Supplemental Methods of "scLongTree: an accurate computational tool to infer the longitudinal tree for scDNAseq data"

Rituparna Khan and Xian Mallory*

Florida State University, Tallahassee FL 32306, USA,
xfan2@fsu.edu

1 Description of our simulator

The simulation is a generative process that simulates a tree, the nodes of which represent subclones, and the edges may have new or back mutations on the child node compared with the parent node. We pre-define the total number of time points to be 3, whereas the first two time points happened 15 and 7 years ago, and the third time point happens currently (i.e., 0 years ago).

We then uniformly sample the total number of cells sequenced at each time point from [100, 300, 600, 1000], like what LACE did [2].

On the tree, each node has an interval which is within the range of 0 and 1. At any time, all the leaf nodes' intervals add up to 1. The purpose of having the intervals for each node is for distributing the cells. At each time point, the percentage of the cells a node may be distributed is linear to the interval the node has. The tree generation process starts with a root node which represents a normal clone without any mutations and is on time point 1. The root node's interval is [0, 1]. This root node at first is a leaf node before we add any child nodes to it. At every iteration, we select a leaf node to split into two child nodes, or have only one child node which has the same set of mutations as the parent, or not have any child node. These three cases is selected by the probability of $1-v2$, $v2-v1$ and $v1$, respectively. Here $v1$ and $v2$ are the two numbers between 0 and 1, and $v2 > v1$.

We then further decide whether the child nodes have new mutations compared with the parent node if the parent node has been decided to split into two child nodes. Specifically, both child nodes are subject to have new mutations with the probability of θ, whereas one of the two child nodes will not have new mutations with the probability $1 - \theta$. Here θ is a variable that is tunable in the simulator.

This whole process of having zero, one persistent or two child nodes, whereas the child nodes may or may not have new mutations mimics the real case scenario of subclonal growth. If a node is decided to have at least one child node, the child nodes become leaf nodes and the parent node is no longer a leaf node. The union of the child nodes's intervals is the same as the parent node's interval. If a node is decided to have two child nodes, the two child nodes split the interval the parent node has, and the two child nodes' interval ratio is decided by a sample from the beta distribution whose $\alpha = 0.5$, $\beta = 0.2$. In this way, we control the size of the subclones by controlling the parameters of the beta distribution. Such a usage of Beta splitting model can also be found in [1]

To decide the child nodes' time point, we consider whether the child nodes shall be at the next sequencing time point, or in between the current and next sequencing time point and thus represents an unobserved subclone. We consider unobserved subclones in our simulation because in the real case scenario, it is possible that some subclones with a unique set of mutations occur in between two sequencing time points, whereas such subclones are not represented by any cells being sequenced. In more detail, at each time point, we randomly select a node and decide that this node has two child nodes which represent unobserved subclones by probability u, a parameter tunable in the simulator. To increase the complexity of the tree, root node always has two child nodes that represent unobserved subclones. Notice that a node selected to split into two child nodes representing unobserved subclones is not subject to the probabilities of not having two child nodes such as $v2 - v1$ and $v1$. This process of selecting a leaf node, deciding whether it will split, or is inherited by one node, or does not have any child nodes, stops when all the nodes except those on the last time point have been selected and settled.

Once the tree structure is decided, next, we place the mutations on the edges of the tree based on the branch length, whereas the branch length is the number of years in between two consecutive sequencing time points. To decide the branch lengths that connect the unobserved nodes, we determine the time points of the unobserved nodes in between two consecutive sequencing time points $t1$ and $t2$ by sampling from a Beta distribution whereas $\alpha = \beta = 2$. Suppose such a sample is s. The time point of the unobserved node is $t1 + (t2 - t1)s$. Finally, given the branch length of each branch, say b, on the tree, the number of mutations each branch is distributed is ab, in which a is a new parameter that we varied to test the robustness of scLongTree. a represents the mutation rate. Here we varied a from 1.7, to 3.4, to 5.1.

Then we distribute the cells on the observed nodes. First, we normalize the intervals of all the observed nodes at each sequencing time point. The normalization is necessary since some nodes do not have any child nodes, and thus shrink the sum of the interval lengths and make it smaller than 1. We then distribute the number of cells according to the length of the interval whereas the total number of cells of each time point has been predefined. Notice that since the unobserved nodes do not have any cell sequenced, we do not distribute any cells on the unobserved nodes. Thus all cells are distributed on the nodes that are placed on the sequenced time points.

After the cells are placed on the tree, the G matrix is decided. A cell's underlying true genotype is decided by the mutations on the path from root to the cell's node. For each cell, we then decide the observed data D which has false positive (FP), false negative (FN) and missing entries. Specifically, given the FP rate and FN rate, we flip from 0 to 1 according to the FP rate, and from 1 to 0 according to the FN rate. We finally flip from 0 and 1 to 3 according to the missing rate. In our simulation, FP, FN and missing rates were the three variables that we varied to test the robustness of scLongTree, respectively. We selected FP rate from 0.001, 0.01, 0.03, and 0.05, and FN rate from 0.1, 0.2, 0.3 and 0.4. Missing rate was selected from 0.2 and 0.3. These numbers were selected based on our prior knowledge of the error rates [1, 2].

We performed the simulation study given different FP, FN and missing rates in two different ways. In the first way, we sampled these error rates once and applied the sampled error rates to all time points. Thus the error rates remained the same across different time points. In the second way, we allowed different error rates at different sequencing time points. Specifically, we tested scLongTree in two different modes in the second way. The first way was called “less varied”, in which the FP rate was sampled between 0.01 and 0.03, and FN rate was sampled between 0.2 and 0.3, whereas missing rate was fixed at 0.2. Each of the sampled error rate was applied to one time point, and thus the FP and FN rates might be different at different time points. The second way was called “more varied”, in which all the error rates were sampled from all ranges tested, i.e., FP rate was sampled from 0.001, 0.01, 0.03 and 0.05; FN rate was sampled from 0.1, 0.2, 0.3 and 0.4, and missing rate was sampled from 0.2 and 0.3. Like the “less varied”, “more varied” mode also sampled the error rate for each time point and thus the error rates at different time points might be different. This simulation was to test scLongTree’s robustness when the error rates varied across time points, which was to mimic the real case scenario as the change of the lab equipment and the sequencing technology may change error rates.

2 Motivation of clustering cells across all time points

We used BnpC to cluster all cells across all time points regardless which time point each cell was sequenced as an initial cell clustering result. We did this as the first step of scLongTree for three reasons.

First, it simplifies the downstream analysis of building the longitudinal subclonal tree. The resulting clusters of cells, after being stratified to each time point, represent the subclones of cells. Therefore, the remaining problem is to build a longitudinal subclonal tree based on the subclones of cells instead of individual cells.

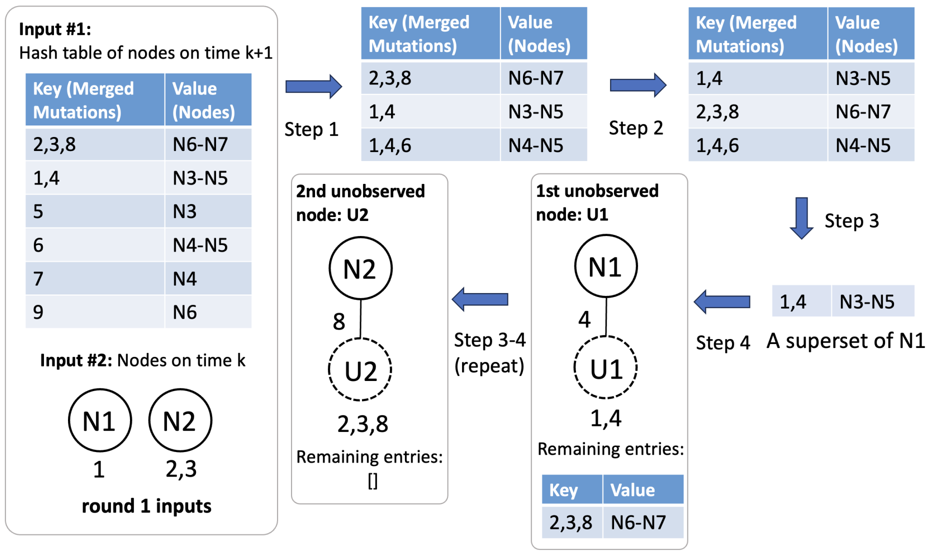
Second, it scales up the downstream analysis. Although the number of single cells may grow due to the decreasing cost of single cell sequencing, the number of subclones is not dependent on the sequencing cost but the tumor evolution. Thus after the step of clustering, the downstream analysis is not dependent on the number of cells any more, and therefore makes scLongTree scalable to thousands of cells.

Third, clustering cells across all time points links together cells with the same underlying genotype scattered among different time points. This can enhance the cell clustering accuracy especially for the subclones that are small but are scattered across two or more time points. In conclusion, it harnesses the fact that some subclones may remain the same in multiple consecutive time points to increase the accuracy of clustering, as well as that of the downstream analysis.

3 Illustration of Tree Inference Algorithm

This section illustrates how our tree inference algorithm works with a simple example. Suppose on time k, scLongTree inferred two subclones: N1 ({1}) and N2 ({2, 3}), the numbers in the parenthesis of which were the mutations that each subclone's genotype had. On time $k+1$, scLongTree inferred five subclones: N3 ({1, 4, 5}), N4 ({1, 4, 6, 7}), N5 ({1, 4, 6}), N6 ({2, 3, 8, 9}) and N7 ({2, 3, 8}).

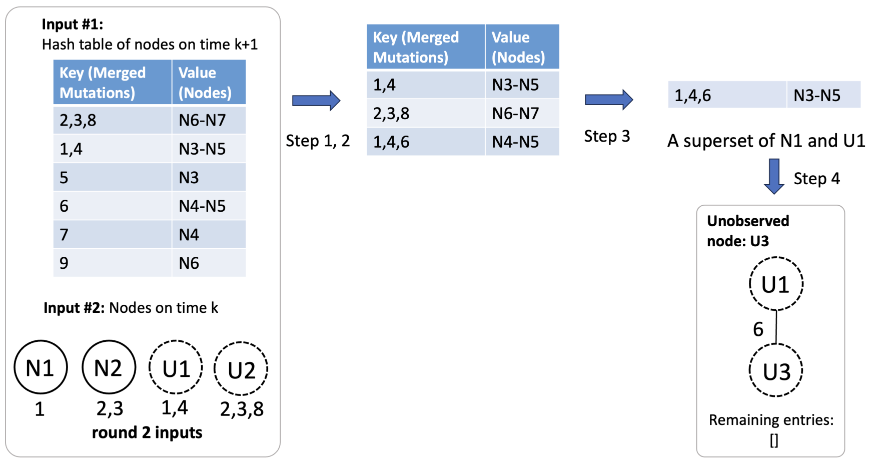
As shown in **Supplemental Fig. S1**, in the first round of the tree inference algorithm, we aimed at inferring the first layer of the unobserved nodes in between time k and $k+1$. There were two inputs to the algorithm. The first input was a hash table in which the key was the mutations and the value was the subclones that contained the mutations. Here we skip the step of merging keys with the same value, and suppose the keys have already been merged. As it can be seen, N6 and N7 share mutations {2, 3, 8} (first entry of the hash table); N3, N4 and N5 share mutations {1, 4}(second entry), and N4 and N5 share mutation {6} (third entry). All the rest of the entries had only one subclone in the value, and thus were eliminated by step 1. In addition, step 1 also replaced the remaining keys with the maximum subset of mutations contained by all subclones in the corresponding values. Thus the key of value N4 and N5 was replaced with mutations {1, 4, 6}. Step 2 decreasing sorted the entries by the number of subclones in the value. Starting from the first entry of the hash table, the algorithm checked whether the key was a superset of one of the subclones in the previous time point, k, in step 4. Since set {1, 4} was a superset of {1} which was the mutation contained by N1 at time k, the first unobserved node was identified, called U1, and U1's mutation set was {1, 4}. The algorithm then connected N1 and U1, and the edge between the two had a new mutation 4. In this step (step 4), we also removed N3, N4 and N5 from the other entries' values if any. Since the third entry had only N4 and N5 in the values, after the removal of the values, this entry became empty and thus was eliminated. We also eliminated the first entry with key {1, 4} since this entry had already been processed. We then repeated step 3 which searched for a key whose set of mutations was a superset of any of the subclones at time k, and step 4 which constructed the unobserved subclone. We found another unobserved subclone, called U2, whose mutations were {2, 3, 8}. We connected U2 with its parent node N2, and annotated a new mutation 8 on their edge. After this, the hash table became empty.



Supplemental Fig. S1: Illustration of the process of searching for the unobserved subclone in the first layer. The first input was a hash table whose keys were the merged mutations and values were the subclones at time $k+1$ containing the merged mutations. The second input was the subclones detected in time k. We put the mutations that these subclones contained under the circles. Outputs were the unobserved nodes in dashed circles, and annotated on the edge was the new mutation.

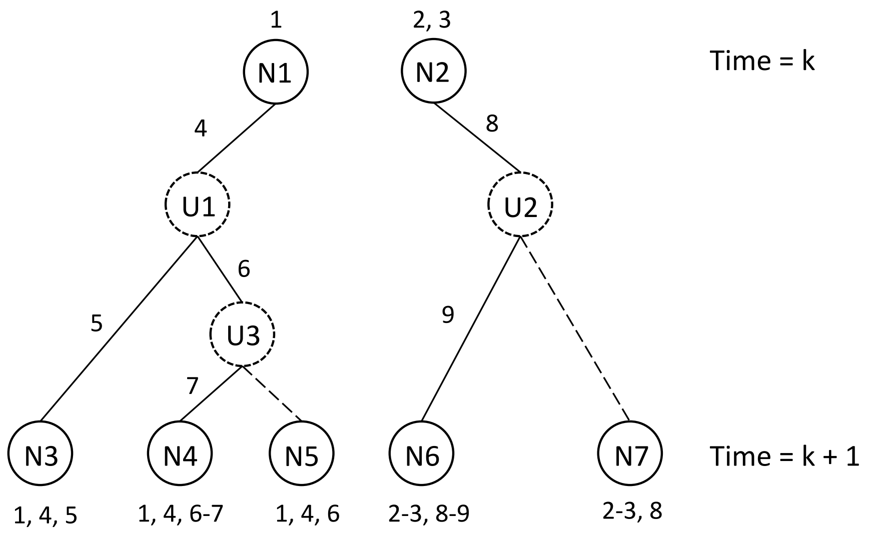
In the second round of the tree inference algorithm (**Supplemental Fig. S2**), we searched for the second layer of the unobserved nodes in addition to those found in the first round. In this round, the first input was exactly the same as that of the first round. The second input contained not only the subclones detected in time k, but also those unobserved subclones identified in the first round. Thus there were in total four nodes as the second input, which were N1, N2, U1 and U2. The first two steps in the second round were exactly the same as that of the first round. In step 3, we eliminated those entries whose key was exactly the same as one of the nodes listed in the second input. Thus the first and second entries were eliminated as they were the same as U1 and U2, respectively. Since the remaining entry's key $\{1, 4, 6\}$ was a superset of the mutations of both N1 and U1, we selected U1 as the priority was given to the unobserved node. In this way, we identified an unobserved node, called U3 whose parent node was U1, and on their edge was a new mutation 6. The algorithm stopped as this last entry was eliminated after it was processed, and the hash table became empty.

At last, for each subclone in time $k+1$, we connected it with the unobserved node whose value contained this subclone. We gave priority to the unobserved node



Supplemental Fig. S2: Illustration of the process of searching for the unobserved subclone in the second layer. The first input was a hash table whose keys were the merged mutations and values were the subclones at time $k+1$ containing the merged mutations. The second input was the subclones detected in time k, as well as those unobserved nodes detected in the first layer. We put the mutations that these subclones contained under the circles. Output was the new unobserved node U3 in dashed circle that had a new mutation 6 and was a child node of U1.

in the second layer. For example, subclones N4 and N5 appeared in the values of both $\{1, 4\}$, and $\{1, 4, 6\}$. Since $\{1, 4, 6\}$ was processed and used to construct unobserved node U3 in the second layer, whereas $\{1, 4\}$ was processed in the first layer, we connected N4 and N5 with U3. The final longitudinal subclone tree for these two time points is shown in **Supplemental Fig. S3**.



Supplemental Fig. S3: The inferred longitudinal subclonal tree between time point k and $k+1$. Unobserved nodes are in dashed circles. Edges without new or back mutations are in dashed lines. Mutations each subclone has are annotated on the top or bottom of the circles.

4 Supplemental Table S1

False positive	0.001, 0.01(d), 0.05
False negative	0.1, 0.2(d), 0.3, 0.4
Missing rate	0.2(d), 0.3
u **controlling unobserved nodes**	0, 0.1, 0.2(d), 0.3
a **controlling # mutations**	1.7(d), 3.4, 5.1
Varying error rates across time points	constant (d), less varied, more varied

Supplemental Table S1: A list of the variables varied in the simulated datasets. Each line has a varying variable (first column) with the values in the second column. The default value is denoted by "(d)" on its right.

Bibliography

[1] R. Khan and X. Mallory. Assessing the performance of methods for cell clustering from single-cell dna sequencing data. *PLOS Computational Biology*, 19(10):e1010480, 2023.

[2] D. Ramazzotti, F. Angaroni, D. Maspero, G. Ascolani, I. Castiglioni, R. Piazza, M. Antoniotti, and A. Graudenzi. Lace: inference of cancer evolution models from longitudinal single-cell sequencing data. *Journal of Computational Science*, 58:101523, 2022.