© 2022 Juho Kim

PROBABILISTIC SUBCLONAL RECONSTRUCTION FOR CANCER

BY

JUHO KIM

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2022

Urbana, Illinois

Doctoral Committee:

Associate Professor Oluwasanmi Koyejo, Chair
Assistant Professor Mohammed El-Kebir, Chair
Professor Olgica Milenkovic
Assistant Professor Ilan Shomorony
Associate Professor Nicholas Chia, Mayo Clinic

ABSTRACT

Cancer consists of genetically heterogeneous populations of cells that arise through a process of subclonal evolution. Reconstructing the evolutionary processes that give rise to cancer can help us better understand cancer progression and prioritize treatment targets. The subclonal reconstruction of cancer gives us the information about the co-occurrence of mutations within the same subclone, the underlying proportion of cells belonging to each subclone, and the ancestral relationships between them. The evolutionary process can be described by inferring tumor phylogenetic trees. The majority of current approaches focus only on either mutation clustering or tree inference in isolation, or rely on computationally expensive algorithms to holistically consider clustering and tree inference concurrently.

In this dissertation, we formalize the problem of reconstructing subclonal structure for cancer via probabilistic modeling. Using variant and total read count obtained from bulk DNA sequencing data as input, we introduce a tree-constrained binomial mixture model and an expectation-maximization (EM) method to estimate the clustering assignment for each mutation and the underlying frequency for each cluster. Our EM algorithm employs a linear programming approach to accurately maximize the likelihood bound subject to tree constraints. We choose the optimal tree topology by repeating the process across all possible tree topologies. Compared to existing work, the resulting ClusTree algorithm more accurately identifies mutation clusters, estimates frequencies for each cluster, and detects the proper tree topology, especially for low-depth sequencing data.

To my family, Hyunji and Luna

ACKNOWLEDGMENTS

I would like to thank my advisor Professor Oluwasanmi Koyejo for his advice and patience. His guidance kept me on the right track to finish my Ph.D. journey. Without his support and encouragement, it would not have been possible to complete this dissertation. I was fortunate to join his research group and work with him throughout my graduate study.

I also would like to thank Professor Mohammed El-Kebir. He introduced cancer genomics to me, which became the main part of this dissertation. His introduction to combinatorial optimization and theoretical computer science broadened my perspective. I would like to thank Professor Nicholas Chia of Mayo Clinic. Visiting Mayo Clinic to collaborate with him and his group was an amazing learning experience. I was able to learn how to collaborate and communicate with people from different scientific fields.

I would like to thank my Doctoral Committee, including Professor Olgica Milenkovic and Professor Ilan Shomorony for their thoughtful feedback and comments. I would like to express my gratitude to Professor Jian Peng. At the beginning of my graduate study, he helped me to start machine learning research to address computational biology and healthcare problems.

I would like to acknowledge my colleagues and friends from the machine-learning group, the cancer-genomics group, and the computational-biology group. It was great fun to discuss research and chat with them. I am grateful for the mentorship of Dr. Christopher Malon and Dr. Asim Kadav during my internship at NEC Laboratories America.

I would like to express my deepest gratitude to my wife Hyunji, for her support and patience. She has always been the source of my happiness and strength. Without her, it would not have been possible for me to overcome the difficulties and challenges during my graduate study.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Cancer is a complex disease associated with evolutionary processes where somatic mutations accumulate and drive tumor growth over time [1]. Tumors often consist of a variety of genetically diverse subclonal populations of cancer cells. This *intra-tumor heterogeneity* [2, 3, 4] is a major reason to cause therapeutic resistance and treatment failure, which makes cancer one of the leading causes of death globally [5]. Thus, characterizing this intra-tumor heterogeneity provides key insights to understanding how cancer progresses and to treating cancer in a comprehensive way.

Reconstructing subclonal structures for cancer consists of several subproblems: detecting the mutations that co-occur within the same subclone, estimating the underlying cell proportion for each subclone, and identifying the ancestral relationships between subclones. We use *tumor phylogenetic trees* to represent the subclonal structures of tumors.

Recent high-throughput molecular sequencing technologies allow us to study the subclonal reconstruction. A standard type of data for this study is bulk DNA sequencing data. Sequencing bulk samples from tumors provides the information about somatic mutations. These consist of single nucleotide variants (SNVs), insertions and deletions of DNA base pairs. These observed variant and reference read counts in bulk DNA sequencing originate from unknown tumor subclones mixed in unknown proportions, each with an unknown complement of SNVs. The resolution of these mixed samples is inherently limited, and this makes the inference of the subclonal structure harder.

Many advanced computational methods have been developed to tackle the subclonal reconstruction problems using bulk DNA sequencing data. However, current algorithms still have plenty of limitations. Multiple solutions from tumor phylogenetic tree inference are consistent with input DNA sequencing data, which leads to a large solution space including equally plausible trees [6]. In addition, most of phylogenetic tree inference algorithms are hard to scale to tumor phylogenetic trees where there are a large number of subclones. Another issue is that many tumor phylogenetic tree inference methods usually are not standalone by highly depending on the output of other existing approaches. For example, some of phylogenetic tree inference methods take mutation clusters as input, which are obtained from other independent mutation clustering algorithms. They cannot control the quality of clustering, and the propagation of the incorrect information about clustering can have a negative impact on the overall performance of the methods.

In this dissertation, we introduce a probabilistic method, ClusTree, utilizing variant and total read count from bulk DNA sequencing data to perform the reconstruction of a subclonal structure for cancer. The method is based on a tree-constrained binomial mixture model and an expectation-maximization (EM) algorithm to cluster mutations and estimate the underlying frequency for each cluster. Our EM algorithm employs a linear programming approach to accurately maximize the likelihood bound subject to the constraints imposed by an underlying tree topology. By repeating this process across all possible tree topologies, we find the clustering assignment, the frequency of each cluster, and the tree topology at the maximum likelihood. We demonstrate on simulation datasets that ClusTree outperforms previous methods in the context of clustering mutations and inferring pairwise relationships between mutations in a tumor phylogenetic tree. Next, we consider a real dataset from an acute myeloid leukemia (AML) patient from Griffith et al. [7]. We create low-depth sequencing datasets by downsampling the original AML data. ClusTree generates better clustering results, compared to both clustering-only and joint methods. In addition, ClusTree recovers the tumor phylogenetic trees reported in Griffith et al. [7] and finds additional possible trees, while other baseline methods return a different set of trees.

We first provide the information about the subclonal reconstruction problem including key concepts, datasets, and related work in Chapter 2. In Chapter 3, we present the main algorithm, ClusTree, of this dissertation with the details about the algorithm including the derivation. We analyze the performance of ClusTree against multiple baseline methods using both simulation and real datasets in Chapter 4. Last, we summarize the main method and results, and present remaining future work in Chapter 5.

CHAPTER 2

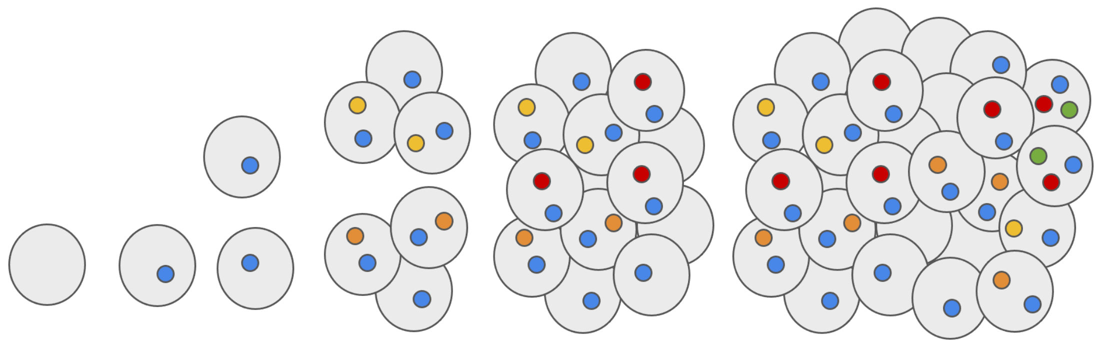
BACKGROUND AND RELATED WORK

In this chapter, we discuss the background materials about reconstructing subclonal structures for cancer. We first introduce the concepts and the problem we are going to solve throughout this dissertation, followed by the previous work that is related to this work.

2.1 Subclonal Reconstruction for Cancer

Clonal Evolution and Tumor Phylogenetic Tree According to the clonal evolution theory [1], tumors initiate when normal cells acquire mutations that grow rapidly and uncontrollably and become founder cells with initial somatic mutations. Some of driver mutations can have growth advantage compared to other neighboring cells. And they lead to faster tumor growth. This evolutionary process continues over time and leads to the *intra-tumor heterogeneity* [2, 3, 4], a distinct population of tumor cells within the same tumor (Fig. 2.1 (a)).

In order to know how cancer progresses, we need to understand the evolutionary process and uncover the heterogeneous population structure of cancer. The evolution of cancer can be mathematically represented using a *tumor phylogenetic tree* T (Figure 2.1(b)). T is a rooted tree whose edges are directed away from the root vertex to represent the order of the process. The root vertex typically corresponds to the normal cells, and other vertices of T represent groups of cells, or *clones*. The leaf vertices correspond to clones observed at the time of sequence sampling, and the internal vertices represent ancestral clones. The edges of T represent the introduction of a new mutation or mutation clusters. Thus, by inferring tumor phylogenetic trees, we can detect which mutations are introduced together into the evolutionary trajectories and know the phylogenetic relationships among subclones.


(a) Cancer evolution and intra-tumor heterogeneity

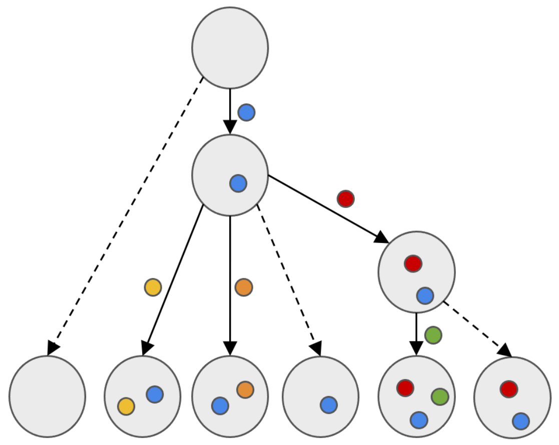

(b) Tumor phylogenetic tree

Figure 2.1: **Cancer evolution and tumor phylogenetic tree** (a) Cancer is an evolutionary process where somatic mutations accumulate and lead to the growth of a tumor during an individual's lifetime. (b) The evolutionary history of cancer can be represented by a tumor phylogenetic tree. The colored circles beside an edge describe the introduction of a new mutation. The leaf vertices of the tree correspond to subclones at the time of sampling.

The majority of current methods [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23] to infer tumor phylogenetic trees using DNA sequencing data have been developed under the *infinite site assumption* [24] where each mutation is gained only once and never lost during the evolutionary process. The research work in this dissertation also adheres to this assumption. We note that there is evidence that this assumption can be violated for some circumstances [25]. The authors found that the same genomic position can be mutated multiple times in individual tumors using 11 of 12 single-cell datasets from various human cancers. They observed parallel mutations at the same position, back mutations at the same site, and losses of mutations through deletions. They said the infinite site assumption should be used carefully.

2.2 Relevant Data and Related Work

Bulk DNA Sequencing Bulk DNA sequencing is typically used to obtain tumor data from patients. The DNA fragments from this sequencing technique are mixed together since cells from a sequencing sample are processed simultaneously. The sequencing reads from the cells are aligned to a *reference genome*. We obtain single nucleotide variants (SNVs) from the aligned sequencing reads. At the position of each SNV, we can observe the number of reads that mismatch with the reference genome, which is *variant read count*. The total number of reads is called *total read count*. Additionally, the subtraction of variant read count from total read count is called *reference read count*. The ratio between variant and total read count is *variant allele frequency* (VAF). Those observed read counts or VAFs are used as input when current methods infer tumor phylogenetic trees based on bulk DNA sequencing data.

The tumor phylogenetic tree inference based on bulk DNA sequencing tries to solve a mixture problem, given an input VAF matrix **F** whose rows correspond to sequenced samples and columns correspond to mutations. The frequency matrix **F** can be factorized into two matrices, **U** and **B**. The matrix **U** is a mixture matrix where the entries of **U** represent a mixture proportion of clones in each sample. The matrix **B** represents the assignment of mutations into clones and is equivalent to a tumor phylogenetic tree. We introduce underlying constraints of $\mathbf{U}$ and $\mathbf{B}$. Each entry of $\mathbf{U}$ should be nonnegative, and the sum of each row should be less than or equal to 1. The tree matrix $\mathbf{B}$ is a binary matrix whose entry is 0 or 1. Its important property is that $\mathbf{B}$ is lower-triangular and invertible. The problem to find an evolutionary model based on this matrix factorization $\mathbf{F} = \mathbf{UB}$ is called the *perfect phylogeny mixture* problem [15].

Based on the perfect phylogeny mixture $\mathbf{F} = \mathbf{UB}$ and the underlying constraints of $\mathbf{U}$ and $\mathbf{B}$, we can obtain the following constraints called the *sum condition* [10, 11, 13, 15]. For vertex v of T and its child vertices, the *sum condition* is

$$f_{i,v} \geq \sum_{v' \in child(v)} f_{i,v'} \tag{2.1}$$

where $f_{i,v}$ is the underlying frequency of vertex v in sequencing sample i, and $child(v)$ is a set of child vertices of v.

Using bulk DNA sequencing data, a number of methods have been developed to tackle the subclonal reconstruction problem. The majority of current approaches focus on either mutation clustering or tree inference in isolation. Methods such as PyClone [26], PyClone-VI [27] and SciClone [28] focus only on detecting mutation clusters, without considering any evolutionary relationships between clusters. PyClone is based on a beta-binomial mixture (or binomial mixture) model with a Dirichlet process prior [29] for clustering mutations. It has been used in many applications, but its posterior inference based on Markov Chain Monte Carlo (MCMC) sampling [30] is computationally expensive. In order to overcome the inefficiency, PyClone-VI is developed by replacing the MCMC sampling-based inference with variational inference [31, 32]. This newer version resolves the scalability issue of the original algorithm, but the accuracy can be decreased due to the approximation. SciClone constructs a Bayesian mixture model based on several underlying distributions (beta, binomial or Gaussian) and uses variational inference for posterior estimation [33, 34]. These three methods are widely used for clustering mutations in practice. However, they fail to accurately infer clusters for low-depth bulk DNA sequencing data ($20\times$), which needs a guide of a tumor phylogenetic tree for better statistical inference based on our experiments and Liu et al. [35].

Another line of methods has been developed to focus mainly on inferring tumor phylogenetic trees given mutation clusters. There exist several methods based on combinatorial optimization [9, 11, 13, 15, 36, 37]. They explore a constrained space determined by the underlying phylogenetic tree and enumerate all possible trees satisfying the constraints. However, since they usually focus on enumerating the solution trees that are equally plausible, the choice of a single tree from the solution trees is ambiguous. Recent probabilistic methods, ClonEvol [38] and Pairtree [18], also belong to this category. These methods first reduce the size of the input data from the number of mutations to the number of mutation clusters via a chosen mutation clustering algorithm such as SciClone and PyClone-VI. ClonEvol uses a bootstrap resampling to estimate the cellular fraction of each clone and model the clonal ordering. Pairtree is based on Bayesian inference to estimate the posterior distributions over the ancestral relationships between pairs. However, many of the methods still operate in the infinite site assumption which imposes tree constraints on frequencies of mutation clusters ordered in a tree. Also, the overall performance of tumor phylogeny inference highly depends on the quality of the clustering method they choose.

There are joint methods to consider both mutation clustering and tumor phylogeny inference, simultaneously. PhyloSub [10] and PhyloWGS [14] introduce a binomial mixture model with a tree-structured stick breaking process prior [39]. They depend on Markov Chain Monte Carlo (MCMC) sampling for the posterior inference. Canopy [16] is based on non-informative prior and searching a space defined by a mixture model using MCMC sampling. The MCMC sampling-based inference of these methods is computationally expensive even for datasets of realistic size. PASTRI [17] is based on importance sampling, which is guided by a proposal distribution Q obtained from the output of an existing mutation clustering method such as SciClone [28]. Using the frequency samples drawn from Q, PASTRI considers all the permutations that can satisfy the sum condition in Eq. 2.1 given a tree topology. It is computationally faster than other joint methods, but its performance highly depends on Q. The clustering-only method for Q is not designed to satisfy the tree constraints, so PASTRI could have no solution for some sets of frequency samples. This would be deteriorated when the number of sequencing samples increases, since the higher number of sequencing samples makes more complicated feasible sets.

Single-cell DNA sequencing Single-cell DNA sequencing allows us to sequence the genome in a individual cell level without any deconvolution. While the data is promising to investigate the subclonal structure of tumors [40, 41, 42], current technologies have challenges [43]. The physical isolation of single cells is not trivial [41, 44, 45, 46], and the data contains high rates of false positives and false negatives which occur from the process of DNA amplification [47, 48, 49]. It includes a number of missing values [41, 50] due to the non-uniformity of amplification. Additionally, single-cell DNA sequencing is still more expensive and harder to scale than bulk DNA sequencing.

Even thought there are still limitations in this sequencing technique, researchers have tried to use the data to infer subclonal structures of cancer. Some methods focus on clustering cells into clones and inferring their genotypes for each clone [51, 52]. SCG [51] develops a parametric model to estimate the clustering assignment and its genotyping via mean-field variational inference [53, 54]. BnpC [52] uses non-parametric Bayesian modeling based on Chinese restaurant process [55] for clustering and genotyping.

Another line of methods focuses on inferring the phylogenetic relationship among cells and providing the information about clustering and genotyping. SCITE [20] and OncoNEM [21] are based on probabilistic modeling. They consider inferring tumor phylogenetic trees using single-cell DNA sequencing data by maximizing the likelihood. To find the maximum likelihood, they depend on their search algorithm based on Markov chain Monte Carlo or heuristics. While other methods stick to the infinite site assumption, SiFit [56] is developed under a finite-site model to describe the cancer evolution. SiCloneFit [57] combines a finite-site model of SiFit with non-parametric mixture modeling based on tree-structured Chinese restaurant process prior [58]. SPhyR [59] considers inferring a k-Dollo phylogeny by restricting the Dollo parsimony model [60] to at most k losses. PhISCS-BnB [61] introduces the branch and bound algorithm to efficiently explore the combinatorial space to infer tumor phylogenetic trees using input single-cell DNA sequencing data.

Some recent methods have tried to use the complimentary features of bulk and single-cell DNA sequencing data by using them simultaneously for joint statistical inference. ddClone [22] considers both types of data simultaneously in its Bayesian model for clustering. The information about Bulk DNA sequencing is used in the likelihood of the model, and the information about single-cell DNA sequencing is used in its prior. B-SCITE [23] is developed for both clustering and inferring tumor phylogenetic trees by extending the idea of SCITE [20]. B-SCITE defines a joint likelihood score that combines bulk and single-cell DNA sequencing data, and it tries to maximize the likelihood by searching the space based on Markov chain Monte Carlo. PhISCS [62] uses combinatorial optimization that is based on integer linear programming to infer tumor phylogenetic trees that slightly violate the infinite site assumption from bulk and single-cell DNA sequencing data.

CHAPTER 3

A NEW PROBABILISTIC SUBCLONAL RECONSTRUCTION METHOD

In this chapter, we introduce our probabilistic model, ClusTree, which performs the reconstruction of subclonal structures of tumors. ClusTree is based on a tree-constrained binomial mixture model and an expectation-maximization (EM) algorithm to cluster mutations and estimate the underlying frequency for each cluster, using variant and total read count from bulk DNA sequencing data as input (Fig. 3.1). Our EM algorithm employs a linear programming approach to accurately maximize the likelihood bound subject to the constraints imposed by an underlying tree topology. By repeating this process across all possible tree topologies, we find the clustering assignment, the frequency of each cluster, and the tree topology at the maximum likelihood. We demonstrate on simulation datasets that ClusTree outperforms previous methods in the context of clustering mutations and inferring pairwise relationships between mutations in a tumor phylogenetic tree. Next, we consider a real dataset from an acute myeloid leukemia (AML) patient from Griffith et al. [7]. We create low-depth sequencing datasets by downsampling the original AML data. ClusTree generates better clustering results, compared to clustering-only and joint methods. In addition, ClusTree recovers the tumor phylogenetic trees [7] and finds additional possible trees, while other baseline methods show a different set of trees. In Section 3.1, we first introduce the underlying generative model of ClusTree with a graphical representation in Fig. 3.2. In Section 3.2, we describe the maximum likelihood estimation framework we use to infer latent variables for the clustering assignment, center frequency of each cluster, and a phylogenetic tree topology. In Section 3.3, we explain model selection that is used to select the number of clusters. We explain the details about the implementation of ClusTree in Section 3.4

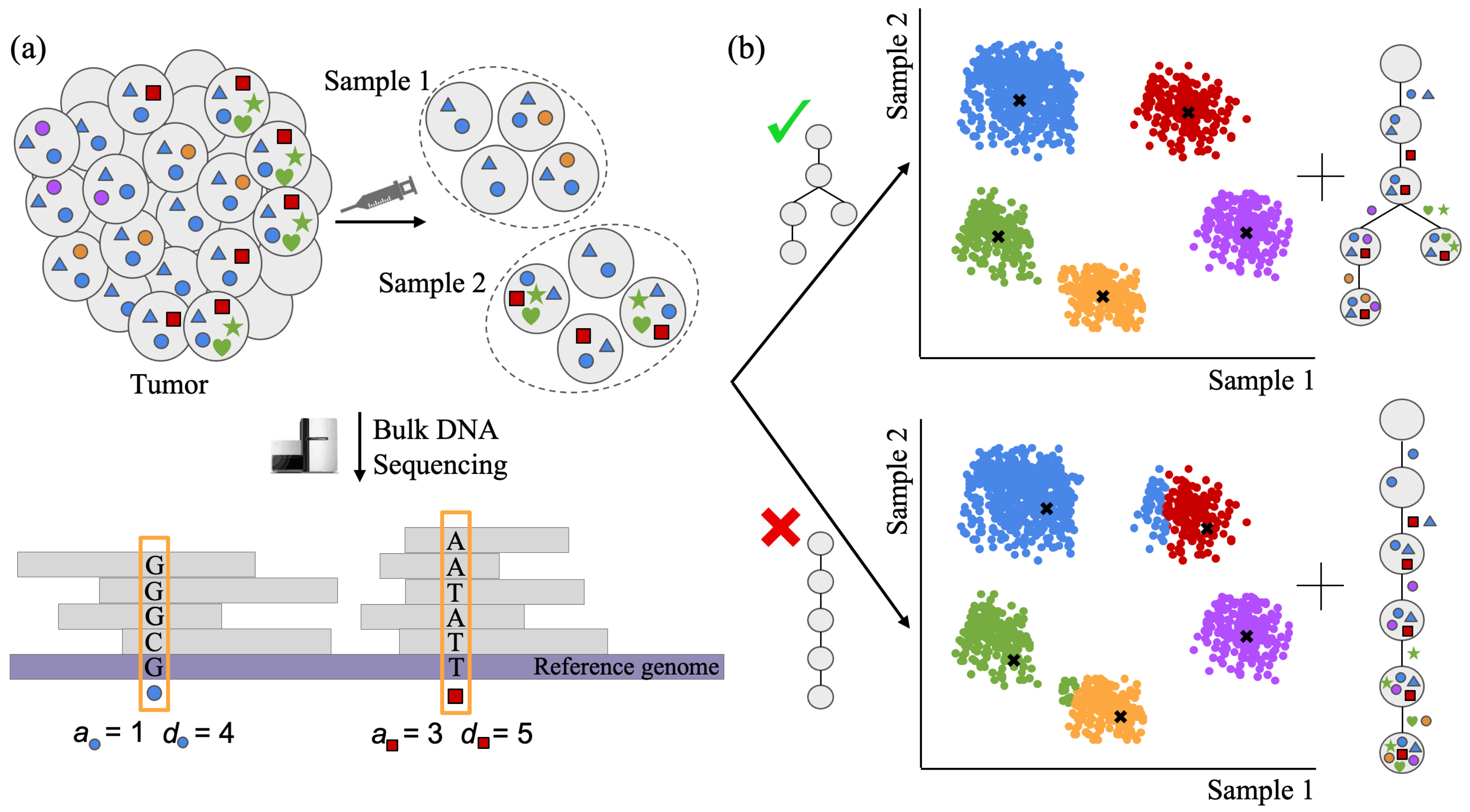


Figure 3.1: ClusTree reconstructs the subclonal population structure of heterogeneous tumors. (a) The input of ClusTree is variant and reference read count of mutations obtained from bulk DNA sequencing data of heterogeneous tumors. Using the read count data, ClusTree (b) detects mutation clusters, estimates the underlying frequency of each cluster, and infers the clonal tree that represents ancestral relationships between mutation clusters. In order to do that, ClusTree introduces a tree-constrained binomial mixture model and efficient inference based on an expectation-maximization algorithm.

3.1 Generative Model

ClusTree is based on a generative model to reconstruct subclonal structures of tumors by inferring mutation clusters, their latent frequency, and an underlying phylogenetic tree. We are given variant reads $\mathbf{A} = [a_{i,j}]$ and total reads $\mathbf{D} = [d_{i,j}]$ from bulk DNA sequencing data as input where $a_{i,j}$ and $d_{i,j}$ are the variant and total read counts for a single nucleotide variant $j \in [n]$ in a DNA sequencing sample $i \in [m]$, respectively. We assume the n mutations originate from ℓ clusters. Let $\mathbf{C} = [\mathbf{C}_1, \cdots, \mathbf{C}_n]^T$ be a random variable to represent the clustering assignment of mutations, such that $\mathbf{C}_j \in [\ell]$ indicates the cluster that mutation j belongs to. Each cluster k of sequencing sample i has latent frequency $f_{i,k}$. The latent frequency is constrained by the underlying tumor phylogeny T, which is a rooted tree whose vertices corresponds to one of the mutation clusters. The frequency of a parent vertex in T is at most the sum of the frequencies of its child vertices, which is the sum condition [10, 11, 13, 15] introduced in Chapter 2.2.

Our method considers the posterior probability of a tumor phylogenetic tree T, given observed variant and total read counts, $\mathbf{A}$ and $\mathbf{D}$. The posterior, $p(T|\mathbf{A}, \mathbf{D})$ is proportional to the likelihood, $p(\mathbf{A}|\mathbf{D}, T)$ and the prior, $p(T)$ by Bayes' theorem. By assuming the prior over T is uniformly distributed, maximizing the posterior is equivalent to maximizing the likelihood. Given a fixed tumor phylogenetic tree, we compute the likelihood by marginalizing latent variables for the clustering assignment $\mathbf{C}$ and underlying frequency $\mathbf{F}$. The domain of $\mathbf{C}$ and $\mathbf{F}$ is determined by model selection of the number of clusters and by the sum condition constrained by the fixed underlying tree, respectively. We assume a binomial distribution as the underlying distribution of the mixture structure to generate variant read counts $\mathbf{A}$.

$$\begin{aligned} p(T|\mathbf{A}, \mathbf{D}) &\propto p(\mathbf{A}|\mathbf{D}, T)p(T) && (3.1) \\ &= \left[\int_{\mathbf{F}} \sum_{\mathbf{C}} p(\mathbf{A}|\mathbf{C}, \mathbf{F}, \mathbf{D}) p(\mathbf{C}) p(\mathbf{F}|T) d\mathbf{F} \right] p(T) && (3.2) \\ &= \mathbb{E}_{\mathbf{F}} \mathbb{E}_{\mathbf{C}} [p(\mathbf{A}|\mathbf{C}, \mathbf{F}, \mathbf{D})] p(T), && (3.3) \end{aligned}$$

where $\mathbb{E}_X[\cdot]$ is the expectation with respect to a random variable X. In Fig. 3.2, we show our generative framework using a graphical model. For indices $i \in [m]$ and $j \in [n]$, the observations $[a_{i,j}]$ and $[d_{i,j}]$ are represented

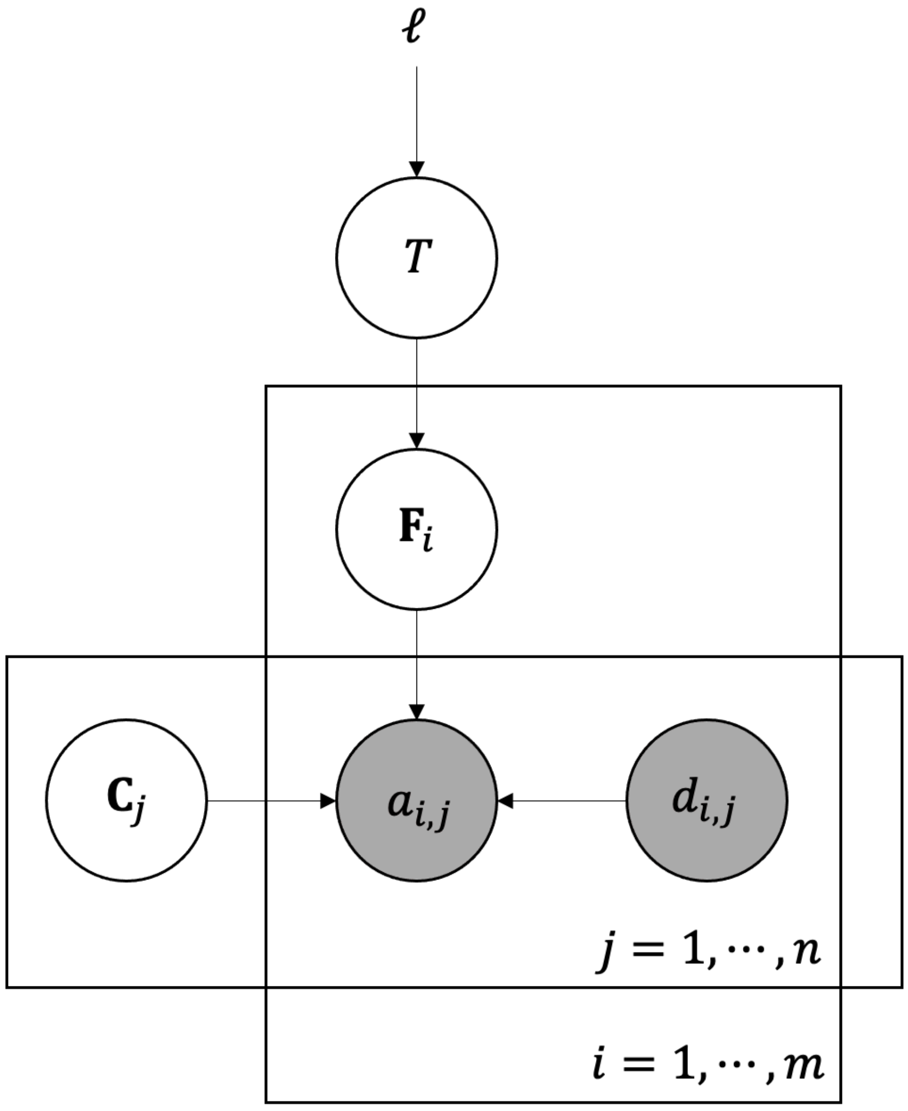


Figure 3.2: ClusTree is based on a generative model for variant read counts $\mathbf{A}$ using bulk DNA sequencing data. Given observed total read counts $d_{i,j}$, variant read counts $a_{i,j}$ for mutation j in sample i is generated by a binomial mixture model with a parameter $\mathbf{C}_j$ for the clustering assignment of mutation j and a parameter $f_{i,\mathbf{C}_j}$ for the underlying frequency of cluster $\mathbf{C}_j$. The latent frequency $\mathbf{F}_i$ for each sample i respects the sum condition, determined by the underlying tree topology T. The choice of T is uniformly determined among the possible tree topologies, which is determined by the choice of the number ℓ of clusters.

as shaded circles, while the latent variables $\mathbf{F}_i$, $\mathbf{C}_j$, and T are represented as circles. The hyperparameter ℓ is a user-specified value.

3.2 Maximum Likelihood Estimation

Computing the likelihood is time-consuming, since it requires the marginalization with respect to the clustering assignment $\mathbf{C}$ and latent frequency $\mathbf{F}$ in a constrained space defined by a fixed tree topology. To avoid the expensive computation, we consider the lower bound of the log-likelihood, $\log p(\mathbf{A}|\mathbf{D}, T)$ by applying Jensen's inequality.

We derive a lower bound of the logarithm of our likelihood, $\log \mathbb{E}_{\mathbf{F}}\mathbb{E}_{\mathbf{C}}[p(\mathbf{A}|\mathbf{C}, \mathbf{F}, \mathbf{D})]$, by applying Jensen's inequality twice.

$$\begin{aligned}
&\log \mathbb{E}_{\mathbf{F}}\mathbb{E}_{\mathbf{C}}[p(\mathbf{A}|\mathbf{C}, \mathbf{F}, \mathbf{D})] && (3.4)\\
&\geq \mathbb{E}_{\mathbf{F}}\log \mathbb{E}_{\mathbf{C}}[p(\mathbf{A}|\mathbf{C}, \mathbf{F}, \mathbf{D})] && (3.5)\\
&= \mathbb{E}_{\mathbf{F}}\left[\log \prod_{j=1}^{n}\sum_{k=1}^{\ell} p(\mathbf{C}_j = k)\prod_{i=1}^{m}\mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,\mathbf{C}_j})\right] && (3.6)\\
&= \mathbb{E}_{\mathbf{F}}\left[\sum_{j=1}^{n}\log \sum_{k=1}^{\ell}\pi_k\prod_{i=1}^{m}\mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,k})\right] && (3.7)\\
&= \mathbb{E}_{\mathbf{F}}\left[\sum_{j=1}^{n}\log \sum_{k=1}^{\ell}\gamma_{j,k}\frac{\pi_k\prod_{i=1}^{m}\mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,k})}{\gamma_{j,k}}\right] && (3.8)\\
&\geq \mathbb{E}_{\mathbf{F}}\left[\sum_{j=1}^{n}\sum_{k=1}^{\ell}\gamma_{j,k}\log \frac{\pi_k\prod_{i=1}^{m}\mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,k})}{\gamma_{j,k}}\right] && (3.9)\\
&= \mathbb{E}_{\mathbf{F}}\left[\sum_{j=1}^{n}\sum_{k=1}^{\ell}\left(\gamma_{j,k}\log \pi_k\prod_{i=1}^{m}\mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,k}) - \gamma_{j,k}\log \gamma_{j,k}\right)\right], && (3.10)
\end{aligned}$$

where Bin() represents a binomial distribution, π_k represents the mixture proportion for each cluster $k \in [\ell]$, and $\gamma_{j,k}$ denotes the posterior probability that mutation $j \in [n]$ belongs to cluster $k \in [\ell]$. The sum of the mixture proportion and the sum of posterior probabilities for all mutations j's are 1. That is, $\sum_{k=1}^{\ell}\pi_k = 1$ and $\sum_{k=1}^{\ell}\gamma_{j,k} = 1$.

Since the marginalization over $\mathbf{F}$ is computationally expensive in Eq. 3.10, we consider the maximization of the lower bound with respect to $f_{i,k}$ and other latent variables π_k and $\gamma_{j,k}$ where $i \in [m], j \in [n]$ and $k \in [\ell]$. That is,

$$\max_{f_{i,k},\pi_k,\gamma_{j,k}} \sum_{j=1}^{n}\sum_{k=1}^{\ell}\left(\gamma_{j,k}\log \pi_k \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,k}) - \gamma_{j,k}\log \gamma_{j,k}\right). \tag{3.11}$$

We apply the augmented Lagrangian method [63, 64] to solve the optimization problem. The Lagrangian function $L(\mathbf{F}, \gamma, \pi, \lambda, \alpha)$ is as follows:

$$\begin{aligned} L(\mathbf{F}, \gamma, \pi, \lambda, \alpha) = &\sum_{j=1}^{n}\sum_{k=1}^{\ell}\left(\gamma_{j,k}\log \pi_k \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,k}) - \gamma_{j,k}\log \gamma_{j,k}\right) \\ &+ \lambda\left(\sum_{k=1}^{\ell}\pi_k - 1\right) + \sum_{j=1}^{n}\alpha_j\left(\sum_{k=1}^{\ell}\gamma_{j,k} - 1\right). \end{aligned} \tag{3.12}$$

where λ and α_j for $j \in [n]$ are the Lagrangian multipliers for each constraint. We compute the gradient of $L(\mathbf{F}, \gamma, \pi, \lambda, \alpha)$ with respect to π_k, $\gamma_{j,k}$, λ and α_j.

$$\frac{\partial L}{\partial \gamma_{j,k}} = \log \pi_k \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,k}) - \log \gamma_{j,k} - 1 + \alpha_j \tag{3.13}$$

$$\frac{\partial L}{\partial \pi_k} = \sum_{j=1}^{n}\frac{\gamma_{j,k}}{\pi_k} + \lambda \tag{3.14}$$

$$\frac{\partial L}{\partial \lambda} = \sum_{k=1}^{\ell}\pi_k - 1 \tag{3.15}$$

$$\frac{\partial L}{\partial \alpha_j} = \sum_{k=1}^{\ell}\gamma_{j,k} - 1. \tag{3.16}$$

We set the gradients to 0 to obtain optimal values.

$$\gamma_{j,k} = e^{\left[\log \pi_k \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,k})\right]} e^{\alpha_j - 1} \tag{3.17}$$

$$\pi_k = -\frac{1}{\lambda}\sum_{j=1}^{n}\gamma_{j,k} \tag{3.18}$$

$$\sum_{k=1}^{\ell}\pi_k = 1 \tag{3.19}$$

$$\sum_{k=1}^{\ell}\gamma_{j,k} = 1. \tag{3.20}$$

By combining Eq. 3.17 - Eq. 3.20, we obtain the closed form solutions for updating the latent variables $\gamma_{j,k}$ and π_k as the followings,

$$\gamma_{j,k} = \frac{\pi_k \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,k})}{\sum_{l=1}^{\ell} \pi_l \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j}|d_{i,j}, f_{i,l})} \tag{3.21}$$

$$\pi_k = \frac{1}{n} \sum_{j=1}^{n} \gamma_{j,k}. \tag{3.22}$$

3.2.1 Linear programming to optimize F

It is not straightforward to apply a gradient-based method to update $\mathbf{F}$ due to the underlying sum condition. Instead, we formulate an optimization problem using linear programming to find the optimal $\mathbf{F}$. Let $L(\mathbf{F})$ denote the terms related to $\mathbf{F}$ from Eq. 3.12.

$$L(\mathbf{F}) = \sum_{j=1}^{n} \sum_{k=1}^{\ell} \gamma_{j,k} \sum_{i=1}^{m} \Big[a_{i,j} \log f_{i,k} + (d_{i,j} - a_{i,j}) \log (1 - f_{i,k}) \Big]. \tag{3.23}$$

Since $L(\mathbf{F})$ is a concave function, we approximate it using piece-wise linear functions with Δ intervals [65]. Let $x_0 < \cdots < x_\Delta$ be uniform intervals where $x_s = s/\Delta$, $x_0 = \epsilon$, and $x_\Delta = 1 - \epsilon$ for small $\epsilon > 0$. We assume that all $f_{i,k}$'s are in $[x_0, x_\Delta]$ and approximate $a_{i,j} \log f_{i,k} + (d_{i,j} - a_{i,j}) \log (1 - f_{i,k})$ by

$$\sum_{s=0}^{\Delta} \lambda_s \Big[a_{i,j} \log x_s + (d_{i,j} - a_{i,j}) \log (1 - x_s) \Big] \tag{3.24}$$

with $\lambda_s \geq 0$, $\sum_{s=0}^{\Delta} \lambda_s = 1$, and $\sum_{s=0}^{\Delta} \lambda_s x_s = f$. Combining them with the underlying sum condition, we have the following linear programming:

$$\underset{\lambda,\mathbf{F}}{\text{maximize}} \quad \sum_{i=1}^{m}\sum_{k=1}^{\ell}\sum_{s=0}^{\Delta}\lambda_{i,k,s}\sum_{j=1}^{n}\gamma_{j,k}\Big[a_{i,j}\log x_s + (d_{i,j}-a_{i,j})\log(1-x_s)\Big] \tag{3.25}$$

$$\text{subject to} \quad \sum_{s=0}^{\Delta}\lambda_{i,k,s} = 1 \qquad \forall i,k \tag{3.26}$$

$$\sum_{s=0}^{\Delta}\lambda_{i,k,s}x_s = f_{i,k} \qquad \forall i,k \tag{3.27}$$

$$f_{i,k} \geq \sum_{k' \in \text{children of } k} f_{i,k'} \qquad \forall i,k \tag{3.28}$$

$$\lambda_{i,k,s} \geq 0 \qquad \forall i,k,s \tag{3.29}$$

$$x_0 \leq f_{i,k} \leq x_\Delta \qquad \forall i,k. \tag{3.30}$$

3.2.2 EM Algorithm

We iteratively update the latent variables $\mathbf{F}$, γ, and π to maximize the lower bound of the log-likelihood in Eq. 3.11. Based on the initialization of $\mathbf{F}$ and π, we compute γ in the E-step. Then, we update $\mathbf{F}$ via the linear programming above and π using the updated γ in the M-step. Based on the updated parameters, we compute a new likelihood value. The process is terminated if the difference between two consecutive likelihood values is smaller than a small threshold value. Our expectation-maximization (EM) algorithm is shown in Algorithm 1.

Algorithm 1: EM algorithm

```
Initialize: frequency matrix $\mathbf{F}$, mixture proportion $\pi$
while not converged do
    E Step: Compute $\gamma_{j,k}$ with current parameters
    $\gamma_{j,k} = \frac{\pi_k \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j}|d_{i,j},f_{i,k})}{\sum_{l=1}^{\ell} \pi_l \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j}|d_{i,j},f_{i,l})}$
    M Step: Maximize log-likelihood with the updated $\mathbf{F}$ matrix
            and current responsibilities
    $\pi_k = \frac{1}{n} \sum_{j=1}^{n} \gamma_{j,k}$
    $f_{i,k} \leftarrow$ Linear Programming($f_{i,k}$)
    Update: log-likelihood and check convergence
```

3.3 Model Selection

We need to choose the number ℓ of clusters, which is a hyperparameter of ClusTree. The number ℓ ranges from 1 to the number n of mutations. To determine ℓ, we apply the Bayesian information criterion (BIC) [66]. Based on likelihood $\widehat{L}$, BIC is computed by the following formula.

$$\mathrm{BIC} = \lambda\, p \log(n) - 2\log(\widehat{L}) \tag{3.31}$$

where p is the number of model parameters of ClusTree, n is the number of mutations, λ is a scaling factor to control the penalty term, and $\widehat{L}$ is the maximum likelihood for given parameters. We increase λ for more dispersed data.

3.4 Implementation

ClusTree is implemented in Python 3 and is available at `https://github.com/elkebir-group/ClusTree`. Our method uses variant and total read counts, $\mathbf{A}$ and $\mathbf{D}$ respectively, as input and requires a hyperparameter ℓ to determine the number of clusters. The number of tree topologies we need to consider is determined based on the hyperparameter. The mixture model of ClusTree is not identifiable, so any permutations of clustering labels are considered as the same for a given tree topology. For each tree topology, ClusTree is enough to consider only one labeled tree due to this unidentifi-

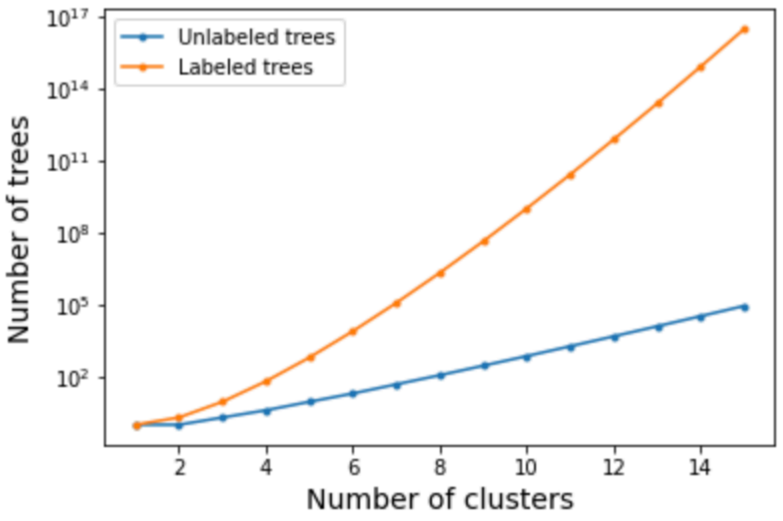


Figure 3.3: ClusTree is enough to consider the number of unlabeled trees for the given number of cluster, instead of the number of all labeled trees. This is computationally beneficial, compared to the algorithms that explore the entire space of labeled trees.

ability of a mixture model. This is beneficial since ClusTree only needs to consider the number of unlabeled trees for the given number k of clusters, instead of the number of all labeled trees, $k^{(k-1)}$ (Fig. 3.3). This can reduce the computational time significantly, compared to the methods considering all labeled trees.

We repeat our EM algorithm with a user-specified number (default: 1,000) of restarts, initializing $(\mathbf{F}, \pi)$ using k-means clustering [67] to infer mutation clusters $\mathbf{C}$, their frequency $\mathbf{F}$, and a tumor phylogeny T. For linear programming, we use the Linear Solver package of Google OR-Tools [68]. Based on the Bayesian information criterion in Eq. 3.31, we select the number ℓ^* of clusters. For the chosen number of clusters, we rank the tree topologies based on their likelihood score and refer the tree topology T^* and corresponding clustering assignment $\mathbf{C}^*$ and frequency $\mathbf{F}^*$ with the maximum likelihood.

CHAPTER 4

ANALYZING THE PERFORMANCE OF CLUSTREE

In this chapter, we analyze the performance of ClusTree thoroughly and compare the results against the performance of baseline methods. In Section 4.1, we first introduce the baseline methods to compare with ClusTree. In Section 4.2, we explain the evaluation metrics for clustering (homogeneity, completeness, and V-measure) and tree inference (ancestral pair recall and incomparable pair recall). We explain the generation process of simulation data and report the detailed results in Section 4.3, and results for real data in Section 4.4

4.1 Baseline Methods

We compared the clustering performance of ClusTree to two clustering-only methods, PyClone-VI [27] and SciClone [28], and two joint inference methods, PhyloWGS [14] and PASTRI [17].

We reported the results of PyClone-VI with a beta-binomial mode, since it had better results than a binomial mode in our experiments for both simulation and real datasets. For each run, we allowed it to have up to 10 clusters and repeated 1,000 times with different random seeds.

We allowed SciClone to have 10 clusters as the maximum number of possible clusters, which was the same condition as PyClone-VI experiments. We used a beta mixture model for its underlying distribution and followed their model selection steps.

PhyloWGS was run with the parameters that the authors suggested for MCMC sampling. For PASTRI, we used the resulting parameters of SciClone, and ran it with 1,000 different frequency samples.

For the comparison of tree inference, in addition to considering PhyloWGS and PASTRI, we included one recent method, Pairtree [18]. Since Pairtree is mainly focused on tree inference, we used each output of PyClone-VI and SciClone as its input mutation clusters. For other parameters, we followed the authors' suggestion.

4.2 Evaluation Metrics

As the clustering performance metrics, we used homogeneity, completeness, and V-measure [69]. Homogeneity (H) represents that each cluster contains only members of a single label, and completeness (C) represents all members of a given label are assigned to the same cluster. V-measure is computed as

$$\text{V-measure} = \frac{(1+\beta)\text{HC}}{\beta H + C}. \tag{4.1}$$

We fixed the hyperparameter β as 1 in this evaluation.

We evaluated the performance of tree inference of ClusTree and baseline methods by considering two metrics, ancestral pair recall and incomparable pair recall [15]. Ancestral pair recall is defined by

$$\text{Ancestral pair recall} = \frac{|A(T) \cap A(T^{*})|}{|A(T^{*})|} \tag{4.2}$$

where $A(T)$ is the set of ordered pairs of mutations that occur on distinct edges of the same branch of T, and T^{*} is the ground truth tree. Incomparable pair recall is defined by

$$\text{Incomparable pair recall} = \frac{|B(T) \cap B(T^{*})|}{|B(T^{*})|} \tag{4.3}$$

where $B(T)$ is the set of unordered pairs of mutations that occur on edges of different branches of T, and T^{*} is the ground truth tree.

4.3 Simulation Data

We generated synthetic instances designed to reflect the characteristics of bulk DNA sequencing data. Given m samples, n mutations, and ℓ clusters, we fixed the ground truth tumor phylogenetic tree topology T^{*} by choosing one tree uniformly at random from a set of trees with ℓ vertices, which was generated using Prüfer sequences [70]. Each mutation was randomly assigned to one of the ℓ clusters by considering the ground truth mixture proportion π^*, which was generated by Dirichlet$(\mathbf{1}_\ell)$ where $\mathbf{1}_\ell$ is a ℓ-dimensional vector whose all entries are one. Every cluster was enforced to have at least one mutation to avoid the empty cluster.

Then, we generated the ground truth frequency matrix $\mathbf{F}^*$, which represents the ground truth underlying frequency $f^*_{i,k}$ of cluster $k \in [\ell]$ in sequencing sample $i \in [m]$, by applying the perfect phylogeny mixture theorem, $\mathbf{F} = \mathbf{UB}$ [15]. Based on the theorem, we created $\mathbf{F}^*$ by multiplying the ground truth mixture matrix $\mathbf{U}^*$ and the ground truth phylogenetic tree matrix $\mathbf{B}^*$. Each row of $\mathbf{U}^*$ was generated by drawing from Dirichlet$(\mathbf{1}_{\ell+1})$ and removing the first element from each row to satisfy the constraints of $\mathbf{U}^*$, $\sum_{k=1}^{\ell} u^*_{ik} \leq 1$. The matrix $\mathbf{B}^*$ is a binary matrix that is corresponding to the tree topology T^* we selected.

Next, we generated variant read counts $\mathbf{A}$, based on total read counts $\mathbf{D}$ and the ground truth frequency $\mathbf{F}^*$. Each entry of $\mathbf{D}$ was generated using Poisson distribution with parameter r, where r is supposed to be the depth of synthetic sequencing data. We sampled variant read counts $\mathbf{A}$ based on a binomial distribution with parameter $\mathbf{D}$ and $\mathbf{F}^*$. We also generated $\mathbf{A}$ using a beta-binomial distribution to represent the over-dispersion of real bulk DNA sequencing data. The mean parameter of a beta-binomial distribution corresponded to the underlying frequency $\mathbf{F}^*$, and the precision parameter was fixed as a constant across all samples.

To validate the robustness of ClusTree to low-depth sequencing depths, we generated simulation data based on several depths, $r \in \{20, 30, 50\}$. For each depth, we considered the 5 different number of samples, $m \in \{1, 3, 5, 7, 9\}$. For each depth and each sample, we generated 10 instances by using different random seeds. In addition, we fixed the number of the ground truth clusters as 5. We considered different number k of clusters, $k \in \{2, \ldots, 7\}$ and its corresponding tree topologies for our experiments.

4.3.1 Results on Simulation Data based on Beta-binomial Mixtures

Fig. 4.1 (a-c) shows the results of clustering for each method with respect to the simulation data based on beta-binomial mixtures with depth 20×. ClusTree outperformed the four baselines across the number of sequencing samples with respect to three clustering metrics, homogeneity, completeness, and V-measure. ClusTree has better homogeneity (median: 0.941), completeness (median: 0.955), and V-measure (median: 0.944) than SciClone (0.660, 0.601 and 0.629, respectively), PyClone-VI (0.723, 0.883 and 0.790, respectively), PhyloWGS (0.684, 0.928 and 0.775, respectively) and PASTRI+SciClone (0.526, 0.708, and 0.605, respectively).

Regarding model selection of ClusTree, we fixed λ as 5 in Eq. 3.31. ClusTree found the number (median: 5) of clusters more correctly than other baseline methods. PyClone-VI (median: 4), PhyloWGS (median: 3), and PASTRI+SciClone (median: 4) underestimated the number of clusters, while SciClone (median: 6) overestimated it in Fig. 4.2 (a). Even though the model selection of PASTRI should be the same as that of the chosen clustering method, the actual number of clusters of PASTRI tended to be less, especially for low depth and over-dispersed data based on a beta-binomial mixture model. Since the cluster centers estimated by SciClone were hard to distinguish for those data, the frequency samples drawn from the clusters were also difficult to tell. During the inference process of PASTRI, some of the clusters were considered to represent the same cluster, and the final number of clusters became smaller than the number of clusters from SciClone. This incorrect model selection deteriorated the performance of clustering for baseline methods.

We also checked the number of successful runs each method returned as a solution in Fig. 4.2 (b). PASTRI failed to return a solution for some instances, while other methods usually provided successful runs for every instance. PASTRI inferred the possible phylogenetic trees by enumerating the frequency values sampled using the SciClone parameters. The estimation process of frequency by SciClone was not designed to satisfy the sum condition based on the underlying evolutionary tree. Thus, PASTRI could have no solution trees for any permutations of sampled frequencies. The increase of the sequencing samples worsened this issue, and the number of tree solutions became smaller.

We compared the performance of tree inference to PhyloWGS, Pairtree, and PASTRI in Fig. 4.3 (a-b). For Pairtree, we considered two different input mutation clusters based on SciClone and PyClone-VI. Across the depths and number of samples, ClusTree has better ancestral pair recall (median: 0.687) and incomparable pair recall (median: 0.151) than PhyloWGS (median: 0.510 and 0.0, respectively), Pairtree+PyClone-VI (median: 0.359 and 0.0, respectively), Pairtree+SciClone (median: 0.563 and 0.0, respectively) and PASTRI+SciClone (median: 0.075 and 0.0, respectively).

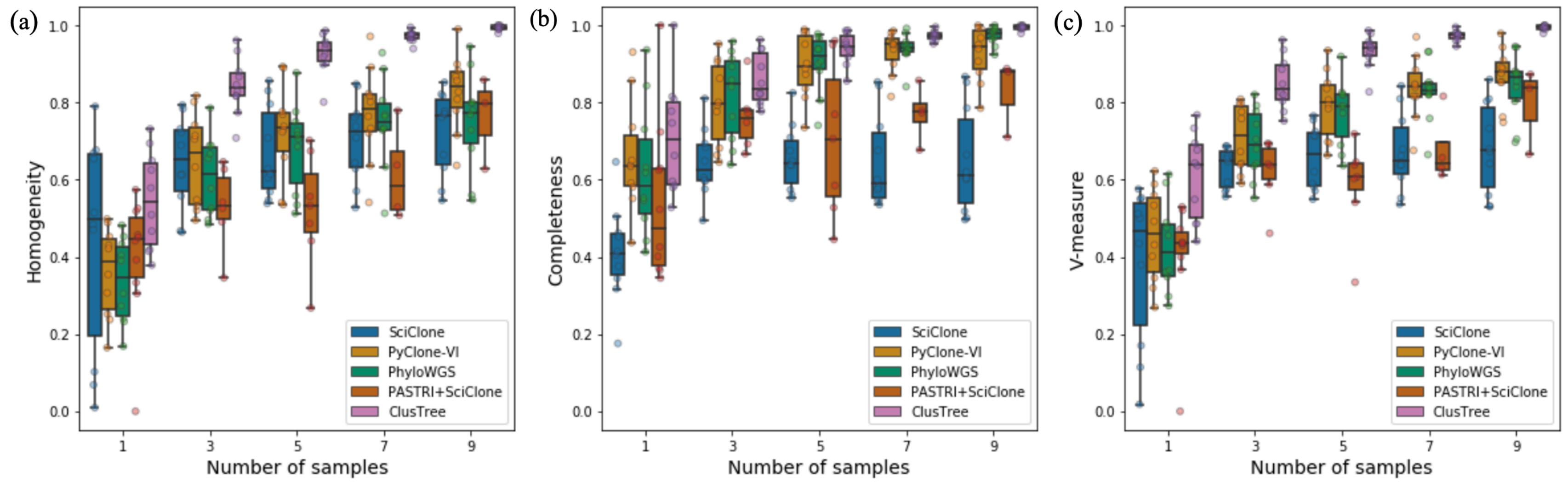
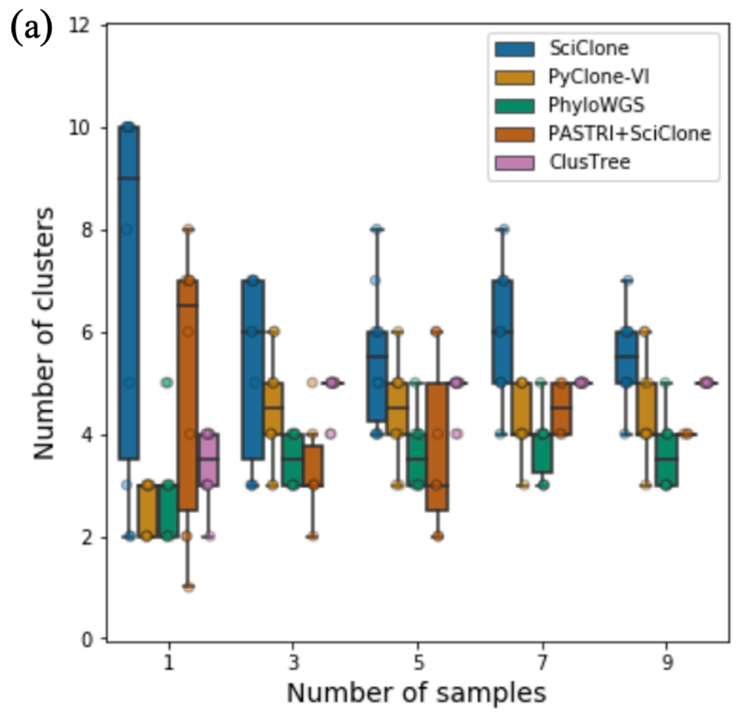


Figure 4.1: ClusTree outperforms the baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone, Pairtree+SciClone and Pairtree+PyClone-VI) with respect to clustering for simulation data generated based on a beta-binomial mixture model with depth 20×. (a-c) Clustering performance of each method with respect to homogeneity, completeness, and V-measure, respectively.



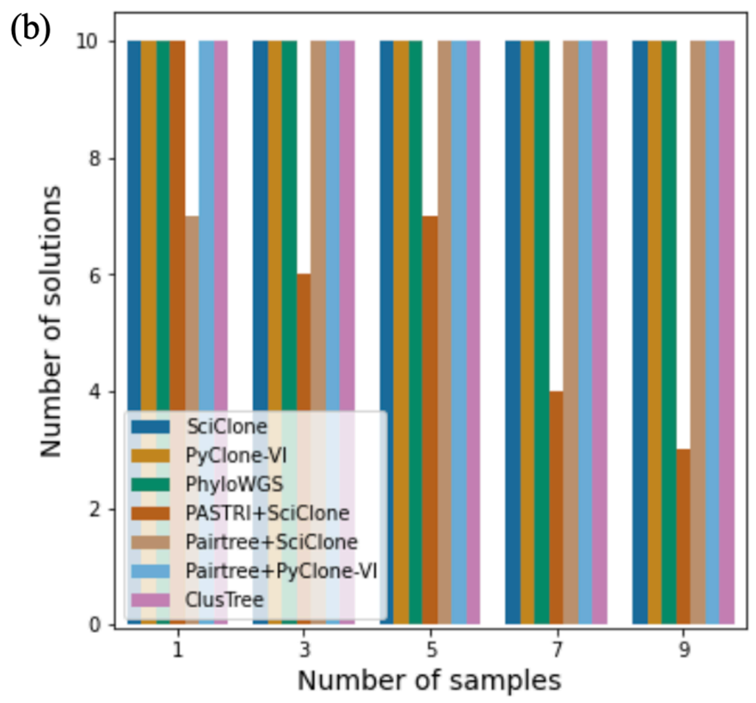


Figure 4.2: ClusTree detects the number of clusters more correctly, compared to the baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone). PASTRI has many failures to find a solution, while other methods successfully find a solution for every run. (a) Number of clusters each method found via model selection. (b) Number of successful runs each method found.

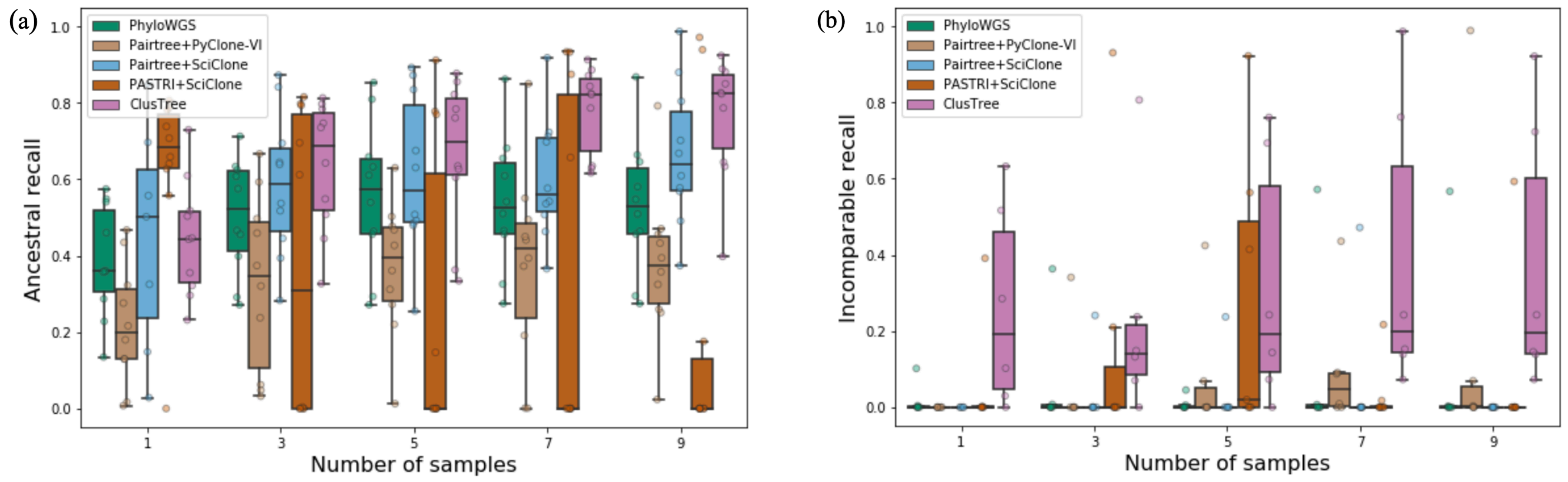


Figure 4.3: ClusTree outperforms the baseline methods (PhyloWGS, PASTRI+SciClone, Pairtree+SciClone and Pairtree+PyClone-VI) with respect to tree inference for simulation data generated based on a beta-binomial mixture model with depth 20×. (a-b) Performance of tree inference of each method with respect to ancestral pair recall and incomparable pair recall, respectively.

We show the results of clustering, model selection, number of successful runs, and tree inference using simulation data based on beta-binomial mixtures with depth 30×.

Fig. 4.4 (a-c) shows the results of clustering for each method with respect to the simulation data based on beta-binomial mixtures with depth 30×. ClusTree outperformed the four baselines across the number of sequencing samples with respect to three clustering metrics, homogeneity, completeness, and V-measure. ClusTree has better homogeneity (median: 0.968), completeness (median: 0.969), and V-measure (median: 0.989) than SciClone (0.809, 0.724 and 0.770, respectively), PyClone-VI (0.750, 0.887 and 0.818, respectively), PhyloWGS (0.709, 0.953 and 0.812, respectively) and PASTRI+SciClone (0.580, 0.706, and 0.588, respectively).

Regarding model selection of ClusTree, we fixed λ as 5 in Eq. 3.31. ClusTree found the number (median: 5) of clusters more correctly than other baseline methods. PyClone-VI (median: 4), PhyloWGS (median: 3), and PASTRI+SciClone (median: 4) underestimated the number of clusters, while SciClone (median: 6) overestimated it in Fig. 4.5 (a). Like the previous depth 20× case, this incorrect model selection deteriorated the performance of clustering for baseline methods.

We also checked the number of successful runs each method returned as a solution in Fig. 4.5 (b). For this depth, PASTRI still failed to return a solution for some instances, while other methods provided successful runs for every instance. This issue was worsened as the number of sequencing samples increased, and the number of tree solutions became smaller.

We compared the performance of tree inference to PhyloWGS, Pairtree, and PASTRI in Fig. 4.6 (a-b). For Pairtree, we considered two different input mutation clusters based on SciClone and PyClone-VI. Across the depths and number of samples, ClusTree has better ancestral pair recall (median: 0.666) and incomparable pair recall (median: 0.227) than PhyloWGS (median: 0.508 and 0.0, respectively), Pairtree+PyClone-VI (median: 0.334 and 0.0, respectively), Pairtree+SciClone (median: 0.582 and 0.0, respectively) and PASTRI+SciClone (median: 0.593 and 0.0, respectively).

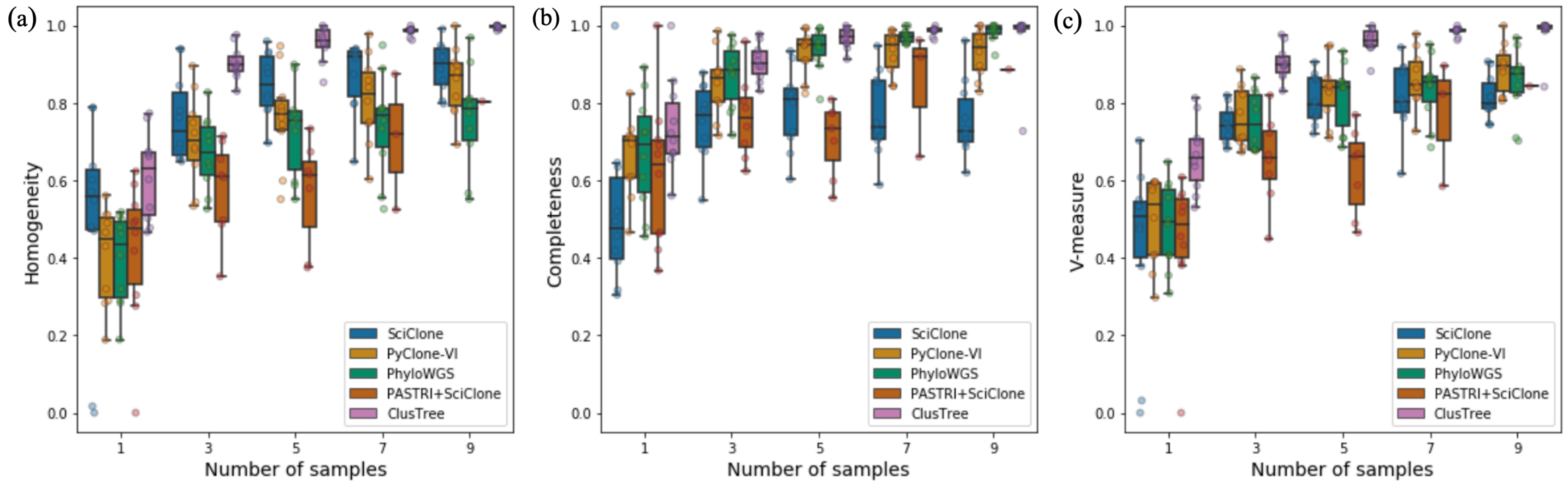


Figure 4.4: ClusTree outperforms the baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone, Pairtree+SciClone and Pairtree+PyClone-VI) with respect to clustering for simulation data generated based on a beta-binomial mixture model with depth 30×. (a-c) Clustering performance of each method with respect to homogeneity, completeness, and V-measure, respectively.

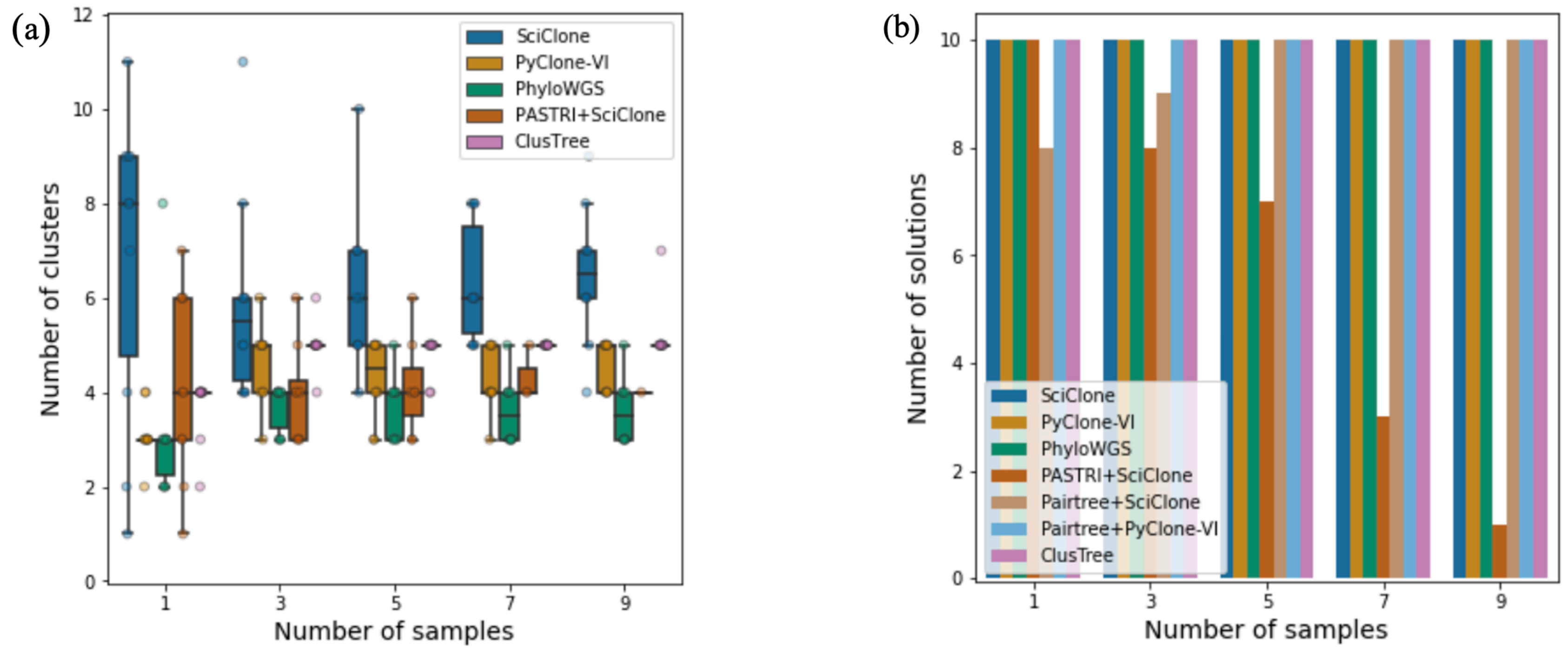
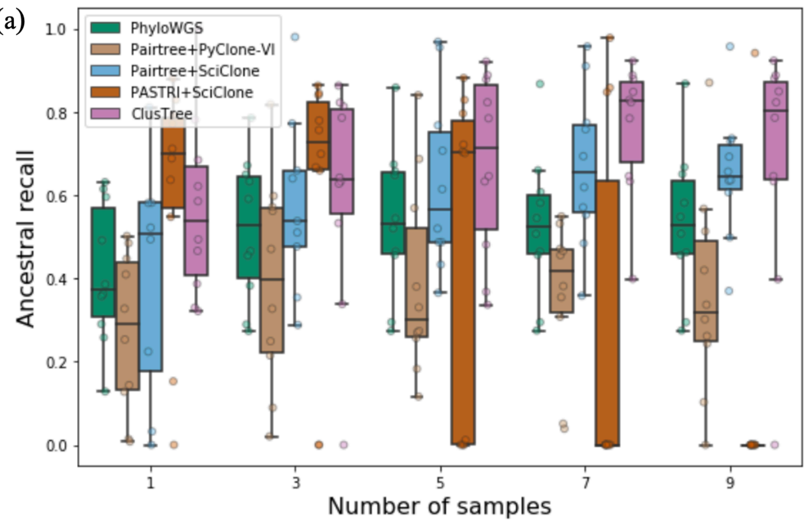


Figure 4.5: ClusTree detects the number of clusters more correctly, compared to the baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone). PASTRI has many failures to find a solution, while other methods successfully find a solution for every run. (a) Number of clusters each method found via model selection. (b) Number of successful runs each method found.

(a)



(b)

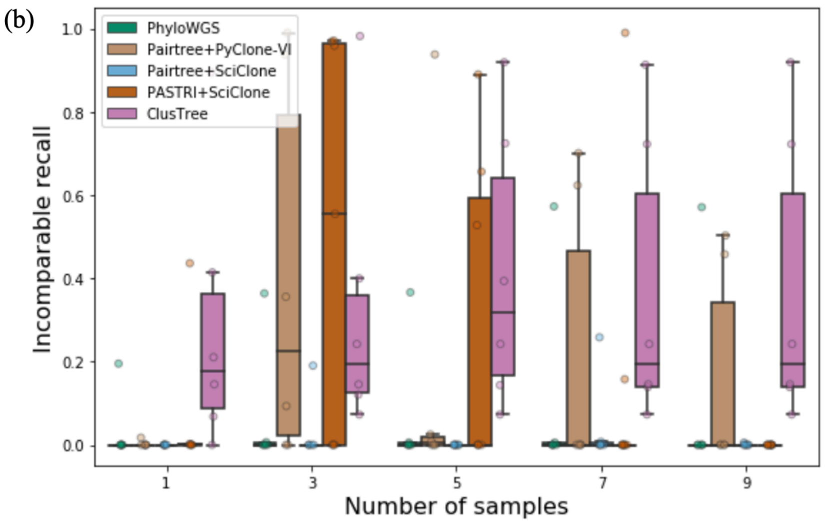


Figure 4.6: ClusTree outperforms the baseline methods (PhyloWGS, PASTRI+SciClone, Pairtree+SciClone and Pairtree+PyClone-VI) with respect to tree inference for simulation data generated based on a beta-binomial mixture model with depth 30×. (a-b) Performance of tree inference of each method with respect to ancestral pair recall and incomparable pair recall, respectively.

Last, we show the results of clustering, model selection, number of successful runs, and tree inference using simulation data based on beta-binomial mixtures with depth 50×.

Fig. 4.7 (a-c) shows the results of clustering for each method with respect to the simulation data based on beta-binomial mixtures with depth 50×. ClusTree outperformed the four baselines across the number of sequencing samples with respect to three clustering metrics, homogeneity, completeness, and V-measure. ClusTree has better homogeneity (median: 0.989), completeness (median: 0.988), and V-measure (median: 0.988) than SciClone (0.932, 0.832 and 0.874, respectively), PyClone-VI (0.808, 0.909 and 0.837, respectively), PhyloWGS (0.729, 0.973 and 0.839, respectively) and PASTRI+SciClone (0.668, 0.795, and 0.720, respectively).

Regarding model selection of ClusTree, we fixed λ as 5 in Eq. 3.31. ClusTree found the number (median: 5) of clusters more correctly than other baseline methods. PyClone-VI (median: 5), PhyloWGS (median: 4), and PASTRI+SciClone (median: 4) underestimated the number of clusters, while SciClone (median: 6) overestimated it in Fig. 4.8 (a). Like the previous depth 20× case, this incorrect model selection deteriorated the performance of clustering for baseline methods.

We also checked the number of successful runs each method returned as a solution in Fig. 4.8 (b). For this depth, PASTRI still had many failures to return a solution for some instances, while other methods gave successful runs for every instance. This issue was worsened as the number of sequencing samples increased, and the number of tree solutions became smaller.

We compared the performance of tree inference to PhyloWGS, Pairtree, and PASTRI in Fig. 4.9 (a-b). For Pairtree, we considered two different input mutation clusters based on SciClone and PyClone-VI. Across the depths and number of samples, ClusTree has better ancestral pair recall (median: 0.762) and incomparable pair recall (median: 0.195) than PhyloWGS (median: 0.486 and 0.0, respectively), Pairtree+PyClone-VI (median: 0.347 and 0.0, respectively), Pairtree+SciClone (median: 0.590 and 0.0, respectively) and PASTRI+SciClone (median: 0.549 and 0.0, respectively).

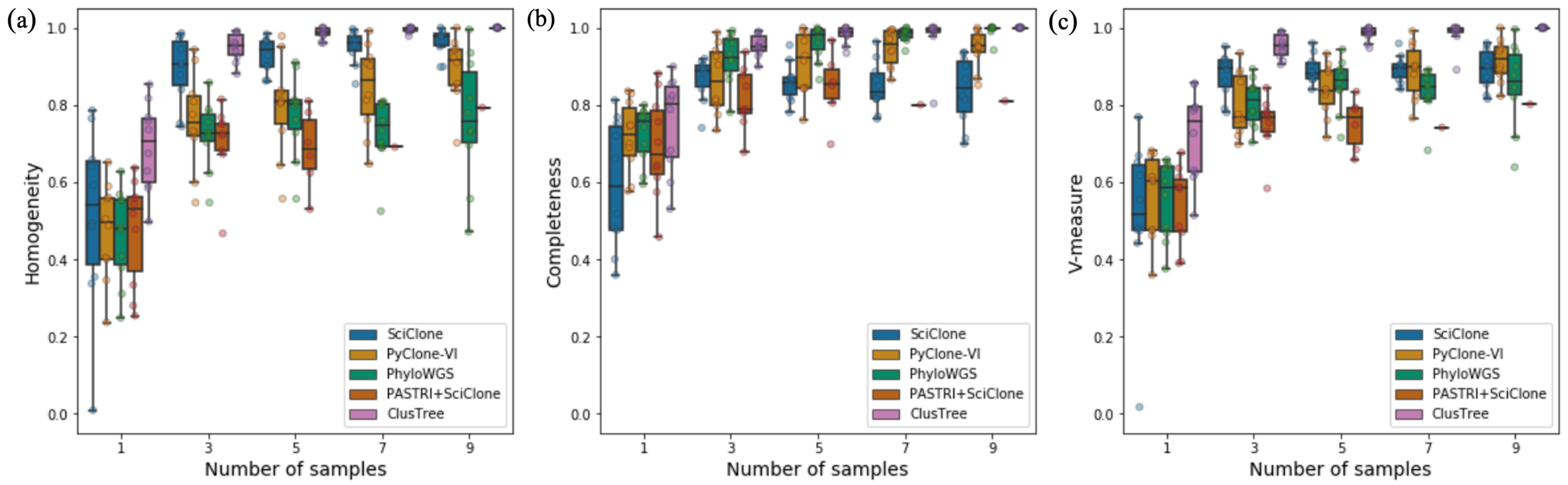


Figure 4.7: ClusTree outperforms the baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone, Pairtree+SciClone and Pairtree+PyClone-VI) with respect to clustering for simulation data generated based on a beta-binomial mixture model with depth 50×. (a-c) Clustering performance of each method with respect to homogeneity, completeness, and V-measure, respectively.

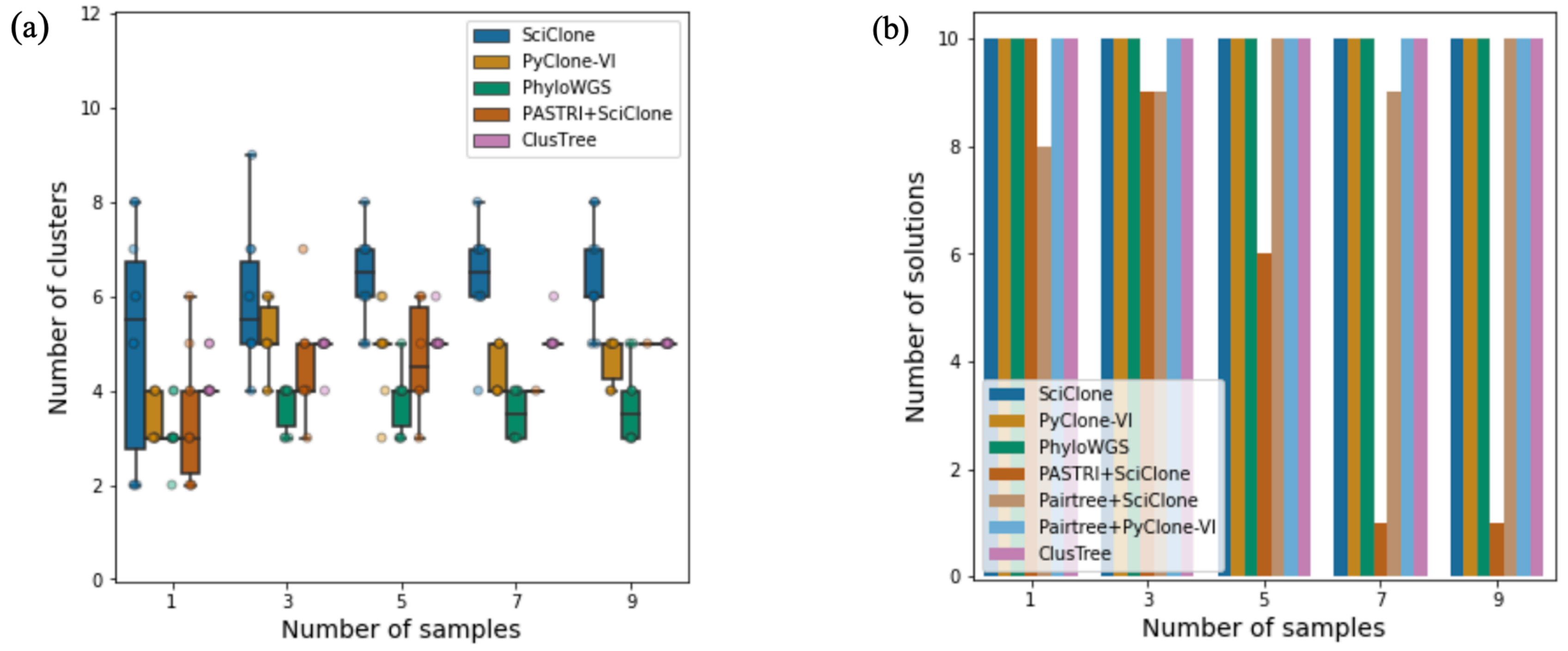


Figure 4.8: ClusTree detects the number of clusters more correctly, compared to the baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone). PASTRI has many failures to find a solution, while other methods successfully find a solution for every run. (a) Number of clusters each method found via model selection. (b) Number of successful runs each method found.

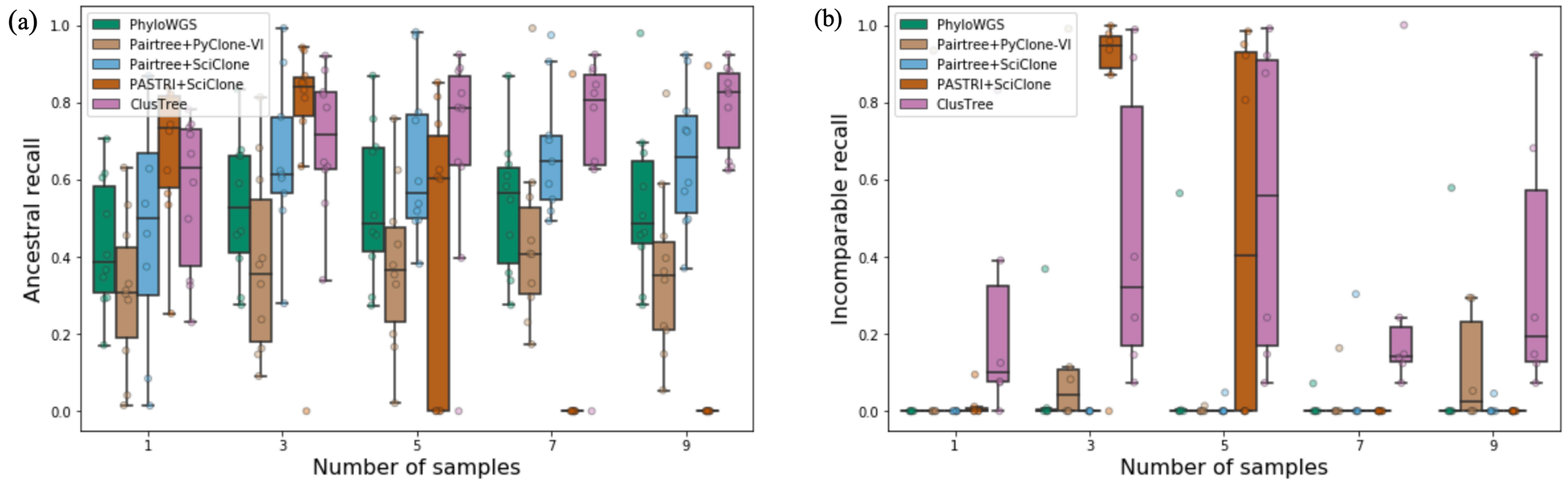


Figure 4.9: ClusTree outperforms the baseline methods (PhyloWGS, PASTRI+SciClone, Pairtree+SciClone and Pairtree+PyClone-VI) with respect to tree inference for simulation data generated based on a beta-binomial mixture model with depth 50×. (a-b) Performance of tree inference of each method with respect to ancestral pair recall and incomparable pair recall, respectively.

4.3.2 Results on Simulation Data based on Binomial Mixtures

Fig. 4.10 (a-c) shows the results of clustering for each method with respect to the simulation data based on binomial mixtures with depth 20×. ClusTree outperformed or was comparable to the four baselines (SciClone, PyClone-VI, PhyloWGS and PASTRI+SciClone) across the number of sequencing samples with respect to three clustering metrics, homogeneity, completeness, and V-measure. ClusTree and SciClone have better homogeneity (median: 1.0), completeness (median: 1.0), and V-measure (median: 1.0) than PyClone-VI (0.813, 1.0, and 0.897, respectively), PhyloWGS (0.696, 1.0 and 0.821, respectively) and PASTRI+SciClone (0.934, 0.990, and 0.960, respectively).

Regarding model selection of ClusTree, we fixed λ as 1 in Eq. 3.31. ClusTree (median: 5), SciClone (median: 5), and PASTRI+SciClone (median: 5) found the correct number of clusters. PyClone-VI (median: 4) and PhyloWGS (median: 3) underestimated the number of clusters in Fig. 4.11 (a). We also checked the number of successful runs each method returned as a solution in Fig. 4.11 (b). PASTRI failed to return a solution for some instances, while other methods usually provided successful runs for every instance. In PASTRI, the frequency samples drawn from SciClone parameters were not able to satisfy the sum condition for some instances. For those cases, PASTRI could not return proper solutions.

We compared the performance of tree inference to PhyloWGS, Pairtree, and PASTRI in Fig. 4.12 (a-b). For Pairtree, we considered two different input mutation clusters based on SciClone and PyClone-VI. Across the depths and number of samples, ClusTree has better incomparable pair recall (median: 0.195) than PhyloWGS (median: 0.0), Pairtree+PyClone-VI (median: 0.0), Pairtree+SciClone (median: 0.0), PASTRI+SciClone (median: 0.0). For ancestral pair recall, ClusTree (median: 0.762) outperformed PhyloWGS (median: 0.475), Pairtree+PyClone-VI (median: 0.399), and Pairtree+SciClone (median: 0.649). PASTRI+SciClone (median: 0.773) had similar performance with ClusTree.

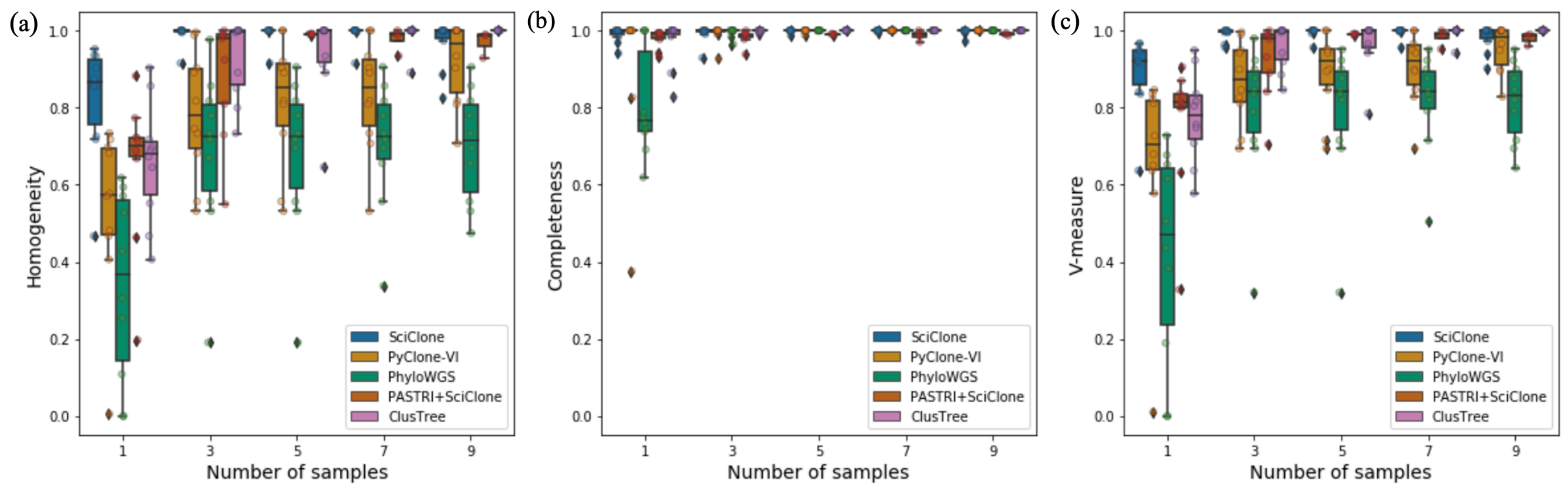
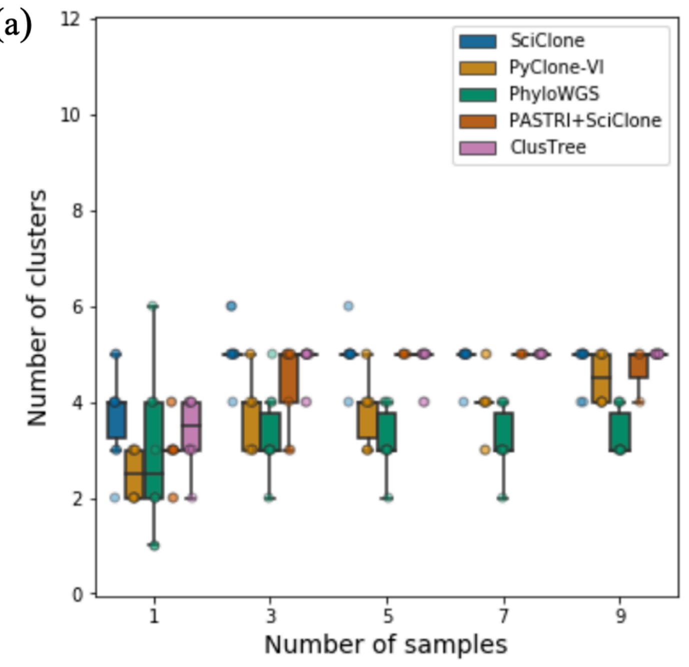


Figure 4.10: ClusTree outperforms the baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone, Pairtree+SciClone and Pairtree+PyClone-VI) with respect to clustering for simulation data generated based on a binomial mixture model with depth 20×. (a-c) Clustering performance of each method with respect to homogeneity, completeness, and V-measure, respectively.

(a)



(b)

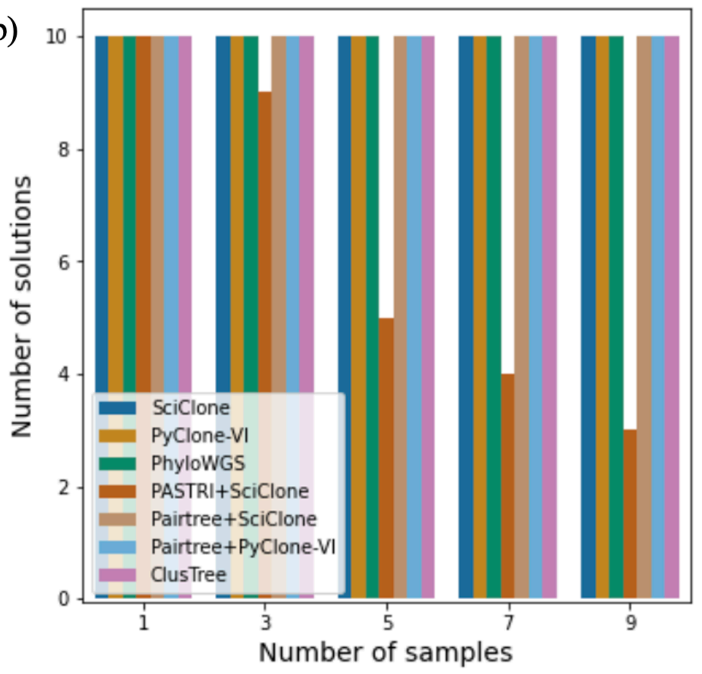
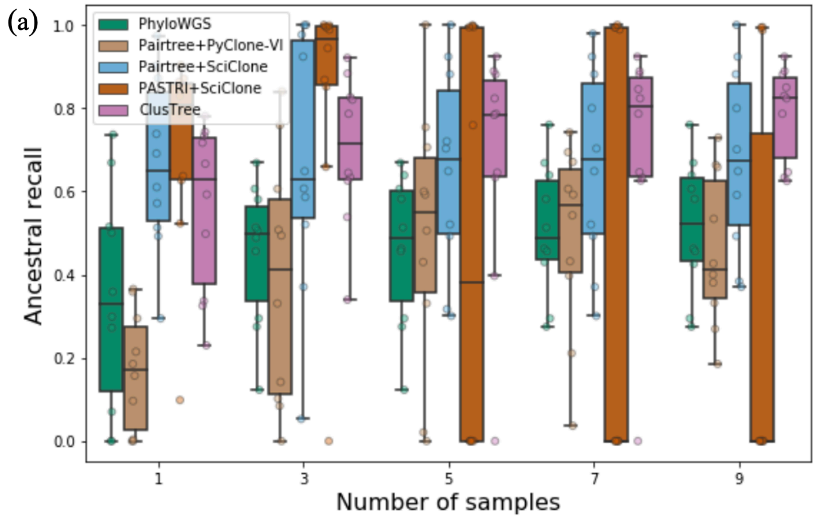


Figure 4.11: ClusTree, SciClone and PASTRI+SciClone correctly found the number of clusters, compared to PyClone-VI and PhyloWGS. PASTRI+SciClone has many failures and cannot return proper solutions, while other methods successfully find a solution for every run. (a) Number of clusters each method found via model selection. (b) Number of successful runs each method found.



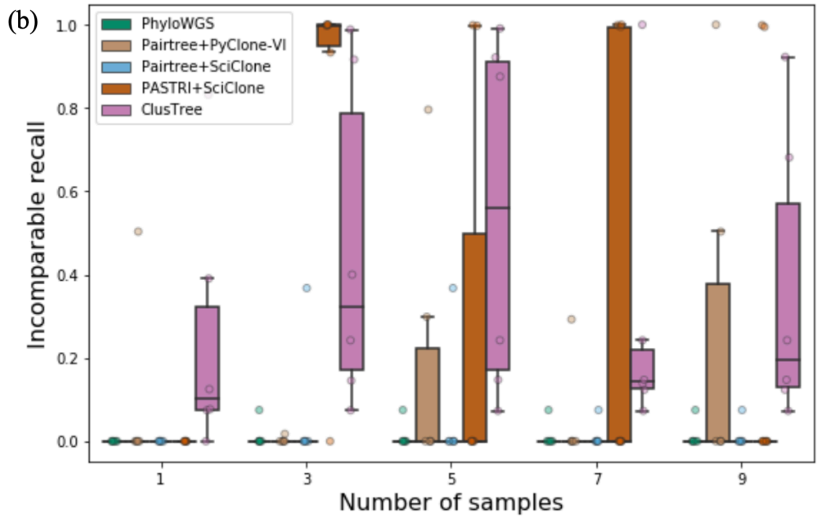


Figure 4.12: ClusTree outperforms some baseline methods (PhyloWGS, Pairtree+SciClone and Pairtree+PyClone-VI) and is comparable to PASTRI+SciClone with respect to tree inference for simulation data generated based on a binomial mixture model with depth 20×. (a-b) Performance of tree inference of each method with respect to ancestral pair recall and incomparable pair recall, respectively.

Fig. 4.13 (a-c) shows the results of clustering for each method with respect to the simulation data based on binomial mixtures with depth $30\times$. ClusTree outperformed or was comparable to the four baselines (SciClone, PyClone-VI, PhyloWGS and PASTRI+SciClone) across the number of sequencing samples with respect to three clustering metrics, homogeneity, completeness, and V-measure. ClusTree and SciClone have better homogeneity (median: 1.0), completeness (median: 1.0), and V-measure (median: 1.0) than PyClone-VI (0.817, 1.0, and 0.900, respectively), PhyloWGS (0.696, 1.0, and 0.821, respectively) and PASTRI+SciClone (0.931, 0.990, and 0.950, respectively).

Regarding model selection of ClusTree, we fixed λ as 1 in Eq. 3.31. ClusTree (median: 5), SciClone (median: 5), and PASTRI+SciClone (median: 5) found the correct number of clusters. PyClone-VI (median: 4) and PhyloWGS (median: 3) underestimated the number of clusters in Fig. 4.14 (a). We also checked the number of successful runs each method returned as a solution in Fig. 4.14 (b). PASTRI failed to return a solution for some instances, while other methods usually provided successful runs for every instance. In PASTRI, the frequency samples drawn from SciClone parameters were not able to satisfy the sum condition for some instances. For those cases, PASTRI could not return proper solutions.

We compared the performance of tree inference to PhyloWGS, Pairtree, and PASTRI in Fig. 4.15 (a-b). For Pairtree, we considered two different input mutation clusters based on SciClone and PyClone-VI. Across the depths and number of samples, ClusTree has better incomparable pair recall (median: 0.131) than PhyloWGS (median: 0.0), Pairtree+PyClone-VI (median: 0.0), Pairtree+SciClone (median: 0.0), PASTRI+SciClone (median: 0.0). For ancestral pair recall, ClusTree (median: 0.727) outperformed PhyloWGS (median: 0.507), Pairtree+PyClone-VI (median: 0.361), and Pairtree+SciClone (median: 0.674). PASTRI+SciClone (median: 0.851) had better performance than ClusTree. The returned successful runs of PASTRI+SciClone had accurate results. Still, the smaller number of solutions of PASTRI+SciClone could be issues for some instances.

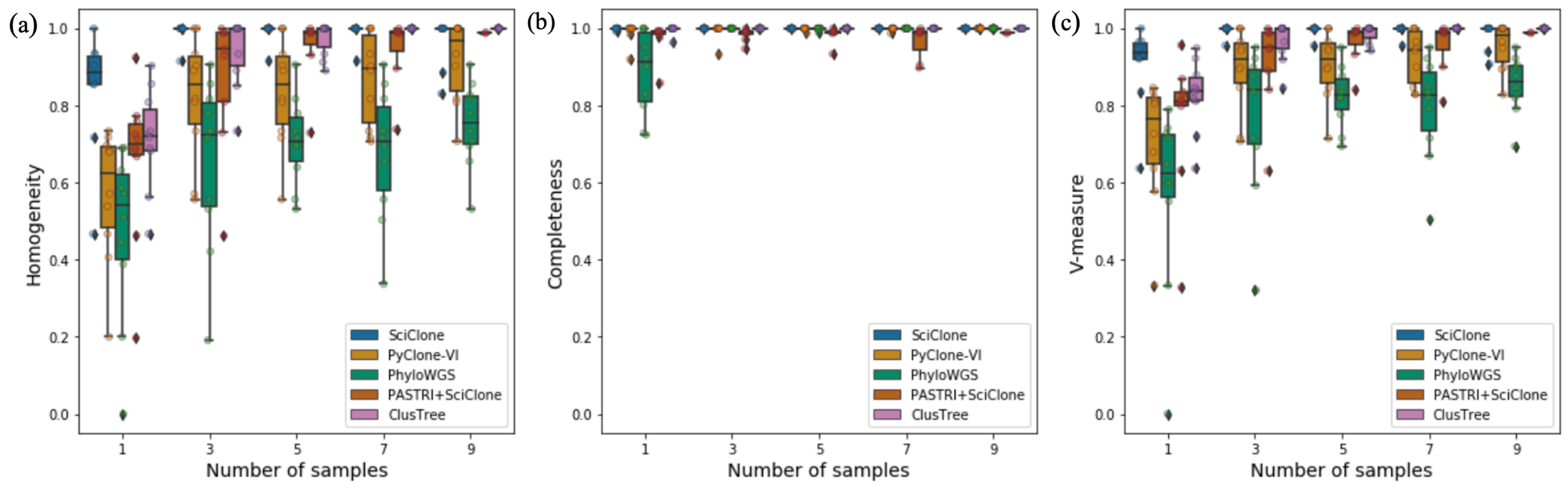
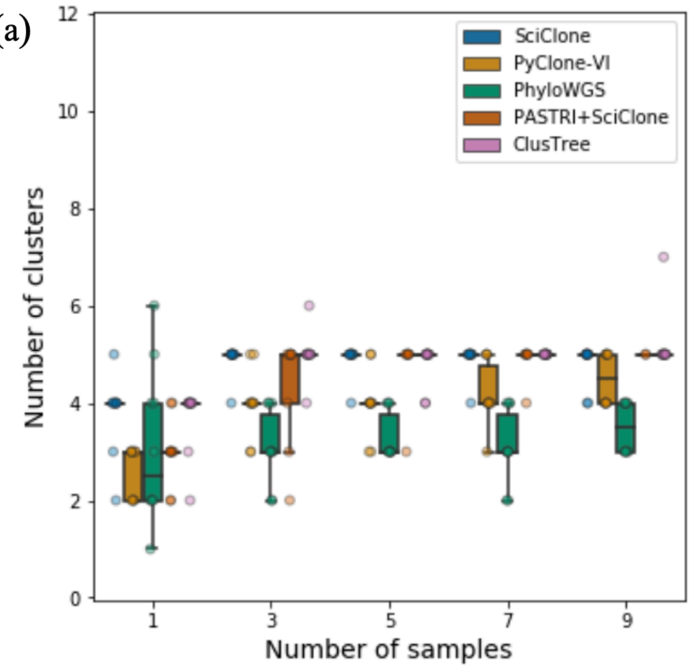


Figure 4.13: ClusTree outperforms the baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone, Pairtree+SciClone and Pairtree+PyClone-VI) with respect to clustering for simulation data generated based on a binomial mixture model with depth 30×. (a-c) Clustering performance of each method with respect to homogeneity, completeness, and V-measure, respectively.

(a)



(b)

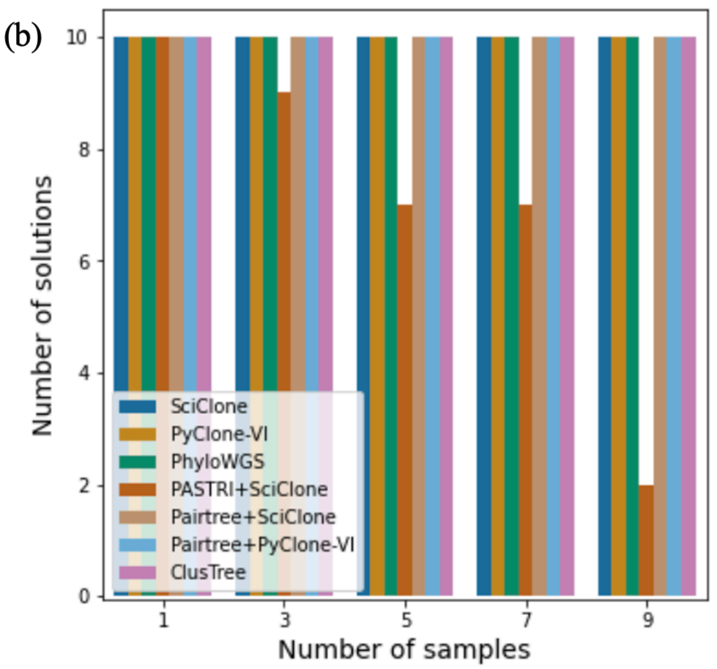
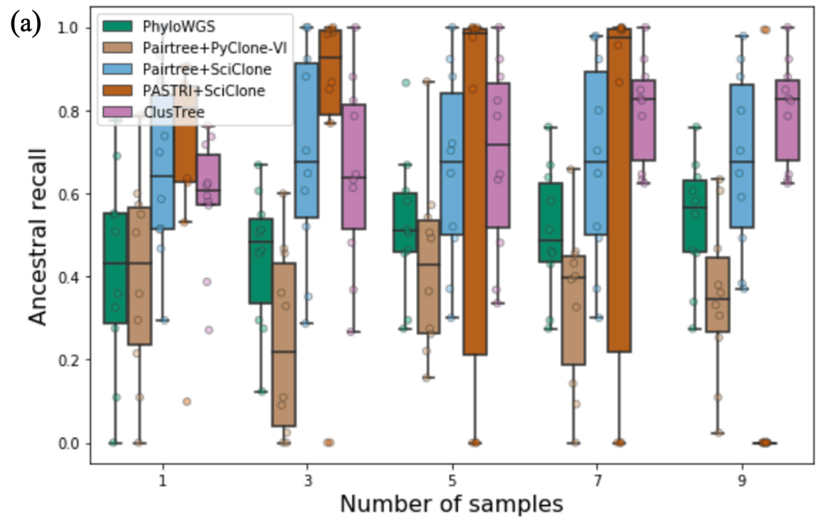


Figure 4.14: ClusTree, SciClone and PASTRI+SciClone correctly found the number of clusters, compared to PyClone-VI and PhyloWGS. PASTRI+SciClone has many failures and cannot return proper solutions, while other methods successfully find a solution for every run. (a) Number of clusters each method found via model selection. (b) Number of successful runs each method found.



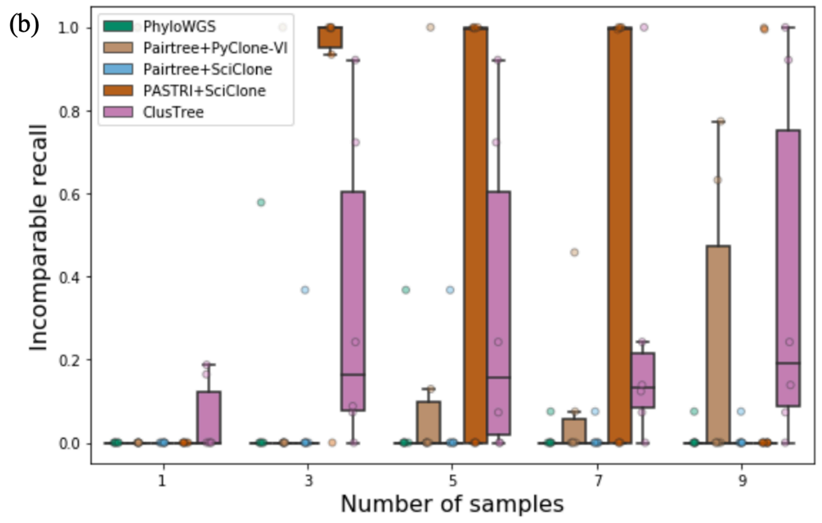


Figure 4.15: ClusTree outperforms some baseline methods (PhyloWGS, Pairtree+SciClone and Pairtree+PyClone-VI) and is comparable to PASTRI+SciClone with respect to tree inference for simulation data generated based on a binomial mixture model with depth 30×. (a-b) Performance of tree inference of each method with respect to ancestral pair recall and incomparable pair recall, respectively.

Fig. 4.16 (a-c) shows the results of clustering for each method with respect to the simulation data based on binomial mixtures with depth 50×. ClusTree outperformed or was comparable to the four baselines (SciClone, PyClone-VI, PhyloWGS and PASTRI+SciClone) across the number of sequencing samples with respect to three clustering metrics, homogeneity, completeness, and V-measure. ClusTree and SciClone have better homogeneity (median: 1.0), completeness (median: 1.0), and V-measure (median: 1.0) than PyClone-VI (0.860, 1.0, and 0.924, respectively), PhyloWGS (0.710, 1.0, and 0.825, respectively) and PASTRI+SciClone (0.927, 0.990, and 0.951, respectively).

Regarding model selection of ClusTree, we fixed λ as 1 in Eq. 3.31. ClusTree (median: 5), SciClone (median: 5), and PASTRI+SciClone (median: 5) found the correct number of clusters. PyClone-VI (median: 4) and PhyloWGS (median: 3) underestimated the number of clusters in Fig. 4.17 (a). We also checked the number of successful runs each method returned as a solution in Fig. 4.17 (b). PASTRI failed to return a solution for some instances, while other methods usually provided successful runs for every instance. In PASTRI, the frequency samples drawn from SciClone parameters were not able to satisfy the sum condition for some instances. For those cases, PASTRI could not return proper solutions.

We compared the performance of tree inference to PhyloWGS, Pairtree, and PASTRI in Fig. 4.18 (a-b). For Pairtree, we considered two different input mutation clusters based on SciClone and PyClone-VI. Across the depths and number of samples, ClusTree has better incomparable pair recall (median: 0.131) than PhyloWGS (median: 0.0), Pairtree+PyClone-VI (median: 0.0), Pairtree+SciClone (median: 0.0), PASTRI+SciClone (median: 0.0). For ancestral pair recall, ClusTree (median: 0.787) outperformed PhyloWGS (median: 0.552), Pairtree+PyClone-VI (median: 0.444), and Pairtree+SciClone (median: 0.679). PASTRI+SciClone (median: 0.857) had better performance than ClusTree. The returned successful runs of PASTRI+SciClone had accurate results. Still, the smaller number of solutions of PASTRI+SciClone could be issues for some instances.

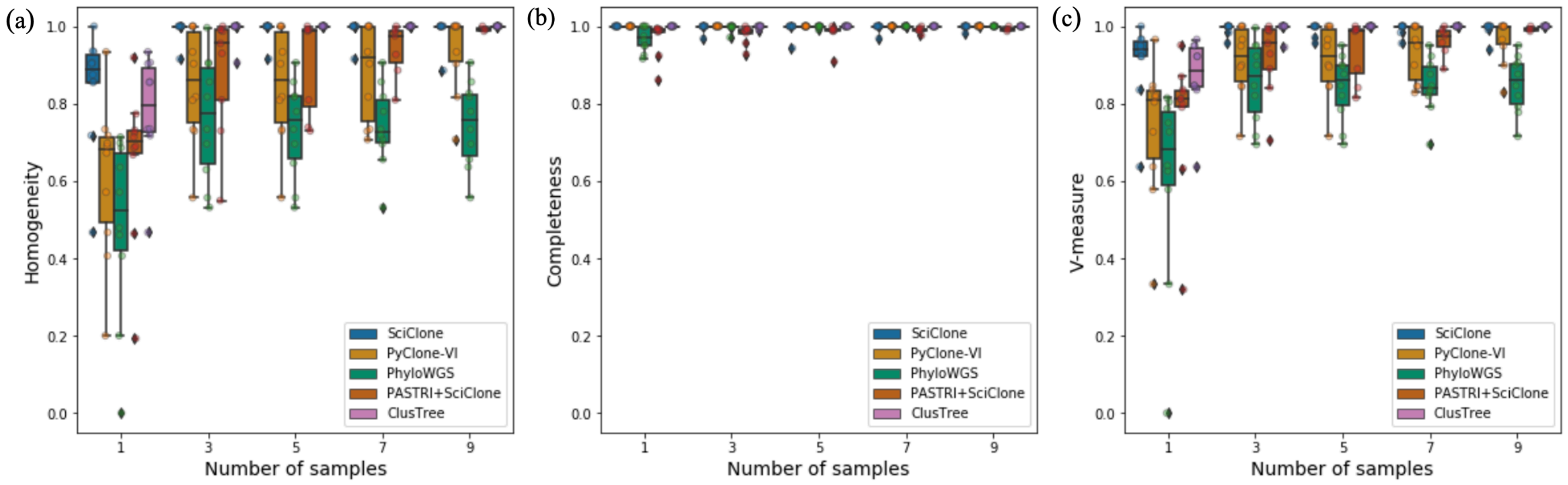
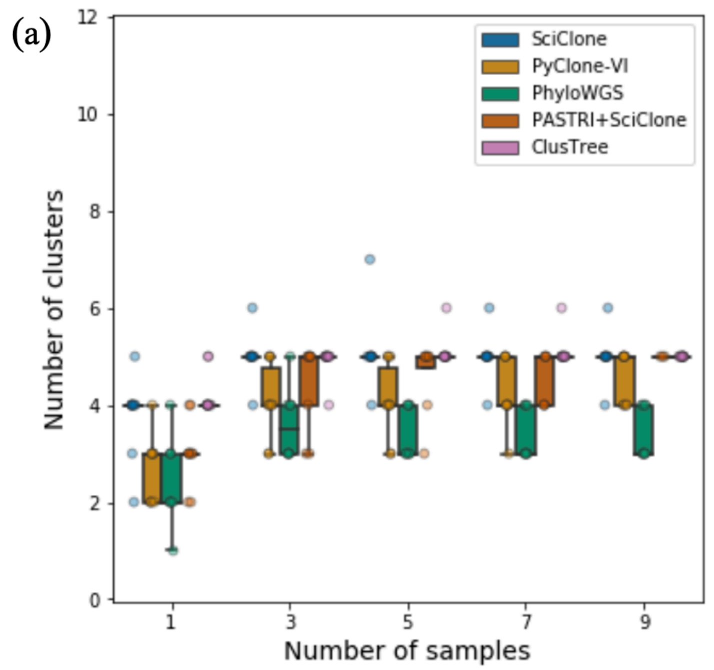


Figure 4.16: ClusTree outperforms the baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone, Pairtree+SciClone and Pairtree+PyClone-VI) with respect to clustering for simulation data generated based on a binomial mixture model with depth 50×. (a-c) Clustering performance of each method with respect to homogeneity, completeness, and V-measure, respectively.



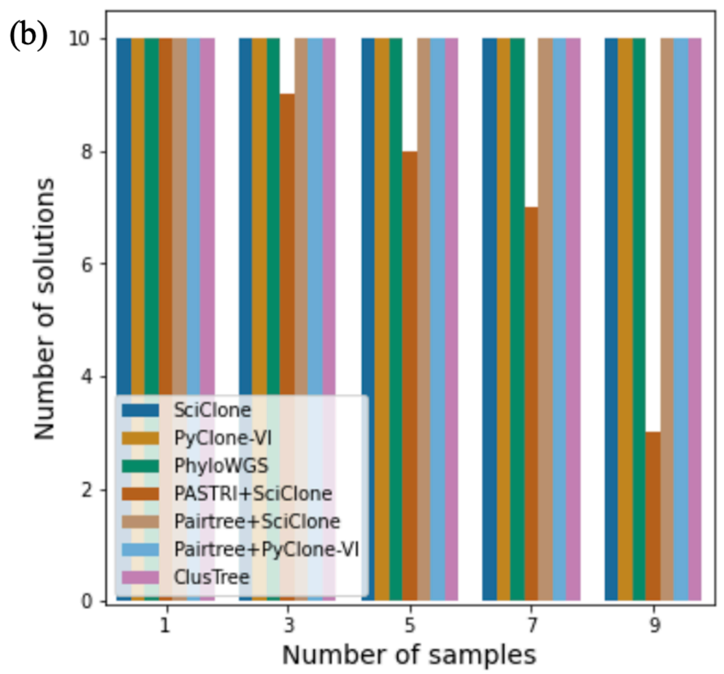


Figure 4.17: ClusTree, SciClone and PASTRI+SciClone correctly found the number of clusters, compared to PyClone-VI and PhyloWGS. PASTRI+SciClone has many failures and cannot return proper solutions, while other methods successfully find a solution for every run. (a) Number of clusters each method found via model selection. (b) Number of successful runs each method found.

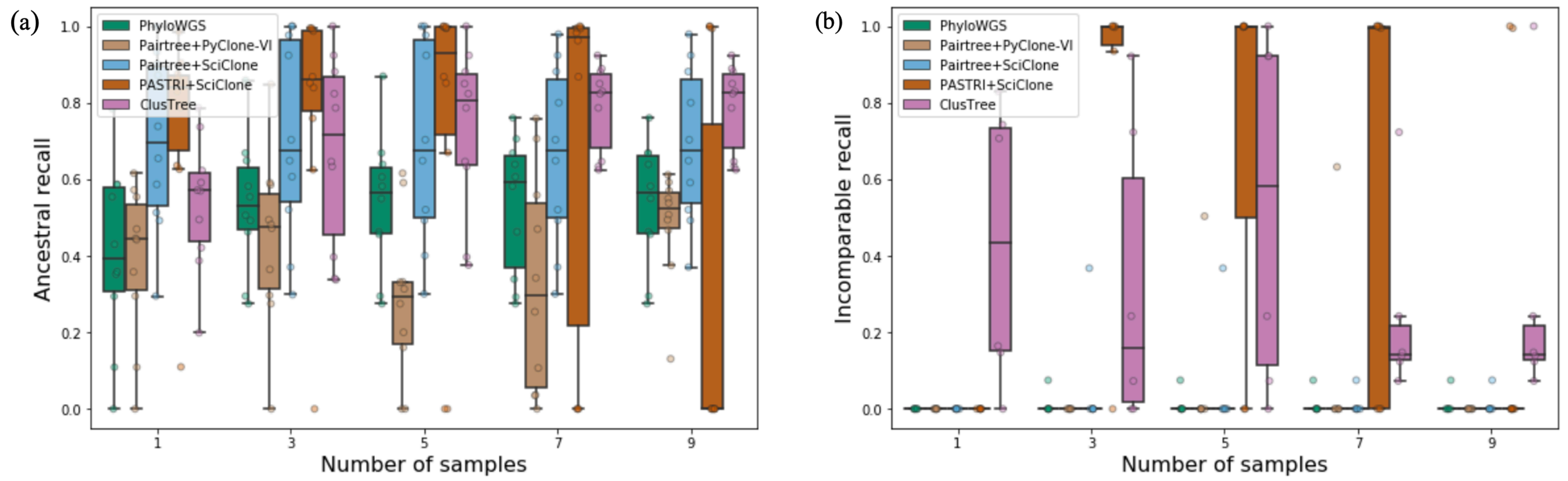


Figure 4.18: ClusTree outperforms some baseline methods (PhyloWGS, Pairtree+SciClone and Pairtree+PyClone-VI) and is comparable to PASTRI+SciClone with respect to tree inference for simulation data generated based on a binomial mixture model with depth 50×. (a-b) Performance of tree inference of each method with respect to ancestral pair recall and incomparable pair recall, respectively.

4.4 Real Data

We validated ClusTree on the dataset obtained from Griffith et al. [7], which provided variant and total read counts for the primary and relapse sample from bulk DNA sequencing of an acute myeloid leukemia (AML) patient. This dataset contains multiple types of sequencing data, including whole genome sequencing, exome sequencing, and targeted sequencing. The authors combined the different types of data after they refined each dataset through the filtering and manual review process. The polished dataset contains 1,343 single nucleotide variants (SNVs) whose depth is higher than 1,000$\times$ on average for the both samples.

We applied ClusTree and baseline methods (SciClone, PyClone-VI, PhyloWGS, PASTRI+SciClone, Pairtree+SciClone, and Pairtree+PyClone-VI) to the AML dataset. We excluded PhyloWGS from the baseline since its output had only one big cluster including all the mutations of the dataset. The clustering results of SciClone, PyClone-VI, PASTRI+SciClone, and ClusTree were almost identical except only a few mutations and found 6 clusters in Fig. 4.19. In a quantitative manner, as we fixed the result of one method as the ground truth clustering labels, we computed the clustering performance of the other methods with respect to the three metrics, homogeneity, completeness, and V-measure in Table 4.1 - Table 4.3, respectively. Each row of the tables represents the fixed ground truth labels from one method, and based on this, we found the performance of the other methods in each column. Considering the performance from the tables, we can see that the four methods have almost the same clustering results.

Table 4.1: The four methods have almost the same clustering results with respect to homogeneity. The performance is computed using the ground truth labels obtained from the result of a fixed method shown in each row.

Methods	SciClone	PyClone-VI	PASTRI+SciClone	ClusTree
SciClone	1.0	0.982	0.963	0.980
PyClone-VI	0.983	1.0	0.959	0.989
PASTRI+SciClone	0.955	0.950	1.0	0.960
ClusTree	0.980	0.989	0.968	1.0

Table 4.2: The four methods have almost the same clustering results with respect to completeness. The performance is computed using the ground truth labels obtained from the result of a fixed method shown in each row.

Methods	SciClone	PyClone-VI	PASTRI+SciClone	ClusTree
SciClone	1.0	0.983	0.955	0.980
PyClone-VI	0.982	1.0	0.950	0.989
PASTRI+SciClone	0.963	0.959	1.0	0.968
ClusTree	0.980	0.989	0.960	1.0

Table 4.3: The four methods have almost the same clustering results with respect to V-measure. The performance is computed using the ground truth labels obtained from the result of a fixed method shown in each row.

Methods	SciClone	PyClone-VI	PASTRI+SciClone	ClusTree
SciClone	1.0	0.983	0.959	0.980
PyClone-VI	0.983	1.0	0.954	0.989
PASTRI+SciClone	0.959	0.954	1.0	0.964
ClusTree	0.980	0.989	0.964	1.0

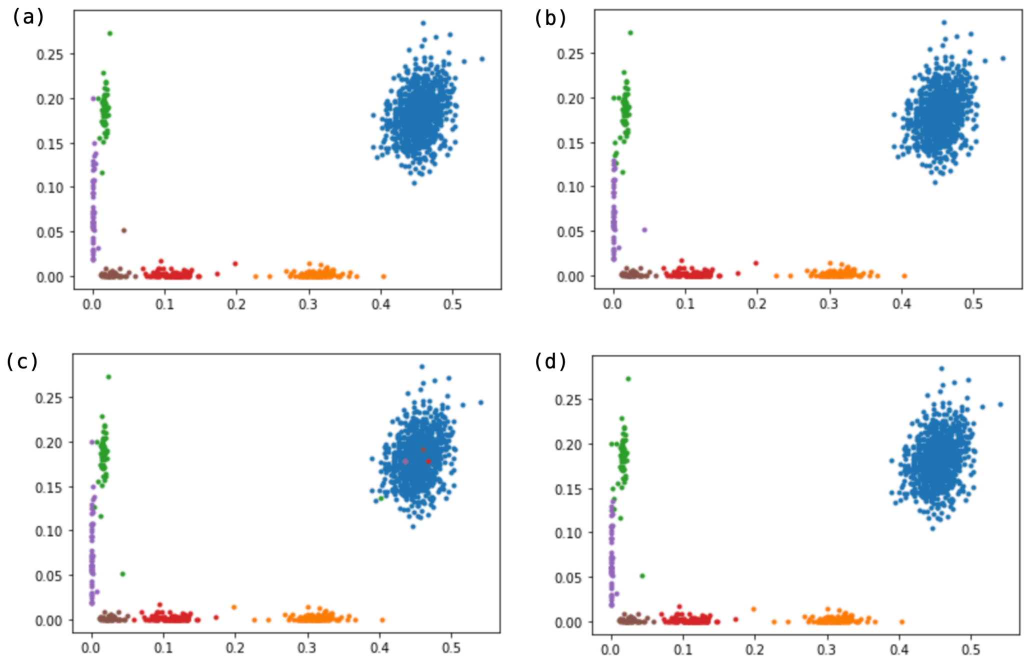


Figure 4.19: The figure shows the scatter plots for clustering results for original high-depth real data of (a) SciClone, (b) PyClone-VI, (c) PASTRI+SciClone, and (d) ClusTree. Except some boundary points, the four methods have almost the same results.

In addition, we compared the phylogenetic trees each method inferred in Fig. 4.20. Griffith et al. [7] first applied SciClone to the AML dataset to find the clustering assignment. Based on the clustering result, the authors applied ClonEvol [38] to obtain tumor phylogenetic trees, and they reported 5 different trees. The set of trees ClusTree detected included the 5 trees and additional 2 trees. PASTRI+SciClone had two overlapped trees with Griffith et al. [7] and two different trees. Pairtree+SciClone had one common tree with Griffith et al. [7], and additionally, it found trees whose germline vertex had multiple child vertices, which were not found from other algorithms. Pairtree+PyClone-VI did not detect any shared tree with Griffith et al. [7].

The color of each vertex is corresponding to the color in the scatter plot in Fig. 4.19. According to the results of Griffith et al. [7], the blue vertex is a founding clone with driver mutations DNMT3A (R882H) and NPM1 (W288fs). The orange, red, and brown vertex are tumor-specific subclones. The orange cluster contains FLT3 (D885H) and IDH1 (R132H), the red cluster contains FOXP1 (e11+1) and FLT3 (ITD), and the brown cluster contains CXCL17 (N83D) as driver mutations. The green and purple vertex are corresponding to subclones enriched in the relapse sample. The green cluster contains IDH2 (R140Q), and the purple cluster contains RUNX1 (P339fs) as a driver mutation.

To evaluate the benefits of ClusTree regarding clustering SNVs and inferring trees for low-depth sequencing data, we applied donwsampling to the original high-depth data. For each mutation, we reduced the total read count d, by sampling the new total read count value from a Poisson distribution with parameter $r \in \{20, 30, 50\}$. Here, r is corresponding to the new depth of the downsampled data. Then, we drew the number d of samples without replacement by considering the number of original variant and reference read counts and generated a new variant read count. For each parameter $r \in \{20, 30, 50\}$, we generated 10 different instances using 10 different random seeds.

Using the downsampled data, we implemented ClusTree by considering different $k \in \{2, \ldots, 7\}$ and the corresponding tree topologies for each k. Then, by applying the Bayesian information criterion (BIC) where $\lambda = 1$ in Eq. 3.31 as model selection, we chose the optimal k among them. We regarded SciClone, PyClone-VI, PhyloWGS, and PASTRI+SciClone as our baseline methods. SciClone was run using the parameters that Griffith et al. [7] used

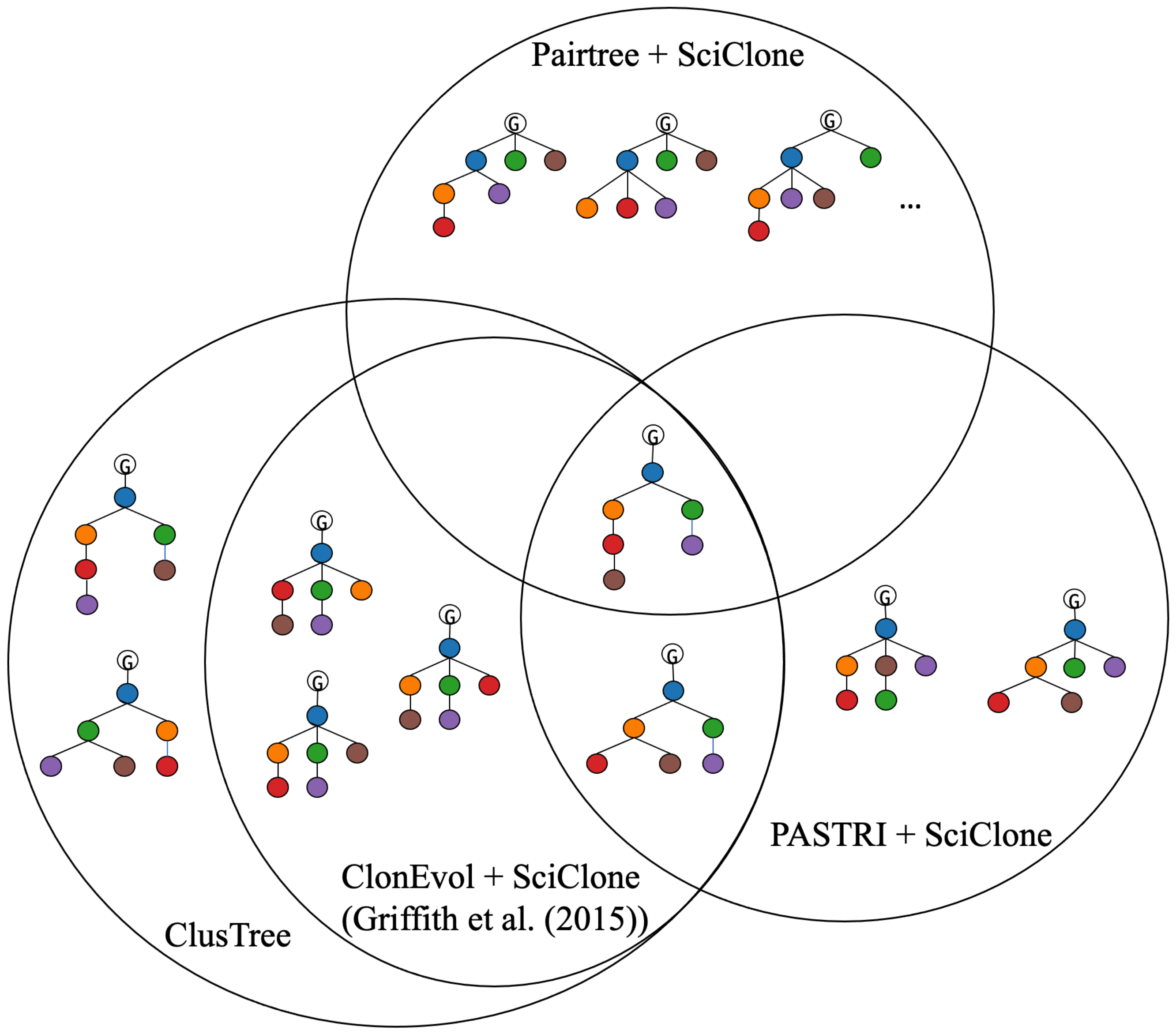


Figure 4.20: ClusTree found 7 tumor phylogenetic trees, which contained the 5 trees Griffith et al. [7] detected using ClonEvol and SciClone. PASTRI+SciClone shared two common trees with Griffith et al. [7] and found 2 different trees. Pairtree+SciClone found one common tree with Griffith et al. [7] and trees whose germline vertex had multiple child vertices.

for their implementation. PyClone-VI was applied based on a beta-binomial mixture mode with default parameters. As we applied PhyloWGS to the downsampled datasets, it provided one big cluster that contained all the mutations, so we excluded it from the baseline. PASTRI was applied to the data using the output of SciClone as input and default parameters.

We evaluated the clustering performance of each method. Since the ground truth labels were not provided, we regarded the clustering result of each method for the original high-depth data as its own ground truth labels. The ground truth labels that each method found were almost the same with each other, and all of them found 6 clusters as their model selection (see Fig. 4.19 and Table 4.1 - Table 4.3). We measured the clustering performance using three metrics, homogeneity, completeness, and V-measure.

In Fig. 4.21 (a), the results with respect to homogeneity showed that ClusTree (median: 0.653, 0.641, and 0.686, respectively) outperformed SciClone (median: 0.253, 0.140, and 0.014, respectively), PyClone-VI (median: 0.380, 0.234, and 0.197, respectively) and PASTRI+SciClone (median: 0.585, 0.366, and 0.419, respectively).

In Fig. 4.21 (b), the results with respect to completeness showed that PASTRI+SciClone (median: 0.930, 0.823, and 0.745, respectively) outperformed SciClone (median: 0.283, 0.141, and 0.070, respectively), PyClone-VI (median: 0.197, 0.132, and 0.120, respectively) and ClusTree (median: 0.692, 0.555, and 0.637, respectively).

In Fig. 4.21 (c), the results with respect to V-measure showed that ClusTree (median: 0.677, 0.589, and 0.659, respectively) outperformed SciClone (median: 0.256, 0.141, and 0.023, respectively), PyClone-VI (median: 0.259, 0.169, and 0.149, respectively) and PASTRI+SciClone (median: 0.710, 0.492, and 0.536, respectively). Since PASTRI had difficulty to distinguish similar but different clusters in this over-dispersed data, PASTRI had a smaller number of clusters in its model selection than SciClone (Fig. 4.22). Due to this reason, PASTRI had very high completeness, which made it have overestimated V-measure, even though its homogeneity was not high enough.

PyClone-VI generated more clusters than the ground truth number, some of which contained just the small number of mutations. This caused the worse clustering performance compared to ClusTree. SciClone estimated the number of clusters better compared to PyClone-VI, but due to the lack of the tree information, its performance was not satisfactory.

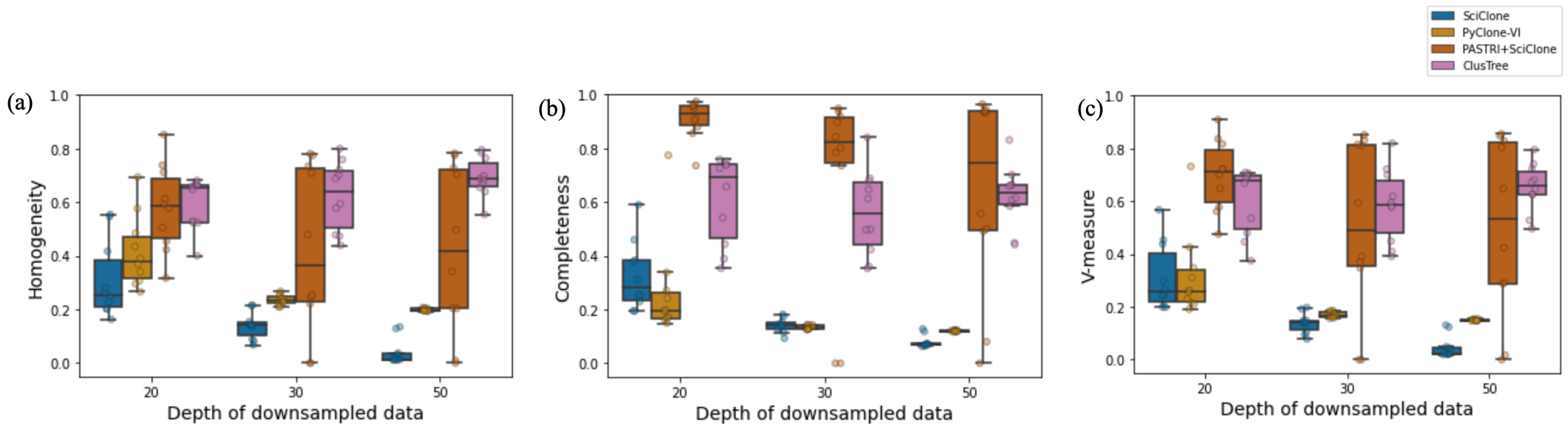


Figure 4.21: ClusTree outperformed the other baseline methods (SciClone, PyClone-VI, and PASTRI+SciClone) with respect to (a) homogeneity and (c) V-measure. PASTRI+SciClone had high completeness in (b) since it underestimated the number of clusters by merging some clusters that were hard to distinguish from each other.

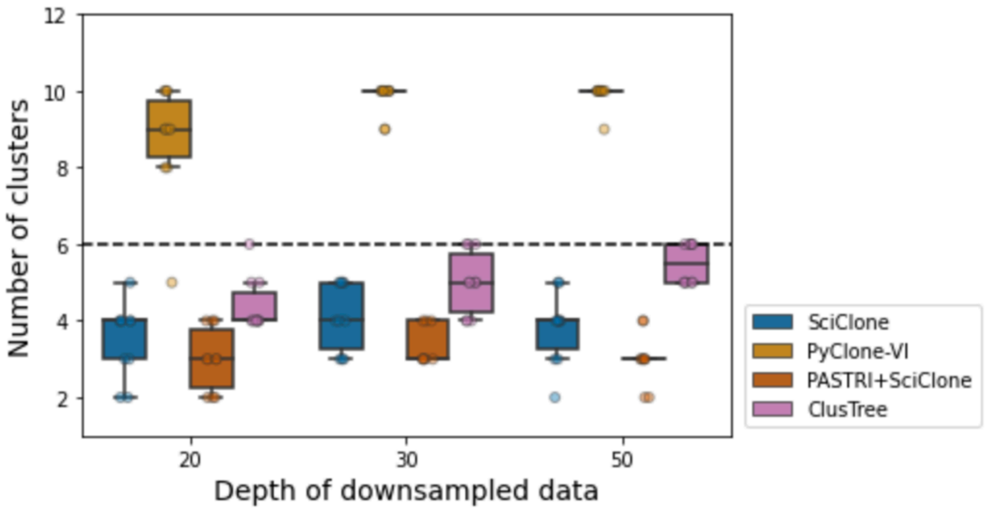


Figure 4.22: ClusTree had better model selection results, compared to other baseline method (SciClone, PyClone-VI, and PASTRI+SciClone). SciClone and PASTRI underestimated the number of clusters and PyClone-VI had more clusters than the number of the ground truth.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this dissertation, we introduced a probabilistic method to reconstruct a subclonal structure for cancer by clustering mutations, estimating the underlying frequency of each cluster, and inferring ancestral relationships between them using variant and total read count from bulk DNA sequencing data. We studied the previous work, including some methods that only focused on clustering mutations or tree inference in isolation, and other methods that considered both of them jointly. We discussed the potential weakness of the approaches and presented a new ClusTree algorithm. Speicifally, ClusTree is based on a tree-constrained binomial mixture model that employs linear programming to deal with the underlying constraints of the problem. It can reduce the computational time by only considering the number of unlabeled trees, instead of exploring the space of all labelled trees. In addition, it is a standalone approach that does not depend on the output from other existing methods.

The results showed that ClusTree was beneficial to cluster highly overdispersed low-depth sequencing data, which the previous methods such as SciClone [28] and PyClone-VI [27] had difficulty to deal with. The method also had better results than existing joint inference methods such as PhyloWGS [14] and PASTRI [17] with respect to both clustering and tree inference.

However, there are still a number of limitations in this current method and challenges to be solved in future work.

Dealing with other types of mutations While ClusTree is developed to deal with only single nucleotide variants (SNVs), it can be extended to deal with other types of mutations such as copy number variation (CNV). CNV refers to the situation where the number of copies in a specific DNA segment varies due to duplications, deletions, or other changes that can affect the a number of base pairs. In addition, larger-scale variations in the number of chromosomes such as aneuploidy [71] could also be considered for better inference in some specific circumstances. Since there is evidence that CNV or aneuploidy affects the observed read counts of SNVs [72, 73, 74], incorporating the information into subclonal reconstruction models could enhance the performance of the inference [75, 76, 77, 78].

Incorporating heterogeneous data We can consider dealing with bulk DNA sequencing data and single-cell DNA sequencing data simultaneously to add more information into inference algorithms. Even though both bulk and single-cell DNA sequencing has limitations (introduced in Section 2.2), there are joint methods to integrate both types of data and perform better than methods using data obtained from only a single source [22, 23, 62]. Beyond DNA sequencing technologies, some recent work considers single-cell RNA sequencing to uncover the heterogeneity of cancer [79, 80, 81]. They combine bulk or single-cell DNA sequencing with single-cell RNA sequencing for better statistical inference. This direction could be considered as future work.

Better representation learning of sequencing data Current approaches for the subclonal reconstruction methods depend on molecular sequencing data pre-processed based on existing tools or practical pipelines [7, 82, 83, 84, 85]. Even though these techniques have been successfully applied in many applications, the output we can use from the tools usually provides just discrete values (e.g. read count) for each mutation. Also, errors from those techniques could be propagated into the modeling procedure.

Borrowing the ideas [86, 87] from representation learning of recent machine learning and natural language processing, we could consider new continuous vector representation of mutations. We could have more modeling choices with these vector representations than with discrete input. Recent methods [88, 89, 90] tried to represent raw DNA sequencing data based on convolutional neural networks [91] or Transformer [92] and use them for other downstream tasks. Similar ideas would be applicable to the subclonal reconstruction for cancer.

Utilizing the Less Constrained Space The ClusTree algorithm is based on maximum likelihood estimation to find the clustering assignment $\mathbf{C}$ of each mutation and the underlying frequency $\mathbf{F}$ for each cluster given a tree topology T. It employs linear programming to deal with the sum condition (Eq. 2.1) in updating $\mathbf{F}$. Even though the linear programming is efficient for trees with the small number of vertices, it would be time-consuming for the larger number of vertices. To make it more efficient, we can consider the less constrained space than the space defined by the sum condition.

To do that, we revisit the perfect phylogeny theorem, $\mathbf{F} = \mathbf{UB}$ in [15]. Instead of optimizing $\mathbf{F}$, we can consider the less constrained space $\mathbf{U}$, each row U_i of which is defined in a space bounded by non-negative orthant $\mathbb{R}^{k+1}_{\geq 0}$ and a standard k-simplex $\triangle^k \subseteq \mathbb{R}^{k+1}_{\geq 0}$. We transform random variable $\mathbf{F}$ to random variable $\mathbf{U}$ in Eq. 3.10. Then, the lower bound is

$$\mathbb{E}_{\mathbf{U}}\left[\sum_{j=1}^{n}\sum_{k=1}^{\ell}\left(\gamma_{j,k}\log\pi_k\prod_{i=1}^{m}\mathrm{Bin}(a_{i,j}|d_{i,j},[\mathbf{UB}]_{i,k}) - \gamma_{j,k}\log\gamma_{j,k}\right)\right]. \tag{5.1}$$

Instead of marginalizing over $\mathbf{U}$, we consider point estimation with respect to $\mathbf{U}$. We apply the augmented Lagrangian method with the following Lagrangian function L.

$$\begin{aligned} L(\mathbf{U},\gamma,\pi,\lambda,\alpha) = &\sum_{j=1}^{n}\sum_{k=1}^{\ell}\left[\gamma_{j,k}\log\pi_k\prod_{i=1}^{m}\mathrm{Bin}(a_{i,j}|d_{i,j},[\mathbf{UB}]_{i,k}) - \gamma_{j,k}\log\gamma_{j,k}\right] \\ &+ \lambda\left(\sum_{k=1}^{\ell}\pi_k - 1\right) + \sum_{j=1}^{n}\alpha_j\left(\sum_{k=1}^{\ell}\gamma_{j,k} - 1\right). \end{aligned} \tag{5.2}$$

We compute the gradient of $L(\mathbf{U},\gamma,\pi,\lambda,\alpha)$ with respect to $u_{i,k}$, π_k, $\gamma_{j,k}$, λ

and α_j.

$$\frac{\partial L}{\partial u_{i,k}} = \sum_{j=1}^{n} \sum_{l=1}^{\ell} \gamma_{j,l} \left[\frac{a_{i,j} b_{k,l}}{\sum_{s=1}^{\ell} u_{i,s} b_{s,l}} - \frac{(d_{i,j} - a_{i,j}) b_{k,l}}{1 - \sum_{s=1}^{\ell} u_{i,s} b_{s,l}} \right] \tag{5.3}$$

$$\frac{\partial L}{\partial \gamma_{j,k}} = \log \pi_k \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j} | d_{i,j}, [\mathbf{UB}]_{i,k}) - \log \gamma_{j,k} - 1 + \alpha_j \tag{5.4}$$

$$\frac{\partial L}{\partial \pi_k} = \sum_{j=1}^{n} \frac{\gamma_{j,k}}{\pi_k} + \lambda \tag{5.5}$$

$$\frac{\partial L}{\partial \lambda} = \sum_{k=1}^{\ell} \pi_k - 1 \tag{5.6}$$

$$\frac{\partial L}{\partial \alpha_j} = \sum_{k=1}^{\ell} \gamma_{j,k} - 1, \tag{5.7}$$

We set the gradients to 0 to obtain optimal values.

$$\gamma_{j,k} = \pi_k \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j} | d_{i,j}, [\mathbf{UB}]_{i,k}) e^{\alpha_j - 1} \tag{5.8}$$

$$\pi_k = -\frac{1}{\lambda} \sum_{j=1}^{n} \gamma_{j,k} \tag{5.9}$$

$$\sum_{k=1}^{\ell} \pi_k = 1 \tag{5.10}$$

$$\sum_{k=1}^{\ell} \gamma_{j,k} = 1. \tag{5.11}$$

By combining the equations, we obtain

$$\gamma_{j,k} = \frac{\pi_k \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j} | d_{i,j}, [\mathbf{UB}]_{i,k})}{\sum_{l=1}^{\ell} \pi_l \prod_{i=1}^{m} \mathrm{Bin}(a_{i,j} | d_{i,j}, [\mathbf{UB}]_{i,l})} \tag{5.12}$$

$$\pi_k = \frac{1}{n} \sum_{j=1}^{n} \gamma_{j,k}. \tag{5.13}$$

To update $\mathbf{U}$, we apply exponentiated gradient descent [93] to restrict $\mathbf{U}$ in the constrained space.

$$u_{ik}^{t+1} \leftarrow \frac{u_{ik}^{t} \exp(-\eta \nabla L(u_{ik}^{t}))}{\sum_{l=1}^{K} u_{il}^{t} \exp(-\eta \nabla L(u_{il}^{t}))}, \tag{5.14}$$

where η is a learning rate, and the domain of U is $\{u_i | u_{ik} \geq 0 \text{ and } ||u_i|| \leq 1\}$ for all i. Thus, our EM algorithm considering $\mathbf{U}$ is as follows:

```
Algorithm 2: EM algorithm
  Initialize: mixture matrix $\mathbf{U}$, mixture proportions $\pi_k$
  while not converged do
      E Step: Compute $\gamma_{j,k}$ with current parameters
      $\gamma_{j,k} = \frac{\pi_k \prod_{i=1}^{m} \text{Bin}(a_{i,j}|d_{i,j},[\mathbf{UB}]_{i,k})}{\sum_{l=1}^{\ell} \pi_l \prod_{i=1}^{m} \text{Bin}(a_{i,j}|d_{i,j},[\mathbf{UB}]_{i,l})}$
      M Step: Maximize log-likelihood with the updated mixture
              matrix and current responsibilities
      $\pi_k = \frac{1}{n} \sum_{j=1}^{n} \gamma_{j,k}$
      $u_{i,k} \leftarrow \frac{u_{i,k} \exp(-\eta \nabla L(u_{i,k}))}{\sum_{l=1}^{\ell} u_{i,l} \exp(-\eta \nabla L(u_{i,l}))}$
      Update: log-likelihood and check convergence
  end
```

By considering $\mathbf{U}$, we can directly estimate the latent variables without depending on other linear programming solvers, which is expected to be more efficient for large-scale data.

REFERENCES

[1] P. C. Nowell, “The clonal evolution of tumor cell populations,” *Science*, vol. 194, pp. 23–28, 1976.

[2] R. Fisher, L. Pusztai, and C. Swanton, “Cancer heterogeneity: implications for targeted therapeutics,” *British Journal of Cancer*, vol. 108, no. 3, p. 479–485, 2013.

[3] M. Jamal-Hanjani, S. A. Quezada, J. Larkin, and C. Swanton, “Translational implications of tumor heterogeneity,” *Clinical Cancer Research*, vol. 21, no. 6, pp. 1258–1266, 2015.

[4] S. Venkatesan and C. Swanton, “Tumor evolutionary principles: How intratumor heterogeneity influences cancer treatment and outcome,” *American Society of Clinical Oncology Educational Book*, vol. 35, pp. e141–e149, 2016.

[5] F. Bray, J. Ferlay, I. Soerjomataram, R. L. Siegel, L. A. Torre, and A. Jemal, “Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries,” *CA: A Cancer Journal for Clinicians*, vol. 68, no. 6, pp. 394–424, 2018.

[6] Y. Qi, D. Pradhan, and M. El-Kebir, “Implications of non-uniqueness in phylogenetic deconvolution of bulk DNA samples of tumors,” *Algorithms for Molecular Biology*, vol. 14, no. 1, pp. 23–14, Sep. 2019.

[7] M. Griffith, C. A. Miller, O. L. Griffith, K. Krysiak, Z. L. Skidmore, A. Ramu, J. R. Walker, H. X. Dang, L. Trani, D. E. Larson, R. T. Demeter, M. C. Wendl, J. F. McMichael, R. E. Austin, V. Magrini, S. D. McGrath, A. Ly, S. Kulkarni, M. G. Cordes, C. C. Fronick, R. S. Fulton, C. A. Maher, L. Ding, J. M. Klco, E. R. Mardis, T. J. Ley, and R. K. Wilson, “Optimizing cancer genome sequencing and analysis,” *Cell Systems*, vol. 1(3), pp. 210–223, 2015.

[8] S. Nik-Zainal, L. B. Alexandrov, D. C. Wedge, P. V. Loo, C. D. Greenman, K. Raine, D. Jones, J. Hinton, J. Marshall, L. A. Stebbings, A. Menzies, S. Martin, K. Leung, L. Chen, C. Leroy, M. Ramakrishna, R. Rance, K. W. Lau, L. J. Mudie, I. Varela, D. J. McBride, G. R. Bignell, S. L. Cooke, A. Shlien, J. Gamble, I. Whitmore, M. Maddison, P. S. Tarpey, H. R. Davies, E. Papaemmanuil, P. J. Stephens, S. McLaren, A. P. Butler, J. W. Teague, G. Jönsson, J. E. Garber, D. Silver, P. Miron, A. Fatima, S. Boyault, A. Langerød, A. Tutt, J. W. M. Martens, S. A. J. R. Aparicio, Åke Borg, A. V. Salomon, G. Thomas, A.-L. Børresen-Dale, A. L. Richardson, M. S. Neuberger, P. A. Futreal, P. J. Campbell, M. R. Stratton, and B. C. W. G. of the International Cancer Genome Consortium, "Mutational processes molding the genomes of 21 breast cancers," *Cell*, vol. 149, no. 5, pp. 994–1007, 2012.

[9] F. Strino, F. Parisi, M. Micsinai, and Y. Kluger, "TrAp: a tree approach for fingerprinting subclonal tumor composition," *Nucleic Acids Research*, vol. 41, no. 17, p. e165, 2013.

[10] W. Jiao, S. Vembu, A. G. Deshwar, L. Stein, and Q. Morris, "Inferring clonal evolution of tumors from single nucleotide somatic mutations," *BMC Bioinformatics*, vol. 15, no. 1, 2014.

[11] S. Malikic, A. W. McPherson, N. Donmez, and C. S. Sahinalp, "Clonality inference in multiple tumor samples using phylogeny," *Bioinformatics*, vol. 31, no. 9, pp. 1349–1356, 2015.

[12] K. Yuan, T. Sakoparnig, F. Markowetz, and N. Beerenwinkel, "BitPhylogeny: a probabilistic framework for reconstructing intra-tumor phylogenies," *Genome Biology*, vol. 16, no. 36, 2015.

[13] V. Popic, R. Salari, I. Hajirasouliha, D. Kashef-Haghighi, R. B. West, and S. Batzoglou, "Fast and scalable inference of multi-sample cancer lineages," *Genome Biology*, vol. 16, 2015.

[14] A. G. Deshwar, S. Vembu, C. K. Yung, G. H. Jang, L. Stein, and Q. Morris, "Phylowgs: Reconstructing subclonal composition and evolution from whole-genome sequencing of tumors," *Genome Biology*, vol. 16, no. 1, 2015.

[15] M. El-Kebir, L. Oesper, H. Acheson-Field, and B. J. Raphael, "Reconstruction of clonal trees and tumor composition from multi-sample sequencing data," *Bioinformatics*, vol. 31, no. 12, pp. 62–70, 2015.

[16] Y. Jiang, Y. Qiu, A. J. Minn, and N. R. Zhang, "Assessing intratumor heterogeneity and tracking longitudinal and spatial clonal evolutionary history by next-generation sequencing." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 113, no. 37, pp. 5528–37, 2016.

[17] G. Satas and B. J. Raphael, "Tumor phylogeny inference using tree-constrained importance sampling," *Bioinformatics*, vol. 33, no. 14, pp. i152–i160, 2017.

[18] J. A. Wintersinger, S. M. Dobson, E. Kulman, L. D. Stein, J. E. Dick, and Q. Morris, "Reconstructing complex cancer evolutionary histories from multiple bulk dna samples using pairtree," *Blood Cancer Discovery*, p. OF1–OF12, 2022.

[19] K. I. Kim and R. Simon, "Using single cell sequencing data to model the evolutionary history of a tumor," *BMC Bioinformatics*, vol. 15, no. 27, 2014.

[20] K. Jahn, J. Kuipers, and N. Beerenwinkel, "Tree inference for single-cell data," *Genome Biology*, vol. 17, no. 86, 2016.

[21] E. M. Ross and F. Markowetz, "OncoNEM: inferring tumor evolution from single-cell sequencing data," *Genome Biology*, vol. 17, no. 69, 2016.

[22] S. Salehi, A. Steif, A. Roth, S. Aparicio, A. Bouchard-Côté, and S. P. Shah, "ddClone: joint statistical inference of clonal populations from single cell and bulk tumour sequencing data," *Genome Biology*, vol. 18, no. 1, 2017.

[23] S. Malikic, K. Jahn, J. Kuipers, S. C. Sahinalp, and N. Beerenwinkel, "Integrative inference of subclonal tumour evolution from single-cell and bulk sequencing data," *Nature Communications*, vol. 10, no. 2750, 2019.

[24] M. Kimura, "The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations," *Genetics*, vol. 61, no. 4, pp. 893–903, 1969.

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

[26] A. Roth, J. Khattra, D. Yap, A. Wan, E. Laks, J. Biele, G. Ha, S. Aparicio, A. Bouchard-Cote, and S. P. Shah, "Pyclone: statistical inference of clonal population structure in cancer," *Nature Methods*, vol. 11, pp. 396–398, 2014.

[27] S. Gillis and A. Roth, "Pyclone-VI: scalable inference of clonal population structures using whole genome data," *BMC Bioinformatics*, vol. 21, no. 571, 2020.

[28] C. A. Miller, B. S. White, N. D. Dees, J. S. Welch, M. Griffith, O. Griffith1, R. Vij, M. H. Tomasson, T. A. Graubert, M. J. Walter, W. Schierding, T. J. Ley, J. F. DiPersio, E. R. Mardis, R. K. Wilson, and L. Ding, "Sciclone: Inferring clonal architecture and tracking the spatial and temporal patterns of tumor evolution," *PLOS Computational Biology*, vol. 10, no. 8, 2014.

[29] R. M. Neal, "Markov chain sampling methods for dirichlet process mixture models," *Journal of Computational and Graphical Statistics*, vol. 9, pp. 249–265, 2000.

[30] N. Metropolis, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[31] D. M. B. M. I. Jordan, "Variational inference for dirichlet process mixtures," *Journal of Bayesian Analysis*, vol. 1, pp. 121–144, 2006.

[32] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, pp. 859–877, 2017.

[33] Z. Ma and A. Leijon, "Bayesian estimation of beta mixture models with variational inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 2160–2173, 2011.

[34] W. Fan, N. Bouguila, and D. Ziou, "Variational learning for finite dirichlet mixture models and applications," *Journal of Bayesian Analysis*, vol. 23, pp. 762–774, 2012.

[35] L. Y. Liu, V. Bhandari, A. Salcedo, S. M. G. Espiritu, Q. D. Morris, T. Kislinger, and P. C. Boutros, "Quantifying the influence of mutation detection on tumour subclonal reconstruction," *Nature Communications*, vol. 11, 2020.

[36] I. Hajirasouliha, A. Mahmoody, and B. J. Raphael, "A combinatorial approach for analyzing intra-tumor heterogeneity from high-throughput sequencing data," *Bioinformatics*, vol. 30, pp. i78–i86, 2014.

[37] M. El-Kebir, G. Satas, and L. O. B. J. Raphael, "Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures," *Cell Systems*, vol. 3, no. 1, pp. 43–53, 2016.

[38] H. X. Dang, B. S. White, S. M. Foltz, C. A. Miller, J. Luo, R. C. Fields, and C. A. Maher, "ClonEvol: clonal ordering and visualization in cancer sequencing," *Annals of Oncology*, vol. 28(12), pp. 3076–3082, 2017.

[39] R. P. Adams, Z. Ghahramani, and M. Jordan, "Tree-structured stick breaking for hierarchical data," *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[40] N. Navin, J. Kendall, J. Troge, P. Andrews, L. Rodgers, J. McIndoo, K. Cook, A. Stepansky, D. Levy, D. Esposito, L. Muthuswamy, A. Krasnitz, W. R. McCombie, J. Hicks, and M. Wigler, "Tumour evolution inferred by single-cell sequencing," *Nature*, vol. 472, pp. 90–94, 2011.

[41] N. E. Navin, "Cancer genomics: one cell at a time." *Genome Biology*, vol. 15, no. 8, p. 452, 2014.

[42] N. E. Navin, "The first five years of single-cell cancer genomics and beyond," *Genome research*, vol. 25, no. 10, pp. 1499–1507, 2015.

[43] C. Gawad, W. Koh, and S. R. Quake, "Single-cell genome sequencing: current state of the science," *Nature Reviews Genetics*, vol. 17, pp. 175–188, 2016.

[44] M. R. Emmert-Buck, R. F. Bonner, P. D. Smith, R. F. Chuaqui, Z. Zhuang, S. R. Goldstein, R. A. Weiss, and L. A. Liotta, "Laser capture microdissection," *Science*, vol. 274, pp. 998–1001, 1996.

[45] M. L. Leung, Y. Wang, J. Waters, and N. E. Navin, "SNES: single nucleus exome sequencing," *Genome Biology*, vol. 16, no. 55, 2015.

[46] E. Shapiro, T. Biezuner, and S. Linnarsson, "Single-cell sequencing-based technologies will revolutionize whole-organism science," *Nature Reviews Genetics*, vol. 14, pp. 618–630, 2013.

[47] C. F. A. de Bourcy, I. D. Vlaminck, J. N. Kanbar, J. Wang, C. Gawad, and S. R. Quake, "A quantitative comparison of single-cell whole genome amplification methods," *PLoS ONE*, vol. 9, no. e105585, 2014.

[48] Y. Hou, K. Wu, X. Shi, F. Li, L. Song, H. Wu, M. Dean, G. Li, S. Tsang, R. Jiang, X. Zhang, B. Li, G. Liu, N. Bedekar, N. Lu, G. Xie, H. Liang, L. Chang, T. Wang, J. Chen, Y. Li, X. Zhang, H. Yang, X. Xu, L. Wang, and J. Wang, "Comparison of variations detection between whole-genome amplification methods used in single-cell resequencing," *Gigascience*, vol. 4, no. 37, 2015.

[49] L. Huang, F. Ma, A. Chapman, S. Lu, and X. S. Xie, "Single-cell whole-genome amplification and sequencing: Methodology and applications," *Annual Review of Genomics and Human Genetics*, vol. 16, pp. 79–102, 2015.

[50] Y. Hou, L. Song, P. Zhu, B. Zhang, Y. Tao, X. Xu, F. Li, K. Wu, J. Liang, D. Shao, H. Wu, X. Ye, C. Ye, R. Wu, M. Jian, Y. Chen, W. Xie, R. Zhang, L. Chen, X. Liu, X. Yao, H. Zheng, C. Yu, Q. Li, Z. Gong, M. Mao, X. Yang, L. Yang, J. Li, W. Wang, Z. Lu, N. Gu, G. Laurie, L. Bolund, K. Kristiansen, J. Wang, H. Yang, Y. Li, X. Zhang, and J. Wang, "Single-cell exome sequencing and monoclonal evolution of a JAK2-negative myeloproliferative neoplasm," *Cell*, vol. 148, no. 5, pp. 873–885, 2012.

[51] A. Roth, A. McPherson, E. Laks, J. Biele, D. Yap, A. Wan, M. A. Smith, C. B. Nielsen, J. N. McAlpine, S. Aparicio, A. B.-C. té, and S. P. Shah, "Clonal genotype and population structure inference from single-cell tumor sequencing," *Nature Methods*, vol. 13, pp. 573–576, 2016.

[52] N. Borgsmüller, J. Bonet, F. Marass, A. Gonzalez-Perez, N. Lopez-Bigas, and N. Beerenwinkel, "BnpC: Bayesian non-parametric clustering of single-cell mutation profiles," *Bioinformatics*, vol. 36, no. 19, pp. 4854–4859, 2020.

[53] C. M. Bishop, *Pattern Recognition and Machine Learning.* New York: Springer, 2006.

[54] T. Broderick, J. Pitman, and M. I. Jordan, "Feature allocations, probability functions, and paintboxes," *Bayesian Analysis*, vol. 8, no. 4, pp. 801–836, 2013.

[55] J. Pitman, "Exchangeable and partially exchangeable random partitions," *Probabilistic Theory and Related Fields*, vol. 102, pp. 145–158, 1995.

[56] H. Zafar, A. Tzen, N. Navin, K. Chen, and L. Nakhleh, "Sifit: inferring tumor trees from single-cell sequencing data under finite-sites models," *Genome Biology*, vol. 18, no. 1, 2017.

[57] H. Zafar, N. Navin, K. Chen, and L. Nakhleh, "SiCloneFit: Bayesian inference of population structure, genotype, and phylogeny of tumor clones from single-cell genome sequencing data," *Genome Research*, vol. 29, p. 1847–1859, 2019.

[58] E. W. Meeds, D. A. Ross, R. S. Zemel, and S. T. Roweis, "Learning stick-figure models using nonparametric bayesian priors over trees," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[59] M. El-Kebir, "SPhyR: tumor phylogeny estimation from single-cell sequencing data under loss and error," *Bioinformatics*, vol. 34, no. 17, pp. i671–i679, 2018.

[60] L. Dollo, “Les lois de l’évolution,” *Bulletin de la Société Belge Géologie de Paleontolologie et d’Hydrologie*, pp. 164–166, 1893.

[61] E. S. Azer, F. R. Mehrabadi, S. Malikić, X. C. Li, O. Bartok, K. Litchfield, R. Levy, Y. Samuels, A. A. Schäffer, E. M. Gertz, C.-P. Day, E. Pérez-Guijarro, K. Marie, M. P. Lee, G. Merlino, F. Ergun, and S. C. Sahinalp, “PhISCS-BnB: a fast branch and bound algorithm for the perfect tumor phylogeny reconstruction problem,” *Bioinformatics*, vol. 36, p. i169–i176, 2020.

[62] S. Malikic, F. R. Mehrabadi, S. Ciccolella, M. K. Rahman, C. Ricketts, E. Haghshenas, D. Seidman, F. Hach, I. Hajirasouliha, and S. C. Sahinalp, “PhISCS: a combinatorial approach for subperfect tumor phylogeny reconstruction via integrative use of single-cell and bulk sequencing data,” *Genome Research*, vol. 29, no. 11, pp. 1860–1877, 2019.

[63] M. Hestenes, “Multiplier and gradient methods,” *Journal of Optimization Theory and Applications*, vol. 4, no. 5, p. 303–320, 1969.

[64] M. Powell, “A method for nonlinear constraints in minimization problems,” *In Fletcher, R., Editor, Optimization, Academic Press*, p. 283–298, 1969.

[65] L. A. Wolsey and G. L. Nemhauser, *Integer and Combinatorial Optimization.* Wiley-Interscience, 1988.

[66] G. E. Schwarz, “Estimating the dimension of a model,” *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[67] S. P. Lloyd, “Least square quantization in pcm,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[68] L. Perron and V. Furnon, “OR-Tools,” Google. [Online]. Available: https://developers.google.com/optimization/

[69] A. Rosenberg and J. Hirschberg, “V-measure: A conditional entropy-based external cluster evaluation measure,” in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 410–420.

[70] H. Prüfer, “Neuer beweis eines satzes über permutationen,” *Archiv der mathematik und Physik*, vol. 27, pp. 742–744, 1918.

[71] S. Santaguida and A. Amon, “Short- and long-term effects of chromosome mis-segregation and aneuploidy,” *Nature Reviews Molecular Cell Biology*, vol. 16, pp. 473–485, 2015.

[72] A. Shlien and D. Malkin, "Copy number variations and cancer," *Genome Medicine*, vol. 1, no. 62, 2009.

[73] M. Zarrei, J. R. MacDonald, D. Merico, and S. W. Scherer, "A copy number variation map of the human genome," *Nature Reviews Genetics*, vol. 16, pp. 172–183, 2015.

[74] A. M. Taylor, J. Shih, G. Ha, G. F. Gao, X. Zhang, A. C. Berger, S. E. Schumacher, C. Wang, H. Hu, J. Liu, A. J. Lazar, The Cancer Genome Atlas Research Network, A. D. Cherniack, R. Beroukhim, and M. Meyerson, "Genomic and functional approaches to understanding cancer aneuploidy," *Cancer Cell*, vol. 33, no. 4, pp. 676–689, 2018.

[75] S. Zaccaria, M. El-Kebir, G. W. Klau, and B. J. Raphael, "Phylogenetic copy-number factorization of multiple tumor samples," *Journal of Computational Biology*, vol. 25, no. 7, pp. 689–708, 2018.

[76] G. Satas, S. Zaccaria, G. Mon, and B. J. Raphae, "SCARLET: Single-cell tumor phylogeny inference with copy-number constrained mutation losses," *Cell Systems*, vol. 10, no. 4, pp. 323–332.e8, 2020.

[77] R. Zeira and B. J. Raphael, "Copy number evolution with weighted aberrations in cancer," *Bioinformatics*, vol. 36, p. i344–i352, 2020.

[78] G. Satas, S. Zaccaria, M. El-Kebir, and B. J. Raphael, "Decifering the elusive cancer cell fraction in tumor heterogeneity and evolution," *Cell Systems*, vol. 12, no. 10, pp. 1004–1018, 2021.

[79] K. R. Campbell, A. Steif, E. Laks, H. Zahn, D. Lai, A. McPherson, H. Farahani, F. Kabeer, C. O'Flanagan, J. Biele, J. Brimhall, B. Wang, P. Walters, I. Consortium, A. Bouchard-Côté, S. Aparicio, and S. P. Shah, "clonealign: statistical integration of independent single-cell rna and dna sequencing data from human cancers," *Genome Biology*, vol. 20, no. 54, 2019.

[80] D. J. McCarthy, R. Rostom, Y. Huang, D. J. Kunz, P. Danecek, M. J. Bonder, T. Hagai, R. Lyu, HipSci Consortium, W. Wang, D. J. Gaffney, B. D. Simons, O. Stegle, and S. A. Teichmann, "Cardelino: computational integration of somatic clonal substructure and single-cell transcriptomes," *Nature Methods*, vol. 17, pp. 414–421, 2020.

[81] S.-H. Jun, H. Toosi, J. Mold, C. Engblom, X. Chen, C. O'Flanagan, M. Hagemann-Jensen, R. Sandberg, S. Aparicio, J. Hartman, A. Roth, and J. Lagergren, "Phylex: Accurate reconstruction of clonal structure via integrated analysis of bulk dna-seq and single cell rna-seq data," *bioRxiv*, 2021.

[82] G. A. Van der Auwera and B. D. O'Connor, "Genomics in the cloud: Using docker, gatk, and wdl in terra (1st edition)," *O'Reilly Media*, 2020.

[83] A. Salcedo, M. Tarabichi, S. M. G. Espiritu, A. G. Deshwar, M. David, N. M. Wilson, S. Dentro, J. A. Wintersinger, L. Y. Liu, M. Ko, S. Sivanandan, H. Zhang, K. Zhu, T.-H. O. Yang, J. M. Chilton, A. Buchanan, C. M. Lalansingh, C. P'ng, C. V. Anghel, I. Umar, B. Lo, W. Zou, D. S. Het Participants, J. T. Simpson, J. M. Stuart, D. Anastassiou, Y. Guan, A. D. Ewing, K. Ellrott, D. C. Wedge, Q. Morris, P. V. Loo, and P. C. Boutros, "A community effort to create standards for evaluating tumor subclonal reconstruction," *Nature Biotechnology*, vol. 38, pp. 97–107, 2020.

[84] D. C. Koboldt, "Best practices for variant calling in clinical sequencing," *Genome Medicine*, vol. 12, no. 91, 2020.

[85] M. Tarabichi, A. Salcedo, A. G. Deshwar, M. N. Leathlobhair, J. Wintersinger, D. C. Wedge, P. V. Loo, Q. D. Morris, and P. C. Boutros, "A practical guide to cancer subclonal reconstruction from dna sequencing," *Nature Methods*, vol. 18, pp. 144–155, 2021.

[86] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, pp. 4171–4186.

[87] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *arXiv:2005.14165*, 2020.

[88] D. R. Kelley, J. Snoek, and J. L. Rinn, "Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks," *Genome Research*, vol. 26, pp. 990–999, 2016.

[89] D. R. Kelley, Y. A. Reshef, M. Bileschi, D. Belanger, C. Y. McLean, and J. Snoek, "Sequential regulatory activity prediction across chromosomes with convolutional neural networks," *Genome Research*, vol. 28, pp. 739–750, 2018.

[90] Žiga Avsec, V. Agarwal, D. Visentin, J. R. Ledsam, A. Grabska-Barwinska, K. R. Taylor, Y. Assael, J. Jumper, P. Kohli, and D. R. Kelley, "Effective gene expression prediction from sequence by integrating long-range interactions," *Nature Methods*, vol. 18, pp. 1196–1203, 2021.

[91] Y. LeCun, L. Bottou, Y. Bengio, , and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, 1998, p. 2278–2324.

[92] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[93] J. Kivinen and M. Warmuth, "Exponentiated gradient versus gradient descent for linear predictors," *Information and Computation*, vol. 132, no. 1, pp. 1–63, 1997.