Reconstructing complex cancer evolutionary histories from multiple bulk DNA samples using Pairtree

Jeff A. Wintersinger[1,2,3,4], Stephanie M. Dobson[5,6], Lincoln D. Stein[3,5], John E. Dick[5,6], and Quaid D. Morris[1,3,4,5,7,*]

[1] *Department of Computer Science, University of Toronto, Toronto, ON, Canada*

[2] *Donnelly Centre for Cellular and Biomolecular Research, University of Toronto, Toronto, ON, Canada*

[3] *Ontario Institute for Cancer Research, Toronto, ON, Canada*

[4] *Vector Institute for Artificial Intelligence, Toronto, ON, Canada*

[5] *Department of Molecular Genetics, University of Toronto, Toronto, ON, Canada*

[6] *Princess Margaret Cancer Centre, University Health Network, Toronto, ON, Canada*

[7] *Memorial Sloan Kettering Cancer Center, New York City, NY, USA*

[*] *Corresponding author*

1 Abstract

Cancers are composed of genetically distinct subpopulations of malignant cells. By sequencing DNA from cancer tissue samples, we can characterize the somatic mutations specific to each population and build *clone trees* describing the evolutionary ancestry of populations relative to one another. These trees reveal critical points in disease development and inform treatment.

Pairtree is a new method for constructing clone trees using DNA sequencing data from one or more bulk samples of an individual cancer. It uses Bayesian inference to compute posterior distributions over the evolutionary relationships between every pair of identified subpopulations, then uses these distributions in a Markov Chain Monte Carlo algorithm to perform efficient inference of the posterior distribution over clone trees. Unlike existing methods, Pairtree can perform clone tree reconstructions using as many as 100 samples per cancer that reveal 30 or more cell subpopulations. On simulated data, Pairtree is the only method whose performance reliably improves when provided with additional bulk samples from a cancer. This suggests a shortcoming of existing methods, as more samples provide more information, and

should always make clone tree reconstruction easier. On 14 B-progenitor acute lymphoblastic leukemias with up to 90 samples from each cancer, Pairtree was the only method that could reproduce or improve upon expert-derived clone tree reconstructions. By scaling to more challenging problems, Pairtree supports new biomedical research applications that can improve our understanding of the natural history of cancer, as well as better illustrate the interplay between cancer, host, and therapeutic interventions. The Pairtree method, along with an interactive visual interface for exploring the clone tree posterior, is available at `https://github.com/morrislab/pairtree`.

2 Introduction

Individual cancers exhibit substantial genetic heterogeneity, reflecting an ongoing evolutionary process of random somatic mutation and selection [1]. Cancers typically arise from a small number of founder mutations that confer a growth advantage [2]. Over time, additional somatic mutations accrue, and their frequency and distribution are shaped by evolutionary forces such as selection and genetic drift, resulting in the emergence of multiple genetically distinct cell subpopulations [3] (Fig. 1a). A *clone tree* is the evolutionary tree delineating the cell subpopulations in a cancer, the genetic mutations specific to each, and the proportions of cells in each sample that arose from each subpopulation (Fig. 1). Within the tree, *subclones* correspond to a cell subpopulation and all its descendants.

Clone trees built from bulk cancer samples have biologically and clinically important applications. Those built from single samples already reveal important genomic events in evolution [3, 4] and provide insights into heterogeneity [1]. But as sequencing costs continue to drop, sequencing different regions of the same tumour [5], multiple tumours of the same cancer [6], or longitudinal samples from different timepoints [7] will become more common. When bulk samples have different mixtures of subpopulations, each sample can provide unique information about the single clone tree that characterizes the cancer's evolutionary history. This can include revealing new subpopulations or disentangling single large subpopulations into smaller constituents. Clone trees built from multiple samples of the same cancer have helped identify factors associated with metastasis [8] and probed how treatment [9–11] or tumour microenvironment [12, 13] shape evolution. This, in turn, can inform strategies to counteract treatment resistance [14].

Current subclonal reconstruction methods [15–21] are severely limited in their ability to build clone trees based on large multi-sample studies. Most of these methods were designed for single cancer samples from which no more than three subclones can be discerned at typical whole-genome sequencing depths

[1]. Recent studies with greater sequencing depth and multiple cancer samples have revealed that a single cancer can have dozens of resolvable subclones [10]. Here we show that existing clone tree reconstruction methods become highly inaccurate on datasets with many subclones or many cancer samples, necessitating a new approach.

Here we introduce Pairtree, a new method that can accurately construct clone trees from up to 100 samples per cancer, revealing as many as 30 subclones. Pairtree outperforms a representative set of state-of-the-art clone tree reconstruction packages on simulated benchmark datasets of variable complexity. Pairtree is also the only method tested that can recover or improve on expert reconstructions of clone trees for 14 B-progenitor acute lymphoblastic leukemias (B-ALLs) containing up to 90 samples and 26 subclones per cancer.

3 Methods and results

3.1 Pairtree inputs and outputs

A clone tree represents the evolutionary history of a cancer. Fig. 1 outlines the process of clone tree reconstruction. Pairtree takes as input allele frequency data for point mutations detected in one or more samples from a single cancer. These data can be derived from whole-genome sequencing (WGS), whole-exome sequencing (WES), or targeted sequencing. Each bulk cancer sample is a mixture of genetically heterogeneous cells (Fig. 1a). For each mutation, Pairtree uses counts of variant and reference reads in each sample to estimate the variant allele frequency (VAF), i.e., the proportion of reads at a mutation's locus that contain the mutation. By correcting a mutation's VAF for copy-number aberrations (CNAs) affecting the locus, Pairtree computes an estimate of the proportion of cells in each sample carrying the mutation, termed the mutation's *subclonal frequency* [22] (Fig. 1b).

Pairtree outputs a set of possible clone trees explaining evolutionary relationships between the input mutations. Clone tree nodes correspond to cancerous subpopulations, while arrows (i.e., directed edges) extend from a subpopulation's node to the nodes representing its direct descendants (Fig. 1c). We define a subpopulation as those cells containing exactly the same subset of the somatic mutations input into Pairtree. In each cancer sample, each subpopulation is assigned a population frequency, representing what proportion of cells in that sample share the same mutation subset. Many, if not most, of a cancer's mutations will not be provided in the input because of incomplete genome coverage or because the mutations are too low in frequency to be detected.

Each subpopulation and its descendant subpopulations (both direct and indirect) form a subclone

(Fig. 1a). Pairtree assigns a tree-constrained subclonal frequency to each subclone in each cancer sample, which is equal to the sum of the population frequencies of all the subpopulations contained within the subclone (Fig. 1a-b). This relationship follows from the infinite sites assumption (ISA), which states that no site is mutated more than once during cancer evolution. The ISA implies that subpopulations inherit all the mutations of their parent populations, and that each mutation appears only once in the evolutionary history of the cancer. Though violations of the ISA occur [23], it remains broadly valid [24], and Pairtree can detect and discard ISA-violating mutations (Section 6.1.3). Pairtree and most other clone tree reconstruction methods use the ISA, though some methods allow limited ISA relaxations [25–27]. Using the ISA, Pairtree identifies what mutations belong to each subclone based on the estimated subclonal frequencies provided by the VAF data (Fig. 1b), then searches for clone trees whose structures allow subclonal frequencies that best match these estimates (Fig. 1c). Pairtree's output consists of a set of clone trees, each scored by a likelihood indicating how well the tree-constrained subclonal frequencies match the frequency estimates given by the VAF data. Although there is a single true clone tree explaining how subpopulations are related, this tree is not observed directly, and the input data often permit multiple solutions.

Grouping mutations into subclones is not necessary—algorithms can instead build clone trees in which each mutation is assigned to a unique subclone, yielding a mutation tree. However, because of limited resolution in the data's estimated subclonal frequencies, sets of mutations often have subclonal frequency estimates that are too similar to separate the mutations into distinct subclones. As such, the first step in clone tree reconstruction is often clustering mutations with similar estimated subclonal frequencies across all input samples, and associating subclones with these clusters. Mutation clustering can be performed with Pairtree (Section 10.1.1) or by another method [28–30] and input into Pairtree. This step simplifies clone tree reconstruction by reducing the number of subclones. Additionally, this approach permits more precise estimates of each subclone's subclonal frequency by combining data from the subclone's mutations (Section 6.2.8), at the risk of grouping together mutations from different subclones. As more cancer samples are used, each of which provides separate subclonal frequency estimates for the mutations, this caveat becomes less problematic.

3.2 Delineating ancestral relationships between pairs of subclones using the Pairs Tensor

Pairtree uses the estimated subclonal frequencies to predict the ancestral relationship between every subclone pair. These pairwise relationships then serve as a guide when Pairtree searches for clone trees that best fit the VAF data. Under the ISA [31], one of three mutually exclusive ancestral relationships exist between a pair of subclones A and B.

1. A is ancestral to B. Here, the subpopulation associated with A contains A's mutations but not B's. No cell subpopulation has B's mutations without also inheriting A's.

2. B is ancestral to A. This is as above, with the roles of A and B switched.

3. Neither A nor B is the ancestor of the other. In this case, they occur on different branches of the clone tree. Consequently, no subpopulations have both A's and B's mutations.

Each relationship constrains the frequencies that can be assigned to the two subclones (Section 6.1.3). For a given subclone pair, Pairtree combines the CNA-corrected VAF data for each subclone's mutations with a prior probability distribution incorporating these constraints, then uses Bayesian inference to compute the probability of each relationship type for the pair (Section 6.1). This yields a data structure termed the *Pairs Tensor*, the elements of which are the marginal posterior probability distributions over the three possible ancestral relationships for every subclone pair.

3.3 Using pairwise ancestry to guide the search for clone trees

Pairtree uses the Pairs Tensor to define a proposal distribution for a Markov Chain Monte Carlo (MCMC) algorithm [32] that samples from the posterior distribution over clone trees (Fig. 2). The algorithm's Metropolis-Hastings scheme generates proposal trees using two discrete distributions derived from the Pairs Tensor (Section 6.2.5). The first distribution helps choose an erroneous subclone to move within the tree, with each subclone's selection probability determined by how inconsistent its ancestral relationships to other subclones in the current tree are relative to the Pairs Tensor. The second distribution guides the choice of new parent for the selected subclone, evaluating potential destinations based on how much this inconsistency is reduced. Though other MCMC-based subclonal reconstruction methods also modify trees by moving subclones [15, 17, 33], they blindly select both the subclone to move and its destination. Pairtree, by contrast, considers the data when making these decisions, with the Pairs Tensor helping the method rapidly navigate to high-probability regions of clone-tree space.

Pairtree uses a MAP approximation of the clone tree's marginal likelihood, both for the Metropolis-Hastings accept-reject decision and to compute the tree's posterior probability. Computing a clone tree's likelihood requires a maximum a posteriori (MAP) estimate of the subclonal frequencies, using a Bayesian prior to enforce tree constraints. By this prior, the root subclone must have a subclonal frequency of 1 in every sample, as it corresponds to the germline and all subclones are descended from it. Additionally, the prior requires that every subclone has a frequency greater than or equal to the sum of its direct descendants' subclonal frequencies. Pairtree computes the MAP estimate using a fast approximate scheme [34] or a slower exact one (Section 6.3). A clone tree's likelihood is then defined by how well the variant and reference read counts for each mutation match the MAP subclonal frequencies under a binomial sequencing noise model.

3.4 Benchmarking Pairtree performance using novel scoring metrics

Evaluating Pairtree against other common subclonal reconstruction methods required developing new metrics, as previously developed metrics are limited to datasets with single cancer samples [21]. Here, we introduce two novel metrics better suited for the multi-sample domain that also permit uncertainty about the best-fitting clone tree.

The first, termed *VAF reconstruction loss*, uses likelihood to compare the data fit of a tree's subclonal frequencies to a baseline (Section 6.5.2). For simulated data, the baseline frequencies are the ground-truth frequencies used to generate the VAF data. For real data with an unknown ground truth, the baseline is MAP subclonal frequencies computed for an expert-constructed clone tree. Negative VAF losses indicate the evaluated frequencies have better data fit than the baseline.

The second evaluation metric, termed *relationship reconstruction error*, compares the structure of candidate clone trees to the ground truth (Section 6.5.3) using the evolutionary relationships between subclone pairs. To compute it, we construct an empirical Pairs Tensor from the clone tree solutions found by a method, then compare it via the Jensen-Shannon divergence (JSD) to a tensor based on the ground truth. As multiple clone trees may be consistent with the ground-truth subclonal frequencies, we construct the ground-truth Pairs Tensor by enumerating all trees consistent with these frequencies [35] and denoting the pairwise relationships between subclones that each expresses. Building this ground truth requires knowing the ground-truth subclonal frequencies with no measurement error, so this metric is best suited to simulated data.

For both metrics, we evaluate the quality of a solution set by computing the average over all trees reported by a method, weighted by the likelihood the method associates with each solution.

3.5 Selecting comparison methods and generating simulated data

Clone tree reconstruction methods use one of two approaches: exhaustive enumeration or stochastic search. To evaluate Pairtree, a stochastic search method, we compared it against three exhaustive enumeration methods (PASTRI [20], CITUP [16], and LICHeE [19]) and one stochastic search method (PhyloWGS [36]). All methods output multiple candidate clone trees.

We assessed method performance on 576 simulated datasets with variable read depths and numbers of subclones, cancer samples, and mutations. These included trees with 3, 10, 30, or 100 subclones. Three subclones are the most that can typically be resolved at WGS read depths of 50x [1]. Ten subclones are often discernible from multi-sample datasets [5], while 30 was the approximate maximum we could resolve in the high-depth, many-sample B-ALL data evaluated here [10]. We included datasets with 100 subclones to probe the methods' limits, anticipating challenges presented by future datasets. The number of simulated cancer samples ranged from 1 to 100. We designed the simulation process (Section 6.4.2) to generate realistic, diverse, and resolvable clone trees (Section 10.7). We did not include one- or three-sample datasets in the 30- and 100-subclone simulations, as resolving so many subclones from so few samples would be unrealistic. Methods were allowed up to 24 hours of wall-clock time to produce results.

Some caveats must be noted. LICHeE does not report subclonal frequencies for its solutions, so we used Pairtree to fit MAP frequencies to LICHeE's trees. Though LICHeE does not produce a likelihood, unlike the other methods here, it reports an error score for each tree that we interpreted as a likelihood when weighting its solutions. PhyloWGS, unlike other methods, could not use a fixed mutation clustering. This led to the method incorrectly merging clusters, causing artificially high VAF loss and relationship error. More generally, all methods except Pairtree failed to produce output on some simulated datasets. These failures stemmed from methods terminating without producing output, crashing outright, or failing to finish within 24 hours (see Section 10.3 for details).

3.6 Pairtree outperforms existing methods on simulated data

Fig. 3 summarizes how the methods performed on simulated data, with a method's scores reflecting its performance on only the datasets for which it produced output. Pairtree was the only method that produced results for all 576 simulations (Fig. 3a). Nevertheless, Pairtree fared better than comparison methods on trees with 30 or fewer subclones, succeeding on all datasets while achieving negative median VAF losses (Fig. 3b-c). In fact, Pairtree always produced lower error than other methods for every such dataset (Fig. S4). Pairtree also performed better than comparison methods with respect to relationship

error. In general, for 30 subclones or fewer, relationship error was almost zero when the number of cancer samples exceeded the number of subclones (Fig. S5b). For these cases, only one possible tree occurred (S10a), with Pairtree achieving low error by finding that tree or a close approximation thereof (S10b-c). When applied to datasets with 100 subclones, Pairtree had higher VAF losses (Fig. 3b) and relationship errors (Fig. 3c) than with fewer subclones. Pairtree outperformed other methods for 100-subclone trees with respect to VAF loss, except for 16 datasets (15%) where PhyloWGS performed better (Fig. S4).

CITUP failed on all datasets with ten or more subclones, and on 32% of three-subclone cases (Fig. 3a). All failures on three-subclone datasets occurred because CITUP crashed (Section 10.3). On ten-subclone datasets, 29% of CITUP runs ran out of time, with the other 71% failing because CITUP crashed. On the three-subclone cases where it ran successfully, its VAF loss was poor (Fig. 3b), perhaps because of a mismatch between its sequencing error model and the model used for computing VAF loss. Conversely, the method exhibited better relationship error than other non-Pairtree methods (Fig. 3c), suggesting its tree structures were more accurate.

PASTRI, which cannot run on datasets with more than 15 subclones [37], failed for 83% of three-subclone cases and 96% of ten-subclone cases (Fig. 3). For datasets with three or ten subclones, PASTRI produced output on 10%, terminated without producing a result on 84%, and ran out of time on the remaining 6% (Section 10.3). When it produced solutions, PASTRI generally performed well, reaching negative median VAF losses for three- and ten-subclone datasets, and relatively low relationship errors. Occasionally, PASTRI produced high-error solutions, with VAF losses up to 492 bits on the three-subclone datasets.

LICHeE fared better, producing results on all cases with 3, 10, or 30 subclones (Fig. 3). However, the method ran out of time for 92% of 100-subclone datasets. After Pairtree, LICHeE was the next-best performing method, with low VAF losses and moderate relationship errors on datasets with three or ten subclones, beating PhyloWGS on both measures. LICHeE performed less well on 30-subclone cases, where it exhibited lower VAF losses than PhyloWGS but higher relationship errors.

PhyloWGS produced clone trees for all datasets with 30 or fewer subclones (Fig. 3). In these cases, PhyloWGS generally had worse VAF losses and relationship errors than Pairtree or LICHeE, except for the 30-subclone datasets, where it had better relationship error than LICHeE but worse VAF loss. PhyloWGS performed better than other non-Pairtree methods on 100-subclone cases, where it finished within 24 hours for 62% of such datasets, but usually had higher VAF losses than Pairtree (Fig. S4).

Relationship error can also be measured for the Pairs Tensor alone, without requiring trees. The Pairs Tensor estimates pairwise relationships accurately (Fig. 3c), requiring only a fraction of the computational

resources of the full Pairtree method (Fig. S8). Although the Pairs Tensor does slightly worse than Pairtree on trees with 30 or fewer subclones, it has less relationship error than any other method. On datasets with 100 subclones, the Pairs Tensor was better able to delineate pairwise relationships between subclones than the full Pairtree method (Fig. 3c).

3.7 Pairtree improves with more cancer samples, but other methods worsen

After controlling for other variables, all methods except Pairtree performed worse when provided more cancer samples. CITUP and PASTRI's failure rates increased with the number of cancer samples (Fig. 4a). Though LICHeE and PhyloWGS produced output for all cases with 30 subclones or fewer, they had higher VAF losses with more cancer samples (Fig. 4b). By contrast, Pairtree never failed and had nearly zero median VAF loss regardless of the number of simulated cancer samples on datasets with 30 subclones or fewer (Fig. 4a-b). Relationship errors decreased for both full Pairtree and the Pairs Tensor with more samples (Fig. 4c). LICHeE, conversely, exhibited rapidly increasing error with more samples, while PhyloWGS' performance fluctuated.

3.8 Pairtree performs better than human experts on complex real clone tree reconstructions

We applied Pairtree, CITUP, LICHeE, PASTRI, and PhyloWGS to genomic data from 14 B-ALL patients [10]. Samples were obtained at diagnosis and relapse for each patient. In addition, each sample was transplanted into immunodeficient mice, generating multiple patient-derived xenografts (PDXs). The patient samples were subjected to WES, while the PDXs were subjected to targeted sequencing based on leukemic variants found in the patient WES data. There were 16 to 509 mutations called per patient (median 40), clustered into 5 to 26 subclones per patient (median 8). By combining patient and PDX samples, we obtained between 13 and 90 tissue samples per cancer (median 42). Across cancers, the median read depth was 212 reads.

To define ground truth for these datasets, we built high-quality clone trees for each dataset manually, subjecting them to extensive review and refinement before evaluating them for biological plausibility [10]. We then fit MAP subclonal frequencies to these trees using Pairtree, yielding the *expert-derived baseline*. As with simulated data, methods that improve on the baseline achieve negative VAF losses.

CITUP and PASTRI failed on 13 of the 14 cancers, and so we excluded these methods from the comparison. Pairtree found trees as good as, or slighter better than, the expert baseline for 12 of 14

cancers (Fig. 5), resulting in VAF losses between 0 and -0.05 bits. On two cancers, Pairtree inferred clone trees that fit the VAF data substantially better than the expert baseline, resulting in negative losses of -0.32 bits and -1.42 bits. LICHeE beat the baseline for one cancer, reaching a negative loss of -0.86 bits; (nearly) matched the baseline for four other patients, incurring between 0 and 0.11 bits of loss; and had substantially worse VAF losses for the remaining nine patients. PhyloWGS suffered at least 0.35 bits of loss on all patients, reaching a median VAF loss of 4.42 bits. As PhyloWGS could not adhere to the expert-derived clustering, unlike other methods, it often merged clusters incorrectly, causing high VAF loss.

3.9 Consensus graphs intuitively illustrate uncertainty in clone trees

Pairtree provides interactive visualizations to help navigate the multiple clone tree solutions that it produces for each dataset (Fig. 6). By using the data likelihoods associated with each solution as weights, Pairtree produces a *weighted consensus graph*, in which the nodes represent subclones, and each directed edge is assigned a weight equal to the marginal probability that it appears in a clone tree drawn from the empirical clone tree distribution produced by Pairtree. Thus, the consensus graph summarizes the estimated posterior probability of each parental relationship between subclones. These summaries are useful for interpreting Pairtree's results, as they provide a concise representation of the evolutionary relationships supported by the data, alongside the confidence underlying each. By taking the maximum-weight spanning tree of this graph, the user can generate a single consensus tree. To demonstrate the consensus graph's utility, we ran Pairtree multiple times on one of the B-ALL cases from Fig. 5, using variable numbers of cancer samples (Fig. 6). As we provided more cancer samples, confidence in evolutionary relationships increased, until all parents were resolved with near certainty. Providing more samples can also correct erroneous inferences—with 30 samples, population 8 appeared to be the likely parent of population 15, but with 90 samples, it became clear that population 15's parent is population 6.

4 Discussion

Pairtree is the first automated method that reliably recovers large, complex clone trees from bulk DNA sequencing data. On simulated data, Pairtree recovers nearly perfect clone trees for cancer datasets with up to 30 subclones. On 14 B-ALL cancers, with up to 26 subclones and 90 samples per cancer, Pairtree's clone trees are objectively as good as, or better than, those manually constructed by experts. No other

tested method was consistently accurate on real or simulated benchmarks containing ten subclones or more. Pairtree was also the only method whose clone trees reliably became more accurate when more samples were used in the reconstructions. This is surprising—as each cancer sample provides additional information about evolutionary relationships between subpopulations, subclonal reconstruction problems should become easier with more cancer samples, not more difficult.

A key factor in Pairtree's success is its efficient search through the space of clone trees. Beyond ten subclones, this tree space quickly becomes too large for exhaustive enumeration (CITUP) or unguided stochastic search (PhyloWGS). Even methods that reduce the search space by applying hard constraints excluding some parent-child relationships (LICHeE, PASTRI) still fail to recover more complex clone trees. Recovering complex trees requires more cancer samples than for simple trees, but when faced with many samples, the hard constraints become inaccurate and exclude the correct solution (Section 10.4). By contrast, Pairtree's stochastic tree search is guided by the Pairs Tensor, which provides soft constraints defined by a well-motivated probability model. Consequently, Pairtree's constraints become more precise as more cancer samples are provided, without excluding the true clone tree.

As Pairtree's performance degrades on the 100-subclone benchmarks, alternative search strategies may be necessary for very large clone trees. While Pairtree almost always fails to correctly resolve a subclone's parent (Fig. S10c), it achieves relatively low relationship error (Fig. S10d), suggesting it may be capturing the coarse tree structure. If so, Pairtree may fare better using a tiered approach, in which it would group together subclones with similar pairwise relations to others, build subtrees for each group separately, and then connect the subtrees using the groups' pairwise relations to compose the full clone tree. Given 100 subclones with 10 or more cancer samples, the Pairs Tensor is already better than Pairtree itself at capturing the correct evolutionary relationships between subclones (Fig. S5b-c). Future work should focus on understanding what conditions (e.g., high read depth or many cancer samples) under which the Pairs Tensor converges to a partial clone tree [35] that succinctly summarizes all clone trees with non-negligible posterior probability.

Throughout this work, we have stressed performance metrics that recognize there are often many solutions consistent with observed data (Section 10.6). These metrics extend previous ones we developed [21] to score multiple candidate solutions from a method against a single ground-truth tree. Our new metrics permit the ground truth to be uncertain, with multiple potential truths equally consistent with noise-free observations. In general, characterizing uncertainty in clone tree reconstructions is critical. Even when methods produce multiple solutions, users typically want a single answer, and so select the highest-scoring tree while neglecting other credible candidates that fit their data nearly as well.

Consequently, they lose information about which evolutionary relationships between subclones are well-defined by the data, and which are uncertain because they have multiple equally likely possibilities. If users are to benefit from a method's ability to produce multiple solutions, the method must provide tools for interpreting this uncertainty. Pairtree's weighted consensus graph characterizes the uncertainty present in each evolutionary relationship, depicting all credible possibilities and the confidence underlying each (Fig. 6). This allows users to make informed conclusions about their cancer datasets.

In summary, Pairtree can reconstruct highly accurate trees representing the evolutionary relationships among up to 30 subclones based on sequencing data from up to 100 samples from a cancer. By scaling to many more subclones and cancer samples than past approaches, and by illustrating the uncertainty present in solutions, Pairtree can address questions in many cancer research domains. These include understanding the origin and progression of tumours, measuring tumour age and heterogeneity, mapping out mechanisms of tumour adaptation to therapy, and understanding the relationship between primaries and metastases. In the future, the Pairtree framework can be extended to scale to even more complex trees, integrate single-cell sequencing data (Section 10.9), and permit violations of the infinite sites assumption (Section 10.8).

5 Figures

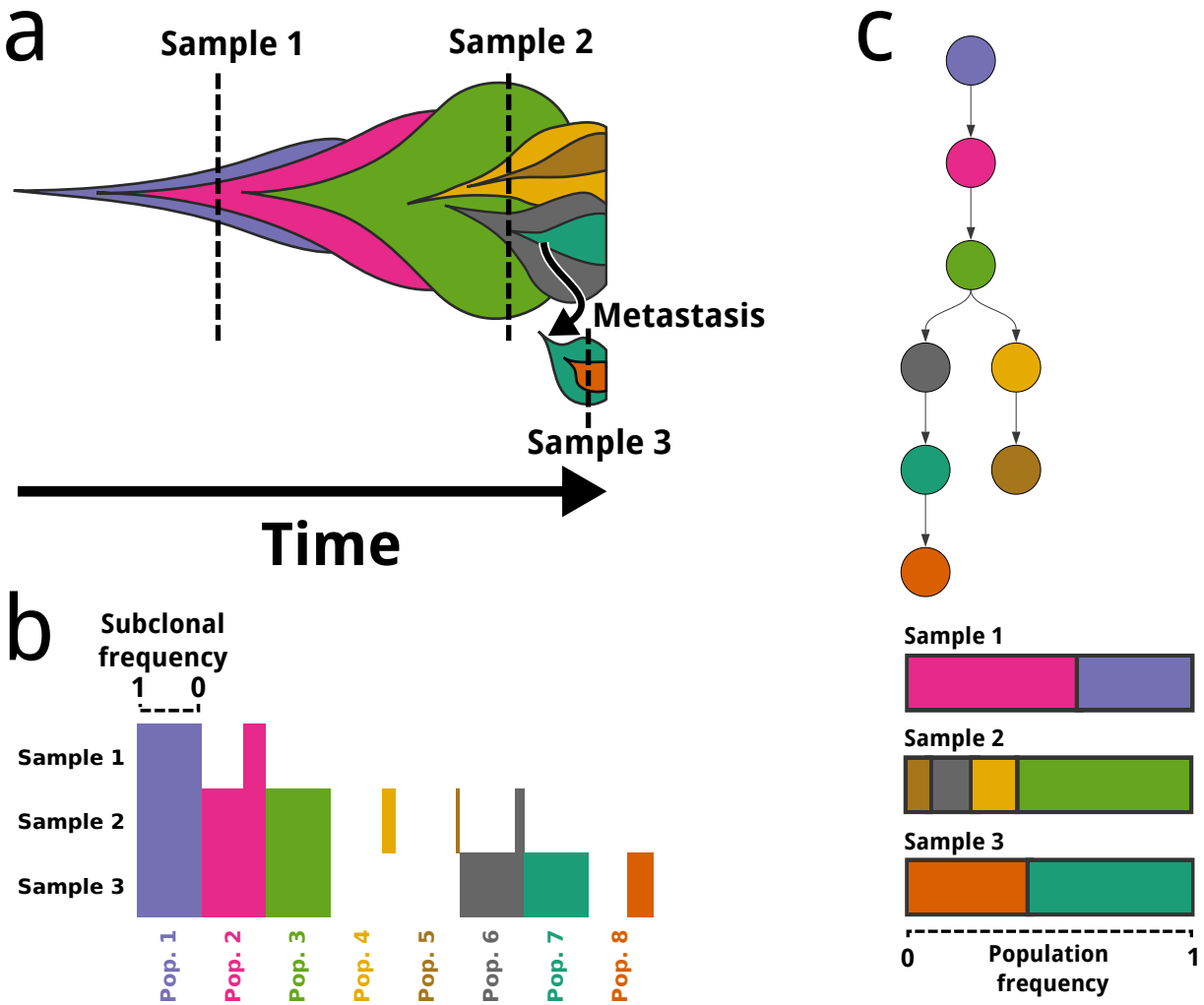

Figure 1: **Construction of clone trees from multiple cancer samples. a.** Schematic illustrates cancer development under the clonal evolution model. Each colour represents a genetically distinct subpopulation. Each subpopulation emerges within the mass of its parent. The leftmost point for a subpopulation denotes the cell that was its most recent common ancestor. Dashed vertical lines indicate when and where cancer samples were taken. The relative abundance of each subpopulation in a cancer sample, including any nested descendent subpopulations composing a subclone, is represented by the height of that subpopulation or subclone along the sample's dashed line. **b.** Horizontal bar plot showing idealized input to clone tree reconstruction algorithms. Bar length indicates the subclonal frequency of each subpopulation and its descendants (column) in each sequenced sample (row). The clonal evolution model asserts that a subpopulation's point mutations are inherited by its descendants. Consequently, mutation VAFs in DNA sequencing data provide estimates of subclonal frequencies, corresponding to the proportion of cells that originated from a subclonal population and its descendants. **c.** Clone tree representing the ancestry of subpopulations (top). Nodes indicate subpopulations. Arrows extend from each subpopulation to its direct descendants. Inferred frequencies of each subpopulation in each sample are based on the clone tree and mutation frequency data (bottom).

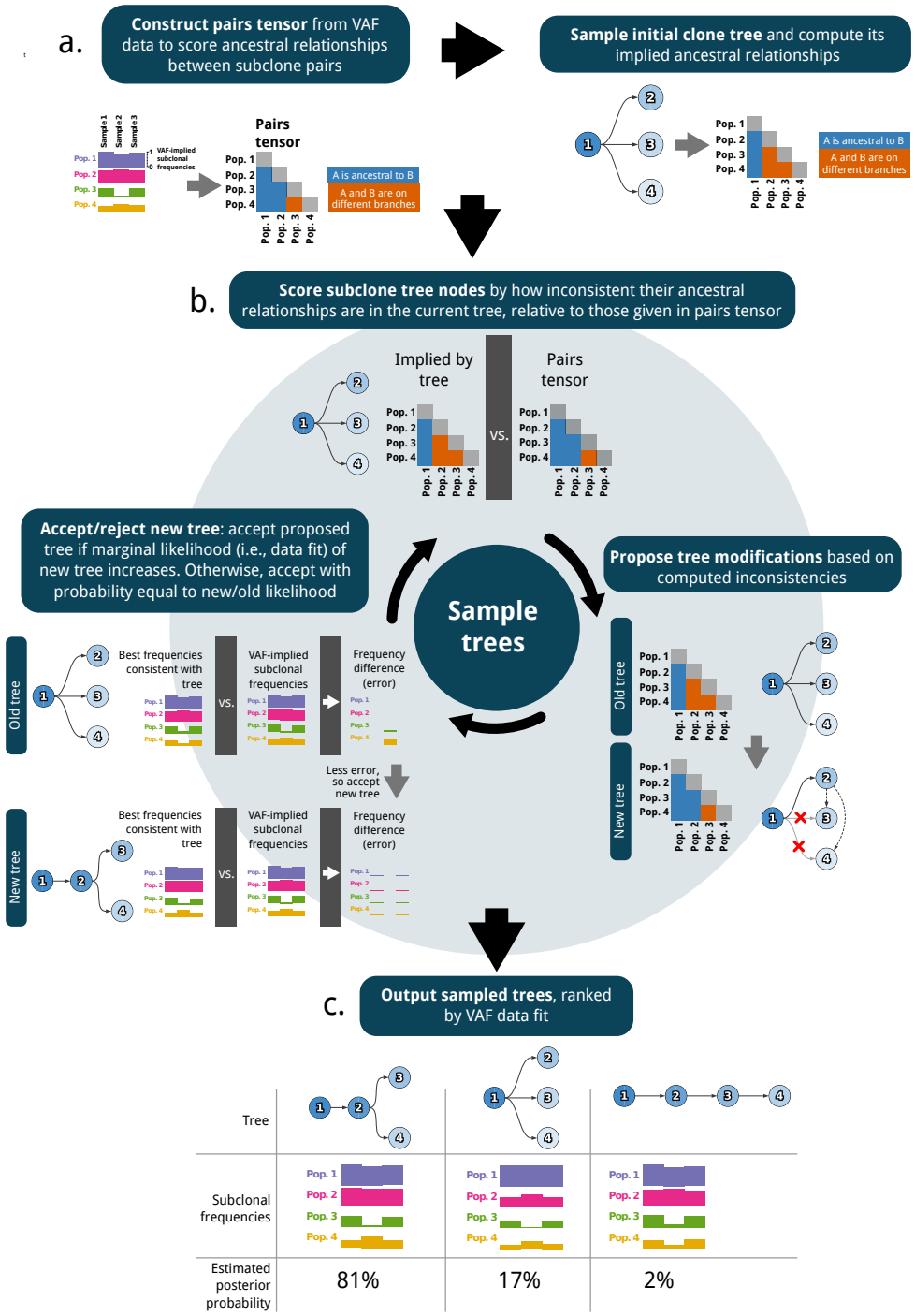

Figure 2: **The Pairtree algorithm. a.** Pairtree uses VAF data to compute the Pairs Tensor, which provides the probability of every possible ancestral relationship between subclone pairs (left). An initial clone tree is built using relationships scored by the Pairs Tensor. **b.** Pairtree samples trees using Markov Chain Monte Carlo. The method proposes tree modifications by identifying a subclone whose ancestral relationships in the current tree are assigned low probability by the Pairs Tensor (top), then ascertaining where that subclone can be moved within the tree to increase its ancestral relationship probabilities (bottom right). Proposed trees are then accepted or rejected based on their marginal likelihoods that reflect how well they fit the VAF data (bottom left). **c.** Sampled clone trees are returned along with posterior probability estimates proportional to the marginal likelihood of each tree. *VAF*, variant allele frequency.

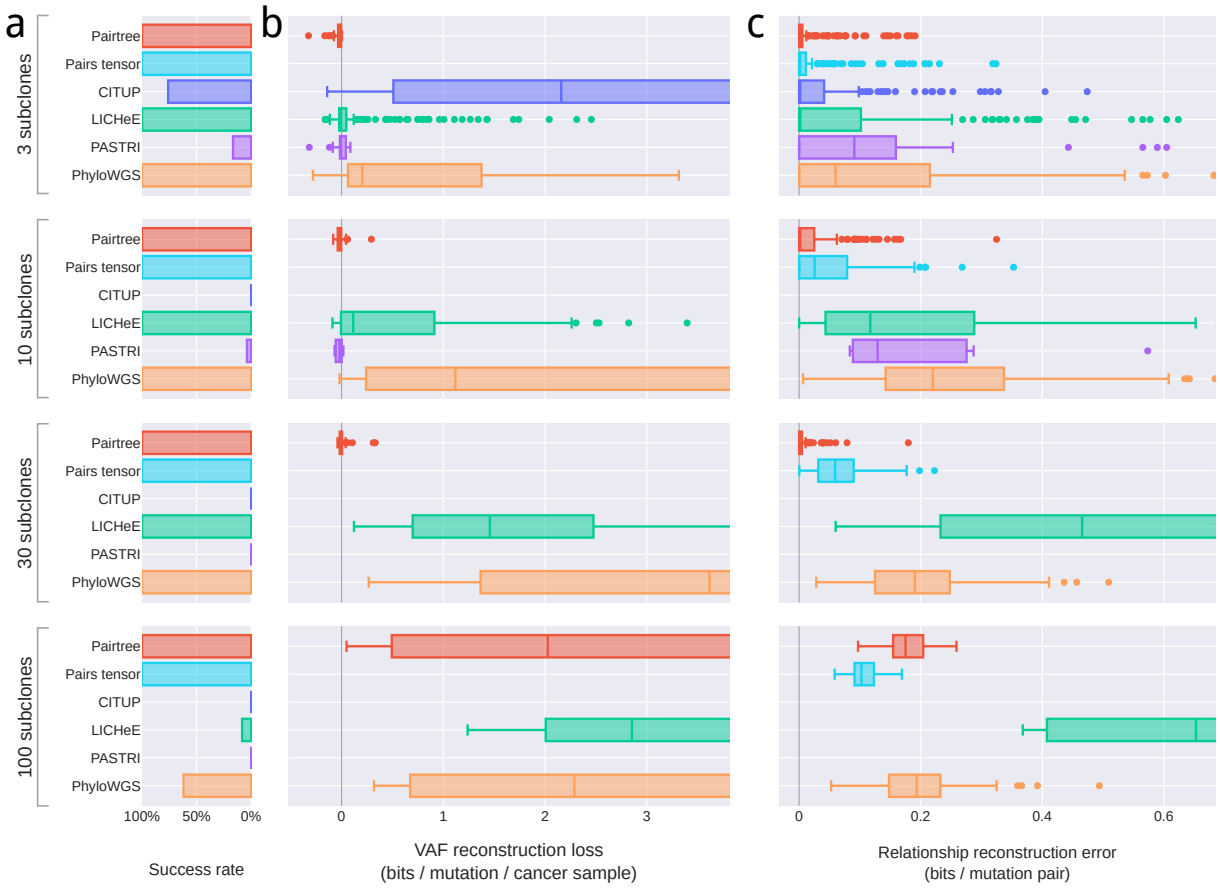

Figure 3: **Benchmark performance on 576 simulated datasets.** Simulations are grouped by number of subclones (rows). **a.** Bar plots show each method's success rate in the group. Successes are reconstruction problems for which the method produced at least one tree in 24 hours (wall-clock time) and did not crash. **b.** Boxplots show distributions of VAF reconstruction losses for a method on a problem group. Scores reflect only datasets where a method ran successfully. VAF reconstruction loss is the decrease in average, per-mutation log likelihood of VAF data using subclonal frequencies assigned by the method, when compared to the true frequencies used to generate the data. Negative loss indicates better VAF reconstructions than true trees, while high loss indicates inaccurate tree structures. Mid-lines in box plots indicate medians. Plots are truncated at four bits. Fig. S1 shows untruncated distributions. **c.** Boxplots show distributions of relationship reconstruction error in each group for each method's successful runs. Relationship reconstruction error is measured as the average Jensen-Shannon divergence per subclone pair between the true distributions over pairwise relations, and empirical distributions computed from the trees output by a method. Errors can range between zero bits (perfect match) and one bit (complete mismatch), and are truncated at 0.7 bits. Fig. S2 shows untruncated distributions.

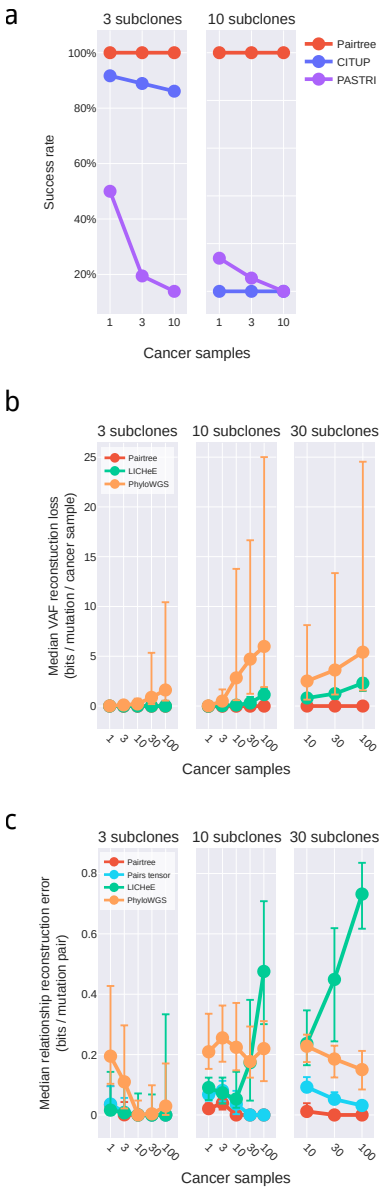

Figure 4: **Performance on simulated datasets as a function of number of subclones and cancer samples. a.** Method success rate. For CITUP and PASTRI, success rate depended on the number of subclones and/or cancer samples in datasets. Pairtree, LICHeE, and PhyloWGS succeeded on all datasets depicted. **b.** Median VAF reconstruction loss as a function of number of samples. For LICHeE and PhyloWGS, VAF loss increases with more cancer samples. **c.** Median relationship reconstruction error as a function of number of samples. LICHeE's error generally increased with more cancer samples, while other methods showed the opposite effect. Error bars represent the first and third quartiles in (b-c).

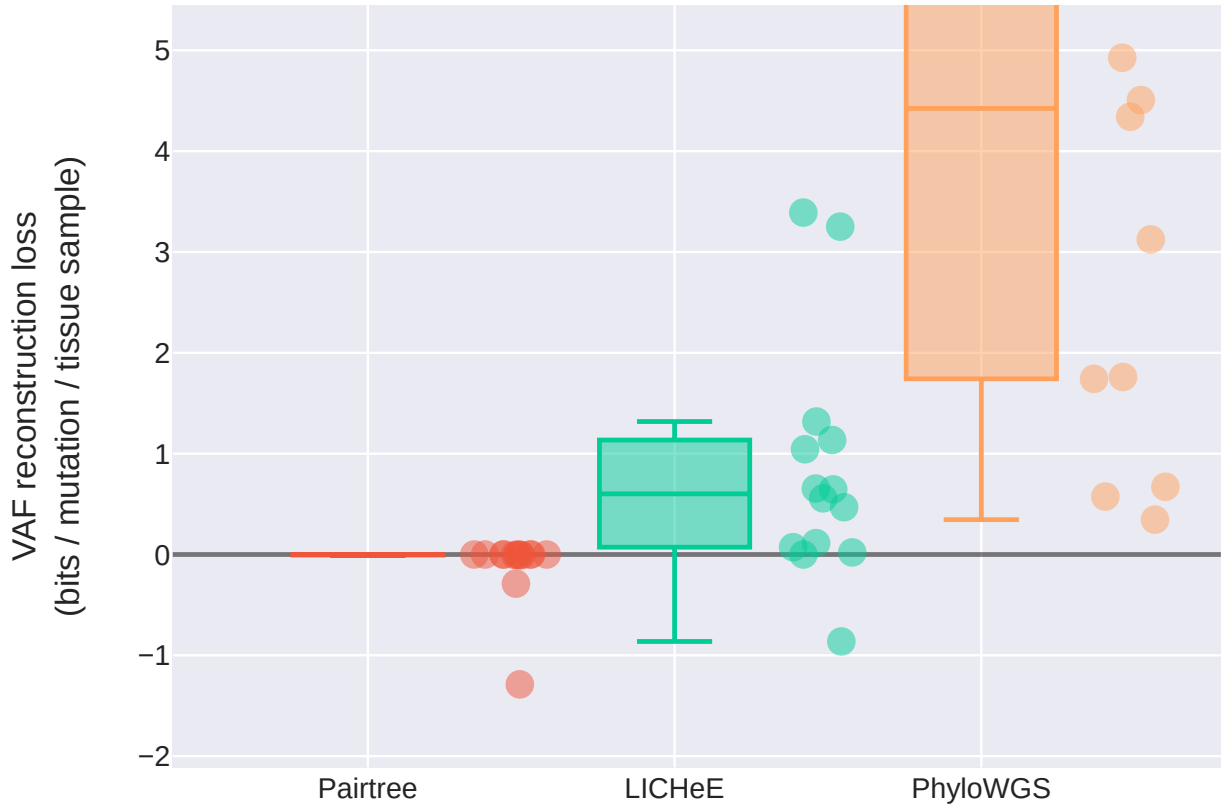

Figure 5: **VAF reconstruction loss for 14 B-ALL patient datasets.** The number of cancer samples for each dataset ranged from 13 to 90. Mid-lines in box plots indicate medians. CITUP and PASTRI each failed on 13 of 14 datasets and so are not shown. VAF reconstruction losses are reported as a negative log likelihood normalized to the number of mutations and cancer samples, relative to the MAP subclonal frequencies for expert-derived trees. Lower loss indicates better performance, while negative loss corresponds to performance better than human experts. Fig. S3 shows untruncated distributions.

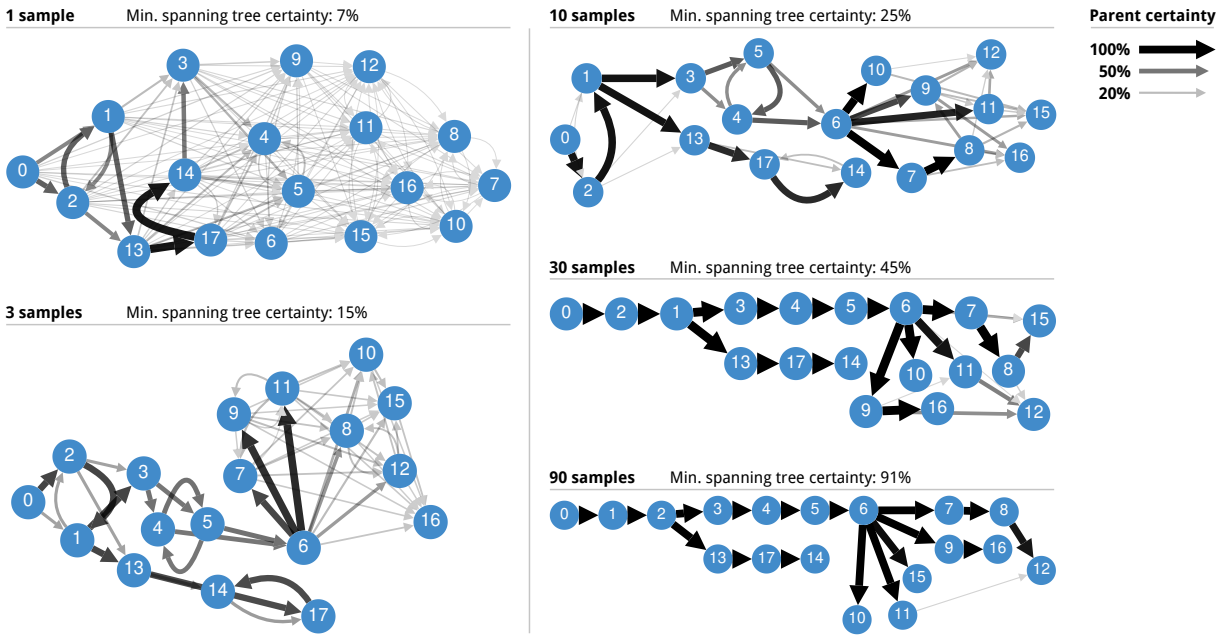

Figure 6: **Consensus graph visualization of posterior tree distributions.** These consensus graph visualizations are based on one of the 14 B-ALL cancers analyzed with Pairtree, for which 90 cancer samples were available. Consensus graphs are shown for variable numbers of samples, ranging from a single sample to all 90. All edges with less than 5% posterior certainty are hidden. The minimum spanning tree certainty indicates, if the graph is fully connected such that every subclone has at least one possible parent, how confident the least-certain single parent is.

6 Methods

6.1 Computing pairwise relations

6.1.1 Establishing a probabilistic likelihood for pairwise relations

Let A and B represent two distinct mutations. We denote their observed read counts, encompassing both variant and reference reads, as x_A and x_B. Assuming both mutations obey the ISA, the pair (A, B) must fall in one of four pairwise relationships, denoted by M_{AB}.

1. $M_{AB} = \textit{coincident}$, meaning A and B are co-occurring. That is, A and B occur in precisely the same cell subpopulations, such that A is never present without B and vice versa. This reflects that A and B occurred proximal to each other in evolutionary time, such that we cannot distinguish an intermediate subpopulation that occurred between them.

2. $M_{AB} = \textit{ancestor}$, meaning A is ancestral to B. That is, A occurred in a population ancestral to B, such that some cells possess A without B, but no cell has B without A. This reflects that A preceded B.

3. $M_{AB} = \textit{descendent}$, meaning B is ancestral to A. This mirrors relationship 2, reflecting B preceded A.

4. $M_{AB} = \textit{branched}$, meaning A and B occurred on different branches of the clone tree, such that they never occur in the same set of cells. This relationship confers no information about the respective timing of A and B.

To the four possible relationships above, we add a fifth, termed the *garbage relation* and denoted by $M_{AB} = \textit{garbage}$. This represents mutation pairs with conflicting evidence for different relationships amongst the four already defined, providing a baseline against which the other four relationships can be compared. This catch-all category assumes that there is no consistent evolutionary relationship denoted by the subclonal frequencies of the two mutations across cancer samples, so it may represent ISA violations arising from the four-gamete test [38]. The garbage relation can also represent unreported CNAs that skew the relationship between VAF and subclonal frequency.

The likelihood of the pair's relationship is written as $p(x_A, x_B | M_{AB})$. First, we note that every cancer sample s can be considered independently of others, allowing us to factor the likelihood.

$$p(x_A, x_B|M_{AB}) = \prod_s p(x_{As}, x_{Bs}|M_{AB})$$

To compute the pairwise relationship likelihood for one cancer sample s, we integrate over the possible subclonal frequencies associated with the subclones that gave rise to mutations A and B, representing the proportions of cells in the cancer sample carrying the mutations. We indicate these subclonal frequencies as ϕ_{As} and ϕ_{Bs}.

$$p(x_{As}, x_{Bs}|M_{AB}) = \iint d\phi_{As} d\phi_{Bs} p(x_{As}|x_{Bs}, M_{AB}, \phi_{As}, \phi_{Bs}) p(x_{Bs}|M_{AB}, \phi_{As}, \phi_{Bs}) p(\phi_{As}, \phi_{Bs}|M_{AB})$$

As each mutation's likelihood is a function solely of its subclonal frequency, and is independent of both the other mutation and the pairwise relationship, we can simplify the integral.

$$p(x_{As}, x_{Bs}|M_{AB}) = \iint d\phi_{As} d\phi_{Bs} p(x_{As}|\phi_{As}) p(x_{Bs}|\phi_{Bs}) p(\phi_{As}, \phi_{Bs}|M_{AB}) \tag{1}$$

6.1.2 Defining a binomial observation model for read count data

We can now begin providing concrete definitions for each factor in the integral given in Eq. (1). For mutation $j \in \{A, B\}$ from cancer sample s, whose observed read count data are represented by x_{js}, we define $p(x_{js}|\phi_{js})$ using the following notation:

- ϕ_{js}: subclonal frequency of subclone where j originated
- V_{js}: number of genomic reads of the j locus where the variant allele was observed
- R_{js}: number of genomic reads of the j locus where the reference allele was observed
- ω_{js}: probability of observing the variant allele in a subclone containing j. Equivalently, this can be thought of as the probability of observing the variant allele in a cell bearing the j mutation. Thus, in a diploid cell, $\omega_{js} = \frac{1}{2}$.

Observe that ω_{js} can be used to indicate changes in ploidy. For instance, a variant lying on either of the sex chromosomes in human males would have $\omega_{js} = 1$, since males possess only one copy of the X and Y chromosomes, so no wildtype allele would be present. Alternatively, ω_{js} can indicate clonal

copy number changes, such that all cancer samples in a sample bore the same CNA. If, for instance, the founding cancerous subclone bearing a mutation underwent a duplication of the wildtype allele, then, once the mutation arose in a descendent subclone, every cell within that subclone would contribute two wildtype alleles and one variant allele. Thus, in this instance, we would have $\omega_{js} = \frac{1}{3}$. While this representation requires that the CNA be clonal, any SNVs affected by the CNA can be subclonal, and can in fact belong to different subclones.

Though this scheme can represent clonal CNAs, it cannot do so for subclonal CNAs. Fundamentally, the tree-building algorithm requires converting the observed $\text{VAF}_{js} = \frac{V_{js}}{V_{js}+R_{js}}$ values into estimates of subclonal frequencies $\hat{\phi}_{js} = \frac{\text{VAF}_{js}}{\omega_{js}} \approx \phi_{js}$. If a subclonal CNA overlapping the mutation j occurs, different subclones will contribute different numbers of alleles to the cancer sample, implying this relationship is no longer valid. While the model could be extended to place subclonal CNA events on the clone tree and estimate how they change $\hat{\phi_{js}}$, the Pan-Cancer Analysis of Whole Genomes projects [39] reported frequent disagreement on allele-specific copy numbers among subclonal CNA-calling algorithms [1], and thus they discarded variants in regions affected by subclonal CNAs before constructing clone trees.

Using this notation, let the likelihood of observing V_{js} variant reads for mutation j in sample s, given a subclonal frequency ϕ_{js}, be defined by the binomial. We have $V_{js} + R_{js}$ observations of j's genomic locus, and probability $\omega_{js}\phi_{js}$ of observing a variant read, representing the proportion of alleles in the sample carrying the variant.

$$p(x_{js}|\phi_{js}) = \text{Binom}(V_{js}|V_{js} + R_{js}, \omega_{js}\phi_{js})$$

6.1.3 Defining constraints on subclonal frequencies imposed by pairwise relationships

To be fully realized, the likelihood Eq. (1) now requires only $p(\phi_{As}, \phi_{Bs}|M_{AB})$ to be defined. We use this factor to represent whether ϕ_{As} and ϕ_{Bs} are consistent with the relationship M_{AB}. For the ancestor, descendent, and branched relationships, the subclonal frequencies ϕ_{As} and ϕ_{Bs} dictate whether a relationship is possible.

$$p(\phi_{As}, \phi_{Bs}|M_{AB} = \textit{ancestor}) = \begin{cases} C & \text{iff } 1 \geq \phi_{As} \geq \phi_{Bs} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p(\phi_{As}, \phi_{Bs}|M_{AB} = \textit{descendent}) = \begin{cases} C & \text{iff } 0 \leq \phi_{As} \leq \phi_{Bs} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

$$p(\phi_{As}, \phi_{Bs}|M_{AB} = \textit{branched}) = \begin{cases} C & \text{iff } \phi_{As} \geq 0 \wedge \phi_{Bs} \geq 0 \wedge \phi_{As} + \phi_{Bs} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

The subclonal frequencies ϕ_{As} and ϕ_{Bs} may each take values on the $[0, 1]$ interval. Thus, $p(\phi_{As}, \phi_{Bs}|M_{AB})$ for $M_{AB} \in \{\textit{ancestor}, \textit{descendent}, \textit{branched}\}$ are each non-zero only inside a right triangle lying within the unit square on the Cartesian plane with corners at $(\phi_{As}, \phi_{Bs}) \in \{(0,0), (0,1), (1,0), (1,1)\}$. The location of the triangle within the unit square differs for each of the three M_{AB} relationships, but all have an area of $\frac{1}{2}$. Consequently, to ensure $\iint d\phi_{As} d\phi_{Bs} p(\phi_{As}, \phi_{Bs}|M_{AB}) = 1$, we set $C = \frac{1}{\frac{1}{2}} = 2$. Thus, $p(\phi_{As}, \phi_{Bs}|M_{AB}) = C = 2$ when ϕ_{As} and ϕ_{Bs} are consistent with M_{AB}, and zero otherwise.

We must still define the remaining two relationships $M_{AB} \in \{\textit{coincident}, \textit{garbage}\}$. The garbage relationship permits all combinations of ϕ_{As} and ϕ_{Bs} lying within the unit square, such that $p(\phi_{As}, \phi_{Bs}|M_{AB} = \textit{garbage}) = 1$. Consequently, unlike the previous three relationships, the garbage relationship imposes no constraints on ϕ_{As} and ϕ_{Bs} relative to each other.

$$p(\phi_{As}, \phi_{Bs}|M_{AB} = \textit{garbage}) = \begin{cases} 1 & \text{iff } \{\phi_{As}, \phi_{Bs}\} \subset \{x|0 \leq x \leq 1\} \\ 0 & \text{otherwise} \end{cases}$$

The garbage relationship serves to establish a baseline against which evidence for the non-garbage relationships can be evaluated. Observe that, in Eq. (1), $p(x_{As}|\phi_{As})p(x_{Bs}|\phi_{Bs})$ is integrated over the unit square when $M_{AB} = \textit{garbage}$. Conversely, when $M_{AB} \in \{\textit{ancestor}, \textit{descendent}, \textit{branched}\}$, we integrate $p(x_{As}|\phi_{As})p(x_{Bs}|\phi_{Bs})$ over a triangle covering half the square. Consequently, $p(x_{AS}, x_{Bs}|M_{AB} = \textit{garbage}) \leq \frac{1}{2} p(x_{AS}, x_{Bs}|M_{AB} \in \{\textit{ancestor}, \textit{descendent}, \textit{branched}\})$. This arises because $p(\phi_{As}, \phi_{Bs}|M_{AB}) = 2$ for subclonal frequencies consistent with $M_{AB} \in \{\textit{ancestor}, \textit{descendent}, \textit{branched}\}$, while $p(\phi_{As}, \phi_{Bs}|M_{AB}) = 1$ for subclonal frequencies consistent with $M_{AB} = \textit{garbage}$. When the read counts for the mutations A and B clearly permit one of the three non-garbage relationships, most of the probability mass of the two

associated binomials will reside within the simplex permitted by the relationship, and so the evidence for the non-garbage relationship will be nearly double that of the evidence for garbage. Conversely, when the read counts push most of the binomial mass outside the permitted simplex, the non-garbage evidence will be substantially lower than the baseline provided by garbage.

By considering accumulated evidence across many cancer samples, the garbage model's utility becomes clear. If, across many cancer samples for a mutation pair, the evidence for one non-garbage relationship is consistently favoured over others, then that relationship will emerge as the most likely when the evidence is considered collectively across samples. However, if different cancer samples favour different relationship types, the steady accumulation of the baseline garbage evidence could, in concert, be more than the evidence for any of the other three relations, meaning garbage would be declared as the most likely relationship for the mutation pair. Mutations that make up many pairs with high garbage evidence are best excluded from clone tree construction, as such mutations likely suffered from uncalled CNAs, violations of the ISA, or highly erroneous read count data.

The only undefined relationship remaining is $M_{AB} = \mathit{coincident}$. As the coincident relationship dictates that two mutations arose from the same subclone, and so share the same subclonal frequency, the corresponding constraint is defined thusly:

$$p(\phi_{As}, \phi_{Bs} | M_{AB} = \mathit{coincident}) = \begin{cases} 1 & \text{iff } 0 \leq \phi_{As} = \phi_{Bs} \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

6.1.4 Efficiently computing evidence for ancestral, descendent, and branched pairwise relationships

We now consider how to compute the pairwise likelihood given in Eq. (1) for $M_{AB} \in \{\mathit{ancestor}, \mathit{descendent}, \mathit{branched}\}$.

$$p(x_{As}, x_{Bs} | M_{AB}) = \iint d\phi_{As} d\phi_{Bs} p(x_{As} | \phi_{As}) p(x_{Bs} | \phi_{Bs}) p(\phi_{As}, \phi_{Bs} | M_{AB})$$

Observe that we can rearrange the integral to move the factor corresponding to the mutation A observations outside the inner integral.

$$p(x_{As}, x_{Bs}|M_{AB}) = \int d\phi_{As} p(x_{As}|\phi_{As}) \int d\phi_{Bs} p(x_{Bs}|\phi_{Bs}) p(\phi_{As}, \phi_{Bs}|M_{AB})$$

Now, because $p(\phi_{As}, \phi_{Bs}|M_{AB})$ is piecewise-constant when $M_{AB} \in \{ancestor, descendent, branched\}$, we can, for these relationships, impose this factor's effect by changing the integration limits. Let $L(\phi_{As}, M_{AB})$ and $U(\phi_{As}, M_{AB}))$ represent functions whose outputs are the lower and upper integration limits, respectively, for the inner integral whose differential is $d\phi_{Bs}$, as a function of ϕ_{As} and the relationship M_{AB}. These functions are defined thusly:

$$(L(\phi_{As}, M_{AB}), U(\phi_{As}, M_{AB})) = \begin{cases} (0, \phi_{As}) & \text{if } M_{AB} = ancestral \\ (\phi_{As}, 1) & \text{if } M_{AB} = descendent \\ (0, 1 - \phi_{As}) & \text{if } M_{AB} = branched \end{cases}$$

By writing the inner integral using these integration limits, and limiting the outer integral to the $[0, 1]$ interval permitted for ϕ_{As}, the factor $p(\phi_{As}, \phi_{Bs}|M_{AB})$ can be replaced by 2, as it is constant over the interval of integration.

$$p(x_{As}, x_{Bs}|M_{AB}) = \int_0^1 d\phi_{As} 2p(x_{As}|\phi_{As}) \int_{L(\phi_{As}, M_{AB})}^{U(\phi_{As}, M_{AB})} d\phi_{Bs} p(x_{Bs}|\phi_{Bs})$$

To render the inner integral more computationally convenient, rather than integrate over ϕ_{Bs}, we would prefer to integrate over $q_{Bs} \equiv \omega_{Bs}\phi_{Bs}$. Thus, we will integrate by substitution, using $\frac{dq_{Bs}}{d\phi_{Bs}} = \omega_{Bs}$.

$$\begin{aligned} \int d\phi_{Bs} p(x_{Bs}|\phi_{Bs}) &= \int d\phi_{Bs} \text{Binom}(V_{Bs}|V_{Bs} + R_{Bs}, \omega_{Bs}\phi_{Bs}) \\ &= \frac{1}{\omega_{Bs}} \int dq_{Bs} \text{Binom}(V_{Bs}|V_{Bs} + R_{Bs}, q_{Bs}) \end{aligned} \tag{2}$$

Observe that the inner integral is now simply integrating the binomial PMF over its parameter q_{Bs}. To compute this integral, we rely on the following equivalence between this integral and the incomplete beta function β:

$$\begin{aligned}\int_0^b dp\text{Binom}(k|N,p) &= \int_0^b dp\binom{N}{k}p^k(1-p)^{N-k} \\ &= \binom{N}{k}\int_0^b dp p^k(1-p)^{N-k} \\ &= \binom{N}{k}\beta(b|k+1,N-k+1)\end{aligned}$$

Now we can compute the integral over an arbitrary limit by the fundamental theorem of calculus.

$$\int_a^b dp\text{Binom}(k|N,p) = \binom{N}{k}\Big[\beta(b|k+1,N-k+1) - \beta(a|k+1,N-k+1)\Big] \tag{3}$$

Finally, we combine the above results, allowing us to compute the pairwise relationship likelihood when $M_{AB} \in \{ancestor, descendent, branched\}$ as a one-dimensional integral.

$$\begin{aligned}p(x_{As},x_{Bs}|M_{AB}) = \frac{2}{\omega_{Bs}}\binom{V_{Bs}+R_{Bs}}{V_{Bs}}\int_0^1 d\phi_{As}&\text{Binom}(V_{As}|V_{As}+R_{As},\omega_{As}\phi_{As})\Big[\\ &\beta(\omega_{Bs}U(\phi_{As},M_{AB})|V_{Bs}+1,R_{Bs}+1) \\ &-\beta(\omega_{Bs}L(\phi_{As},M_{AB})|V_{Bs}+1,R_{Bs}+1)\Big]\end{aligned} \tag{4}$$

To compute this numerically, we use the one-dimensional quadrature algorithm from `scipy.integrate.quad`.

6.1.5 Efficiently computing evidence for garbage and coincident pairwise relationships

We now examine how to compute the pairwise relationship likelihood for $M_{AB} = garbage$ using the general likelihood given in Eq. (1). First, observe that we are integrating over $\phi_{As} \in [0,1]$ and $\phi_{Bs} \in [0,1]$, meaning there is no constraint placed on ϕ_{Bs} by ϕ_{As}. By removing the dependence of ϕ_{Bs} on ϕ_{As}, the likelihood can be broken into the product of two one-dimensional integrals, each taken over the interval $[0,1]$. Then, by drawing on results Eq. (2) and Eq. (3), we can compute an analytic solution to each integral.

$$
\begin{aligned}
p(x_{As}, x_{Bs}|M_{AB} = \textit{garbage}) &= \iint d\phi_{As} d\phi_{Bs} p(x_{As}|\phi_{As}) p(x_{Bs}|\phi_{Bs}) p(\phi_{As}, \phi_{Bs}|M_{AB}) \\
&= \left[\int_0^1 d\phi_{As} p(x_{As}|\phi_{As})\right]\left[\int_0^1 d\phi_{Bs} p(x_{Bs}|\phi_{Bs})\right] \\
&= \frac{1}{\omega_{As}\omega_{Bs}} \beta(\omega_{As}|V_{As}+1, R_{As}+1)\beta(\omega_{Bs}|V_{Bs}+1, R_{Bs}+1) \qquad (5)
\end{aligned}
$$

Finally, we compute the likelihood for $M_{AB} = \textit{coincident}$. As our coincident constraint requires $\phi_{As} = \phi_{Bs}$, we are integrating along the diagonal line $\phi_{As} = \phi_{Bs}$ that cuts through the unit square formed by $\phi_{As} \in [0,1]$ and $\phi_{Bs} \in [0,1]$. This can be evaluated as a line integral along the curve $r(\phi) \equiv \langle \phi, \phi \rangle$ for $\phi \in [0,1]$, with the Euclidean norm $\|r'(\phi)\| = \sqrt{2}$.

$$
\begin{aligned}
p(x_{As}, x_{Bs}|M_{AB} = \textit{coincident}) &= \iint d\phi_{As} d\phi_{Bs} p(x_{As}|\phi_{As}) p(x_{Bs}|\phi_{Bs}) p(\phi_{As}, \phi_{Bs}|M_{AB}) \\
&= \int_0^1 d\phi p(x_{As}|\phi) p(x_{Bs}|\phi) \|r'(\phi)\| \\
&= \sqrt{2} \int_0^1 d\phi \text{Binom}(V_{As}|V_{As}+R_{As}, \omega_{As}\phi) \text{Binom}(V_{Bs}|V_{Bs}+R_{Bs}, \omega_{Bs}\phi)
\end{aligned} \qquad (6)
$$

As with the ancestral, descendent, and branched relationships, we use the one-dimensional quadrature algorithm from `scipy.integrate.quad` to compute this.

6.1.6 Computing the posterior probability for pairwise relationships

In Eq. (4), Eq. (5), and Eq. (6), we established how to compute the evidence for each of the five possible relations between mutation pairs, which takes the general form $p(x_A, x_B|M_{AB})$.. By combining these evidences with a prior probability $p(M_{AB})$ over relationships for mutation pair (A, B), we can compute the posterior probability $p(M_{AB}|x_A, x_B)$ of each relationship.

$$
p(M_{AB}|x_A, x_B) = \frac{p(x_A, x_B|M_{AB})p(M_{AB})}{\sum_{M'} p(x_A, x_B|M')p(M')} \qquad (7)
$$

As we discuss in Section 6.2.8, we assume that, when Pairtree is run, mutations have already been clustered into subpopulations and "garbage" mutations have already been discarded. Consequently, we

are computing pairwise relations between groups of mutations comprising subclones, and so we assign zero prior mass to the *coincident* and *garbage* relationships, ensuring these relationships also have zero posterior mass. The other three relationships are assigned the same prior probability, as we have no reason to believe one is more likely than the others.

$$p(M_{AB}) = \begin{cases} \frac{1}{3} & \text{if } M_{AB} \in \{ancestor, descendent, branched\} \\ 0 & \text{if } M_{AB} \in \{coincident, garbage\} \end{cases}$$

6.2 Performing tree search

6.2.1 Representing cancer evolutionary histories with trees

Most clone tree reconstruction algorithms group mutations into subclones, with mutations that share the same subclonal frequency across cancer samples placed together. While thousands of mutations are typically observed using whole-genome sequencing, the mutations can typically be grouped into a much smaller number of subclones, simplifying the cancer's evolutionary history. This grouping is valid because, as a cell population expands within a cancer, the frequencies of all mutations shared by cells in that population will increase in lockstep. Although Pairtree does not explicitly require that mutations be grouped into subclones, it can take these groupings as input. In this case, it replaces each mutation group with a single mutation, termed a *super-variant*, that represents the subclone.

When provided with K mutation clusters as input, each consisting of one or more mutations, Pairtree will produce a distribution over trees with $K + 1$ nodes. Node 0 corresponds to the non-cancerous cell lineage that gave rise to the cancer, while node $k \in \{1, 2, ..., K\}$ corresponds to the subclone associated with mutation cluster k. Node 0 always serves as the tree root, representing that the patient's cancer developed from non-cancerous cells, and thus has no assigned mutations and a subclonal frequency of $\phi_{0s} = 1$ in every cancer sample s.

An edge from node A to node B indicates that subclone B evolved from subclone A, acquiring the mutations associated with cluster B while also inheriting all mutations present in A and A's ancestral nodes. The children of node 0 are termed the *clonal cancer populations*. Typically, there is only one clonal cancer population, but the algorithm allows multiple such populations when the data imply them. Multiple clonal cancer populations indicate that multiple cancers developed independently in the patient, such that they shared no common cancerous ancestor.

An edge from node A to node B means that, at the resolution permitted by the data, we cannot discern

any intermediate cell subpopulations that occurred between these two evolutionary points. Nevertheless, such subpopulations may well have existed in the cancer.

6.2.2 Tree likelihood

To describe the tree likelihood, we develop the following notation:

- K: number of cancerous subpopulations (and mutation clusters), with individual populations indexed as $k \in \{1, 2, ..., K\}$
- S: number of cancer samples, with individual samples indexed as $s \in \{1, 2, ..., S\}$
- M_k: set of mutations associated with subclone k. Note this is distinct from the M_{AB} notation used in Section 6.1 to denote the pairwise relationship between mutations.
- V_{ms}: observed variant read count for mutation m in cancer sample s
- R_{ms}: observed reference read count for mutation m in cancer sample s
- ω_{ms}: probability of observing a variant read at mutation m's locus within a subclone possessing m, in cancer sample s
- ϕ_{ks}: subclonal frequency of subclone k in cancer sample s
- Φ: set of ϕ_{ks} values for all K and S

The data x consists of the set of all V_{ms}, R_{ms}, and ω_{ms} mutation values, as well as the M_k clustering of those mutations into subclones. Given the tree t, consisting of a tree structure and associated subclonal frequencies $\Phi = \{\phi_{ks}\}$, Pairtree uses the likelihood $p(x|t, \Phi)$ to score the tree. We describe how to compute the subclonal frequencies in Section 6.3. Below we use x_{ks} to represent all data in sample s for the mutations associated with subclone k, while x_{ms} refers to the data for an individual mutation m.

$$\begin{aligned} p(x|t,\Phi) &= \prod_{k\in\{1,2,...,K\}} \prod_{s\in\{1,2,...,S\}} p(x_{ks}|t,\Phi) \\ &= \prod_{k\in\{1,2,...,K\}} \prod_{m\in M_k} \prod_{s\in\{1,2,...,S\}} p(x_{ms}|\phi_{ks}) \\ &= \prod_{k\in\{1,2,...,K\}} \prod_{m\in M_k} \prod_{s\in\{1,2,...,S\}} \text{Binom}(V_{ms}|V_{ms}+R_{ms}, \omega_{ms}\phi_{ks}) \end{aligned} \tag{8}$$

The likelihood Eq. (8) demonstrates that tree structure is not explicitly considered in the tree likelihood. Instead, we assess tree likelihood by how well the observed mutation data are fit by the tree-constrained subclonal frequencies accompanying the tree. Typically, we obtain a tree's subclonal frequencies by making a maximum a posteriori (MAP) estimate, as described in Section 6.3.

Though Eq. (8) is ultimately the likelihood used by Pairtree for tree search, examining another perspective can help us understand what this likelihood represents. If we wished to directly assess the quality of a tree structure independent of its subclonal frequencies, thereby obtaining the likelihood $p(x|t)$ rather than $p(x|t,\Phi)$, we would integrate over the range of tree-constrained subclonal frequencies permitted by the tree structure.

$$
\begin{aligned}
p(x|t) &= \prod_{k\in\{1,2,\ldots,K\}} \prod_{s\in\{1,2,\ldots,S\}} p(x_{ks}|t) \\
&= \prod_{k\in\{1,2,\ldots,K\}} \prod_{m\in M_k} \prod_{s\in\{1,2,\ldots,S\}} p(x_{ms}|t) \\
&= \prod_{k\in\{1,2,\ldots,K\}} \prod_{m\in M_k} \prod_{s\in\{1,2,\ldots,S\}} \int d\Phi p(x_{ms},\Phi|t) \\
&= \prod_{k\in\{1,2,\ldots,K\}} \prod_{m\in M_k} \prod_{s\in\{1,2,\ldots,S\}} \int d\Phi p(x_{ms}|\phi_{ks}) p(\Phi|t) \qquad (9) \\
&\approx \prod_{k\in\{1,2,\ldots,K\}} \prod_{m\in M_k} \prod_{s\in\{1,2,\ldots,S\}} p(x_{ms}|\phi_{ks}) \qquad (10)
\end{aligned}
$$

In Eq. (9), the factor $p(\Phi|t)$ is an indicator function representing whether the set of subclonal frequencies Φ obeys the constraints imposed by the tree structure t:

1. All subclonal frequencies exist within the unit interval, such that $\phi_{ks} \in [0,1]$ for all k and s.
2. The non-cancerous node 0 is an ancestor of all subpopulations, such that $\phi_{ks} = 1$ for all k and s.
3. Let $C(k)$ represent the children of population k in the tree. The subclonal frequency for k must be at least as great as the sum of its childrens' frequencies, such that $\phi_{ks} \geq \sum_{c\in C(k)} \phi_{cs}$.

To obtain Eq. (9), we assume that only a narrow range of subclonal frequencies are permitted by the tree structure, and so we can use the MAP subclonal frequencies to approximate the integral and obtain Eq. (10), which is the likelihood function that Pairtree uses, as per Eq. (8). Consequently, we use Pairtree's likelihood $p(x|t,\Phi)$ of the tree t and subclonal frequencies Φ as an approximation of the marginal likelihood of the tree $p(x|t)$.

As an aside, note that a set of subclonal frequencies Φ obeying the three constraints enumerated above may be consistent with multiple tree structures—i.e., we may have $p(\Phi|t) \neq 0$ for a fixed Φ and different tree structures t. This shows how ambiguity may exist: a tree's subclonal frequencies may permit multiple possible tree structures, all of which would be assigned the same likelihood. Each cancer sample's subclonal frequencies typically impose additional constraints on possible tree structures, reducing this ambiguity.

6.2.3 Using Metropolis-Hastings to search for trees

Pairtree uses the Metropolis-Hastings algorithm [32], a Markov Chain Monte Carlo method, to search for trees that best fit the observed read count data x. For notational convenience, our references to a tree t should be understood to implicitly include a set of subclonal frequencies Φ that have been computed for t, such that the likelihood denoted $p(x|t)$ actually represents the likelihood $p(x|t, \Phi)$ described in Section 6.2.2.

According to the Metropolis-Hastings algorithm, to obtain samples from the posterior distribution over trees $p(t|x)$, we must modify an existing tree t to create a new proposal tree t'. The t' tree is accepted or rejected as a valid sample from the posterior according to how its likelihood $p(x|t')$ compares to the existing tree's $p(x|t)$, as well as the probabilities $p(t \rightarrow t')$ of transitioning from the t tree to the t' tree, and $p(t' \rightarrow t)$ of returning from t' to t. By Metropolis-Hastings, we assume that, given enough samples generated in this manner, we are eventually obtaining samples from the posterior distribution over trees $p(t|x) = \frac{p(x|t)p(t)}{p(x)} = \frac{p(x|t)p(t)}{\sum_{t'} p(x|t')p(t')}$. To establish our tree prior $p(t)$, we denote the number of possible tree topologies for K subclones as $T(K)$, which is a large but finite number that is exponential as a function of K [20]. Thus, we define our tree prior as a uniform distribution $p(t) \equiv \frac{1}{T(K)}$, as we have no reason to prefer one tree structure to another a priori. Consequently, in computing the posterior ratio $\frac{p(t'|x)}{p(t|x)}$ required for Metropolis-Hastings, all factors except the likelihoods $p(x|t)$ and $p(x|t')$ cancel.

Pairtree can run multiple MCMC chains in parallel, with each starting from a different initialization (Section 6.2.7). By default, Pairtree runs a total of C chains, with C set to the number of CPU cores present on the system by default, and $P = C$ executing in parallel. Both P and C can be customized by the user. From each chain, $S = 3000$ samples are drawn by default. The first $B \in [0, 1]$ proportion of trees are assumed to be early attempts by the sampling procedure to migrate toward high-probability regions of tree space, and so are discarded as burn-in samples because they are assumed to poorly reflect the true posterior. By default, we set $B = \frac{1}{3}$. To reduce correlation between successive samples, Pairtree supports thinning, by which only a fraction $T \in [0, 1]$ of non-burn-in samples are retained. By default,

Pairtree does not thin samples, so $T = 1$. Pairtree uses T to calculate a parameter $N = \text{round}(\frac{1}{T})$, such that the algorithm records every Nth sample. Thus, the actual number of trees recorded from a chain is $L = 1 + \lfloor \frac{S-1}{N} \rfloor$. Only after thinning the chain are the burn-in samples discarded, resulting in round(BL) trees being returned as posterior samples from the chain. The C, P, S, B, and T parameters can all be changed by the user.

Once all chains finish sampling, Pairtree combines their results to provide an estimate of the posterior tree distribution. Given the uniform tree prior $p(t)$, the posterior tree probability simplifies to $p(t|x) = \frac{p(x|t)}{\sum_{t'} p(x|t')}$. If the same tree t appears multiple times in this multiset—as it will, for instance, if proposal trees are rejected in Metropolis-Hastings and the last accepted tree is sampled again—each instance will appear as a separate term in the sum over t', reflecting that each is a distinct sample from the posterior estimate.

6.2.4 Modifying trees via tree proposals

To generate a new proposal tree t' from an existing tree t, Pairtree relies on tree updates similar to those established in [15, 33]. The algorithm modifies t by moving an entire sub-tree under a new parent, or by swapping the position of two nodes. Specifically, Pairtree generates a pair (A, B), where B denotes a tree node to be moved, and A represents its destination. This pair is subject to the constraints $\{A, B\} \subset \{0, 1, ..., K\}$, such that $A \neq B$, A is not the current parent of B, and B is not the root node 0. Two possible cases result. If A is a descendant of B, then the positions of A and B are swapped, without modifying any other tree nodes. This implies that the previous descendants of B (excluding A itself) become the descendants of A, while the previous descendants of A become the (only) descendants of B. Otherwise, A is not a descendant of B (i.e., A is an ancestor of B, or A is on a different tree branch), and so the sub-tree with B at its head is moved so that A becomes its parent. Observe that both moves can be reversed, which is a necessary condition for the Markov chain to satisfy detailed balance. In the first case, if A was descendent of B in t, then the pair (B, A) applied to the tree t' will restore t. In the second case, if A was not descendent of B in t, and B's parent in t was node P, then the pair (P, B) applied to tree t' will restore t.

Pairtree provides two means of choosing the pair (A, B). The first mode uses the pairs tensor to inform tree proposals (Section 6.2.5). The second mode proposes tree updates blindly without reference to the data (Section 6.2.6), and is helpful for escaping pathologies associated with the first mode. Pairtree randomly selects between these modes for each update (Section 6.2.6).

6.2.5 Using the pairs tensor to generate tree proposals

One of Pairtree's key contributions is to recognize that the pairs tensor provides an effective guide for tree search, conferring insight into what portions of an existing tree suffer from the most error, and how those portions should be modified to reduce error. To create the proposal (A, B) for modifying the tree t, as described in Section 6.2.3, Pairtree generates discrete probability distributions $W^{(A,B)}$ and $W^{(B)}$, corresponding to distributions over $0, 1, ..., K$ that are used to sample A and B, respectively. The choice of B depends only on the current tree state t, and so we denote the corresponding probability distribution as $W^{(B)}$. The choice of A, conversely, depends both on the current tree state t and whatever choice we made for B, and so we denote the corresponding probability distribution as $W^{(A,B)}$. Every $W^{(A,B)}$ and $W^{(B)}$ depends solely on the tree state, such that the Markov chain used for Metropolis-Hastings is time-invariant.

The algorithm generates the probability distribution $W^{(B)}$ such that the most probability mass is placed on elements corresponding to tree nodes with the highest pairwise error. First, observe that a tree induces a pairwise relationship between every pair of mutations—i.e., a tree places every mutation pair in a coincident, ancestral, descendent, or branched relationship. In Section 6.1, we described how to use mutation read counts to compute a probability distribution over these four relationships for every pair. For a given mutation B, we can thus compute the joint probability of the pairwise relationships between B and every other mutation induced by the tree t to determine how well-placed B is within the tree. Consider the mutation pair (k, B). If $p(M_{kB}|x_k, x_B)$ represents the probability of the pair taking pairwise relation M_{kB}, then the probability of the pair taking one of the three other possible relationships is $p(\neg M_{kB}|x_k, x_B) = 1 - p(M_{kB}|x_k, x_B)$, which we can think of as the pairwise relationship error. Then, the joint pairwise relationship error for all $K - 1$ pairs that include B is $E(B) \equiv p(\neg M_{1B}, \neg M_{2B}, ..., \neg M_{KB}|x_1, x_2, ..., x_B, ..., x_K) = \prod_{k \neq B} 1 - p(M_{kB}|x_k, x_B)$.

We compute the probability distribution $W^{(B)}$, whose elements represent the probability of selecting the node B to be moved within the tree, in accordance with the pairwise relationship error $E(B)$. To accomplish this, we treat $\log E(B)$ as the logarithms of elements in an unnormalized probability distribution. To normalize the tuple $(E(1), E(2), ..., E(K))$ to create a probability distribution, we use the scaled softmax function $\text{ssmax}(x) \equiv \text{softmax}(Sx)$, where the S scalar is chosen such that $\frac{\max(\text{ssmax}(x))}{\min(\text{ssmax}(x))} \leq R \equiv 100$. Specifically, the S scalar is set to 1 if $\frac{\max(\text{softmax}(x))}{\min(\text{softmax}(x))} \leq R$, or otherwise to whatever value greater than 1 is necessary to make $\frac{\max(\text{softmax}(Sx))}{\min(\text{softmax}(Sx))} = R$. The scaled softmax can be understood as a "softer softmax," ensuring no element in $W^{(B)} \equiv \text{ssmax}((\log E(1), \log E(2), ..., \log E(K)))$ has more than 100

times the probability mass of any other. In practice, this results in every tree node having a non-trivial probability of being selected for modification.

With the probability distribution $W^{(B)}$ established, we sample $B \sim W^{(B)}$. We now need to establish how to compute the probability distribution $W^{(A,B)}$, whose elements represent the probability of selecting the destination A for the node B. Critically, pairwise relations provide a computationally efficient means of evaluating hypothetical trees that modify B's position—we can, in fact, test every possible proposal for $A \in \{0, 1, ..., K\} - \{B, P_B\}$, where P_B denotes the current parent of B. With the choice of B already made, let $D_B(A) \equiv \prod_{(j,k)} p(M_{jk}|x_j, x_k)$ represent the joint probability of choosing A as the destination for B. By this formulation, (j, k) ranges over all $\binom{K}{2}$ pairs within the set $\{1, 2, ..., K\}$, and $D_B(A)$ represents the joint probability of all pairwise relations induced by the tree $t^{(A,B)}$, which results from making the modification to tree t denoted by (A, B). Similar to $W^{(B)}$, we apply the scaled softmax to the $\log D_B(A)$ elements to create $W^{(A,B)}$, with $W^{(A,B)} \equiv \text{ssmax}((\log D_B(1), \log D_B(2), ..., \log D_B(K)))$. We then sample $A \sim W^{(A,B)}$.

We now have a concrete realization of the (A, B) pair that we can apply to tree t, yielding a modified tree t'. By using the pairwise relations as a guide, we selected a node (or subtree) B to modify, whose selection probability was dictated by the pairwise errors induced by its position in the tree. Then, we selected a destination A, which we swapped with the node B if A was already a descendant of B, or otherwise made the parent of the B subtree. In choosing B, we considered only the joint pairwise error of the $K - 1$ pairs including B; however, in choosing A, we considered the pairwise probabilities of all $\binom{K}{2}$ pairs that would result from the modified tree. Considering all pairs is necessary because moving the whole subtree rooted by B changed the position of all B's descendants, potentially affecting many pairs that did not include A or B.

Thus, we selected a modification (A, B) to t that should, on average, yield a t' tree with less error in pairwise relations. Ultimately, however, the question of whether to accept t' as a posterior tree sample is decided by the Metropolis-Hastings decision rule that requires computing new subclonal frequencies Φ' for t', then considering the likelihood of the previous tree $p(x|t, \Phi)$ relative to the new likelihood $p(x|t', \Phi')$. Intuitively, once B is chosen, considering the change in pairwise relations induced by every possible choice of A captures substantial information about the quality of the tree that would be created by the (A, B) modification, while incurring only a modest computational cost. To fully evaluate the new tree t', we must, however, use the full likelihood, which captures more subtle information about higher-order relations beyond pairwise. Though this is a more reliable indicator of the new tree's quality, it requires the computationally expensive step of computing Φ', which is why Pairtree does not do this

when evaluating potential tree modification proposals.

6.2.6 Escaping local maxima in tree space by allowing uniformly sampled tree proposals

Sampling the (A, B) tree modifications solely using the pairs tensor sometimes results in Pairtree becoming stuck in local maxima that exist in the tree space whose likelihood is defined with respect to the pairs tensor, but that have low likelihood in the tree space defined by the tree likelihood. Consequently, the tree-proposal algorithm may repeatedly propose tree modifications that improve consistency with pairwise relationships while worsening the overall tree, leading to many successive proposals being rejected. That is, some tree nodes may have high pairwise error, such that they are often sampled as the B subtree to modify. These nodes may in addition have destinations A within the tree that substantially reduce this pairwise error, resulting in the (A, B) modification being sampled with high probability. When the tree t' induced by this modification is evaluated using the tree likelihood $p(x|t', \Phi')$, however, it may have poor likelihood, resulting in the modified tree being rejected by Metropolis-Hastings. This pathology occurs because t' may appear to be a good candidate when only pairwise relations are considered, but when higher-degree relationships, such as those between mutation triplets, are captured in the subclonal frequency-based likelihood $p(x|t', \Phi')$, the tree is revealed to be poor.

Were the tree proposals (A, B) generated solely using the pairwise relations, Pairtree would repeatedly propose the same modification only to have it rejected, resulting in the algorithm becoming stuck at a sub-optimal point in tree space. To overcome this, we added two decision points in the tree generation process that permit uniformly sampled modifications. Firstly, when sampling the node B to move within the tree, Pairtree will use the pairwise relation-informed choice only $\gamma = 70\%$ of the time. In the other $1 - \gamma = 30\%$ of cases, Pairtree will sample B from the discrete uniform distribution over $\{1, 2, ..., K\}$. Secondly, in $\zeta = 70\%$ of cases, Pairtree will sample the destination node A from the discrete uniform distribution over $\{0, 1, ..., K\} - \{B, P_B\}$, where P_B denotes the current parent of B. Both decisions are made independently and at random when generating the tree proposal, such that a proposal using pairwise relations for both A and B is generated for only $\gamma\zeta = 49\%$ of tree modifications. Conversely, $(1 - \gamma)(1 - \zeta) = 9\%$ of tree modifications are generated without considering the pairwise relations in any capacity. Both γ and ζ can be modified at runtime by the user. Their default values were chosen to ensure that approximately half of tree modification proposals are fully informed by pairwise relations, while the remaining half ignore the pairwise relations for at least part of the proposal generation, allowing the algorithm to explore regions of tree space that might otherwise be rendered difficult to reach.

6.2.7 Tree initialization

To sample trees via Metropolis-Hastings, the MCMC chain must be initialized with a tree structure. Similar to the tree-sampling process, which can generate proposals using the pairs tensor (described in Section 6.2.5) or without it (Section 6.2.6), the initialization algorithm can use the pairs tensor to infer reasonable relationships between subclones, or can ignore the pairs tensor and thereby avoid potential biases that would inhibit tree search.

We first describe tree initialization using the pairs tensor. In this mode, Pairtree constructs the tree in a top-down fashion, selecting subclones to add to the tree with a sampling probability based on which appear to have the most ancestral relationships relative to subclones not yet added. Once the algorithm determines which subclone to add, it considers all possible parents from amongst the nodes already added, sampling a choice based on which induces the least pairwise relation error for all subclones. This algorithm uses the scaled softmax described in Section 6.2.5.

```
function init_tree_from_pairs_tensor {
  # Only the root node exists in the tree initially.
  let added = {0}
  # Track which nodes we've added.
  let unadded = {1, 2, ..., K}
  # List of edges in tree. This starts as the empty set.
  # Each element consists of pair '(a, b)',
  # representing edge from 'a' to 'b'.
  let tree = {}

  while length(unadded) > 0:
    # Set 'c_weights' elements according to joint probability
    # that 'c' is ancestral to other unadded nodes.
    let c_weights = ∑_{k∈unadded∧k≠c} log p(M_ck = ancestral|x_c, x_k)
    let c_normed = ssmax(c_weights)
    # 'c_normed' is now a categorical probability distribution, so sample from it.
    let c_choice = sample(c_normed)

    # Now we have the node to place in the tree.
    # We must sample a parent for it from the nodes already
    # placed in the tree. 'p_weights' is a dictionary.
    let p_weights = {}
    for p in added:
      let candidate_tree = tree ∪ {(p, c)}
```

```
      # Let (a, b) range over ever pair of elements
      # generated from the set 'added ∪ c'.
      # We only consider every pair once -- i.e., only (a, b) and not (b, a).
      # We also do not consider the pairs (a, a).
      # M_ab is the relation this pair takes in 'candidate_tree'.
      p_weights[p] = ∑_{a,b} log p(M_ab|x_a, x_b)

    let p_normed = ssmax(p_weights)
    let p_choice = sample(p_normed)
    tree.add((p_choice, c_choice))
    unadded.remove(c_choice)
    added.add(c_choice)
  return tree
}
```

Listing 1: Tree initialization algorithm using pairs tensor.

In the second mode, Pairtree initializes a tree without reference to the pairwise relations, by placing every subclone as an immediate child of the root. This initialization is unbiased insofar as it imposes no ancestral or descendent relationships amongst subclones, assuming instead that the Metropolis-Hastings update scheme can rapidly alter this initial tree to produce a reasonable solution.

When initializing an MCMC chain, Pairtree selects between the two initialization modes at random, with probability $\iota = 70\%$ of selecting the pairwise-relation-based mode, and $1 - \iota = 30\%$ probability of the unbiased mode. The ι parameter can be specified by the user, with the default value chosen under the assumption that Pairtree will typically be used in multi-chain mode, such that different chains will benefit from different initializations that allow the algorithm to more fully explore tree space.

6.2.8 Reducing Pairtree's computational burden using supervariants

Pairtree assumes that mutations have been clustered into subpopulations, with "garbage" variants discarded, before the tree-construction algorithm begins. As a result, all mutations within a subpopulation are rendered *coincident* relative to one another. Mutations within a subclone also share the same evolutionary relationships to all mutations outside the subclone. Thus, to reduce the computational burden imposed by the method, rather than working with individual mutations, we can instead represent each subpopulation with a single *supervariant*, then compute pairwise relations between these rather than their constituent mutations.

Conceptually, relative to the individual mutations that compose it, a supervariant should provide a

more precise estimate of the subclonal frequency of its corresponding subclone. Specifically, a mutation m in a cancer sample s has V_{ms} variant reads and R_{ms} reference reads, yielding total reads $T_{ms} \equiv V_{ms} + R_{ms}$ and a VAF $\equiv \frac{V_{ms}}{T_{ms}}$. Given a probability of observing the variant allele ω_{ms}, we conclude that $\omega_{ms} T_{ms}$ reads originated from the variant allele, and so we can estimate the corresponding subclone's subclonal frequency by $\hat{\phi}_{ms} \equiv \frac{V_{ms}}{\omega_{ms} T_{ms}}$. Each mutation's $\hat{\phi}_{ms}$ should thus serve as a noisy estimate of its subclone's true ϕ_{ms}.

Let x_{ms} represent the data associated with mutation m in sample s, such that $x_{ms} \equiv \{V_{ms}, R_{ms}, \omega_{ms}\}$. Under a binomial observation model (Section 6.2.2), given subclonal frequency ϕ_{ks} for the subclone k harboring mutation m in sample s, we have the mutation likelihood $p(x_{ms}|\phi_{ks}) \equiv \text{Binom}(V_{ms}|V_{ms} + R_{ms}, \omega_{ms}\phi_{ks})$. Let M_k be the set of mutations associated with subclone k. Then, from all $j \in M_k$, we get the following joint likelihood for cancer sample s:

$$\begin{aligned} p(M_k|\phi_{ks}) &= \prod_{j \in M_k} p(x_{js}|\phi_{ks}) \\ &= \prod_{j \in M_k} \text{Binom}(V_{js}|V_{js} + R_{js}, \omega_{js}\phi_{ks}) \end{aligned}$$

Assuming ω_{js} takes the same value ω_{ks} for all $j \in M_k$, the joint likelihood takes the following form:

$$\begin{aligned} p(M_k|\phi_{ks}) &= \prod_{j \in M_k} \binom{V_{js} + R_{js}}{V_{js}} (\omega_{js}\phi_{ks})^{V_{js}} (1 - \omega_{js}\phi_{ks})^{R_{js}} \\ &= \Big[\prod_{j \in M_k} \binom{V_{js} + R_{js}}{V_{js}} \Big] (\omega_{js}\phi_{ks})^{\sum_j V_{js}} (1 - \omega_{js}\phi_{ks})^{\sum_j R_{js}} \end{aligned} \tag{11}$$

We want the likelihood for the supervariant k representing the variants in M_k to take the same functional form. Thus, we set $V_{ks} \equiv \sum_{j \in M_k} V_j s$ and $R_{ks} \equiv \sum_{j \in M_k} R_{js}$, yielding the following supervariant likelihood.

$$\begin{aligned} p(x_{ks}|\phi_{ks}) &= \binom{V_{ks} + R_{ks}}{V_{ks}} (\omega_{ks}\phi_{ks})^{V_{ks}} (1 - \omega_{ks}\phi_{ks})^{R_{ks}} \\ &= \binom{\sum_j V_{js} + R_{js}}{\sum_j V_{js}} (\omega_{ks}\phi_{ks})^{\sum_j V_{js}} (1 - \omega_{ks}\phi_{ks})^{\sum_j R_{js}} \end{aligned} \tag{12}$$

Observe that Eq. (12) takes the same functional form as Eq. (11), such that they differ only by a

constant of proportionality C that does not depend on ϕ_{ks}.

$$
\begin{aligned}
C &= \frac{p(M_k|\phi_{ks})}{p(x_{ks}|\phi_{ks})} \\
&= \frac{\prod_{j \in M_k} \binom{V_{js}+R_{js}}{V_{js}}}{\binom{\sum_j V_{js}+R_{js}}{\sum_j V_{js}}} \\
\therefore p(x_{ks}|\phi_{ks}) &\propto p(M_k|\phi_{ks}) = \prod_{j \in M_k} p(x_{js}|\phi_{ks})
\end{aligned} \tag{13}
$$

Consequently, the supervariant's likelihood accurately reflects the joint likelihood of the subclone's constituent variants, while reducing the algorithm's computational burden. In practice, the constant factor C by which the two differ does not matter, as the Metropolis-Hastings scheme (Section 6.2.3) that uses the likelihood (Section 6.2.2) requires only the ratio of two likelihoods to navigate tree space, such that C cancels.

Of course, Eq. (13) holds only when $\omega_{ks} = \omega_{js}$ for all $j \in M_k$. Most often, we are given diploid variants with $\omega_{js} = \frac{1}{2}$, and so we fix $\omega_{ks} = \frac{1}{2}$ for all supervariants. Thus, supervariants are assured to accurately represent their constituent variants when those variants are from diploid genomic regions. For non-diploid variants with $\omega_{js} \neq \frac{1}{2}$, we must rescale the provided data x_{js} to use a fixed $\omega_{ks} = \frac{1}{2}$, allowing us to use an approximation of the correct likelihood. To achieve this, we establish the following:

$$
\begin{aligned}
\hat{x}_{js} &= \{\hat{V}_{js}, \hat{R}_{js}, \hat{\omega}_{js}\} \\
T_{js} &= V_{js} + R_{js} \\
\hat{T}_{js} &= 2\omega_{js} T_{js} \\
\hat{V}_{js} &= \min(V_{js}, \hat{T}_{js}) \\
\hat{R}_{js} &= \hat{T}_{js} - \hat{V}_{js} \\
\hat{\omega}_{js} &= \frac{1}{2}
\end{aligned}
$$

This representation ensures the corrected variant read count $\hat{V}_{js}$ cannot exceed the corrected total read count $\hat{T}_{js}$, which could otherwise occur because of binomial sampling noise inherent to the genomic sequencing process, or an erroneous ω_{js} value that does not correctly reflect a copy number change. Note that both $\hat{T}_{js}$ and $\hat{V}_{js}$ can take non-integer values. If the original $\omega_{js} = \frac{1}{2}$, then the corrected read counts

are unchanged from their original values. From this point, for all mutations $j \in M_k$ associated with subclone k, we compute corrected supervariant read counts $\hat{V}_{ks}$ and $\hat{R}_{ks}$:

$$\begin{aligned}
\hat{x}_{ks} &= \{\hat{V}_{ks}, \hat{R}_{ks}, \hat{\omega}_{ks}\} \\
\hat{V}_{ks} &= \text{round}(\sum_j \hat{V}_{js}) \\
\hat{T}_{ks} &= \text{round}(\sum_j \hat{T}_{js}) \\
\hat{R}_{ks} &= \hat{T}_{ks} - \hat{V}_{ks} \\
\hat{\omega}_{ks} &= \frac{1}{2}
\end{aligned}$$

Based on Eq. (13), if the mutations $j \in M_k$ all had $\omega_{js} = \frac{1}{2}$, the ϕ_{ks} value we obtain in maximizing the supervariant likelihood $p(\hat{x}_{ks}|\phi_{ks})$ is also optimal for the full joint likelihood over the individual mutations $p(M_k|\phi_{ks}) = \prod_{j \in M_k} p(x_{js}|\phi_{ks})$, since the two likelihoods differ only by a constant of proportionality. If some mutations j had $\omega_{js} \neq \frac{1}{2}$, the supervariant likelihood $p(\hat{x}_{ks}|\phi_{ks})$ approximates the full joint likelihood, and so the obtained ϕ_{ks} value is only approximately optimal for the latter. To overcome this, Pairtree's implementation of the rprop optimization algorithm could be easily modified to optimize ϕ_{ks} with respect to the individual variants j, each with its own ω_{js}, rather than the combined supervariant representation that requires a single ω_{ks}. Equivalently, rprop could use multiple supervariants per subclone, with a single supervariant representing all constituent mutations possessing the same ω_{js}. The projection algorithm, however, necessitates using a single supervariant, which in turn requires a single ω_{ks}. Though the adjusted supervariant read counts yield only an approximation of the likelihood for non-diploid mutations, this is not a critical flaw, as projection is already computing a Gaussian approximation of the likelihood, rather than the exact binomial likelihood used by rprop.

6.3 Fitting subclonal frequencies to trees

Pairtree provides two algorithms for computing subclonal frequencies for a tree structure. Both attempt to maximize the data likelihood (Section 6.2.2), fitting the observed read count data as well as possible while fulfilling all constraints imposed by the tree structure. The first algorithm, named *rprop*, is based on gradient descent (Section 6.3.2), and directly maximizes the tree's binomial likelihood. The second algorithm, named *projection*, uses techniques from convex optimization to compute subclonal frequencies

maximizing the likelihood of a Gaussian approximation to the binomial [34]. While rprop typically produces higher-likelihood subclonal frequencies than projection, particularly for subclones where the Gaussian approximation to the binomial is poor, the projection algorithm runs substantially faster with many subclones (e.g., for 30 subclones or more). By default, Pairtree uses the projection algorithm, but the user can select rprop at runtime.

6.3.1 Converting between subclonal frequencies and subpopulation frequencies

To permit a more convenient representation, both rprop and projection work with subpopulation frequencies $H = \{\eta_{ks}\}$, rather than the subclonal frequencies $\Phi = \{\phi_{ks}\}$, where k and s are indices over subclones and cancer samples, respectively. Given a tree structure t, we can readily convert from one representation to the other. Let $D(k)$ represent the set of descendants of subclone k in tree structure t, and $C(k)$ represent the set of direct children of subclone k. Then, in cancer sample s, we have

$$\phi_{ks} = \eta_{ks} + \sum_{j \in D(k)} \eta_{js} \ .$$

Equivalently, we obtain

$$\eta_{ks} = \phi_{ks} - \sum_{j \in C(k)} \phi_{js} \ .$$

.

From the subclonal frequency constraints described in Section 6.2.2, we see that, because the root node takes $\phi_{0s} = 1$, we must have the constraint

$$\sum_{j=0}^{K} \eta_{js} = 1$$

across all K subclones, and that each individual $\eta_{js} \in [0, 1]$. As each cancer sample s is independent from every other, both rprop and projection optimize the set $\{\eta_{ks}\}$ separately for each fixed s.

6.3.2 Fitting subclonal frequencies using rprop

The rprop algorithm is a simpler version of RMSprop [40, 41], intended for use with full data batches rather than mini-batches. To perform unconstrained optimization on the parameters $H_s = \{\eta_{ks}\}$ for a fixed cancer sample s, the algorithm first reparameterizes to $H_s = \text{softmax}(\{\psi_{ks}\})$, so that we need not enforce constraints on $\{\psi_{ks}\}$ but can ensure $H_s \subset [0, 1]$ and $\sum_k \eta_{ks} = 1$.

On each iteration, given a tree structure t and existing subclonal frequencies Φ, rprop converts Φ to population frequencies H, then computes the derivatives

$$\frac{\partial p(x|t, \Phi)}{\partial \psi_{ks}} = \frac{\partial p(x|t, \Phi)}{\partial \eta_{js}} \frac{\partial \eta_{js}}{\partial \psi_{ks}}$$

for all subclone combinations j and k, using the tree likelihood (Section 6.2.2). The algorithm then uses the sign of the gradient to update the ψ_{ks} values, ignoring the gradient's magnitude. For each value of k, rprop maintains a step-size parameter λ_k, which is limited to lie within the interval $[10^{-6}, 50]$, preventing excessively small or large step sizes. The algorithm also maintains a step-size multiplier M_{ki} for subclone k on iteration i, with $M_{ki} = 1.2$ if $\text{sign}(\frac{\partial p(x|t,\Phi)}{\partial \psi_{ks}})$ agrees with the sign from the previous iteration $i - 1$, and $M_{ki} = \frac{1}{2}$ otherwise. Using these values, rprop performs the gradient update

$$\begin{aligned}
\lambda_k &:= M_{ki} \lambda_k \\
\lambda_k &:= \min(\lambda_k, 50) \\
\lambda_k &:= \max(\lambda_k, 10^{-6}) \\
\psi_{ks} &:= \psi_{ks} + \lambda_k \text{sign}(\frac{\partial p(x|t, \Phi)}{\partial \psi_{ks}})
\end{aligned}$$

The rprop algorithm continues this process until none of the $\frac{\partial p(x|t,\Phi)}{\partial \psi_{ks}}$ values exceed 10^{-5} in a particular iteration, or until $I = 10000$ iterations elapse, with I being customizable by the user.

To initialize the $H_s = \{\eta_{ks}\}$ values, we generate initial values $\hat{\eta}_{ks}$ with the following algorithm. $C(k)$ represents the set of direct children of k in the tree.

$$\hat{\phi_{ks}} := \frac{V_{ks}}{\omega_{ks}(V_{ks} + R_{ks}})$$

$$\hat{\phi_{ks}} := \min(1, \hat{\phi}_{ks})$$

$$\hat{\phi_{ks}} := \max(0, \hat{\phi}_{ks})$$

$$\hat{\eta}_{ks} := \hat{\phi}_{ks} - \sum_{j \in C(k)} \hat{\phi}_{ks}$$

$$\hat{\eta}_{ks} := \min(1, \hat{\eta}_{ks})$$

$$\hat{\eta}_{ks} := \max(0, \hat{\eta}_{ks})$$

Observe that the constraint $\hat{\eta}_{ks} \in [0, 1]$ is satisfied. To ensure $\sum_j \hat{\eta}_{js} = 1$, we finally set $\hat{\eta}_{ks} := \frac{\hat{\eta}_{ks}}{\sum_j \hat{\eta}_{js}}$. This initialization reflects that, if the provided tree structure t is consistent with the data and there is minimal noise in the data, the $\hat{\phi}_{ks} = \frac{V_{ks}}{\omega_{ks}(V_{ks}+R_{ks}})$ subclonal frequencies should be close to the maximum likelihood estimate for Φ in $p(x|t, \Phi)$.

6.3.3 Fitting subclonal frequencies using projection

The projection algorithm draws on the approach provided in [34]. The authors describe a method to efficiently enumerate mutation trees, in which individual nodes correspond to genomic mutations. To make this enumeration feasible, they developed an algorithm to rapidly compute tree-constrained subclonal frequencies. Using our supervariant representation, we can apply their approach to computing subclonal frequencies for clone trees by representing our binomial likelihood with a Gaussian approximation. First, we review the authors' notation and map it to the equivalent notation in Pairtree.

- : q: number of mutations, equivalent to our number of subclones K
- : p: number of cancer samples, equivalent to our S
- : $F \in \mathbb{R}^{q \times p}$: equivalent to our subclonal frequencies Φ, with F_{vs} equivalent to our ϕ_{ks}
- : $U \in \{0, 1\}^{q \times q}$: ancestry matrix created from tree structure t, such that $U_{j,k} = 1$ iff subclone j is an ancestor of subclone k or $j = k$
- : $M \in \mathbb{R}^{q \times p}$: equivalent to our population frequencies $H = \{\eta_{ks}\}$, with M_{vs} equivalent to our η_{ks}

With $\mathcal{U}$ representing the set of all ancestral matrices consistent with the perfect phylogeny problem (Section 10.8), the authors solve the optimization problem $\min_{U \in \mathcal{U}} \mathcal{C}(U)$, such that

$$\mathcal{C}(U) = \min_{M, F \in \mathbb{R}^{q \times p}} \|\hat{F} - F\| \text{ subject to } F = UM, M \geq 0, M^{\intercal}\mathbf{1} = \mathbf{1} \ .$$

Here, $\| \cdot \|$ is the Frobenius norm, and $\hat{F} \in \mathbb{R}^{q \times p}$ is the noisy estimate of the subclonal frequencies obtained from the data. Observe there is a one-to-one correspondence between U and t, as changing the structure of t will necessarily change ancestral relations described in U, and vice versa. Thus, the authors attempt to find the optimal ancestry matrix U, corresponding to an optimal tree t, that allows tree-constrained subclonal frequencies F best matching the noisy subclonal frequencies $\hat{F}$ observed in the data. Ultimately, the authors solve this problem through enumeration. While this scales better than previous enumerative approaches because of the authors' efficient computation of the optimal M for a given ancestry matrix U, the approach is still rendered infeasible for the large trees that Pairtree works with using a search-based method.

Useful for Pairtree is the authors' extremely efficient means of projecting the observed frequencies $\hat{F}$ on to the space of valid perfect-phylogeny models using Moreau's decomposition for proximal operators and a tree reduction scheme [34]. We utilize this to quickly compute subclonal frequencies Φ for a given tree t that corresponds to an ancestry matrix U. To allow us to use a Gaussian estimate of our binomial likelihood, the authors developed an extended version of their algorithm [42], in which they additionally take as input a scaling matrix $D \in \mathbb{R}^{q \times p}$ with all $D_{ks} > 0$. Using the element-wise multiplication operator $\odot$, the modified algorithm computes

$$\mathcal{C}'(U) = \min_{M, F \in \mathbb{R}^{q \times p}} \|D \odot \hat{F} - D \odot F\| \text{ subject to } F = UM, M \geq 0, M^{\intercal}\mathbf{1} = \mathbf{1} \ . \tag{14}$$

We will refer to the algorithm as the "projection optimization algorithm," and to Eq. (14) as the "projection objective." We now show how to use the projection objective to compute the MAP for a Gaussian approximation of our original binomial likelihood. First, observe that our goal is to maximize the binomial likelihood defined in Section 6.2.2 by finding optimal subclonal frequencies Φ for a given tree t. Thus, we wish to find

$$\max_{\Phi_s=\{\phi_{ks}\}} p(x_s|t,\Phi_s) = \max_{\Phi_s} p(V_{1s},V_{2s},...|t,T_{1s},T_{2s},\omega_{1s},\omega_{2s},\Phi) \text{ subject to } p(\Phi_s|t) \neq 0 \ . \tag{15}$$

Here, t represents the provided tree structure, while Φ_s refers to a set of scalar ϕ_{ks} values that obey the tree constraints described in Section 6.2.2, with $p(\Phi_s|t) \neq 0$ indicating that the set obeys the constraints. The s index represents the cancer sample, with each sample optimized independently. Our data x_s consists of, for subclone k, a count of variant reads V_{ks} and reference reads R_{ks}, yielding total reads $T_{ks} = V_{ks} + R_{ks}$. We define this as a binomial likelihood, in which we are optimizing the ϕ_{ks} values.

$$p(V_{1s},V_{2s},...|t,T_{1s},T_{2s},\omega_{1s},\omega_{2s},\Phi) = \prod_k p(V_{ks}|T_{ks},\omega_{ks},\phi_{ks}) \tag{16}$$

To approximate this using the Gaussian, we perform the following operations.

$$\prod_k p(V_{ks}|T_{ks},\omega_{ks},\phi_{ks}) = \prod_k \text{Binom}(V_{ks}|T_{ks},\omega_{ks}\phi_{ks}) \tag{17}$$

$$\approx \prod_k \mathbb{N}(V_{ks}|T_{ks}\omega_{ks}\phi_{ks}, T_{ks}\omega_{ks}\phi_{ks}(1-\omega_{ks}\phi_{ks})) \tag{18}$$

$$\propto \prod_k \mathbb{N}(\frac{V_{ks}}{\omega_{ks}T_{ks}} \approx \hat{\phi}_{ks}|\phi_{ks}, \frac{\phi_{ks}}{\omega_{ks}T_{ks}}(1-\omega_{ks}\phi_{ks})) \tag{19}$$

$$\approx \prod_k \mathbb{N}(\phi_{ks}|\hat{\phi}_{ks}, \frac{\hat{\phi}_{ks}}{\omega_{ks}T_{ks}}(1-\omega_{ks}\hat{\phi}_{ks})) \tag{20}$$

We relied on the following operations to achieve the above:

- Eq. (17) defined Eq. (16) with respect to the binomial distribution.
- Eq. (18) approximated Eq. (17) with the Gaussian distribution. We represent the Gaussian PDF for a random variable x drawn from a Gaussian with mean μ and variance σ^2 as $\mathbb{N}(x|\mu,\sigma^2)$.
- Eq. (19) divided the Gaussian random variable by the scalar $\omega_{ks}T_{ks}$, yielding another Gaussian proportional to the preceding. The new Gaussian random variable is $\frac{V_{ks}}{\omega_{ks}T_{ks}} \approx \hat{\phi}_{ks}$, our MAP of the subclonal frequency ϕ_{ks} for $\text{Binom}(V_{ks}|T_{ks},\omega_{ks}\phi_{ks})$. As $\phi_{ks} \in [0,1]$, we set $\hat{\phi}_{ks} \equiv \min(1, \frac{V_{ks}}{\omega_{ks}T_{ks}})$.
- To achieve a distribution over the unknown ϕ_{ks}, Eq. (20) swaps the Gaussian's random variable $\hat{\phi}_{ks}$ and mean ϕ_{ks}, yielding the same Gaussian PDF. Additionally, it approximates the variance of

the Gaussian in *Eq.* (19) by replacing ϕ_{ks} with its MAP in the variance definition.

Let the variance of each Gaussian be represented with $\sigma_{ks}^2 = \max(10^{-4}, \frac{\hat{\phi}_{ks}}{\omega_{ks} T_{ks}}(1 - \omega_{ks}\hat{\phi}_{ks}))$. We set a minimum variance of 10^{-4} to prevent our ϕ_{ks} estimates from being too precise to permit effective optimization. To transform Eq. (20) into the form required by the projection objective Eq. (14), observe

$$\prod_k \mathbb{N}(\phi_{ks}|\hat{\phi}_{ks}, \sigma_{ks}^2) \propto \exp - \sum_k \frac{(\phi_{ks} - \hat{\phi}_{ks})^2}{\sigma_{ks}^2} \ . \tag{21}$$

Thus, maximizing Eq. (21) is equivalent to optimizing the objective

$$\min_{\Phi_s} \exp \sum_k \frac{(\phi_{ks} - \hat{\phi}_{ks})^2}{\sigma_{ks}^2} \ . \tag{22}$$

As both $\exp x$ and $\sqrt{x}$ are monotonic functions, this is equivalent in turn to

$$\min_{\Phi_s} \sqrt{\sum_k \frac{(\phi_{ks} - \hat{\phi}_{ks})^2}{\sigma_{ks}^2}} \ . \tag{23}$$

To complete the transformation of Eq. (23) to the projection objective Eq. (14), we establish the following notation.

$$\begin{aligned} D_s &= [D_{1s}, D_{2s}, ..., D_{Ks}] = [\frac{1}{\sigma_{1s}}, \frac{1}{\sigma_{2s}}, ..., \frac{1}{\sigma_{Ks}}] \\ F_s &= [\phi_{1s}, \phi_{2s}, ..., \phi_{Ks}] \\ \hat{F}_s &= [\hat{\phi}_{1s}, \hat{\phi}_{2s}, ..., \hat{\phi}_{Ks}] \\ U &= \text{ ancestry matrix corresponding to tree } t \end{aligned}$$

Now, Eq. (23) can be rewritten using the Frobenius norm:

$$\min_{\Phi_s} \sqrt{\sum_k \frac{(\phi_{ks} - \hat{\phi}_{ks})^2}{\sigma_{ks}^2}} = \min_{M_s, F_s} \|D_s \odot (F_s - \hat{F}_s)\|$$
$$= \min_{M_s, F_s} \|D_s \odot (UM_s) - D_s \odot \hat{F}_s\| \ ,$$

Thus, we can now call the projection optimization algorithm to compute F_s and M_s, which are K-length vectors representing tree-constrained subclonal frequencies and subpopulation frequencies, respectively. Both obey the constraints inherent to the tree t that are expressed through the ancestry matrix U. The F_s values are the MAP under the Gaussian approximation Eq. (20) of binomial likelihood Eq. (17), ultimately achieving a near-optimal solution to the original optimization objective Eq. (15).

6.4 Creating simulated data

6.4.1 Parameters for simulating data

We first define parameters characterizing the different simulated cancers.

- K: number of subpopulations
- S: number of cancer samples
- M: number of variants
- T: number of total reads per variant

We created simulated datasets with the following parameter combinations.

Parameter	Values
K	$3, 10, 30, 100$
S	$1, 3, 10, 30, 100$
Mutations per cluster	$10, 20, 100$
T	$50, 200, 1000$

Table 1: Simulated data parameters. All combinations of these parameter values were used to generate simulated data, excepting cases when $K \in \{30, 100\}$ and $S \in \{1, 3\}$. This provided 144 parameter combinations, with four datasets generated from each, yielding 576 simulated datasets.

Observe there are $4 \times 5 \times 3 \times 3 = 180$ parameter combinations. When $K \in \{30, 100\}$, we did not simulate datasets with $S \in \{1, 3\}$ samples, as trees with so many subpopulations and so few cancer samples are

unrealistic—to resolve a large number of distinct mutation clusters, a large number of cancer samples is typically needed. Simulated datasets with $K \geq 30$ and $S < 10$ would thus correspond to complex trees with few cancer samples, posing a highly underconstrained computational problem that would not reflect how methods perform on realistic datasets. Thus, as there are $2 \times 2 \times 3 \times 3 = 36$ parameter sets yielding under-constrained simulations, we used the remaining $180 - 36 = 144$ sets to generate simulations. For each valid parameter set, we generated four distinct datasets, yielding $144 \times 4 = 576$ simulated datasets.

Above, rather than setting the number of mutations per dataset M directly, we instead specified the average number of mutations per cluster. This reflects that, because each subclone is distinguished by one or more unique mutations, trees with more subclones should have more mutations. Consequently, the number of mutations generated per dataset was $M = K(\text{mutations per cluster})$. Nevertheless, as described in Section 6.4.2, mutations are assigned to subclones in a non-uniform probabilistic fashion, such that the number of mutations in each subclone is only rarely equal to the parameter value for number of mutations per cluster used when generating the data.

6.4.2 Algorithm to generate simulated data

We generated simulated data using the following algorithm. Python code implementing this algorithm is available at `https://github.com/morrislab/pearsim`.

1. Generate the tree structure. For each subclone k, with $k \in \{1, 2, \ldots, K-1\}$, sample a parent $\mathcal{P}(k)$. We extended the previous subpopulation (i.e., $\mathcal{P}(k) = k - 1$) with probability $\mu = 0.75$, and otherwise sample $\mathcal{P}(k)$ from the discrete Uniform$(0, k-1)$ distribution. This extension probability created "linear chains" of successive subpopulations, with each member of the chain taking only a single child, interrupted sporadically by the creation of new tree branches. As the normal tree root, denoted as node 0, exists at the outset, node 1 will always take it as a parent. Note that this scheme allows for the creation of "polyprimary" trees, in which the root 0 takes multiple clonal cancerous children. Such polyprimary cases are created for approximately $1 - \mu = 0.25$ of datasets.

2. Generate the subpopulation frequencies η_{ks} for each subpopulation k in each cancer sample s, with $s \in 1, 2, \ldots, S$. These values were sampled separately for each s, with $[\eta_{0s}, \eta_{1s}, \ldots, \eta_{Ks}] \sim$ Dirichlet$(\alpha, \ldots, \alpha)$ = Dirichlet$(0.1, \ldots, 0.1)$. We use the symmetric Dirichlet distribution with a single α parameter because we have no reason to desire that any population frequency tend to be greater or less than others a priori. The choice of α has important implications for the structure of the simulated data (Section 10.7). As the η vector is drawn from the Dirichlet, we have $\sum_{k=0}^{K} \eta_{ks} = 1$

for each sample s.

3. Compute the subclonal frequencies ϕ_{ks} for each subclone k in each cancer sample s using the tree structure and η_{ks} values. Let $D(k)$ represent the set of k's descendants in the tree. Then, we have

$$\phi_{ks} = \eta_{ks} + \sum_{d \in D(k)} \eta_{ds} \ .$$

4. Assign the M variants to subclones. To ensure every subclones has at least one variant, set the subclones of the first K SNVs to $1, 2, \ldots, K$. To assign the remaining $M - K$ SNVs, sample subclone weights from the K-dimensional Dirichlet$(1, 1, \ldots, 1)$, then sample assignments from the K-dimensional categorical distribution using these weights.

5. Sample read counts for the variants. Let $A(m) \in \{1, 2, \ldots, K\}$ represent the subclone to which variant m was assigned. Let $\omega_{ms} = \frac{1}{2}$ represent the probability of observing a variant read when sampling reads from the variant's locus, for all subpopulations contained within m's subclone, reflecting a diploid variant not subject to any CNAs. Then, for each cancer sample s, given the fixed total read count T used for all variants in a dataset, we sample the number of variant reads $V_{ms} \sim \text{Binomial}(T, \omega_{ms}\phi_{A(m),s})$.

6.5 Evaluation metrics for method comparisons

6.5.1 Intuitive explanation of metrics

We developed two metrics for evaluating clone-tree reconstruction algorithms that are suitable for use with multiple cancer samples. The first, termed *VAF reconstruction loss* (henceforth "VAF loss"), measures how well a tree's subclonal frequencies match the allele frequency for each mutation implied by its CNA-corrected VAF. Each tree structure permits a range of subclonal frequencies, with the best subclonal frequencies matching the data as well as possible while also satisfying the tree constraints. Thus, the VAF loss evaluates a tree by determining how closely its subclonal frequencies match the observed data. VAF loss is reported in bits per mutation per cancer sample, representing the number of bits required to describe the data using the tree, normalized to the number of mutations and cancer samples. Lower values reflect better trees. As LICHeE could not compute subclonal frequencies itself, producing only tree structures, we used Pairtree to compute the MAP subclonal frequencies for its trees.

All evaluated methods report multiple solutions for each dataset, scored by a method-specific likelihood or error measure. To determine a single VAF loss for each method on each dataset, we used the method-

specific solution scores to compute the expectation over VAF loss (equivalent to the weighted-mean VAF loss). VAF loss is always reported relative to a baseline. For simulated data, the baseline is the VAF loss achieved using the true subclonal frequencies that generated the data. For real data, the baseline is expert-constructed, manually-built trees that were subjected to extensive refinement, with Pairtree used to compute the MAP subclonal frequencies. Thus, VAF loss indicates the average extra number of bits necessary to describe the data using a method's solutions rather than the baseline solution. Methods can find solutions that fit the data better than the baseline, yielding a negative VAF loss.

The second evaluation metric we developed, termed *relationship reconstruction error* (henceforth "relationship error"), recognizes that a clone tree defines pairwise relations between its constituent mutations, placing every pair in one of the four relationships discussed earlier. Using the set of trees reported by a method for a given dataset, we computed the empirical categorical distributions over pairwise mutation relations, with each tree's relationships weighted by the likelihood or error measure reported by the method. We then compared these distributions to the distributions imposed by all tree structures permitted by the true subclonal frequencies, computing the Jensen-Shannon divergence (JSD) between distributions for each pair. This yields a relationship error ranging between 0 bits and 1 bit. Using these, we report the joint JSD across all mutation pairs to summarize the quality of the solution set, normalized to the number of pairs. Thus, the relationship error for a given dataset ranges between 0 bits and 1 bit, with smaller values indicating that a method better recovered the full set of trees consistent with the data. We did not apply this metric to real data, whose true subclonal frequencies, and thus true possible tree structures, are unavailable.

6.5.2 VAF reconstruction loss

The VAF reconstruction loss represents how closely the subclonal frequencies associated with a method's clone tree solution set match the simulated data's VAFs (Section 3.4). The constraints imposed by good solution trees should permit subclonal frequencies that closely match the data. In Section 6.2.2, we described the tree likelihood Eq. (8), which we also use to define the VAF reconstruction loss.

Assume the method provides a distribution over different clone trees t, with the posterior probability of t represented as $p(t)$, such that $\sum_t p(t) = 1$. The loss is defined for each tree t over the mutation read count data x, with mutations m and cancer samples s. We use ϕ_{ms} to indicate the subclonal frequency in t for sample s associated with the subpopulation containing mutation m. For mutation m in sample s, we define the likelihood

$$\begin{aligned} p(x_{ms}) &= \sum_t p(x_{ms}|t)p(t) \\ &= \sum_t p(x_{ms}|\phi_{ms})p(t) \\ &= \sum_t p(t)\text{Binom}(V_{ms}|V_{ms}+R_{ms},\omega_{ms}\phi_{ms}) \end{aligned}$$

To compute the VAF reconstruction loss ϵ_Φ, we calculate the mean negative log-likelihood across all M mutations and S cancer samples, with

$$\epsilon_\Phi = -\frac{1}{MS}\sum_{m=1}^{M}\sum_{s=1}^{S}\log_2 \sum_t p(x_{ms}|\phi_{ms})p(t) \ . \tag{24}$$

As $p(x_{ms}|\phi_{ms}) \leq 1$ and $p(t) \leq 1$, given that both are discrete distributions, we have $\epsilon_\Phi \geq 0$. We report VAF reconstruction loss relative to a baseline, though this is not necessary—the absolute metric is still useful for quantifying the error in the tree-constrained subclonal frequencies that are part of a solution set. Nevertheless, by reporting error relative to a baseline, we can more easily see how well a method is faring, given that some datasets will necessarily yield higher absolute VAF losses than others. For simulated data, we use as the baseline the true subclonal frequencies that generated the data. For real data, we use as the baseline the subclonal frequencies computed by Pairtree (Section 6.3) for our expert-derived trees. In both cases, we use Eq. (24) to compute the baseline VAF loss $\tilde{\epsilon}_\Phi$, with the distribution over trees $p(t)$ consisting of a single tree, for which $p(t) = 1$. This yields the relative VAF loss

$$\hat{\epsilon}_\Phi = \epsilon_\Phi - \tilde{\epsilon}_\Phi \ .$$

These are the values reported in this study for VAF loss. The relative VAF loss $\hat{\epsilon}_\Phi$ can be negative, indicating that a method has found a better solution than the baseline. On simulated data, for instance, this can occur if there is only one tree consistent with the simulated subclonal frequencies, and the clone-tree-reconstruction method finds only that tree, to which it then fits the MAP subclonal frequencies. These will necessarily fit the observed data better than the true frequencies, yielding a negative relative VAF loss.

6.5.3 Relationship reconstruction error

In determining relationship reconstruction error (Section 3.4), we wish to compare the distribution over pairwise mutation relationships imposed by a method's set of candidate solutions relative to the simulated truth. Though there was a single true tree structure used to generate the observed data, we cannot simply compare the candidate solutions to the relations imposed by this true tree—the observed VAF data are noisy reflections of the true subclonal frequencies accompanying the true tree structure, and while the true tree will be consistent with the noise-free frequencies (i.e., it will not violate the constraints they impose), there may also be other consistent tree structures. Thus, our baseline must be not the single set of relationships imposed by the true tree, but the distribution over relationships implied by all tree structures consistent with the true subclonal frequencies. Determining this baseline requires that we enumerate all such trees (Section 6.5.4). We can then measure the quality of a set of proposed solution trees by the extent to which the distribution over pairwise relations they imply recapitulates the baseline. To excel according to this metric, methods must be able to recover the full set of trees permitted by the observed VAF data, rather than only a single consistent tree. Moreover, methods must be able to deal with noise inherent to the VAF observations, such that the methods find trees that make small violations of tree constraints if we take the VAFs as exact observations of the subclonal frequencies.

Suppose a dataset consists of M mutations. Every clone tree built for this dataset by a method places each mutation pair (A, B) unambiguously into one of the four pairwise relationships. We use M_{AB} to delineate the pairwise model for the mutation pair induced by a given clone tree. (Provided the method uses a fixed mutation clustering provided as input, the coincident relations are determined by the clustering, and so are fixed before the method is run.) Assume the method provides a distribution over different clone trees t, with the posterior probability of t represented as $p(t)$, such that $\sum_t p(t) = 1$. In this case, we can compute the posterior probability of the M_{AB} relation as $p(M_{AB}) = \sum_t p(M_{AB}|t)p(t)$, where

$$p(M_{AB}|t) = \begin{cases} 1 & \text{iff } (A, B) \text{ are in the } M_{AB} \text{ relation in } t \\ 0 & \text{otherwise} \end{cases} .$$

Using the set of true trees (Section 6.5.4), we will define $p(\tilde{M}_{AB})$ as the distribution over different relations for all N trees consistent with the true subclonal frequencies. For the true tree set, we will establish a uniform prior $p(t) = \frac{1}{N}$, since no true tree should be privileged over another. For the mutation

pair (A, B), we can now compute the Jensen-Shannon divergence (JSD) between a clone-tree-construction method's $p(M_{AB})$ and the true $p(\tilde{M}_{AB})$, which we denote as $\text{JSD}(M_{AB} \parallel \tilde{M}_{AB})$. We use the base-two logarithm in computing JSD, yielding a measurement in bits.

Given M mutations in a dataset, there are $\binom{M}{2} = \frac{M(M+1)}{2}$ mutation pairs (A, B). We thus define the relationship reconstruction error ϵ_R for a solution set as the mean JSD between pairs, such that

$$\epsilon_R = \frac{2}{M(M+1)} \sum_{(A,B)} \text{JSD}(M_{AB} \parallel \tilde{M}_{AB}) \ .$$

Using the mean allows us to compare ϵ_R values for datasets with different numbers of mutations, so that we can understand which result sets have more or less error. As an aside, though it may be tempting to view ϵ_R as the joint JSD for all mutation pairs, normalized to the number of mutation pairs, this perspective is wrong. The JSD can be defined with respect to the Kullback-Leibler (KL) divergence. Under our definition of $p(M_{AB}|t)$, every pair is independently distributed, such that the KL divergence of the joint distribution over all pairs is equal to the sum of KL divergences of individual pairs. This property is not, however, true for the JSD, and so our sum over pairs does not equal the JSD of the joint distributions.

Note that relationship error is similar to the probabilistic ancestor-descendant matrix (ADM) metric developed in [21], where it is referred to as metric 3B. To represent the ground truth, given M mutations and a single true tree $\tilde{t}$, metric 3B constructs four matrices of size $M \times M$, which can be represented by the $M \times M \times 4$ tensor denoted by T. Let T_{ijk} be the binary indicator corresponding to whether mutations i and j fall into pairwise relationship $k \in \{\text{ancestor}, \text{descendant}, \text{branched}, \text{coincident}\}$ (Section 6.1). Similarly, a candidate solution set can be represented with an $M \times M \times 4$ tensor denoted by R, with R_{ijk} indicating the probability that mutations i and j fall into relationship k. Both T and R are thus akin to the pairs tensor computed by Pairtree. To compute the similarity between T and R, the 3B metric concatenates the column vectors of each tensor's constituent $M \times M$ matrices, forming vectors of length $4M^2$ that we denote with $\overrightarrow{T}$ and $\overrightarrow{R}$. The metric 3B is then computed as the Pearson correlation between $\overrightarrow{T}$ and $\overrightarrow{R}$, equivalent to the mean-centered cosine similarity between these vectors.

Relationship error differs from metric 3B in two ways [21]. Though both operate on information about similarity in pairwise relations between a ground truth and candidate solution set, they compute distance differently. Relationship error uses the mean JSD between all pairs, and so ranges between 0 and 1, while metric 3B uses Pearson correlation, and so ranges between -1 and 1. More importantly,

relationship error's truth is defined with respect to all trees, and thus pairwise relationships, consistent with the true subclonal frequencies. Metric 3B, conversely, defines truth with respect to the single tree structure used to generate the data. Relationship error thus better reflects a method's performance, as it recognizes the fundamental ambiguity in tree structure.

6.5.4 Enumerating trees quickly

To enumerate all trees consistent with the true subclonal frequencies for a simulated dataset, henceforth termed "consistent trees," we first construct a directed graph tau. Given K subclones and S cancer samples, tau consists of a graph of $K+1$ nodes, with the ith node corresponding to the ith subclone, and the implicit node 0 that has no incoming edges. We place an edge from node i to node j in tau, such that $tau_{ij} = 1$, if node i is a potential parent of subclone j in a tree consistent with the subclonal frequencies $\Phi = \{\phi_{ks}\}$. The tau graph represents edges that will be present in at least one consistent tree. Thus, the spanning trees of tau compose a superset of the consistent trees—i.e., all consistent trees exist as a spanning tree of tau, but not all spanning trees of tau must be consistent trees.

By definition, $\phi_{0s} = 1$ for all cancer samples s. Without loss of generality, assume $\phi_{is} \geq \phi_{(i+1)s}$ for $i \in \{1, 2, \ldots, K-1\}$ for all cancer samples s, as the subclones can be sorted to fulfill this requirement without affecting the problem structure. We then construct τ as follows.

```
function make_tau(Φ) {
    let τ = {0} # Dimensions: (K + 1) × S
    for i ∈ (0, 1, ..., K − 1):
        for j ∈ (i + 1, ..., K):
            let τ_ij = 1 iff φ_is ≥ φ_js ∀s
    return τ
}
```

Listing 2: Algorithm to create graph adjacency matrix τ.

```
function enum_trees(Φ, τ, traversal ∈ {DFS, BFS}) {
    # Each partial is a triplet.
    # Element 1: index j ∈ {1, 2, ..., K} of node for which we must
    # next resolve parent, with 1 ≤ j′ < j fully resolved and j″ > j not yet resolved
    # Element 2: graph τ′ whose edges are a subset of those in τ,
    # with in-degree of nodes 1 ≤ i < j equal to 1.
    # Element 3: sum of subclonal frequencies for each node's children, for the portions
    # of the graph that have been fully resolved. This data structure allows us to
    # quickly determine whether a prospective parent choice violates tree constraints.
```

```
    let partials = [(1, copy(τ), zeroes(K + 1, S))]
    let trees = []

    while len(partials) > 0:
        if traversal == DFS:
            # For depth-first search, remove last element
            j, τ′, childsum = trees.pop(-1)
        else:
            # For breadth-first search, remove first element
            j, τ′, childsum = trees.pop(0)

        if j == K + 1:
            # We have resolved a single parent for nodes iin{1, 2, . . . , K},
            # so the tree is fully resolved.
            trees.append(τ′)
            continue
        parents = {i|i ∈ {0, 1, . . . , K} ∧ τ_ij = 1}
        for i in parents:
            # It's possible to leave this loop with all possible parents having been
            # deemed invalid because of a previous parent choice made in this
            # partial tree. In that case, the partial tree is effectively discarded.
            if ∃s s.t. childsum[i, s] + φ_js > φ_is:
                # We violate tree constraints in cancer sample s,
                # so reject this as a potential solution.
                continue
            new_csum = copy(childsum)
            new_csum[i, s] += φ_js ∀s
            τ̃ = copy(τ′)
            # We don't violate tree constraints, so for this partial tree,
            # resolve j's parent as node i.
            τ̃_cj = 0 ∀c ∈ {0, 1, . . . , K} − {i}
            partials.append((i + 1, τ̃, new_csum))
    return trees
}
```

Listing 3: Algorithm to enumerate trees based on τ graph.

By implementing this algorithm in Python and exploiting Numba, we can enumerate trees for all 576 simulated datasets quickly.Improving runtime through parallelization would be trivial, given that the algorithm need make only a single pass through each τ' graph, without having to backtrack "up" the

graph to alter edges corresponding to fully resolved parents. Though the algorithm offers the choice of DFS or BFS when exploring the τ graph, DFS is generally superior. As the tree enumeration algorithm proceeds down the τ graph, DFS allows it to quickly determine whether a parental choice made upstream of the nodes being considered was invalid, making it impossible for a downstream node to find any parent. DFS will quickly find this parent-less downstream node and so discard the partial tree. BFS, conversely, will keep the invalid partial tree in memory as it futilely resolves parents of other nodes before locating the parent-less node, while also storing in memory other variants of the invalid partial tree that retain the erroneous parental choice. The memory demands of the BFS algorithm variant can thus be much higher than DFS, while conferring no benefit.

Additionally, we could alter the `make_tau` algorithm to remove edges that are clearly invalid before beginning enumeration. Suppose in τ we have a node j whose only possible parent is i, and that there exists another node k who is also a possible child of i, implying $\phi_{is} \geq \phi_{js}$ and $\phi_{is} \geq \phi_{ks}$ for all cancer samples s. Furthermore, suppose $\phi_{is} - \phi_{js} < \phi_{ks}$ for at least one s. This implies that, by exploiting the knowledge that i must be the parent of j, we can eliminate i as being a possible parent of k. Moreover, by eliminating the i-to-k edge from τ, we may have determined with certainty the parent of k. Supposing this is true, we label k's parent as i', and can eliminate any edges from i' to other possible children k' that would now violate the tree constraints. In this manner, we can propagate constraints through τ at the algorithm's outset to eliminate edges from consideration. We have not implemented this optimization here, as tree enumeration was already sufficiently fast for our purposes.

7 Acknowledgements

J.A.W. was supported by a Canada Graduate Scholarship from the National Sciences and Engineering Research Council of Canada, a Sir James Lougheed Award of Distinction from the Government of Alberta, and additional awards and funding from the University of Toronto Department of Computer Science and School of Graduate Studies, the Ontario Institute for Cancer Research, and the Vector Institute for Artificial Intelligence. Experiments were run using computational resources provided by SciNet and Compute Canada. The authors gratefully acknowledge Bei Jia and José Bento for extending their method for computing subclonal frequencies.

8 Author contributions

Q.D.M. conceived of and supervised the project. Q.D.M. and J.A.W. designed the project with input from S.M.D., and J.A.W. implemented Pairtree and ran the experiments. J.A.W. and Q.D.M. drafted the manuscript, and L.D.S. provided extensive edits and feedback. S.M.D. and J.E.D. designed the project and collected the data that motivated Pairtree's development, and provided feedback throughout the project that guided the design of how Pairtree reports and visualizes its results. All authors reviewed and approved the final manuscript.

9 Competing interests statement

J.A.W., S.M.D., J.E.D., L.D.S., and Q.D.M. declare no competing interests.

10 Supplementary information

10.1 Clustering mutations into subclones

10.1.1 Clustering overview

Pairtree takes as input a clustering of mutations into subclones. Pairtree provides two mutation clustering algorithms for grouping mutations into subclones. Mutation clusters may also be generated by other methods. Alternatively, Pairtree may be run on the mutations directly without first clustering them into subclones, yielding a *mutation tree* instead of a clone tree. A mutation tree is equivalent to a clone tree in which each clone bears only a single distinct mutation, such that every tree node corresponds to a unique mutation.

Both of Pairtree's mutation-clustering algorithms use a Dirichlet process mixture model (DPMM) and perform inference via Gibbs sampling. The algorithms differ in how they define their probabilistic clustering models. Let $\Pi = \{\pi_1, \pi_2, ..., \pi_M\}$ represent a clustering of M mutations into K clusters, with π_i indicating the assignment to a cluster of mutation i, such that $\pi_i \in \{1, 2, ..., K\}$. Each cluster corresponds to a genetically distinct subclone. By virtue of using a DPMM, K is not fixed, but instead inferred from the data.

Let x represent the mutation read count data. From these, we will define the posterior distribution over clusterings

$$p(\Pi|x) = \frac{p(x|\Pi)p(\Pi)}{p(x)} \ . \tag{25}$$

Each clustering model defines its own likelihood $p(x|\Pi)$, but uses the same clustering prior $p(\Pi)$. The clustering prior draws on the DPMM concentration hyperparameter α, representing the cost of placing a mutation in a new cluster relative to adding it to an existing cluster. For K clusters over M mutations, with n_k mutations in cluster k, we define

$$p(\Pi) = \frac{\alpha^K \prod_{k=1}^{K}(n_k - 1)!}{\alpha(\alpha+1)\dots(\alpha+M-1)} \ . \tag{26}$$

Both clustering models use Gibbs sampling, such that each clustering iteration resamples the cluster assignment of each mutation individually, conditioned upon the assignments of all other mutations. Thus, we wish to compute $p(\pi_i|\tilde{\Pi}_i, \tilde{x}_i)$, where π_i indicates the cluster assignment of mutation i, Π is the cluster assignments of all mutations including i, and $\tilde{\Pi}_i$ represents the cluster assignments of all mutations excluding i, such that $\tilde{\Pi}_i = \Pi - \{\pi_i\} = \{\pi_1, \pi_2, \dots, \pi_{i-1}, \pi_{i+1}, \dots, \pi_M\}$.

By representing the data associated with all mutations except i with $\tilde{x}_i$, we get

$$\begin{aligned} p(\pi_i|\tilde{\Pi}_i, x) &= \frac{p(\Pi|x)}{p(\tilde{\Pi}_i|x)} \\ &= \frac{p(\Pi|x)}{p(\tilde{\Pi}_i|\tilde{x}_i)} \\ &= \frac{\frac{p(x|\Pi)p(\Pi)}{p(x)}}{\frac{p(\tilde{x}_i|\tilde{\Pi}_i)p(\tilde{\Pi}_i)}{p(\tilde{x}_i)}} \\ &\propto \frac{p(x|\Pi)p(\Pi)}{p(\tilde{x}_i|\tilde{\Pi}_i)p(\tilde{\Pi}_i)} \ . \end{aligned} \tag{27}$$

In Eq. (27), we use Eq. (26) to establish

$$\frac{p(\Pi)}{p(\tilde{\Pi}_i)} = \begin{cases} \frac{n_k}{\alpha+M-1} & \text{if } \pi_i = k \text{ and cluster } k \text{ already exists with } n_k \text{ members} \\ \frac{\alpha}{\alpha+M-1} & \text{if } \pi_i = k \text{ and cluster } k \text{ is a new cluster} \end{cases} \ . \tag{28}$$

To complete Eq. (27), we need only define $\frac{p(x|\Pi)}{p(\tilde{x}_i|\tilde{\Pi}_i)}$. We leave this definition to the clustering models

described in Section 10.1.2 and Section 10.1.3. Once this factor is defined, we can compute $p(\pi_i|\tilde{\Pi}_i, x)$ because we have in Eq. (27) a quantity proportional to it.

$$p(\pi_i = k|\tilde{\Pi}_i, x) = \frac{\frac{p(x|\tilde{\Pi}_i,\pi_i=k)p(\tilde{\Pi}_i,\pi_i=k)}{p(\tilde{x}_i|\tilde{\Pi}_i)p(\tilde{\Pi}_i)}}{\sum_{k'=1}^{K}\frac{p(x|\tilde{\Pi}_i,\pi_i=k')p(\tilde{\Pi}_i,\pi_i=k')}{p(\tilde{x}_i|\tilde{\Pi}_i)p(\tilde{\Pi}_i)}} \ . \tag{29}$$

We use this definition to perform Gibbs sampling, as described in Section 10.1.4.

10.1.2 Clustering mutations using subclonal frequencies

For each mutation i in each cancer sample s, we have a variant read count V_{is}, reference read count R_{is}, total read count $T_{is} = V_{is} + R_{is}$, and probability of observing the variant allele ω_{is}. To cluster mutations using subclonal frequencies, we first define for each mutation m in each cancer sample s an adjusted total read count $T'_{ms} = \max(\omega_{ms}T_{ms}, V_{ms})$. Thus, T'_{ms} represents the (potentially fractional) number of reads originating from the variant allele across all cells, regardless of whether the reads include mutation m on that allele. The complete data likelihood is then defined using the following notation:

- S: number of cancer samples
- K: number of clusters
- M: number of mutations
- ϕ_{ks}: subclonal frequency of cluster k in sample s
- $C_k \subseteq \{1, 2, ..., M\}$: set of mutations assigned to cluster k, with $C_i \cap C_j = \emptyset$ for all i and j

This yields the complete data likelihood

$$p(x|\Pi) = \prod_{s=1}^{S}\prod_{k=1}^{K}\int_0^1 d\phi_{ks} \prod_{m\in C_k} p(x_{ms}|\phi_{ks}) \ ,$$

with $p(x_{ms}|\phi_{ks}) = \text{Binom}(V_{ms}|T'_{ms}, \phi_{ks})$. Strictly speaking, as T'_{ms} may take a fractional value, it may not be a valid parameter choice for the binomial. Nevertheless, for computational convenience, we compute the integral over the binomial using the beta function, which allows for continuous T'_{ms} values. Consequently, we have

$$p(x|\Pi) = \prod_{s=1}^{S}\prod_{k=1}^{K}\left[\prod_{m\in C_k}\binom{T'_{ms}}{V_{ms}}\right]\beta(1 + \sum_{m\in C_k} V_{ms}, 1 + \sum_{m\in C_k} T'_{ms} - V_{ms}) \ . \tag{30}$$

By Eq. (29), we need only define $\frac{p(x|\Pi)}{p(\tilde{x}_i|\tilde{\Pi}_i)}$ to complete the definitions required for Gibbs sampling. This follows easily from Eq. (30), yielding

$$\frac{p(x|\Pi)}{p(\tilde{x}_i|\tilde{\Pi}_i)} = \prod_{s=1}^{S} \binom{T'_{is}}{V_{is}} \frac{\beta(1 + V_{is} + \sum_{m \in C_k} V_{ms}, 1 + (T'_{is} - V_{is}) + \sum_{m \in C_k} T'_{ms} - V_{ms})}{\beta(1 + \sum_{m \in C_k} V_{ms}, 1 + \sum_{m \in C_k} T'_{ms} - V_{ms})} \ . \tag{31}$$

This allows us to proceed with Gibbs sampling, as described in Section 10.1.4.

10.1.3 Clustering mutations using pairwise relations

As an alternative to clustering with subclonal frequencies, we can cluster mutations using the pairwise relations described in Section 6.1. To do so, we compute the posterior distributions over pairwise relations for every pair of individual variants A and B, rather than the supervariants defined from an established clustering that are used for tree search. Computing the pairwise posterior distributions over relationships M_{AB} necessitates that we first redefine the pairwise prior described in Section 6.1.6 to permit non-zero mass on the *coincident* relationship. For this, we allow the user to set a constant P representing the prior probability that mutations A and B are coincident, with $P = \frac{1}{4^S}$ for S cancer samples by default, yielding

$$p(M_{AB}) = \begin{cases} P & \text{if } M_{AB} = \textit{coincident} \\ \frac{1}{3}(1-P) & \text{if } M_{AB} \in \{\textit{ancestor}, \textit{descendent}, \textit{branched}\} \\ 0 & \text{if } M_{AB} = \textit{garbage} \end{cases} \ .$$

We define $p(M_{ab} \neq \textit{coincident}|x) = 1 - p(M_{ab} = \textit{coincident}|x)$. After computing these pairwise relation posteriors for every mutation pair $(a, b) \in \{1, 2, \ldots, M\} \times \{1, 2, \ldots, M\}$ with $a > b$, we can define the clustering data likelihood as

$$p(x|\Pi) = \prod_{(a,b)} \mathbb{1}_{\pi_a = \pi_b} p(M_{ab} = \textit{coincident}|x) + \mathbb{1}_{\pi_a \neq \pi_b} p(M_{ab} \neq \textit{coincident}|x) \ . \tag{32}$$

As we consider every pair (a, b) without also including the pair (b, a), there are $\binom{M}{2}$ factors in the

product for M mutations. This notation relies on the indicator function

$$\mathbb{1}_c = \begin{cases} 1 & \text{if } c \text{ is true} \\ 0 & \text{otherwise} \end{cases}.$$

From this, we can define $\frac{p(x|\Pi)}{p(\tilde{x}_i|\tilde{\Pi}_i)}$, completing the definitions required for Gibbs sampling.

$$\frac{p(x|\Pi)}{p(\tilde{x}_i|\tilde{\Pi}_i)} = \prod_{s=1}^{S} \left[\prod_{a \in \{1,2,\ldots,i-1,i+1,\ldots,M\}} \mathbb{1}_{\pi_i = \pi_a} p(M_{ia} = coincident|x) + \mathbb{1}_{\pi_i \neq \pi_a} p(M_{ia} \neq coincident|x) \right]. \tag{33}$$

Thus, $\frac{p(x|\Pi)}{p(\tilde{x}_i|\tilde{\Pi}_i)}$ is a product over the S cancer samples and $M-1$ pairs that include mutation i. This allows us to proceed with Gibbs sampling, as described in Section 10.1.4.

10.1.4 Performing Gibbs sampling

Pairtree clusters mutations using Gibbs sampling, drawing on the probabilistic framework given in Eq. (29), and the subclonal frequency likelihood Eq. (31) or pairwise relationship likelihood Eq. (33). The primary advantage of the subclonal frequency model is that, unlike the pairwise model, it does not require the time-intensive computation of the pairs tensor before clustering can begin. The pairwise model, conversely, can be easily applied to data types other than bulk sequencing that can be represented within the pairwise relation framework, such as single-cell sequencing.

By default, the algorithm runs a total of C chains, with C set to the number of CPU cores present on the system by default, and $P = C$ executing in parallel. Both P and C can be customized by the user. Each chain takes 1000 samples by default, which can also be changed by the user. Unlike the tree search algorithm, the clustering algorithm makes no attempt to discard burn-in samples from each chain. As tree search relies on a single clustering common to all trees, we select the clustering result with the highest posterior probability as the algorithm's output. Nevertheless, the user could easily adapt the implementation to represent different possible clusterings alongside their posterior probabilities, conferring insight into multiple possible solutions.

The subclonal frequency and pairwise relationship clustering models use different clustering initializations, purely as an implementation artifact. The subclonal frequency models simply assigns all variants to a single cluster. Conversely, the pairwise relationship model places each variant in a separate cluster.

Alternative, the pairwise model also permits the user to specify an initial clustering to use for initialization. In this case, user-specified clusters can be merged, but will never be split, such that the user can force multiple variants to always remain in the same cluster.

Two hyperparameters affect clustering results. The first, α, is used in Eq. (26), with higher values corresponding to an increased number of clusters. Let $\hat{\alpha}$ be the value provided by the user as input to the algorithm. Given a dataset with S cancer samples, The α value used in Eq. (26) is computed from this as $\alpha = 10^{S\hat{\alpha}}$, with $\hat{\alpha} = -2$ by default. Representing α on a logarithmic scale via $\hat{\alpha}$ makes representing especially large or small values of α more convenient for the user, while scaling it with S ensures that the algorithm's preference for placing data points in new clusters is unaffected by the magnitude of posterior weight contributed by data likelihood factors—i.e., each cancer sample-specific likelihood is effectively weighted by its own $10^{\hat{\alpha}}$ prior factor in computing the posterior. Finally, to prevent numerical issues, we force $\alpha \in [\exp(-600), \exp(600)]$.

The second clustering hyperparameter is P, the prior probability of two mutations being coincident (Section 10.1.3). Similar to how the α parameter is specified, the algorithm ensures that the number of cancer samples S does not affect the algorithm's preference for starting new clusters by taking as input $\hat{P}$, with $P = \hat{P}^S$. By default, we take $\hat{P} = \frac{1}{4}$, such that we enforce a uniform distribution over the four possible pairwise relations for each cancer sample.

10.2 Running comparison methods

All methods were run on systems with dual Intel Xeon 6148 CPUs, with 40 CPU cores and 192 GB of RAM. Methods were allowed up to 24 hours of compute time per dataset, and were terminated if they exceeded this threshold.

We used CITUP v0.1.2 from `https://anaconda.org/dranew/citup`, corresponding to the most recent revision at `https://bitbucket.org/dranew/citup/`. CITUP offers both a quadratic integer programming (QIP) mode and a faster iterative approximation to it. We used the QIP mode because it alone was able to take a fixed clustering as input. The iterative approximation insists on clustering mutations itself, which would have unfairly disadvantaged CITUP relative to other methods, as it would not have known which mutations belonged to which clusters. Regardless, we tried running CITUP's iterative mode with the same supervariant-based approach we used for PhyloWGS (described below), but this did not improve CITUP's failure rate.

We used LICHeE version `26c2a701` from `https://github.com/viq854/lichee`. LICHeE could not compute subclonal frequencies, so we invoked Pairtree to perform this task using the tree structures

LICHeE produced. LICHeE can optionally cluster mutations itself, but we gave it the correct mutation clustering as input.

We used PASTRI version `1d2fb83c` from `https://github.com/raphael-group/PASTRI`, which is limited to running on datasets with 15 or fewer subclones. PASTRI was given the correct mutation clusters as input.

We used PhyloWGS version `2205be16` from `https://github.com/morrislab/phylowgs`. PhyloWGS did not offer a means of taking a fixed clustering as input, unlike the other four methods examined, and so was disadvantaged in the method comparisons. We provided as much clustering information to PhyloWGS as possible by using *supervariants* (Section 6.2.8), preventing the method from splitting clusters such that mutations from the same cluster would be assigned to different subpopulations. Nevertheless, PhyloWGS could still merge clusters such that multiple clusters' variants would be assigned to the same subpopulation.

10.3 Examining method failures

CITUP produced results for 137 of the three-subclone datasets (76%), failing on the remainder. CITUP also failed on all datasets with 10, 30, or 100 subclones. For 3- and 10-subclone failures, 137 exited with the error `failed to optimize LP: Infeasible`, while 34 failed with `failed to optimize LP: Unknown`. Another 52 of the 10-subclone runs failed to finish in 24 hours. All 216 datasets with 30 or 100 subclones failed with the error `create_trees failed to complete`.

LICHeE succeeded on 477 cases. its 99 failures all occurred on 100-subclone datasets, where the method failed to finish in 24 hours.

PASTRI only supports 15 or fewer subclones, and so failed on all 216 datasets with 30 or 100 subclones. For 37 datasets with 3 or 10 subclones, PASTRI succeeded in sampling at least one tree with subclonal frequencies. On 22 datasets, all of which had 10 subclones, PASTRI failed to finish within 24 hours. PASTRI terminated without sampling any trees for 220 datasets, comprising a mixture of 3- and 10-subclone cases. Additionally, on 81 datasets, PASTRI sampled one or more trees, but failed at later steps of its pipeline, without writing usable output. These 81 cases included four types of failure.

- PASTRI failed the assertion `assert(round(slack[j],10) >= 0)` in `gabow_myers.py` for one ten-subclone case.
- PASTRI failed with a `ValueError: too many values to unpack` exception for other cases.

- In some cases, the trees had fewer nodes than expected, despite being given the correct number of subclones as input.

- Some cases included invalid blank lines for some of their subclonal frequencies, evidently stemming from an error when frequencies of exactly 1 were output as blanks.

PhyloWGS succeeded on 535 datasets. Amongst these, it finished all 1000 burn-in and 2500 posterior samples within 24 hours for 463. For another 72 cases, comprising a mixture of 30- and 100-subclone datasets, it finished the burn-in samples and at least one posterior sample, without finishing all 2500 posterior samples. These 72 cases were counted as successes, but assigned wall-clock times and CPU times of 24 hours (Section 10.5.2). The remaining 41 runs failed to complete their burn-in portion within 24 hours, and so were counted as failures. All such cases had 100 subclones.

10.4 Why existing algorithms failed

Given that the algorithms we compared against often failed to produce results on our simulated datasets, considering possible reasons for this poor performance is a worthwhile exercise. When building trees with few subpopulations, exhaustive enumeration algorithms are attractive, as they promise to find the single best tree by considering all possibilities. As our simulations demonstrated, however, enumeration algorithms cannot cope with more than ten subpopulations, as the number of possible trees becomes too great, even when constraints are employed to reduce possible tree configurations. Stochastic search algorithms are a superior approach when faced with numerous subpopulations, provided they can locate high-likelihood regions of tree space and limit their search to those areas. When this space is searched blindly, however, it remains difficult to navigate, given the massive number of possible clone trees formed from having many subpopulations.

We hypothesize that CITUP attempted to enumerate all trees with a given number of subpopulations, but faced too many trees to make this approach feasible when provided with more than three subpopulations. Thus, CITUP is limited to datasets with only a small number of subclones.

PASTRI attempted to overcome the difficulties of enumeration by first sampling subclonal frequencies, then enumerating only trees consistent with those frequencies. Because mutation VAFs are independent from the tree when conditioned upon the subclonal frequencies, PASTRI can treat its approximate posterior over subclonal frequencies as a proposal distribution for importance sampling, where the target is the posterior distribution over subclonal frequencies permitted by the true tree. The PASTRI implementation is nevertheless limited to 15 subpopulations [37]. Even with ten subpopulations or fewer, because

PASTRI samples frequencies without considering tree structure, the frequencies are often inconsistent with any tree when the algorithm is given many cancer samples, as the frequencies collectively impose constraints that rule out all possible trees. A weakness of this approach becomes apparent in real cancer datasets, where new subpopulations often emerge when they acquire driver mutations that confer a strong selective advantage, leading to them displacing their parents such that the subclonal frequency of the child is only slightly greater than that of the parent. Indeed, this situation often occurs in the leukemias considered here. As PASTRI samples subclonal frequencies before enumerating consistent trees, the frequencies sampled for children in this situation will often by chance be slightly higher than their parent, rendering the correct tree structure impossible to recover.

LICHeE fared better than CITUP and PASTRI, as it first constructed a directed acyclic graph (DAG) containing possible trees permitted by the noisy subclonal frequency estimates provided by the VAFs, then only considered spanning trees of this graph [19]. However, this approach could not scale to most 100-subpopulation trees, presumably because the corresponding DAGs have too many spanning trees. Even in settings with 30 or fewer subclones, LICHeE exhibited considerably higher error than Pairtree both with respect to subclonal frequencies and pairwise relations, despite us computing subclonal frequencies for LICHeE's tree structures using the same algorithm as Pairtree. This suggests that the DAGs did not include as spanning trees good tree candidates, or that the error scoring function LICHeE used to indicate tree quality did not properly reflect tree quality. Some of LICHeE's shortcomings may have arisen because it takes as input only VAFs, rather than mutation read counts. Consequently, LICHeE has no knowledge of how precisely the VAFs should reflect underlying subclonal frequencies, unlike methods such as Pairtree that use a binomial observation model.

When PhyloWGS fared poorly, its performance could often be attributed to its inability to use a fixed clustering, unlike the other methods. Because we gave PhyloWGS supervariants rather than individual mutations in an attempt to mitigate this discrepancy, even though PhyloWGS could not split clusters into multiple subclones, the algorithm could effectively merge distinct subclones into single entities, causing considerable pairwise relationship error.

Given that non-Pairtree methods may have been particularly prone to failing on the most challenging simulations, summary statistics reported for these methods may be unfairly biased in their favour, as they would only reflect performance on less-challenging datasets. Nevertheless, when we compare Pairtree to each method on only the subset of datasets for which the comparison method succeeded (Fig. S4), we see that Pairtree almost always produces better VAF losses, with the only exception being several 100-subpopulation datasets where PhyloWGS beat Pairtree.

In general, stochastic search algorithms are a superior approach relative to exhaustive enumeration methods when faced with numerous subpopulations, since they avoid the exponential growth in number of trees as a function of number of subclones [20]. For stochastic search algorithms to work well, they must locate high-likelihood regions of tree space and limit their search to those areas. However, as data become richer, tree space is rendered more complex, such that existing search algorithms struggle to navigate through it. This was apparent with PhyloWGS, which consistently exhibited higher error for many-cancer-sample simulations than few-cancer-samples ones. By constructing the pairs tensor and using this as a guide to tree search, Pairtree is better able to cope with many cancer samples and the constraints they impose.

10.5 Comparing the computational costs of methods

10.5.1 Criteria for measuring computational costs

Pairtree and the four methods we compared to it differed substantially in the computational costs they imposed, as well as their ability to conduct computations in parallel using multiple CPU cores, using either multiple processes or multiple threads. Pairtree, CITUP, and PhyloWGS had the ability to conduct computations in parallel, while LICHeE and PASTRI did not. We used this ability only for Pairtree, however. For CITUP, using the method's multiple-process mode did not improve its failure rate. Though PhyloWGS allows running multiple MCMC chains in parallel, doing so was not helpful for this study—PhyloWGS' failures stemmed from an inability to sample enough trees to form a posterior estimate in 24 hours from a single chain, and so increasing the number of chains only amplified the computational burden without improving the failure rate.

We measured runtime on each simulated dataset for each method both with respect to CPU time and wall-clock time. CPU time indicates the number of CPU seconds consumed by a method's primary process and any subprocesses or threads it spawned, in either user or kernel mode. Wall-clock time measures the elapsed time a method took. Runs that exited with an error without producing a result, or that failed to finish in 24 hours of wall-clock time, are excluded from the results. Thus, the maximum wall-clock time observed for any method is 86,400 seconds. Considering both CPU time and wall-clock time is worthwhile—CPU time reflects the total computational burden imposed by a method, while wall-clock time indicates how long a method will take to finish in a multi-CPU environment. We conducted all experiments on compute nodes using dual Intel Xeon Gold 6148 CPUs, such that 40 CPU cores were available to each method. On systems with only one CPU, we expect that wall-clock time will generally

be slightly more than CPU time, as that single CPU must also be used for the operating system and other concurrent tasks. In our experiments, however, non-Pairtree methods that used only a single CPU core for a run typically achieved wall times that were less than CPU times, given that system or library calls they made (e.g., to numerical routines in the Python library NumPy) could be parallelized.

10.5.2 Examining method runtime

In cases with 3, 10, or 30 subclones, we see similar patterns of CPU time consumed for Pairtree, LICHeE, and PhyloWGS (Fig. S6). These three methods succeeded on all simulations with 30 or fewer subclones, simplifying comparisons. Across datasets with 3, 10, or 30 subclones, LICHeE was fastest, realizing median CPU times of 0.46 seconds, 1.6 seconds, and 2,722 seconds, respectively. This characterization is unfair to other methods, however, as LICHeE did not compute subclonal frequencies for the tree structures it produced. To overcome this deficiency, we invoked Pairtree to compute subclonal frequencies for LICHeE's results, but did not include the time this step took in LICHeE's CPU time or wall-clock time measurements. Pairtree was slower than LICHeE, taking median times of 993 seconds, 1506 seconds, and 4391 seconds in settings with 3, 10, or 30 subclones, respectively. PhyloWGS was faster than Pairtree for 3-subclone cases, needing only a median CPU time of 509 seconds, but slower in 10- and 30-subclone cases, taking median times of 1,781 and 35,472 seconds. When we compare each method's CPU time to Pairtree's on only the subset of datasets for which each method succeeded, these observations are reinforced, with LICHeE usually being faster than Pairtree excepted for outliers corresponding to 100-subclone cases, and PhyloWGS usually being slower than Pairtree (Fig. S8). As CITUP could not produce results for datasets with more than three subclones, and PASTRI failed on most three- and ten-subclone cases, we do not consider their performance in depth, except to note that CITUP and PASTRI are generally fast when they can produce results for three-subclone cases, while PASTRI is slower than all other methods on the 4% of 10-subclone datasets where it ran successfully (Fig. S6).

When examining wall-clock times, however, we see that Pairtree fares better because of its use of multiple CPU cores. In few-subclone cases, Pairtree is still slower than LICHeE, with Pairtree taking median wall times of 55 seconds and 69 seconds in the 3- and 10-subclone settings, respectively, while LICHeE took 0.326 and 0.93 seconds, respectively (Fig. S7). Conversely, Pairtree is faster than LICHeE in settings with more subclones. For 30-subclone datasets, Pairtree takes a median 148 seconds, while LICHeE takes 2,685 seconds. PhyloWGS was considerably slower with respect to wall-clock time than LICHeE and Pairtree across all three settings. When runtime on individual datasets is examined, Pairtree demonstrates a comparable or superior wall-clock time relative to PhyloWGS and LICHeE (Fig. S9).

Datasets with 100 subclones warrant separate consideration. Pairtree took a median 23,827 seconds of CPU time on 100-subclone cases (Fig. S6), but only a median 675 seconds of wall-clock time (Fig. S7). LICHeE produced results for only 8% of these datasets, where it took a median 74,790 seconds of CPU time. PhyloWGS yielded output for 62% of such datasets, taking median times of 86,400 seconds for both CPU time and wall-clock time. The method's median times being equal to 24 hours reflects how we handled incomplete runs. According to the (default) parameter settings used for these experiments, PhyloWGS discards the first 1000 samples from its MCMC chain as burn-in samples not reflective of the true posterior, then takes an additional 2500 posterior samples. If the method finished the 1000 burn-in samples within the 24-hour wall-clock period permitted, but completed fewer than the 2500 posterior samples, we used whatever partial set of posterior samples the algorithm produced to evaluate its accuracy, while recording its runtime as 24 hours. The median times being 24 hours indicate that most successful 100-subclone runs fell into this category. Conversely, the 68% of 100-subclone cases where we recorded no output correspond to instances where PhyloWGS could not finish its initial 1000 burn-in samples.

10.5.3 Evaluating the performance costs of Pairtree's two stages

The two primary steps composing the Pairtree algorithm are computing pairwise relations between subclones and searching for trees. Tree search includes computing MAP subclonal frequencies for each tree structure. The amount of computation needed to build the pairs tensor is fixed, as a distribution over relations for every pair must be computed regardless of how many CPU cores are available. As relations for each subclone pair are independent of all other subclones, the pairwise computations are embarrassingly parallel, such that they can be trivially computed in parallel for all pairs. Thus, though the total computational burden represented by CPU time is constant, the wall-clock time can be greatly reduced by using more CPU cores, with N cores reducing the time needed for this stage nearly by a factor of N. By comparison, tree search requires that each MCMC chain acquire samples serially, such that any one chain cannot be parallelized. Multiple chains, however, can execute in parallel, increasing CPU time consumed in proportion to the number of chains, but with little effect on wall-clock time.

In the Pairtree experiments illustrated throughout this paper, we used all available 40 CPU cores on our compute nodes to calculate pairwise relations in parallel, and to run 40 parallel MCMC chains for tree search. Doing so greatly inflated CPU time relative to wall-clock time, but likely was not necessary to realize good results. Results of nearly equal quality could perhaps have been obtained from Pairtree using fewer chains—while any one chain may become mired in pathological regions of tree space corresponding

to a local optimum, such that multiple chains initialized from different positions can yield better samples, we likely did not need all 40 chains to realize this benefit. Nevertheless, even if all 40 chains were necessary to produce results of this quality, running those chains serially on a single CPU would have been feasible. In this case, the wall-clock time would have been approximately equal to the CPU time. Amongst the 576 simulations, Pairtree's longest run was on a 100-subclone, 100-cancer-sample dataset that took 1,110 seconds of wall-clock time (Fig. S7) and 36,606 seconds of CPU time (Fig. S6). Running all 40 chains serially on a single CPU would thus have resulted in a wall-clock time of slightly over 10 hours.

We can understand the relative computational costs of Pairtree's two primary steps by comparing the runtimes of the full Pairtree algorithm to the portion that computes the pairwise relations, denoted as *pairs tensor*. By subtracting the pairs tensor runtime from that of full Pairtree, we reveal the cost of tree search alone. Comparisons are most informative for the 100-subclone, 100-cancer-sample datasets, where the runtimes are longest and differences are thus clearest. For instance, the single most costly Pairtree run took 1,110 seconds of wall-clock time and 36,606 seconds of CPU time, as above (Figs. S6 and S7). Computing the pairs tensor alone took 81 seconds of wall-clock time and 2,666 seconds of CPU time. Whether we consider CPU times or wall-clock times, we see 7% of Pairtree's time went to computing pairwise relations, while 93% went to tree search. If the number of CPU cores dedicated to this run were cut tenfold to four CPUs rather than 40, we would expect the wall-clock cost of computing pairwise relations to increase proportionally to 810 seconds, while the CPU time would remain constant. Conversely, the wall-clock cost of tree search could be kept constant at 1,110 seconds by reducing the number of MCMC chains to four, at a potential cost in result quality. In this instance, we would expect Pairtree to take $810 + 1,110 = 1,920$ seconds, with tree search consuming 58% of the total. Thus, the relative burdens of computing the pairs tensor and performing tree search depend both on the number of CPU cores used in parallel, and on the number of MCMC chains from which the user elects to sample trees.

10.6 Multiple trees are often consistent with observed data, which Pairtree can accurately characterize

When building trees, algorithms draw on the subclonal frequencies of constituent subclones across cancer samples and relationships between these frequencies to determine possible tree structures. Thus, to assess method performance on simulated data, we can enumerate all tree structures consistent with the true subclonal frequencies used to generate the data, yielding a distribution over trees. This distribution will

include the true tree used to generate the data, as well as any other tree structures that are also consistent with the subclonal frequencies. A perfect method would be able to recover this distribution exactly, despite being given only noisy estimates of the true subclonal frequencies via the observed mutation frequencies. To evaluate a method, we can then determine the extent to which its tree distribution matches the true distribution of all trees consistent with the true subclonal frequencies.

Amongst our 576 simulated datasets, if only one cancer sample is provided, there are usually multiple trees consistent with the data (Fig. S10a), regardless of how many subclones are in the tree. This reaches an extreme in our ten-subclone, single-sample simulations. This illustrates the importance of understanding uncertainty in these reconstructions, rather than simply producing a single answer (Section 3.9)—the perfect method should represent all of these trees as being equally consistent with the data, such that the user should have no reason to prefer any one structure over the others. Drawing on more cancer samples reduces this uncertainty, with most ten-sample datasets possessing only a single possible tree across the three-, ten-, and 30-subclone settings (Fig. S10a). With 100 subclones, ten samples still permits little uncertainty, with the number of possible trees rarely exceeding ten. Note, however, that in this simulated setting, multiple samples are likely to be more powerful than they would be for real cancers. Here, each sample had its subclonal frequencies generated independently from other samples, increasing the chance that the sample induces tree structure constraints because its frequencies are different from all other samples. In reality, samples are likely to have correlated frequencies, given that they may be taken from similar spatial or temporal sites in the cancer that have similar population proportions.

By computing the entropy of tree distributions, we can characterize how many high-confidence trees exist in the distribution. Effectively, the entropy is a posterior-weighted count of the number of trees, with the weights in the true tree distribution being uniform because all solutions are equally consistent with the data. To determine how many high-confidence solutions was Pairtree was finding relative to the number of possible solutions, we compared Pairtree's tree entropy for each simulated dataset to the entropy of the true tree distribution (Fig. S10b). Pairtree's entropy generally tracked the true entropy well, suggesting that Pairtree's uncertainty was usually consistent with the uncertainty in the true tree distribution. Notably, in settings where the number of cancer samples was higher than the number of subclones, there was only ever one true tree (Fig. S10a), while Pairtree's tree distribution entropy exceeded the true distribution's entropy by more than 5.9×10^{-6} bits with only one exception across 181 simulations (Fig. S10b). These results demonstrate that, when the data is sufficiently high-resolution as to permit only a single solution, Pairtree finds only a single solution.

Though examining tree distribution entropies reveals the number of high-confidence trees Pairtree

finds, it says nothing about the quality of those trees. To gain further insight, we can view a distribution over trees as inducing a distribution over the *parents* of each subclone. For a given dataset, to compare the Pairtree-computed tree distribution to the distribution of trees consistent with the true subclonal frequencies, we can consider the joint Jensen-Shannon divergence between parent distributions induced by these tree distributions, normalized to the number of subclones in the tree such that the divergence will always lie between zero bits and one bit. We refer to this metric as the *parent JSD*. Even if the tree distributions have no overlap—which could occur, for instance, if there is only a single true tree that Pairtree fails to locate—the parent JSD nevertheless allows the distributions to have a small divergence if they agree on parent choice for most subclones. We see that the parent JSD falls as the number of samples increases for a given number of subclones (Fig. S10c), suggesting that Pairtree can efficiently exploit the constraints provided by additional cancer samples to produce higher-quality trees. Moreover, when the number of samples exceeds the number of subclones such that there is only one tree consistent with the true subclonal frequencies (Fig. S10a), the parent JSD is effectively always zero, complementing the tree entropy analysis (Fig. S10b) to show that the one tree Pairtree finds is almost perfectly consistent with the true tree. Additionally, when the pairwise relation error is examined at a more granular level (Fig. S10d), we see that for a given number of subclones and samples it is always less than the parent JSD. This suggests that, even when Pairtree doesn't perfectly determine the parents of each subclone, the distributions over relationships between subclones (e.g., ancestor-descendant or on-different-branches) are closer to the truth. The quality difference between pairwise relation distributions and parent distributions is stark for the 100-subclone setting. Though Pairtree only rarely finds the correct parents, demonstrated by the parent JSDs that are close to one (Fig. S10c), the pairwise relation errors are much lower (Fig. S10d), indicating that the higher-level relationships between subclones are closer to being correct.

10.7 Characteristics of simulated data

10.7.1 Trees are dominated by small subclones

Examining statistics of simulated data illustrates factors that affect each clone-tree-reconstruction algorithm's ability to recover good solutions. The nodes of each clone tree correspond to populations, with subclones consisting of sub-trees made up of a population and all its descendants (Section 3.1). Thus, a tree with K populations defines K subclones. Subclones are nested within trees—a subclone with population i at its head and c total populations is also part of a subclone with i's parent at its head and $c+1$ total populations (excluding the root subclone that corresponds to the entire tree, which has

no parental subclone). Characterizing subclone composition within simulated data is helpful, as several properties of the simulated trees depend on how many populations compose each subclone.

A fully linear tree with no branching that contains K populations would yield a uniform distribution over subclones consisting of $1, 2, \ldots, K$ populations, with exactly one subclone of each size. Branching within trees depletes the contribution of larger subclones, replacing them with smaller ones. Because of how we constructed simulated tree structures (Section 6.4.2), we see that small subclones dominate regardless of the number of populations within a tree (Fig. S11), with most subclones consisting of ten or fewer populations in the 30- or 100-subclone trees. In the tree generation algorithm, we choose parents for each population in turn, selecting the preceding population as parent with 75% probability, and otherwise choosing a parent uniformly from the other nodes already in the tree. As a result, the length of linear chains of populations within the tree roughly follows a geometric distribution. Linear chain length deviates from the distribution, however, because a node may choose as its parent the end of a different chain, allowing that chain to continue extending under a new geometric process.

10.7.2 Tree construction becomes increasingly difficult with more subclones

Large trees containing many subclones are more difficult to reconstruct than small trees. In part, this is because the number of possible tree structures scales exponentially with the number of populations [20]. We must also consider, however, how relationships between subclones become more difficult to infer as the number of subclones grows, which is a factor independent of tree structure. Given how we generated the simulated data (Section 6.4.2), we can derive statistics of the simulated data, then use them to show how the difficulty of inferring relationships between subclones changes according to the numbers of subclones and cancer samples.

In determining the proper placement of a population within a clone tree, two properties related to population frequencies affect the difficulty of this task. Firstly, if a population k has a near-zero population frequency η_{ks} in a cancer sample s, the VAFs associated with its mutations in that sample will be difficult to distinguish from the VAFs of mutations in k's parent, which we will denote as population j. This occurs because the VAFs for mutations that arose in each population are sampled based on the subclonal frequencies of the populations' subclones (Section 6.4.2), which are computed from the sum of the population frequencies composing the subclone (Section 6.3.1). Thus, when $\eta_{ks} \approx 0$, we have $\phi_{ks} \approx \phi_j s$, and the VAFs in k and j will be nearly the same. Assuming there are no cancer samples other than sample s, we could thus swap the positions of k and j in the tree without affecting tree likelihood—both populations would have nearly the same subclonal frequency fit to them in the tree, which would

fit the two sets of VAFs almost equally well. Larger population frequencies avoid this situation, making clearer the proper ordering of parents and children.

Intuitively, as more populations appear in a tree, the η_{ks} frequencies will become smaller on average, as the unit mass apportioned by the Dirichlet distribution from which the frequencies are drawn must be split amongst more entities. Indeed, by the properties of the Dirichlet distribution, for K subpopulations in a sample s with $[\eta_{0s}, \eta_{1s}, \ldots, \eta_{Ks}] \sim \text{Dirichlet}(\alpha, \alpha, \ldots, \alpha)$ (Section 6.4.2), we have $\mathbb{E}[\eta_{ks}] = \frac{1}{K}$. This is evident when we examine the distribution over η_{ks} frequencies for each population in the simulated trees (Fig. S12A), where the largest frequency observed across cancer samples for each population is typically close to 1 for trees with three subclones, but gets progressively smaller as the number of subclones increases, with populations in 100-subclone trees dominated by small frequencies. To distinguish a population from its parent, it need have a non-negligible η_{ks} frequency in only one sample s, which is part of why adding cancer samples is so helpful in resolving evolutionary relationships between populations, and ultimately reconstructing an accurate clone tree.

The second property related to population frequency that affects the difficulty of clone tree reconstruction is the variance over cancer samples s in a subclone k's frequencies ϕ_{ks}. Suppose you are trying to resolve the position of two subclones A and B in a tree, using the frequencies in cancer samples s and s'. To gain the greatest benefit from having two samples rather than only one, we want there to be as much variance as possible in the subclonal frequencies between samples. The power of multiple samples comes from these differences—for instance, if $\phi_{As} > \phi_{Bs}$, but $\phi_{As'} < \phi_{Bs'}$, we conclude that A cannot be the ancestor of B, and B cannot be the ancestor of A, since an ancestral subclone must have a frequency at least as high as its descendants across every cancer sample. This is termed the *crossing rule* [36], and leads to the conclusion that A and B must occur on separate tree branches. Unfortunately, as we observe only a noisy estimate of the subclonal frequencies through the VAFs, if the subclonal frequencies for A and B are nearly the same in both samples, the noise in VAFs can obscure this relationship. The less variance there is between ϕ_{As} and $\phi_{As'}$, and between ϕ_{Bs} and $\phi_{Bs'}$, the more likely that $|\phi_{As} - \phi_{Bs}| = |\phi_{As'} - \phi_{Bs'}| < \epsilon$ for some near-zero ϵ, and the more difficult it will be to utilize the crossing rule with our noisy observations.

Suppose we have a subclone C composed of $|C| \leq K$ populations, such that $C \subseteq \{0, 1, \ldots, K\}$. As before, given cancer sample s, we have population frequencies $[\eta_{0s}, \eta_{1s}, \ldots, \eta_{Ks}] \sim \text{Dirichlet}(\alpha, \alpha, \ldots, \alpha)$ (Section 6.4.2), and $\phi_{Cs} = \sum_{i \in C} \eta_{is}$. By the properties of the Dirichlet distribution, we know that the sum of Dirichlet-distributed variables is itself Dirichlet-distributed, such that

$$\left[\sum_{i\in C}\eta_{is}, \eta_{(|C|+1)s}, \ldots, \eta_{Ks}\right] \sim \text{Dirichlet}(|C|\alpha, \alpha, \ldots, \alpha) \ ,$$

where the first element of the vector represents the subclonal frequency $\sum_{i\in C}\eta_{is} = \phi_{Cs}$, and the final $K - |C|$ elements represent the population frequencies of all populations not in subclone C. From this, we get

$$\text{var}(\phi_{Cs}) = \frac{\frac{|C|}{K}(1-\frac{C}{K})}{K\alpha+1} \ .$$

From the denominator, we see that variance is reduced either with more populations K, or with a larger Dirichlet parameter α. By plotting both the (theoretical) population standard deviation and (empirical) sample standard deviation (Fig. S12B), we see that the latter conforms to the former, and that variance is maximized for subclones with $\frac{K}{2}$ populations, conferring the greatest benefit from multiple cancer samples to populations near the root of the tree, such that they have half the populations as descendants. Conversely, subclones with less variance in frequency across samples will either be at the very top of the tree, with almost all populations as descendants, or at the bottom of the tree, with few populations as descendants. Note that, in Fig. S12, the sample standard deviation appears less than the population standard deviation, particularly in the three- and ten-subclone cases. This effect is exaggerated for those settings because they include single-sample datasets with zero sample standard deviation, whereas the 30- and 100-subclone datasets do not.

10.7.3 Simulated data often include subclones that are impossible to resolve

If a population k has a near-zero population frequency η_{ks} across all cancer samples s, its position in a clone tree relative to its parent j is difficult or impossible to resolve. Since k's subclonal frequency ϕ_{ks} is equal to the sum of the population frequencies of all populations in the subclone, when $\eta_{ks} \approx 0$, we have $\phi_{ks} \approx \phi_{js}$. When this occurs, we will have two candidate trees that fit the data equally well—one in which k is the parent of j, and one in which j is the parent of k. Both tree structures would permit tree-constrained subclonal frequencies that fit the observed VAF data almost equally well. Well-behaved algorithms should find both tree structures. Thus, populations whose frequencies are negligible across all cancer samples lead to their subclonal frequencies being nearly equal across all cancer samples, which

leads to ambiguity. In real data, we are unlikely to be faced with this situation. The observed VAFs for two variants serve as noisy estimates of their subclones' subclonal frequencies. When the observation noise exceeds the negligible differences in the subclonal frequencies, we will deem the two variants as having originated from the same subclone, such that the variants are placed in a single cluster.

Nevertheless, examining how often this situation occurs in simulated data is worthwhile, as it grants insight into how well algorithms deal with ambiguity. Note that noisy observations of near-zero population frequencies are not the only source of ambiguity—ambiguity can exist even given noise-free frequencies, or with large population frequencies. All cases where tree enumeration using the noise-free subclonal frequencies found multiple trees (Section 6.5.4) are demonstrations of this alternative ambiguity. Tree-reconstruction algorithms should be able to deal with both sources of ambiguity by finding the full range of solutions permitted for a dataset. With respect to our evaluation metrics, VAF loss (Section 3.4) does not capture algorithms' performance in this respect, since it penalizes discrepancies between VAFs and tree-constrained subclonal frequencies, and so algorithms can do well regardless of whether they find a single good solution or multiple equivalent solutions. Relationship reconstruction error (Section 3.4), however, properly reflects algorithms' performance in the face of ambiguity—in the example above in which subclones j and k had nearly equal subclonal frequencies across all cancer samples, the solutions recovered by a tree-reconstruction algorithm should show both that k could be an ancestor of j, and j could be an ancestor of k.

To understand the role near-zero population frequencies play in introducing ambiguity, we must first define a threshold ϵ on population frequencies, such that we will say a population frequency η is near-zero if $\eta < \epsilon$. This ϵ should ideally be defined as a function of read depth, since depth determines how precisely the observed VAFs reflect the underlying subclonal frequencies, and ultimately how small population frequencies can get before they are swamped by noise. To set this threshold, consider a fixed read depth of $D = 200$, such that with V variant reads and R reference reads we have $D = V + R = 200$. By our simulation framework, we have $V \sim \text{Binom}(D, \omega\phi)$, yielding $[E](V) = \omega\phi D$. We will define a non-negligible population frequency as that which produces a difference of one read in the mean read counts. While this is a subtle difference, we must remember that, in tree search, the read counts for all variants belonging to a cluster will be summed, exaggerating the difference in observations for the two clusters. Thus, for populations j and k, we will assume we have subclonal frequencies ϕ_j and ϕ_k with $\phi_j > \phi_k$. Moreover, assume j is the parent of k, such that $\phi_j = \phi_k + \eta_j$. This gives us

$$\begin{aligned}\omega\phi_j D - \omega\phi_k D &\geq 1\\ \phi_j - \phi_k &\geq \frac{1}{\omega D}\\ \eta_j &\geq \frac{1}{\omega D}\end{aligned}$$

With $\omega = \frac{1}{2}$, this results in a non-negligible population frequency of $\eta_j \geq 0.01$ for read depth $D = 200$. Conversely, we will define a near-zero population frequency as the complement of this, resulting in a threshold $\epsilon = 0.01$. To simplify the analysis, we will use this threshold regardless of read depth. With read depths $D \in \{50, 200, 1000\}$ (Section 6.4.2), this choice of ϵ will yield a greater difference in binomial mean for $D = 1000$, and a smaller difference for $D = 50$. Nevertheless, the conclusions we reach for fixed ϵ will be broadly applicable regardless of read depth.

First, we will consider how many populations within each simulated dataset have population frequencies less than $\epsilon = 0.01$ across all cancer samples s. Let η_{ks} denote the population frequency of population k in cancer sample s. For K subpopulations, we have $[\eta_{0s}, \eta_{1s}, \ldots, \eta_{Ks}] \sim \text{Dirichlet}(\alpha, \alpha, \ldots, \alpha)$. By the properties of the Dirichlet distribution, we have

$$\begin{aligned}\eta_{ks} &\sim \text{Beta}(\alpha_{ks}, \sum_{j=0}^{K} \mathbb{1}_{j\neq k}\alpha_j s)\\ &= \text{Beta}(\alpha, K\alpha)\ .\end{aligned}$$

Consequently, we since each cancer sample's population frequencies are independent of every other, for S cancer samples we get

$$\begin{aligned}p(\eta_{k1} < \epsilon, \ldots, \eta_{kS} < \epsilon) &= \prod_{s=1}^{S} p(\eta_{ks} < 0.01)\\ &= \prod_{s=1}^{S} \int_0^{\epsilon} dx p(\eta_{ks} = x)\\ &= \prod_{s=1}^{S} \frac{\beta(\epsilon|\alpha, K\alpha)}{\beta(\alpha, K\alpha)}\\ &= \left[\frac{\beta(\epsilon|\alpha, K\alpha)}{\beta(\alpha, K\alpha)}\right]^S\end{aligned} \tag{34}$$

Here, $\beta(\epsilon|\alpha, K\alpha)$ refers to the incomplete beta function, and $\beta(\alpha, K\alpha)$ refers to the complete beta function. Empirically, the proportion of simulated populations with near-zero population frequencies across samples agrees with the result predicted above (Fig. S13). Datasets with 30 or 100 populations and one or three cancer samples would have at least 38% of populations with near-zero population frequencies in all cancer samples, rendering their positions in the tree difficult to resolve. This would create excessive ambiguity, which is why we did not include such datasets in our simulated data.

The relationship reconstruction error we used to evaluate method performance on simulated data reflected how algorithms dealt with two sources of ambiguity: firstly, the multiple tree structures potentially permitted by the noise-free frequencies (Section 10.6); and, secondly, the additional tree structures permitted by populations with near-zero population frequencies. As we established above, if a population k has near-zero population frequencies across all cancer samples, the subclonal frequencies of k and its true parent j will be almost equal, such that the noisy VAF observations will render difficult the task of determining whether j is the parent of k or vice versa. Observe that 14% of populations in 100-subclone, 10-sample trees have noise-free population frequencies less than $\epsilon = 0.01$ across cancer samples. In the average tree, these would correspond to 14 populations with near-zero frequencies. Since each such population could be swapped with its parent while minimally affecting tree likelihood, these would generate $2^{14} \approx 16,000$ additional trees. This assumes that none of the populations with near-zero frequencies have edges between them; chains of two or more populations with near-zero frequencies would further increase the number of potential tree configurations. We expect noisy observations to be the dominant source of ambiguity. In the 100-subclone, 10-sample setting, none of the 36 simulated datasets permitted more than 42 trees given the noise-free frequencies (Fig. S10), which is a value far smaller than the 16,000 trees we expect to be permitted by the noisy observations.

This analysis also helps us understand how many cancer samples we must simulate to remove ambiguity in tree search arising from noisy observations for a given number of subclones. Taking our threshold $\epsilon = 0.01$, we can ask how many cancer samples we need before $p(\eta_{k1} < \epsilon, \ldots, \eta_{kS} < \epsilon)$. By solving for S in Eq. (34), we find that need 24 or more samples before the probability of a population frequency being less than ϵ across all samples falls below 1%. This has implications for variant clustering as well, since a population's variants become distinguishable from other variants by the clustering algorithm only when one or more cancer samples with non-negligible frequencies for the associated population render the VAFs clearly distinct.

To complement the above analysis concerning lone populations, we will also examine the probability of simulated trees containing sub-trees that consist entirely of populations whose frequencies are less than

$\epsilon = 0.01$. We define a sub-tree to consist of a subset of the full tree's nodes, as well as all edges between them, ensuring the sub-tree is connected. Thus, a sub-tree can correspond to a subclone (Section 3.1), but is more general in that may omit parts of the subclone defined by the ancestral population at the root of the sub-tree. For this analysis, we did not conduct an empirical examination of the simulated data, but used only theoretical results derived from the Dirichlet distribution properties. Given a complete tree composed of K populations as well as the root node 0, and a sub-tree composed of populations $T \subseteq \{0, 1, \ldots, K\}$ with size $|T|$, we have in cancer sample s the result

$$\begin{aligned}\sum_{i \in T} \eta_{is} &\sim \text{Dirichlet}(\sum_{i \in T} \alpha_i, \sum_{j \notin T} \alpha_j) \\ &= \text{Dirichlet}(|T|\alpha, (K - |T| + 1)\alpha)\end{aligned}$$

Note that if the sub-tree $T = \{j\} \cup \{k | k \text{ is descendent of } j\}$, then T is equivalent to the subclone with population j at its head, and $\sum_{i \in T} \eta_{is} = \phi_{js}$. By using the Dirichlet's marginal beta distribution, as in the previous analysis, we can compute the probability of the arbitrary sub-tree T consisting exclusively of populations whose summed frequencies across cancer samples are small, such that $\sum_{i \in T} \eta_{is} < \epsilon = 0.01$ for every cancer sample s (Fig. S14). For instance, in the 100-subclone, single-sample case, we have a 6% probability of an arbitrary eleven-population sub-tree having a near-zero population frequency sum. With $|T|$ populations in such a sub-tree, there are $(T + 1)!$ orderings of nodes in the sub-tree that would permit nearly equal tree-constrained subclonal frequencies, and thus nearly equal tree likelihood. In the eleven-population case, there would thus be $(11 + 1)! = 4.79e8$ solution trees resulting from this single ambiguous sub-tree.

To compute the probability of observing such a case in the simulated trees, we must first consider how many linear chains of J populations exist in a tree with K nodes, as each has an equal chance of being assigned these small frequencies. If a tree is fully linear with no branching, there would be $(K+1)-J+1$ chains of J nodes, such that our chain of 11 populations in a 100-subclone tree would have $101-11+1 = 91$ sub-trees, assuming that tree was fully linear. This in turn yields a $(100\% - 6\%)^{91} = 0.36\%$ chance that we would not observe any near-zero-frequency 11-population chains in our tree—i.e., with near certainty, we would encounter such a chain. Any degree of branching in a tree can reduce the number of node chains of a given length, thereby lessening the chance we would see this scenario. Nevertheless, the probability can remain considerable, which is another reason we omitted the many-subclones, few-samples cases from our simulated data. Amongst the settings we included, we see, for instance, that in ten-subclone, single-

sample trees, 6% of five-population chains will have small population frequency sums, yielding a 35% chance that we would encounter such a case in a fully linear tree.

10.7.4 Justifying our choice of the Dirichlet parameter for generating simulated data

In Sections 10.7.1 to 10.7.3, we saw that our choice of the Dirichlet parameter α when generating simulated data (Section 6.4.2) affects multiple aspects of simulated data.

1. A smaller α leads to more variance in population frequencies between samples, increasing the chance that multiple samples will make clear the proper pairwise relations between subclones.

2. A smaller α also leads, however, to a greater probability of observing near-zero frequencies for a population across all cancer samples, inhibiting tree-reconstruction algorithms' attempts to infer the proper place for such populations in the tree. (We do not present results with alternative α values here, but used these analyses to inform our choice of α.)

Our chosen $\alpha = 0.1$ thus achieved a compromise between three factors.

1. It led to sufficient variance in population frequencies between cancer samples for algorithms to benefit from having access to multiple cancer samples.

2. It avoided creating too many populations with near-zero frequencies across samples, which would have created excessive ambiguity.

3. Yet it created enough such populations so that we could evaluate how algorithms dealt with ambiguity stemming from this source.

10.8 Impact of the infinite sites assumption

To simplify subclonal reconstruction, algorithms make the ISA, which posits that the genome is so large as to be effectively infinite in size, meaning that each genomic site is mutated at most once during the cancer's evolution. This implies that the same site can never be mutated twice by separate events, and that it can never return to the wildtype. Moreover, two cells bearing the same mutation are assumed to share a common ancestor in which that mutation occurred. Most clone tree reconstruction algorithms make this assumption. Equivalently, ISA violations can be understood as violations of the four-gamete test [38]. Under this assumption, the cancer phylogeny is a *perfect phylogeny*, such that descendant subclones inherit all the mutations of their ancestors. Critically, the ISA allows us to characterize more subclones

than we have cancer samples. In addition, the ISA is necessary to infer the pairwise relationships between mutations from their frequencies (Section 6.1).

Given complete genomes for each cancer cell, a perfect phylogeny can be constructed in linear time [43], with mutations that deviate from the ISA detected via the four-gamete test [38]. However, the bulk-tissue DNA sequencing data commonly used today do not provide complete genomes. Instead, the samples consist of mixtures of different subclones, rendering NP-complete the construction of a perfect phylogeny consistent with the exact subclonal frequencies of mutations across multiple samples [44]. Nevertheless, the ISA implies relationships between mutation frequencies that can assist subclonal reconstruction. Firstly, mutations in ancestral subclones must always have subclonal frequencies at least as high as those in descendent subclones, across every observed cancer sample. Secondly, two mutations on different tree branches can never have frequencies that sum to greater than one in any sample.

Pairtree can often detect such violations and discard the offending mutations using its garbage relation (Section 6.1.3). Specifically, Pairtree's pairwise-relation-based mutation clustering algorithm (Section 10.1.3) could be trivially modified to use this information to temporarily remove mutations violating the ISA. After building a clone tree using all other mutations, the ISA-violating mutations could be layered over the tree using a separate inference step. These extensions would also be relevant to scDNA-seq settings (Section 10.9).

10.9 Using single-cell DNA sequencing data for building clone trees

Single-cell DNA sequencing (scDNA-seq) is becoming more popular for studying cancer evolution [45, 46]. In principle, scDNA-seq gives unambiguous knowledge of each cancer cell's genotype, avoiding the need to deconvolve the signal from many cell subpopulations that is inherent to bulk sequencing. However, scDNA-seq data is noisy, with amplification biases giving rise to inaccurate estimates of mutation prevalence [47]. The same issues result in many mutations being missed altogether. As a result, bulk sequencing will likely remain widely used for many years, including in initial clinical applications of clone trees—bulk data gives a more complete depiction of a cancer's mutation spectrum, and better estimates of mutation prevalence.

Nevertheless, scDNA-seq is likely to grow in popularity in the coming years. Pairtree can be extended to construct clone trees from single-cell DNA sequencing (scDNA-seq) data. This can be accomplished by modifying Pairtree's pairwise relation framework to use binary valued information about the presence or absence of mutations, rather than the mutation's estimated subclonal frequencies. This would allow trees to be built from mixtures of scDNA and bulk data, or from scDNA data alone [17]. Tree search would

remain mostly unchanged, with modifications required only in defining a likelihood that incorporates single-cell information.

We have demonstrated that Pairtree can accurately recover clone trees with more subclones than cancer samples by deconvolving bulk samples. This suggests the potential for using Pairtree with quasi-bulk data, whereby single cells would be pooled together to reduce sequencing costs, then deconvolved post-hoc using techniques inspired by compressed sensing. This deconvolution ability could also be useful in detecting and resolving cell doublets.

11 Supplementary figures

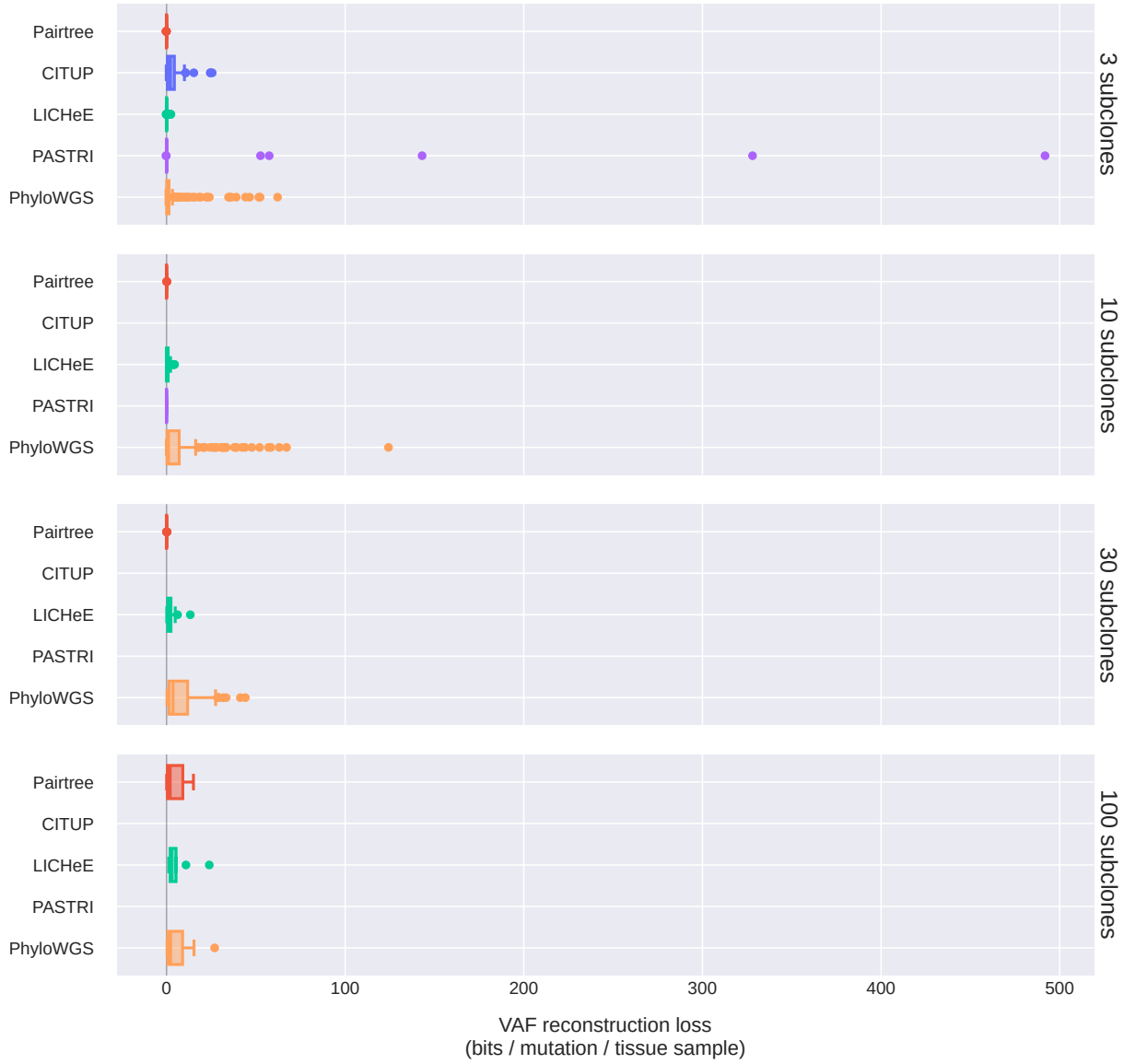

Figure S1: **Untruncated VAF reconstruction losses on 576 simulated datasets**. These results are the same as in Fig. 3b, but without axis truncation. As in the truncated plots, results reflect each method's performance on the subset of datasets where it succeeded in running.

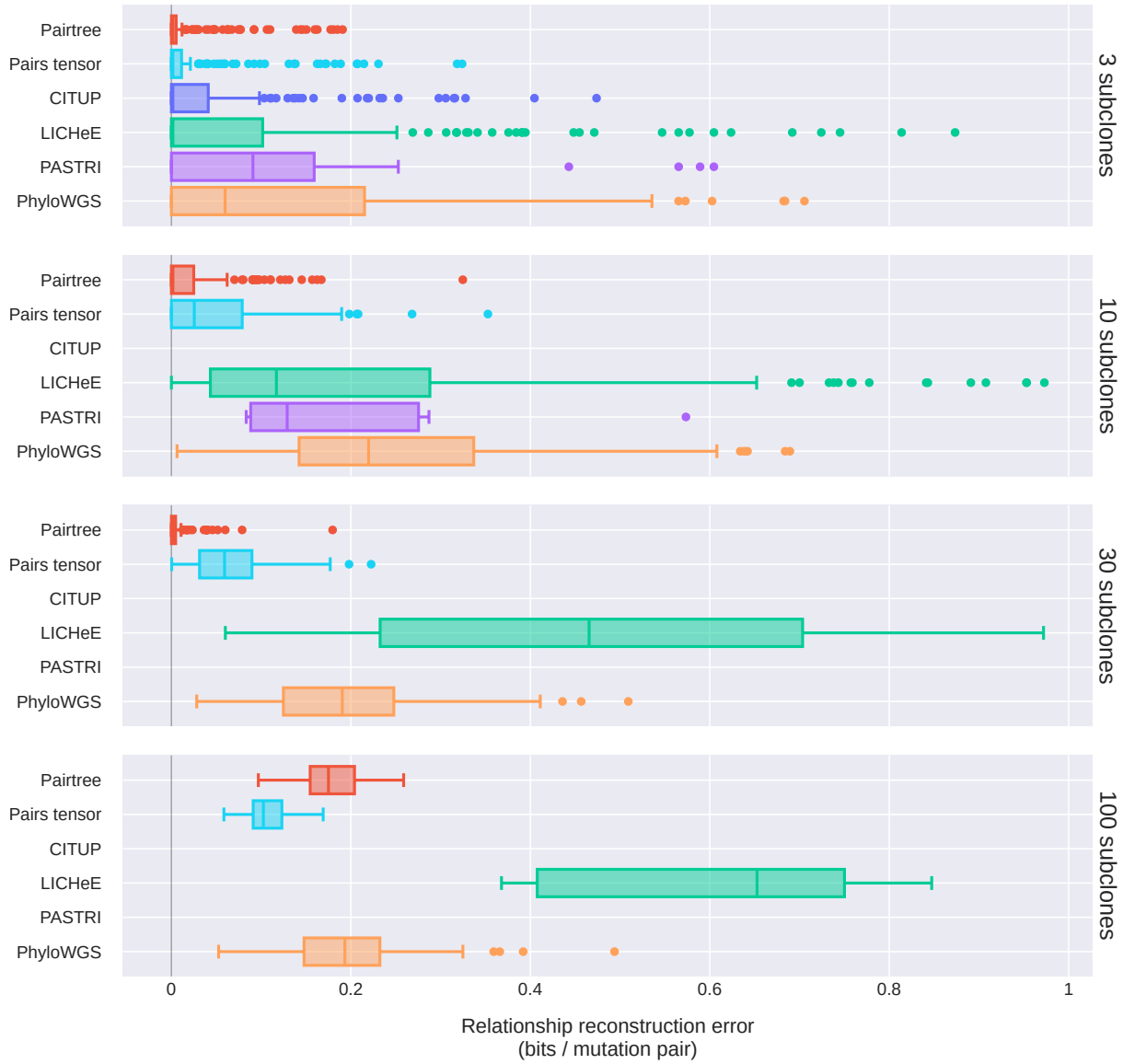

Figure S2: **Untruncated relationship reconstruction errors on 576 simulated datasets**. These results are the same as in Fig. 3c, but without axis truncation. As in the truncated plots, results reflect each method's performance on the subset of datasets where it succeeded in running.

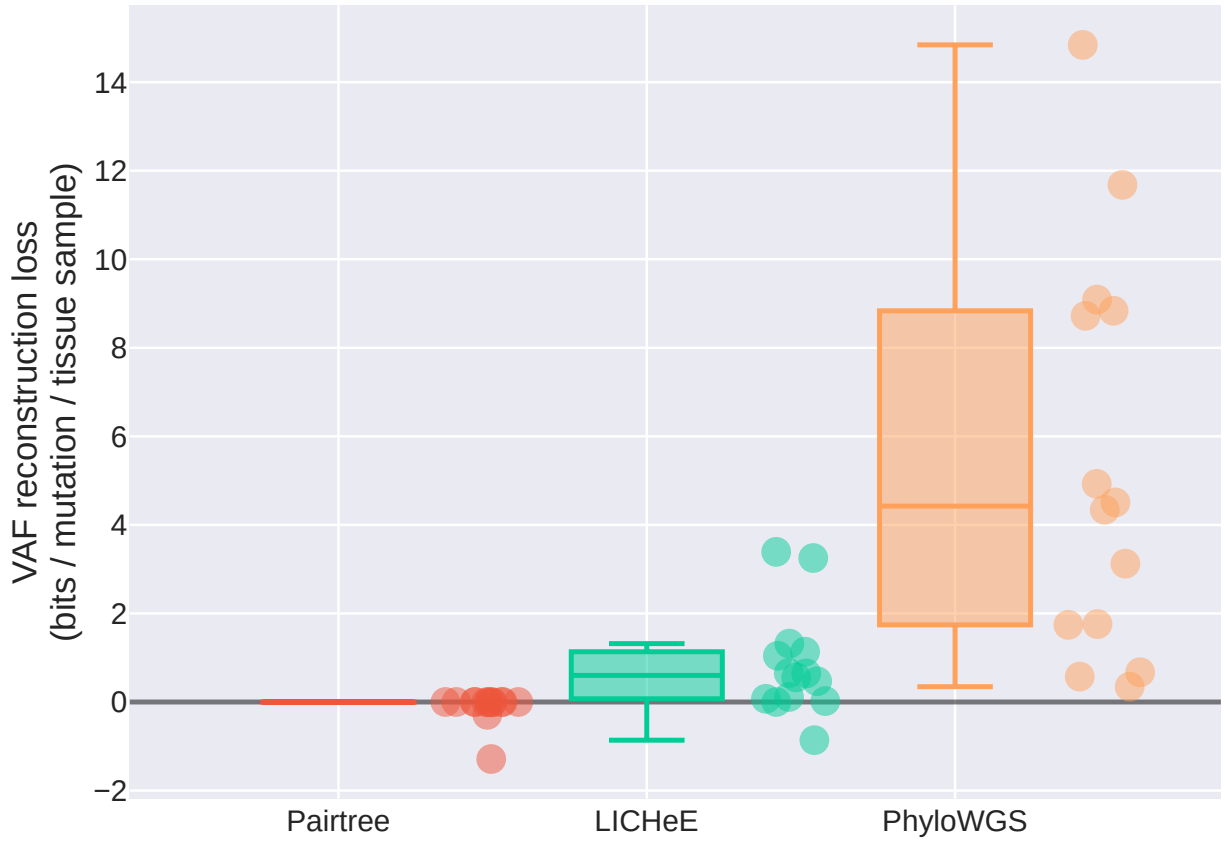

Figure S3: **Untruncated VAF reconstruction losses on 14 B-ALL datasets**. These results are the same as in Fig. 5, but without axis truncation. As in the truncated plots, results reflect each method's performance on the subset of datasets where it succeeded in running.

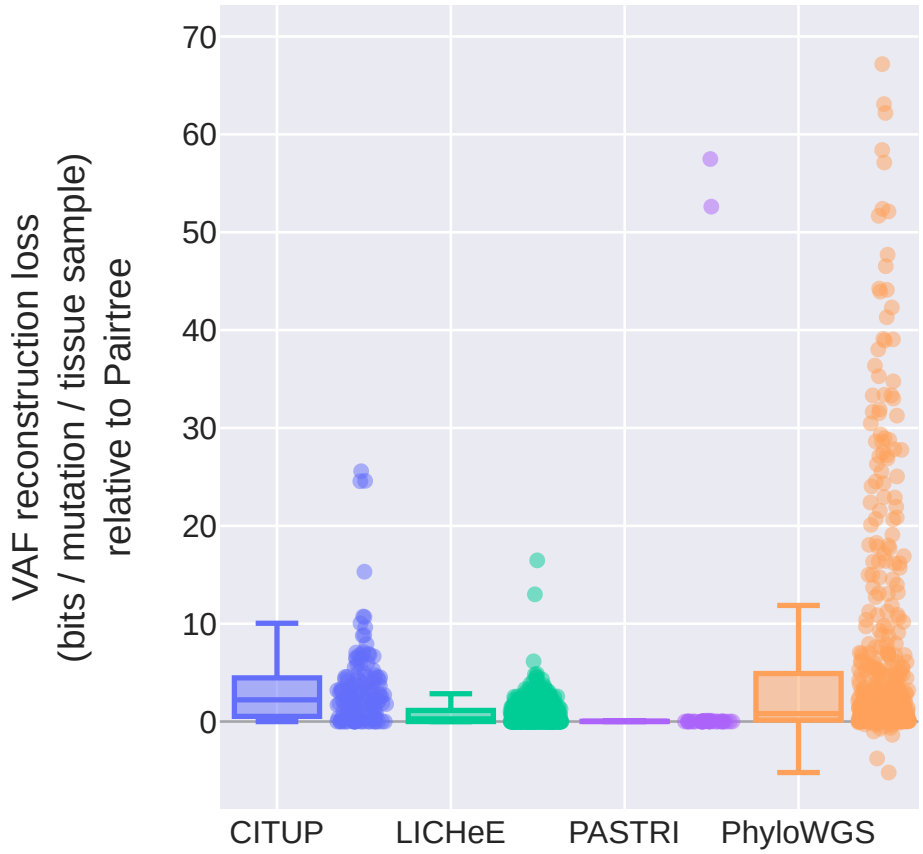

Figure S4: **VAF reconstruction loss of each method relative to Pairtree.** Each point represents a method's VAF reconstruction loss on a simulated dataset relative to Pairtree, with positive values indicating worse error. As each method failed on different simulations (Fig. 3a), values are reported only on datasets where a method produced a result.

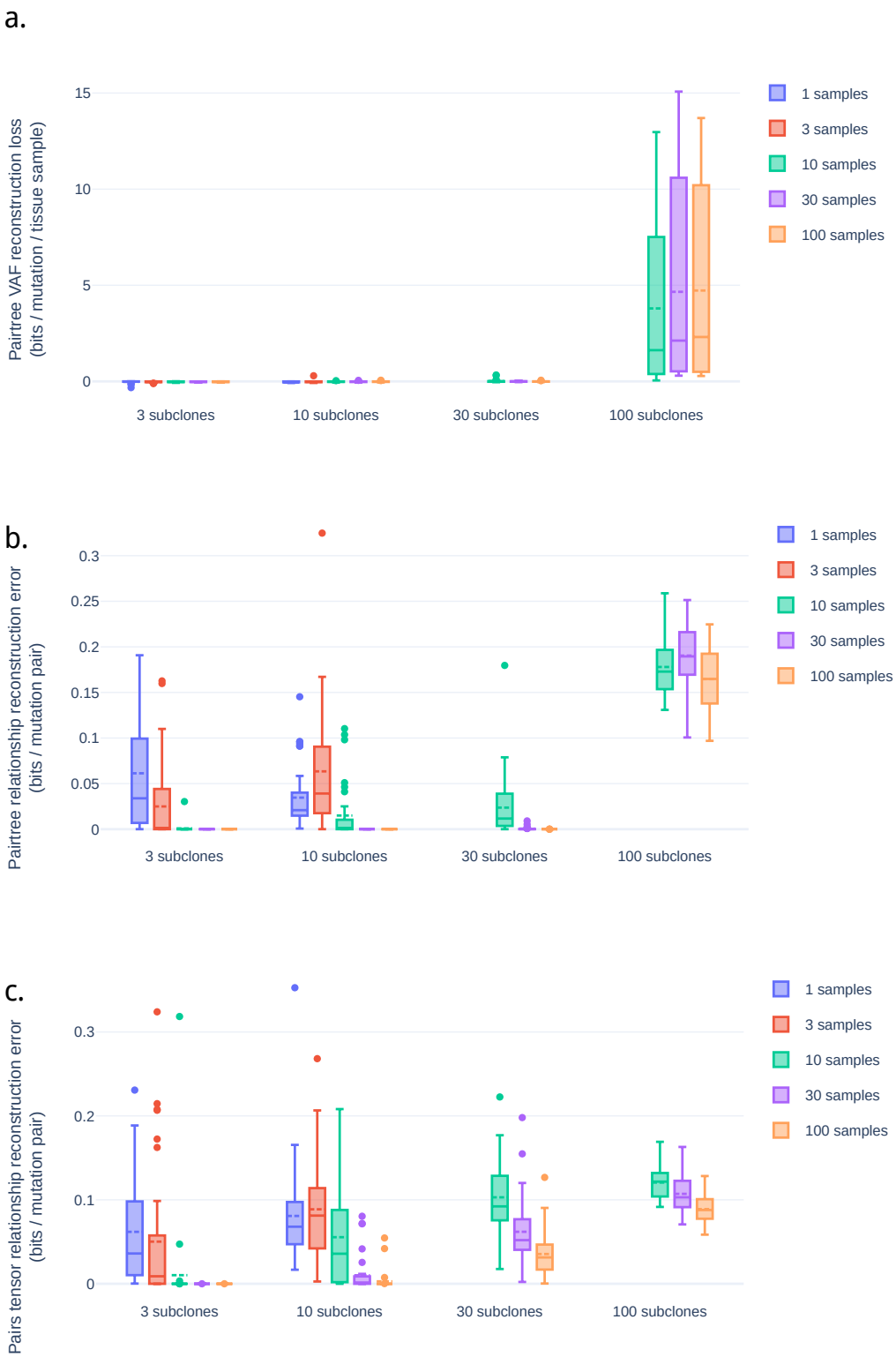

Figure S5: **Pairtree's performance on different numbers of subclones and cancer samples.** **a.** Pairtree's VAF reconstruction loss for each number of subclones and number of cancer samples. **b.** Pairtree's relationship reconstruction error for each number of subclones and number of cancer samples. **c.** Pairtree's Pairs Tensor's relationship reconstruction error for each number of subclones and number of cancer samples.

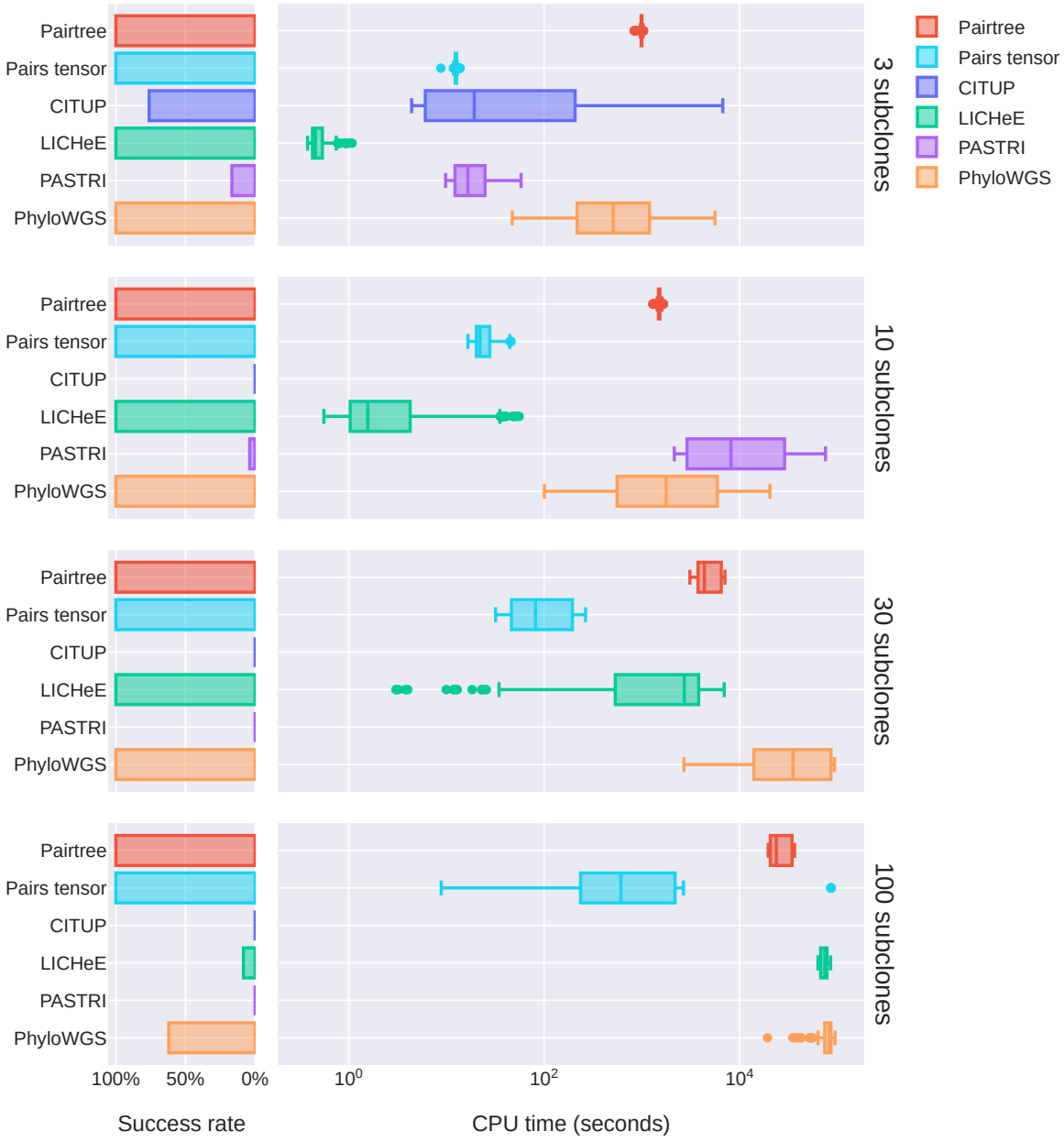

Figure S6: **Number of CPU seconds methods took to produce results.** Box mid-lines indicate medians. When using multiple CPU cores, these numbers can be much higher than elapsed wall-clock time (Fig. S7). Results for each method reflect only its performance on the datasets where it could produce a result.

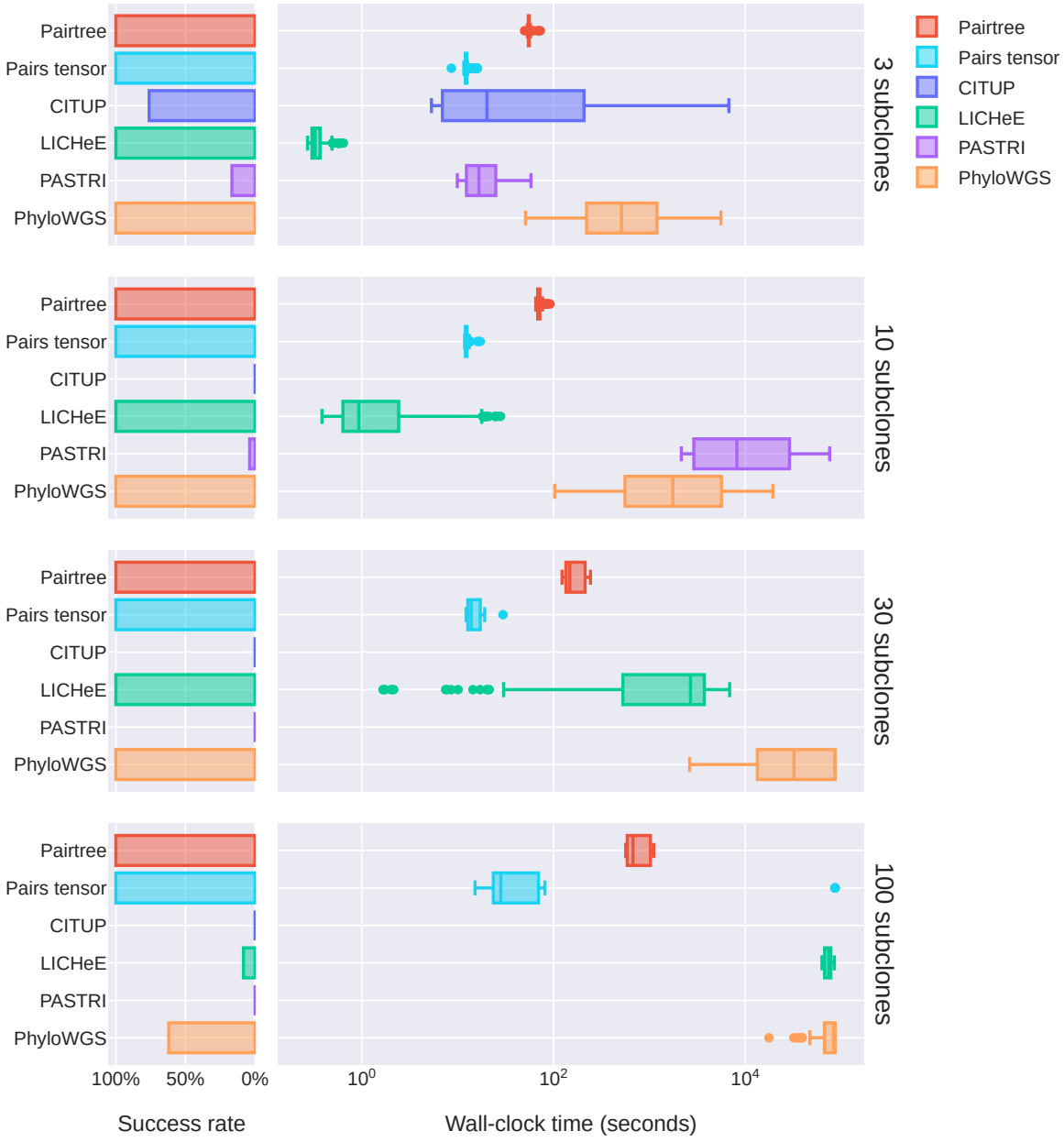

Figure S7: **Elapsed wall-clock seconds methods took to produce results.** Box mid-lines indicate medians. When using multiple CPU cores, these numbers can be much lower than the number of CPU seconds consumed (Fig. S6). Results for each method reflect only its performance on the datasets where it could produce a result.

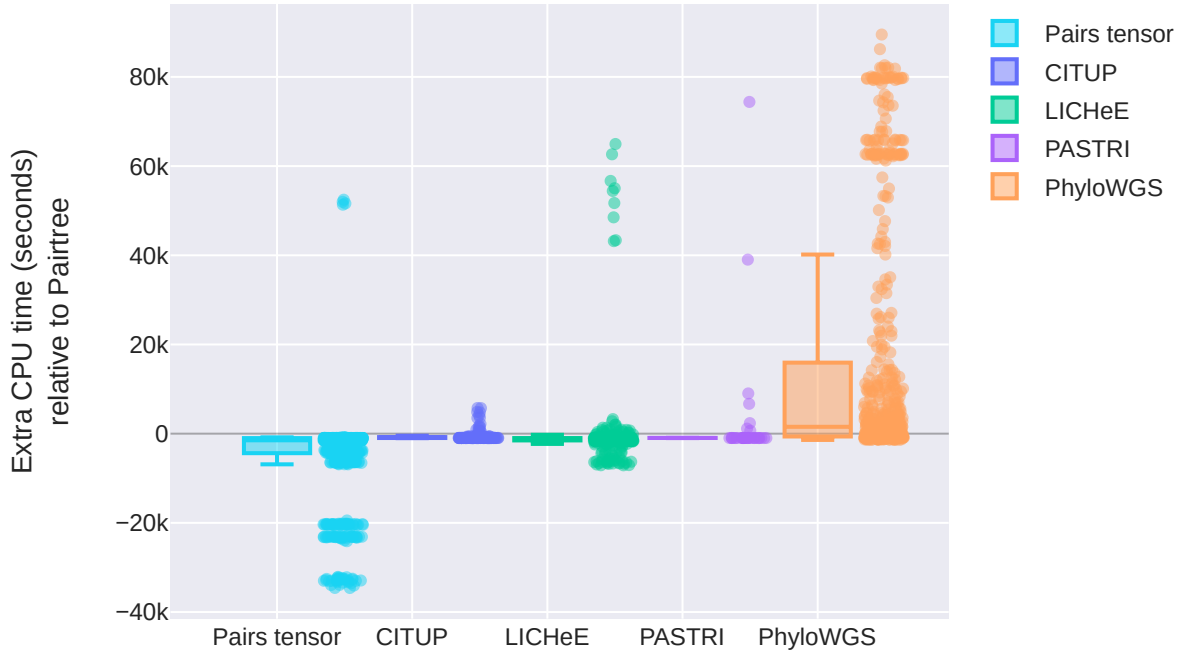

Figure S8: **Number of CPU seconds each method took to produce results relative to Pairtree.** Each point indicates the number of additional CPU seconds a method took on a dataset relative to Pairtree on that dataset. Points below zero indicate a method took less time than Pairtree on those datasets.

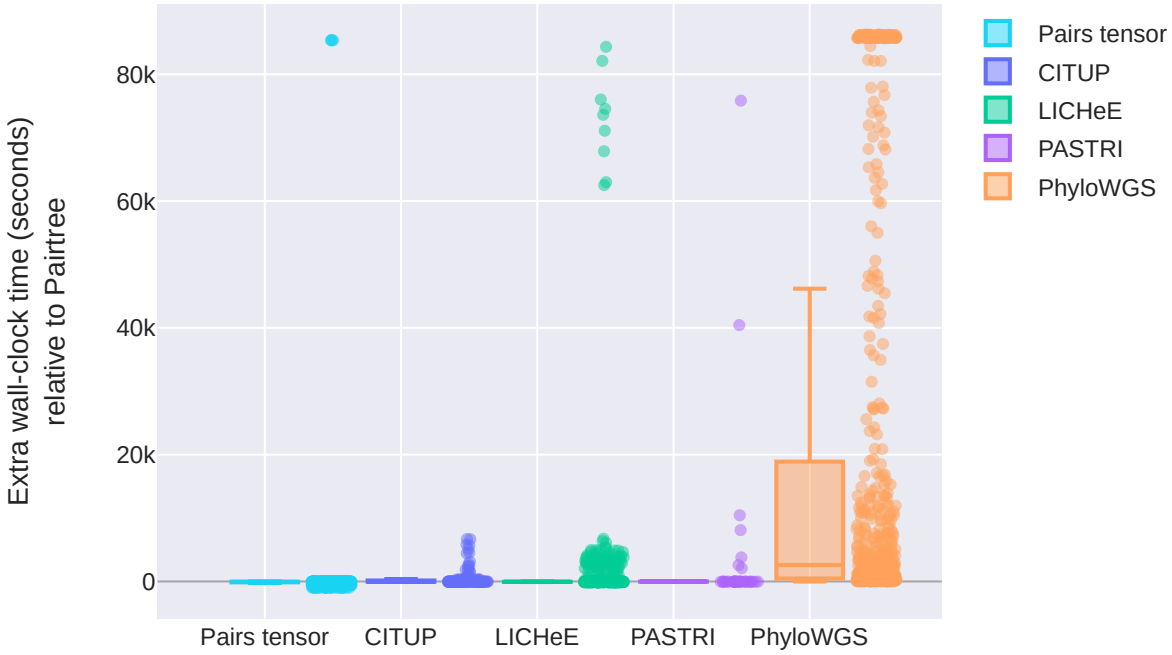

Figure S9: **Elapsed wall-clock seconds each method took to produce results relative to Pairtree.** Each point indicates the number of additional wall-clock seconds a method took on a dataset relative to Pairtree on that dataset. Points below zero indicate a method took less time than Pairtree on those datasets.

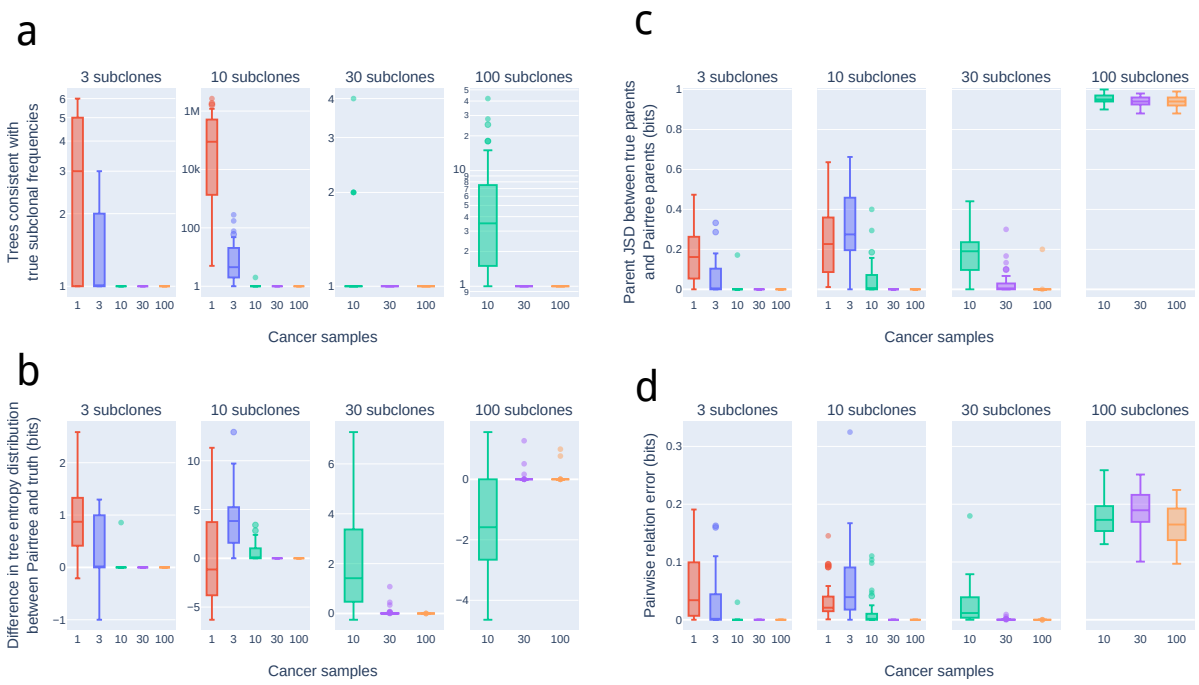

Figure S10: **Characteristics of the distributions over possible trees for the 576 simulated clone tree reconstruction problems.** Mid-lines in box plots indicate medians. **a.** Regardless of the number of subclones, with one cancer sample there are usually multiple trees consistent with the true subclonal frequencies. The highest median number of true trees (88,860) is reached for 10-subclone, single-sample reconstructions problems. Given ten or more samples, the tree becomes highly constrained, and there is usually only a single consistent tree. **b.** The entropies of the Pairtree-recovered tree distribution and true tree distribution reflect how many high-confidence trees Pairtree recovers relative to the number of possible trees. In general, Pairtree recognizes when the true tree is highly constrained, and returns only one high-confidence tree. **c.** For a simulated dataset, a distribution over possible trees induces a distribution over parent choice for every population represented in the tree. Shown are the joint Jensen-Shannon divergence between parent distributions for Pairtree relative to truth for each simulated dataset, normalized to the number of subclones in each tree. These divergences range between zero and one, with small values indicating that parent choices are nearly always correct. For a given number of subclones, Pairtree generally exhibits lower divergences with more cancer samples, indicating it was able to use the information provided by those samples to improve its solution set. **d.** Relationship reconstruction errors show that, even when the parents chosen for subclones are sometimes incorrect (panel c), the relationship reconstructions can be more accurate. This is the same information as presented in Fig. 3b, but partitioned by number of cancer samples.

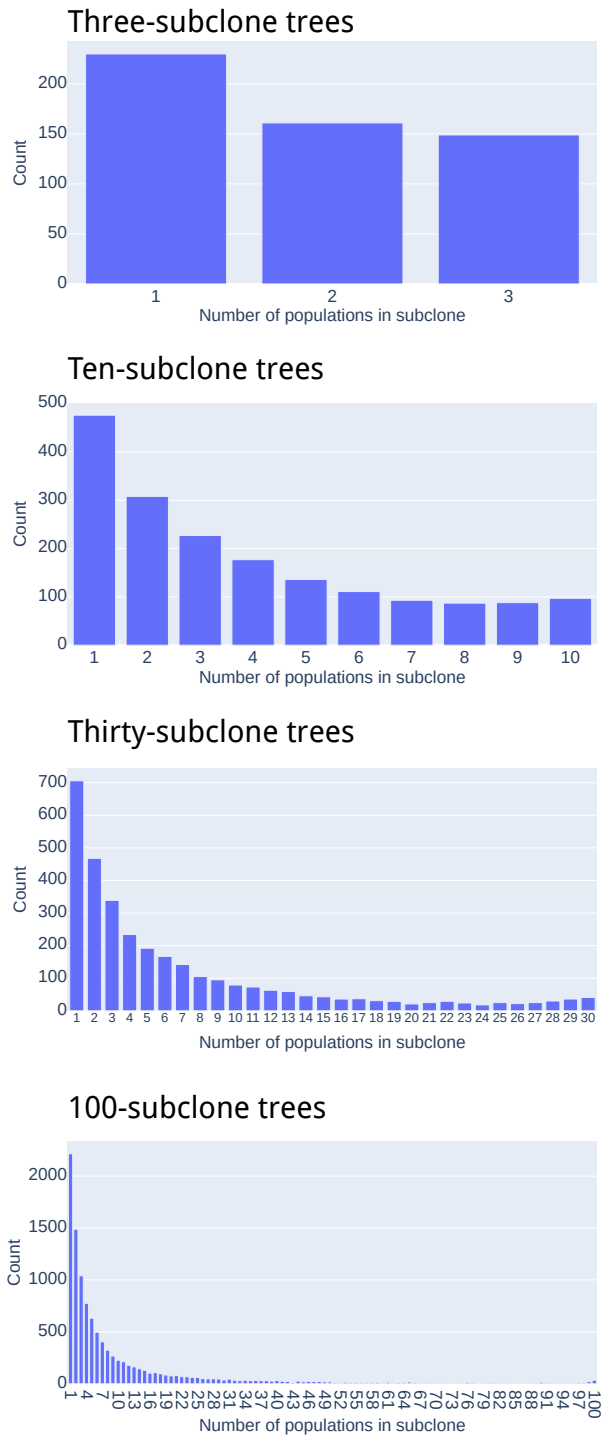

Figure S11: **Prevalence of different subclone sizes within simulated trees.** Subclone size indicates the number of subpopulations present within a subclone, reflecting the number of subpopulations that are descendants of the subpopulation that initiated the subclone.

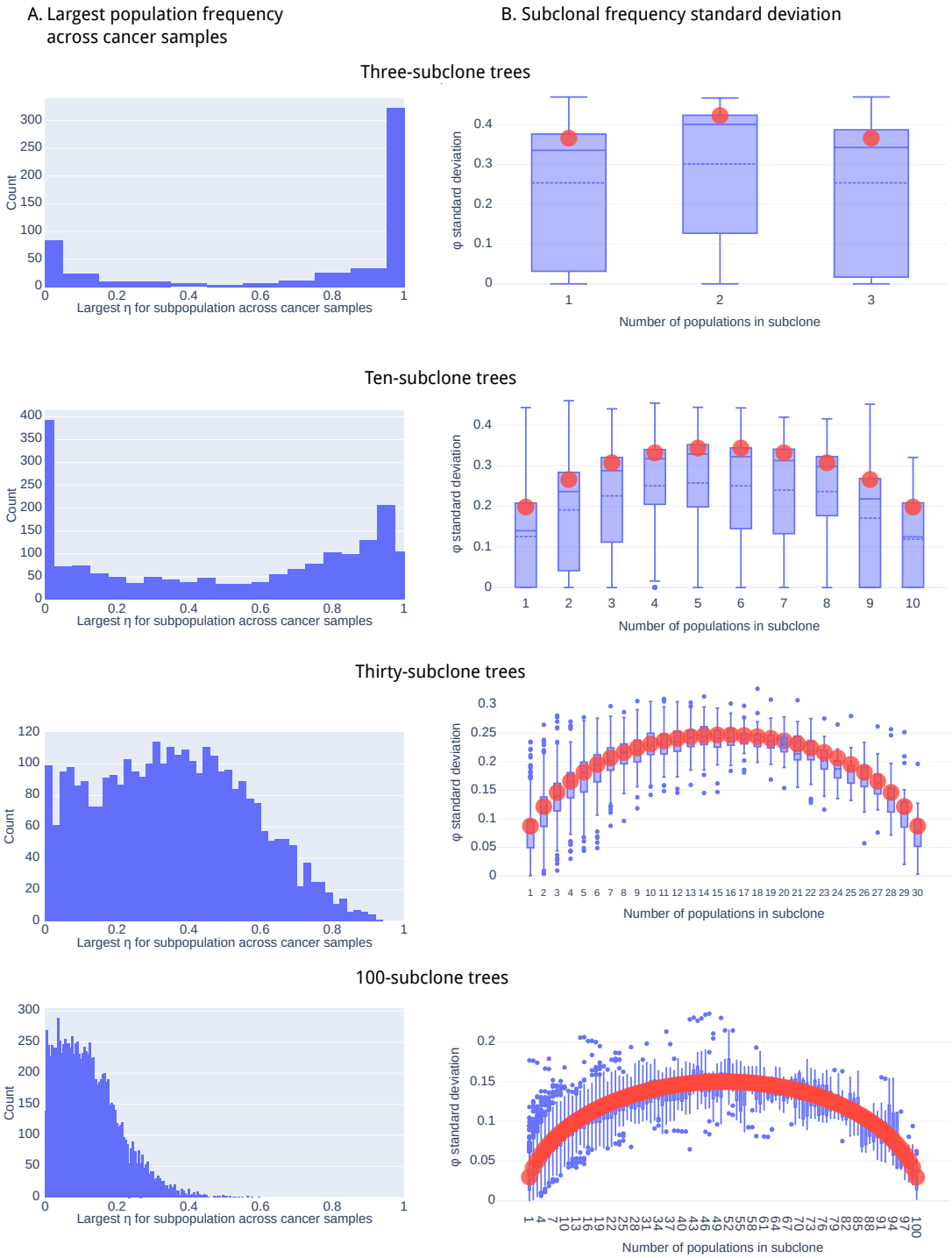

Figure S12: **Properties of population and subclone frequencies. a.** Largest population frequency η_{ks} for each population k across cancer samples s in simulated data. **b.** Standard deviation of subclonal frequencies ϕ_{ks} for each subclone k across cancer samples s in simulated data, as a function of the number of populations in the subclone. Box plots show the empirical standard deviation measured in (noise-free) simulated data, with solid line indicating the median and dashed line showing the mean. Orange circles show the predicted standard deviation derived from Dirichlet distribution properties.

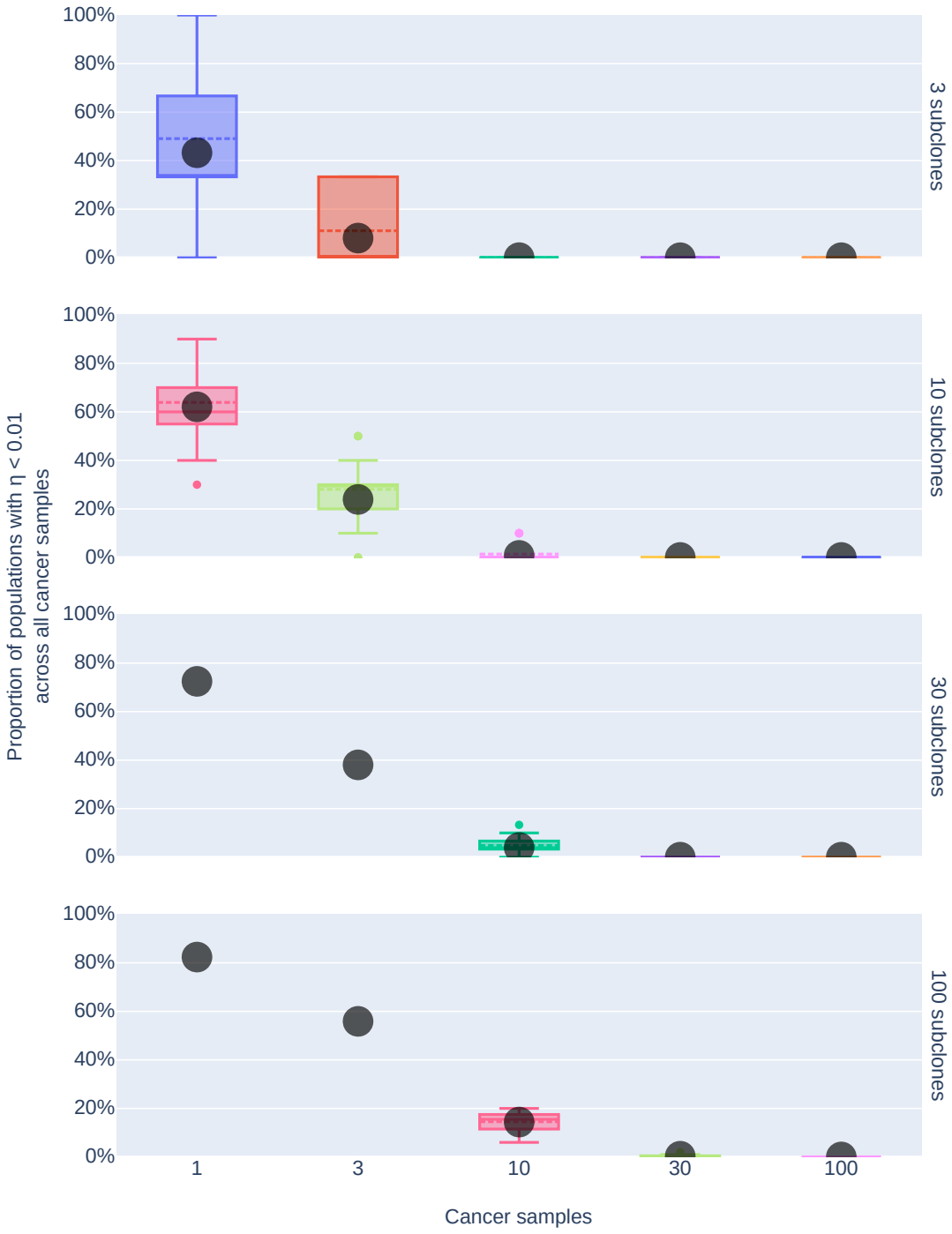

Figure S13: **Proportion of populations with small population frequencies in all cancer samples.** Proportion of populations k with population frequencies $\eta_{ks} < 1\%$ across all cancer samples s. Box plots show the empirical proportions measured in (noise-free) simulated data, with solid line indicating the median and dashed line showing the mean. Grey circles show the predicted proportions derived from Dirichlet distribution properties.

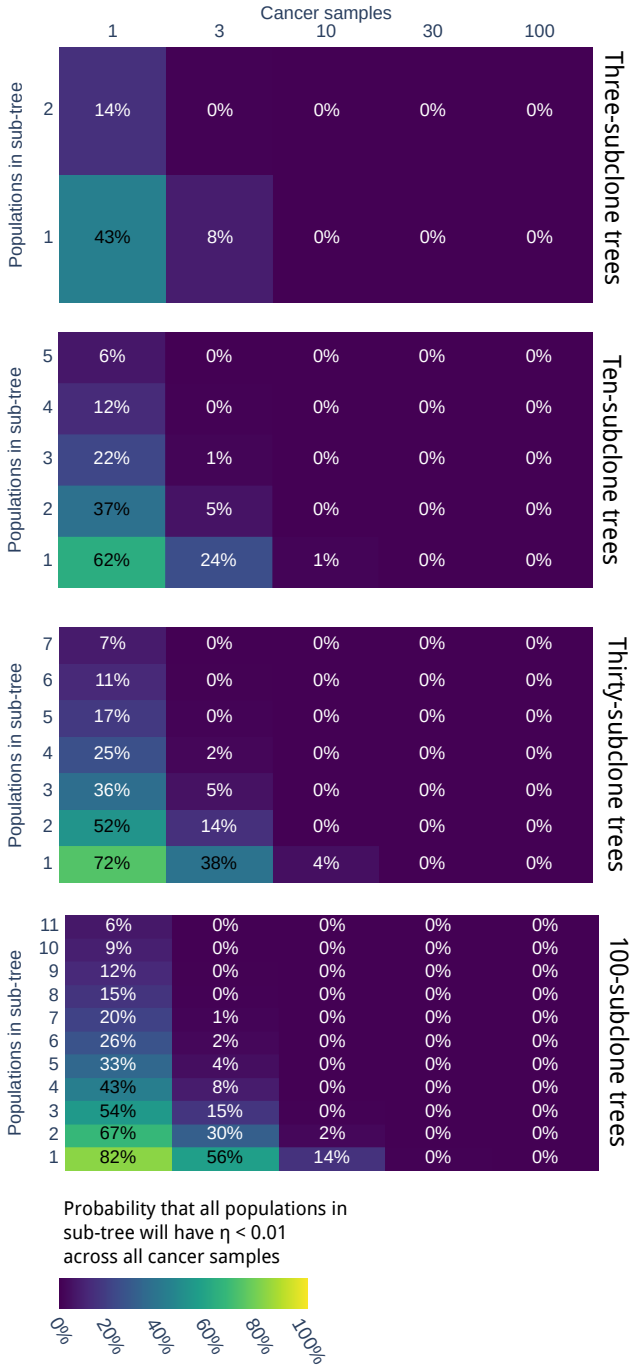

Figure S14: **Probability that sub-trees will consist entirely of populations with small frequencies in all cancer samples.** Probability that sub-tree containing given number of populations will have population frequencies $\eta_{ks} < 1\%$ for all populations k in the sub-tree across all cancer samples s, computed using properties of Dirichlet distribution. A sub-tree consists of a subset of nodes from the full-tree and all edges between those nodes. By this definition, all subclones are sub-trees, but a sub-tree need not be a subclone.

References

1. Dentro, S. C. *et al.* Pervasive intra-tumour heterogeneity and subclonal selection across cancer types. *Accepted at Cell* (2021).

2. Hanahan, D. & Weinberg, R. a. Hallmarks of cancer: the next generation. *Cell* **144,** 646–74. ISSN: 1097-4172. http://www.ncbi.nlm.nih.gov/pubmed/21376230 (Mar. 2011).

3. Gerstung, M. *et al.* The evolutionary history of 2,658 cancers. *Nature* **578,** 122–128. ISSN: 1476-4687. https://www.nature.com/articles/s41586-019-1907-7 (2020) (Feb. 2020).

4. Espiritu, S. M. G. *et al.* The Evolutionary Landscape of Localized Prostate Cancers Drives Clinical Aggression. *Cell* **173,** 1003–1013.e15. ISSN: 0092-8674. http://www.sciencedirect.com/science/article/pii/S009286741830309X (2018) (May 3, 2018).

5. Jamal-Hanjani, M. *et al.* Tracking the Evolution of Non–Small-Cell Lung Cancer. *New England Journal of Medicine* **376,** 2109–2121. ISSN: 0028-4793, 1533-4406. http://www.nejm.org/doi/10.1056/NEJMoa1616288 (2018) (June 2017).

6. Gundem, G. *et al.* The evolutionary history of lethal metastatic prostate cancer. *Nature* **520,** 353–357. ISSN: 1476-4687 (Apr. 16, 2015).

7. Sakamoto, H. *et al.* Evolutionary Origins of Recurrent Pancreatic Cancer. *bioRxiv,* 811133. https://www.biorxiv.org/content/10.1101/811133v1 (2019) (Oct. 31, 2019).

8. Alves, J. M., Prado-López, S., Cameselle-Teijeiro, J. M. & Posada, D. Rapid evolution and biogeographic spread in a colorectal cancer. *Nature Communications* **10.** Number: 1 Publisher: Nature Publishing Group, 5139. ISSN: 2041-1723. https://www.nature.com/articles/s41467-019-12926-8 (2020) (Nov. 13, 2019).

9. Hu, Z. *et al.* Quantitative evidence for early metastatic seeding in colorectal cancer. *Nature Genetics,* 1. ISSN: 1546-1718. https://www.nature.com/articles/s41588-019-0423-x (2019) (June 17, 2019).

10. Dobson, S. M. *et al.* Relapse-Fated Latent Diagnosis Subclones in Acute B Lineage Leukemia Are Drug Tolerant and Possess Distinct Metabolic Programs. *Cancer Discovery.* Publisher: American Association for Cancer Research Section: Research Articles. ISSN: 2159-8274, 2159-8290. https://cancerdiscovery.aacrjournals.org/content/early/2020/03/12/2159-8290.CD-19-1059 (2020) (Feb. 21, 2020).

11. Hu, Z., Li, Z., Ma, Z. & Curtis, C. Multi-cancer analysis of clonality and the timing of systemic spread in paired primary tumors and metastases. *Nature Genetics* **52.** Number: 7 Publisher: Nature Publishing Group, 701–708. ISSN: 1546-1718. https://www.nature.com/articles/s41588-020-0628-z (2020) (July 2020).

12. Zahir, N., Sun, R., Gallahan, D., Gatenby, R. A. & Curtis, C. Characterizing the ecological and evolutionary dynamics of cancer. *Nature Genetics* **52.** Number: 8 Publisher: Nature Publishing Group, 759–767. ISSN: 1546-1718. https://www.nature.com/articles/s41588-020-0668-4 (2020) (Aug. 2020).

13. Williams, M. J. *et al.* Quantification of subclonal selection in cancer from bulk sequencing data. *Nature Genetics,* 1. ISSN: 1546-1718. http://www.nature.com/articles/s41588-018-0128-6 (2018) (May 28, 2018).

14. Pogrebniak, K. L. & Curtis, C. Harnessing Tumor Evolution to Circumvent Resistance. *Trends in Genetics* **34,** 639–651. ISSN: 01689525. https://linkinghub.elsevier.com/retrieve/pii/S0168952518300921 (2018) (Aug. 2018).

15. Jiang, Y., Qiu, Y., Minn, A. J. & Zhang, N. R. Assessing intratumor heterogeneity and tracking longitudinal and spatial clonal evolutionary history by next-generation sequencing. *Proceedings of the National Academy of Sciences* **113,** E5528–E5537. ISSN: 0027-8424, 1091-6490. http://www.pnas.org/content/113/37/E5528 (2017) (Sept. 13, 2016).

16. Malikic, S., McPherson, A. W., Donmez, N. & Sahinalp, C. S. Clonality inference in multiple tumor samples using phylogeny. *Bioinformatics* **31,** 1349–1356. ISSN: 1460-2059, 1367-4803. https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btv003 (2019) (May 1, 2015).

17. 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.** Number: 1 Publisher: Nature Publishing Group, 2750. ISSN: 2041-1723. https://www.nature.com/articles/s41467-019-10737-5 (2020) (June 21, 2019).

18. Deshwar, A. G., Vembu, S. & Morris, Q. Comparing nonparametric Bayesian tree priors for clonal reconstruction of tumors. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing,* 20–31. ISSN: 2335-6936 (2015).

19. Popic, V. *et al.* Fast and scalable inference of multi-sample cancer lineages. *Genome Biology* **16.** ISSN: 1465-6906. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4501097/ (2017) (2015).

20. Satas, G. & Raphael, B. J. Tumor phylogeny inference using tree-constrained importance sampling. *Bioinformatics* **33,** i152–i160. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/33/14/i152/3953987/Tumor-phylogeny-inference-using-tree-constrained (2017) (July 15, 2017).

21. Salcedo, A. *et al.* A community effort to create standards for evaluating tumor subclonal reconstruction. *Nature Biotechnology* **38.** Number: 1 Publisher: Nature Publishing Group, 97–107. ISSN: 1546-1696. https://www.nature.com/articles/s41587-019-0364-z (2020) (Jan. 2020).

22. Dentro, S. C., Wedge, D. C. & Van Loo, P. Principles of Reconstructing the Subclonal Architecture of Cancers. *Cold Spring Harbor Perspectives in Medicine,* a026625 (2017).

23. Kuipers, J., Jahn, K., Raphael, B. J. & Beerenwinkel, N. Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors. *Genome Research* **27,** 1885–1894. ISSN: 1088-9051. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5668945/ (2020) (Nov. 2017).

24. Singer, J., Kuipers, J., Jahn, K. & Beerenwinkel, N. Single-cell mutation identification via phylogenetic inference. *Nature Communications* **9,** 5144. ISSN: 2041-1723. https://www.nature.com/articles/s41467-018-07627-7 (2020) (Dec. 4, 2018).

25. Bonizzoni, P., Ciccolella, S., Della Vedova, G. & Soto, M. Does relaxing the infinite sites assumption give better tumor phylogenies? An ILP-based comparative approach. *bioRxiv.* http://biorxiv.org/lookup/doi/10.1101/227801 (2019) (Dec. 3, 2017).

26. Ciccolella, S. *et al.* Inferring Cancer Progression from Single-cell Sequencing while Allowing Mutation Losses. *bioRxiv.* http://biorxiv.org/lookup/doi/10.1101/268243 (2019) (Apr. 13, 2018).

27. Satas, G., Zaccaria, S., Mon, G. & Raphael, B. J. SCARLET: Single-Cell Tumor Phylogeny Inference with Copy-Number Constrained Mutation Losses. *Cell Systems* **10,** 323–332.e8. ISSN: 2405-4712. http://www.sciencedirect.com/science/article/pii/S2405471220301150 (2020) (Apr. 22, 2020).

28. Roth, A. *et al.* PyClone: statistical inference of clonal population structure in cancer. *Nature methods* (2014).

29. Miller, C. A. *et al.* SciClone: Inferring Clonal Architecture and Tracking the Spatial and Temporal Patterns of Tumor Evolution. *PLoS Computational Biology* **10** (ed Beerenwinkel, N.) e1003665. ISSN:

1928 1553-734X. JSTOR: {PMC}4125065. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4125065/
1929 (Aug. 2014).

1930 30. Gillis, S. & Roth, A. PyClone-VI: Scalable inference of clonal population structures using whole
1931 genome data. *bioRxiv*. Publisher: Cold Spring Harbor Laboratory Section: New Results, 2020.08.31.276212.
1932 https://www.biorxiv.org/content/10.1101/2020.08.31.276212v1 (2020) (Sept. 1, 2020).

1933 31. Tarabichi, M. *et al.* A practical guide to cancer subclonal reconstruction from DNA sequencing.
1934 *Nature Methods* **18.** Number: 2 Publisher: Nature Publishing Group, 144–155. ISSN: 1548-7105.
1935 https://www.nature.com/articles/s41592-020-01013-2 (2021) (Feb. 2021).

1936 32. Hastings, W. K. Monte Carlo Sampling Methods Using Markov Chains and Their Applications.
1937 *Biometrika* **57.** Publisher: [Oxford University Press, Biometrika Trust], 97–109. ISSN: 0006-3444.
1938 https://www.jstor.org/stable/2334940 (2020) (1970).

1939 33. Jahn, K., Kuipers, J. & Beerenwinkel, N. Tree inference for single-cell data. *Genome Biology* **17,**
1940 86. ISSN: 1474-760X. http://dx.doi.org/10.1186/s13059-016-0936-x (2016) (2016).

1941 34. Jia, B., Ray, S., Safavi, S. & Bento, J. Efficient Projection onto the Perfect Phylogeny Model.
1942 *arXiv:1811.01129 [cs]*. arXiv: 1811.01129. http://arxiv.org/abs/1811.01129 (2018) (Nov. 2,
1943 2018).

1944 35. Sundermann, L. K., Wintersinger, J., Rätsch, G., Stoye, J. & Morris, Q. Reconstructing tumor
1945 evolutionary histories and clone trees in polynomial-time with SubMARine. *PLOS Computational*
1946 *Biology* **17.** Publisher: Public Library of Science, 1–28. https://doi.org/10.1371/journal.
1947 pcbi.1008400 (2021).

1948 36. Deshwar, A. G. *et al.* PhyloWGS: Reconstructing subclonal composition and evolution from whole
1949 genome sequencing of tumors. *Genome Biology* **16,** 35 (2015).

1950 37. Satas, G. & Raphael, B. *PASTRI source code* Aug. 16, 2019. https://github.com/raphael-
1951 group/PASTRI (2020).

1952 38. Hudson, R. R. & Kaplan, N. L. Statistical Properties of the Number of Recombination Events
1953 in the History of a Sample of DNA Sequences. *Genetics* **111,** 147–164. ISSN: 0016-6731. https:
1954 //www.ncbi.nlm.nih.gov/pmc/articles/PMC1202594/ (2019) (Sept. 1985).

1955 39. Campbell, P. J. *et al.* Pan-cancer analysis of whole genomes. *Nature* **578.** Number: 7793 Publisher:
1956 Nature Publishing Group, 82–93. ISSN: 1476-4687. https://www.nature.com/articles/s41586-
1957 020-1969-6 (2021) (Feb. 2020).

40. Riedmiller, M. & Braun, H. *A direct adaptive method for faster backpropagation learning: The RPROP algorithm* in *IEEE international conference on neural networks* (1993), 586–591.

41. Hinton, G. *Neutral Networks for Machine Learning: Lecture 6a: Overview of min-batch gradient descent.* 2014. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

42. Bei, J. & Bento, J. *libprojectppm source code* original-date: 2018-10-26T22:41:45Z. Apr. 8, 2019. https://github.com/bentoayr/Efficient-Projection-onto-the-Perfect-Phylogeny-Model.

43. Gusfield, D. Efficient algorithms for inferring evolutionary trees. *Networks* **21,** 19–28. ISSN: 1097-0037. http://onlinelibrary.wiley.com/doi/10.1002/net.3230210104/abstract (2017) (Jan. 1, 1991).

44. El-Kebir, M., Oesper, L., Acheson-Field, H. & Raphael, B. J. Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. *Bioinformatics (Oxford, England)* **31,** i62–70. ISSN: 1367-4811 (June 15, 2015).

45. Stewart, C. A. *et al.* Single-cell analyses reveal increased intratumoral heterogeneity after the onset of therapy resistance in small-cell lung cancer. *Nature Cancer* **1.** Number: 4 Publisher: Nature Publishing Group, 423–436. ISSN: 2662-1347. https://www.nature.com/articles/s43018-019-0020-z (2020) (Apr. 2020).

46. Miles, L. A. *et al.* Single cell mutational profiling delineates clonal trajectories in myeloid malignancies. *bioRxiv.* Publisher: Cold Spring Harbor Laboratory Section: New Results, 2020.02.07.938860. https://www.biorxiv.org/content/10.1101/2020.02.07.938860v1 (2020) (Feb. 9, 2020).

47. Lähnemann, D. *et al.* Eleven grand challenges in single-cell data science. *Genome Biology* **21,** 31. ISSN: 1474-760X. https://doi.org/10.1186/s13059-020-1926-6 (2020) (Feb. 7, 2020).