Title

Fast intratumor heterogeneity inference from single-cell sequencing data

Journal

Nature Computational Science, 2(9)

ISSN

2662-8457

Authors

Kızılkale, Can
Rashidi Mehrabadi, Farid
Sadeqi Azer, Erfan
et al.

Publication Date

2022-09-01

DOI

10.1038/s43588-022-00298-x

Peer reviewed

Fast Intratumor Heterogeneity Inference from Single-Cell Sequencing Data

Can Kızılkale[1,2,†], Farid Rashidi Mehrabadi[3,5,†], Erfan Sadeqi Azer[5,‡], Eva Pérez-Guijarro[4], Kerrie L. Marie[4], Maxwell P. Lee[4], Chi-Ping Day[4], Glenn Merlino[4], Funda Ergün[5], Aydın Buluç[1,2], S. Cenk Sahinalp[3,*], and Salem Malikić[3,*]

[1]Department of Electrical Engineering and Computer Sciences UC Berkeley, Berkeley, CA 94720, USA
[2]Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA
[3]Cancer Data Science Laboratory, Center for Cancer Research, National Cancer Institute, National Institutes of Health, Bethesda, MD 20892, USA
[4]Laboratory of Cancer Biology and Genetics, Center for Cancer Research, National Cancer Institute, National Institutes of Health, Bethesda, MD 20892, USA
[5]Department of Computer Science, Indiana University, Bloomington, IN 47408, USA
[‡]Now at Google LLC, Sunnyvale, CA 94089, USA
[†]Joint first authors
[*]Corresponding authors

Abstract

We introduce HUNTRESS, a computational method for mutational intratumor heterogeneity inference from noisy genotype matrices derived from single-cell sequencing data, whose running time is linear with the number of cells and quadratic with the number of mutations. We prove that under reasonable conditions HUNTRESS computes the true progression history of a tumor with high probability. On simulated and real tumor sequencing data HUNTRESS is demonstrated to be faster than available alternatives with comparable or better accuracy. Additionally, the progression histories of tumors inferred by HUNTRESS on real single-cell sequencing datasets agree with the best known evolution scenarios for the associated tumors.

Contact: cenk.sahinalp@nih.gov; malikics2@nih.gov

Preprint notice: This version of the article has been accepted for publication at Nature Computational Science, after peer review and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: https://doi.org/10.1038/s43588-022-00298-x

1 Introduction

Following the introduction of single-cell sequencing (SCS), we have witnessed major developments in computational methods for inferring mutational heterogeneity and progression history of tumors [1, 2]. Among available methods of interest, SCITE [3] and OncoNEM [4] represent the first principled computational approaches for inferring trees of tumor progression from SCS data, both employing the infinite sites assumption (ISA). More recent methods, SiFit [5] and SiCloneFit [6], consider a more general finite sites model of tumor progression, ScisTree [7] employs neighbor-join heuristic to generalize SCITE, OncoNEM allows user defined cell and site-specific probabilities of false mutation calls, and B-SCITE [8], combines single-cell and bulk sequencing data. While the above methods search for a solution directly in the space of tumor progression histories, other methods perform search among binary matrices that correspond to tumor progression trees [9]; these combinatorial optimization based techniques include SPhyR [10], PhISCS [11], scVILP [12], PhISCS-BnB [13] and gpps [14].

Despite all the progress, emerging, larger scale datasets provide a challenge to available methods. For example, the running time of the combinatorial optimization based techniques above increase exponentially with n, the number of cells and m, the number of mutations, as the number of constraints and variables in their formulation are polynomial in n and m (respectively $O(nm^2)$ and $O(nm + m^2)$ [9]). As a result these methods can not handle datasets consisting of several hundreds of mutations and cells. In fact the problem of tumor progression tree reconstruction is NP-hard [10] and it is unlikely that a polynomial time solution could handle all input types and parameters.

In this work we introduce HUNTRESS (Histogrammed UNion Tree REconStruction Scheme), a computational method with a running time that is linear with the number of cells, n and quadratic with the number of mutations, m. We show that under some reasonable conditions on the ground truth and the observed input genotype matrix, HUNTRESS infers a tree that closely matches the ground truth with high probability. Such a guarantee can not be provided by many of the alternative, e.g., deep-learning based approaches [15].

We have compared HUNTRESS with a number of available alternatives with respect to running time and accuracy on both simulated and real tumor datasets. HUNTRESS has comparable or better accuracy than the best available methods on these datasets - especially those with a higher number of cells than mutations - while being substantially faster. We have also demonstrated it to be robust to additional sources of noise such as doublets and loss of heterozygosity events (LOH). With all these features, we believe HUNTRESS offers a timely advance in the resolution of intratumor heterogeneity and tumor progression history on datasets with increasing scale and complexity.

2 Results

We first present results of HUNTRESS on a previously published [16] high-grade serous ovarian cancer (HGSOC) dataset consisting of 891 cells and a total of 14,068 SNVs. The dataset is characterized by a very low coverage and high missing entry rate (i.e., per cell, the presence/absence of ∼83% of mutations could not be determined) making it very challenging to perform tree reconstruction. The thorough analysis of the evolutionary history conducted in the original study includes pre-clustering of cells in order to reduce the effect of low sequencing coverage and obtain more reliable mutation calls. This step resulted in 9 clusters representing distinct clonal populations of cells, and the reported tumor progression tree relating them is given in Figure 1a .

In our analysis, in order to focus on the most informative mutations, we removed those mutations for which more than 650 of the sequenced cells have missing entries, as well as the mutations that are present in fewer than 10 cells. The resulting genotype matrix contains 891 cells and 744 mutations with an improved (i.e., lower) missing entry rate of ∼69%. The tree inferred in the original study, constrained on this reduced size but denser matrix, is shown in Figure 1b. The results obtained by HUNTRESS on this matrix are shown in Figure 1c. As can be seen, the placement of the cells in this tree largely matches the placement of the cells in the tree reported in the original study. In particular, cells from clones G,H,I and E,F are well separated. These results are further supported by high ancestor-descendant and different-lineages measures (see Supplementary Section 1.6 for definitions) between trees in Figure 1b and Figure 1c, which equal 0.9769 and 0.9986, respectively. Note that HUNTRESS was able to achieve this in less than 30 minutes, when 8 threads (on 4 cores) were used.

Next, we tested HUNTRESS on two targeted single-cell DNA sequencing datasets, each consisting of several thousands of single cells, from recently published study which involved 123 acute myeloid leukemia (AML) patients [17]. We selected two of the patient datasets which had the largest number of detected somatic mutations and compared results obtained by HUNTRESS to those inferred by SCITE and reported in the original study. As can be seen in Figure 2, the trees inferred by HUNTRESS are identical to trees reported by SCITE - with the exception of two minor differences in the trees shown in Figure 2a and Figure 2b: (1) Mutation `DNMT3A_1` is the ancestor of

mutation ASXL1 in the SCITE tree while they are merged in the tree inferred by HUNTRESS. (2) This is also the case for mutations STAG2 and BCOR_1. Both of these differences are due to HUNTRESS relying on the assumption that each subclone is represented by a non-trivial number of cells in the input. Here, because the number of cells having only these mutations is small (see Figure 3f in the original study[17]), they were combined to form a single subclone in the HUNTRESS tree.

Next, we compared the running time as well as the inference accuracy of HUNTRESS against several published tools, namely ScisTree [7], gpps [14], SPhyR [10] and SiCloneFit [6], on both HGSOC and AML datasets. Among other well known tools, OncoNEM crashes on large input sizes as reported previously [6, 11], while optimization methods including SiFit, SCITE, PhISCS and PhISCS-BnB or deep learning techniques [15] are too slow to handle both datasets. We ran each tool on each dataset for up to 48 hours on the LBNL NERSC cluster. (SiCloneFit was run with the default number of MCMC iterations. For additional results with more iterations see Supplementary Section 1.9. Also see Supplementary Section 1.8 for details involved in running each tool.

On the AML datasets with large number of cells but small number of mutations HUNTRESS was more than three orders of magnitude faster than alternatives; see Supplementary Table 1. On the HGSOC dataset, HUNTRESS produced an output in ~ 6 minutes, while SPhyR produced a trivial output (comprised of a a single node), and others did not terminate. In order to assess the alternative tools' accuracy and running time on a reasonably large, real dataset, we reduced the number of cells and mutations in this dataset randomly, by a factor of 3, without altering the tree topology (i.e. leaving at least one cell per node and one mutation per edge). On this sparsified dataset, ScisTree and SiCloneFit produced an output, albeit being substantially slower (respectively $3\times$ and $8\times$) than HUNTRESS, but gpps (being ILP based) could not.

With respect to accuracy, we compared the output of each tool to the trees published in the original studies, using ancestor-descendant (AD) and different-lineages (DL) accuracy measures (see Supplementary Section 1.6 for definitions). HUNTRESS was the most accurate on all four datasets - except the first AML dataset where it was slightly outperformed by gpps, an ILP tool which is prohibitively slow on datasets with many mutations (e.g. the HGSOC datasets). On this dataset, the low prevalence of cells harboring mutations specific to two subclones, namely $\{DNMT3A, ASXL1\}$ and $\{STAG2, BCOR\}$ prompted HUNTRESS to merge them, while they were separated in the original study (see Figure 2).

HUNTRESS guarantees theoretical optimality on input datasets with no false positives, thus we first benchmarked it on simulated data with only false negatives. For these experiments, we simulated data consisting of 100 to 300 cells and a similar number of mutations, with false negative rates varying between 0.05 and 0.20. We tested HUNTRESS against SPhyR, ScisTree, as well as PhISCS-BnB, the fastest combinatorial optimization method that provides accuracy guarantees on datasets with no false positives. Note that, since it is not designed to handle false positives, PhISCS-BnB was not applied to real tumor datasets or simulated datasets that do feature false positives. We allowed each tool to run up to 8 hours on each instance. Due to their long running times, gpps and SiCloneFit could not complete most of the tasks so they are not included here. Our benchmarking results provided in Extended Data Figures 1, 2, 3 (as well as Supplementary Table 2) clearly demonstrate that HUNTRESS is faster than all other tools by a factor of 10 to 1000 and is at least as accurate.

We performed several experiments on simulated data with both false positives and false negatives. In our first set of experiments the number of mutations, m, and cells, n, both varied between 300 and 1000 reminiscent of their sizes in the HGSOC datasets. We had two settings for fn, false negative entry rate: $fn = \{0.05, 0.2\}$, while the false positive and missing entry rates were set to 0.001 and 0.05, respectively. In Extended Data Figures 4, 5, 6 (as well as Supplementary Table 3) we give the AD and DL accuracy measure distributions, together with the running times for each tool on each parameter setting. Note that for each of the experiments, each tool was allowed to run for 48 hours, the maximum time allowed for a single job in the LBNL NERSC cluster.

Among the tools tested, SPhyR's AD performance is poor, even though it achieves comparable DL values to HUNTRESS. This is due to the fact that SPhyR mistakenly infers a star-like topology (with leaf nodes all connected directly to the root) for all inputs. SPhyR is faster than HUNTRESS when the number of cells are small however its low accuracy values makes it not a good choice for datasets with similar or larger sizes. (SPhyR's reported performance is based on default parameters; see Supplementary Section 1.8.4 on how SPhyR's performance depends on parameter choices.)

With respect to accuracy, HUNTRESS is comparable to or better than ScisTree; HUNTRESS performs especially well when $n > m$ - this is the setting for which HUNTRESS provides accuracy guarantees. It must be considered that HUNTRESS achieves these figures while being substantially faster, especially when $n \geq m$. As sequencing costs decrease, we expect that sequencing experiments involving thousands of single cells will become a common practice in the near future. The above results suggest that HUNTRESS could provide the necessary runtime improvement for highly accurate tumor progression tree reconstruction on emerging datasets.

Next, we assessed the robustness of HUNTRESS to the presence of deletion events, a source of 'biological'

noise which is the main cause of ISA violations. As can be be seen in Supplementary Figure 2, deletion events only slightly impact the performance of HUNTRESS with respect to the AD and DL measures. We also assessed the impact of doublets (i.e. two cells sequenced as one) as a source of noise; as per Extended Data Figure 7 and Supplementary Table 6, HUNTRESS' performance is again minimally impacted by doublets.

To further demonstrate robustness of HUNTRESS, we also evaluated its performance against published tools on (i) larger datasets with $n = 5000$, $m = 500$ (see Extended Data Figure 8 as well as Supplementary Table 4 for details), and (ii) datasets with higher false positive rates of 0.003 (see Extended Data Figure 9 as well as Supplementary Table 5 for details). Finally, (iii) we benchmarked HUNTRESS on simulations with parameters resembling those we observed on the AML dataset, generated by the Tapestri platform; we used an external simulator for these experiments, which was employed in earlier studies [4] (see Extended Data Figure 10 and Supplementary Table 7 for details). Our results on all these simulations illustrate that HUNTRESS consistently outperforms all available alternatives and its comparative advantage improves as the number of cells increases.

3 Discussion

There are a number of potential avenues for algorithmic improvement. Currently, HUNTRESS assumes uniform noise rates across all cells and mutation sites; this can be extended to account for read count data at each mutation locus in each single cell [18, 12, 7]. It may also be possible to cluster cells based on their mutational profiles prior to or during tree reconstruction for the purpose of obtaining more reliable trees - especially when the per-cell read coverage is limited. Another way to address high noise levels in single-cell sequencing data could be through the integration of bulk sequencing data into the existing model. This approach already showed promising results with tools operating on smaller scale datasets [8, 11].

4 Methods

In this work we introduce three distinct but related computational methods for cancer progression analysis. The first is a combinatorial algorithm for finding the tree of tumor progression based on mutation calls in single cells in the absence of false positives. We show that this algorithm is optimal, i.e. given a noisy genotype matrix as the input, it computes the smallest number of false negative corrections required to convert it to a conflict-free (output) genotype matrix from which a tree of tumor progression can be directly obtained[19, 9]. Since the problem solved by this algorithm is NP hard[10], it is not surprising that its worst case running time is exponential with the input size and, as a result, it is of limited practical interest. However, it forms the basis of our second, but arguably the main contribution, which is a very efficient algorithm that computes the optimal solution with high probability, provided some reasonable conditions on the input genotype matrix are satisfied. Importantly, the running time of this algorithm is only quadratic with the number of mutations and linear in the number of cells. To the best of our knowledge, this is the first polynomial time algorithm to optimally solve the tumor progression history inference problem under reasonable conditions on the input genotype matrix. Our final contribution is an extension to the second algorithm to account for the three most commonly observed types of noise in single-cell sequencing data, i.e., false positives, false negatives and missing entries. The resulting approach, which we call HUNTRESS, is a fast computational method for inferring progression histories of tumors from large scale single-cell DNA sequencing data. Below, after providing some introduction on the tree representation of tumor progression history and details of the notation that we use, we present the three methods in the order listed above.

4.1 Tree representation of tumor progression history

Assume that we have performed a single-cell DNA sequencing experiment in which single cells $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$ were sequenced. Let $\mathbf{M} = \{M_1, M_2, \ldots, M_m\}$ denote the set of single-nucleotide variants that were reported in the variant calling step as present in at least one of the sequenced cells. Assume for simplicity that each pair of a parent and a daughter cell differ by at most one mutation and that each of mutations from $\mathbf{M}$ is indeed present in at least one tumor cell (we make these assumptions solely for the purpose of simplifying description in this section and they are not used by our algorithm). Then the history of tumor progression can be represented by a mutation tree, which is a rooted tree having $m + 1$ nodes in which each node, except the root, is labeled by exactly one mutation from $\mathbf{M}$ [3]. Each node of T can also be associated with a unique integer v from the set $\{0, 1, 2, \ldots, m\}$ and a genotype g_v, which is a binary vector of size m such that $g_v[i] = 1$ if and only if M_i is a label of some node that belongs to the path from the root of T to v (inclusively). In this work, we assume that the infinite sites assumption holds, which, together with the above assumption that each non-root node has a mutational label,

guarantees that each g_v is unique. In addition, we assume that the root of a mutation tree is mutation-free and represents a population of healthy cells.

As each of the sequenced single cells originates from one of the nodes of the tree, it can be associated with a unique binary genotype g_i for some $i \in \{0, 1, 2, \ldots, m\}$. Genotypes of all sequenced cells can then be arranged in a (true) *genotype matrix* G, which is a binary matrix with n rows (cells) and m columns (mutations). The value $G[i, j]$ is equal to 1 if and only if mutation M_j is present in cell $\mathcal{C}_i$. Due to technical limitations of single-cell sequencing, which yields data characterized by various types of noise, some mutation calls made from raw SCS data do not reflect the true status of a mutation in a cell. In other words, what we observe in practice as an output of SCS experiment and data processing step is a noisy genotype matrix I, which for the sake of simplicity we assume is of the same dimensions as G, but the two matrices typically differ at a number of entries. In I, the value of $I[i, j]$ is set to 1 if and only if during the mutation calling step mutation M_j is reported as present in cell $\mathcal{C}_i$. In some cases there might be insufficient read count information to call presence or absence of a mutation in some cell so some of the entries of I are regarded as *missing* and their value is set to "?".

The presence of false mutation calls typically prevents direct reconstruction of the tumor progression history from the matrix I. In other words, there does not exist a mutation tree T such that each row of I matches genotype of some node of T. Namely, as it was previously shown [19], for a given binary matrix Z with rows corresponding to cells and columns corresponding to mutations, such a tree exists if and only if Z satisfies *three-gametes rule*, that is, there does not exist a pair of columns (i, j) and a triplet of rows (a, b, c) in Z such that

$$\begin{bmatrix} Z[a,i] & Z[a,j] \\ Z[b,i] & Z[b,j] \\ Z[c,i] & Z[c,j] \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

If such a triplet of cells and a pair of columns exists, we refer to it as a *conflict* and thus we also call any matrix satisfying three-gametes rule as a *conflict-free matrix* and this term will be used more frequently in the description provided below.

Now, given the observed noisy genotype matrix I, our goal is to *flip* some entries of I in order to obtain a conflict-free matrix Y. Here, by *flipping* an entry $I[i, j]$ we refer to setting $Y[i, j] = 1$ when $I[i, j] = 0$ or setting $Y[i, j] = 0$ when $I[i, j] = 1$ and these two flip types respectively represent corrections for false negative and false positive mutation calls in I. When searching for conflict-free matrix Y, in order to find Y which implies the most likely tree of tumor progression, our goal is typically to minimize (the weighted) number of corrections (flips) used in obtaining Y from I[9].

While in SCS data false negatives are present even at rates of 0.10 (i.e., 10%) or higher, the false positive rates of the existing datasets are much lower and are typically below 0.01. For these reasons, in this work we will first focus on the case where only the presence of false negative mutation calls in I is considered when making corrections in order to obtain Y. For this problem, we first provide a simple, but recursive, algorithm and show that it finds the optimal conflict-free matrix Y. We then present a fast algorithm that runs in $O(nm^2)$ time and $O(nm)$ space. In addition, we also show that the matrix Y returned as the output of this algorithm is conflict-free and, with high probability, the tree implied by it matches the ground truth tree in terms of branching events (i.e., achieves perfect different-lineages accuracy values) and does not swap the order of any pair of mutations that are in ancestor-descendant dependency in the ground truth tree.

4.2 Notation

In this work, for an arbitrary matrix B with n rows and m columns, for $j \in \{1, 2, \ldots, m\}$ we denote the j-th column of B as B_j. In other words, $B_j = [B[1, j], B[2, j], \ldots, B[n, j]]^T$. Next, we introduce a function $\mathcal{S}$ defined on the set of column vectors such that, for an arbitrary column vector v, $\mathcal{S}(v)$ equals the set of indices i for which $v[i] = 1$, where 1-based indexing is used. More formally, for k-dimensional vector $v = [v[1], v[2], \ldots, v[k]]^T$, we have $\mathcal{S}(v) = \{i \mid v[i] = 1\}$. As an example of the use of the above notation, note that $\mathcal{S}(I_j)$ represents the set of indices of cells $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$ in which mutation M_j was reported to be present in the observed genotype matrix I. In other words, $\mathcal{S}(I_j)$ is the set of all integers i such that $1 \leq i \leq n$ and $I[i, j] = 1$. With the slight abuse of notation, we introduce a function $\mathcal{S}^{-1}$ defined on the pairs (A, k), where k is a positive integer and A is a subset of $\{1, 2, \ldots, k\}$. For a pair (A, k) given as an argument, $\mathcal{S}^{-1}(A, k)$ is defined to be equal to a binary column vector v of length k with $v[i] = 1$ if and only if $i \in A$. For the convenience of notation, we also allow the use of $\mathcal{S}^{-1}(A)$ with the same meaning as $\mathcal{S}^{-1}(A, n)$, where n denotes the number of cells in the input matrix I, as defined earlier. Lastly, we introduce flips counting function $\mathcal{F}$, defined on a pair (A, B) of matrices (or vectors) of the same dimension. This function returns the total number of entries (i, j) for which $(A[i, j], B[i, j]) = (0, 1)$ or $A([i, j], B[i, j]) = (1, 0)$.

4.3 A constrained exhaustive search algorithm for genotype matrix reconstruction

In this section we discuss a provably optimal algorithm, referred to as Supplementary Algorithm 1 (complete pseudocode is available in the Supplementary Materials), for tumor progression history reconstruction under the assumption that I has no false positives and given that our goal is to find a conflict-free matrix that differs from I by the smallest possible number of flips. Note that our objective is then equivalent to the following: convert I to conflict-free matrix Y by the use of the smallest number of 0 to 1 flips.
Our algorithm utilizes the following well-known property of a conflict-free binary matrix Z[19]:

Lemma 1. *Let Z be a binary conflict-free matrix. Then for any pair of columns Z_i and Z_j in Z, one of the following must be true:*

1. $\mathcal{S}(Z_i) \subseteq \mathcal{S}(Z_j)$
2. $\mathcal{S}(Z_j) \subseteq \mathcal{S}(Z_i)$
3. $\mathcal{S}(Z_i) \cap \mathcal{S}(Z_j) = \emptyset$.

Let us now consider the relation between the above lemma and the tree implied by Z. We assume that Z contains no columns having all entries equal to 0 (columns having all entries equal to 0 are usually filtered from Z as they imply that the corresponding mutations are absent in all single cells). The above lemma implies that for a given conflict-free genotype matrix Z and mutations M_i and M_j corresponding respectively to the columns Z_i and Z_j, the sets of cells, indexed by $\mathcal{S}(Z_i)$ and $\mathcal{S}(Z_j)$, in which these mutations are present, are either (i) disjoint, and in this case mutations belong to different branches of a tree implied by Z, or (ii) one is a superset of the other, and in this case mutation corresponding to the larger set is ancestor of the other mutation, or (iii) $\mathcal{S}(Z_i)$ and $\mathcal{S}(Z_j)$ are equal, and in this case mutations M_i and M_j occur for the first time at the same node of the tree. Note that in the description provided here, in order to simplify presentation, we typically do not consider the third case, although in all of our experimental results we take into account this possibility.

Supplementary Algorithm 1 represents a top down approach to correct the errors in the input genotype matrix I so that, under the assumption that I has no false positives, the output genotype matrix Y contains no conflicts.

This algorithm starts with a set consisting of (an index of) some column $U = \{i\}$ (i.e., mutation M_i) of the input matrix I. It then considers the set of cells in which mutation M_i is present and searches for a mutation M_j such that there exists a cell in which both M_i and M_j are present. If such cell exists, the set U is extended by adding j to it. We observe that, due to the assumption that I does not contain false positive mutation calls (i.e., no $1 \rightarrow 0$ flips are allowed), it follows that in any conflict-free matrix Z obtained from I by the use of $0 \rightarrow 1$ flips only, we must have $\mathcal{S}(Z_i) \cap \mathcal{S}(Z_j) \neq \emptyset$. Therefore in any such matrix Z, due to the above lemma, we must have $\mathcal{S}(Z_i) \subseteq \mathcal{S}(Z_j)$ or $\mathcal{S}(Z_j) \subseteq \mathcal{S}(Z_i)$. As the absence of false positives obviously implies $\mathcal{S}(I_i) \subseteq \mathcal{S}(Z_i)$ and $\mathcal{S}(I_j) \subseteq \mathcal{S}(Z_j)$, we can easily conclude that at least one of the following must hold true: $\mathcal{S}(I_i) \cup \mathcal{S}(I_j) \subseteq \mathcal{S}(Z_i)$ or $\mathcal{S}(I_i) \cup \mathcal{S}(I_j) \subseteq \mathcal{S}(Z_j)$. Similarly, if there exists a mutation M_k such that M_i and M_k are present in the same cell or M_j and M_k are present in the same cell (which is equivalent to $\mathcal{S}(I_k) \cap (\mathcal{S}(I_i) \cup \mathcal{S}(I_j)) \neq \emptyset$) we extend U so that it equals $\{i, j, k\}$ and observe that at least one of the following must hold true: $\mathcal{S}(Z_i) \subseteq \mathcal{S}(Z_k)$, or $\mathcal{S}(Z_k) \subseteq \mathcal{S}(Z_i)$, or $\mathcal{S}(Z_j) \subseteq \mathcal{S}(Z_k)$, or $\mathcal{S}(Z_k) \subseteq \mathcal{S}(Z_j)$. In any case, we can conclude that for at least one $l \in \{i, j, k\}$, in any conflict free matrix Z obtainable from I by the use of $0 \rightarrow 1$ flips only, we have $\mathcal{S}(I_i) \cup \mathcal{S}(I_j) \cup \mathcal{S}(I_k) \subseteq \mathcal{S}(Z_l)$. The search is then continued, each time looking for a column v such that $\mathcal{S}(I_v)$ has non-empty intersection with union of sets $\mathcal{S}(I_u)$ where $u \in U$. Eventually at some point no further extensions of the set U will be possible. Then, based on what was shown above, we must have $\cup_{u \in U} \mathcal{S}(I_u) \subseteq \mathcal{S}(Z_v)$, for some entry $v \in U$. In other words, in any conflict-free matrix Z that is a candidate solution for our problem, it is necessary that at least one of the columns Z_v, for $v \in U$, has 1's in all rows (cells) in which at least one of I_u, for $u \in U$, has 1.

Observe that reordering columns and rows of I together with their mutational and cellular labels in principle does not impact the tree reconstructed from I. So, let us reorder the columns of I such that the columns corresponding to mutations M_i, where $i \in U$, come first from the left and rows having 1 in at least one of these $|U|$ columns come first from the top. Assume that there are q such rows. Then the entries at the intersection of all remaining rows (i.e., rows $q+1, q+2, \ldots, n$) and the first $|U|$ columns of (reordered) I are all zeros. What is more, due to the choice of U, the entries at the intersection of columns $|U|+1, |U|+2, \ldots, m$ and rows $1, 2, \ldots, q$ are also all equal to zero. In other words, the reordered I has the following shape

$$\begin{bmatrix} A & \mathbf{0}_2 \\ \mathbf{0}_1 & B \end{bmatrix}$$

where $\mathbf{0}_1$ and $\mathbf{0}_2$ are null matrices. Note that in the pseudocode given in Supplementary Algorithm 1, when constructing A and B from I, we do not remove rows that have all entries equal to zero. This is done solely for the

purpose of simplifying presentation, but it clearly does not affect the final result since entries of I corresponding to $\mathbf{0}_1$ and $\mathbf{0}_2$ are never flipped from 0 to 1.

Converting I to some conflict free matrix Z clearly requires resolving all conflicts in A, as well as in B. On the other hand, resolving all conflicts in A first and then separately resolving all conflicts in B will also resolve all conflicts in I (because it can be easily verified that all conditions listed in Lemma 1 will be satisfied for the resulting matrix). Therefore, we can recursively resolve conflicts in A and then in B and merge the two obtained results.

While matrix B has at most $m-1$ columns and recursive formula can be applied to it directly, it can occur that the number of columns of A equals to m (i.e., $|U| = m$), in which case A has the same number of columns as I so simple recursive call on A would result in non-terminating recursion. Observe now that: (i) $\cup_{i \in U} \mathcal{S}(A_i) = \{1, 2, \ldots, q\}$, which follows from the definition of A and (ii) If C is a conflict free-matrix that can be obtained from A by the use of smallest number of $0 \rightarrow 1$ flips only, then, as we observed above, $\cup_{i \in \{1,2,\ldots,|U|\}} \mathcal{S}(A_i) \subseteq \mathcal{S}(C_j)$ for some $j \in \{1, 2, \ldots, |U|\}$. Combining these two observations we conclude that at least one column in C consists of entries that are all equal to 1. Performing the exhaustive search over all possible columns $i \in \{1, 2, \ldots, |U|\}$ in order to find the one which has all 1's in the output matrix and using recursive calls as described in Supplementary Algorithm 1 will obviously result in finding the desired (optimal) conflict-free matrix C.

While the above approach results in a solution Y that is conflict-free and differs from I by a fewest number of $0 \rightarrow 1$ flips, it is exhaustive and has a running time $O(n(m+1)!)$ (one of the worst cases can occur for example when mutation tree implied by the optimal solution is a linear tree without any branching events).

4.4 A fast computational method for many cells

In this section, we introduce and discuss a greedy computational method, which is a slight modification of the algorithm given above, but has a running time of $O(nm^2)$ and requires $O(nm)$ space. While it does not necessarily always return an optimal output matrix (i.e., closest to the input matrix I in terms of the total number of $0 \rightarrow 1$ flips), we show that, if some reasonable assumptions are satisfied, the tree implied by the output matrix, with high probability, matches the ground truth with perfect different-lineages (DL) accuracy measure (i.e., two mutations belonging to different lineages in the true tree also belong to different-lineages in the tree implied by solution returned from the algorithm) and any pair of mutations that have an ancestor-descendant relationship in the ground truth tree will be either in ancestor-descendant order in the inferred tree or clustered together (i.e., reported as present in the same set of cells).

The new computational method is highly similar to Supplementary Algorithm 1. The main difference is that, instead of performing an exhaustive search over all columns of A (line 13 in Supplementary Algorithm 1), we greedily choose a column A_i with the highest number of 1's and set the corresponding column of the output matrix Y to $\mathcal{S}^{-1}(\cup_{j \in U} \mathcal{S}(I_j))$. We then recursively apply our method to matrices B and $A^{(i)}$ and merge the obtained results. The pseudocode of this algorithm, which we refer to as Supplementary Algorithm 2, is provided in the Supplementary Materials.

As mentioned earlier, due to the greedy choices that it makes, Supplementary Algorithm 2 does not necessarily find the optimal solution. However, if the conditions listed in Assumption 1 (see below) are satisfied, then the tree implied by Y closely matches the tree implied by the ground truth genotype matrix G.

Assumption 1. *Matrix I does not contain false positive mutation calls. Furthermore, I satisfies the following for each pair of mutations M_i and M_j that are in a parent-child dependency in the ground truth tree (i.e., the node labeled by M_i is parent of the node labeled by M_j): $|\mathcal{S}(I_i)| > |\mathcal{S}(I_j)|$ and $\mathcal{S}(I_i) \cap \mathcal{S}(I_j) \neq \emptyset$.*

We will prove that the above assumption is satisfied with high probability provided that the number of sequenced single cells is much higher than the number of distinct (sub)clones in the tumor sample and the cellular prevalence of each distinct (sub)clone is non-trivial. As such, Supplementary Algorithm 2 expands the capability of PhISCS-BnB, a recent branch and bound algorithm [13] that achieves worst case optimality on genotype matrices with no false positives, on emerging single-cell tumor datasets, where the number of cells is much higher than that of mutations and thus the number of (sub)clones. Due to the space constraints, we only provide statements of the following results. Their proofs can be found in Supplementary Materials, where we also provide an extension of Supplementary Algorithm 2 to handle data containing both false negatives and false positives, as well as the missing entries (some discussion of this algorithm is also provided in Section 4.5).

Lemma 2. *Assume that we are given a tumor having the progression history that can be represented by a mutation tree in which infinite-sites assumption is satisfied. Let φ denote the fraction of the smallest population of cells in the entire tumor sample. Given that the false negative mutation detection rate is $\beta < \varphi/2e$, where e is the base of the natural logarithm, if $n \gg m$ cells are i.i.d. sampled from the tumor sample, then Assumption 1 is satisfied with "high" probability.*

Theorem 1. *Given an input matrix I that satisfies Assumption 1, Supplementary Algorithm 2 computes an output matrix Y that is conflict-free and the tree reconstructed from this matrix matches the ground truth tree in preserving all different-lineages (DL) dependencies between mutations, as well as either preserving ancestor-descendant (AD) dependencies or, sometimes, possibly merging together mutations that are in ancestor-descendant order in the ground truth (but never swapping their relative order in the tree implied by Y).*

Lemma 3. *Supplementary Algorithm 2 can be implemented in $O(nm^2)$ time using $O(nm)$ space.*

4.5 A generalized method for handling false positives and missing entries

In this section, we briefly describe how we generalize the algorithm presented in Section 4.4 to cases where the observed genotypes matrix I possibly contains some false positive mutation calls, as well as missing entries, in addition to false negative mutation calls. A detailed pseudocode of this generalized method, along with a description, can be found in the Supplementary Section 1.3. Specifically, Supplementary Algorithm 3 forms the main body of the generalized method, which searches for the best possible solution by calling Supplementary Algorithm 4 for every combination (of specified values) of the parameters λ and μ, as explained below.

In Supplementary Algorithm 2 we can call the set of columns that are united a "connected component" because they are considered to represent a subtree of the tumor progression tree. One key difference in Supplementary Algorithm 4 with respect to Supplementary Algorithm 2 is that rather than computing the union of the columns in a connected component (line 9 of Supplementary Algorithm 2) and using this as a "pivot" column, Supplementary Algorithm 4 computes the sum of each row across the columns of the connected component to create a "histogram". The column with the highest number of 1s (more specifically, expected number of 1s - after estimating the number of potential 1s among its missing entries) is then used as the pivot, and the intersection between this pivot column and every other column is computed. In case the fraction of 1s in the intersection is less than λ, where λ is one of the two key parameters we mentioned above, we remove the column (mutation) from the connected component. Similarly, if the ratio of the number of 1s in any given row with the largest row of the histogram is less than $1/\mu$, where μ is the second key parameter mentioned above, then that row is set to 0 in the columns of the entire connected component (including the pivot). As can be seen, λ is used to remove false positive columns (mutations) from the connected component, and μ is used to detect and eliminate false positive entries among the (true positive) columns of the connected component.

For each pair of values of λ and μ from a given range, Supplementary Algorithm 4 computes the output conflict-free matrix R. Then, based on the scoring function defined in the line 7 of Supplementary Algorithm 3, among all such matrices R, the best scoring matrix, denoted as R_{opt}, is selected. Given this best scoring matrix and the input matrix I, by calling Supplementary Algorithm 5, we associate each column i of the matrix I with a column t of the matrix R_{opt}. The column t is selected such that the conditional probability of observing the column I_i given that its true status matches the column t of R_{opt} is maximized. Note that we can now define a matrix Y such that its i-th column equals the column of R_{opt} that is closest (in terms of the conditional probability) to the column I_i. Analogous is done for the rows of I, and the whole process is repeated several times (each time by first updating the value of R_{opt} to Y before repeating the other steps of Supplementary Algorithm 5). Even though we do not provide theoretical optimality guarantees for the generalized version of our method, we demonstrate in the next section that it is (asymptotically) as fast, and produces highly accurate results on simulated and real genotype matrices.

Data availability

Our study makes use of two publicly available datasets introduced in previous studies [16, 17]. For the leukemia dataset, single-cell and bulk sequencing data have been deposited at NCBI BioProject ID PRJNA648656, and SNP array data with NCBI GEO ID GSE156934. For the HGSOC dataset, the single-cell FASTQs have been deposited in the European Genome-phenome Archive under accession number EGA: EGAS00001003190. The OV2295 datasets are available at Zenodo [20]. Source data for performance results for Figures 1 and 2 are available in Supplementary Table 1. Source data for Extended Data Figures 1 to 10 is available with this manuscript. Simulated data generated and used in this study for obtaining results shown in Extended Data Figures 1 to 10 are available at Zenodo [21].

Code availability

The open-source implementation of HUNTRESS is available at Zenodo [22].

Acknowledgements

This work is supported in part by the Intramural Research Program of the National Institutes of Health, National Cancer Institute (to F.R.M., E.P., K.L.M., M.P.L., C.D., G.M., S.C.S., and S.M.) and utilized the computational resources of the NIH Biowulf high performance computing cluster (http://hpc.nih.gov) and Gurobi (http://www.gurobi.com) to solve some optimization problems. Additionally, F.R.M. was supported in part by Indiana U. Grand Challenges Precision Health Initiative. C.K. and A.B were supported by the Advanced Scientific Computing Research (ASCR) Program of the Department of Energy Office of Science under contract No. DE-AC02- 05CH11231.

Author contributions

S.C.S. and A.B. jointly initiated and supervised the project. The algorithmic approach was developed by C.K. HUNTRESS was implemented by C.K., and was tested by F.R.M. Theorem 1, related lemmas and their proofs are by S.C.S., S.M., F.E. and C.K. Experimental results on real data and external simulators are by F.R.M. The internal simulator was developed and the simulated data were generated by S.M. The experimental results on the internal simulator are by C.K. and F.R.M. The bulk of paper was written by C.K., S.M., S.C.S., F.R.M., and F.E. with feedback from all co-authors. E.P, K.L.M., M.P.L., E.S.A., C.D. and G.M. contributed to the interpretation of the results and biological implications.

Competing interests

The authors declare no competing interests.

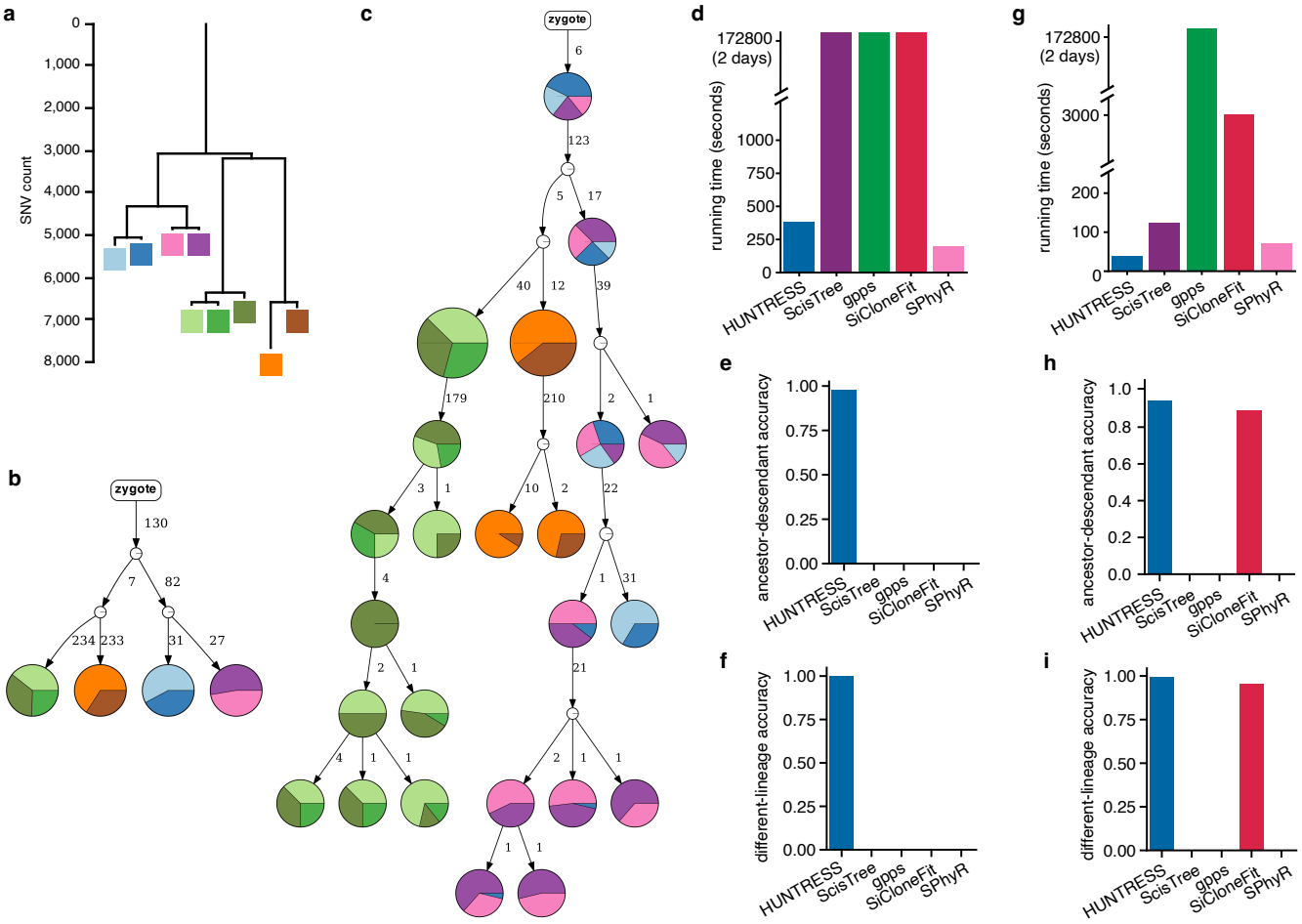


Figure 1: Analysis of high-grade serous ovarian cancer (HGSOC) dataset[16]. (a) Tumor progression tree representing 9 clonal populations of HGSOC dataset. This tree (with some minor formatting differences) was reported in Figure 3H in [16]. (b) The reduced size tree obtained after filtering from the input all mutations (columns) that have more than 650 missing entries, as well as mutations that are reported to be present in less than 10 cells (i.e., have less than 10 entries being equal to 1 in the corresponding column of the genotype matrix I). (c) Tree inferred by HUNTRESS. In (b) and (c) each edge is labeled with the number of mutations occurring between the corresponding parent and child nodes (each node represents a subclone, i.e. a distinct cellular population that have the same mutational profile). The proportion of cells in each node of the trees in (b) and (c) is color coded with respect to the leaves in (a). To better visualize the inferred tree, we collapsed most of the linear chains (i.e., chains of nodes having in/out-degrees equal to one) into single nodes. We also provide a comparison of HUNTRESS' running time and accuracy against ScisTree [7], gpps [14], SPhyR [10] and SiCloneFit [6] using ancestor-descendant (AD) and different-lineages (DL) accuracy measures (the higher the value, the better; see Supplementary Section 1.6 for definitions). The trees published by the original study are used as ground truth. (d),(e),(f) Results for the full size HGSOC data with 891 cells and 744 mutations. (g),(h),(i) Results for the reduced size HGSOC data with 297 cells and 248 mutations. The time limit for each tool was set to 48 hours. All multi-threaded tools (i.e. gpps, SPhyR and HUNTRESS) were run using 16 threads. For SiCloneFit we used 16 restarts in parallel, and reported the best result in terms of the reported likelihood. See Supplementary Table 1 for a tabular depiction of panels (d)-(i).

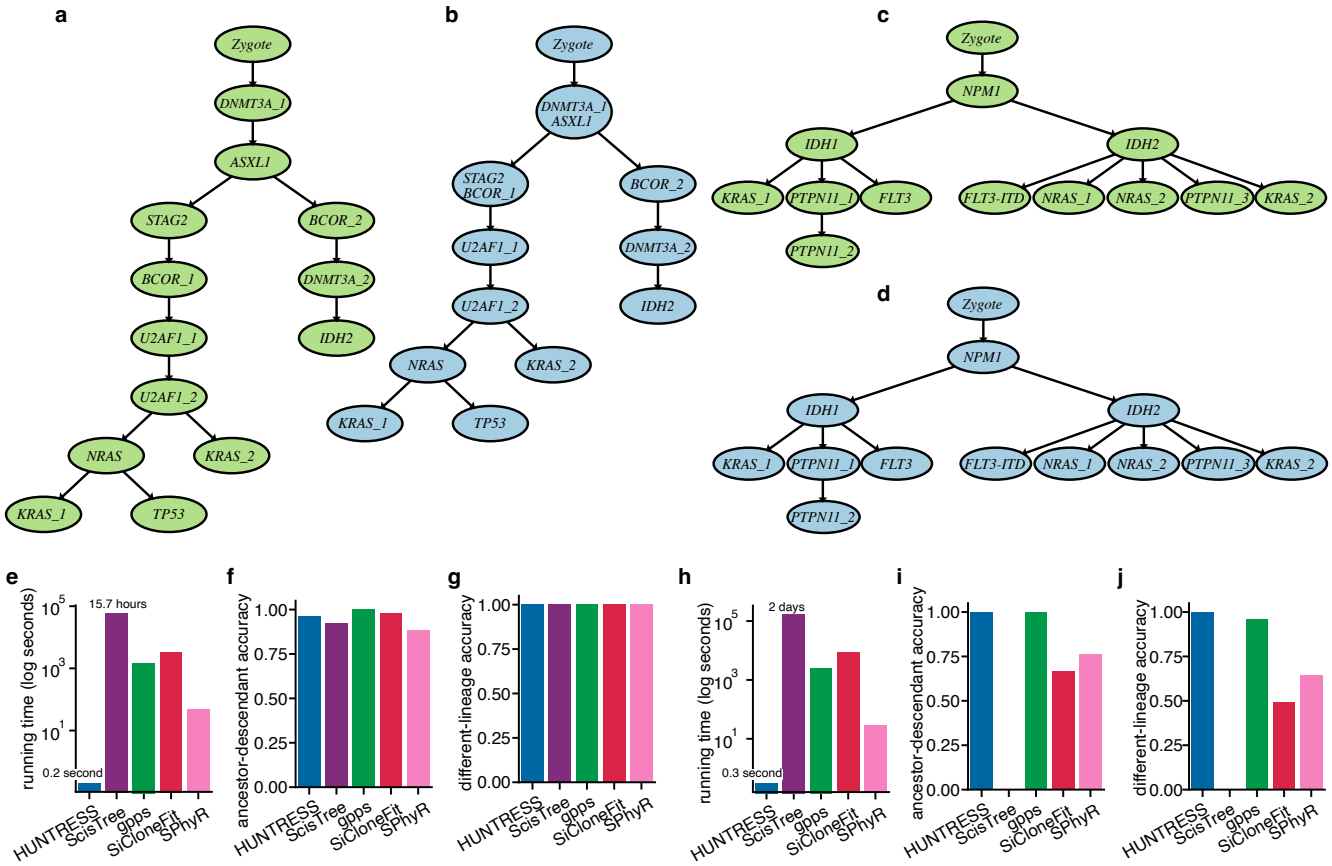
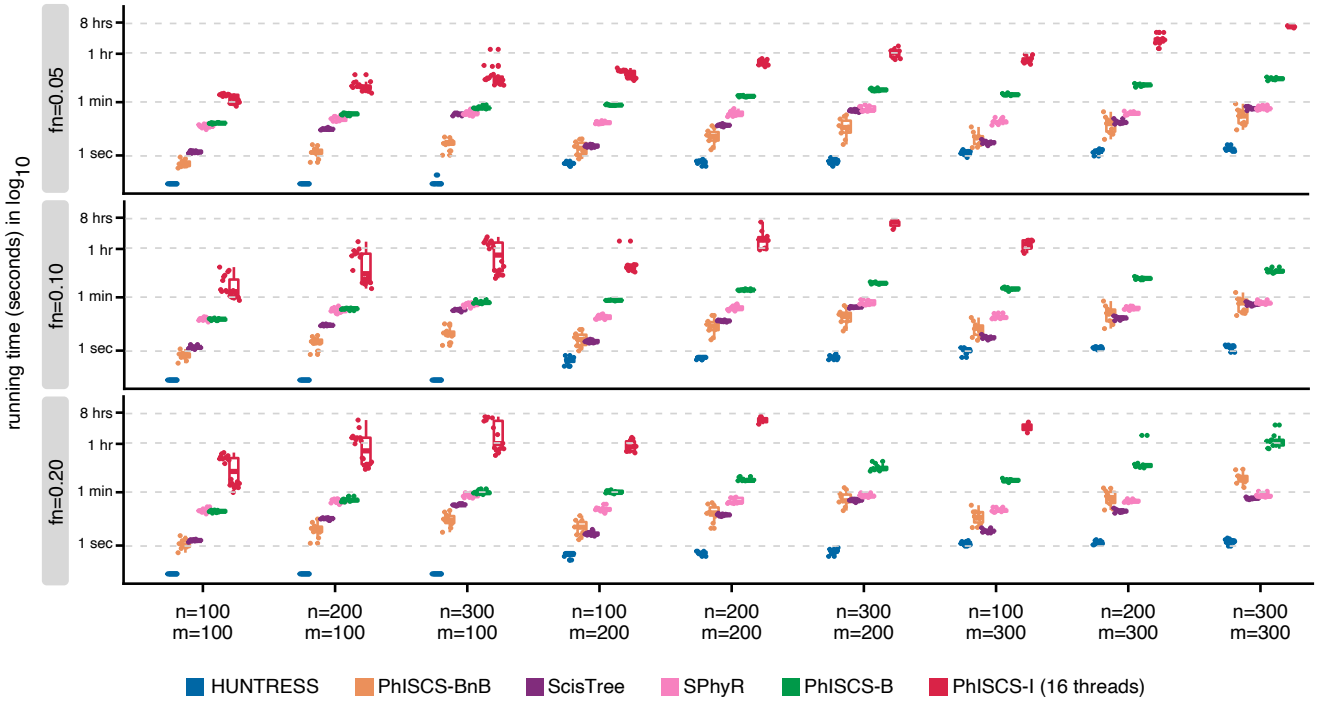
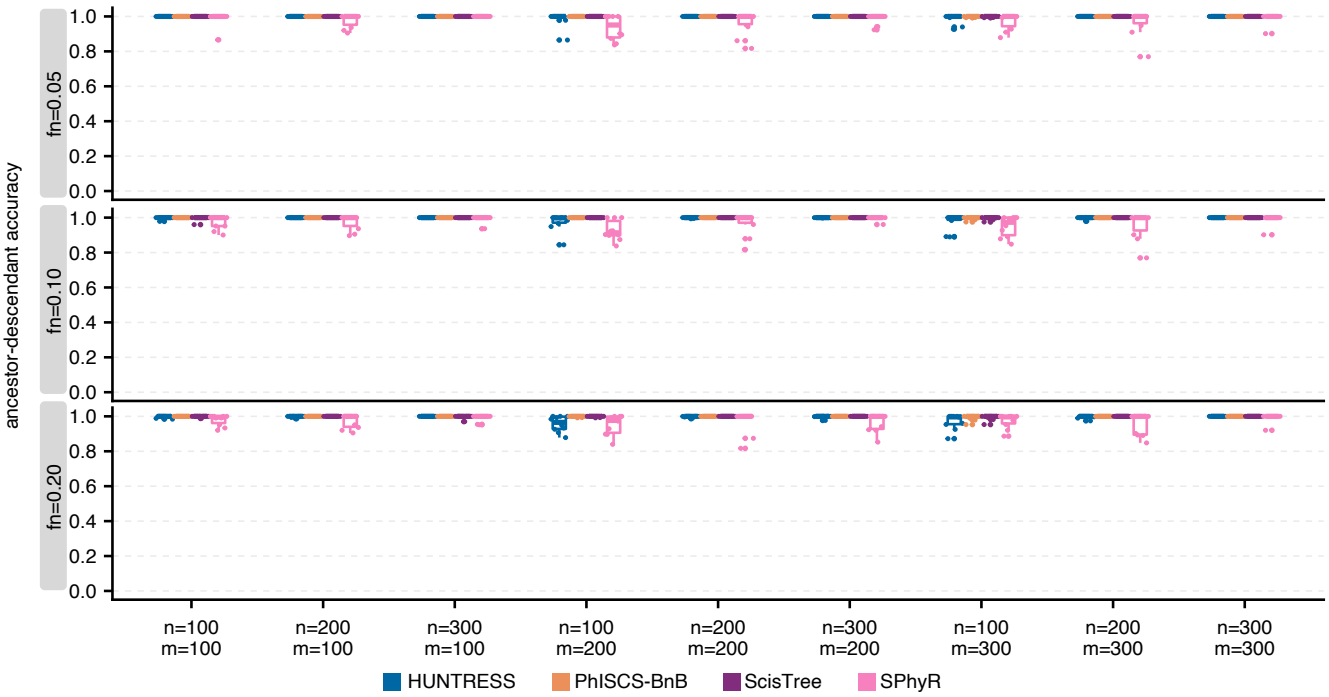


Figure 2: Analysis of acute myeloid leukemia (AML) dataset[17]. (a): The mutation tree of patient "AML-67-001" inferred by the use of SCITE[3] and reported in the original study[17]. The dataset consists of 3347 single cells sequenced using Tapestri platform for targeted single-cell DNA sequencing. In total, 13 somatic mutations were detected and used for the phylogenetic analysis. (b): The tree of tumor progression for the same patient dataset as in (a), but inferred by the use of HUNTRESS. (c): The mutation tree of patient "AML-38-001" inferred by the use of SCITE and reported in the original study[17]. The dataset consists of 6784 cells sequenced using Tapestri platform for targeted single-cell DNA sequencing. In total, 12 somatic mutations were detected and used for the phylogenetic analysis. (d): The tree of tumor progression for the same patient dataset as in (c), but inferred by the use of HUNTRESS. Note that this tree turned out to be isomorphic to the tree reported in the original study. In each tree shown in this figure, mutations are placed on the nodes (i.e., a cell population) of their first occurrence. Since the sequencing data was generated by targeting a specific set of genes, each mutation can be associated with a unique gene. However, some genes are mutated at multiple sites. In order to distinguish between multiple mutations on the same gene, we use distinct subscripts (e.g., `BCOR_1` and `BCOR_2` represent two distinct mutations in gene `BCOR`). We also provide a comparison of HUNTRESS' running time and accuracy against ScisTree [7], gpps [14], SPhyR [10] and SiCloneFit [6] using ancestor-descendant (AD) and different-lineages (DL) accuracy measures (the higher the value, the better; see Supplementary Section 1.6 for definitions). The trees published by the original study are used as ground truth. (d),(e),(f) Results for the patient AML-67-001. (g),(h),(i) Results for the patient AML-38-001. The time limit for each tool was set to 48 hours. All multi-threaded tools (i.e. gpps, SPhyR and HUNTRESS) were run using 16 threads. For SiCloneFit we used 16 restarts in parallel, and reported the best result in terms of the reported likelihood. See Supplementary Table 1 for a tabular depiction of panels (e)-(j).

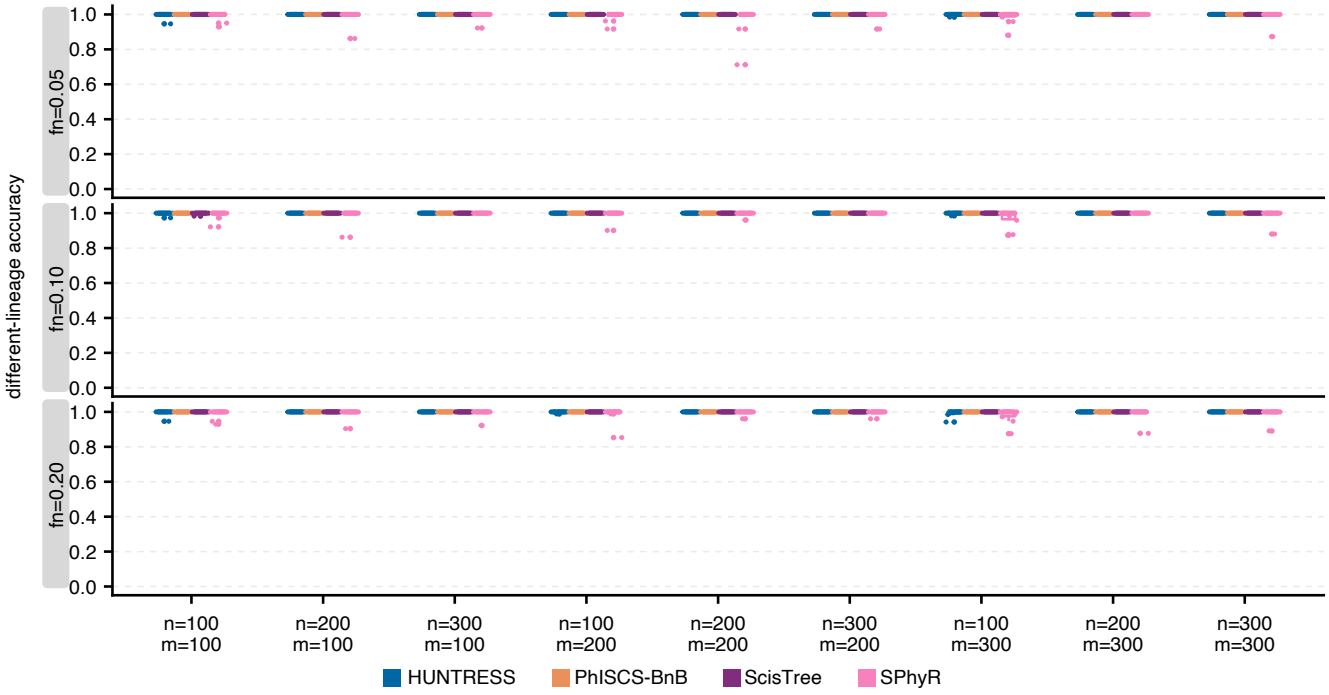
Extended Data Figures



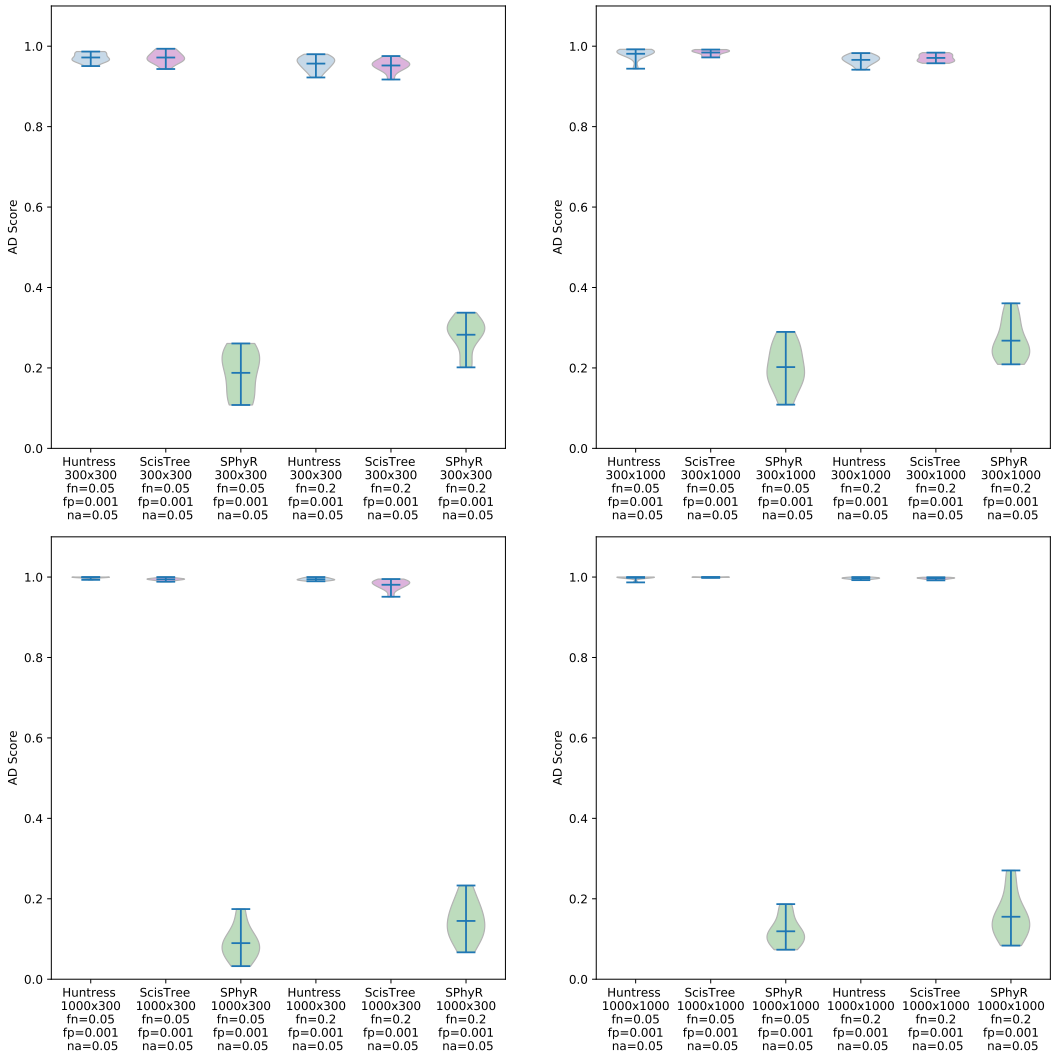
Extended Data Figure 1: A running time assessment of HUNTRESS on simulated data with no false positives, in comparison to ScisTree [7], SPhyR [10], and PhISCS-BnB [13], as well as its slower but more general variants, PhISCS-I [11], and PhISCS-B [11]. All numbers on y-axis are in log_{10} scale. Here n, m and fn, respectively, denote the number of cells, the number of mutations and the false negative error rate in single-cell data. For each setting of n and m, we report the distribution of the running time for each tool - over 10 distinct trees of tumor progression, with false negative error rates of $0.05, 0.1$ and 0.2. Each tool was run with a time limit of 8 hours (those cases that exceed the time limit are not included here). The corresponding accuracy measures for each setting are shown in Extended Data Figure 2 (ancestor-descendant accuracy measure) and Extended Data Figure 3 (different-lineages accuracy measure).



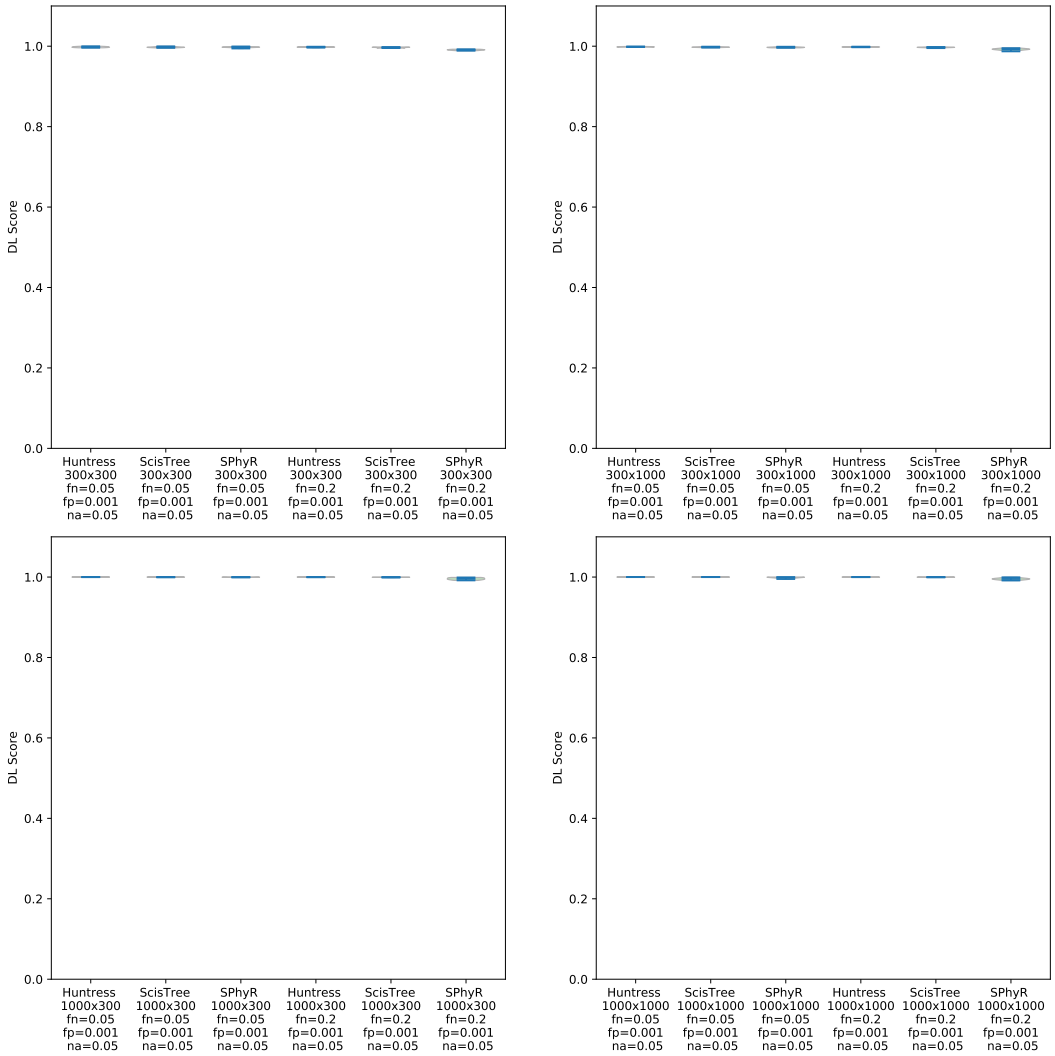
Extended Data Figure 2: Comparison of ancestor-descendant (AD) accuracy measures for HUNTRESS, PhISCS-BnB [13], ScisTree [7] and SPhyR [10] on simulated data with no false positives. Here n, m and fn, respectively, denote the number of cells, the number of mutations and the false negative error rate in single-cell sequencing. For each setting of n and m, we report the ancestor-descendent (AD) accuracy measure for each tool with respect to the ground truth. The experiments were performed over 10 distinct trees of tumor progression, using a false negative error rate of $0.05, 0.1$ or 0.2. Each tool was run with a time limit of 8 hours (those cases that exceed the time limit are not included here). Note that we have not included results of PhISCS-I and PhISCS-B as their accuracy values are identical to that of PhISCS-BnB (on these instances on which they completed the task) due to the same underlying objective function and optimality guarantee that they all provide.



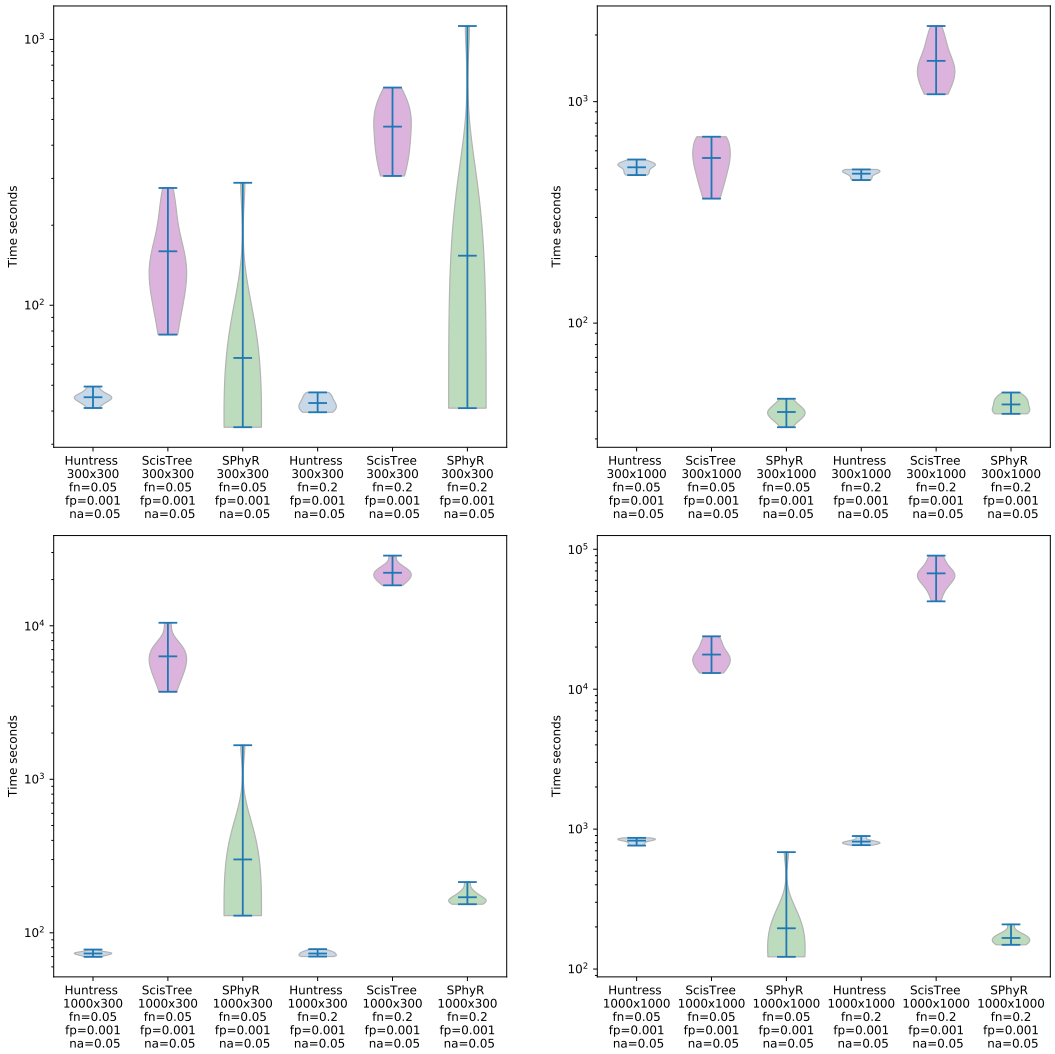
Extended Data Figure 3: Comparison of different-lineages (DL) accuracy measure distributions for HUNTRESS, PhISCS-BnB [13], ScisTree [7] and SPhyR [10] on simulated data with no false positives. Here n, m and fn, respectively, denote the number of cells, the number of mutations and the false negative error rate for single-cell sequencing. For each setting of n and m, we report the different-lineages (DL) accuracy measure for each tool with respect to the ground truth. The experiments were performed over 10 distinct trees of tumor progression, with a false negative error rate varying across $0.05, 0.1$ and 0.2. Each tool was run with a time limit of 8 hours (those cases that exceed the time limit are not included here). Note that we have not included results of PhISCS-I and PhISCS-B separately as their accuracy values match those of PhISCS-BnB (on these instances on which they completed the task) due to the same underlying objective function and optimality guarantee that they all provide.



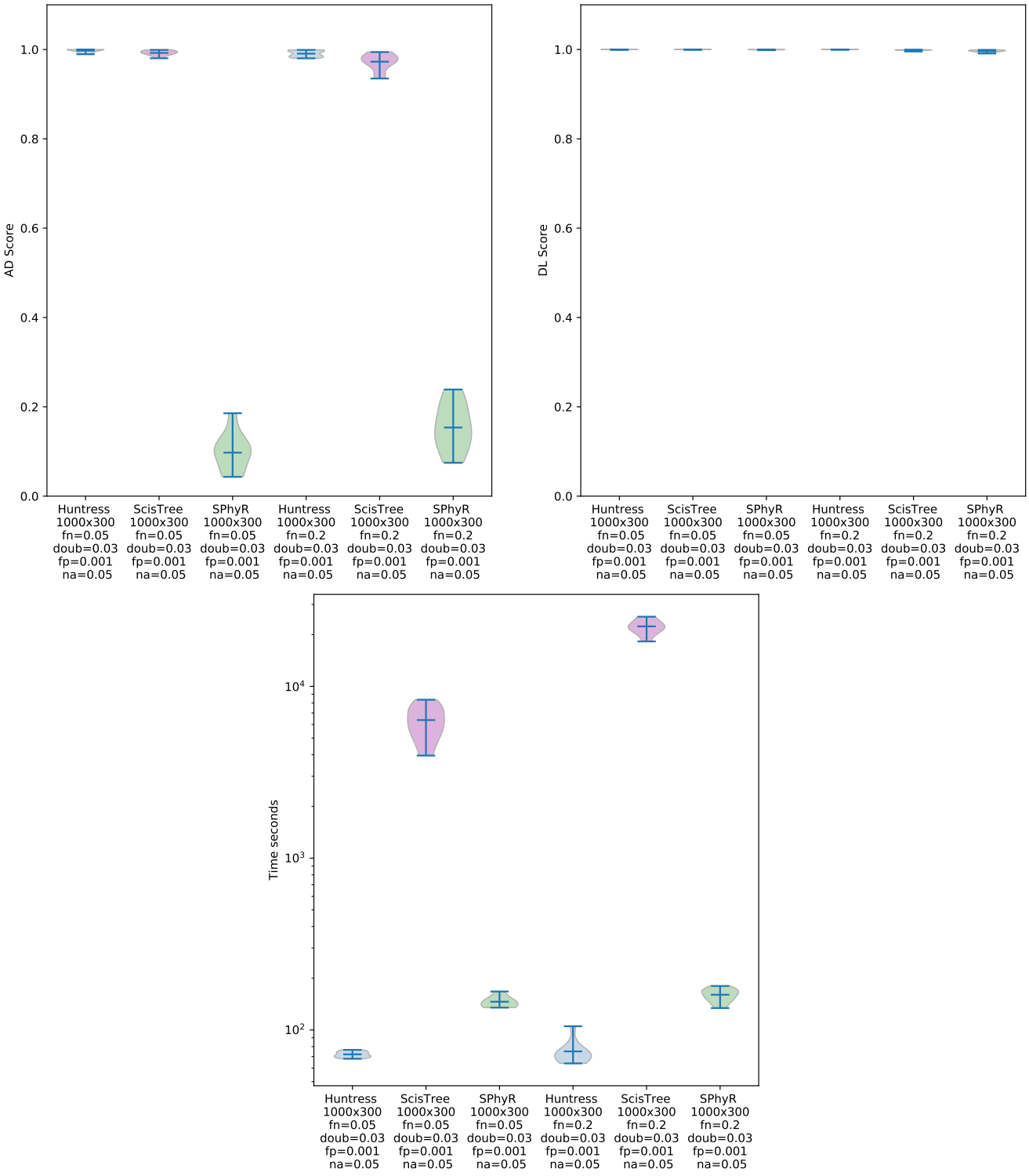
Extended Data Figure 4: Ancestor-Descendant (AD) accuracy measure distributions for HUNTRESS, ScisTree and SPhyR on simulated data with false positives, false negatives and missing entries. Here n, m, fn, fp and na respectively, denote the number of cells, the number of mutations, the false negative, false positive error and missing entry rates in single-cell sequencing data. For each setting we report the distribution for each tool over 10 distinct trees of tumor progression. Each tool was allowed to run with a time limit of 48 hours (those cases that exceed the time limit are not included here).



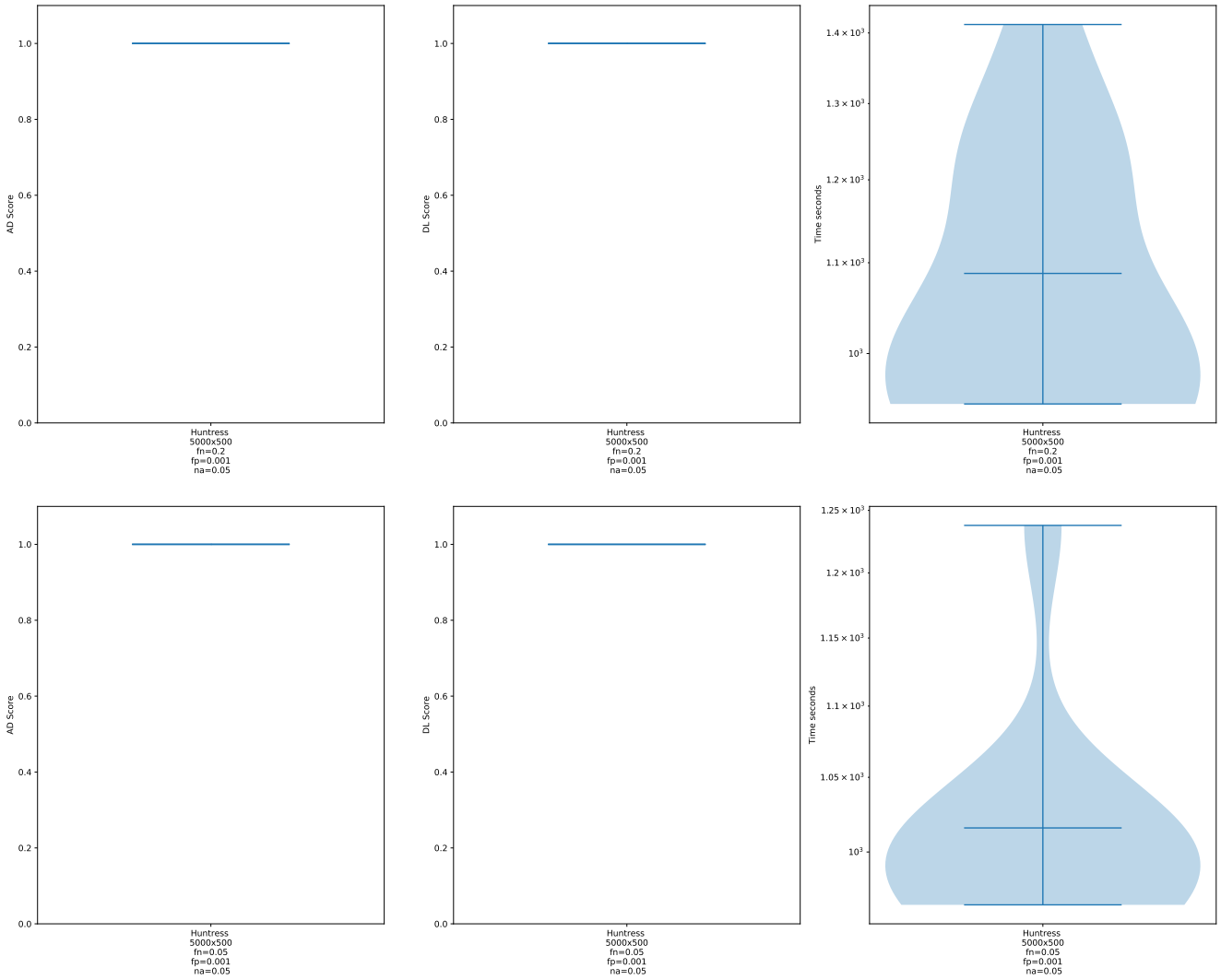
Extended Data Figure 5: Different Lineages (DL) accuracy measure distributions for HUNTRESS, ScisTree and SPhyR on simulated data with false positives, false negatives and missing entries. Here n, m, fn, fp and na respectively, denote the number of cells, the number of mutations, the false negative, false positive error and missing entry rates in single-cell sequencing data. For each setting we report the distribution for each tool over 10 distinct trees of tumor progression. Each tool was allowed to run with a time limit of 48 hours (those cases that exceed the time limit are not included here).



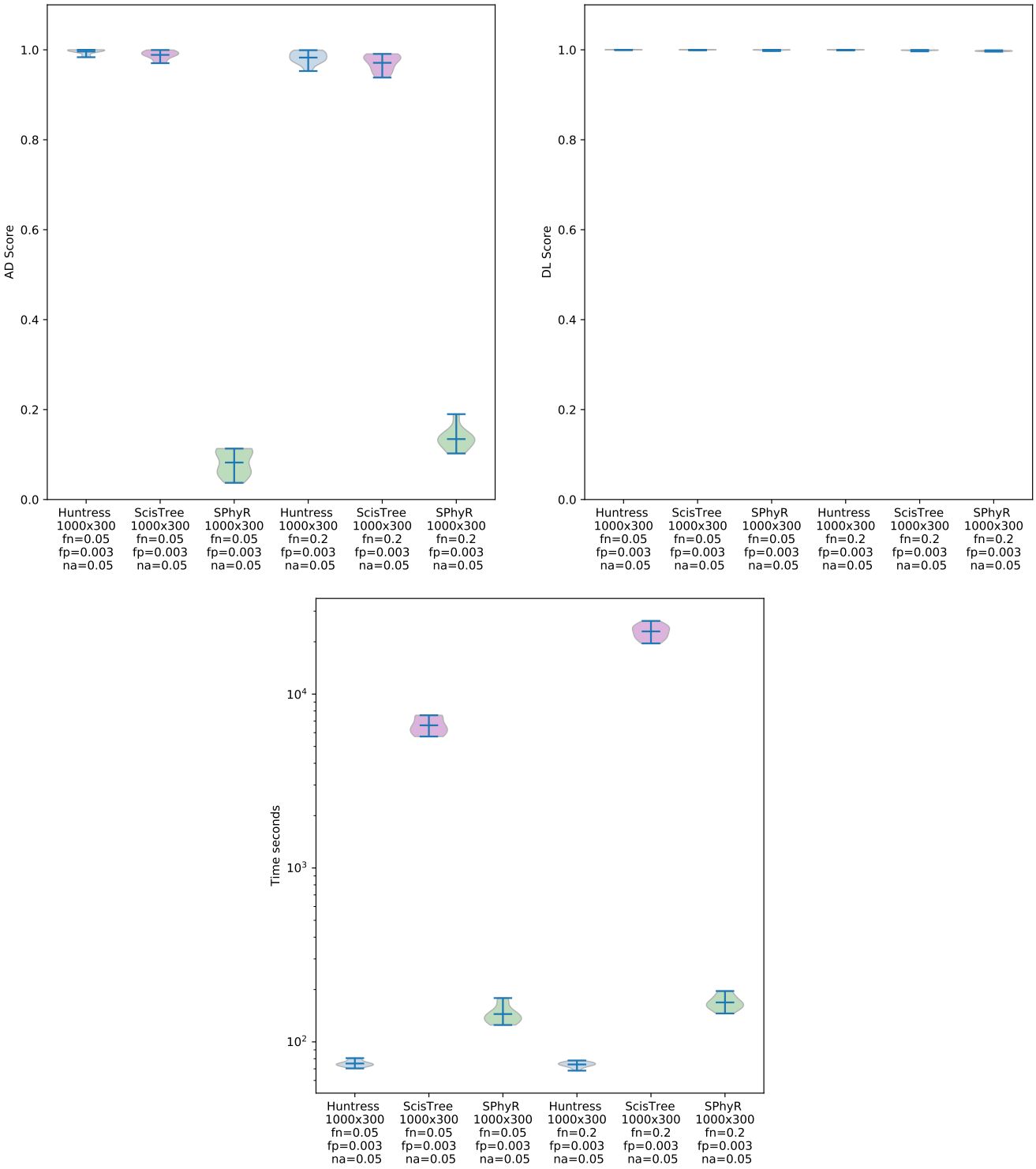
Extended Data Figure 6: Running time distributions for HUNTRESS, ScisTree and SPhyR on simulated data with false positives, false negatives and missing entries. Here n, m, fn, fp and na respectively, denote the number of cells, the number of mutations, the false negative, false positive error and missing entry rates in single-cell sequencing data. For each setting we report the distribution for each tool over 10 distinct trees of tumor progression. For any given task each tool was allowed to run with a time limit of 48 hours (those cases that exceed the time limit are not included here).



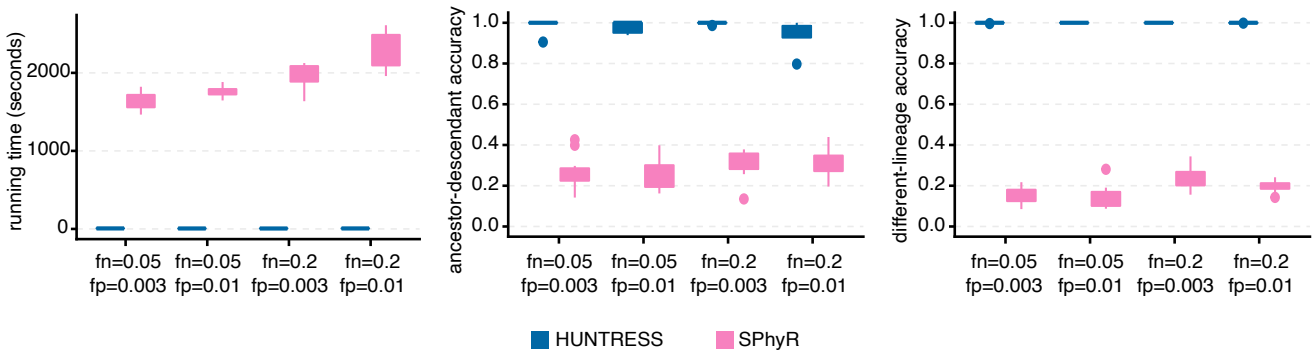
Extended Data Figure 7: Ancestor-Descendant (AD) and Different Lineages (DL) accuracy measures as well as running time distributions for HUNTRESS, ScisTree and SPhyR on simulated datasets with doublets. Here $n = 1000$, $m = 300$, $fn = 0.2$ or 0.05, $fp = 0.001$, $na = 0.05$, and the doublet rate is set to 0.03. Each distribution is over 10 distinct trees of tumor progression. For any given task each tool was allowed to run with a time limit of 48 hours.



Extended Data Figure 8: Ancestor-Descendant (AD) and Different Lineages (DL) accuracy measures, as well as the running time distributions for HUNTRESS on large simulated datasets. In these simulations we set $n = 5000$, $m = 500$, $fn = 0.05$ or 0.2, $fp = 0.001$ and $na = 0.05$. For each setting, the distributions are reported over 10 distinct trees of tumor progression. Note that because ScisTree could not finish any of the tasks within the time limit of 48 hours and SPhyR failed to generate any output, they are not presented in the figure.



Extended Data Figure 9: Ancestor-Descendant (AD) and Different Lineages (DL) accuracy measures as well as running time distributions for HUNTRESS, ScisTree and SPhyR on simulated datasets with a high(er) false positive rate of $fp = 0.003$. Here $n = 1000$, $m = 300$, $fn = 0.2$ or 0.05, and $na = 0.05$. Each distribution is over 10 distinct trees of tumor progression. For any given task each tool was allowed to run with a time limit of 48 hours.



Extended Data Figure 10: Running time, ancestor-descendant and different lineages accuracy measure distributions for HUNTRESS and SPhyR on simulations with parameters similar to those observed in the AML dataset (the Tapestri platform). Here $n = 5000$, $m = 50$, $fn = 0.2$ or 0.05, $fp = 0.01$ or 0.003 and $na = 0.1$. All simulated data used in this figure were generated by the simulator developed for OncoNEM [4]. For any given task each tool was allowed to run with a time limit of 48 hours.

5 Supplementary Materials

5.1 Pseudocode of constrained exhaustive search algorithm for genotype matrix reconstruction

Supplementary Algorithm 1 *Input:* a boolean genotype matrix $I \in \{0,1\}^{n \times m}$, which may have false negatives. *Output:* a "corrected" genotype matrix Y that contains no conflicts and differs from I by the smallest possible number of $0 \to 1$ flips among all conflict free matrices obtainable from I by the use of $0 \to 1$ flips only.

function PTREEEXACTFNONLY(I)
  $Y \leftarrow I$
  if number of columns in I is zero or one **then**
    return Y
  end if
  $E \leftarrow \{1, 2, \ldots, m\}$
  $U \leftarrow \{1\}$ (instead of 1 this can also be an arbitrary column $j \in \{1, 2, \ldots, m\}$)
  while there exists $j \in E \setminus U$ such that $(\cup_{i \in U} \mathcal{S}(I_i)) \cap \mathcal{S}(I_j) \neq \emptyset$ **do**
    $U \leftarrow U \cup \{j\}$
  end while
  $A \leftarrow$ matrix formed of all columns I_u of I for which $u \in U$
  $B \leftarrow$ matrix formed of all columns I_v of I for which $v \in E \setminus U$
  for $i \in \{1, 2, \ldots, |U|\}$ **do**
    $A^{(i)} \leftarrow$ matrix obtained by removing i-th column from A
    $C^{(i)} \leftarrow$ PTREEEXACTFNONLY($A^{(i)}$)
    $C^{(i)} \leftarrow$ extend $C^{(i)}$ by adding $\mathcal{S}^{-1}(\cup_{i \in U} \mathcal{S}(I_i))$ as its last column
  end for
  $C \leftarrow \arg\min_{i \in \{1,2,\ldots,|U|\}} \mathcal{F}\left(A, C^{(i)}\right)$
  $D =$ PTREEEXACTFNONLY(B)
  $Y \leftarrow$ concatenate matrices C and D
  $Y \leftarrow$ using column labels reorder columns in Y so that their ordering matches the ordering of columns in I
  return Y
end function

5.2 Pseudocode of the fast computational method for many cells in the absence of false positives

Supplementary Algorithm 2 *Input:* a boolean genotype matrix $I \in \{0,1\}^{n \times m}$, which may have false negatives. *Output:* a "corrected" conflict-free matrix Y, which, if Assumption 1 is satisfied, implies the ground truth tree.

function ApproximatePtreeReconstruct(I)
  $Y \leftarrow I$
  if number of columns in I is zero or one **then**
    return Y
  end if
  $E \leftarrow \{1, 2, \ldots, m\}$
  $U \leftarrow \{1\}$ (instead of 1 this can also be an arbitrary column $j \in \{1, 2, \ldots, m\}$)
  while there exists $j \in E \setminus U$ such that $(\cup_{i \in U} \mathcal{S}(I_i)) \cap \mathcal{S}(I_j) \neq \emptyset$ **do**
    $U \leftarrow U \cup \{j\}$
  end while
  $A \leftarrow$ matrix formed of all columns I_u of I for which $u \in U$
  $B \leftarrow$ matrix formed of all columns I_v of I for which $v \in E \setminus U$
  $i \leftarrow \arg\min_{j \in \{1,2,\ldots,|U|\}} \mathcal{F}\left(A_j, \mathcal{S}^{-1}(\cup_{u \in U} \mathcal{S}(I_u))\right)$
  $A^{(i)} \leftarrow$ matrix obtained by removing i-th column from A
  $Y \leftarrow$ a single column matrix with the column equal to $\mathcal{S}^{-1}(\cup_{u \in U} \mathcal{S}(I_u))$
  $Y \leftarrow$ concatenate Y and ApproximatePtreeReconstruct($A^{(i)}$)
  $Y \leftarrow$ concatenate Y and ApproximatePtreeReconstruct(B)
  $Y \leftarrow$ using column labels reorder columns in Y so that their ordering matches the ordering of columns in I
  return Y
end function

5.3 Pseudocode of the most general method for genotype matrix reconstruction handling false positives, false negatives, and missing entries

Supplementary Algorithm 3, as described below, is the heart of the extended version of HUNTRESS which can handle not only false negatives but also false positives and missing entries. Primarily, Supplementary Algorithm 3 sweeps through the parameters λ and μ. For each parameter setting, it calls Supplementary Algorithm 4 to compute the output matrix R for which it computes the likelihood score defined as the sum of (i) the number of 1 to 0 flips between I and R, weighted by γ, and (ii) 0 to 1 flips between I and R, weighted by 1 where γ is the estimated ratio of number of false negatives to number of false positives. The specific output matrix R_{opt} minimizing the likelihood score is then forwarded to the post-processing step for the final refinement and as specified previously α, β correspond to estimated false positive and negative probabilities respectively.

Note that all experiments in this paper were conducted by using the default values. Unless the ratio of the number of false negatives to the number of false positives is too small (close to 1 or less), the output is not very sensitive to these parameters and default values yield good results. If the user has knowledge on the false positive and/or false negative rates in the data, they can be input for further refinement of the results. For the case where there are no false positives (i.e. $\alpha = 0$), HUNTRESS calls Supplementary Algorithm 2 that we have presented earlier, specifically for this case.,

Supplementary Algorithm 3 HUNTRESS: Histogrammed UNion Tree REconStruction Scheme, an autotuned progression history reconstruction method with post-processing refinement. *Input:* Observed genotype matrix I and estimated false positive and false negative rates of single-cell data. *Output:* Conflict-free matrix Y representing noise corrected version of the input matrix I (ideally, Y contains *true* genotypes of the sequenced cells).

1: **function** HUNTRESS(I, α, β, γ)
2: $f_{opt} \leftarrow +\infty$ (this variable stores the smallest weighted number of flips between I and R's as given below)
3: $R_{opt} \leftarrow$ any matrix of dimension $n \times m$ (value will be set below)
4: **if** $\alpha > 0$ **then**
5: **for** λ in $\{0.2, 0.3, 0.4\}$ **do**
6: **for** μ in $\{1, 3, \ldots, 97, 99\}$ **do**
7: $R \leftarrow$ PtreeApproxGeneral(I, λ, μ)
8: $f \leftarrow \gamma|\{(i,j)|I[i,j] = 1 \ \& \ R[i,j] = 0\}| + |\{(i,j)|I[i,j] = 0 \ \& \ R[i,j] = 1\}|$
9: **if** $f < f_{opt}$ **then**
10: $f_{opt} \leftarrow f$
11: $R_{opt} \leftarrow R$
12: **end if**
13: **end for**
14: **end for**
15: $Y \leftarrow$ Post-Proc($I, R_{opt}, \alpha, \beta$)
16: **else**
17: $Y \leftarrow$ ApproximatePtreeReconstruct(I)
18: **end if**
19: Return Y
20: **end function**

Supplementary Algorithm 4 generates the outline of the topology for the tumor progression tree. For ordering the columns of the input matrix I in the presence of missing entries, for each column we define its *weight* as the ratio of the number of 1s to the total number of 0s and 1s present in the column. Because of potential false positives, columns that are disjoint in the ground truth genotype matrix G might have a non-empty intersection in I. Parameter λ helps eliminate the false positive columns of a connected component. Parameter μ, on the other hand, helps eliminate the false positive entries of a true positive column. After the elimination of the false positive columns and rows of a connected component (to form a subtree of the tumor progression tree), we compute the union of the remaining columns as the pivot column and perform a final sweep across the columns that are not a part of the connected component. Those that have a substantial overlap with the pivot column are added to the connected component (see lines 21 to 25). For all other columns, we set their rows that intersect with the pivot column to 0. This approach ensures that the three-gametes rule is satisfied for the output matrix R.

Supplementary Algorithm 5 postprocesses the matrix R returned by Supplementary Algorithm 4 by comparing it to the input matrix I. For that, it first scans through columns of I and replaces each column with the column of R with the highest conditional probability with respect to the entries that are not missing in the input data. It then takes the resulting matrix Y and repeats the same procedure for its rows, again using those of the input

Supplementary Algorithm 4 Generalized Approximate P-Tree Reconstruction with Parameterization. *Input:* Genotype matrix I with false positives, false negatives, and potentially some missing entries (denoted as "?"); user defined parameters λ and μ. *Output:* Conflict-free matrix R.

function PTREEAPPROXGENERAL(I, λ, μ)
  $I_{temp} \leftarrow I$ (some local changes will be made to I below so we store its original value here)
  $w \leftarrow$ a vector of size m such that $w[i]$ holds the "weight" of i-th column of I (i.e., i-th mutation)
  for $i \in \{1, 2, \ldots, m\}$ **do**
    $ones \leftarrow$ the number of entries in the column I_i that are equal to 1
    $known \leftarrow$ the number of non-missing entries in the column I_i
    $w[i] \leftarrow ones/known$
  end for
  $C \leftarrow \{1, 2, \ldots, m\}$
  while $C \neq \emptyset$ **do**
    choose $i \in C$ such that $w[i] \geq w[j]$ for all $j \in C$
    $h \leftarrow$ a vector of dimension n (a histogram)
    for $j \in \{1, 2, \ldots, n\}$ **do**
      $h[j] \leftarrow |\{k | k \in C, |\mathcal{S}(I_k) \cap \mathcal{S}(I_i)| \geq \lambda \cdot \min\{|\mathcal{S}(I_k)|, |\mathcal{S}(I_i)|\}, j \in S(I_k)\}|$
    end for
    $s \leftarrow \emptyset$
    for $j \in \{1, 2, \ldots, n\}$ **do**
      if $h[j] \geq \max\{h[1], h[2], \ldots, h[n]\}/\mu$ **then** $s \leftarrow s \cup \{j\}$
      end if
    end for
    for $\ell \in C$ **do**
      if $|\mathcal{S}(I_\ell) \cap s| > |\mathcal{S}(I_\ell)|/2$ **then** $I_\ell \leftarrow \mathcal{S}^{-1}(\mathcal{S}(I_\ell) \cap s)$
      else $I_\ell = \mathcal{S}^{-1}(\mathcal{S}(I_\ell) \setminus s)$
      end if
    end for
    $C \leftarrow C \setminus \{i\}$
    $R_i \leftarrow \mathcal{S}^{-1}(s)$
  end while
  $I \leftarrow I_{temp}$ (this ensures that the value of I is restored to its original value passed through the function argument)
  Return R
end function

matrix I. The postprocessing step can stop at this point or could be repeated until the output matrix reaches stability; this takes no more than two iterations in simulated problem instances.

In Supplementary Algorithm 5, $P(I_i \mid R_j, \alpha, \beta)$ is taken as $\alpha^{N_{10}^{ij}} \beta^{N_{01}^{ij}} (1-\alpha)^{N_{00}^{ij}} (1-\beta)^{N_{11}^{ij}}$. Here N_{kl}^{ij} denotes the number of entries that are equal to k in I_i and equal to l in R_j where $k, l \in \{0, 1\}$. $P(I_i \mid R_j, \alpha, \beta)$ is thus the conditional probability over the non-missing entries, given that false positives and false negatives are i.i.d.; here α and β represent the probability of false positives and false negatives, respectively.

Supplementary Algorithm 5 Manipulates rows and columns of a given conflict-free matrix R in order to create a conflict-free matrix Y which is closer in terms of the conditional probability to the noisy input matrix I. Here, we assume that probabilities of false positive and false negative mutation calls in I, denoted respectively as α and β, are given as arguments (in addition to the noisy input genotype matrix I and matrix R obtained by (multiple) calls of the function defined in Supplementary Algorithm 4).

1: **function** POST-PROC(I, R, α, β)
2: $\hat{N}_{01} \leftarrow$ number of entries that are 0 in I but 1 in Y , $N_{01} \leftarrow \hat{N}_{01} + 1$
3: **while** $\hat{N}_{01} < N_{01}$ (based on our experimental results this loop typically repeats only a few times) **do**
4: $N_{01} \leftarrow \hat{N}_{01}$
5: $Y \leftarrow R$
6: **for** $i \in \{1, 2, \ldots, m\}$ **do**
7: $t \leftarrow \arg\max_{j \in \{1,2,\ldots,m\}} P(I_i \mid R_j, \alpha, \beta)$ (see the main text for details of computing $P(I_i \mid R_j, \alpha, \beta)$)
8: $Y_i \leftarrow R_t$
9: **end for**
10: $R \leftarrow Y$
11: Repeat lines 4-9 this time comparing rows of I and R instead of comparing their columns
12: $\hat{N}_{01} \leftarrow$ number of entries that are 0 in I but 1 in Y
13: **end while**
14: return Y
15: **end function**

5.4 Supplementary Proofs

In this section we provide proofs of the two lemmas and the theorem stated in the Methods section of the main manuscript.

Assumption 1 (restated) Matrix I does not contain false positive mutation calls. Furthermore, I satisfies the following for each pair of mutations M_i and M_j that are in a parent-child dependency in the ground truth tree (i.e., the node labeled by M_i is parent of the node labeled by M_j): $|\mathcal{S}(I_i)| > |\mathcal{S}(I_j)|$ and $\mathcal{S}(I_i) \cap \mathcal{S}(I_j) \neq \emptyset$.

Lemma 2 (restated) Assume that we are given a tumor having the progression history that can be represented by the use of a mutation tree in which infinite-sites assumption is satisfied. Let φ denote the fraction of the smallest population of cells in the entire tumor sample. Given that the false negative mutation detection rate is $\beta < \varphi/2e$, where e is the base of the natural logarithm, if $n \gg m$ cells are i.i.d. sampled from the tumor population, then Assumption 1 is satisfied with "high" probability.

Proof. We first show that this lemma is valid for the case where the (ground truth) tree implied by G has a "linear" topology and each two populations with a parent-child relationship differ by one mutation. Without loss of generality, assume that the columns of the input genotype matrix I are sorted in increasing order with respect to the number of 1s they have in the matrix G. Then $G[k, i] \leq G[k, i + 1]$ for all i, k (see [9] for a more detailed discussion of correlations between conflict-free matrices and linear trees). As there are m cancerous populations (due to the infinite-sites assumption and the assumption that parent-child populations differ by one mutation), we have $m \cdot \varphi \leq 1$, implying that $m \leq 1/\varphi$.

In the case of tree with linear topology, Assumption 1 is satisfied if and only if I satisfies the two conditions below for all $1 \leq i < m$:

(a) $\sum_{k=1}^{n} I[k, i] < \sum_{k=1}^{n} I[k, i + 1]$; i.e., the number of 1s in columns increases from left to right, and

(b) $\mathcal{S}(I_i) \cap \mathcal{S}(I_{i+1}) \neq \emptyset$ for all columns I_i and I_{i+1} of the matrix I; i.e., any two adjacent columns have a nonempty intersection (of the locations of their 1s).

We first consider (a). For (a) to be violated by columns I_i and I_{i+1}, enough entries of the column I_{i+1} must be false negatives so that the number of 1s in column I_{i+1} is not greater than the number of 1s in column I_i. More precisely, the inequality $\sum_{k=1}^{n} I[k, i] \geq \sum_{k=1}^{n} I[k, i + 1]$ must hold. Define $l_i = \sum_{k=1}^{n} G[k, i]$ (i.e., l_i is the number of 1s in the column G_i) and $n_{i+1} = l_{i+1} - l_i$. Then, in order for part (a) of the assumption to be violated, we must have at least n_{i+1} false negative mutation calls in I_{i+1}. Given n_{i+1} locations and the fact that each can be a false negative independently with probability β, the probability that all n_{i+1} will be false negatives is $\beta^{n_{i+1}}$. These locations can be chosen in $\binom{l_{i+1}}{n_{i+1}}$ ways in column $i+1$. By applying the union bound, we can bound the probability

ρ_i that condition (a) will be violated by the pair of columns I_i and I_{i+1}, as follows (for more detailed proof of this bound see the subsection after the end of this proof):

$$\rho_i \leq \beta^{n_{i+1}} \binom{l_{i+1}}{n_{i+1}}.$$

In order for I to satisfy (a), (a) must be satisfied by all adjacent column pairs of I. The probability of this happening is at least

$$\prod_{i=1}^{m-1}(1-\rho_i) \geq \prod_{i=1}^{m-1}\left[1-\beta^{n_{i+1}}\binom{l_{i+1}}{n_{i+1}}\right] =$$
$$\prod_{i=2}^{m}\left[1-\beta^{n_i}\binom{l_i}{n_i}\right] \geq \prod_{i=2}^{m}\left[1-\beta^{\varphi_i n}\binom{n}{\varphi_i n}\right]$$

where $\varphi_i = n_i/n$ and we used the fact that each term in the above products is positive real number (proof provided below).

Using the well known inequality

$$\binom{n}{k} \leq \left(\frac{en}{k}\right)^k.$$

we have

$$\prod_{i=2}^{m}\left[1-\beta^{\varphi_i n}\binom{n}{\varphi_i n}\right] \geq \prod_{i=2}^{m}\left[1-\left(\frac{\beta e}{\varphi_i}\right)^{\varphi_i n}\right]$$

Now, observe that $\varphi \leq \varphi_i$, combined with the assumption $\frac{\beta}{\varphi} < \frac{1}{2e}$, implies that $\frac{\beta e}{\varphi_i} \leq \frac{\beta e}{\varphi} < \frac{1}{2}$ for each i. We know from (a version of) Bernoulli's inequality that, for real numbers $t \geq 1$ and $0 \leq x \leq 1$, $(1-x)^t \geq 1-tx$. Combining this with the above observation, we can bound the probability of (a) being satisfied by all adjacent column pairs as

$$\prod_{i=2}^{m}\left[1-\left(\frac{\beta e}{\varphi_i}\right)^{\varphi_i n}\right] \geq \left[1-\left(\frac{1}{2}\right)^{\varphi n}\right]^{m-1} \geq 1-(m-1)\cdot\left(\frac{1}{2}\right)^{\varphi n} > 1-\frac{1}{\varphi}\left(\frac{1}{2}\right)^{\varphi n}$$

where the last inequality follows from $(m+1)\cdot\varphi \leq 1$ (this holds true because, including the healthy population, there are in total $m+1$ genetically distinct populations of cells, their frequencies add up to 1 and each frequency is lower bounded by φ), which implies $m-1 = (m+1)-2 \leq \frac{1}{\varphi}-2 < \frac{1}{\varphi}$.

The probability of satisfying (b) can also be simply bounded. As at least φn cells are sampled from each population, we have that in G the intersection of two columns G_i and G_{i+1} is of size at least φn (i.e., $|\mathcal{S}(G_i) \cap \mathcal{S}(G_{i+1})| \geq \varphi n$). For this intersection to become empty, for each row j where $G[j,i]=1$ and $G[j,i+1]=1$, at least one of $I[j,i]$ or $I[j,i+1]$ must be a false negative. Therefore, the probability that $|\mathcal{S}(I_i) \cap \mathcal{S}(I_{i+1})| = 0$, denoted as ψ_i, can be bounded as follows

$$\psi_i \leq (2\beta)^{\varphi n} = 2^{\varphi n}\beta^{\varphi n}.$$

Note that $\varphi \leq \frac{1}{m+1} \leq \frac{1}{2}$ and using the well known inequality $\left(\frac{n}{k}\right)^k \leq \binom{n}{k}$ we have $2^{\varphi n} \leq \binom{n}{\varphi n}$. Now we have that the probability that all adjacent pairs of columns of I have nonempty intersection is at least

$$\prod_{i=1}^{m-1}(1-\psi_i) \geq \prod_{i=1}^{m}(1-2^{\varphi n}\beta^{\varphi n}) \geq \prod_{i=2}^{m}\left[1-\beta^{\varphi n}\binom{n}{\varphi n}\right].$$

Using the same arguments as part (a) we can prove that the last product is also lower bounded by $1-\frac{1}{\varphi}\left(\frac{1}{2}\right)^{\varphi n}$. Thus, the probability that both (a) and (b) will be satisfied by the entirety of I; i.e., Assumption 1 will be satisfied, is at least

$$1-\frac{2}{\varphi}\left(\frac{1}{2}\right)^{\varphi n}.$$

By establishing the above inequality we have completed the proof for the case where the ground truth tree implied by G has a linear topology and all parent-child cellular populations (subclones) differ by exactly one

mutation. This result can be extended to cases of linear trees where possibly more than one mutations distinguish parent-child populations. Namely, in such case, all of the above inequalities would still hold, except that in the last ones the coefficient $m-1$ in front of $\left(\frac{1}{2}\right)^{\varphi n}$ needs to be replaced by $\frac{m^2}{2} \leq \frac{1}{2\varphi^2}$ (because in this case the number of terms in the products is upper bounded by the total number of mutational pairs that are in parent-child relations and this number is certainly less than $\frac{m^2}{2}$). In summary, in this more general case of linear trees, the coefficient $\frac{2}{\varphi}$ present in the lower bound for the above special case, now needs to be replaced by $\frac{1}{\varphi^2}$.

Finally, this proof immediately generalizes to the most general case where the ground truth tree implied by matrix G might contain some branching events and multiple mutations per node. Namely, in such cases the proof essentially remains the same as for the case of linear trees that have one or multiple mutations per node. □

More detailed proof of the bound on the value of ρ_i used in the proof of Lemma 2

Observe that, in order for the inequality $\sum_{k=1}^{n} I[k,i] \geq \sum_{k=1}^{n} I[k,i+1]$ to hold, the total number of false negatives in I_{i+1} needs to be higher for at least n_{i+1} compared to the total number of false negatives in I_i. Let $\mathcal{P}$ be a function defined on the pairs (Q,k), where Q is an arbitrary set and k is a non-negative integer, such that $\mathcal{P}(Q,k)$ equals the set of subsets of Q of size k. For $q \in \mathcal{P}(\mathcal{S}(G_{i+1}), n_{i+1})$ let E_q be an event that I_{i+1} contains false negatives at all positions (i.e., rows/cells) from the set q (note that we allow the presence of other false negatives in I_{i+1}). In other words, E_q is an event that at a given set of n_{i+1} positions from $\mathcal{S}(G_{i+1})$ we have false negative mutation calls in I_{i+1}. Assuming that false negatives occur independently of each other, the probability of each E_q is $\beta^{n_{i+1}}$. Due to the observation stated above, ρ_i is clearly upper bounded by (below, P denotes the probability function)

$$P\left(\bigcup_{q \in \mathcal{P}(\mathcal{S}(G_{i+1}), n_{i+1})} E_q\right) \leq \sum_{q \in \mathcal{P}(\mathcal{S}(G_{i+1}), n_{i+1})} P(E_q)$$
$$= \beta^{n_{i+1}} \binom{l_{i+1}}{n_{i+1}}$$

where in the last step we used the fact that the size of the set $\mathcal{P}(\mathcal{S}(G_{i+1}), n_{i+1})$ equals $\binom{l_{i+1}}{n_{i+1}}$.

Theorem 1 (restated) Given an input I that satisfies Assumption 1, Supplementary Algorithm 2 computes an output matrix Y that is conflict-free and the tree reconstructed from this matrix matches the ground truth tree in preserving all different-lineages dependencies between mutations, as well as either preserving ancestor-descendant dependencies or, sometimes, possibly merging together mutations that are in ancestor-descendant order (but never swapping their relative order).

Proof. We first prove that for two mutations M_i and M_j such that M_i is parent of M_j in tree T_G implied by G, in the output matrix Y we have $\mathcal{S}(Y_i) \supseteq \mathcal{S}(Y_j)$. Based on the Assumption 1, it follows that $\mathcal{S}(I_i) \cap \mathcal{S}(I_j) \neq \emptyset$, which guarantees that columns corresponding to M_i and M_j are kept together in matrices passed as arguments in recursive calls of function implemented in Supplementary Algorithm 2, until one of the two is chosen as column that best matches $\mathcal{S}^{-1}(\cup_{u \in U} \mathcal{S}(I_u))$. As $|\mathcal{S}(I_i)| > |\mathcal{S}(I_j)|$, due to the greedy approach used in our algorithm, the column that is selected earlier will be the one that corresponds to M_i. Then, before making the next recursive call of the function, mutation M_i gets removed from the input provided to the recursive call. In other words, M_i and is not a part of the resulting matrix, which is denoted as $A^{(i)}$ in the pseudocode and is provided as an argument in the recursive call. From the description of the algorithm, it is obvious that for any column corresponding to the mutation M_k of the matrix $A^{(i)}$, in the final solution we have $Y_k \subseteq Y_i$. Since the column corresponding to M_j belongs to this matrix, we then have $\mathcal{S}(Y_i) \supseteq \mathcal{S}(Y_j)$. As this holds for each pair of mutations that are in parent-child dependency, it trivially extends to all pairs of mutations that are in ancestor-descendant dependency in T_G. From the observation that $\mathcal{S}(Y_i) \supseteq \mathcal{S}(Y_j)$ holds for arbitrary pair of mutations that are in ancestor-descendant dependency in T_G, it follows that no such pair of mutations will be placed on different lineages nor in descendant-ancestor dependency in the tree implied by Y.

Second, we prove that for two mutations M_i and M_j that are on different lineages in the tree T_G implied by G, we have that they are also on different lineages in the tree implied by Y. To prove this, note that it is sufficient to prove that $\mathcal{S}(Y_i) \cap \mathcal{S}(Y_j) = \emptyset$. As M_i and M_j are on different lineages in the tree T_G we have $\mathcal{S}(G_i) \cap \mathcal{S}(G_j) = \emptyset$. Consider the topology of T_G. Since $\mathcal{S}(G_i) \cap \mathcal{S}(G_j) = \emptyset$ we know that mutations M_i and M_j belong to different lineages (equiv. different tree branches), implying that neither of them is an ancestor of the other. What we have just shown above is that, for two mutations that are in ancestor-descendant order in T_G, the one which is an ancestor will be processed first. Consider the point $\mathcal{E}$ in the execution of Supplementary Algorithm 2 when the

column corresponding to the most recent common ancestor of M_i and M_j in T_G, denote it as M_k, is processed and the column Y_k constructed. Prior to the point $\mathcal{E}$, M_k and all of its descendants (including M_i and M_j) will be together in the same matrix passed as an argument to the function implemented in Supplementary Algorithm 2 (this follows from the Assumption 1 as it guarantees that, for each pair of parent-child mutations M_a and M_b on the path from M_k to M_i or M_j in T_G, the intersection of $\mathcal{S}(M_a)$ and $\mathcal{S}(M_b)$ is non-empty so columns corresponding to all of these mutations will be added to the union set U). However, once column corresponding to M_k is processed, in the subsequent recursive calls, the two mutations M_i and M_j will eventually be split into two different matrices A and B. This is true due to the fact that M_i and M_j belong to different lineages and for any pair of mutations from different lineages there does not exist a cell in I harboring both of them (here we exploit the assumption that there are no false positives, which implies $\mathcal{S}(I_i) \cap \mathcal{S}(I_j) = \emptyset$). As soon as mutations M_i and M_j are separated into two different matrices (i.e., A and B) during the execution of our algorithm, we obviously must have $\mathcal{S}(Y_i) \cap \mathcal{S}(Y_j) = \emptyset$ in the reported solution Y (see discussion related to the reordered matrix I that is provided in the Section 4.3).

Note that above we used the assumption that each pair of mutations M_i and M_j has the most recent common ancestor M_k, which is not necessarily always true (e.g., in case of some multicentric tumors). However, it can be easily verified that even if M_k does not exist, then we still have $\mathcal{S}(Y_i) \cap \mathcal{S}(Y_j) = \emptyset$. One way of showing it is by introducing an artificial *null* mutation M_0 present in all cells in I and applying the above argument (in this case each pair of mutations obviously has a common ancestor, namely M_0). As M_0 is present in all cells, its corresponding column Y_0 will be generated first and the algorithm will then be recursively called on the original matrix I hence removing column Y_0 from Y would give us the same output as we would originally obtain when providing the observed matrix I (with m columns) as the input matrix. □

Lemma 3 (restated): Supplementary Algorithm 2 can be implemented in $O(nm^2)$ time using $O(nm)$ space.

Proof. Supplementary Algorithm 2 can be implemented in the two steps described below. In the first step, it simply identifies "connected components" of the graph $\mathcal{G}_I$ in which each vertex corresponds to a specific column of the input genotype matrix I and there is an edge between two vertices if and only if their corresponding columns have a row at which both of them are equal to 1 in I. The graph $\mathcal{G}_I$ can be constructed in $O(nm^2)$ time, whereas finding its connected components will only take $O(m^2)$ time. The second key step involves computing the "union" of all columns in a connected component. This will take $O(nm)$ time and will need to be repeated at most m_c times for a connected component with m_c vertices. As all values of m_c add up to m, the overall running time required for completing this step is $O(nm^2)$. Combining the above we conclude that Supplementary Algorithm 2 can be implemented so that its total running time does not exceed $O(nm^2)$.

The memory requirements of Supplementary Algorithm 2 are no more than a constant number of matrices of size $O(nm)$ together with some pointers that store the information about which columns (vertices of $\mathcal{G}_I$) are included in the recursive calls. As these pointers obviously do not require more than $O(nm)$ space, the total memory required by the algorithm is $O(nm)$. □

5.5 Details of simulated data generation

Simulated data used in this work were generated by adopting the source code previously used in [8] and [23]. These simulations are to some extent based on the commonly used clonal theory of cancer growth, according to which tumors are composed of a number of genetically highly similar populations of cells (subclones). However, in order to avoid to extensively rely on this theory, in most simulations where tree inference accuracy was assessed, we set the number of subclones to a very high value of 100. More discussion of this parameter choice can also be found at the end of this section. The process of generating simulated data can be divided into three main steps:

1. **Simulating tree of tumor progression**

   We first simulate a random rooted clonal tree of tumor progression of a given size k. Each of the tree nodes represents a genetically distinct population of cells. This tree is generated iteratively, starting from the root, which is mutation-free and represents the population of healthy cells, and adding each new node as a child of one of the randomly selected existing nodes. We then randomly assign a set of $m \geq k-1$ mutations, denote them as $M_1, M_2, \ldots, M_m$, to the nodes of the tree. While making this assignment, we assume that commonly used infinite sites assumption holds, that is: mutation M_i is assigned to exactly one node and is present at that node as well as in all of its descendants (we will see below how violations of infinite sites assumption are simulated). In order to ensure that nodes of the tree correspond to genetically distinct populations of cells, we require that each node is assigned at least one mutation. The node to which M_i is assigned represents the node of the first occurrence of M_i. Each node of the tree is then assigned a *frequency*, which represents

the cellular prevalence of the related cellular population (subclone) in the simulated tumor. This frequency is a real number greater than some small pre-defined constant (in all simulations this value was set to only 0.005). Frequencies of all nodes are required to add up to 1.

2. **Single-cell data simulation**

   Once the tree of tumor progression is generated, we draw a given number n of single cells, denote them as $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n$, from the tree nodes. The probability of drawing a cell from a given node equals the frequency that was assigned to the node during the tree simulation step. Each of the sampled cells has a unique *true genotype*, which is a binary vector having i-th coordinate set to 1 if and only if mutation M_i is present at the node of origin of the cell or some of its ancestors. After single-cell sampling is completed, we obtain a *true genotype matrix* G, which is a binary matrix with n rows (representing cells) and m columns (representing mutations). We set $G_{ij} = 1$ if and only if mutation M_j is present in cell $\mathcal{C}_i$. In order to simulate noise introduced (in practical applications) during single-cell sequencing and mutation calling, we first add false positive noise to G by considering its entries that are equal to 1 and flipping them to 0 independently with flip probability equal to a given *false positive rate* constant α. Similarly, we add false negatives at a given *false negative rate* constant β. Lastly, we independently change entries G to ? (*missing entry*) with a probability equal to a given *missing entries rate* constant γ. We denote by I the *noisy single-cell matrix* obtained from G after the above noise-adding steps are completed.

3. **Simulating copy number aberrations that cause violations of infinite-sites assumption**

   All methods used in this work rely on the infinite sites assumption. As mentioned above, it states that each somatic single nucleotide variant occurs exactly once during the course of tumor progression and once it is acquired in some cell it is not lost in any of its descendants. However, the presence of copy number aberrations is the major cause of the violations of this assumption. Deletions and loss of heterozygosity events in which a variant allele is lost cause losses of previously gained mutations. On the other hand, copy number gains do not have a major effect on the status (presence vs. absence) of a single nucleotide variant in a given cell. While we require that mutations from regions affected by deletions and loss of heterozygosity events are filtered prior to running HUNTRESS, due to imperfect calling of copy number variants, some of such mutations might remain in the input.

   In order to assess the robustness of the methods to the presence of this type of noise, we also generated simulated data where 10% of the total number of mutations (i.e., $\frac{m}{10}$ mutations) are affected by losses. Mutations on which losses would have a major effect in the context of tree reconstruction are these occurring at non-leaf (i.e., internal) nodes of the simulated tree. Therefore we first randomly choose $\frac{m}{10}$ among mutations occurring for the first time at the internal nodes and then for each mutation M from this set we do the following: assume that M occurs for the first time at node v. We consider the set of all descendants of v in the simulated tree, denote it as $D(v)$, and then randomly choose one node, denote it as w, from the set $D(v)$. We assume that M is lost at w. In other words, M is present at the node v and all of its descendants, except at the node w and all of its descendants.

   After we have selected mutations affected by losses and nodes where these losses occur, we proceed with generating true and noisy single-cell genotype matrices, denote them as G^{del} and I^{del}, respectively. We start by setting $G^{del} = G$ and $I^{del} = I$, where G and I were obtained in the Step 2 described above. In order to get values of G^{del} and I^{del} we first observe that only columns of these matrices corresponding to mutations affected by deletions need to be updated. More precisely, for a given mutation M that is lost at node w, the only entries of G^{del} that need to be updated are these at the intersection of (i) column corresponding to M and (ii) rows corresponding to cells sampled from node w or its descendants. Namely, all these entries are set to 1 in G and they need to be flipped from 1 to 0 in G^{del} as they correspond to cells where M was lost. Once this correction in the column related to mutation M is made in G^{del}, we then consider matrix I^{del}. For all entries G^{del}_{ij} that were updated from 1 to 0 and such that $I^{del}_{ij} \neq ?$, we first update I^{del}_{ij} to $G^{del}_{ij} = 0$ (i.e., to the true status). We then flip I^{del}_{ij} to 1 with probability φ, which is done in order to simulate false positive noise. The above process ensures the minimal difference between matching noisy matrices I and I^{del}. It preserves nodes of origin of single cells for both matrices, all missing entries, as well as the values of all non-missing entries in I and I^{del} for which ground truth matrices G and G^{del} do not differ in the corresponding entries. We use this approach of obtaining I^{del} in order to minimize the effect of factors other than these introduced by mutational losses when assessing the robustness of tools to the presence of losses.

After noisy single-cell matrices are generated, we filter from the input all columns that have no entry equal to 1. These columns represent non-observed mutations for which no information useful for tree reconstruction is

available and in real settings such mutations would not be considered as a part of the input. This filtering step, on average, removes only between 1% and 2% of the total number of simulated mutations.

Note that, as mentioned above, in all of our experiments we set tree size parameter k to 100. It is important to observe that simulations with a smaller value of k (e.g., $k = 15$) are in general less challenging and imply higher reconstruction accuracy (experiments not included in this work). Namely, for smaller values of k we typically have a higher number of cells sampled from each node, as well as a higher number of mutations separating cellular populations, which in turn gives more informative signals for inference of relative placement (ancestor-descendant vs. different-lineages) of mutations.

5.5.1 Details of generating doublets

In order to simulate doublets, for each simulated instance of size $n = 1000$, $m = 300$ from Supplementary Table 3 (which does not contain any doublets), we first selected 3% of cells uniformly at random. For each of the selected cells, we replaced its genotype in the true (noise-free) matrix with the union of its true genotype and the true genotype of one randomly selected cell from the set of the remaining cells. This procedure yields an input data with 0.03 doublet rate. In order to generate the noisy version of the true matrix, the false positive, false negative and missing entries were then added to the simulated doublets at the specified rates, whereas the remaining rows were copied from the matching noisy matrix used in Supplementary Table 3.

5.6 Measuring tree inference accuracy on simulated and real data

In order to assess the performance of methods on simulated data where ground truth is available, as well as to compare results obtained by HUNTRESS to the previously published results on real datasets, we use two commonly used measures for tree inference accuracy: ancestor-descendant (AD) accuracy measure and different-lineages (DL) accuracy. Below we provide a brief description of how each of the measures was computed. For the case of simulated data, we assume that G is the true simulated genotype matrix and for the case of real data we assume that this is a noise-corrected genotype matrix reported by previously published studies. We assume that Y denotes a conflict-free matrix obtained by the use of some (specified) tool that we run in this study using the matching noisy version of G as input. We also assume that mutation corresponding to column i of G (and Y) is denoted as M_i.

Two mutations M_i and M_j can be in one of the following dependencies:

- **Ancestor-descendant** if node of occurrence of M_i is ancestor of node of occurrence of M_j.
- **Descendant-ancestor** if node of occurrence of M_i is descendant of node of occurrence of M_j.
- **Same node** if M_i and M_j occur for the first time at the same node of the tree.
- **Different-lineages** if none of the above is the case.

Mutual dependencies between M_i and M_j can be directly obtained from the simulated ground truth tree. However, in the single-cell sampling step, there might be some nodes of the tree from which no cells are sampled. Consequently, the inference of some dependencies is dependent solely on the "luck", which is related to the distribution of missing entries and false positives and false negatives in I. For example, even if the ground truth matrix G is provided as input (which, under the infinite sites assumption and assuming that we penalize for flipping entries of the input matrix, would result in $Y \equiv G$), it would be impossible to infer some of the dependencies (e.g., the ancestor-descendant dependency between M_i and M_j in case where M_i and M_j respectively occur at nodes v and w, where w is a child of v and no cell was sampled from v). Therefore, in order to focus on these dependencies which can be inferred correctly in the theoretically most ideal case (i.e., noise-free input single-cell data), we compute "true" dependencies between mutations from the matrix G.

To compute dependency implied by G for a given pair of mutations M_i and M_j, we do the following: for each pair $(a, b) \in \{(0, 1), (1, 0), (1, 1)\}$ we compute the number of cells c in which $G[c, i] = a$ and $G[c, j] = b$ and denote this number as B_{ab}. We can conclude that M_i and M_j are in ancestor-descendant order if and only if $B_{11} > 0$ and $B_{10} > 1$. Namely, $B_{11} > 0$ indicates that there exists a cell harboring both M_i and M_j, hence these mutations can not be at different lineages of the tree. Furthermore B_{10} indicates that there exists a cell harboring M_i and not harboring M_j. Combining the two observations, we conclude that in this case G implies that M_i and M_j are in ancestor-descendant order. Similarly, if $B_{01} > 0$ and $B_{10} > 0$ we can conclude that M_i and M_j are in different-lineages, whereas $B_{11} > 0$ and $B_{10} = B_{01} = B_{00} = 0$ implies that they occur at the same node. Lastly, $B_{11} > 0$ and $B_{01} > 0$ implies that M_i is descendant of M_j. We refer to [9] for more detailed discussion of this. It is also trivial to show that at least one of the four dependencies must be the case for any pair of mutations as soon as the matrix from which we are computing dependencies is conflict-free and does not contain any column having all entries equal to 0.

Using the above we compute the following two accuracy measures:

1. **Ancestor-descendant (AD) accuracy measure** For each pair of mutations M_i and M_j we check whether G implies that they are in ancestor-descendant order. If it does not, we continue. Otherwise, we check whether the same holds in the reported tree or, in cases of tools that report conflict-free matrix Y, whether it is implied by Y. If it is, we declare a success for this pair, otherwise we declare a failure. The ancestor-descendant accuracy measure is then given as the total number of successes divided by the sum of the number of success and failures (i.e., the number of successes is divided by the number of pairs of mutations for which G implies that they are in ancestor-descendant order).
2. **Different-lineages (DL) accuracy measure** We do analogous as above, this time focusing on pairs of mutations M_i and M_j for which G implies that they are in different-lineages dependency. In order to save some computational time, when computing this measure it suffices to focus only on pairs for which $i < j$.

Note that for the case where losses of mutations are simulated, due to infinite sites assumption violations, the true genotype matrix G^{del} is typically not conflict-free. Consequently, it can be impossible to infer dependencies in cases where at least one of M_i and M_j is affected by a loss. However, we still do not perform filtering of mutations affected by losses when computing scores, but rather obtain their true mutual dependencies from the matching true genotype matrix before losses (denoted above as G) and check whether these dependencies are preserved in the solution reported by HUNTRESS. We emphasize that this is less favoring to HUNTRESS and other tools in comparison to the approach where all mutations affected by deletions are filtered and excluded from score computation. As a simple illustrative example, assume that mutations M_i and M_j occur close to the root of the ground truth tree and that M_i is an ancestor of M_j, but M_i gets lost even before M_j occurs. It is very likely that in this case there will be a very weak or even a *negative* (i.e., opposite to the expected) signal in I^{del} for correct placement of M_i with respect to M_j (and with respect to many other mutations as well). Consequently, the tools will end up getting penalized for incorrect ordering of such pairs. We believe that this approach, where all mutations are kept when computing accuracies, enables more realistic assessment of the loss in tree inference accuracy caused by the presence of mutations for which infinite sites assumption is violated.

5.7 Details of the LBNL NERSC compute cluster

```
Processor :                 Intel Xeon Processor E5-2698 v3
Sockets per node:           2
Physical cores per socket:  16
Physical cores per node:    32
Operating frequency:        2.3 ghz
Memory per node:            128gb
```

5.8 Parameter Setup for SiCloneFit, gpps, ScisTree and SPhyR

5.8.1 SiCloneFit

SiCloneFit was downloaded from https://bitbucket.org/hamimzafar/siclonefit with commit number 7492d9c, which is the latest version that was available on December 25, 2021 when we last accessed the repository. The full command used to run SiCloneFit is given below:

```
java -jar SiCloneFiTComplete.jar \
    -m $NUMBER_OF_CELLS \
    -n $NUMBER_OF_MUTATIONS \
    -ipMat $PATH_TO_GENOTYPE_MATRIX \
    -fp $FALSE_POSITIVE_RATE \
    -fn $FALSE_NEGATIVE_RATE \
    -df 0 \
    -missing $FRACTION_OF_MISSING_DATA \
    -iter $NUMBER_OF_ITERATIONS \
    -cellNames $PATH_TO_CELL_NAME_FILE \
    -geneNames $PATH_TO_GENE_NAME_FILE \
    -r 8 \
    -burnin $NUMBER_OF_BURNIN \
    -outDir $PATH_TO_OUTPUT_DIR > $PATH_TO_LOG_FILE
```

5.8.2 gpps

gpps was downloaded from https://github.com/AlgoLab/gpps with commit number b77a2a6, which is the latest version that was available on December 25, 2021 when we last accessed the repository. The full command used to run gpps is given below:

```
gpps \
    -f $PARH_TO_INFPUT_FILE \
    -a $FALSE_NEGATIVE_RATE \
    -b $FALSE_POSITIVE_RATE \
    -k 0 \
    -o $PATH_TO_OUTPUT_FILE \
    -c $NUMBER_OF_THREADS \
    -t $TIME_LIMIT \
    -d -1 \
    -N 30 \
    -M 100 \
    -n $PATH_FO_FILE_NAME
```

5.8.3 ScisTree

ScisTree was downloaded from https://github.com/yufengwudcs/ScisTree with commit number cca7677, which is the latest version that was available on December 25, 2021 when we last accessed the repository. The input to ScisTree is the probability of individual genotype for each cell and each mutations. To provide this input from a binary matrix using known false-positive (*alpha*) and false-negative (*beta*) error, we use the following conversion. If the genotype is 1, we convert it to *alpha* error rate, if its is 0 we convert it to $1 - beta$ and finally if it is missing entry we convert it to 0.5. The full command used to run ScisTree is given below:

```
scistree \
    -v \
    -d 0 \
    -e \
    -o $PATH_TO_OUTPUT_FILE \
    $PATH_TO_INPUT_GENOTYPE_PROBABILITIES
```

5.8.4 SPhyR

SPhyR was downloaded from https://github.com/elkebir-group/SPhyR with commit number f083ce6, which is the latest version that was available on December 25, 2021 when we last accessed the repository. The full command used to run SPhyR is given below:

```
kDPFC \
    $PATH_TO_INPUT_FILE \
    -a $FALSE_POSITIVE_RATE \
    -b $FALSE_NEGATIVE_RATE \
    -t $NUMBER_OF_THREADS \
    -T $TIME_LIMIT \
    -k 0 \
    > $PATH_TO_OUTPUT_FILE
```

For running SPhyR on simulated data that contain only false negatives, the false positive rate parameter was set to a very small number (1e-7).

Note that in all our experiments SPhyR was run with default parameters. To get more accurate predictions, SPhyR needs to be provided with the number of subclones s, and the number of mutation clusters m in the tree. Given this information (which is unrealistic, since prior to tree construction the size of the progression tree is unknown) SPhyR's output improves in accuracy. However this substantially increases its running time since SPhyR first establishes m' clusters of mutations and n' clusters of cells, and feeds them to its ILP formulation to establish the progression tree. The running time of this ILP formulation is exponential with n' and m' (and thus these parameters are respectively set to a maximum of 10 and 15 respectively in the default settings). In fact, when we set them to the ground truth values, SPhyR is 266 times slower than HUNTRESS for $n = m = 300$, the

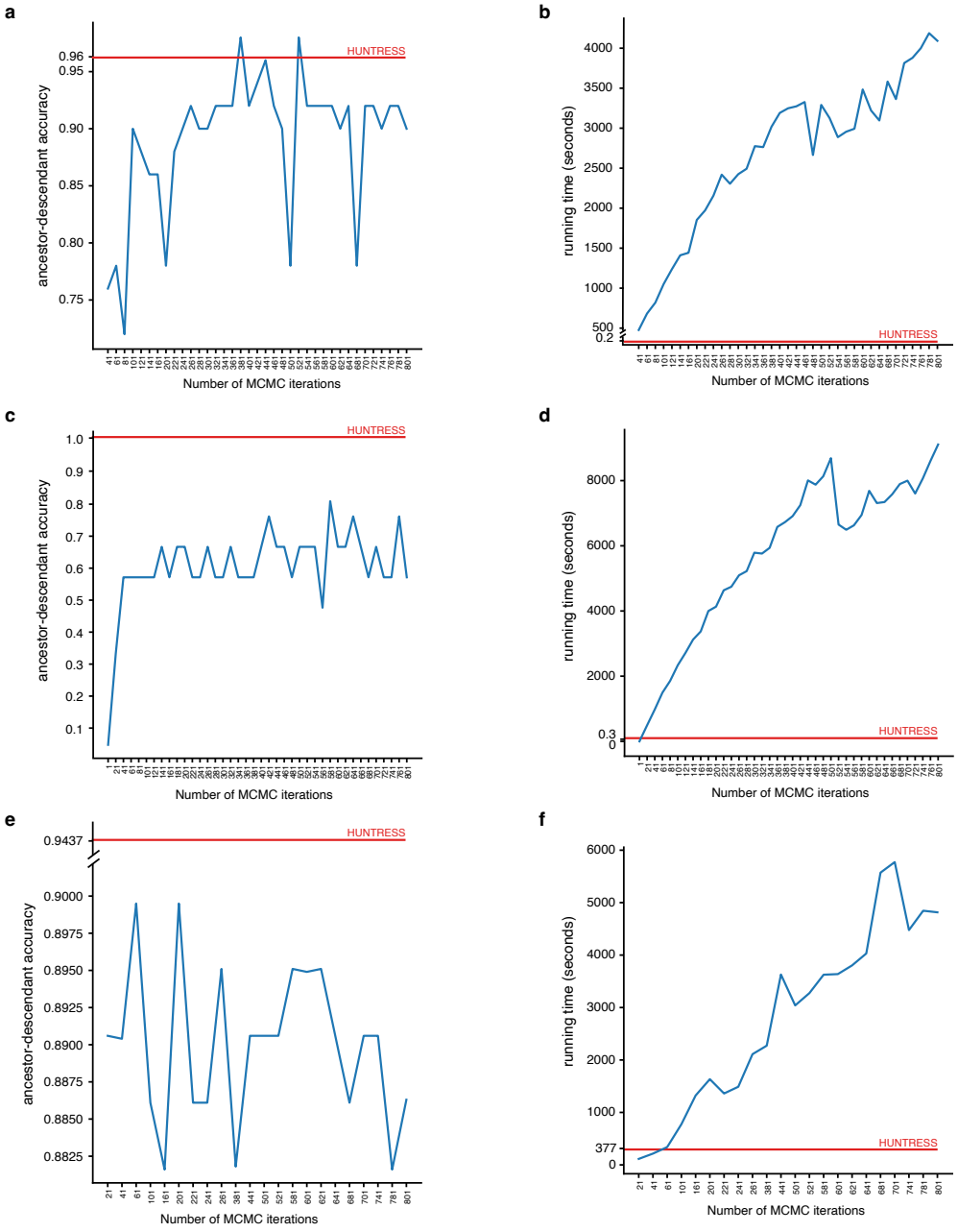
smallest input size we used in our simulations. Even for this case, it is not as accurate as HUNTRESS. Without the information about the ground truth values of n' and m', SPhyR would need to set $n' = n$ and $m' = m$, which makes it even slower.

5.9 SiCloneFit performance as a function of the number of iterations on real data

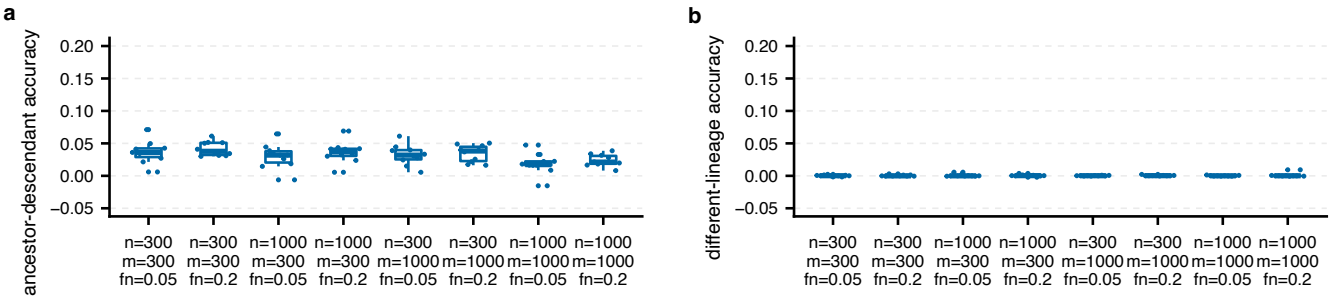
In order to further investigate the running time and accuracy performance of SiCloneFit (see the default setup results in Supplementary Table 1) we run it on the same dataset while varying the number of the MCMC iterations.

For each setting for the number of iterations, we report the running time and accuracy figures for SiCloneFit in Supplementary Figure 1. As can be seen, SiCloneFit's running time increases roughly linearly with the number of MCMC iterations. While its accuracy also improves with the number of iterations initially, it reaches a peak and then deteriorates. In comparison to HUNTRESS (shown by the red line) SiCloneFit's performance is not only poorer but is also unpredictable.

5.10 Supplementary Figures and Tables



Supplementary Figure 1: SiCloneFit's performance as a function of the number of iterations on AML and HGSOC datasets. (a,b) SiCloneFit's performance on the AML-67-001 patient dataset (c,d) SiCloneFit's performance on the AML-38-001 patient dataset (e,f) SiCloneFit's performance on the reduced size HGSOC dataset. Panels (a,c,e) demonstrate how the ancestor-descendant (AD) accuracy measure varies as a function of the number of MCMC iterations. Panels (b,d,f) represent the running time of SiCloneFit in seconds as a function of the number of MCMC interactions. In each panel the corresponding performance value of HUNTRESS is shown by the red line. For each value for the number of MCMC iterations, report the mean value for the running time and the AD measure over 16 independent runs of SiCloneFit.



Supplementary Figure 2: Assessment of HUNTRESS' robustness to the presence of deletion events (i.e., mutations for which infinite sites assumption is violated). Panel (a) shows the change in the Ancestor-Descendant (AD) accuracy measure after the introduction of deletion events, whereas panel (b) shows the change in the Different-Lineages (DL) accuracy measure. Here n, m and fn, respectively, denote the number of cells, the number of mutations and the false negative error rate. The false positive rate, fp was set to 0.01 and the missing entry rate was set to 0.05. The experiments were performed over 10 distinct trees of tumor progression. Full details on how these deletion events are simulated can be found in the Supplementary Section 5.5.

Dataset (size)	Tool	Run Time (secs)	Ancestor-Descendant (AD) Measure	Different-Lineages (DL) Measure
HGSOC (891×744) Figure 1	HUNTRESS	376.9	0.9769	0.9986
	ScisTree	∞	-	-
	gpps	∞	-	-
	SiCloneFit	∞	-	-
	SPhyR	192.4	Each mutation is deemed to be present in all cells, i.e. AD = DL = 0.0	
HGSOC (297×248) reduced size	HUNTRESS	38.3	0.9437	0.9947
	ScisTree	122.6	Each mutation is deemed to be present in all cells, i.e. AD = DL = 0.0	
	gpps	∞	-	-
	SiCloneFit	3040.3	0.8906	0.9508
	SPhyR	70.0	Each mutation is deemed to be present in all cells, i.e. AD = DL = 0.0	
AML 1 (3347×13) Figure 2a,b	HUNTRESS	0.2	0.9600	1.0000
	ScisTree	56657.5	0.9200	1.0000
	gpps	1461.5	1.0000	1.0000
	SiCloneFit	3127.1	0.9800	1.0000
	SPhyR	49.1	0.8800	1.0000
AML 2 (6783×12) Figure 2c,d	HUNTRESS	0.3	1.0000	1.0000
	ScisTree	∞	-	-
	gpps	2532.7	1.0000	0.9556
	SiCloneFit	8688.7	0.6667	0.4889
	SPhyR	27.8	0.7619	0.6444

Supplementary Table 1: A comparison of HUNTRESS' running time and accuracy with the existing tools on HGSOC and AML datasets. HUNTRESS was compared against ScisTree [7], gpps [14], SPhyR [10] and SiCloneFit [6] using ancestor-descendant (AD) and different-lineages (DL) accuracy measures (the higher the value, the better; see Supplementary Section 5.6 for definitions). Comparisons were done by comparing results of each tool to the trees published by the original studies. The time limit for each tool was set to 48 hours. In cases where a tool could not finish the task within this time limit, its running time is reported to be ∞. In each dataset, the tool with the best performance (with respect to the running time or accuracy) is highlighted in green. All multi-threaded tools were run with the same number of threads (i.e. 16) including gpps, SPhyR and HUNTRESS. For SiCloneFit we used 16 restarts in parallel modes and reported the best result in terms of the likelihood.

Dataset	Tool	Min Run time (secs)	Max Run time (secs)	Avg AD Measure	Avg DL Measure
$n = m = 300$	HUNTRESS	0.9	1.7	1	1
$fn = 0.2$	ScisTree	35.54	43.84	1	1
	SPhyR	43.8	69	0.992	0.989
	PhISCS-BnB	93.11	393.28	1	1
$n = m = 300$	HUNTRESS	1.3	2.2	1	1
$fn = 0.05$	ScisTree	32.51	45.77	1	1
	SPhyR	33.6	51	0.99	0.987
	PhISCS-BnB	7.14	56.25	1	1
$n = m = 200$	HUNTRESS	0.4	0.6	0.998	1
$fn = 0.2$	ScisTree	9.35	12.15	1	1
	SPhyR	26.1	43.3	0.969	0.996
	PhISCS-BnB	3.68	27.89	1	1
$n = m = 200$	HUNTRESS	0.4	0.7	1	1
$fn = 0.05$	ScisTree	8.96	11.77	1	1
	SPhyR	20.3	38.4	0.962	0.963
	PhISCS-BnB	1.48	9.85	1	1
$n = m = 100$	HUNTRESS	0.1	0.1	0.997	0.995
$fn = 0.2$	ScisTree	1.22	1.25	0.999	1
	SPhyR	11.5	20.1	0.979	0.988
	PhISCS-BnB	0.53	2.05	1	1
$n = m = 100$	HUNTRESS	0.1	0.1	1	0.995
$fn = 0.05$	ScisTree	1.10	1.4	1	1
	SPhyR	7.3	11.4	0.987	0.988
	PhISCS-BnB	0.347	0.827	1	1

Supplementary Table 2: Benchmarking results on simulated data with no false positives. Here n, m and fn, respectively, denote the number of cells, the number of mutations and the false negative error rate of single-cell data. For each setting of n and m, we report the maximum and minimum running times for each tool over 10 distinct trees of tumor progression, for false negative error rates of 0.05 and 0.2. Each tool was allowed to run with a time limit of 8 hours (those cases that exceed the time limit are not included here). Average ancestor-descendant (AD) and different-lineages (DL) accuracy measures for each tool is also provided for each parameter setting. All tools were run on a single thread.

Dataset	Tool	Min Run time (secs)	Max Run time (secs)	Avg AD score	AD score std	Avg DL score	DL score std
$n = 300$,$m = 300$	HUNTRESS	39.57	46.99	0.95672	0.01673	0.99779	0.00066
$fn = 0.2$	ScisTree	306.48	659.18	0.95202	0.01510	0.99669	0.00099
	SPhyR	41.00	1123.17	0.28255	0.04119	0.99078	0.00118
$n = 300$,$m = 300$	HUNTRESS	41.05	49.44	0.97193	0.011403	0.99785	0.001204
$fn = 0.05$	ScisTree	77.55	276.21	0.97183	0.014695	0.99760	0.001196
	SPhyR	34.76	288.84	0.18777	0.05392	0.99745	0.00129
$n = 300$,$m = 1000$	HUNTRESS	442.54	494.14	0.96600	0.011655	0.99798	0.000533
$fn = 0.2$	ScisTree	1079.94	2195.55	0.97085	0.00914	0.99692	0.00081
	SPhyR	38.94	48.60	0.26782	0.04683	0.99154	0.00262
$n = 300$,$m = 1000$	HUNTRESS	466.13	547.93	0.98116	0.01060	0.99846	0.00047
$fn = 0.05$	ScisTree	364.98	694.55	0.98421	0.00689	0.99768	0.00076
	SPhyR	33.87	45.55	0.20208	0.05230	0.99710	0.00087
$n = 1000$,$m = 300$	HUNTRESS	70.03	78.20	0.99458	0.00312	0.99991	0.00020
$fn = 0.2$	ScisTree	18335.27	28640.59	0.98084	0.01209	0.99943	0.00028
	SPhyR	153.48	214.05	0.14490	0.04730	0.99552	0.00257
$n = 1000$,$m = 300$	HUNTRESS	69.75	77.79	0.99892	0.00203	0.99996	0.00011
$fn = 0.05$	ScisTree	3716.14	10459.41	0.99435	0.00308	0.99984	0.00025
	SPhyR	129.36	1666.65	0.08973	0.03937	0.99958	0.00029
$n = 1000$,$m = 1000$	HUNTRESS	768.06	892.55	0.99669	0.00228	0.99994	0.00010
$fn = 0.2$	ScisTree	42441.18	90069.36	0.99651	0.00228	0.99968	0.00024
	SPhyR	149.03	208.49	0.15532	0.05289	0.99528	0.00221
$n = 1000$,$m = 1000$	HUNTRESS	762.95	866.27	0.99812	0.00388	0.99994	0.00007
$fn = 0.05$	ScisTree	13049.38	23854.09	0.99937	0.00062	0.99989	0.00009
	SPhyR	122.29	684.63	0.11915	0.03423	0.99863	0.00152

Supplementary Table 3: Benchmarking results on simulated data with false positives, false negatives and missing entries. Here n, m, fn, fp respectively, denote the number of cells, the number of mutations, the false negative and false positive error rates in single-cell sequencing data. For each setting of n and m, we report the maximum and minimum running times for each tool over 10 distinct trees of tumor progression, for false negative error rates of 0.05 and 0.2. The false positive error rate was set to 0.001 and the missing entry rate was set to 0.05. Each data set have 10 samples. Each tool was allowed to run with a time limit of 48 hours (those cases that exceed the time limit are not included here). Average ancestor-descendant (AD) and different-lineages (DL) accuracy measures for each tool are also provided for each parameter setting.

Dataset	Tool	Min Run time (secs)	Max Run time (secs)	Avg AD Score	AD score std	Avg DL Score	DL score std
$n = 5000$,$m = 500$	HUNTRESS	948.46	1412.50	0.99998	0.00003	1.0	0
$fn = 0.2$	ScisTree	Not completed in 48 hours					
	SPhyR	Failed to generate an output					
$n = 5000$,$m = 500$	HUNTRESS	966.06	1237.81	1.0	0	0.99999	0.00002
$fn = 0.05$	ScisTree	Not completed in 48 hours					
	SPhyR	Failed to generate an output					

Supplementary Table 4: Benchmarking results on larger simulated data with false positives, false negatives and missing entries. Here n, m, fn, fp respectively, denote the number of cells, the number of mutations, the false negative and false positive error rates in single-cell sequencing data. For each setting of n and m, we report the maximum and minimum running times for each tool over 10 distinct trees of tumor progression, for false negative error rates of 0.05 and 0.2. The false positive error rate was set to 0.001 and the missing entry rate was set to 0.05. Each data set have 10 samples. Each tool was allowed to run with a time limit of 48 hours, the maximum time allowed on a job at the LBNL NERSC cluster. Average ancestor-descendant (AD) and different-lineages (DL) accuracy measures for each tool are provided for each parameter setting. None of the datasets where $n = 5000$, $m = 500$ could be completed by ScisTree. While SPhyR terminated on these datasets, it did not generate an output.

Dataset	Tool	Min Run time (secs)	Max Run time (secs)	Avg AD Score	AD score std	Avg DL Score	DL score std
$n = 1000, m = 300$	HUNTRESS	68.32	78.13	0.98269	0.01060	0.99962	0.00047
$fn = 0.2$	ScisTree	19563.15	26347.99	0.97104	0.01745	0.99870	0.00104
	SPhyR	145.64	196.02	0.13439	0.02312	0.99743	0.00087
$n = 1000, m = 300$	HUNTRESS	70.37	80.60	0.99613	0.00539	0.99991	0.00021
$fn = 0.05$	ScisTree	5702.75	7555.06	0.98870	0.00918	0.99971	0.00033
	SPhyR	125.04	178.64	0.08223	0.02685	0.99941	0.00079

Supplementary Table 5: Benchmarking results on simulated data with a higher false positive rate of 0.003, in addition to false negatives and missing entries. Here n, m and fn, respectively, denote the number of cells, the number of mutations and the false negative error rate of single-cell data. For each setting of n and m, we report the maximum and minimum running times for each tool over 10 distinct trees of tumor progression, for false negative error rates of 0.05 and 0.2. Each data set have 10 samples. Each tool was allowed to run with a time limit of 48 hours Average ancestor-descendant (AD) and different-lineages (DL) accuracy measures for each tool is also provided for each parameter setting.

Dataset	Tool	Min Run time (secs)	Max Run time (secs)	Avg AD Score	AD score std	Avg DL Score	DL score std
$n = 1000, m = 300$	HUNTRESS	63.86	105.12	0.99067	0.00716	0.99983	0.00014
$fn = 0.2$	ScisTree	18274.80	25433.56	0.97262	0.01844	0.99861	0.00114
	SPhyR	134.15	180.25	0.15360	0.05055	0.99599	0.00272
$n = 1000, m = 300$	HUNTRESS	67.99	76.57	0.99733	0.00363	0.99981	0.00022
$fn = 0.05$	ScisTree	3957.38	8362.22	0.99228	0.00556	0.99984	0.00020
	SPhyR	134.86	167.45	0.09741	0.04075	0.99937	0.00043

Supplementary Table 6: Benchmarking results on simulated data with doublets in addition to false positives, false negatives and missing entries. Here n, m, fn, respectively, denote the number of cells, the number of mutations, and the false negative error rate in single-cell sequencing data. For each setting of n and m, we report the maximum and minimum running times for each tool over 10 distinct trees of tumor progression, for false negative error rates of 0.05 and 0.2. The doublet rate was set to 0.03, false positive error rate was 0.001 and the missing entry rate was 0.05. Each data set have 10 samples. Each tool was allowed to run with a time limit of 48 hours, the maximum time allowed on a job at the LBNL NERSC cluster. Average ancestor-descendant (AD) and different-lineages (DL) accuracy measures for each tool are provided for each parameter setting.

Dataset	Tool	Min Run time (secs)	Max Run time (secs)	Avg AD Score	AD score std	Avg DL Score	DL score std
$n = 5000, m = 50$	HUNTRESS	4.1	8.9	0.99054	0.0299	0.99962	0.0012
$fn = 0.05, fp = 0.003$	SPhyR	1465	1734	0.26418	0.08766	0.15306	0.04310
	ScisTree	Not completed in 48 hours					
$n = 5000, m = 50$	HUNTRESS	3.7	7.5	0.97905	0.02803	1	0
$fn = 0.05, fp = 0.01$	SPhyR	1646.5	1882.6	0.25271	0.0862	0.14349	0.0595
	ScisTree	Not completed in 48 hours					
$n = 5000, m = 50$	HUNTRESS	4.2	8.4	0.99865	0.00426	1	0
$fn = 0.2, fp = 0.003$	SPhyR	1636.9	2126	0.30541	0.07277	0.23706	0.0566
	ScisTree	Not completed in 48 hours					
$n = 5000, m = 50$	HUNTRESS	4.7	9.3	0.94324	0.0595	0.99961	0.00067
$fn = 0.2, fp = 0.01$	SPhyR	1959.3	2610.7	0.31418	0.0681	0.19569	0.029244
	ScisTree	Not completed in 48 hours					

Supplementary Table 7: Benchmarking results on simulations with parameters similar to those observed in the AML dataset generated by the Tapestri platform. Here n, m, fn, and fp respectively, denote the number of cells, the number of mutations and the false negative error rate and false positive rate observed in single-cell data. In addition the missing entries rate was set to a higher value of 10%. For each setting of fp and fn, we report the maximum and minimum running times for each tool over 10 distinct trees of tumor progression. Each tool was allowed to run with a time limit of 48 hours Average ancestor-descendant (AD) and different-lineages (DL) accuracy measures for each tool is also provided for each parameter setting.

References

[1] Kuipers, J., Jahn, K. & Beerenwinkel, N. Advances in understanding tumour evolution through single-cell sequencing. *Biochimica et Biophysica Acta (BBA)-Reviews on Cancer* **1867**, 127–138 (2017).

[2] Schwartz, R. & Schäffer, A. A. The evolution of tumour phylogenetics: principles and practice. *Nature Reviews Genetics* **18**, 213–229 (2017).

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

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

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

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

[7] Wu, Y. Accurate and efficient cell lineage tree inference from noisy single cell data: the maximum likelihood perfect phylogeny approach. *Bioinformatics* **36**, 742–750 (2020).

[8] Malikic, S., Jahn, K., Kuipers, J., Sahinalp, S. C. & Beerenwinkel, N. Integrative inference of subclonal tumour evolution from single-cell and bulk sequencing data. *Nature communications* **10**, 1–12 (2019).

[9] Malikić, S., Mehrabadi, F. R., Azer, E. S., Ebrahimabadi, M. H. & Sahinalp, S. C. Studying the history of tumor evolution from single-cell sequencing data by exploring the space of binary matrices. *Journal of Computational Biology* **28**, 857–879 (2021).

[10] El-Kebir, M. Sphyr: tumor phylogeny estimation from single-cell sequencing data under loss and error. *Bioinformatics* **34**, i671–i679 (2018).

[11] Malikic, S. *et al.* Phiscs: a combinatorial approach for subperfect tumor phylogeny reconstruction via integrative use of single-cell and bulk sequencing data. *Genome research* **29**, 1860–1877 (2019).

[12] Edrisi, M., Zafar, H. & Nakhleh, L. A Combinatorial Approach for Single-cell Variant Detection via Phylogenetic Inference. In Huber, K. T. & Gusfield, D. (eds.) *19th International Workshop on Algorithms in Bioinformatics (WABI 2019)*, vol. 143 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 22:1–22:13 (2019).

[13] Sadeqi Azer, E. *et al.* Phiscs-bnb: a fast branch and bound algorithm for the perfect tumor phylogeny reconstruction problem. *Bioinformatics* **36**, i169–i176 (2020).

[14] Ciccolella, S. *et al.* gpps: an ilp-based approach for inferring cancer progression with mutation losses from single cell data. *BMC bioinformatics* **21**, 1–16 (2020).

[15] Azer, E. S., Ebrahimabadi, M. H., Malikić, S., Khardon, R. & Sahinalp, S. C. Tumor phylogeny topology inference via deep learning. *iScience* **23**, 101655 (2020).

[16] Laks, E. *et al.* Clonal decomposition and dna replication states defined by scaled single-cell genome sequencing. *Cell* **179**, 1207–1221 (2019).

[17] Morita, K. *et al.* Clonal evolution of acute myeloid leukemia revealed by high-throughput single-cell genomics. *Nature communications* **11**, 1–17 (2020).

[18] Singer, J., Kuipers, J., Jahn, K. & Beerenwinkel, N. Single-cell mutation identification via phylogenetic inference. *Nature communications* **9**, 1–8 (2018).

[19] Gusfield, D. Efficient algorithms for inferring evolutionary trees. *Networks* **21**, 19–28 (1991).

[20] McPherson, A. W. Clonal decomposition and DNA replication states defined by scaled single cell genome sequencing (2019). URL https://doi.org/10.5281/zenodo.3445364.

[21] Malikic, S., Mehrabadi, F. R. & Kizilkale, C. Fast Intratumor Heterogeneity Inference from Single-Cell Sequencing Data (simulated data - Extended Data Figures) (2022). URL https://doi.org/10.5281/zenodo.6829082.

[22] Kizilkale, C., Buluc, A. & Rashidi, F. Passionlab/huntress: Huntress (2022). URL https://doi.org/10.5281/zenodo.6803392.

[23] Mehrabadi, F. R. *et al.* Profiles of expressed mutations in single cells reveal subclonal expansion patterns and therapeutic impact of intratumor heterogeneity. *bioRxiv* (2021).