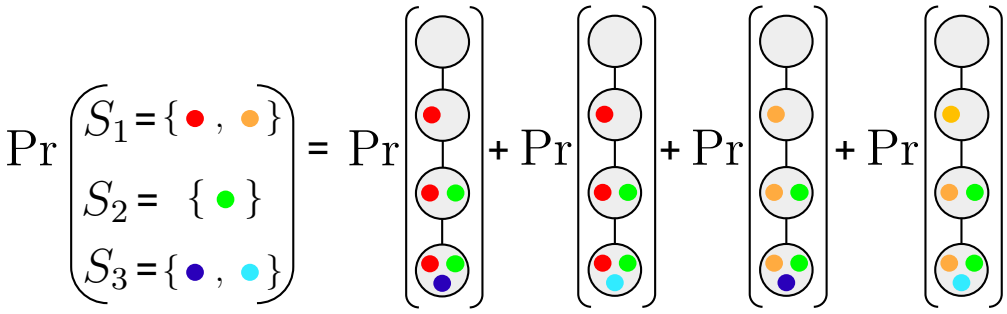


Figure S2: **A diagram of evolutionary pathways.** Eq. (17) states that the probability of an evolutionary pathway is equal to the sum of the probability of all trees that fit the evolutionary pathway (sequentially acquiring one mutation from each step in the pathway). In this example, S_1 has two mutations, S_2 has a single mutation and S_3 has two mutations, leading to $2 \cdot 1 \cdot 2 = 4$ possible trees. The probability of this evolutionary pathway equals the sum of the probabilities of the four resulting trees.

A.2.2 Determining multi-mutation causality

One advantage of CloMu is that it allows for complex interactions between mutations while utilizing a small number of parameters. The ability of CloMu to detect the effects of combinations of mutations was demonstrated in Simulations (II) in the Main Text. However, here we introduce *multi-mutation causality* to more explicitly describe this phenomenon. While in the Main Text we defined absolute and relative causality between a mutation s and another mutation t, here we extend the definition to capture the causal relationship between a combination of mutations and some other mutation. Specifically, we define absolute *multi-mutation causality* from a set of mutations in a clone $\mathbf{c}$ to a mutation t as $f_\theta(\mathbf{c}, t) - f_\theta(\mathbf{c}_0, t)$. Therefore, for multi-mutation causality, we say the mutations in $\mathbf{c}$ *cause* t if $f_\theta(\mathbf{c}, t) - f_\theta(\mathbf{c}_0, t) > \tau$, the mutations in $\mathbf{c}$ *inhibit* t if $f_\theta(\mathbf{c}, t) - f_\theta(\mathbf{c}_0, t) < -\tau$, and there is *no causal relationship* from the mutations in $\mathbf{c}$ to t otherwise. The simplest non-trivial case of multi-mutation causality is a pair of mutations causing some third mutation. Let $\mathbf{c}$ be the clone with exactly mutations s and t. We say s and t cause r if $f_\theta(\mathbf{c}, r) - f_\theta(\mathbf{c}_0, r) > \tau$, s and t inhibit r if $f_\theta(\mathbf{c}, r) - f_\theta(\mathbf{c}_0, r) < -\tau$, and there is no causal relationship from s and t to r otherwise. Our ability to detect multi-mutation causality is tested in Supplementary Material B.3.11.

A.2.3 Identifying consensus trees

One prediction task introduced in the RECAP (Christensen et al., 2020) and REVOLVER (Caravagna et al., 2018) papers is the identification of a small number of consensus trees. Accomplishing this task requires the knowledge that there are a small number of true trees in each dataset, which is not an intrinsic property

of our model. To adjust for this fact, a post processing step to our model must be added. Specifically, tree predictions are generated for each patient according to the following procedure. As previously defined, let $\Pr(T \mid f_\theta)$ be the probability that tree T is generated according to the model, and $\mathcal{T}_p$ be the set of possible trees for patient p. For a set $\mathcal{S}$ of trees define $\Pr(\mathcal{S} \mid f_\theta)$ as the below equation.

$$\Pr(\mathcal{S} \mid f_\theta) = \prod_{p=1}^{n} \max_{T \in \mathcal{S} \cap \mathcal{T}_p} \Pr(T \mid f_\theta). \tag{21}$$

In other words, $\Pr(\mathcal{S} \mid f_\theta)$ is the probability of the maximum likelihood possible trees $(T_1, \ldots, T_n) \in \mathcal{T}_1 \times \ldots \times \mathcal{T}_n$ when we restrict the set of trees to $\mathcal{S}$. Thus, $\Pr(\mathcal{S} \mid f_\theta)$ represents how well $\mathcal{S}$ works as the set of true trees.

Define $\mathcal{S}_0 = \bigcup_{p=1}^{n} \mathcal{T}_p$ as the set of all input trees. Define $\mathcal{S}_{t+1}$ given $\mathcal{S}_t$ as follows.

$$\mathcal{S}_{t+1} = \mathcal{S}_t \setminus \{\text{argmax}_{T \in \mathcal{S}_t} \Pr(\mathcal{S}_t \setminus \{T\} \mid f_\theta)\}. \tag{22}$$

In other words, $\mathcal{S}_{t+1}$ is $\mathcal{S}_t$ with the element removed which decreases $\Pr(\mathcal{S} \mid f_\theta)$ by the smallest quantity. Define t^* as the last t with $\mathcal{T}_p \cap \mathcal{S}_t$ non-empty for all p. Our final predicted tree T_p^* for each patient p is given by the below equation.

$$T_p^* = \text{argmax}_{T \in S_{t^*} \cap \mathcal{T}_p} \Pr(T \mid f_\theta) \tag{23}$$

In simple terms, this procedure works by iteratively removing trees from the set of possible trees while predicting the maximum likelihood tree for each patient until no more trees can be removed from the set of possible trees. This then results in predictions $T_1^*, \ldots, T_n^*$.

A.3 Bootstrapping

One way to verify the stability of predictions is to perform bootstrapping on the patients and running CloMu on the bootstrapped data. Specifically, one can randomly sample patients with replacement to produce a new datasets of the same size as the original data set. After running CloMu on bootstrapped instances, one can produce predictions on each instance such as causality or fitness. The distribution of predictions across bootstrapped instances gives information about the confidence of these predictions.

Row	Measurement	Mutation to be acquired by clone		
		Mut. $t = 2$	Mut. $t = 3$	Mut. $t = 4$
1	Mutation rates $\lambda(\mathbf{c}_0, t)$	1	1	1
2	Mutation rates $\lambda(\mathbf{c}_1, t)$	5	10	15
3	Absolute causality $A(s, t)$	$\log(5/1)$	$\log(10/1)$	$\log(15/1)$
4	Mutation probabilities for $\mathbf{c}_1$	$1/6$	$1/3$	$1/2$
5	Mutation rates $\lambda(\mathbf{c}_2, t)$	0	1	1
6	Mutation probabilities for $\mathbf{c}_2$	0	$1/2$	$1/2$
7	Mutation rates $\lambda(\mathbf{c}_3, t)$	1	0	1
8	Mutation probabilities for $\mathbf{c}_3$	$1/2$	0	$1/2$
9	Mutation rates $\lambda(\mathbf{c}_4, t)$	1	1	0
10	Mutation probabilities for $\mathbf{c}_4$	$1/2$	$1/2$	0
11	Averaged mutation probabilities	$7/24$	$8/24$	$9/24$
12	Relative causality $R(s, t)$	$\log(4/7) < 0$	$\log(1/3) = 0$	$\log(4/3) > 0$

Table S1: **An example demonstrating absolute and relative causality in the presence of a highly fit mutation.** We consider a theoretical dataset with $m = 4$ mutations including a highly fit mutation $s = 1$. For the sake of simplicity, we assume mutation 1 occurs with a rate $\lambda(\mathbf{c}, 1)$ of 0 on all clones $\mathbf{c}$. Row 3 is calculated as the log of row 2 divided by row 1. Row 4 is calculated from row 2 by dividing each column by the sum of the three columns. This is calculating $\lambda(s, t)/\sum_r \lambda(s, r)$. The same calculation gives row 6 from row 5, row, row 8 from row 7 and row 10 from row 9. Row 11 is the average of rows 4, 6, 8, and 10. Row 12 is the log of row 4 divided by row 11. Mutation 1 is highly fit as reflected by the high rates in row 2 (as compared to rows 1, 5, 7, and 9). Consequently, it causes all other mutations in terms of absolute causality (row 3). However, in terms of relative causality, it causes mutation 4, inhibits mutation 2, and has no effect on mutation 3 (row 12).

B Supplemental results for simulations

We discuss the setup of our simulations in Supplemental Material B.1. Next, we discuss the baseline methods and their parameters in Supplemental Material B.2. Finally, we provide detailed simulation results in Supplemental Material B.3.

B.1 Setup

Simulations (I-a). In the Main Text, we discussed simulations with $n = 500$ patients, $m = 10$ mutations including 5 driver mutations, and random positive causal relationships between driver mutations. To generate the set $\mathcal{T}_p$ of trees for each patient p, we simulated bulk sequencing using five sequencing samples. Specifically, we determined mixture proportions for each clone in each sample uniformly at random, obtained the corresponding mutation frequencies, and then enumerated all possible trees that could have possibly generated these mutation frequencies (using Prüfer sequences (Prüfer, 1918)). These simulation instances are referred to as Simulations (I-a). The rate multipliers were chosen to be 11. That is, log rates were increased by $\log(11)$ so that rates are increased by a factor of 11. The rationale is that increasing the

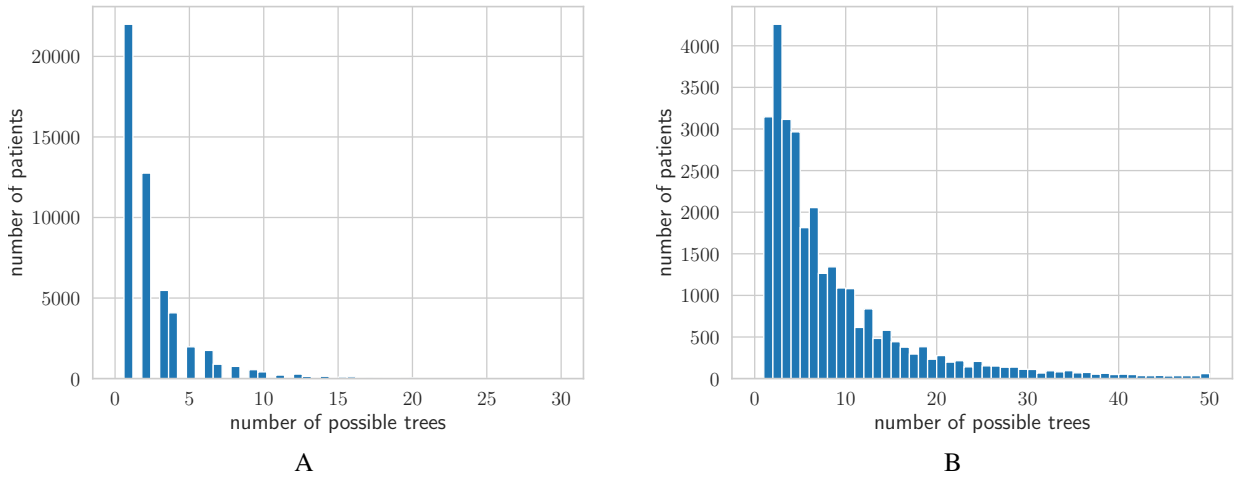


Figure S3: **Number of trees in simulated data.** Histogram of the number of trees for each patient in (A) Simulations (I-a,b,c) used to assess tree selection and causality. (B) Simulations (II) used to assess mutation interchangeability and evolutionary pathway identification.

rate of a single mutation by a factor of 11 results in the overall rate of mutations on a clone doubling if there exists a total of 10 mutations. The initial rate of mutations on the clone is $10 = 1 + 9$, and the new rate is $20 = 11 + 9$. Thus, a driver mutation that causes N other mutations has a fitness $(1 + N)$ times that of a passenger (which is very realistic as shown in our analysis fitness in breast cancer and AML datasets).

Simulations (I-b) and (I-c). To simulate causality with interchangeable mutations, we considered each mutation as a driver mutation. We partitioned the set of m mutations into 5 pairwise disjoint sets $S_1, \ldots, S_5$ of equal size. For each pair (S_i, S_j) of mutation sets (where $i, j \in \{1, \ldots, 5\}$), we determined S_i as causing S_j, S_i as inhibiting S_j and S_i and S_j having no causal effect with uniform probabilities of $1/3$ for each case. With a slight abuse of notation, we set the rate multiplier $g(S_i, S_j)$ to 11 if S_i causes S_j, $1/11$ if S_i inhibits S_j, and 1 if there is no causal relationship from S_i to S_j. The resulting rate multiplier $g(s, t)$ of mutations $s, t \in [m]$ equals the rate multiplier $g(S_i, S_j)$ of their comprising sets (i.e., $s \in S_i$ and $t \in S_j$). Similarly, we computed rates $\lambda(\mathbf{c}, t)$ as in the Main Text and generated a ground-truth T_p^* for each patient $p \in [n]$ by first drawing the number ℓ of mutations of tree T_p^* uniformly from $\{5, 6, 7\}$. Subsequent bulk sequencing simulation with five sequencing samples, as discussed above, resulted in sets $\mathcal{T}_1, \ldots, \mathcal{T}_n$ of trees per patient (Fig. S3A). We varied the number $m \in \{10, 15\}$ of mutations. Consequently, the size of each set S_i will either be 2 (for $m = 10$) or 3 (for $m = 15$). We simulated $n = 400$ patients for $m = 10$ and $n = 600$ patients for $m = 20$. For each setting of m, we generated 20 simulation instances. These instances

are denoted as Simulations (I-b) and (I-c) (see Table 2).

Simulations (II). For evolutionary pathway assessment, we generated simulation instances that consist of one pathway (25% probability) or two pathways (75% probability). We considered $m = 20$ mutations, some of which will occur in pathways and are drivers and the remaining mutations are passengers. Each pathway consists of three pairwise disjoint sets S_1, S_2, S_3 of mutations. In the case of two pathways all mutation sets are pairwise disjoint. The number of mutations in a set S_i is drawn uniformly from $\{1, 2, 3\}$. As such, the number of drivers ranges from 3 mutations (in the case of a single pathway with one mutation per set) to 18 mutations (in the case of two pathways with three mutations per set). To generate trees according to these evolutionary pathways we defined rates $\lambda(\mathbf{c}, t)$ according to the stage and the pathway a particular clone is at. Specifically, clone $\mathbf{c}$ is at stage $i \in \{0, \ldots, 3\}$ of the pathway S_1, S_2, S_3 if clone $\mathbf{c}$ contains at least one mutation from each set $S_1, \ldots, S_i$ and does not contain a mutation from set S_{i+1} (here if $i = 3$ then $S_4 = \emptyset$). We set the rate $\lambda(\mathbf{c}, s)$ to 5 if $\mathbf{c}$ is at stage 0 for both pathways and $s \in S_1$ for some pathway. We set the rate $\lambda(\mathbf{c}, s)$ to 20 if $\mathbf{c}$ is at stage $i > 0$ for some pathway and $s \in S_{i+1}$ for that pathway. If neither condition is met, we set rate $\lambda(\mathbf{c}, s) = 1$. Note, by definition, the normal clone $\mathbf{c}$ is at stage 0 for both pathways. Additionally, if a clone is at stage 3 then all mutations have rate 1. Similarly to the other simulations, we use these rates to generate a ground-truth T_p^* for each patient $p \in [n]$ by first drawing the number ℓ of mutations of tree T_p^* uniformly from $\{5, 6, 7\}$. Subsequent bulk sequencing simulation with five sequencing samples, as discussed above, resulted in sets $\mathcal{T}_1, \ldots, \mathcal{T}_n$ of trees per patient (Fig. S3B). We set $n = 500$ patients, and generated a total of 30 simulation instances. These instances are denoted as Simulations (II) (see Table 2).

Simulations (III). In addition to newly generated simulation instances, we also considered the previously published RECAP simulated dataset. These simulations have a small number of patient subtypes each of which has a single phylogeny tree. Specifically, within the RECAP simulated data, two groups of simulated datasets are used. One group has 5 mutations per patient and 5 mutations total. The second group has 7 mutations per patient and 12 mutations total. For each dataset (in each group), there are $k \in \{1, \ldots, 5\}$ clusters among $n \in \{50, 100\}$ patients. Each patient cluster corresponds to one true tree. In the published RECAP simulations, each patient was randomly assigned to a true tree, then the set of possible trees is generated by simulated bulk frequency measurements. The goal in this simulated dataset is to assign the correct tree to each patient. We denote these instances as Simulations (III) as shown in Table 2.

Simulations (IV). Finally, we considered previously published TreeMHN simulations. Specifically, we restricted our comparison to instances with $n = 300$ patients, and randomly selected 20 instances for each number $m \in \{10, 15, 20\}$ of mutations. These simulations instances are referred to as Simulations (IV).

B.2 Baseline methods and parameters

We compared CloMu to TreeMHN (Luo et al., 2023), RECAP (Christensen et al., 2020), REVOLVER (Caravagna et al., 2018), GeneAccord (Kuipers et al., 2021), and HINTRA (Khakabimamaghani et al., 2019). In the following, we will discuss which methods were included for each class of simulation instances, list method parameters and provide details on any post-processing that was conducted.

Methods per simulation class. We used Simulations (I-a) to assess tree selection and causality inference. For the former, we compared CloMu to RECAP and REVOLVER. For the latter, we included all methods in our benchmarking. We used Simulations (I-b) and (I-c) to further asses the differences between CloMu and TreeMHN in causality prediction in the presence of interchangeable mutations. Simulations (II) were used to demonstrate the detection of evolutionary pathways of interchangeable mutations. The methods RECAP and REVOLVER had their predictions post-processed in order to form a baseline to compare with CloMu. Simulations (III) are from the RECAP paper and further asses tree selection. On these simulations, CloMu is compared against results form the RECAP paper including results for RECAP, REVOLVER, and HINTRA. Simulations (IV) are from the TreeMHN paper and give an additional comparison between TreeMHN and CloMu for the task of predicting causal relationships.

Method parameters. On Simulations (IV), TreeMHN was ran with default parameters and the use of stability selection. For comparison on Simulations (I-a,b,c), we ran TreeMHN with the penalty parameter set to maximize precision and recall. Additionally, we did not use stability selection, since improving precision at the cost of recall would not improve their results. We ran RECAP and REVOLVER with their default parameters. We ran GeneAccord with default parameters. We did not run HINTRA due to its lack of scalability. However, we did compare with HINTRA on RECAP's data (Simulation (III)), which was ran with default parameters.

Predicted relationship	True relationship		
	No relationship	Positive relationship	Negative relationship
No relationship	True Negative	False Negative	False Negative
Positive relationship	False Positive	True Positive	False Positive
Negative relationship	False Positive	False Positive	True Positive

Table S2: This table demonstrates how true positives, true negatives, false positives and false negatives are determined based on predicted and true causal relationships in the case of signed and bidirectional relationships.

Post-processing. For Simulations (I-a) and (II), which assess causality and evolutionary pathway inference, respectively, we included RECAP and REVOLVER despite their implementation not directly supporting these tasks. RECAP and REVOLVER are consensus tree methods (Main Text Table 1), which select one tree per patient. To support causality inference for RECAP and REVOLVER, we applied a post-processing step. Specifically, we processed RECAP's and REVOLVER's selected trees for each patient from the set of possible trees. We then extracted all edges from these trees and determined the frequency of each edge. For Simulations (I-a), we considered there to be a positive causal relationship from mutation s to mutation t if the edge (s, t) occurred with a frequency above a fixed threshold. We used a threshold of 40 based on the number $n = 500$ patients for Simulations (I-a). For Simulations (II), the goal is to predict the edges which occur in the simulated pathways. Specifically, if a ground-truth pathway contains sets S_i and S_{i+1}, we say that pathway contains all edges (s, t) with $s \in S_i$ and $t \in S_{i+1}$. Given a threshold frequency, RECAP/REVOLVER predicts an edge to exist for a pathway in a simulation if that edge occurs in the selected trees with a frequency above the threshold. We used a threshold of 20 based on the number $n = 500$ of patients for Simulations (II).

B.3 Results

Evaluating bidirectional and signed causal relationships. To evaluate bidirectional causality for signed causal relationships, one must calculate precision and recall from predicted and true causal relationships. First, every ordered pair of mutations is considered. Then, if some causal relationship (positive or negative) from s to t is predicted correctly, this is considered a true positive. Similarly, if no causal relationship is predicted from s to t and there is no true causal relationship from s to t, then this is considered a true negative. If there is a predicted causal relationship (positive or negative) from s to t, and this predicted relationship is incorrect for any reason, then it is considered a false positive. Note, this implies that if the predicted

causal relationship is positive (or negative) and the true causal relationship is negative (or positive), then it is considered a false positive, despite there being a true causal relationship. This information is summarized in Table S2. Using this, each ordered pair of mutations can be classified as either a true positive, true negative, false positive or false negative. Summing these numbers across all ordered pairs of mutations gives total true positives, true negative, false positives and false negatives. Then, the precision is defined as the true positives divided by the sum of true positives and false positives. The recall is defined as the true positives divided by the sum of true positives and false negatives. These definitions follow those used in TreeMHN (Luo et al., 2023).

B.3.1 Bootstrapping – Simulations (I-a)

We applied bootstrapping to the last simulation instance from Simulations (I-a), for a total of 100 times. We recorded the median as well as the 90% confidence interval of the predicted absolute causality values. Fig. S4 shows predictions on the instance from Simulations (I-a), showing that the lower bound for pairs of mutations with a true causal relationship (Fig. S4C) vastly exceeds the upper bound for pairs of mutations without a true causal relationship (Fig. S4D). Additionally, we ranked the strength of the predicted causal relationship for the 90 ordered pairs of mutations. There were 10 actual causal relationships, and thus having a ranking of at least 81 is consistent with there being a causal relationship and having a ranking of at most 80 is consistent with there being a non-causal relationship. Fig. S5 shows the 90% confidence interval on the predicted causal relationship strength rankings. In all cases on all 100 simulation instances, actual causal relationships have a ranking at least 81 and non-causal relationships have a ranking at most 80, indicating perfectly correct rankings on all instances.

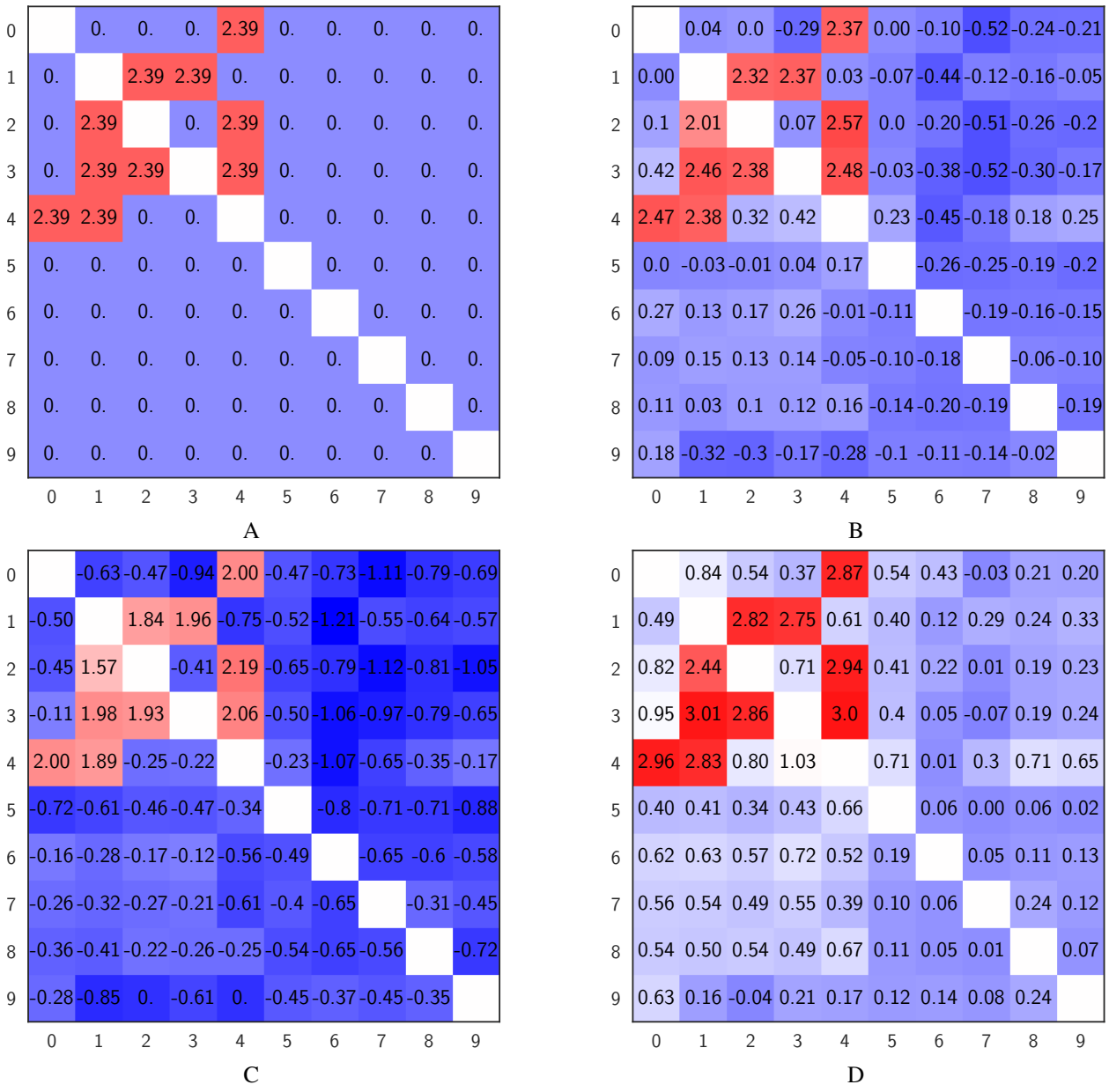


Figure S4: **Bootstrapping with** 100 **replicates demonstrates CloMu's robust absolute causality predictions on a Simulations (I-a) instance.** True and predicted causal relationships are shown. Mutations are sorted such that the first 5 mutations are driver mutations and the remaining 5 mutations are passenger mutations. The 90% confidence interval for each mutation contains 90 values from 90 simulation instances and excludes the highest and lowest 5 values. (A) Ground-truth absolute causality values. (B) The median absolute causality predictions on 100 bootstrapped instances. (C) The 6th lowest absolute causality predictions. (D) The 6th highest absolute causality predictions. The lower bound for pairs of mutations with a true causal relationship vastly exceeds the upper bound for pairs of mutations without a true causal relationship.

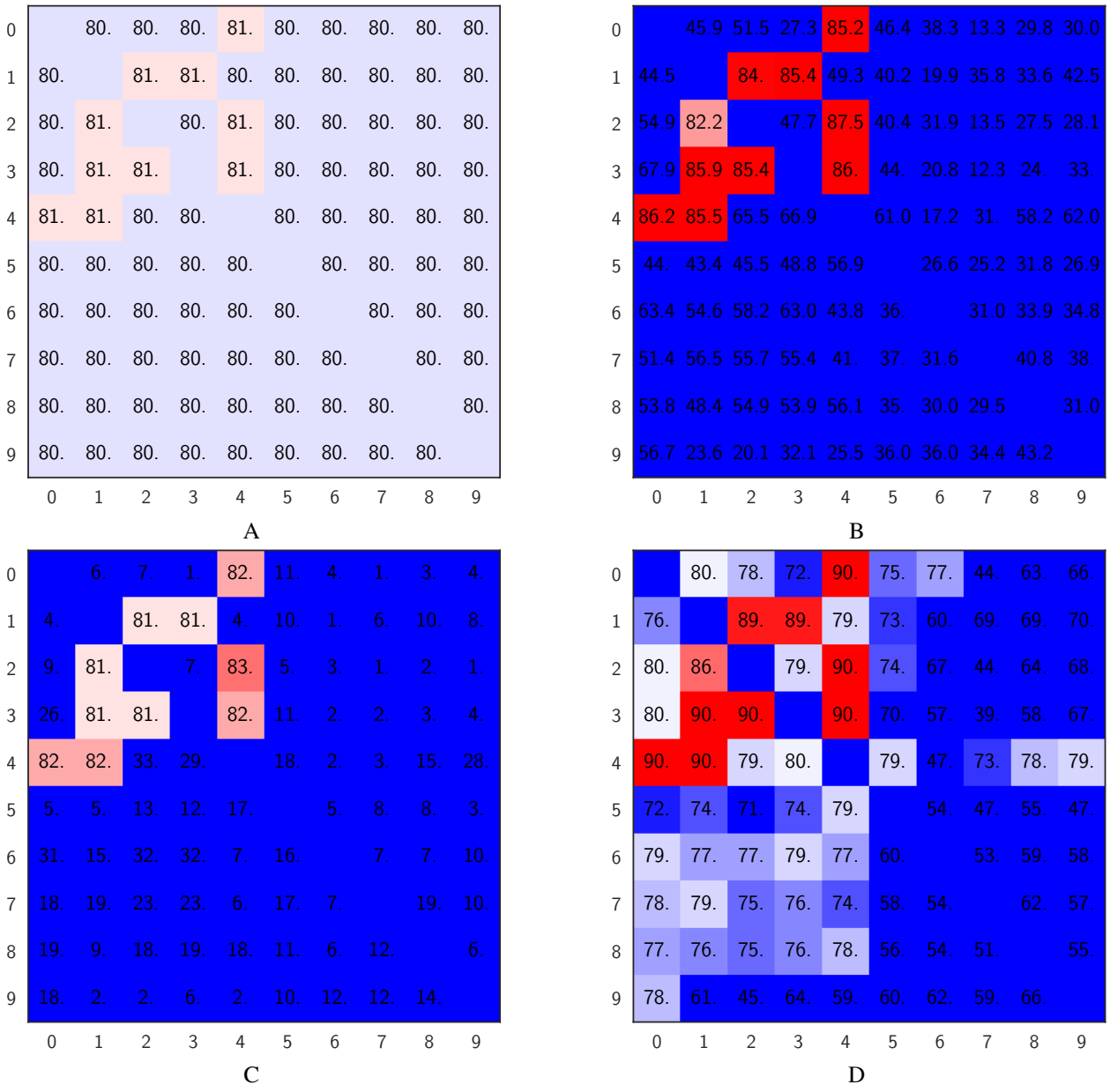


Figure S5: **The absolute causality strength rankings are shown to be robust via boostrapping with** 100 **replicates on a Simulations (I-a) instance.** Causal relationships are ranked from 1 to 90 based on the absolute causality value. There are 10 ground truth causal relationships, and so a ranking of at least 81 indicates a causal relationship and a ranking of at most 80 indicates no causal relationship. To emphasize rankings near these values, the colormap ranges from $76 = 81 - 4$ to $85 = 81 + 4$. True and predicted causal relationships are shown. Mutations are sorted such that the first 5 mutations are driver mutations and the remaining 5 mutations are passenger mutations. The 90% confidence interval for each mutation contains 90 rankings from 90 simulation instances and excludes the highest and lowest 5 rankings. (A) Ground-truth rankings are shown. Specifically, causal relationships are given the lowest valid ranking for a causal relationship (81), and non-causal relationships are given the highest valid ranking for non-causal relationships (80). There is no ground-truth difference in ranking among causal relationships or among non-causal relationships. (B) The median absolute causality rankings on 100 bootstrapped instances. (C) The 6th lowest absolute causality rankings. (D) The 6th highest absolute causality rankings.

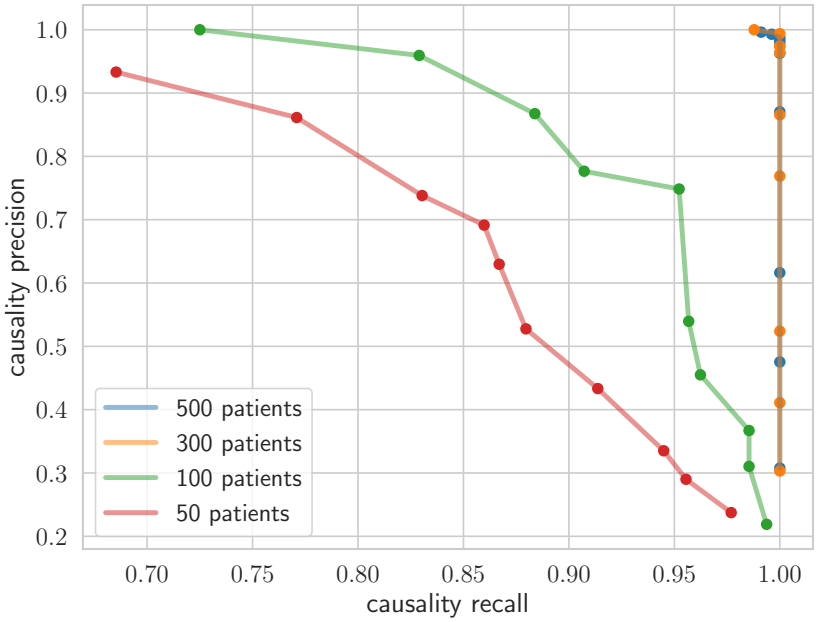
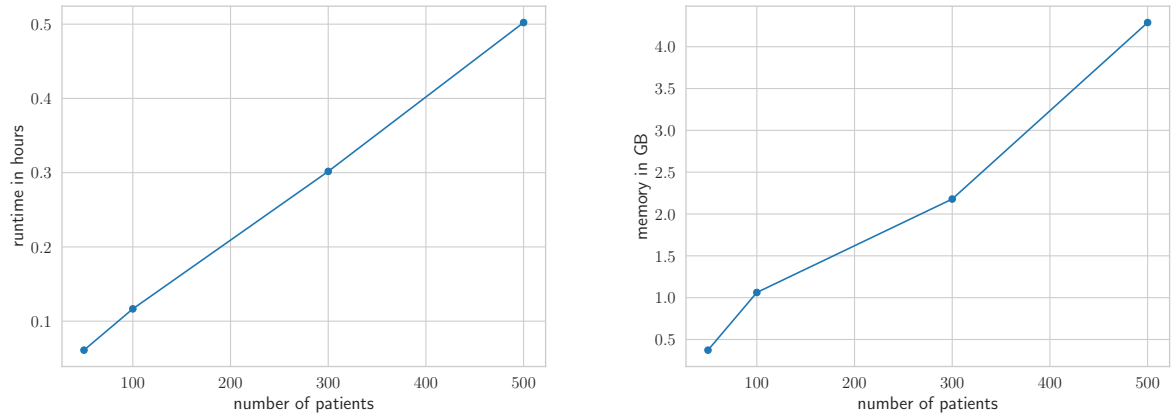


Figure S6: **CloMu gives accurate predictions even on simulations with few patients**. We consider a modification of Simulations (I-a), reducing the number n of patients from 500 (blue) to 300 (orange), 100 (green) and 50 (red). The resulting precision and recall values are somewhat reduced but remain high. The curve for 500 patients and 300 patients overlap since they are both nearly perfect.

B.3.2 Decreasing numbers of patients – adaptation of Simulations (I-a)

While the breast data had $n = 1756$ patients (Razavi et al., 2018), typically the number of patients in current cohort-level cancer phylogeny data consist of $n \approx 100$ patients (Morita et al., 2020; Jamal-Hanjani et al., 2017; Turajlic et al., 2018). However, Simulations (I-a) contain $n = 500$ patients per simulation instance. To assess CloMu performance with fewer patients, we generated new simulations with the same exact setup as Simulations (I-a), but with $n \in \{50, 100, 300\}$ patients per simulation instance. More specifically, for each number n of patients we generated 20 simulation instances. Fig. S6 demonstrates that although the performance of CloMu in terms of causality precision and recall decreased with decreasing number of patients, it remained acceptable.

The CloMu software has very reasonable runtimes and memory requirements. Consequently, for all experiments we trained CloMu on a laptop with a 2.4 GHz CPU and 64 GB of RAM without using a GPU. To demonstrate the effect of the number of patients on computational requirements, we plotted the memory usage and runtime of CloMu on simulations based on Simulations (I-a) but with differing numbers $n \in \{50, 100, 300, 500\}$ of patients. These results are shown in Fig. S7. The memory usage values are each for a single instance, and the runtimes are averaged over 20 instances. The mean running time ranged from



A B

Figure S7: **The runtimes and memory usage of CloMu ran on instances based on Simulations (I-a) with differing numbers of patients.** Simulations were generated following Simulations (I-a) but with $n \in \{50, 100, 300, 500\}$ patients. (A) The mean runtime of CloMu across 20 instances for varying number n of patients. (B) The memory usage of CloMu for a single instance for each number n of patients.

3 minutes for $n = 50$ patients to 30 minutes for $n = 500$ patients. The memory usage was 382 MB for $n = 50$ patients and 4.288 GB for $n = 500$ patients. The runtime and memory usage values seem to be approximately linear in the number of patients.

B.3.3 Decreasing numbers of mutations – adaptation of Simulations (I-a)

Simulations (I-a) also contain a randomly selected number of mutations per patient, ranging from 5 to 7. However, in the AML dataset (Morita et al., 2020), which performed panel-based single-cell DNA sequencing, the number of mutations per patient can be much smaller, reducing the amount of information provided by each patient. To assess performance in this regime, we generated new 20 simulations based on Simulations (I-a) with $n = 100$ patients and the number of mutations per patient generated based on the distribution observed in the AML dataset. Since there are only $m = 10$ total mutations in Simulations (I-a), we did not allow the number of mutations in any patient to reach too close to 10. Thus, we set the maximum number of mutations per patient to 7. Fig. S8A shows this distribution of the number of mutations per patient (where all values above 7 are set to 7). Fig. S8B shows that CloMu performance is similar to the original Simulation (I-a) instances with $n = 100$ patients.

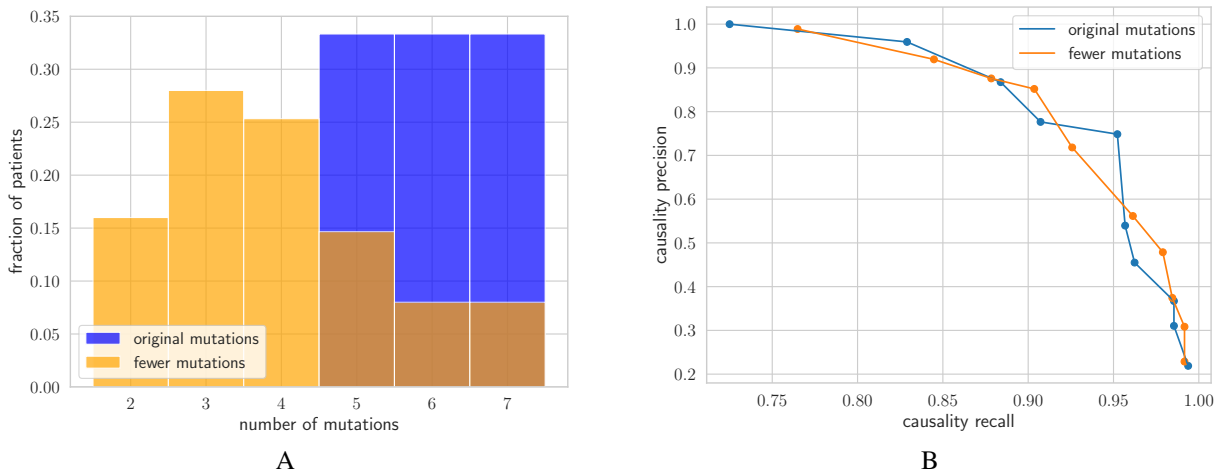


Figure S8: **CloMu's performance remains high for simulations with a reduced number of mutations per patient.** We assessed the effect of a reduced number of mutations per patient for simulations with $n = 100$ patients. (A) The original (blue) and modified distributions (orange); the latter are based on the AML dataset (Morita et al., 2020). (B) The causality precision and recall remain high.

B.3.4 Causation strength based on real data – adaptation of Simulations (I-a)

In Simulations (I-a), the strength of the absolute causality was set to $\log(11)$ such that the rate of a mutation is increased by a factor of 11 if it is caused by another mutation. However, in real cancer evolution, the strength of all causal relationships are not identical. To improve realism, we determined a distribution of the strength of causal relationships using the real data. On the AML dataset (Morita et al., 2020), we considered the strongest causal relationship caused by each mutation as determined by CloMu. Then, we picked the top 5 mutations with the highest value for their strongest (absolute causality) causal relationship. For these 5 mutations, we observed the absolute causality value for their top 5 strongest causal relationships. This gives a total of 25 causal relationships from the AML dataset. A histogram of these values is shown in Fig. S9A. We generated 20 new simulation instances based on Simulations (I-a), but with causal relationship strengths sampled from this distribution and with the number n of patients reduced to 300. As shown in Fig. S9B, CloMu is still able to give accurate causal relationship predictions.

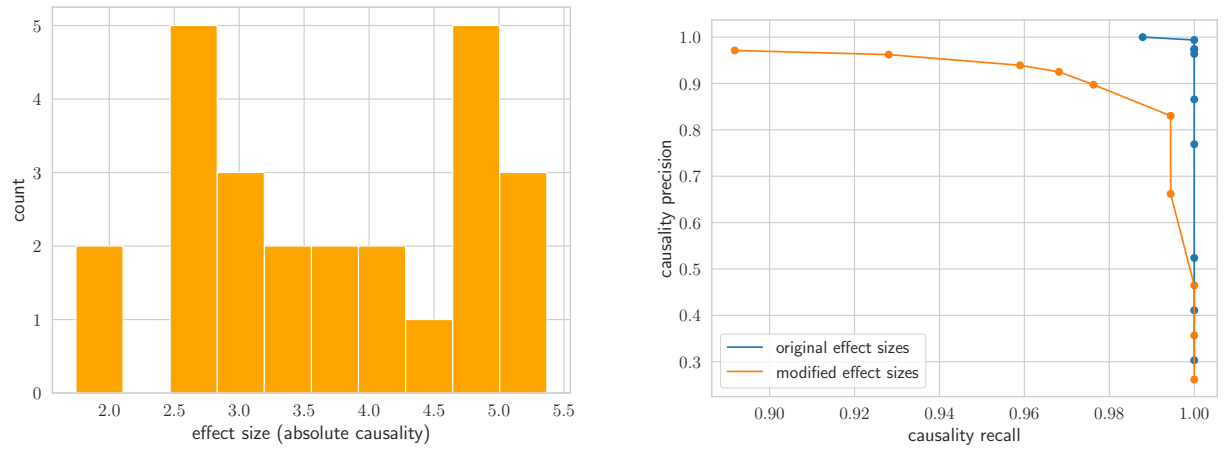


Figure S9: **CloMu's performance is slightly reduced when using varying effect sizes in causality.** We assessed the impact of varying effect sizes in causality using simulations with $n = 300$ patients. (A) The used distribution of effect sizes, again based on the AML dataset. (B) CloMu has a small reduction in causality precision and recall when the effect size is modified to match real data.

B.3.5 Large numbers of passenger mutations – adaptation of Simulations (I-a)

In this section, we introduce simulations with a large number of passenger mutations that do not affect the tumor. In all of the other simulations CloMu was tested on, the total number of mutations is at most 20. However, in real cancer data, there is often a much larger number of mutations, of which the vast majority are passengers. Fig. S10A shows a histogram of the fraction of $n = 1250$ breast cancer patients (with at most 10 mutations) for which each of the $m = 410$ mutations occurs (Razavi et al., 2018). We generated simulations that are identical to Simulation (I-a), but with the same number of mutations as the breast cancer dataset and with each mutation occurring with a baseline rate proportional to the frequency at which they occur in the breast cancer set. Additionally, we set the 5 driver mutations to be the 5 highest baseline likelihood mutations and set all other mutations as passengers. Fig. S10B shows the causality precision and recall of this simulation. The accuracy was somewhat reduced by the high number of passenger mutations but remains acceptable. The primary reason for the reduction in accuracy is the difficulty of differentiating the tiny number of positive relationships from the vast number of pairs of mutations with no causal relationships. Specifically, there are at most 20 true causal relationships and at least $410^2 - 410 - 20 = 167,670$ non-causal relationships. If non-causal relationships were falsely identified as causal in even $1/1000$ of cases, there would be over 8 times as many false positives as true positives (resulting in a precision below 0.11).

In addition to testing how the number of patients affects the computational requirements as discussed

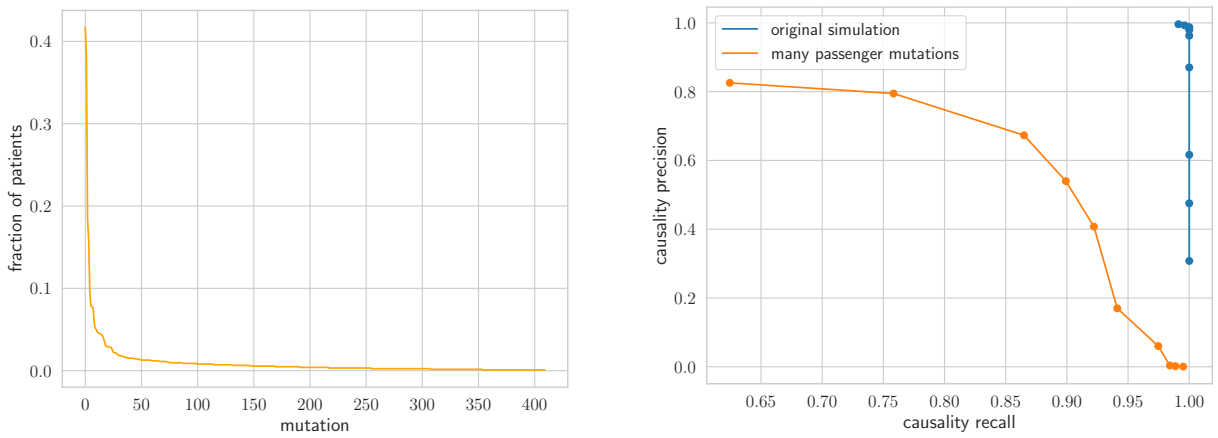


Figure S10: **CloMu's performance is reduced when including a large number of passenger mutations.** We assessed the impact of including a large set of passenger mutations among $n = 500$ patients. (A) The frequency of different mutations in the breast cancer dataset (Razavi et al., 2018) that we used to determine the default rate at which mutations occur (noting that the rates can then be modified by the presence of driver mutations). (B) CloMu has somewhat reduced accuracy when 405 passenger mutations are included. However, the performance is still acceptable, given that even a 99.9% accuracy at identifying non-causal relationships would give a precision below 0.11.

in Supplementary Material B.3.2, we recorded the effect of the number of mutations. Specifically, for the above set of simulations discussed in this section, we dramatically increased the number of mutations from 10 to 410. The new runtime for a single instance is 3 hours and 5 minutes, and the new memory usage is 5.378 GB compared to 30 minutes and 4.289 GB for $n = 300$ patients and $m = 10$ mutations.

B.3.6 Violations of independent clonal evolution – adaptation of Simulations (I-a)

CloMu is built off of the independent clonal evolution assumption that only the mutations on a clone will affect the probability of new mutations occurring on that clone. This assumption may be generally realistic but could have exceptions in real tumor evolution. Therefore, ideally, we would want CloMu to be somewhat robust to this possibility. We tested this by using a simulation where the causal relationships between mutations also occur to a smaller extent between mutations in different clones. In other words, if mutation s being in a clone causes mutation t to be much more likely to occur in that clone, then mutation s being in any clone will cause mutation t to be somewhat more likely to occur in any, not necessarily the same, clone.

We found that CloMu still performs well on a causal relationship simulation when clonal interactions are included in the simulation. Specifically, our 20 simulation instances result from a modified version of Simulations (I-a) with no interchangeable mutations, 5 driver mutations, and 5 passenger mutations.

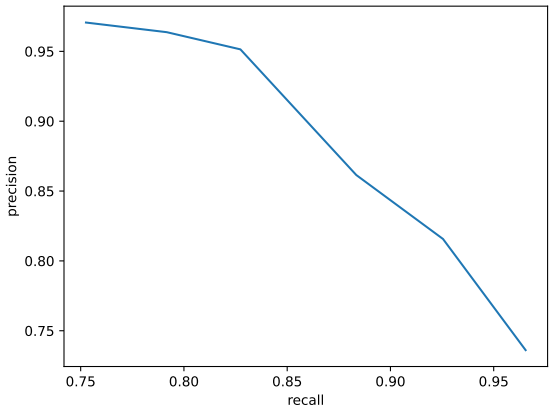


Figure S11: **CloMu is robust to violations of the independent clonal evolution assumption**. We consider a modification of Simulations (I-a) that results in a violation of the assumption. The resulting causality precision and recall values are slightly reduced but remain very high, as for the original Simulations (I-a) results shown in Main Text Fig. 2B).

If mutation s causes mutation t, then as a higher proportion of clones have mutation s, the probability of mutation t on all clones will increase. Specifically, if a proportion p of clones have some mutation s, then the log rate of mutation t will be increased by $p \log(2)$. Applying CloMu on this simulation gives a causality precision-recall curve shown in Fig. S11, with values close to 1, similar to the original Simulations (I-a) results shown in Main Text Fig. 2B). Hence, for this scenario, CloMu was robust to violations of the independent clonal evolution.

B.3.7 Including highly incorrect possible trees – adaptation of Simulations (I-a)

One important ability of CloMu is accounting for the fact that the set of possible trees for each patient can contain completely incorrect trees. In all of our simulations, we simulate bulk sequencing to add tree uncertainty. However, here we explicitly test CloMu's ability to deal with entirely random incorrect trees being included in the set of possible trees. We generate a set of simulations with the same setup as Simulations (I-a), with the exact same procedure for generating the true tree for each patient. However, now the set of possible trees for each patient contains both the true tree and a randomly generated tree with the same set of mutations as the true tree. The random tree is generated by starting with a normal clone, and then iteratively adding mutations (from the true tree) to clones in order to form a tree. Mutations are added to clones uniformly at random, with no causal relationships. As a more extreme case, we also generated 20 simulation instances in which there exists 2 random trees per patient in addition to the one true tree. Fig. S12

shows the precision recall curve on these new sets of simulations. In the case of one random tree per patient, the accuracy is only slightly reduced by half of the trees being random. In the case of two random trees per patient, the accuracy is noticeably reduced by the majority of trees being random. We also tested CloMu's tree selection capability. In the case with one random tree per patient, CloMu achieved an average tree selection accuracy of 81.82% across 20 simulation instances. In the case with two random trees per patient, CloMu's average tree selection accuracy was 57.50 % (in comparison to 33.33% that would be achieved by random selection). To investigate tree selection further, we define the *relative probability of a candidate tree* for a patient as the probability of the candidate tree divided by the sum of probabilities of all candidate trees for the patient. Fig. S13A shows a histogram of the relative probability of the correct tree and incorrect tree on all 20 simulation instances with one random tree per patient. Note that a relative probability above 0.5 indicates the correct tree would be selected. Fig. S13B shows a histogram of the relative probability of the correct tree and one incorrect tree on all 20 simulation instances with two random trees per patient Note that, the average relative probability of all possible trees is $1/3$ in this case, since there are now 3 possible trees per patient.

As a control case, we also generated simulations in which both trees are randomly generated. In this case, no causal relationships are predicted for the cutoff value $\tau = 0.8$, which allows for perfect 100% recall on the previously described simulations with one random tree per patient (and one true tree generated as in Simulations (I-a)). Additionally, out of 1800 possible causal relationships on 20 simulation instances, only 5 causal relationships exceed an absolute causality of 0.4. This cutoff value of 0.4 also allows for precision and recall values of 0.9515 and 0.9121, respectively, on the simulations where each patient has two random trees and one correct tree generated as in Simulations (I-a). The few false positive weak causal relationships that occur may be caused by weak patterns that occur in the trees by random chance. A histogram of the relative probability of the (random) correct tree is shown in Fig. S13C. This is much more uniform since the correct tree and non-correct tree are both drawn from the same random distribution. However, note that the process of randomly adding mutations to clones does not generate all trees with equal probability. In particular, branching topologies are generated by this random process with a higher likelihood than chain-like trees (since there are more unique ways of generating branching trees). Consequently, our model, trained on this random process, does not assign all trees exactly equal probability. Despite this, the probabilities of different trees are much more similar to each other than under a model of evolution in which there exist causal relationships.

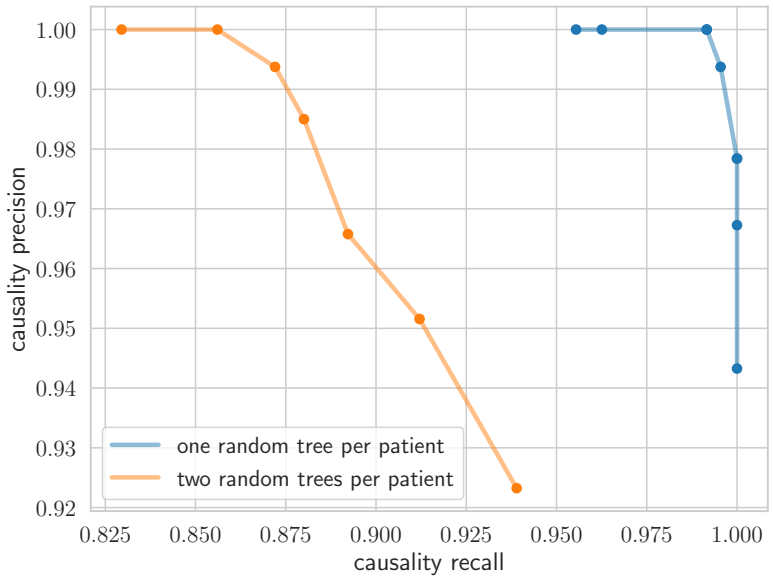


Figure S12: **CloMu is robust to random trees in the set of possible trees**. In these simulations based on Simulations (I-a), each patient contains one true tree and either one or two trees with a completely random topology on the same set of mutations as the true tree. When half of the trees are random (one random tree per patient, blue), the precision and recall remain very high. However, when the majority of the trees are random (two random trees per patient, orange), CloMu's accuracy is noticeably diminished.

These simulations demonstrate the robustness of CloMu on datasets where the set of possible trees is derived in a manner that could commonly include highly incorrect trees.

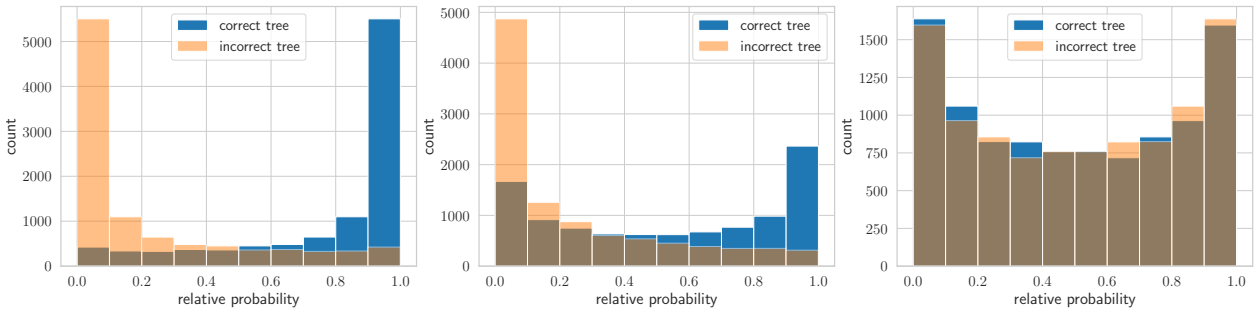


Figure S13: **CloMu assigns reasonable relative tree probabilities on simulations with random trees included**. The relative probability of correct and incorrect trees on all 20 simulation instances is shown in histograms for several simulation setups. (A) In this set of simulations, the set of possible trees for each patient contains one true tree generated as in Simulations (I-a) and one random tree. There is frequently a much higher probability assigned to the true tree than to the random tree. (B) In these simulations, the set of possible trees for each patient contains one true tree and two random trees. The probability of the true tree and one random tree is shown (since the two random trees are generated identically). The true tree typically has a higher probability than the random tree. (C) As a control case, simulations are created in which both the correct tree and incorrect tree are random. In this case, both random trees typically have a fairly similar probability.

B.3.8 Inhibition between interchangeable mutations – adaptation of Simulations (I-b,c)

Interchangeable mutations that belong to the same functional pathway have been observed to exhibit mutual exclusivity and thus inhibit each other's occurrence (Yeang et al., 2008). This effect was not assumed to be the case in Simulations (I-b) and (I-c). We generated new simulations exactly mimicking Simulations (I-b) and (I-c), but with all pairs of interchangeable mutations having negative causal relationships with each other. The performance of TreeMHN slightly improved, but the general pattern in the performance of CloMu and TreeMHN remains unchanged as shown in Main Text Fig. 2C and Fig. S14. Specifically, CloMu outperforms TreeMHN in the presence of two interchangeable mutations in each set of interchangeable mutations. Additionally, CloMu vastly outperforms TreeMHN in the presence of three interchangeable mutations in each set of interchangeable mutations.

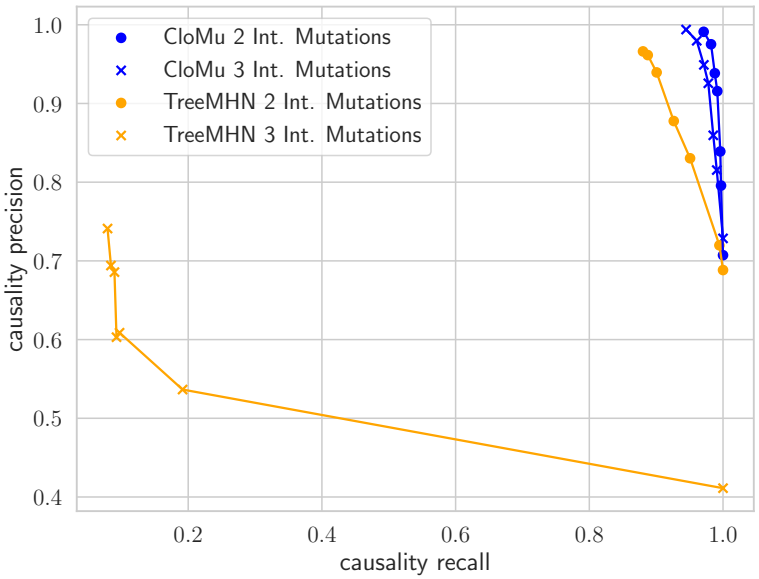


Figure S14: **Precision recall curves for simulations based on Simulations (I-b) and (I-c) still demonstrate the advantage of CloMu when inhibition between interchangeable mutations is added.** Simulations with 2 interchangeable mutations in each set of interchangeable mutations and 3 interchangeable mutations in each set of interchangeable mutations are shown. The addition of inhibition somewhat improved the performance of TreeMHN. However, CloMu still outperforms this baseline method in the presence of two interchangeable mutations in each set and greatly outperforms it in the presence of three interchangeable mutations in each set.

B.3.9 CloMu with increased number L of hidden neurons – Simulations (I-c)

CloMu's usage of a neural network with $L = 5$ hidden neurons allows it to have less parameters than existing methods and thus resistance to overfitting. The exact sensitivity of CloMu to the value of L is therefore worth investigating. In theory, increasing L creates the potential for overfitting. However, our use of L1 regularization may result in only a subset of hidden neurons being used, thus avoiding overfitting. For instance, on our breast cancer cohort and AML cohort, only a small number of hidden neurons were used as shown in our latent representation plots (Fig. S15). To test this, we applied CloMu with $L = 20$ hidden neurons to our Simulations (I-c) in which there are 3 interchangeable mutations per type and 5 types of mutations ($m = 15$). As $L = 20 > m = 15$, CloMu technically has more parameters than TreeMHN in this case – i.e., $O(Lm)$ for CloMu vs. $O(m^2)$ for TreeMHN. However, due to CloMu's neural network structure, only a small number of CloMu's parameters need to be used to fit the data. Fig. S15 demonstrates that CloMu with $L = 20$ performs vastly outperforms TreeMHN.

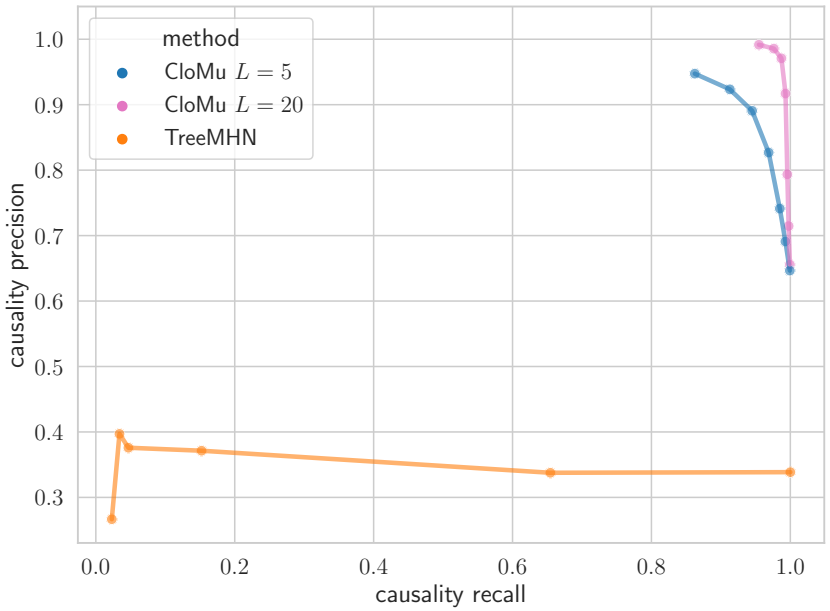


Figure S15: **CloMu is robust to increasing the number L of hidden neurons**. CloMu is ran on Simulations (I-c) with $L = 20$ as well as the original $L = 5$ hidden neurons. CloMu greatly outperforms TreeMHN for both values of L, demonstrating CloMu's robustness to differing numbers of hidden neurons.

B.3.10 Mutations with shared properties

On Simulations (II) it was demonstrated that interchangeable mutations can be detected by determining mutations with similar latent representations. However, in real cancer evolution, two mutations may have similar or related effects on tumors without being exactly interchangeable. We generated an additional set of simulations to demonstrate the ability of our latent representations to pick up on these shared properties of mutations. Additionally, on Simulations (I-b) and (I-c) it was demonstrated that CloMu predicts causal relationships more accurately than TreeMHN in the presence of interchangeable mutations. The simulations generated in this section also demonstrate CloMu's ability to more accurately predict causal relationships in the presence of mutations with shared properties.

In these simulations, there exist $m = 20$ mutations. Each mutation has two values associated with it called the value of property 1 and the value of property 2. These values range from -1 to 1. Generally speaking, mutations with similar values of the two properties tend to cause each other. One can imagine that there exists some capability the tumor must develop in order to grow effectively (such as avoiding being stopped by the immune system) and that there are multiple independent and mutually exclusive methods of accomplishing this strategy. If property 1 represents some required feature of the tumor, then high values

and low values of property 1 indicate mutations that develop the tumor towards independent and mutually exclusive strategies for accomplishing this feature. This results in inhibition between mutations with opposite values of property 1 and causation between mutations with similar values of property 1. The same is true for property 2. Define $P_L(s)$ to be the 2-dimensional vector containing the values of property 1 and 2 for some mutation s. If mutation s exists in a clone, then the rate of mutation t is multiplied by a factor of 10 to the power of $P_L(s) \cdot P_L(t)/\|P_L(t)\|_2$. Equivalently, the value $\log(10)P_L(s) \cdot P_L(t)/\|P_L(t)$ is added to the log rate. As in previous simulations, the rate of mutation t occurring on a clone $\mathbf{c}$ is simply the product of these rate multipliers and 0 if $\mathbf{c}$ already contains t.

For each mutation, we generate $P_L(s)$ by first generating a random ϕ between 0 and 2π, then generating a random R between $1/2$ and 1, and then setting $P_L(s) = [R\cos(\phi), R\sin(\phi)]^\top$. This simulation setup is intentionally somewhat more complex than our other simulations for the purpose of creating a complex landscape of related mutations with partially shared yet partially differing effects. We refer to Fig. S16 for a worked example.

After training the model, one goal is for similar mutations (with similar values of $P_L(s)$) to have similar latent representations. Fig. S17A shows a scatter plot of the Euclidean distance between latent representations and the distance between mutation property vectors ($P_L(s)$). The Pearson correlation between these two distance measurements is 0.8316, showing high concordance. Additionally, we wish to reconstruct the properties $P_L(s)$ of a mutation s exclusively through observing the latent representation. This more ambitious goal requires some post-processing to be possible. Specifically, we normalized $P_L(s)$ to have a mean of zero and a norm of 1 across all mutations s. Then we applied PCA to the latent representations of all mutations, reducing the dimension to 2 and normalized this representation to also have an average norm of 1. This allows $P_L(s)$ and the latent representations to be of the same dimension and on the same scale. Then, these two-dimensional processed latent representations are rotated by some angle and/or flipped in order to reduce the Euclidean distance between these representations and P_L. This only requires two parameters (a rotation angle and a Boolean of possibly flipping the first component) shared across all 20 mutations. This step is necessary since there is no reason to assume by default that the first PCA component of the latent representation corresponds to the first mutation property in $P_L(s)$. We define this processed version of the latent representation to be the predicted mutation property.

We compare the predicted mutation property derived from the latent representation vector with the true property vector. Fig. S17B shows CloMu gives highly accurate predictions on an example simulation.

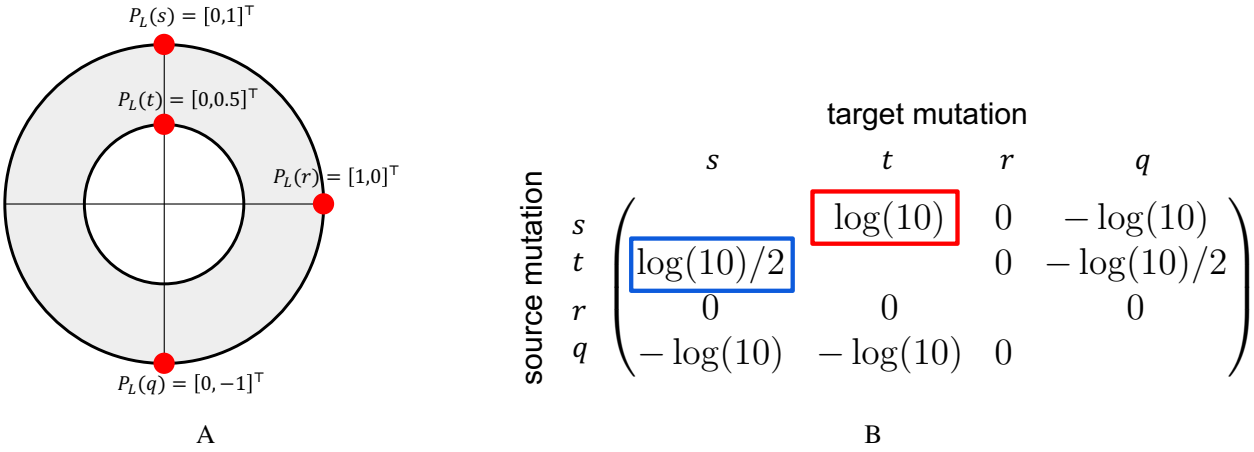
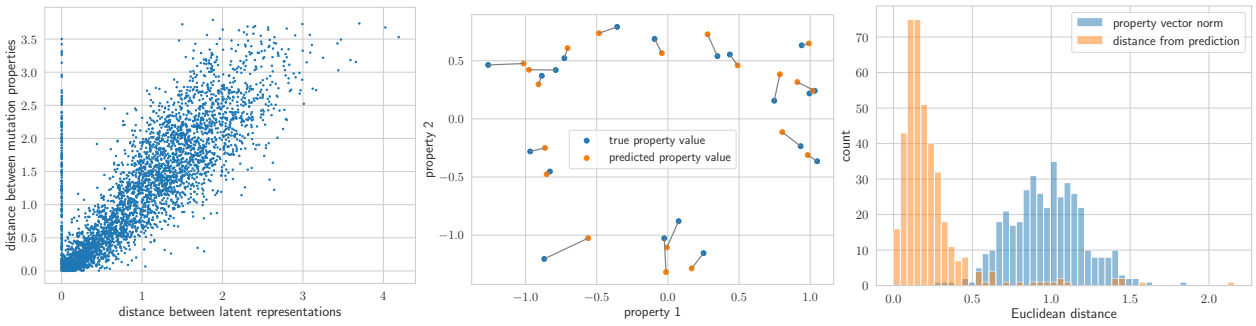


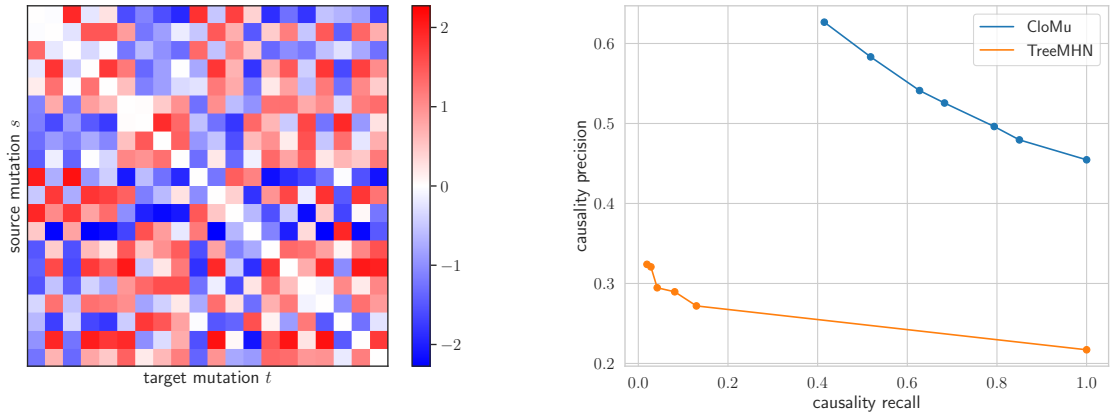
Figure S16: **A demonstration of mutations with shared properties.** (A) Hypothetical mutations with their property value vectors are shown. The area in grey shows possible mutation property vectors. (B) The log rates of each new mutation on each clone with a single other mutation are shown. The log rate $f(\mathbf{c}_s, t) = \log(10)$ is circled in red and demonstrates the strongest possible positive causal relationship. This log rate is calculated as $f(\mathbf{c}_s, t) = \log(10)[0,1]^\top \cdot [0,0.5]^\top / |[0,0.5]|_2 = \log(10)0.5/0.5 = \log(10)$. The log rate $f(\mathbf{c}_t, s) = \log(10)/2$ is circled in blue and demonstrates how our causal relationships are not symmetric. This log rate is calculated as $f(\mathbf{c}_t, s) = \log(10)[0,0.5]^\top \cdot [0,1]^\top / |[0,1]|_2 = \log(10)0.5/1 = \log(10)/2$.

Fig. S17C shows CloMu generally reconstructs the mutation properties well, evaluated on all simulation instances. We also applied CloMu to predict causal relationships. In these simulation instances, causal relationships can be arbitrarily weak, which increases the difficulty of predicting causal relationships. Fig. S18A shows an example of true causal relationships for one simulation instance. Fig. S18B shows a precision recall curve for CloMu and TreeMHN. This demonstrates that CloMu's advantage in simulations with interchangeable mutations also applies to simulations with shared mutation properties in general. In contrast, Simulations (IV) show that CloMu does not have an advantage over TreeMHN on simulations with 20 mutations but random independent linear causal relationships.



A B C

Figure S17: **CloMu accurately determines similar mutations and reconstructs mutation properties**. (A) Similar mutations (with close property vectors) tend to have close latent representations. A scatter plot of Euclidean distances between all pairs of mutations in all simulations is shown, where the x-axis shows distances between latent representations and the y-axis shows distances between mutation property vectors. The Pearson correlation is 0.8316. (B) Predicted property vectors are reconstructed from the latent representations. A plot of these true property vectors and predicted property vectors is shown for one simulation instance. Grey lines connect the predicted and true property values for each mutation. (C) The Euclidean distance between predicted property vectors and true property vectors is plotted in orange. This is compared against a baseline of predicting the mean normalized property vector of zero for all mutations. The error of this baseline is exactly equal to the norm of the property vector by definition.



A B

Figure S18: **CloMu accurately predicts causal relationships on simulations with shared mutation properties**. (A) A plot of causal relationships for one simulation instance. (B) CloMu's advantage over existing methods on simulations with interchangeable mutations also applies to simulations where mutations share properties.

B.3.11 Isolating multi-mutation effects

One advantage of CloMu is that it allows for complex interactions between mutations while utilizing a small number of parameters. The ability of CloMu to detect the effects of combinations of mutations was demonstrated in Simulations (II). However, Simulations (II) also introduced other concepts such as interchangeability and pathways. We generated an additional set of simulation instances to specifically demonstrate the combination of two mutations affecting the likelihood of other mutations. These simulations contain two driver mutations and eight passenger mutations, totaling $m = 10$ mutations. The combination of the two driver mutations together in a clone affects the rate of all eight passenger mutations. For each of the eight passenger mutations, there is a 0.5 probability of being inhibited and a 0.5 probability of being caused by the combination of the two driver mutations. Inhibited mutations have their rate reduced by a factor of 10, and positively caused mutations have their rate increased by a factor of 10. There is no effect of any individual mutation and the driver mutations only impact the clone when they occur together. The baseline rate of the driver mutations is 5 times the rate of the passenger mutations. Each simulation instance contains $n = 500$ patients, and 20 simulation instances were generated.

Given the definitions of multi-mutation causality outlined in Supplemental Material A.2.2, we predict causal relationships from pairs of mutations to third mutations. The precision-recall curve is shown in Fig. S19A, which uses the definition of precision and recall for signed causal relationships stated in Supplemental Material B.3. The average F1 score across simulation instances reaches 0.9286 for the optimal threshold value of τ.

Without additional analysis, it is theoretically possible that CloMu is only picking up on causal relationships from pairs in a linear manner by attributing half of the effect to each individual mutation. For instance, if s and t cause r it is possible CloMu is falsely concluding that s and t both individually cause r each with half the total effect size. To disprove this, we compare the predicted causal effect of a pair of mutations with the sum of the individual effects of the two mutations in the pair. Specifically, we define the *linear predicted causality* from s and t to r as $f_\theta(\mathbf{c}_s, r) + f_\theta(\mathbf{c}_t, r) - f_\theta(\mathbf{c}_0, r)$. In Fig. S19 we compare this with the predicted multi-mutation causality $f_\theta(\mathbf{c}, r)$ where $\mathbf{c}$ contains exactly mutations s and t. Specifically, Fig. S19B shows this for positive causal relationships and Fig. S19C shows this for negative inhibitory relationships. This demonstrates that CloMu is successfully determining non-linear effects.

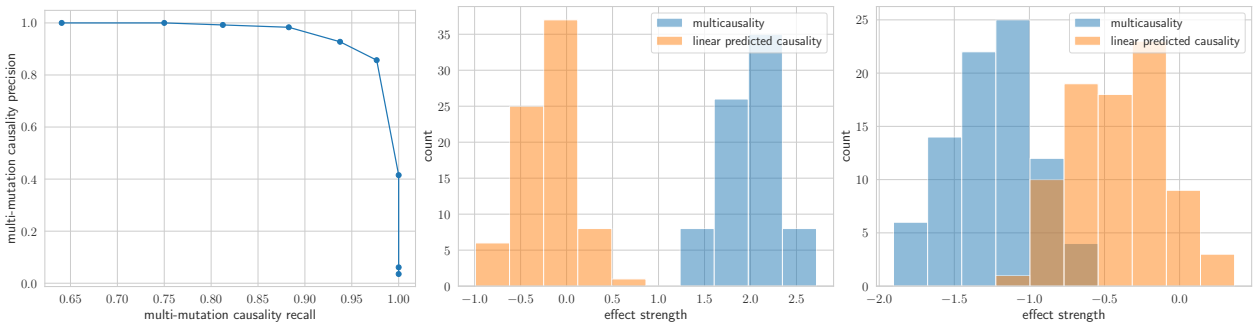


Figure S19: **CloMu detects non-linear multi-mutation causality caused by pairs of mutations** (A) A precision recall curve of CloMu detecting multicausality. The average F1 score across simulation instances reaches 0.9286 for the optimal threshold value of τ. (B) For positive relationships, CloMu predicts multi-mutation causality from a pair of mutations beyond the causality from each individual mutation in the pair. (C) For negative relationships, CloMu predicts negative multi-mutation causality from a pair of mutations beyond the causality from each individual mutation in the pair.

B.3.12 Results on RECAP simulations – Simulations (III)

We compared CloMu to RECAP as well as REVOLVER. As in the Main Text, we consider the tree selection accuracy, which is the fraction of patients for which the correct true tree is predicted. Fig. S20 shows that CloMu performed almost identically to RECAP and far outperforms REVOLVER. In fact, our method and RECAP both achieved a 98.6% accuracy on average for simulations with 5 mutations and an average 100% accuracy for 12 mutations (Fig. S20A,C). We additionally investigated the predicted number of patient clusters for our method and RECAP, finding perfect performance for CloMu and near perfect performance for RECAP across varying numbers of mutations (Fig. S20B,D).

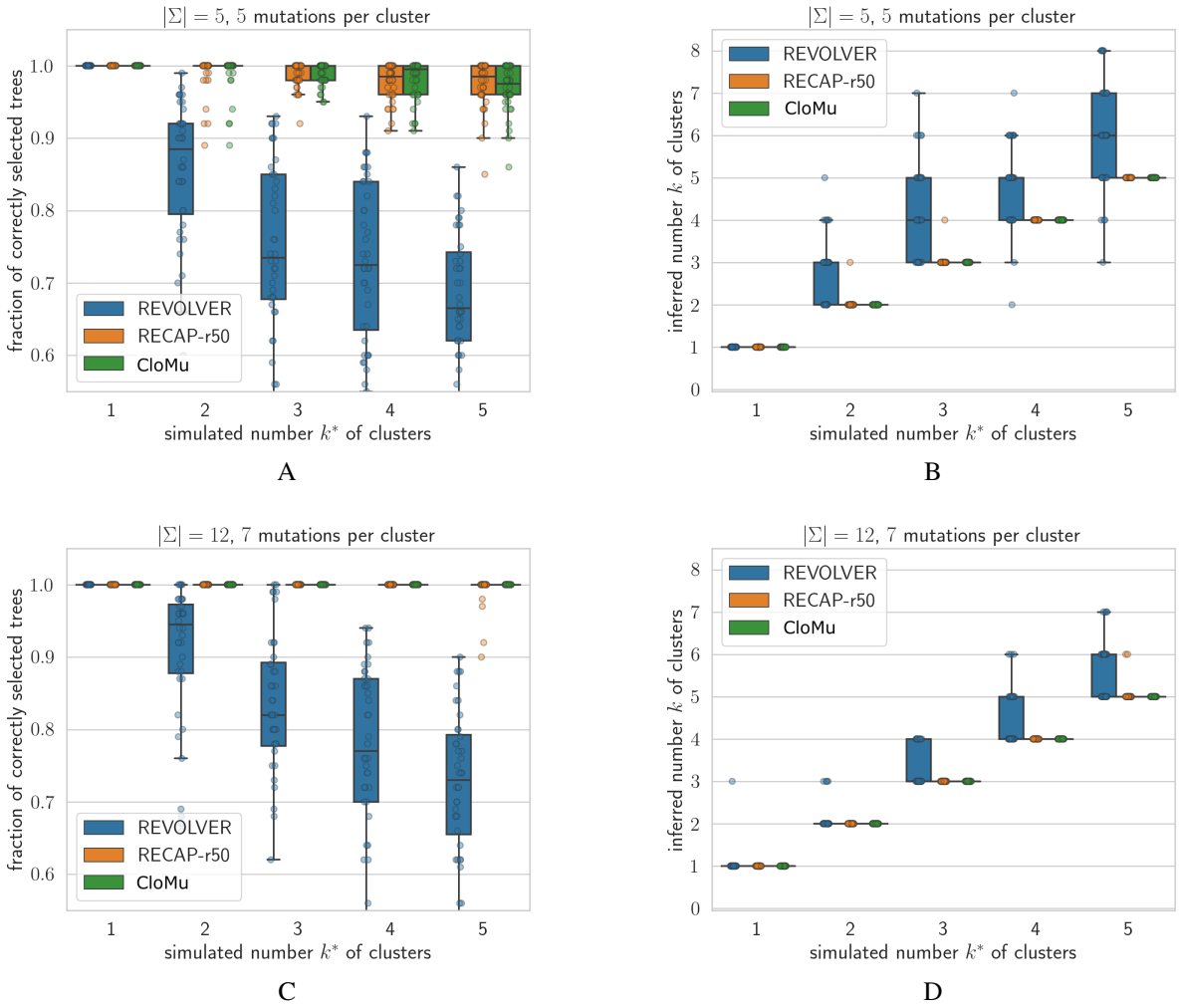


Figure S20: **CloMu's matches RECAP's performance on RECAP's simulated data (Simulations (III)).** These plots show RECAP results with 50 restarts and additionally include REVOLVER results reported in Christensen et al. (2020). (A,C) Tree selection accuracy. (B,D) The number of ground-truth patient clusters vs. the predicted number of patient clusters. Panels (A-B) show results for $|\Sigma| = 5$ total mutations and $m = 5$ mutations per patient whereas panels (C-D) show results for $|\Sigma| = 12$ total mutations and $m = 5$ mutations per patient.

B.3.13 Results on TreeMHN simulations – Simulations (IV)

CloMu is generally competitive with but slightly less effective than TreeMHN on datasets from the TreeMHN paper. Datasets in the TreeMHN paper use completely independent causal relationships between all mutations, with no related or interchangeable mutations. In this case, using CloMu with a neural network function creates a slight disadvantage on datasets with a very poor signal-to-noise ratio. Therefore, we included a version of CloMu where the neural network function is replaced with a simple linear model, denoted as "Linear CloMu". One virtue of the training setup of CloMu is that it is completely independent of the function chosen for f_θ. Therefore, this only requires a minor tweak to the code. We compared CloMu to TreeMHN with stability selection in the cases of $n = 300$ patients, and $m \in \{10, 15, 20\}$ mutations. The TreeMHN setup and hyperparameters were set exactly as in the TreeMHN paper. Like in the TreeMHN paper, we plot the precision and recall curves. Unlike the TreeMHN paper, we report the precision and recall on all mutations, not only the top half most frequent mutations.

Fig. S21 shows the precision and recall curves averaged over the 20 simulations per simulation setup. One can see that TreeMHN outperformed CloMu on these data, but CloMu remained competitive especially when a linear function is used for f_θ. For instance, it seems CloMu with a linear model may extend to higher recall values more effectively than TreeMHN, but this is unclear since the range of hyperparameters used in the TreeMHN paper does not extend into high recall. From these results, we conclude the following. If one knows for certain that there are no shared properties between mutations and that all causal relationships are independent, then TreeMHN is a slightly more effective method than CloMu. However, if there exists shared properties between mutations, CloMu ranges from slightly more effective to vastly more effective than TreeMHN.

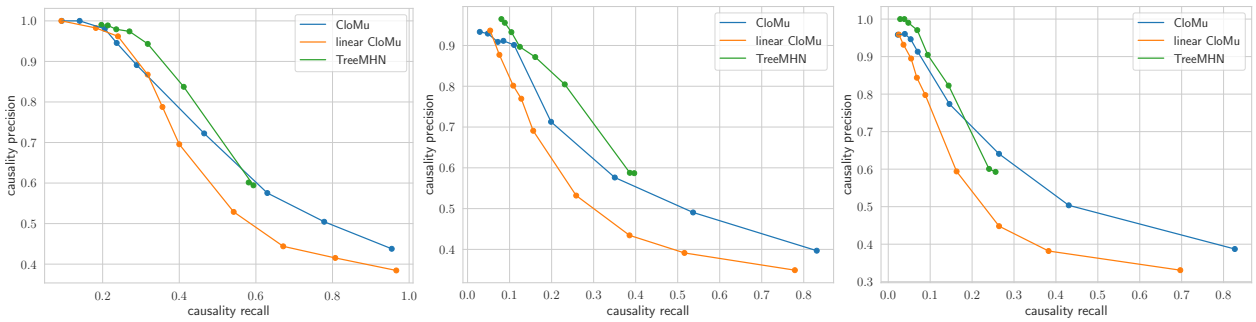


Figure S21: **CloMu performs slightly worse than TreeMHN on TreeMHN's simulated data (Simulations (IV).** (A) Simulations with $m = 10$ total mutations. (B) Simulations with $m = 15$ mutations. (C) Simulations with $m = 20$ total mutations.

C Supplemental results for real data

In Supplemental Material C.1 we describe additional analyses on the breast cancer dataset (Razavi et al., 2018). In Supplemental Material C.2 we describe additional analyses on the AML dataset (Morita et al., 2020).

C.1 Breast cancer

CloMu was ran on the breast cancer dataset (Razavi et al., 2018) using a laptop with a 2.4 GHz CPU and 64 GB of RAM without using a GPU. The runtime was 4 hours and 23 minutes, and the memory usage was 10.777 GB. The method of bootstrapping to demonstrate statistical bounds on predictions is demonstrated in this dataset in Supplemental Material C.1.1. The connection between latent representations and breast cancer hormone receptor status is shown in Supplemental Material C.1.2. Relative causality predictions are systematically compared to TreeMHN's predictions in Supplemental Material C.1.3. CloMu's ability to predict the next mutation to occur on subtrees is demonstrated in Supplemental Material C.1.4.

C.1.1 Bootstrapping

Supplemental Material A.3 introduced bootstrapping for CloMu. We applied this procedure to the breast cancer dataset (Razavi et al., 2018), using 100 bootstrapping replicates. Fig. S22 shows box plots of relative causality predictions on the breast cancer data, showing that our predictions fall within the statistical bounds established by bootstrapping. Fig S23 shows box plots of the rankings of these relative causality predictions, again demonstrating the robustness of our predictions. Additionally, a 90% confidence interval of predicted

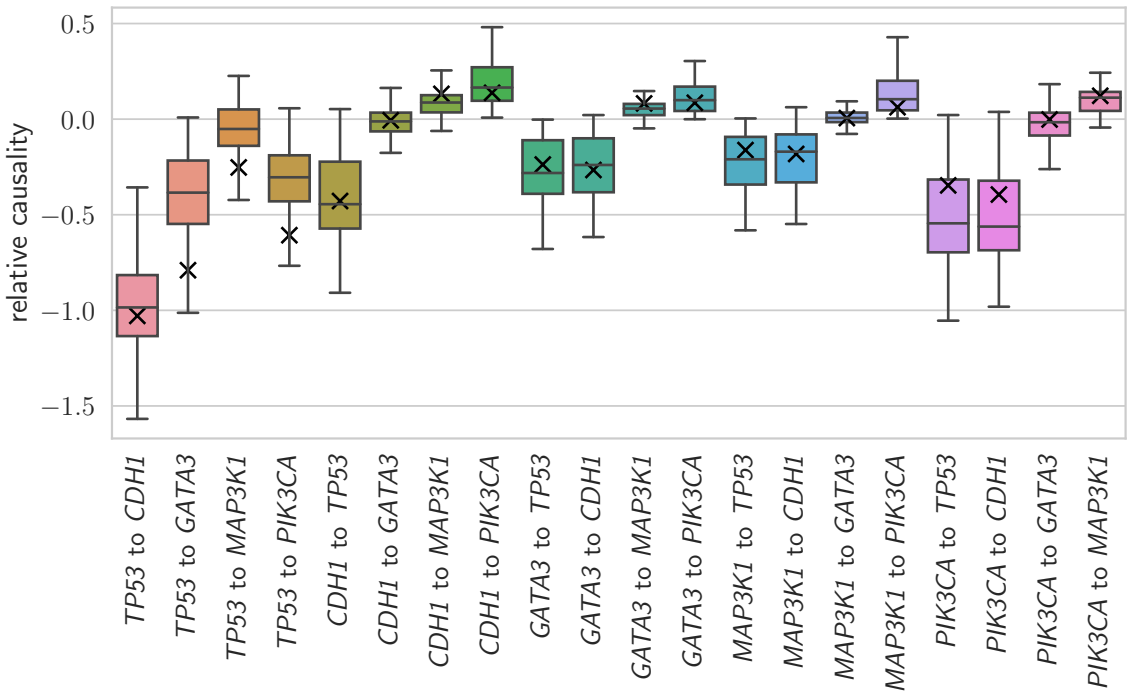


Figure S22: **Bootstrapping provides statistical bounds for relative causality predictions on the breast cancer dataset (Razavi et al., 2018)** . A box plot of bootstrapping predicted relative causality is shown. The value predicted on the original dataset is shown with a black 'X'. This demonstrates statistical bounds on our causal relationship predictions.

672 fitness values on bootstraps is shown in Fig. S24. The set of high-fitness mutations is very stable across

673 bootstrapping instances.

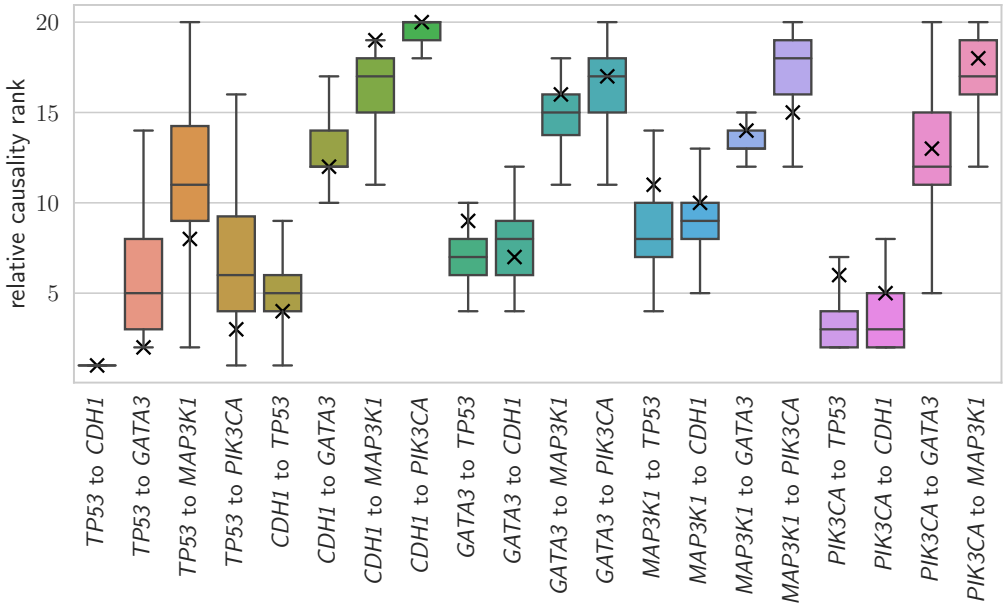


Figure S23: **Bootstrapping provides statistical bounds for the rankings of relative causality predictions on the breast cancer dataset (Razavi et al., 2018)** . The 20 causal relationships being analyzed are ranked from 1 to 20, where 20 indicates the most positive relationship and 1 indicates the most negative relationship. A box plot of bootstrapping predicted relative causality rankings is shown. The ranking predicted on the original dataset is shown with a black 'X'. The strongest positive and negative causal relationships are fairly consistent.

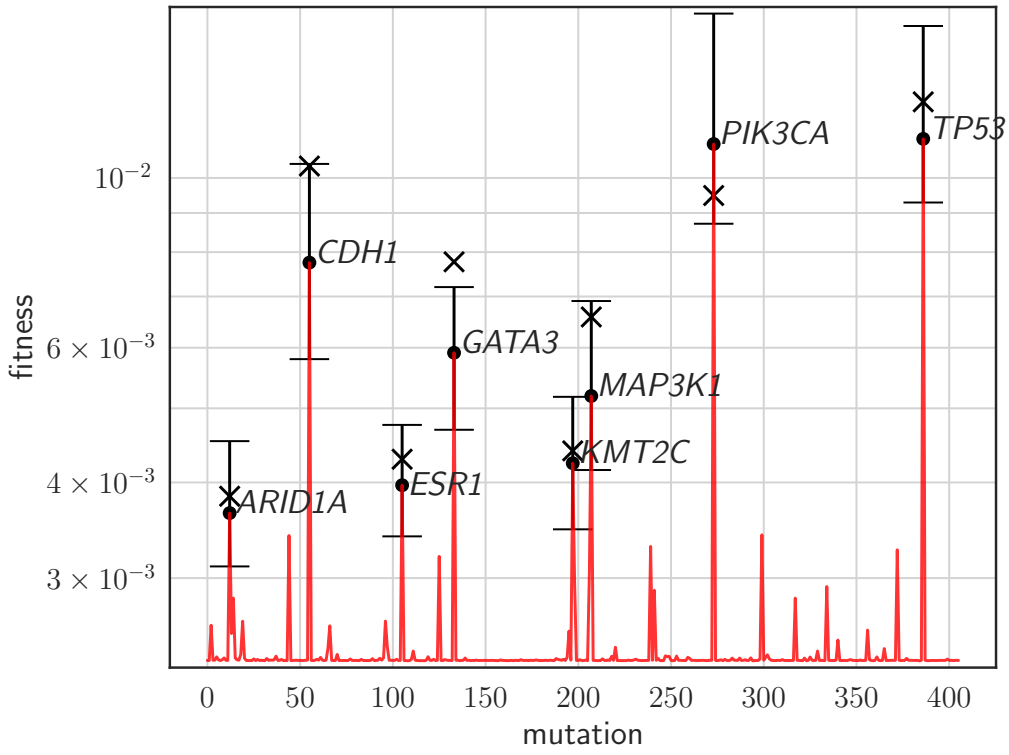
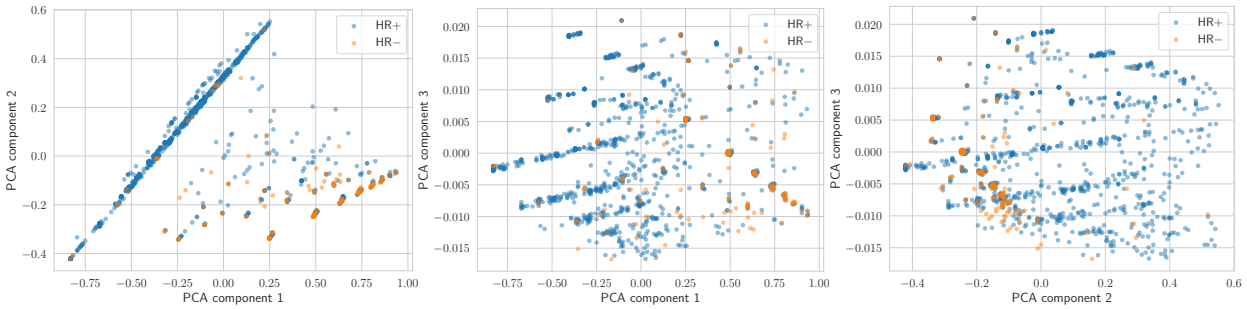


Figure S24: **Bootstrapping gives statistical bounds for fitness predictions on the breast cancer data set (Razavi et al., 2018).** The median fitness values for all mutations are shown in red, with the 90% confidence interval on 100 bootstrapped intervals shown in black. Specifically, this shows the 6th largest and 6th smallest fitness values, therefore excluding the 5 highest and lowest values. The fitness values predicted on the original dataset are shown with a black 'X'.

C.1.2 Breast cancer hormone receptor status and latent representations

The breast cancer dataset contains patients with different hormone receptor statuses, which include HR+/HER2+, HR+/HER2- and triple negative (Razavi et al., 2018). Specifically, there are 179 patients with HR+ and HER2+, 1501 patients with HR+ and HER2-, 62 patients with HR- and HER2+, and 176 patients that are triple negative. Within the subset of patients our model was run on (patients with at most 9 mutations), there are 113 patients with HR+ and HER2+, 943 patients with HR+ and HER2-, 39 patients with HR- and HER2+, and 129 patients that are triple negative. We investigated if different latent representations are associated with different hormone receptor statuses. To do so, we must first define a latent representation of a patient. The latent representation of a clone is directly produced by our neural network by inputting the vector representation of the clone. We define the latent representation of a tree as the mean representation of all clones in the tree. We define the latent representation of a patient as the mean latent representation of all possible trees for that patient (since the true tree is unknown). We found no connection between the latent representation of patients and the HER2+ hormone status, however, we did observe an association with the HR+ hormone status. Fig. S25 shows plots of the first three PCA components of the latent representations of patients with the HR+ and HR- hormone statuses,. We see that there tends to be separation between patients with the HR+ hormone status and those with the HR- hormone status, and that this is mainly driven by the first two components.

The main text contains plots showing the full latent representations of all mutations in the breast cancer dataset (Razavi et al., 2018). These plots have the advantage of showing all relevant information but the disadvantage of potentially being less visually intuitive. PCA allows for the dimension of the latent representation to be reduced, allowing for a scatter plot visualization. Additionally, we compare the latent representation with mutation fitness values. Fig. S26 shows these PCA values. The correlation between PCA component 1 and fitness is 0.8947 with a p-value less than 10^{-142}. The correlation between PCA component 2 and fitness is 0.4323 with a p-value less than 10^{-19}. There is no significant correlation between PCA component 3 and fitness (p-value of 0.7073). We find that, especially in the first two components, low fitness mutations cluster together, whereas high fitness mutations are at more distinct positions. Another interesting analysis of latent representations is the Euclidean distance between the latent representations of different mutations. In Fig. S27, we see there are large numbers of pairs with a near zero Euclidean distance apart, indicating interchangeable mutations. Many such mutations are low fitness mutations with a latent



A B C

Figure S25: **CloMu produces latent representations somewhat associated with hormone receptor status.** Scatter plots are shown of latent representations of breast cancer patients with hormone receptor status HR+ an HR-. There appears to be an association between latent representation and hormone receptor status. (A) A plot of PCA components 1 and 2. (B) A plot of PCA components 1 and 3. (C) A plot of PCA components 2 and 3.

representation near zero.

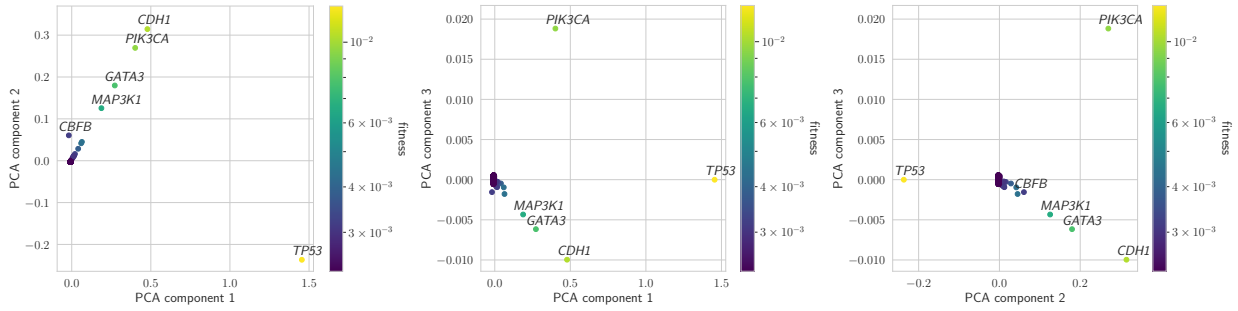


Figure S26: **PCA of latent representations on the breast cancer dataset (Razavi et al., 2018).** Mutations are colored based on their fitness. All mutations are shown in the scatter plot, however, low-fitness mutations tend to cluster near $(0, 0)$. High-fitness mutations are indicated with a distinct label. The correlation between PCA component 1 and fitness is 0.8947 with a p-value less than 10^{-142}. The correlation between PCA component 2 and fitness is 0.4323 with a p-value less than 10^{-19}. There is no significant correlation between PCA component 3 and fitness. (A) PCA components 1 and 2 are shown. (B) PCA components 1 and 3 are shown. (C) PCA components 2 and 3 are shown.

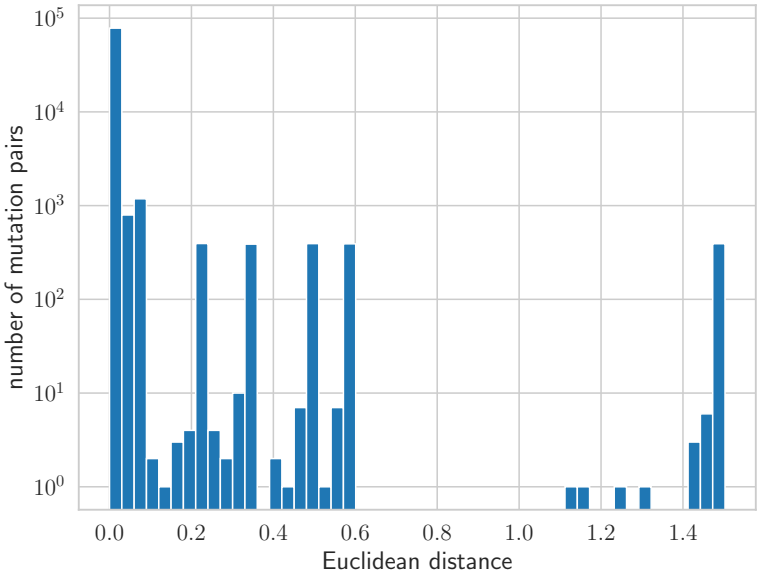


Figure S27: **Many mutations have highly similar latent representations in the breast cancer data.** A histogram of Euclidean distances between pairs of mutation latent representations on the breast cancer data set (Razavi et al., 2018) is shown. There is a clear spike around zero for interchangeable mutations (including low-fitness mutations).

C.1.3 Systematic comparison with TreeMHN

Unlike TreeMHN's causality measurement, our definition of relative causality explicitly removes increased mutation likelihoods due to high fitness. Consequently, CloMu should predict lower relative causality values from fit mutations when compared to TreeMHN's causality predictions. If TreeMHN predicts a negative causal relationship then CloMu should also theoretically predict a negative causal relationship but not vice versa. Similarly, if CloMu predicts a positive causal relationship then TreeMHN should also theoretically predict a positive causal relationship but not vice versa. To systemically compare CloMu and TreeMHN's causality predictions, we restricted our analysis to causal relationships between pairs of mutations with meaningfully nonzero latent representations when predicting causal relationships as previously described. On this restricted set of mutation pairs, we analyzed all ordered pairs of mutations in which either TreeMHN predicted a negative causal relationship or CloMu predicted a positive causal relationship. We found strong agreement between the causal relationship predictions of CloMu and TreeMHN on these mutations pairs.

All of TreeMHN's negative predictions on the breast cancer cohort were also predicted negative by CloMu (Table S3, four cases). Additionally, all of CloMu's positive predictions on the breast cancer cohort were also predicted positive by TreeMHN. However, as expected, there exists disagreements in which CloMu predicts a negative relationship and TreeMHN predicts a positive relationship. Specifically there are four such cases, all of which are from the extremely fit mutations *TP53* and *PIK3CA*.

Source mutation	Target mutation	TreeMHN prediction	CloMu prediction
GATA3	*TP53*	Negative	Negative
GATA3	*CDH1*	Negative	Negative
CDH1	*PIK3CA*	Positive	Positive
GATA3	*PIK3CA*	Positive	Positive
MAP3K1	*PIK3CA*	Positive	Positive
CDH1	*MAP3K1*	Positive	Positive
GATA3	*MAP3K1*	Positive	Positive
PIK3CA	*MAP3K1*	Positive	Positive

Table S3: On the breast cancer cohort, all negative causal relationships predicted by TreeMHN are also predicted by CloMu. Additionally, all positive causal relationships predicted by CloMu are also predicted by TreeMHN.

C.1.4 Predicting subsequent mutations on subtrees

Another way of evaluating performance on real data is observing subtrees of patients' trees and predicting which mutations will subsequently occur. Specifically, if there exists some tree T and some rooted subtree T' of T, methods could predict a new mutation s to occur on some clone in T'. To that end, we define $\text{Extend}(T', T)$ such that $(\mathbf{c}, s) \in \text{Extend}(T', T)$ if and only if $\mathbf{c}$ is a clone in T', and T includes mutation s being added to $\mathbf{c}$ to produce a new clone in T. More informally, $\text{Extend}(T', T)$ is the set of clones in T' in which additional mutations occur as part of T (together with those mutations).

Using CloMu's tree generative process, the probability that s is the next mutation to occur on T' is simply $\Pr((\mathbf{c}_i, s) \mid V(T'), f_\theta) = \Pr((\mathbf{c}_i, s) \mid \mathbf{c}_1, \ldots \mathbf{c}_k, f_\theta)$ where $V(T') = \{\mathbf{c}_1, \ldots \mathbf{c}_k\}$ is the set of clones in T'. An example of this calculation is shown in Fig. S28. As a minor caveat, on the breast cancer dataset (Razavi et al., 2018), the infinite sites assumption is made and so the probability of mutations that violate this assumption is set to zero (and the probability is normalized to sum to 1).

We define $\text{Next}_C(T', T)$ as the sum over all $(\mathbf{c}, s) \in \text{Extend}(T', T)$ of these predicted probabilities (of s occuring on $\mathbf{c}$ being the next mutation) according to CloMu. We define $\text{Next}_M(T', T)$ as the analogous computation being performed with TreeMHN's probabilistic model. Finally, we define $\text{Next}_B(T', T)$ as the analogous computation according to a baseline model in which mutations occur at a rate proportional to the percentage of patients in which they occur (independent of the clone they occur on). Specifically, $\text{Next}_B(T', T) = 1/|V(T')| \sum_{(\mathbf{c},s)\in\text{Extend}(T',T)} \text{freq}(s)$, where $\text{freq}(s)$ is the proportion of patients with mutation s and $|V(T')|$ is the number of clones in T'.

Our procedure consists of picking random subtrees of random patients and then evaluating these three models in terms $\text{Next}_C(T', T)$, $\text{Next}_M(T', T)$, and $\text{Next}_B(T', T)$. There are many ways of randomly

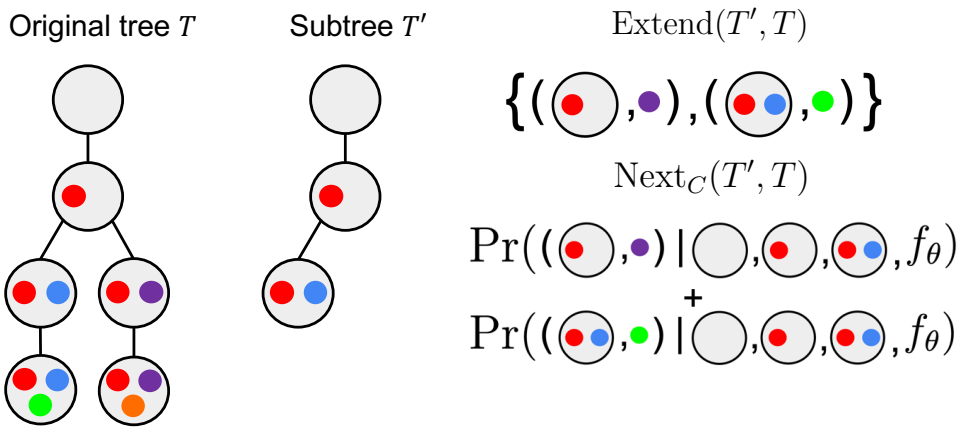
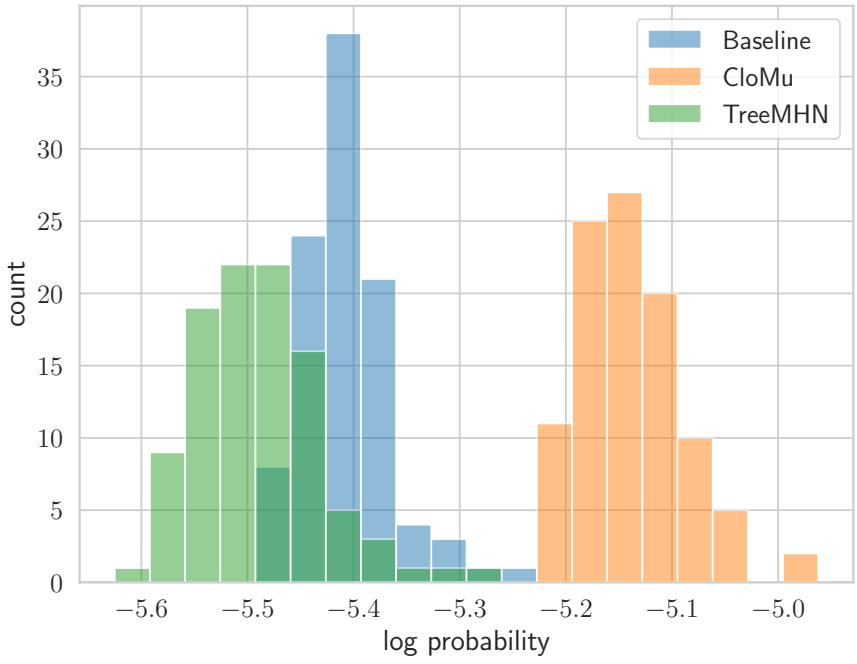


Figure S28: **A diagram of next mutation prediction on subtrees.** An original tree T as well as a subtree T' are shown. The set of extensions $\mathrm{Extend}(T', T)$ is shown. Additionally, the calculation of the probability $\mathrm{Next}_C(T', T)$ is shown as the sum of the probability of extensions in $\mathrm{Extend}(T', T)$ according to CloMu's probability estimates.

generating subtrees from a data set, so we must precisely define our random sampling procedure. First, we uniformly randomly sampled a patient and uniformly randomly sampled one of the possible trees from that patient. Then, we uniformly randomly sampled the size of the subtree in terms of vertices in the range of 1 to the number of vertices in the full tree minus 1. Then, we generated a random subtree by starting with the full tree and uniformly randomly removed leaves until the desired tree size is reached. Theoretically, we wish to maximize the product of probabilities (such as $\mathrm{Next}_C(T', T)$) across samples for a large number of sampled. This is equivalent to maximizing the average value of $\log(\mathrm{Next}_C(T', T))$ across samples. To avoid extreme outliers greatly biasing the data, we instead evaluate an average of $\log(\mathrm{Next}_C(T', T) + 0.001)$ across the data. Specifically, we generate 100 batches of 1000 samples and individual average each of $\mathrm{Next}_C(T', T)$, $\mathrm{Next}_M(T', T)$, and $\mathrm{Next}_B(T', T)$ within each batch. This averaging allows for a plot that is not dominated by noise (due to tumor evolution being a highly random probabilistic process). Histograms of these results on the breast cancer dataset (Razavi et al., 2018) are given in Fig. S29. CloMu greatly exceeds the frequency based baseline and TreeMHN.



A

Figure S29: **CloMu accurately predicts future mutations given a partial tree breast cancer data (Razavi et al., 2018).** CloMu substaintially outperforms TreeMHN and the mutation frequency based baseline on this dataset.

C.2 AML

CloMu was ran on the AML dataset (Morita et al., 2020) using a laptop with a 2.4 GHz CPU and 64 GB of RAM without using a GPU. The runtime was 41 minutes, and the memory usage was 261 MB. Further analysis of high fitness mutations and clonal prevalence data is performed in Supplemental Material C.2.1. An additional analysis of the latent representations on this data set is shown in Supplemental Material C.2.2. The relative causality predictions of CloMu are compared with TreeMHN's predictions in Supplemental Material C.2.3. CloMu's ability to predict subsequent mutations on subtrees of phylogeny trees is tested in Supplemental Material C.2.4.

C.2.1 Additional analysis of high fitness mutations and prevalence

We have orthogonally validated our fitness predictions on AML data through the use of clonal prevalence data. One relevant question is whether higher fitness mutations tend to occur earlier or later in clonal evolu-

tion. To do so, we measured the average number of mutations that have already occurred on a clone when some new mutation s is introduced. We averaged this over all patients for which that mutation is introduced. A histogram of these data for high-fitness mutations and all other mutations is shown in Fig. S30A. Fig. S30B shows the same for the breast cancer dataset (Razavi et al., 2018). This shows that high-fitness mutations generally but not always occur earlier in cancer evolution. One possible limitation in our fitness predictions is the fact that terminal mutations at the time of sampling do not have observed mutations occurring after them and thus do not contain any evidence for their fitness. Consequently, if a mutation always occurs terminally our model could not conclude that the mutation is highly fit.

Fig. S30C shows a scatter plot of the fitness of a mutation and what proportion of patients have that mutation occur terminally. Generally, higher fitness mutations do not occur terminally the majority of the time. However, we were not able to detect a statistically significant (p-value under 0.05) Pearson correlation between proportion terminal and mutation fitness given the relatively small number of high fitness mutations. For instance, Fig. S30C shows that two mutations (*NPM1* and *DNMT3A*) are vastly more fit than other mutations in the AML data set. Additionally, among mutations that are at most 50% terminal, correlations between proportion terminal and mutation fitness have a p-value above 0.75 on both datasets. It is unknown whether future data sets could uncover a statistically significant connection between fitness and percentage terminal.

Another relevant question is whether earlier mutations have higher log prevalence ratios due to the root clone (of normal cells) having a lower prevalence than the initial clones with one additional mutation added to the root. If one assumes a constant percentage rate of clonal growth, the log prevalence ratio of mutations would not be biased by how early or late they occur. Instead, the log prevalence ratio would only be affected by how much earlier the parent clone formed relative to the child clone (in addition to clonal growth rates). However, tumor growth may be more complex than this simple exponential model. Fig. S31A shows a histogram of the prevalence of the root node and the prevalence of clones with only a single mutation. We measured the log prevalence ratio of the healthy root clone and its children clones averaged across all patients. We find this mean log prevalence ratio to be 0.0569 ± 0.0738. This is sufficiently close to zero that our analysis is not biased by some mutations occurring earlier and thus being more likely to have occurred on the root clone. An additional question is whether terminal mutations tend to have a negative log prevalence ratio in addition to their low fitness. Fig. S31B shows a histogram of the prevalence of a terminal clone and its parent clone. Additionally, Fig. S31C shows a histogram of the log prevalence ratios of terminal

mutations. We find this mean log prevalence ratio to be -0.3799 ± 0.0903. This is meaningfully negative, however, this effect is of a smaller magnitude than the difference in log prevalence ratios between high-fitness mutations (mean: 0.8316) and low-fitness mutations (mean: -0.3703), shown in Main Text Figure 4B. To conclude, although our method may have diminished sensitivity in detecting high fitness mutations that typically occur terminally, we believe that specificity is not affected by mutation timing.

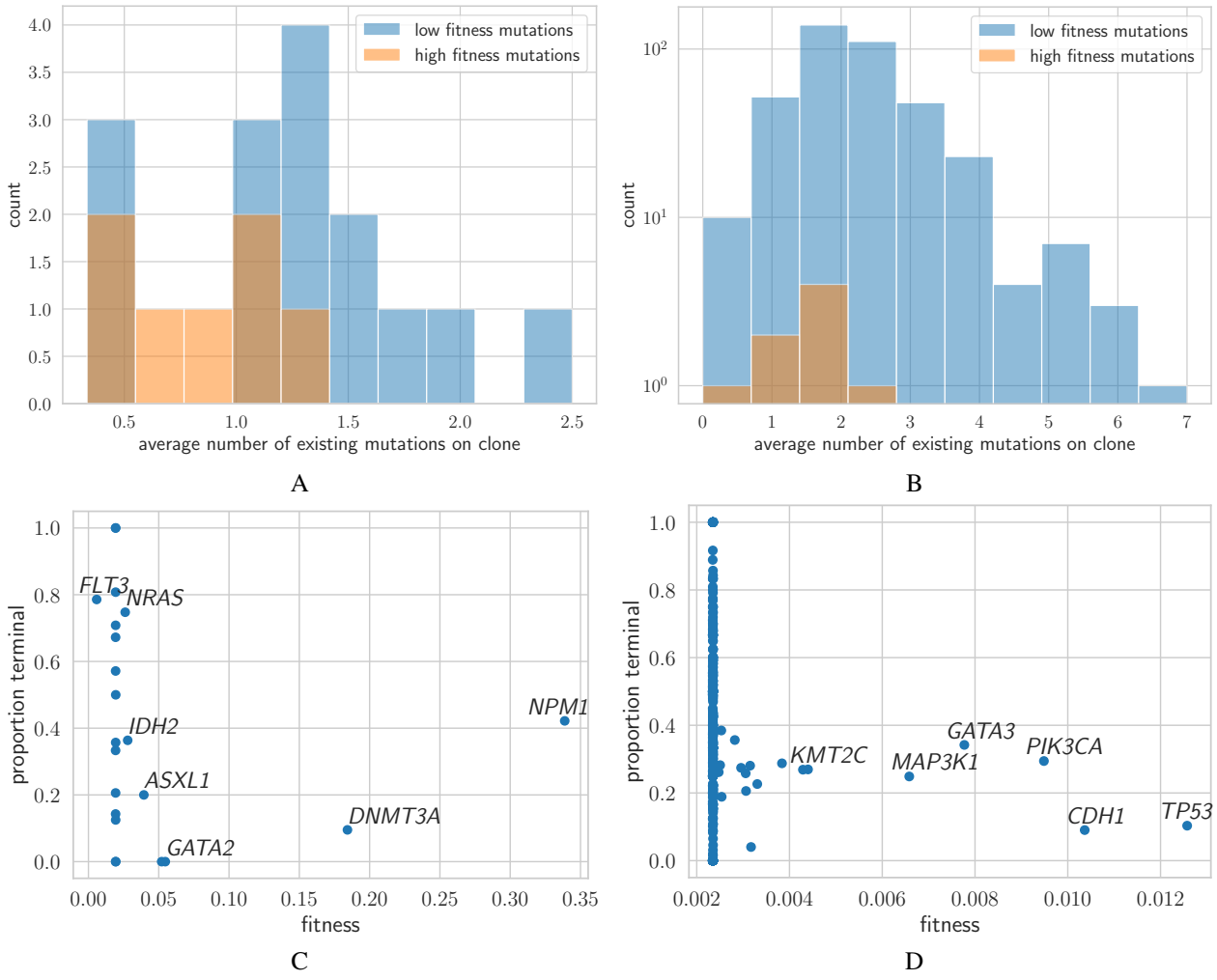
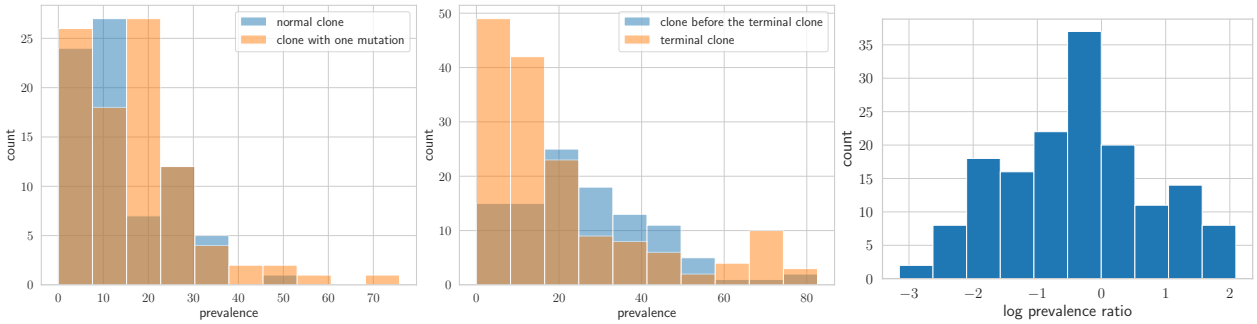


Figure S30: **High fitness mutations tend to occur earlier in evolution**. (A) On the AML dataset (Morita et al., 2020), higher fitness mutations often but not always occur early in tumor evolution. One exception is the mutation *NRAS* for which the average existing number of mutations on the clone is 1.384 which is above the median value of 1.147 across mutations. (B) On the breast cancer dataset (Razavi et al., 2018), higher fitness mutations often but not always occur early in tumor evolution. One exception is the mutation *ARID1A* with an average existing number of mutations on the clone equal to 2.123, which is above the median value of 2.054 across mutations. (C) On the AML dataset (Morita et al., 2020), mutations that are terminal the majority of the time tend not to be high fitness. However, among the mutations that are frequently non-terminal, there is no clear connection between fitness and the proportion for which it is terminal. (D) On the breast cancer dataset (Razavi et al., 2018), mutations that are terminal the majority of the time are not high fitness. However, among the mutations that are frequently non-terminal, the higher fitness mutations do not have the lowest proportion occurring terminally.



A B C

Figure S31: **Analyzing the prevalence of initial clones and terminal clones**. (A) The prevalence of normal clones and clones with one mutation are found to be very similar. (B) The prevalence of terminal clones is found to be smaller on average than clones immediately before the terminal clone. (C) The log prevalence ratios of the terminal mutations are slightly negative on average.

C.2.2 Additional analysis of latent representations

The main text contains plots showing the full latent representations of all mutations on the AML dataset (Morita et al., 2020). However, as mentioned in Supplemental Material C.1.2 applying PCA can provide a more intuitive visualization of the results. Fig. S32 shows these PCA values with mutations colored by their fitness value. There is a strong correlation between PCA component 1 and fitness. The correlation between PCA component 1 and fitness is 0.9709 with a p-value under 10^{-13}. There is no statistically significant correlation between either PCA component 2 or component 3 with fitness (p-values of 0.7138 and 0.6175 respectively). Another interesting analysis of latent representations is the Euclidean distance between the latent representations of different mutations. Fig. S33 shows a histogram of Euclidean distances between the latent representations of all pairs of mutations on the AML dataset (Morita et al., 2020). Many pairs of mutations are a very small Euclidean distance apart.

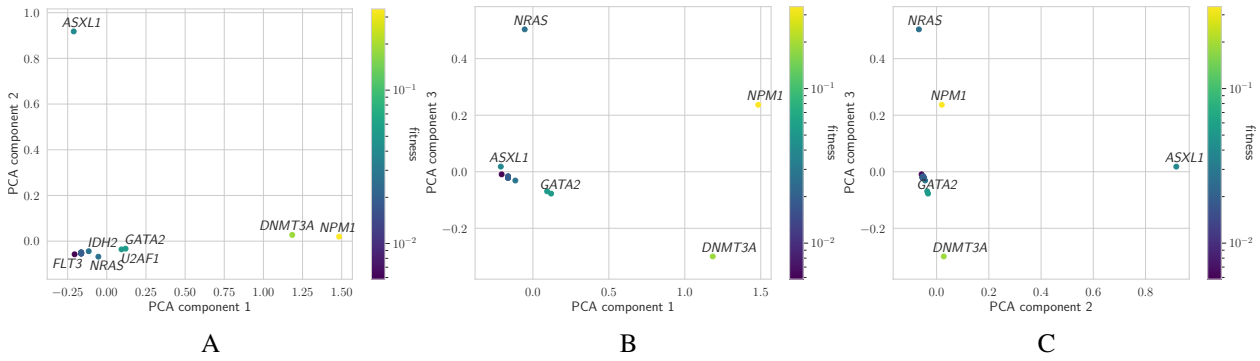


Figure S32: **PCA of latent representations on the AML dataset (Morita et al., 2020).** All mutations are shown in the scatter plot, however, low-fitness mutations tend to cluster near the bottom left. Mutations are colored based on their fitness value. The correlation between PCA component 1 and fitness is 0.9709 with a p-value under 10^{-13}. There is no statistically significant correlation between either of the other PCA components and fitness. (A) PCA components 1 and 2 are plotted. (B) PCA components 1 and 3 are plotted. (C) PCA components 2 and 3 are plotted.

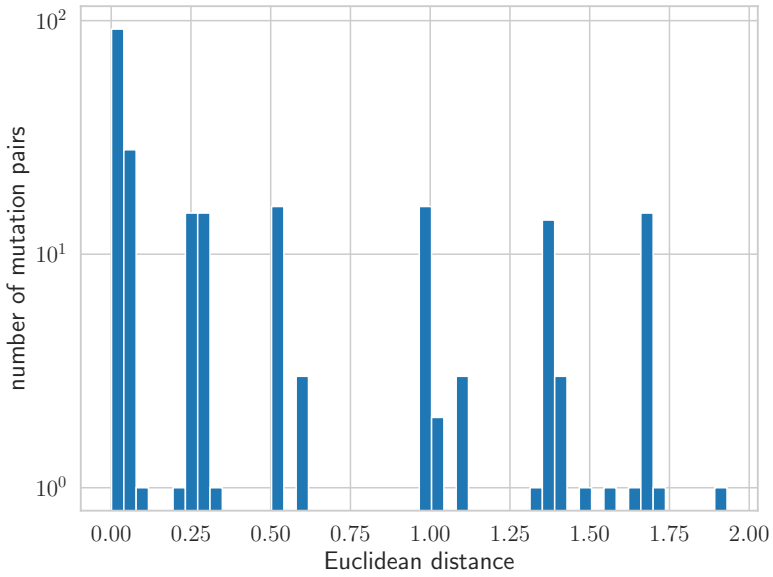


Figure S33: **Many mutations have highly similar latent representations in the AML dataset (Morita et al., 2020)** . A histogram of Euclidean distances between pairs of mutation latent representations is shown. The large spike around 0 demonstrates interchangeable mutations (including low-fitness mutations).

C.2.3 Systematic comparison with TreeMHN

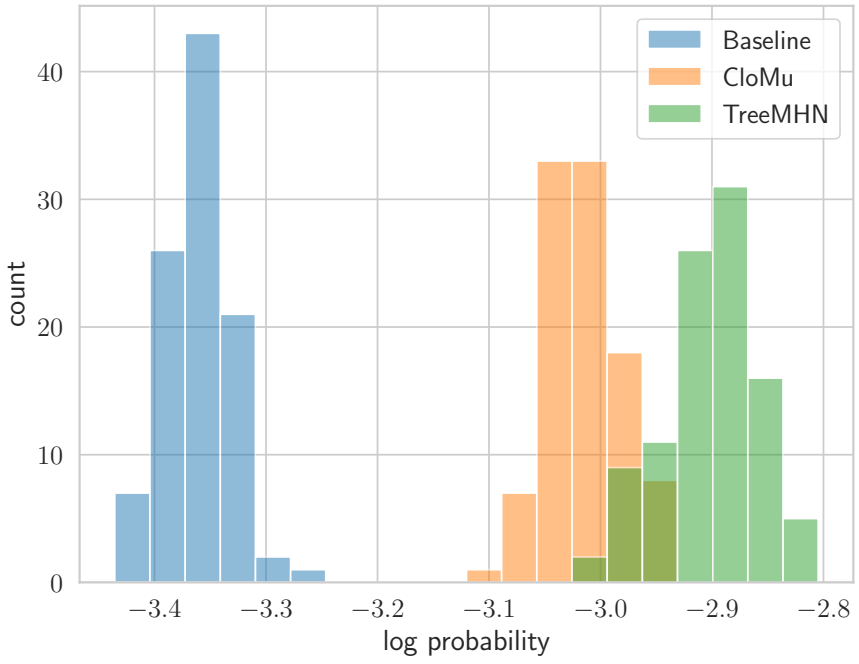
As mentioned in Supplemental Material C.1.3, CloMu is expected to predict lower relative causality values from fit mutations when compared to TreeMHN's causality predictions (due to CloMu accounting for fitness differently). Thus, if TreeMHN predicts a negative causal relationship then CloMu should as well and if CloMu predicts a positive causal relationship then TreeMHN should as well. As in Supplemental Material C.1.3, we restricted to pairs of mutations with meaningfully nonzero latent representations when predicting causal relationships. Additionally, we did not consider causal relationships from *FLT3* since *FLT3* is the only low-fitness mutation with a meaningfully nonzero latent representation. However, we still do consider causal relationships where *FLT3* is the target mutation. On this restricted set of mutation pairs, we analyzed all ordered pairs of mutations in which either TreeMHN predicted a negative causal relationship or CloMu predicted a positive causal relationship. We specifically analyzed the 4 strongest positive causal relationships according to CloMu, since all other positive causal relationships were extremely weak. As in Supplemental Material C.1.3, there was a general agreement between the causal relationship predictions of CloMu and TreeMHN. All of TreeMHN's negative predictions were also predicted negative by CloMu. Additionally, all of CloMu's positive predictions were either predicted positive of neutral by TreeMHN (Table S4, 2 positive cases, 2 neutral cases). However, there exist 2 cases in which TreeMHN predicts a positive causal relationship and CloMu predicts a negative causal relationship.

Source mutation	Target mutation	TreeMHN prediction	CloMu prediction
ASXL1	*NPM1*	Negative	Negative
ASXL1	*DNMT3A*	Negative	Negative
NPM1	*ASXL1*	Negative	Negative
NPM1	*DNMT3A*	Negative	Negative
ASXL1	*NRAS*	Neutral	Positive
NPM1	*NRAS*	Positive	Positive
NPM1	*FLT3*	Positive	Positive
NRAS	*FLT3*	Neutral	Positive

Table S4: On the AML cohort, all negative causal relationships predicted by TreeMHN are also predicted by CloMu. Additionally, all positive causal relationships predicted by CloMu are either positive or neutral according to TreeMHN.

C.2.4 Predicting subsequent mutations on subtrees

Finally, we evaluate performance by observing subtrees of patients' trees and predicting which mutations will subsequently occur. A description of how we perform this analysis is provided in Supplemental Material C.1.4. A histogram of the results is Fig. S34. Similarly to the breast dataset, CloMu also outperforms the simple mutation frequency based baseline on the AML dataset. On the other hand, TreeMHN performs excellent on the AML dataset (slighlty outperforming CloMu), but has worse performance on the breast cancer data set. We believe this is due to advantages of CloMu and the simple frequency based baseline on datasets with vast numbers of passenger mutations. Specifically, the mutation frequency based baseline has the advantage of exactly knowing the frequency of mutations rather than only relying on estimates of mutation frequency stored in the model. Additionally, CloMu has the advantage of modeling how high-fitness mutations increase the rates of all passenger mutations (rather than having to individually model pairwise relationships as in TreeMHN).



A

Figure S34: **CloMu accurately predicts future mutations given a partial tree on the AML dataset (Morita et al., 2020).** CloMu vastly outperforms the mutation frequency based baseline but slightly underperforms TreeMHN.

References

Caravagna G, Giarratano Y, Ramazzotti D, Tomlinson I, Graham TA, Sanguinetti G, and Sottoriva A. 2018. Detecting repeated cancer evolution from multi-region tumor sequencing data. *Nature Methods* **15**: 707–714.

Christensen S, Kim J, Chia N, Koyejo O, and El-Kebir M. 2020. Detecting evolutionary patterns of cancers using consensus trees. *Bioinformatics* **36**: i684–i691.

Jamal-Hanjani M, Wilson GA, McGranahan N, Birkbak NJ, Watkins TB, Veeriah S, Shafi S, Johnson DH, Mitter R, Rosenthal R, et al.. 2017. Tracking the evolution of non–small-cell lung cancer. *New England Journal of Medicine* **376**: 2109–2121.

Khakabimamaghani S, Malikic S, Tang J, Ding D, Morin R, Chindelevitch L, and Ester M. 2019. Collaborative intra-tumor heterogeneity detection. *Bioinformatics* **35**: i379–i388.

Kuipers J, Moore AL, Jahn K, Schraml P, Wang F, Morita K, Futreal PA, Takahashi K, Beisel C, Moch H, et al.. 2021. Statistical tests for intra-tumour clonal co-occurrence and exclusivity. *PLOS Computational Biology* **17**: e1009036.

Luo XG, Kuipers J, and Beerenwinkel N. 2023. Joint inference of exclusivity patterns and recurrent trajectories from tumor mutation trees. *Nature Communications* **14**: 3676.

Morita K, Wang F, Jahn K, Hu T, Tanaka T, Sasaki Y, Kuipers J, Loghavi S, Wang SA, Yan Y, et al.. 2020. Clonal evolution of acute myeloid leukemia revealed by high-throughput single-cell genomics. *Nature communications* **11**: 1–17.

Prüfer. 1918. Neuer beweis eines satzes uber permutationen. *Arch Math Phys* **27**: 742–4.

Razavi P, Chang MT, Xu G, Bandlamudi C, Ross DS, Vasan N, Cai Y, Bielski CM, Donoghue MT, Jonsson P, et al.. 2018. The genomic landscape of endocrine-resistant advanced breast cancers. *Cancer cell* **34**: 427–438.

Turajlic S, Xu H, Litchfield K, Rowan A, Horswell S, Chambers T, O'Brien T, Lopez JI, Watkins TB, Nicol D, et al.. 2018. Deterministic evolutionary trajectories influence primary tumor growth: TRACERx renal. *Cell* **173**: 595–610.

Williams RJ. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **8**: 229–256.

Yeang CH, McCormick F, and Levine A. 2008. Combinatorial patterns of somatic gene mutations in cancer. *The FASEB journal* **22**: 2605–2622.