scLongTree: an accurate computational tool to infer the longitudinal tree for scDNAseq data

Rituparna Khan and Xian Mallory*

Florida State University, Tallahassee, FL 32304, USA, xfan2@fsu.edu

Abstract. A subclonal tree that depicts the evolution of cancer cells is of interest in understanding how cancer grows, prognosis and treatment of cancer.

Longitudinal single-cell DNA sequencing data (scDNA-seq) is the single-cell DNA sequencing data sequenced at different time points. It provides more knowledge of the order of the mutations than the scDNA-seq taken at only one time point, and thus facilitates the inference of the subclonal tree. There is only one existing tool LACE that can infer a subclonal tree based on the longitudinal scDNA-seq, and it is limited in accuracy and scale.

We presented scLongTree, a computational tool that can accurately infer the longitudinal subclonal tree based on the longitudinal scDNA-seq. ScLongTree can be scalable to hundreds of mutations, and outperformed state-of-the-art methods SCITE, SiCloneFit and LACE on a comprehensive simulated dataset. The test on a real dataset SA501 showed that scLongTree can more accurately interpret the progressive growth of the tumor than LACE. ScLongTree is freely available on https://github.com/compbio-mallory/sc_longitudinal_infer.

Introduction

Cancer grows by acquiring somatic mutations, whereas different set of cells may gain different sets of mutations. Such a phenomenon, known as intra-tumor heterogeneity (ITH), has confound the cancer treatment, prognosis and the prevention of metastasis [14, 2, 22, 15]. Unraveling the evolutionary history of tumor cells helps to characterize ITH and thus facilitates the design of the treatment plan. Such an evolutionary history can be characterized by a subclonal tree, on which the nodes represent the subclones of cells, each subclone having a unique set of mutations, and the edges represent the parent child relationship [26] where the mutations are accumulated. Thus the subclonal tree is a rooted directed tree, and the root represents the first normal cell that has not yet gained any somatic mutations.

Unlike the traditional "bulk" sequencing that combines millions of cells together for sequencing and thus cannot tell which mutation belongs to which cell, single-cell DNA sequencing (scDNA-seq) makes it possible to separately sequence each cell [18, 16], providing a high resolution of the subclonality and allowing a more accurate reconstruction of the subclonal tree [24]. However, scDNA-seq is erroneous, and the errors include false positive (FP) mutations, false negative (FN) mutations and missing entries due to uneven amplifications and low coverage, in addition to doublet errors [24]. Since the advent of scDNA-seq, there have been multiple attempts to decipher ITH using scDNA-seq data which all recruited necessary steps to overcome these errors, such as SCITE [9], OncoNEM [21], SASC [3], Sifit [23] and SiCloneFit [25]. These state-of-the-art computational tools typically assume that the data collected were sequenced at one time point, and thus do not consider the case when the data were collected at multiple time points whereas the time points are known for each cell sequenced.

Recently emerged the scDNA-seq datasets that were collected from different time points, such as whole exome or targeted sequencing [19, 27], patient-derived tumor xenograft models (PDXs) [5, 7], as well as the datasets that were collected pre-, mid- and post-treatment [11], or before and after relapse [17]. To reconstruct the evolutionary history given such longitudinal scDNA-seq data, new computational tools considering the time points when each cell was sequenced are needed. As far as we know, LACE [20] is the only computational tool that is specifically designed for inferring a subclonal tree given the longitudinal scDNA-seq data, in which each node represents a genotype that is contained by a set of cells, and each edge represents either a parental or persistence relationship. Such a tree was called "longitudinal subclonal tree" by LACE [20] since it incorporates the time points each cell was sequenced. Specifically, LACE models the problem of inferring the longitudinal subclonal tree by looking for the solution to a weighted maximum likelihood problem that allows distinct error rates across different time points, whereas the weight at each time point can be tuned by the prior knowledge of the quality of the data, as well as the number of cells at each time point. At each time point, LACE searches for the genotype each cell contains, as well as the mutations each genotype has, the latter of which was called "genotype-mutation" matrix and is shared by all time points. Since LACE holds the infinite site assumption (ISA) [12, 8], there is a one to one correspondence between the "genotype-mutation" matrix and the tree, whereas the tree corresponding to the "genotype-mutation" matrix is a perfect phylogeny [6]. However, the ISA disallows parallel mutations (mutations that occur in more than one edge) and back mutations (mutations that were gained and then lost) [13], the latter of which often occur due to the loss of copies in the genome [23]. Parallel mutations, although rare, cannot be totally dismissed either as convergent evolution may occur in cancer growth [4]. On the other hand, to find the solutions that maximize the weighted likelihood, LACE searches for the optimal solution by Monte Carlo Markov Chain to solve the Boolean matrix factorization problem. However, LACE is limited in its scalability of the number of mutations, and thus cannot render any tree when the number of mutations of interest is in the hundreds. Moreover, the genotypes present in LACE's tree have to be represented by at least one cell. In another word, LACE's methodology cannot reconstruct a subclone that once existed in the cancer evolution history but was sequenced in at least one cell. However, such reconstruction of the unobserved subclones is important because when two sequencing time points were far from each other, many events occurred in between the two time points and it is possible that a subclone was not sequenced before evolving and branching into other subclones.

We developed scLongTree, a computational tool to infer the longitudinal subclonal tree based on the longitudinal scDNA-seq data from multiple time points. Different from LACE [20], scLongTree does not hold a ISA and thus allows parallel and back mutations. In addition, scLongTree is scalable to hundreds of mutations

and can finish the inference within a couple of hours. Moreover, scLongTree has been designed to be able to reconstruct unobserved subclones that are not represented by any cells sequenced. Most importantly, by adopting a myriad of statistical methods as well as corroborating the cells all across distinct time points, scLongTree is able to identify spurious subclones and eliminate them, and thus having a high accuracy in the tree reconstruction and mutation and cell placement. We conducted comprehensive simulation and real dataset experiments to test the accuracy, robustness and computational resources of scLongTree, and found that scLongTree outperformed existing state-of-the-art methods such as LACE, SCITE and SiClone-Fit.

Methods

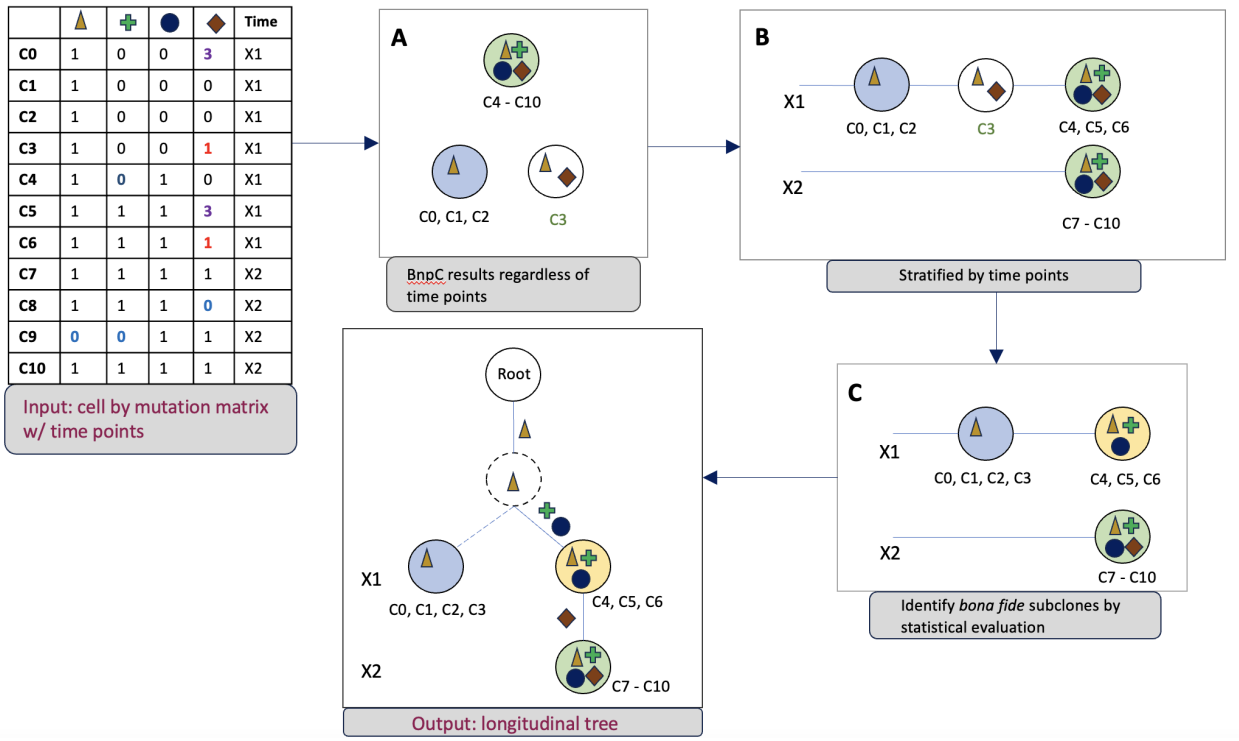


Fig. 1: Overview of scLongTree. Input is a cell by mutation matrix whereas each cell's time point is known. Observed data may have FP SNVs (red "1"s), FN SNVs (blue "0"s) and missing entries ("3"). ScLongTree first runs an existing cell clustering tool BnpC to cluster cells across all time points without considering the time point each cell was sequenced (**A**). It then separates the clusters by the time points (**B**). To correct spurious subclones such as the one composed by only cell C3, as well as the erroneously placed mutations such as the red diamond on the subclone of cells C4-C6, scLongTree statistically evaluates each subclone at each time point, and identifies the *bona fide* subclones and their mutations (**C**). Lastly, scLongTree infers a longitudinal tree that connects the subclones among different time points, and places the mutations on the edges. If necessary, scLongTree adds the unobserved nodes in between two consecutive time points (dashed circle in the output panel). Dashed edges represent edges without new or back mutation.

Given a cell by mutation matrix whereas each cell's time point is known, scLongTree starts with clustering the cells using an existing tool BnpC across all time points (**Fig. 1A**) to harness the power of number of cells that are scattered on different time points, based on the observation that a subclone may persist in multiple time points. Since BnpC is unaware of the time point of the cells which is informative of a more accurate clustering, scLongTree stratifies the clusters by the time points (**Fig. 1B**), and then further identifies the *bona fide* subclones, as well as corrects the mutations and cells in each subclone (**Fig. 1C**). Lastly, scLongTree infers a longitudinal tree in which each node represents a subclone of cells, and the new mutations gained or the mutations lost are annotated on the edges of the tree. ScLongTree's tree

inference algorithm is sophisticated in the sense that it can infer up to two levels of unobserved nodes in between two consecutive time points, and it searches for a tree with the least number of back mutations and parallel mutations. In the following, we define the problem, followed by the detailed description of cell clustering, identification of the *bona fide* subclones and their mutations and cells, as well as the inference of the longitudinal tree. Finally, we describe how we handle the non-deterministic result from BnpC.

Problem definition Input of the problem is a cell by mutation matrix, whereas each cell's time point is also known (**Fig. 1** top left panel). The matrix is composed of 0's and 1's that indicate the SNVs are absent and present in the cells, respectively, and 3's that indicate missing entries.

Our goal is to infer a longitudinal tree whose nodes represent subclones in each time point, and edges link the nodes for a parent-child relationship. When two consecutive sequencing time points are far from each other, there could be some branching events occurring between two time points. We also aim to infer the unobserved nodes that occur in between two consecutive time points.

Clustering cells ScLongTree's first step is to cluster cells across all time points (**Fig. 1A**). We utilize an existing state-of-the-art Bayesian-based method, BnpC [1], to perform an initial cell clustering. We chose BnpC because it is scalable to thousands of cells and its running time and accuracy are overall advantageous over other cell clustering methods such as SCG and SCClone [10]. To harness the power of the cells across all time points, we apply BnpC to all cells regardless their time points, and obtain the cell membership for each cell (**Supplemental Methods Section 2**).

Stratifying clusters by time points After the clustering by BnpC, we stratify the cells by the time point of their sequencing, such that the clusters of cells from BnpC are decomposed by the time points (**Fig. 1C**). As a result, the cells belonging to the same cluster are those assigned by BnpC to the same cluster as well as the cells that are sequenced at the same time point.

Identifying *bona fide* subclones Since BnpC is ignorant of the time points, it is prone to erroneously merge two clusters that differ from each other by only one or two mutations, whereas given the time points, the two clusters can be clearly distinguished from each other. Such an erroneous merging of the two clusters would lead to erroneous consensus genotype. We therefore correct the clustering, aiming to obtain an accurate consensus genotype of each cluster at each time point. First, for each stratified cluster at each time point that are supported by more than two cells, we obtain the consensus genotype by the majority voting. If the consensus genotype for a mutation cannot be decided such as a tie or all the cells' entries are missing, we use "unknown" as this mutation's consensus genotype. In addition, we require that a mutation's consensus genotype to be supported by at least two or more cells if the consensus genotype is not unknown. In this way, we eliminate the possibility that a consensus genotype is decided by only one cell. For those stratified cluster that has only one or two cells, we do not come up with a consensus genotype because a large percentage of the entries of the consensus genotype is unknown. After coming up with the consensus genotype, we subsequently estimate the initial FP rate α, FN rate β and missing rate γ for each time point using consensus genotype as the underlying genotype. Notice that the error rates for each time point are independently inferred and thus allow varying error rates across time points.Such error rates will be further refined after the refinement of cell clustering.

Based on the consensus genotype for each stratified cluster at each time point, we then evaluate the probability that the cluster is a *bona fide* subclone instead of a result of false clustering. Such a measurement should give a high probability if the stratified cluster is composed of a large number of supporting cells, or if the cells in one stratified cluster highly agree with each other on their genotypes and do not have many missing entries. Suppose there are n_l cells that are in subclone l at time point k. Suppose there are in total m mutations involved in the analysis in all time points. Let g be the consensus genotype being evaluated for subclone l at time point k, and $g = g_1, g_2, ..., g_m$. Suppose there are in total $v = u_l$ cells that are assigned to subclone l at time point k. Let the observed data for these cells be D_1, D_2, ..., D_v, each of which is a vector composed of m mutations. Thus D_{ij} represents the observed genotype for mutation j at cell i that

was assigned to subclone l at time point k. We calculate the probability of the consensus genotype g given the data D_1, D_2, ..., D_v from the v cells belonging to this subclone in **Eq. 1** by the bayes theorem and the assumption that each mutation is independent from each other.

$$P(g|D_1, D_2, ..., D_v) = \prod_{j=1}^{m} \frac{P(g_j) \prod_{i=1}^{v} P(D_{ij}|g_j)}{\sum_{\hat{g}_j} P(\hat{g}_j) \prod_{i=1}^{v} P(D_{ij}|\hat{g}_j)} \tag{1}$$

We write $P(D_{ij}|g_j)$ in **Eq. 2** that uses the initially estimated FP rate α, FN rate β and missing rate γ at time point l. We also define the prior probabilities for the consensus genotype $P(g_j)$ used in **Eq. 1** by $P(g_j) = \gamma$ if g_j is unknown, or $(1-\gamma)/2$ otherwise.

$$P(D_{ij}|g_j) = \begin{cases} (1-\gamma)(1-\alpha), & \text{if } D_{i,j} = 0, g_j = 0 \\ (1-\gamma)\alpha, & \text{if } D_{i,j} = 1, g_j = 0 \\ \gamma, & \text{if } D_{i,j} = 3, g_j = 0 \\ (1-\gamma)\beta, & \text{if } D_{i,j} = 0, g_j = 1 \\ (1-\gamma)(1-\beta), & \text{if } D_{i,j} = 1, g_j = 1 \\ \gamma, & \text{if } D_{i,j} = 3, g_j = 1 \\ \gamma/3, & \text{if } g_j = \text{unknown} \end{cases} \tag{2}$$

$$L_j(i|g) = \begin{cases} 1-\alpha, & \text{if } D_{i,j} = 0, g_j = 0 \\ \alpha, & \text{if } D_{i,j} = 1, g_j = 0 \\ \beta, & \text{if } D_{i,j} = 0, g_j = 1 \\ 1-\beta, & \text{if } D_{i,j} = 1, g_j = 1 \end{cases} \tag{3}$$

Once we obtain the posterior probability for each subclone at time k, we eliminate those subclones that have low posterior probabilities. We set up the threshold in the following way.

First, at each time point, we calculate the weighted average of the posterior probability whereas the weight is the number of cells supporting each cluster. Next, we set up the upper bound of threshold to be the maximum of the weighted average of the posterior probabilities across all time points. Starting from the subclone with the lowest posterior probability, we iteratively eliminate one more subclone in each round until we reach the upper bound. Each time we iterate, we run the whole algorithm and obtain a longitudinal tree. We then evaluate the number of back mutations and parallel mutations of each longitudinal tree that we obtain in each iteration. The threshold under which the subclones are eliminated that renders the least number of back and parallel mutations is selected. In this way, the threshold of eliminating spurious subclones is automatically determined. Notice that since our algorithm takes only 1-2 seconds to complete for each iteration, we are not concerned about the running time even when there are multiple runs of our algorithm to come up with a longitudinal tree. After this step, the number of subclones is identified at each time point.

Identifying cells for each subclone After identifying *bona fide* subclones, we reassign cells to the subclones and refine the clustering result. Specifically, for each cell i in time point k, we use **Eq. 3** to calculate its likelihood of belonging to a subclone l , $L_j(i|g)$, in terms of one mutation j. In this calculation, we do not consider the mutations with missing data or unknown consensus genotype because these entries are not informative of subclone that the cell belongs to. We then calculate the log likelihood $\sum_j \log L_j(i|g)$ for all eligible mutations which do not have missing entry on cell i and do not have an "unknown" on the consensus genotype for cluster l. We compute the average of the log likelihood by dividing $\sum_j \log L_j(i|g)$ by the number of eligible mutations for each cluster l, and select the cluster that has the highest likelihood to be the cluster that cell i belongs to. Once the cell membership is refined, we recalculate the consensus genotype by the majority voting like what we did in the previous step. Such consensus genotype will be used in the next step to come up with a longitudinal tree.

Inferring the longitudinal tree It has been observed that the subclones may persist from one time point to another. We first link the nodes that have the same consensus genotypes in two consecutive time points. The remaining nodes may be connected to a node in the previous time point, or an unobserved node in

between two consecutive time points, the genotype of which is not present in the nodes identified in the previous steps. In the following, we describe our algorithm of identifying unobserved nodes and linking the rest of nodes which has not found their parent node yet. The principle of obtaining unobserved nodes is to find the set of mutations contained by a set of nodes at the same time point, whereas such set of mutations and set of nodes are both as large as possible. An illustration of the entire process of identifying unobserved nodes is shown in **Supplemental Methods Section 3**.

At time $k+1$, we build a hash table whose key is a mutation that appear in any of the nodes in time $k+1$ and value is a node at time $k+1$ that has this mutation. We then merge the mutations that have the same value and eliminate the keys if their value has only one node, or if the key is exactly the same as the consensus genotype of a node in time k. After this, we decreasingly order the merged keys by the number of nodes in the value. In the same order, we check if a merged key is a proper superset of the consensus genotype of a node in time k. If yes, the merged key is designated as an unobserved node, and the two nodes, the unobserved node and the node in time k, are linked. We then remove all the nodes that are in the values of the unobserved node from the values of the rest of the hash table, as well as the unobserved node's key value entry from the hash table. We also remove any entry whose value has only one node or is empty. This process continues until no more unobserved node can be identified. Since it is possible that back mutations occur in the unobserved nodes leading to that the keys are not the proper superset of the nodes at time k, we use the following method to designate more unobserved nodes. We first decreasingly sort the remaining merged keys by the number of mutations they overlap with the consensus genotype of any of the nodes in time k, and designated the unobserved node according to this order. The designated unobserved node is linked to the node in time k whose consensus genotype has the maximum overlap with the unobserved node. We then remove the nodes in the values of the unobserved nodes from the rest of the hash table, as well as the entry corresponding to the newly designated unobserved node like what we did in the first round. Likewise, any entry whose value has only one node or is empty is removed. This process continues until no more unobserved node can be identified in this way. Through this whole process, we identified one layer of unobserved nodes.

ScLongTree allows at most more than one layer of unobserved nodes, considering a cascade of branching events may have been missed due to the large gap between two consecutive time points. The second layer of unobserved nodes is constructed in exactly the same way as the first layer except when checking the nodes in the previous time point to be connected to a designated unobserved node, instead of using only nodes in time k, we use both the nodes in time k and the unobserved nodes identified in the first layer, whereas the priority is given to the unobserved nodes in the first layer. In this way the second layer of the unobserved nodes is placed below the first layer, and thus become the first layer's unobserved nodes' children. After the construction of the second layer of unobserved nodes, we connect all identified unobserved nodes with their child nodes which are in the values corresponding to the merged keys, with the priority given to the second layer of unobserved nodes.

Finally we connect each remaining node in time $k+1$ by looking for a parent node in time k whose consensus genotype is a subset of the node in time $k+1$. If such a parent node does not exist, we connect the remaining node with a node in time k or the identified unobserved nodes in between time k and $k+1$ such that the resulting back mutations is the least.

Determining the optimal clustering Since many single cell clustering methods, such as BnpC, are not deterministic, leading to different clustering results for every run, scLongTree selects the optimal clustering results given multiple solutions and the corresponding tree by selecting the solution that has the least number of parallel mutations and back mutations.

Results

Simulation

We did a comprehensive simulation benchmark for the comparison among scLongTree, LACE, SCITE and SiCloneFit. Among these four methods only scLongTree and LACE were specifically designed for the longitudinal scDNAseq data. We chose SCITE and SiCloneFit because both of them have been widely used to

infer the phylogenetic tree given the mutation calls based on the scDNA-seq data. Notice that different from scLongTree and LACE, SCITE and SiCloneFit do not consider the time point each cell was sequenced. Also notice that since SCITE's inferred tree has only one new mutation appearing in each node on the mutation tree, it does not place two or more than two mutations on the same edge. For a fair comparison, for SCITE's results, we merged the mutations that were placed on the same segment of the path if it did not contain any branches.

We generated a tree whose nodes were designated to be at certain time points. Since it is possible that some nodes branched out in between two time points, we also allowed the branching to occur in between two time points. To accommodate this, we added nodes in between two time points that were not represented by any cells sequenced. We used a variable u to control this number of unobserved nodes, and u was a variable that we varied in our simulator to test the robustness of scLongTree. In addition, we also tested the robustness of scLongTree when there were different numbers of mutations involved. We used the variable a to control the number of mutations. Since the sequencing errors are the major issue of placing mutations on a tree, we tested scLongTree in response to different FP rates, FN rates and missing rates. In addition, we also tested scLongTree when the error rate at each time point varied from each other. Detailed explanation of the whole generative process as well as the variables used is in **Supplemental Methods Section 1**. **Supplemental Table S1** listed all the variables that we varied in the simulated dataset, along with the values that we used.

For evaluation metric, we used pairwise SNV accuracy as well as the SNV placement accuracy. Pairwise SNV accuracy evaluates the percentage of the correct pairwise SNV relationship among all pairs of SNVs. Two SNVs may have four possible pairwise relationships: ancestral, descendant, same edge and parallel. In addition, we also evaluated the accuracy of SNV placement which focused on the time points each SNV was placed on. Such an accuracy of SNV placement aimed at evaluating whether the SNV was placed at the right timing since we had multiple sequencing time points in the longitudinal data. In particular, the accuracy of SNV placement was evaluated by a harmonic mean of recall and precision, whereas recall evaluated the percentage of SNV correctly placed on a time point over all mutations placed at a time point by the method, and precision evaluated the percentage of mutations correctly placed on a time point over all mutations placed at a time point in the ground truth data. The recall and precision of all time points were calculated as the average of the recalls and precisions of all time points, respectively.

Fig. 2 showed the pairwise SNV accuracy of scLongTree, LACE, SCITE and SiCloneFit on the six variables that we varied in the simulated datasets. ScLongTree's median was the highest among all methods for all six variables at all values.

Specifically, scLongTree's pairwise SNV accuracy had a decreasing trend with the increase of FP rate. So did LCE and SiCloneFit (**Fig.2a**). However, for all values of FP rate, scLongTree's median pairwise SNV accuracy remained the highest of all methods, ranging between 0.75 to over 0.9, whereas LACE's ranged between 0.3 to over 0.5; SCITE's ranged between 0.5 to over 0.6; and SiCloneFit's ranged between over 0.5 to over 0.8. Thus scLongTree's median pairwise SNV accuracy was about 0.15 to 0.35 higher than all other methods for varying FP rate.

Like FP rate, increasing FN rates resulted in a decreasing trend of pairwise SNV accuracy for scLongTree (**Fig.2b**). Nevertheless, scLongTree's median pairwise SNV accuracy remained the highest for all FN rates being tested compared with LACE, SCITE and SiCloneFit, and the difference between scLongTree and the method with the second highest median pairwise SNV accuracy ranged between around 0.1 to 0.2.

Increasing missing rates, like FP and FN rates, resulted in a decreasing trend of pairwise SNV accuracy for scLongTree, as well as for SCITE and slightly for SiCloneFit (**Fig.2c**). We noticed that when the missing rate was 0.3, the median pairwise SNV accuracy between scLongTree and SiCloneFit was very close. However, scLongTree remained the method with the highest pairwise SNV accuracy of all four methods for all missing rates being tested.

In addition to the sequencing error rates such as FP, FN and missing rates, it was also important to test the robustness of the methods given different number of unobserved nodes, since the corresponding subclones were not sequenced. We thus varied the probability, u, of a node having child nodes as unobserved nodes

(**Supplemental Methods Section 1**). When u was 0, there were no unobserved nodes on the tree. The higher the u, the more unobserved nodes we simulated on the tree. We then applied the four methods to the resulting cell by mutation matrix (**Fig.2d**). As expected, the median pairwise SNV accuracy had a decreasing trend with the increase of u. However, the decrease was slight, and the median pairwise SNV accuracy ranged between around 0.94 to 1. The other methods did not show a clear trend of decreasing accuracy due to the increase of the unobserved nodes. Nevertheless, scLongTree's median pairwise SNV accuracy remained the highest among all methods, and its accuracy was higher than the second highest method, SiCloneFit, by around 0.1 to 0.33. We noticed that LACE's pairwise SNV accuracy was very low, the median of which was < 0.55 when the degree of the unobserved node was the highest at 0.3. This showed that LACE cannot accommodate the trees with unobserved nodes, as was also explained theoretically in the **Introduction**.

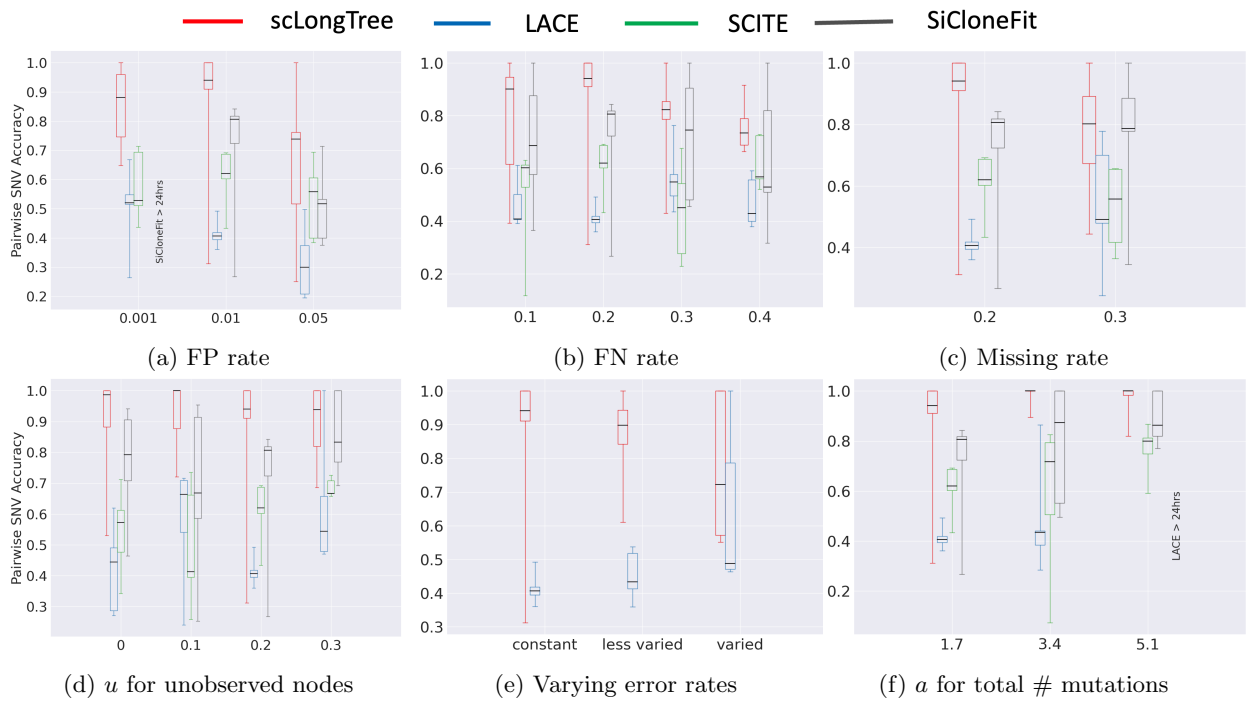
Next, we further compared scLongTree and LACE when the error rates (FP, FN and missing rates) varied across different time points, since both were designed to accommodate varying error rates at different time points (**Fig.2e**). As expected, scLongTree's pairwise SNV accuracy decreased with the increase of the range that the error rates may differ across time points. Specifically, scLongTree's median pairwise SNV accuracy decreased from around 0.93 to 0.72 when the range of the error rate increased from constant among all time points to more varied **Supplemental Methods Section 1**. LACE's pairwise SNV accuracy, in contrast, increased with the varying range of the error rate, from around 0.4 to < 0.5. However, even at the largest range of varying error rate, LACE's pairwise SNV accuracy was still at least 0.2 small than that of scLongTree's.

Finally, we looked into both accuracy and scalability of the four methods when the number of mutations on the tree increased. We did this by increasing a, from 1.7 to 5.1 (**Supplemental Methods Section 1**). This resulted in the increase of number of mutations from around 50 to 150. We found that scLongTree's accuracy increased with the increase of the mutation rate, and the median of the pairwise SNV accuracy for the two highest mutations rates for scLongTree remained 1 (**Fig.2f**). We observed a similar trend for SCITE and SiCloneFit. However, scLongTree's median pairwise SNV accuracy was at least 0.1 higher than the second highest method SiCloneFit for all mutation rates. LACE's median pairwise SNV accuracy remained the lowest of all methods, a little above 0.4. Importantly, when $a = 5.1$ which corresponded to around 150 total mutations in the input matrix, LACE cannot scale up and failed to finish the run within 24 hours. We then further looked into the running time of all four methods for varying as (**Fig.3b**). We found that SCITE was the fastest of all four methods. When the number of mutations was around 50, corresponding to $a = 1.7$, LACE was faster than scLongTree. When the number of mutations was around 100, corresponding to $a = 3.4$, scLongTree and LACE's running time were comparable to each other, at about 2-3 hours. When the total number of mutations on the tree was around 150, corresponding to $a = 5.1$, scLongTree's running time increased to about 5 hours, whereas LACE's increased to be over 24 hours. SiCloneFit's running time was the biggest next to LACE's when there were around 150 mutations on the tree, at around 11 hours for a run.

In addition to pairwise SNV accuracy, we were also interested in the accuracy of SNV placement on a certain time point, as shown in **Fig.3a**. Since the accuracy of SNV placement was measured by the time points each SNV was placed, whereas SCITE and SiCloneFit did not have the time points on their tree, we did not measure the accuracy of SNV placement for Sifit and SiCloneFit. Consistent with the observation from pairwise SNV accuracy, the median accuracy of SNV placement of scLongTree was much higher than LACE's, by about 0.06 to 0.08. LACE's median accuracy of SNV placement decreased with the increase of a, whereas scLongTree's median accuracy of SNV placement remained over 0.99 for all as being tested, showing the robustness of scLongTree.

Real data

We applied scLongTree to a real triple-negative breast tumor sample SA501 which had the longitudinal targeted PDX scDNA-seq data from three time points, $X1$, $X2$ and $X4$ [5]. Each time point had 27, 36 and 27 cells sequenced, respectively. LACE [20] was also applied to SA501, in which 20 somatic variants were selected to construct the longitudinal tree. In comparing scLongTree with LACE, we used the same 20 somatic variants as the input to scLongTree.

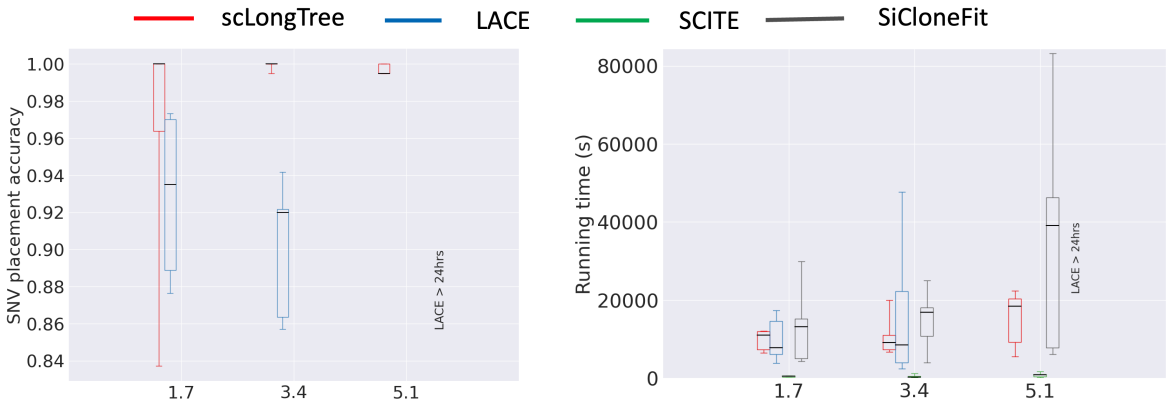


(a) FP rate (b) FN rate (c) Missing rate

(d) u for unobserved nodes (e) Varying error rates (f) a for total # mutations

Fig. 2: Boxplots are shown for pairwise SNV accuracy for scLongTree (red), LACE (blue), SCITE (green) and SiCloneFit (black). Six variables were varied to test the robustness of these four methods, including FP rate (a), FN rate (b), missing rate (c), variable u that controls the number of unobserved node (d), the range of error rate across different time points (e), and variable a that controls the number of mutations on the tree. Median of each boxplot was highlighted in black. Each set of variables were repeated for five times to avoid extreme cases.

As shown in **Fig. 4**, scLongTree reconstructed the longitudinal subclonal tree whose trunk had mutations 0-6, resulting in the first unobserved node (the first dotted circle). It then branched out into two subclones, one subclone gained new mutations 7-10, represented by the yellow circles with 11% of the cells in X1, and 39% of the cells in X2. These were consistent with LACE's tree and the tree shown in [5]. The other subclone that branched out from the trunk had new mutations 11-13 in an unobserved node before X1 (second dotted circle), followed by gaining new mutations 14-18, the resulting subclone of which had 15% of the cells in X1 and 39% of the cells in X2 (light blue circles). This lineage further gained mutation 19, resulting in a subclone with 100% of cells in X4 (dark blue circle). In addition to one branch that gained mutations 14-18, the other branch of the second unobserved subclone was also inherited by a subclone in X1 without any new or back mutations, represented by 74% of the cells in X2 (green circle). In contrast, LACE's other branch from the trunk placed new mutations 11-12 (chr6:79,509,700 and chr10:55,582,495) and mutation 13 (chr2:45,941,842), separately, on two different branches. The major reason that LACE separated the mutations 11-12 and mutation 13 was due to that a subclone of 8 cells in X2 had only mutations 11-12, but not mutation 13. In addition, this subclone had mutation 7, which had occurred on the parallel branch together with mutations 8-10 in the yellow circle. To reconcile the occurrence of such a subclone of 8 cells that had mutations 11-12 and mutation 7, scLongTree allowed a back mutation of 13 and a parallel mutation of 7 (orange circle), whereas LACE, constrained by the ISA, separated the gain of mutations 11-12 and mutation 13, and ignored the mutation 7 on these cells. We claim that the parallel gain of mutation 7 was biologically possible due to the convergent evolution resulting from the positive selection of the activating driver mutations [4, 23, 26]. Likewise, the loss of mutation 13 was possible due to the loss of heterozygosity or chromosomal deletion [23]. We further looked into the deep targeted scDNA-seq data [5] to check the sequencing support of the presence of mutations 7, 11 and 12 and the absence of mutation 13 for this orange subclone. Specifically, we checked how many cells had mutations 7, 11 and 12 but did not have mutations 13 on X2, as well as how strong the evidences of the presence and absence of the mutations were. We found that there were three nuclei, nuclei 18, 28 and 35 on X2 that had strong support of the presence of mutations 7, 11



(a) Accuracy of SNV placement (b) Running time

Fig. 3: Boxplots are shown for accuracy of SNV placement (a) and running time in seconds (b) for varying variable a for scLongTree (red), LACE (blue), SCITE (green) and SiCloneFit (black). Median of each boxplot was highlighted in black.

and 12 and the absence of mutation 13. To be more specific, each variant allele that was present had at least 0.19 variant allele fraction (VAF), and most of the variant alleles had the fraction near 0.5 for mutations 7, 11 and 12. Since it was targeted sequencing data, the total number of reads covering each locus was at thousands. Therefore mutations 7, 11 and 12 had hundreds of variant supporting reads. In contrast, mutation 13's VAF was small than 0.001 for these three nuclei, and thus the variant supporting read number was at a single digit compared with thousands of read covering the site. In addition, we also noticed a nucleus 13 at X2 that had high VAF at mutations 11 and 7, at 0.41 and 0.997, respectively, but low VAF at mutations 12 and 13 (lower than 0.0021 for both). We inferred that nucleus 13 probably had an allele dropout of the variant allele on mutation 12, and it should belong to the orange node together with the nuclei 18, 28 and 35. We further check the reference and variant alleles for these four mutations given the prior knowledge that FP mutations in scDNA-seq occur much more frequently at C->T or G->A conversions due to the deamination of cytosine bases [23, 24]. We found that none of these four mutations were C->T or G->A. Therefore the presence of the variant supporting reads was less likely due to the FP mutations but that the true variant allele was present in the cells. These sequencing evidences showed the validity of the orange subclone, and thus that the parallel mutation of 7 and back mutation of 13 truly existed in sample SA501.

Discussion

ScLongTree is a computational tool that can infer the phylogenetic tree based on the longitudinal scDNA-seq data. A myriad of approaches have been adopted in scLongTree to obtain a high accuracy such as harnessing the power of the number of cells across different time points for a more powerful clustering, using statistical analysis to eliminate small spurious subclones, and re-estimating the FP, FN and missing rates at each time point. In addition, we developed an efficient algorithm to construct the phylogenetic tree, in which we allow the inference of the unobserved subclones that occur in between two consecutive time points and thus are not represented by any cells. We compared with the existing only method that infer the longitudinal subclonal tree from the longitudinal scDNA-seq data, LACE, as well as two state-of-the-art methods that infer phylogenetic trees based on the scDNA-seq data from one time point, SCITE and SiCloneFit. We found that scLongTree outperformed all other three methods for all six experiments being tested in terms of pairwise SNV accuracy. In addition, scLongTree was scalable to hundreds of mutations, and its running time was the shortest next to SCITE, whereas LACE cannot finish the run within 24 hours when there were over 100 mutations in the input matrix. An application of scLongTree to a triple-negative breast cancer dataset SA501 that had three sequencing time points showed that not only was scLongTree able to recover the evolutionary history of the cancer, but also was able to detect a novel subclone on the second sequencing time point (X2) which was not reported by LACE, possibly due to that LACE did not allow parallel and back

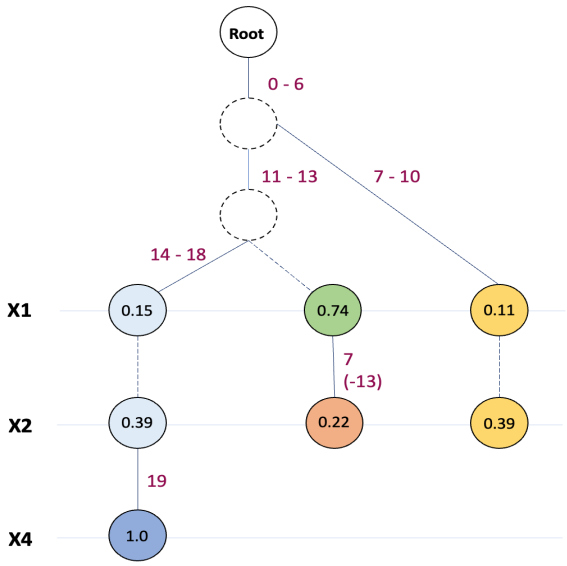


Fig. 4: ScLongTree's inferred longitudinal subclonal tree for sample SA501. Nodes with the same consensus genotype has the same color. Nodes not having any new or back mutations have dotted edges. If there is no new or back mutations on an edge, the edge is a dotted line.

mutations due to its adopting the ISA. We further checked the sequencing support of this novel subclone, and found strong read and cell support of one parallel and one back mutation in this subclone.

It is important to note that although scLongTree requires an existing cell clustering tool to cluster cells across different time points and thus is limited by the existing cell clustering tool being used, it has been made modular so that users can plug in their own clustering results as the input to scLongTree. A potential improvement of scLongTree is to combine the first three steps, i.e., cell clustering, stratification of the clusters based on the time points, and the elimination of spurious small subclones, into one procedure. This can help improve the cell clustering and eliminating spurious small subclones due to the fact that a small subclone in a time point may become a large subclone in the next time point, and a cell clustering algorithm considering the time points of all cells is needed to achieve a high accurate clustering result. Moreover, it is also a future direction of scLongTree to combine the clustering and tree inference in one step, which although may increase the computational complexity, may result in a higher accuracy of the inferred tree.

Lastly, we also aim at identifying *bona fide* mutations as one of our future directions. A mutation appearing in the cell by mutation matrix may not have occurred in any of the cells, but due to FP errors. On the other hand, mutations tend to co-occur in a subclone, and these mutations can corroborate each other so that they will not be identified as spurious mutations. A future work of scLongTree is to statistically identify spurious mutations in the cell by mutation matrix by considering the prior knowledge of the range of FP and FN rates, as well as the mutations that co-occur in the cells.

Data availability

Triple-negative breast tumor sample SA501 was obtained from the European Genome-phenome Archive under accession number EGAS00001000952. Further processed data was available in http://www.cbioportal.org. ScLongTree is freely available on https://github.com/compbio-mallory/sc_longitudinal_infer.

Acknowledgement

We would like to thank Daniele Ramazzotti for helping us interpret LACE's result. We also would like to thank Adi Steif and Peter Eirew for sharing the SA501 raw tables for our analysis.

Bibliography

[1] N. Borgsmüller, F. M. Jose Bonet, A. Gonzalez-Perez, N. Lopez-Bigas, and N. Beerenwinkel. BnpC: Bayesian non-parametric clustering of single-cell mutation profiles. *Bioinformatics*, 36:4854–4859, 2020. https://doi.org/https://doi.org/10.1093/bioinformatics/btaa599.

[2] R. A. Burrell, N. McGranahan, J. Bartek, and C. Swanton. The causes and consequences of genetic heterogeneity in cancer evolution. *Nature*, 501(7467):338–345, 2013.

[3] S. Ciccolella, C. Ricketts, M. Soto Gomez, M. Patterson, D. Silverbush, P. Bonizzoni, I. Hajirasouliha, and G. Della Vedova. Inferring cancer progression from single-cell sequencing while allowing mutation losses. *Bioinformatics*, 37(3):326–333, 2021.

[4] A. Davis and N. E. Navin. Computing tumor trees from single cells. *Genome biology*, 17:1–4, 2016.

[5] S. A. K. J. e. a. Eirew, P. Dynamics of genomic clones in breast cancer patient xenografts at single-cell resolution. *Nature*, 2019. https://doi.org/https://doi.org/10.1038/nature13952.

[6] M. El-Kebir, G. Satas, L. Oesper, and B. J. Raphael. Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures. *Cell systems*, 3(1):43–53, 2016.

[7] H. Gao, J. M. Korn, S. Ferretti, J. E. Monahan, Y. Wang, M. Singh, C. Zhang, C. Schnell, G. Yang, Y. Zhang, et al. High-throughput screening using patient-derived tumor xenografts to predict clinical trial drug response. *Nature medicine*, 21(11):1318–1325, 2015.

[8] D. Gusfield. Algorithms on stings, trees, and sequences: Computer science and computational biology. *Acm Sigact News*, 28(4):41–60, 1997.

[9] K. Jahn, J. Kuipers, and N. Beerenwinkel. Tree inference for single-cell data. *Genome biology*, 17(1): 1–17, 2016.

[10] 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.

[11] C. Kim, R. Gao, E. Sei, R. Brandt, J. Hartman, T. Hatschek, N. Crosetto, T. Foukakis, and N. E. Navin. Chemoresistance evolution in triple-negative breast cancer delineated by single-cell sequencing. *Cell*, 173(4):879–893, 2018.

[12] M. Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 61(4):893, 1969.

[13] J. Kuipers, K. Jahn, B. J. Raphael, and N. Beerenwinkel. Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors. *Genome research*, 27(11):1885–1894, 2017.

[14] M. S. Lawrence, P. Stojanov, P. Polak, G. V. Kryukov, K. Cibulskis, A. Sivachenko, S. L. Carter, C. Stewart, C. H. Mermel, S. A. Roberts, et al. Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature*, 499(7457):214–218, 2013.

[15] D. A. Lawson, K. Kessenbrock, R. T. Davis, N. Pervolarakis, and Z. Werb. Tumour heterogeneity and metastasis at single-cell resolution. *Nature cell biology*, 20(12):1349–1360, 2018.

[16] B. Lim, Y. Lin, and N. Navin. Advancing cancer research and medicine with single-cell genomics. *Cancer cell*, 37(4):456–470, 2020.

[17] K. Morita, F. Wang, K. Jahn, T. Hu, T. Tanaka, Y. Sasaki, J. Kuipers, S. Loghavi, S. A. Wang, Y. Yan, et al. Clonal evolution of acute myeloid leukemia revealed by high-throughput single-cell genomics. *Nature communications*, 11(1):5327, 2020.

[18] N. E. Navin. Cancer genomics: one cell at a time. *Genome biology*, 15(8):1–13, 2014.

[19] M. Pellegrino, A. Sciambi, S. Treusch, R. Durruthy-Durruthy, K. Gokhale, J. Jacob, T. X. Chen, J. A. Geis, W. Oldham, J. Matthews, et al. High-throughput single-cell dna sequencing of acute myeloid leukemia tumors with droplet microfluidics. *Genome research*, 28(9):1345–1352, 2018.

[20] 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.

[21] E. M. Ross and F. Markowetz. Onconem: inferring tumor evolution from single-cell sequencing data. *Genome biology*, 17(1):1–14, 2016.

[22] S. Turajlic, A. Sottoriva, T. Graham, and C. Swanton. Resolving genetic heterogeneity in cancer. *Nature Reviews Genetics*, 20(7):404–416, 2019.

[23] H. Zafar, A. Tzen, N. Navin, K. Chen, and L. Nakhleh. Sifit: inferring tumor trees from single-cell sequencing data under finite-sites models. *Genome biology*, 18(1):1–20, 2017.

[24] H. Zafar, N. Navin, L. Nakhleh, and K. Chen. Computational approaches for inferring tumor evolution from single-cell genomic data. *Current Opinion in Systems Biology*, 7:16–25, 2018.

[25] H. Zafar, N. Navin, K. Chen, and L. Nakhleh. SiCloneFit: Bayesian inference of population structure, genotype,and phylogeny of tumor clones from single-cell genome sequencing data. *Genome Research*, 2019. https://doi.org/https://doi.org/10.1101/gr.243121.118.

[26] H. Zafar, N. Navin, K. Chen, and L. Nakhleh. Siclonefit: Bayesian inference of population structure, genotype, and phylogeny of tumor clones from single-cell genome sequencing data. *Genome research*, 29(11):1847–1859, 2019.

[27] L. Zhang, X. Dong, M. Lee, A. Y. Maslov, T. Wang, and J. Vijg. Single-cell whole-genome sequencing reveals the functional landscape of somatic mutations in b lymphocytes across the human lifespan. *Proceedings of the National Academy of Sciences*, 116(18):9014–9019, 2019.