Popic *et al. Genome Biology* (2015) 16:91
DOI 10.1186/s13059-015-0647-8

Genome Biology

METHOD Open Access

Fast and scalable inference of multi-sample cancer lineages

Victoria Popic[1], Raheleh Salari[1], Iman Hajirasouliha[1], Dorna Kashef-Haghighi[1], Robert B West[2] and Serafim Batzoglou[1*]

Abstract

Somatic variants can be used as lineage markers for the phylogenetic reconstruction of cancer evolution. Since somatic phylogenetics is complicated by sample heterogeneity, novel specialized tree-building methods are required for cancer phylogeny reconstruction. We present LICHeE (Lineage Inference for Cancer Heterogeneity and Evolution), a novel method that automates the phylogenetic inference of cancer progression from multiple somatic samples. LICHeE uses variant allele frequencies of somatic single nucleotide variants obtained by deep sequencing to reconstruct multi-sample cell lineage trees and infer the subclonal composition of the samples. LICHeE is open source and available at http://viq854.github.io/lichee.

Background

Cancer is driven by the accumulation of somatic mutations that confer fitness advantages to the tumor cells. Numerous studies have shown tumors to be highly heterogeneous, consisting of mixtures of cell subpopulations with distinct sets of somatic variants (for example see review papers [1,2]). With the advent of next-generation sequencing technologies, many large-scale efforts are underway to catalog the somatic mutational events driving the progression of cancer [3,4] and infer the phylogenetic relationships of tumor subclones. Characterizing the heterogeneity and inferring tumor phylogenies are key steps for developing targeted cancer therapies [5] and understanding the biology and progression of cancer.

To reconstruct tumor phylogenies, studies have utilized variant allele frequency (VAF) data of somatic single nucleotide variants (SSNVs) obtained by whole-genome [6,7], exome [8], and targeted deep sequencing [6,9]. Clustering of SSNVs based on VAF similarity [10-12] and detection of copy number aberrations, while accounting for variable sample purity [8,13,14], have been used to differentiate and order groups of mutational events. While many evolutionary studies of cancer have focused on single-sample intra-tumor heterogeneity [15], several studies have also compared multiple tumor samples extracted from a single patient either at different points in time during cancer progression [16-18] or from different regions of the same tumor or its metastases [7,19-23]. In multi-sample approaches, the patterns of SSNV sharing (that is, distinguishing somatic mutations that are omnipresent, partially shared, or private among the samples) can serve as phylogenetic markers from which lineage trees are reconstructed [24]. On the basis of the lineage trees, the evolutionary timing of each mutational event can then be inferred with high confidence [7,17,19,25].

Most existing multi-sample studies with a relatively small number of SSNVs infer the tumor phylogenies manually by analyzing SSNV VAFs and presence patterns across samples [7,22,26]. Several other studies used implementations of traditional phylogeny reconstruction methods, such as neighbor joining with Pearson correlation distances [27], or maximum parsimony [21] on patterns of somatic mutational sharing across samples. However, to scale to datasets comprising large numbers of samples per patient and extract fine-grained SSNV timing information, as well as handle sample heterogeneity, which traditional tree-building techniques are not designed to do, specialized computational approaches need to be developed for tumor cell lineage reconstruction.

*Correspondence: serafim@cs.stanford.edu
[1]Department of Computer Science, Stanford University, Stanford, CA, USA
Full list of author information is available at the end of the article

Several computational methods have been recently developed to address this need. The method SubcloneSeeker [28] takes as input clusters of variant cell prevalence (CP) estimates and generates all possible subclone structures in each tumor sample separately. The per-sample solutions are then trimmed by checking their compatibilities during a merge step, which reports which sample trees are compatible across a given pair of samples. However, the merge step is currently designed to check compatibilities of two tumor samples only (for example, relapse/primary tumor sample pairs that are common in clinical studies) and it cannot merge the subclone structures of more than two samples. The method PhyloSub [29] infers tumor phylogenies using a Bayesian non-parametric prior over trees and Markov chain Monte Carlo sampling. It performs reasonably on samples with very few mutations that form simple (chain) topologies; however, it produces unsatisfactory results on larger multi-sample datasets, such as [21] (see Additional file 1 for details). Most recently, PhyloWGS [30] was developed for subclonal reconstruction using whole-genome sequencing datasets. PhyloWGS is a probabilistic framework based on the earlier development of PhyloSub. This new algorithm utilizes both VAFs of SSNVs and the effect of copy number variants (CNVs) already inferred in regions overlapping with those SSNVs. Finally, CITUP [31] is a combinatorial method that uses an exact quadratic integer programming formulation to obtain optimal lineage trees that are in concordance with the VAF data. CITUP reports higher accuracies when compared to Phylosub [31]; however, its optimization problem may be intractable when the lineage tree is arbitrarily large.

In this work, we introduce LICHeE (Lineage Inference for Cancer Heterogeneity and Evolution), a novel computational method for the reconstruction of multi-sample tumor phylogenies and tumor subclone decomposition from targeted deep-sequencing SSNV datasets. Given SSNV VAFs from multiple samples, LICHeE finds the set of lineage trees that are consistent with the SSNV presence patterns and VAFs within each sample and are valid under the cell division process. Given each such tree, LICHeE provides estimates of the subclonal mixtures of the samples by inferring sample heterogeneity simultaneously with phylogenetic cell lineage tree reconstruction. LICHeE is able to search for lineage trees very efficiently by incorporating the SSNVs into an evolutionary constraint network that embeds all such trees and applying VAF constraints to reduce the search space. LICHeE runs in only a few seconds given hundreds of input SSNVs and does not require data preprocessing.

We demonstrate that LICHeE is highly effective in reconstructing the lineage trees and sample heterogeneity by evaluating it on simulated trees of heterogeneous cancer cell lineage evolution, as well as on three recently published ultra-deep-sequencing multi-sample datasets of clear cell renal cell carcinoma (ccRCC) by Gerlinger et al. [21], high-grade serous ovarian cancer (HGSC) by Bashashati et al. [27], and breast cancer xenoengraftment in immunodeficient mice by Eirew et al. [32], for which single-cell validation results are also available. LICHeE found unique trees for each ccRCC patient and for all except one patient (for which multiple valid trees were found) of the HGSC study. For the ccRCC dataset, LICHeE trees were nearly identical to the published trees, which are the result of a multi-step thorough analysis of the data, involving SSNV calling, clustering, and tree-building using maximum parsimony. For the HGSC dataset, LICHeE improved on the results reported by the study, producing trees with better support from the data. LICHeE also revealed additional heterogeneity in the samples of both studies. In particular, LICHeE identified subclones in one more sample of the ccRCC study (in addition to the reported six samples) and three samples of the HGSC study, all supported by the data. Finally, the trees reconstructed by LICHeE on the xenoengraftment dataset can be highly validated by the single-cell analysis presented in the paper. LICHeE is open source and freely distributed at [33], and includes an intuitive graphical user interface (GUI) that may aid users in performing quality control on the output trees as well as interpreting the trees biologically.

Results and discussion

Overview of the multi-sample cancer phylogeny inference method, LICHeE

LICHeE is a method designed to reconstruct cancer cell lineages using SSNVs from multiple related normal and tumor samples of individual cancer patients, allowing for heterogeneity within each sample. Given a set of validated deeply sequenced SSNVs, LICHeE uses the presence patterns of SSNVs across samples and their VAFs as lineage markers by relying on the perfect phylogeny model [34]. This model assumes that mutations do not recur independently in different cells; hence, cells sharing the same mutation must have inherited it from a common ancestral cell. This assumption can be used to derive the following SSNV ordering constraints. Firstly, (1) a mutation present in a given set of samples cannot be a successor of a mutation that is present in a smaller subset of these samples, since it could not have arisen independently by chance in the additional samples. Similarly, (2) a given mutation cannot have a VAF higher than that of its predecessor mutation (except due to CNVs), since all cells containing this mutation will also contain the predecessor. Finally, (3) the sum of the VAFs of mutations disjointly present in distinct subclones cannot exceed the VAF of a common predecessor mutation present in these subclones,

since the subclones with the descendent mutations must contain the parent mutations (this constraint is formally defined in 'Materials and methods'). These constraints provide key information about the topology of the true cell lineage tree and are leveraged by LICHeE to define the search space of the possible underlying lineage trees and evaluate the validity of the resulting topologies. The final goal of LICHeE is to find phylogenetic trees encoding an evolutionary ordering of the input SSNVs that does not violate any of these three constraints.

At a high level, the LICHeE algorithm can be broken down into the following main steps (Figure 1). First, LICHeE partitions SSNVs into groups based on their occurrence in each sample, such that each group stores all the mutations that were called in the same subset of samples. To separate subclone lineages, the SSNV members of each resulting group are then further clustered based on their VAFs, such that SSNVs with similar VAFs across samples are clustered together. The final lineage tree needs to provide a valid ordering of these resulting SSNV clusters (that is, an ordering that does not violate the three constrains defined above). To find such a tree, we construct an evolutionary constraint network, which encodes whether a given cluster of SSNVs could have preceded another, for each pair of clusters. More specifically, this network is a directed acyclic graph (DAG) that has SSNV clusters as its nodes and whose edges encode possible predecessor relationships among the nodes' mutations (that is, an edge denotes that the mutations of a given pair of clusters satisfy ordering constraints (1) and (2)). This network greatly reduces the search space of possible valid trees and allows us to formulate the task of inferring such trees as a search for spanning trees of the network that satisfy constraint (3) (within a given error margin), which ensures that the reconstructed trees are composed of parent–daughter edges that exhibit somatic VAF consistency with the cell lineage expansion. If multiple valid lineage trees are found during the search, the trees are ranked based on how well they support the cluster VAF data, the top-ranking tree minimizing the use of the permitted error margin (see 'Materials and methods' for details). Finally, as shown in Figure 1, the leaves of the resulting trees are the individual samples (added post-search), whose composition can be reconstructed by tracing back their respective subclone cell lineages in the tree. We detail each of these steps in 'Materials and methods'.

Since finding true SSNV groups is a crucial step of the algorithm, it is important to minimize false positive and false negative SSNV calls across samples. However, accurately detecting SSNVs in each sample is a challenging task due to high levels of noise in the data, which can come from various sources, such as sequencing errors, systematic amplification bias, mapping errors, and sample impurities. Multiple techniques have been developed to address this problem to date [35-39], many employing a Bayesian approach to model the distributions of noise and true genotypes in matched-normal samples. LICHeE can work with variant calls produced for each sample by any specialized existing method. However, it does not require the users to preprocess the data using these tools, since it provides its own heuristic mechanism to call SSNVs using the multi-sample VAF data. At a high level, it first finds SSNVs that can be called reliably in each sample using two hard thresholds T_{present} and T_{absent}, above and below which, respectively, the SSNVs are considered robustly present or absent. Then, assuming that the presence patterns of such SSNVs capture most of the topology of the true underlying evolutionary tree, it uses this inferred tree to inform the group assignment of the SSNVs whose VAF falls in between the thresholds (the 'greyzone').

Currently LICHeE does not automatically detect or incorporate the CNVs explicitly into the model, although the method can still find valid phylogenies even in the presence of SSNVs within such regions (for example, each patient in the ccRCC dataset had numerous variants from CNV regions). To address this limitation, LICHeE also accepts CP values instead of VAFs, which can be computed by several recently developed tools, such as PyClone [12], ABSOLUTE [8], and ASCAT [40], and account for CNVs, loss of heterozygosity (LOH) status, and sample purity. The same algorithmic steps can then be directly applied to CP values of each variant. It can be easily seen that the three perfect phylogeny ordering constraints still hold for CP values and can be used to search for the underlying lineage tree. Furthermore, to support outputs of specialized clustering approaches (for example, PyClone [12]), LICHeE also accepts already computed clusters of mutations (with given CP and VAF-based centroid values) and uses these clusters as nodes of the phylogenetic constraint network.

We evaluated LICHeE on three recently published ultra-deep-sequencing multi-sample datasets of ccRCC by Gerlinger et al. [21], HGSC by Bashashati et al. [27], and breast cancer xenoengraftment by Eirew et al. [32], as well as on simulated trees of heterogeneous cancer cell lineage evolution. On the ccRCC dataset, LICHeE constructs near-identical trees to the trees published in the study. We show that the interesting difference in the topology of one tree arises due to potential heterogeneity of a sample that cannot be discovered using the traditional maximum parsimony approach and analyze when this approach can fail to detect existing sample subclones. For each patient, LICHeE finds a unique valid tree. On the HGSC datasets we show that the trees generated by LICHeE are better supported by the data and demonstrate why applying neighbor joining with Pearson correlation

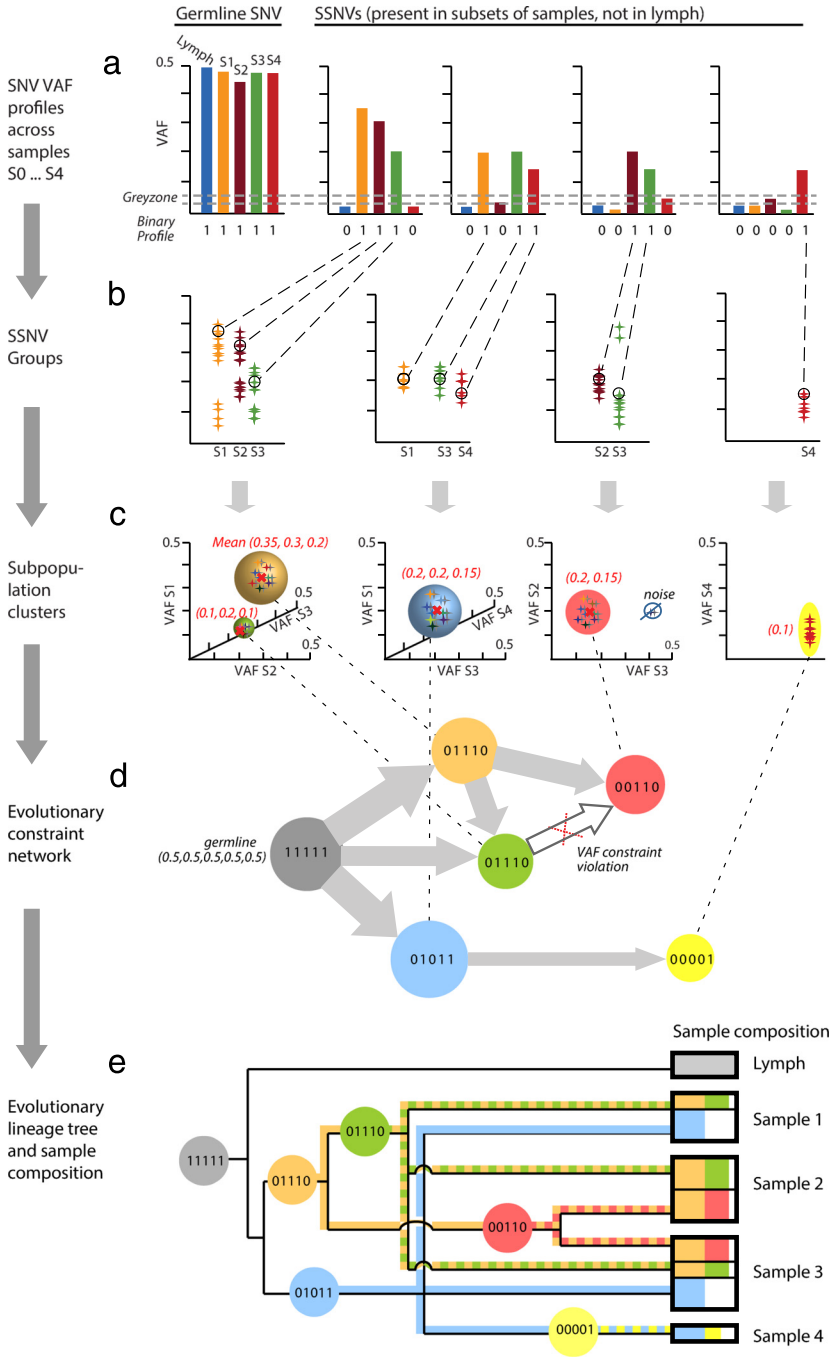

Figure 1 LICHeE method overview. **(a)** Toy example of five samples (lymph control and four tumor samples S1 to S4) with one germ-line single nucleotide variant (SNV) and four SSNVs, each associated with a binary sample profile. **(b)** SSNVs are partitioned into groups based on their binary profile (displayed groups contain other SSNVs with varying VAFs). **(c)** SSNV groups are clustered based on their VAFs. **(d)** The clusters of each SSNV group are incorporated into an evolutionary constraint network. An edge is placed between a parent node and child node if, for each sample, the VAF of the parent is greater than or equal to that of the child node. For example, this constraint is violated for the green 01110 node and node 00110. **(e)** The lineage tree is constructed from the constraint network. The leaves of the tree are the individual samples. The horizontal line subdivisions in a sample indicate mixed lineages, separating the different subpopulations of cells in the sample. The colors in each subdivision describe the mutation groups that the cells in this subpopulation have. Consider the orange node 01110, which denotes SSNVs of class 01110. Those SSNVs are found in samples 1, 2, and 3. After an ancestral cell division, the daughter cells' lineages accumulated SSNVs too (01110, green; 00110) that are now present in their descendent samples or subclones. About 20% of sample 1 are cells that come from the orange and green lineages, and about 40% come from the blue lineage. Samples 1, 2, and 3 grew from two or more subclones, whereas sample 4 only grew from one subclone. SNV, single nucleotide variant; SSNV, somatic single nucleotide variant; VAF, variant allele frequency.

distance metric, used by the study, might not be suitable for cancer datasets. LICHeE finds a unique valid tree for all patients except Case 5. Finally, we show that the trees inferred by LICHeE on the xenoengraftment dataset are consistent with the single-cell analysis done by the study. On each patient dataset LICHeE takes only a few seconds to run.

Lineage tree reconstruction on clear cell renal cell carcinoma data

The ccRCC study by Gerlinger et al. [21] validated 602 non-synonymous nucleotide substitutions and indels from multiple samples of eight individuals. It used VAF-based clustering of each sample to detect subclones prior to determining the variant presence patterns used in tree reconstruction. The phylogenetic trees were then reconstructed using maximum parsimony, revealing a branched pattern of ccRCC evolution in all tumors. Figure 2 juxtaposes the trees produced by LICHeE with the trees presented by Gerlinger et al. [21] (for details regarding the parameters used by LICHeE and the ccRCC dataset see Additional file 2). We can see that the trees generated by LICHeE are topologically identical (consisting of the same branches) to the published trees for patients EV005, EV007, RMH002, RMH008, and RK26. Furthermore, LICHeE identified subclones in all the samples reported to be heterogeneous by the study. In particular, it identified the following regions as a mixture of two subclones: EV005 R6 (with frequencies of 0.29 and 0.04), EV007 R3 (0.15 and 0.03), EV007 R9 (0.21 and 0.03), RMH008 R4 (0.19 and 0.14), RMH008 R6 (0.21 and 0.15), and RK26 R5 (0.15 and 0.03). These subclones correspond to the dominant (dom) and minor (min) shown in the published trees. Evidence supporting each subclone can be analyzed using the presented trees.

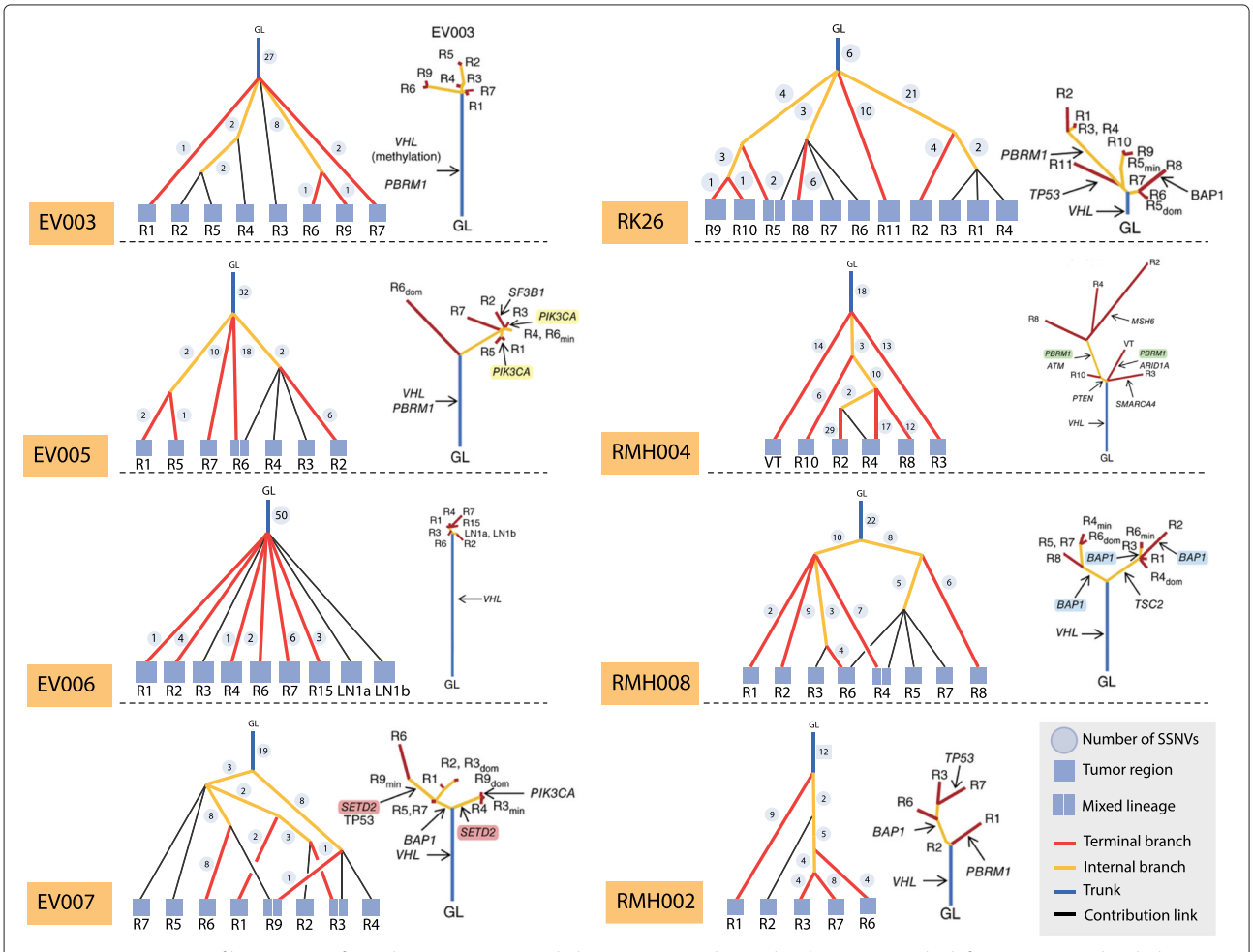

Figure 2 Comparison of lineage trees for eight ccRCC patients. Phylogenetic trees obtained with LICHeE (on the left) are contrasted with the trees published in [21] for each patient in the study. ccRCC, clear-cell renal cell carcinoma; dom, dominant; min, minor; GL, germ line; SSNV, somatic single nucleotide variant.

The trees generated for patients EV003 and EV006 also highly match the published results. For EV006, the LICHeE-generated tree does not contain the following two partially shared groups: (R2, LN1a, and LN1b) and (R3, R1, R4, R7, R15). For the first group we find no evidence in the data: no SSNVs are shared in these three samples and absent from the others. We do find one mutation that supports the second group. Because, by default, LICHeE eliminates nodes that have evidence from only one SSNV, this group is not shown in our tree.

The RMH004 dataset contains three partially overlapping groups: (R3, VT, R10, R4, R2), (R3, VT, R10, R4, R8), and (R10, R4, R2, R8). These groups represent separate branches where their lowest common ancestor is the group with mutations present across all samples: (R3, VT, R10, R4, R2, R8). However, the VAF of this parent group in sample R10 is 0.34, while the average VAF across each of the three groups is 0.32, 0.27, and 0.21, respectively. Therefore, no more than one of these groups can be a descendant of the parent group, without violating the VAF phylogenetic constraint. To generate a valid tree, the two least populated conflicting branches among the three are removed from the dataset. Similarly, these two groups are ignored by the maximum parsimony algorithm and are not present in the published tree. The difference between our tree and the published one comes from the mixed lineage we observe in sample R4. Due to the VAF of 0.17 of the group of R4 private mutations, the private group is not a descendant of the group shared by R4 and R2 since the average VAF of that node is 0.06, which is too small to be a parent to 0.17. Therefore, our method suggests two different subclones in R4. Since the group (R2, R4) is small (total of three mutations) and has a low frequency in R4, the evidence of the two subclones cannot be considered very strong; however, it does constitute a signal in the data for the R4 mixed lineage possibility.

While the ccRCC study does perform VAF clustering of each sample to find subclones at the onset, it runs the phylogenetic reconstruction using the presence pattern profiles only. On the other hand, our tree reconstruction with LICHeE applies the VAF constraint to the resulting tree topologies and clusters SSNVs in each group based on their VAFs across all the samples. For patient RMH004, applying the VAF constraint reveals additional sample heterogeneity, for which we produced a potentially improved tree compared to the tree reported using maximum parsimony. It can also be shown that clustering across all the samples rather than each sample individually, is a more appropriate approach for revealing the heterogeneity in the data. For example, if two subclones occur with high and low VAFs in one sample but are uniform in another sample, single-sample clustering will detect them only in the sample where they differ. For example, if the two subclones in samples R3 and R9 of patient EV007 (discussed above) had highly similar VAFs in these samples, clustering would not be able to differentiate the subclones. On the other hand, LICHeE would still be able to detect the mixed lineages in the two samples due to the presence of groups (R1, R2, R3, R5, R6, R7, R9) and (R3, R4, R9) in the data, which must form divergent branches in the lineage tree.

Lineage tree reconstruction on high-grade serous ovarian cancer data

We further evaluated LICHeE on the HGSC dataset from the study by Bashashati et al. [27]. This study validated 340 somatic mutations from 19 tumor samples of six patients. It used neighbor joining with Pearson correlation distances (computed on the binary sample presence patterns) to infer lineage trees. The trees generated by LICHeE juxtaposed with those presented in the paper are shown in Figure 3 (for details regarding the parameters used by LICHeE and the HGSC dataset see Additional file 3). In four out of six cases (Cases 2, 3, 4, and 6), the possible tree topologies are very simple and the trees produced by the two methods are unsurprisingly highly similar. Below we discuss the difference between the two remaining trees.

For Case 1, the tree reported in the paper suggests that sample d diverged first, followed by c. This suggests that there should be a group of mutations shared exclusively between samples a and b and another group shared between samples a, b, and c. However, examining the dataset VAF values, as well as the results of the binomial test determining SSNV presence in the samples used by this study, we found no evidence for these two groups. On the other hand, the tree produced by LICHeE suggests the presence of mutations shared by samples a, b, and d only, which we found both in the results of the study's binomial test and by applying the LICHeE hard threshold caller.

For Case 5, the study reports an early divergence of sample c. This suggests that there should be mutations shared between all the samples except c. We have confirmed that no such mutational profile exists in the data. On the other hand, the data shows the presence of mutations that exist in samples a, b, c, e, and f but not in d that cannot be supported by the reported tree. The reason why the neighbor-joining algorithm of the study chose sample c as the first diverging branch of the tree must be because of the large presence of private mutations in sample c, which led to a low Pearson correlation (and hence a greater distance) with the profiles of other samples. This shows that using the Pearson correlation metric is not suitable for this data. Furthermore, directly applying traditional phylogeny reconstruction techniques (for example, neighbor joining) cannot reveal sample heterogeneity.

The tree produced by LICHeE for Case 5, reveals mixed lineages in samples b, e, and f. Interestingly, LICHeE

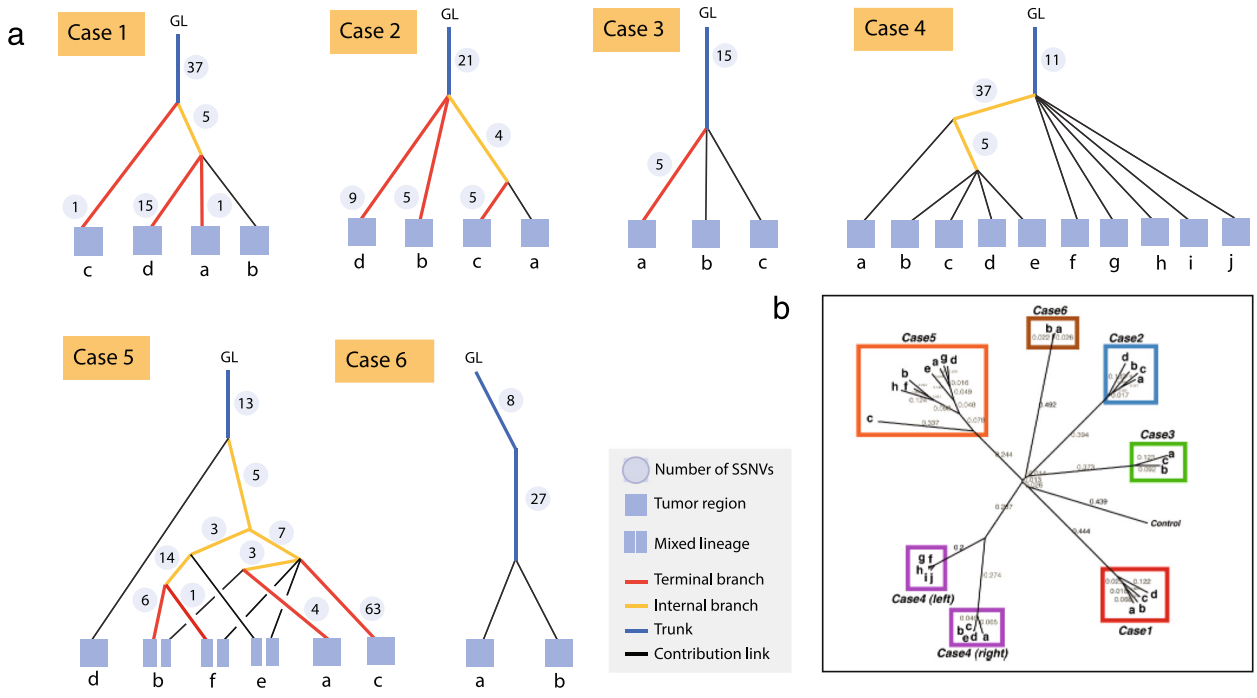

Figure 3 Comparison of lineage trees for HGSC Cases 1 to 6. Phylogenetic trees obtained with LICHeE **(a)** are contrasted with the trees published in [27] **(b)** for each patient in the study. GL, germ line; HGSC, high-grade serous ovarian cancer; SSNV, somatic single nucleotide variant.

detects two clusters in group (a, b, c, e, f) with mean VAFs of [0.29, 0.34 0.3, 0.19, 0.4] and [0.17, 0.14, 0.13, 0.1, 0.13], respectively. The two subclones in each sample are then produced by the divergence of the higher VAF cluster into two branches: one branch containing the lower VAF cluster and the other branch containing the group of mutations (e, f, b). While the presence of group (e, f, b) is weak (supported by three mutations only), the presence of group (b, f) is substantial (supported by 14 mutations) and provides strong evidence for the mixed lineage of b and f. Furthermore, due to the presence of group (a, b), the group (b, f) cannot be assigned as a child of the low VAF cluster (a, b, c, e, f) without violating the VAF phylogenetic constraint. However, since there are only three mutations in group (a, b), eliminating this group may be a reasonable alternative reconstruction. Therefore, multiple topologies are possible in this scenario, each supported by varying degrees of evidence. LICHeE presents the tree that best fits the data given input parameters of expected noise level and minimum mutational support needed for a node. Using LICHeE with different parameters and interacting with the trees can allow users to explore other evidence existing in the data. It is important to point out that, as opposed to the results of the neighbor-joining algorithm that produces trees with lack of evidential support in some branches, each branch in a tree reported by LICHeE reflects the presence of the corresponding SSNV groups in the data.

Lineage tree reconstruction on xenoengraftment data

The study by Eirew et al. [32] used deep-genome and single-cell sequencing to evaluate the clonal dynamics of xenoengraftment of breast cancer tissue into immunodeficient mice. The study applied PyClone [12] to infer mutational clusters representing clonal genotypes and then validated these results using single-cell analysis of two cases, SA494 and SA501. The study used Bayesian phylogenetic inference to reconstruct the evolutionary relationships between the single-cell nuclei. We applied LICHeE to the deep-sequencing VAF data of this study. We then compared the results of LICHeE to the single-cell phylogenetic trees and clonal genotypes for both SA494 and SA501.

Single-cell phylogenetic inference of SA501 passages X1, X2, and X4 reveals an ancestral genotype that branched into two sibling clades A and B. Sequential acquisition of additional mutations in clade B then gave rise to genotypes C, D, and E. Samples X1 and X2 were found to be a mixture of clones with genotypes A, B, C, and D, while, sample X4 was found to be dominated by genotype E and did not contain clones with genotype A (see Figure Two in [32]). As can be seen in Figure 4a, the tree reconstructed

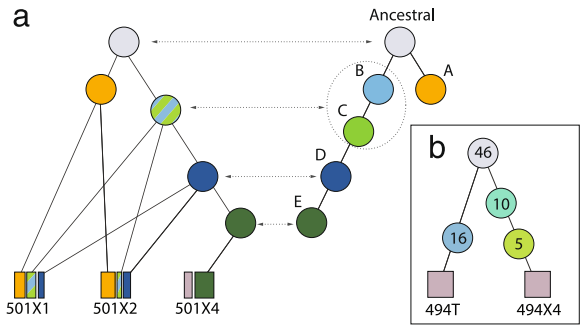

Figure 4 LICHeE reconstruction on breast cancer xenoengraftment data [32]. **(a)** Lineage tree generated by LICHeE (on left) is juxtaposed with the phylogeny derived from single-cell data in the study (on right) for SA501. **(b)** Lineage tree generated by LICHeE for SA494.

by LICHeE mirrors the phylogeny and sample compositions revealed by the study. In particular, it presents the same two sibling clades derived from the ancestral genotype. Samples X1 and X2 both contain subclones with genotype A (with a higher percentage of this genotype in sample X2, as confirmed by the single-cell analysis), while this genotype is missing from sample X4. Similarly, genotype E is found to be private to and dominates sample X4. The only difference in the LICHeE tree is the collapse of genotypes B and C into one cluster; however, examining the bulk CP values reported by the study, we can see that the CP values of these genotypes are highly similar in the three given samples (X1, X2, and X4) and, as a result, cannot get subdivided into two separate clusters during the clustering step (note, the PyClone analysis of the study was simultaneously performed on three additional samples T, X3, and X5, which showed higher CP value differences).

The single-cell analysis of SA494 samples T and X4 reveals two clades, with mutually exclusive sets of mutations, emerging from an ancestral clone present in both samples (see extended data of Figure three in [32]). LICHeE reconstructs the exact same topology (Figure 4b), showing two groups of mutations private to T and X4, respectively. In accordance with the results reported by PyClone, LICHeE also finds two clusters of mutations private to X4 (with the descendent cluster present in about 20% of the sample).

Simulations

We developed a cancer cell lineage simulator to better assess the performance of LICHeE. Our simulator models cancer evolution from normal tissue producing a branching hierarchy of monoclonal cell populations in accordance with the branched-tree cancer evolution model [26,41,42]. Starting with the normal cell population, the simulator iteratively expands the cell lineage tree by introducing (with some given probability) new daughter cell populations corresponding to newly acquired SSNV or CNV events. In particular, in every iteration, each cell population present in the tree can give rise to a new population of cells (with a given randomly generated size) representing a new SSNV with probability P_{SSNV} or a CNV event with probability P_{CNV}. Each cell population can also undergo a cell death event with probability P_{Death}. Each simulated SSNV is randomly associated with a genome location (chromosome and position) and haplotype; the CNVs are associated with a chromosome arm and haplotype and correspond to a duplication of this chromosome arm. For the evaluation, we generated 100 lineage trees, each expanded over 50 iterations, with the following parameters: $P_{\text{SSNV}} = 0.15$; $P_{\text{CNV}} = 0$, 0.1, and 0.18; and $P_{\text{Death}} = 0.06$. This process results in lineage trees with an arbitrary number of branches and nodes (several hundred to thousands of nodes on average). Figure 5 illustrates one such lineage tree.

Multiple samples are then collected from each lineage tree. Each sample consists of several cell populations (nodes) of the tree, where each such cell population represents a subclone in the sample. We implemented two sampling schemes: *randomized* sampling and *localized* sampling. The randomized sampling process selects a random subset of nodes from the tree for each sample; on the other hand, the localized sampling process is meant to mimic biopsies from spatially distinct sites and selects nodes such that samples mostly contain subclones from distinct branches of the simulated tree. Localized sampling for n samples is achieved by selecting n disjoint subtrees in the simulated tree using an approach based on breadth-first search, which finds disjoint subtree roots as high on the tree as possible (if n disjoint subtrees do not exist in the tree, the maximum number of subtrees is used and some samples are obtained from the same subtree in a round-robin fashion). Figure 5 illustrates the localized sampling procedure for ten samples.

Given a selected subset of cell populations, the sample is then created by obtaining a fraction of the cells from each population by sampling from a multinomial distribution with probabilities corresponding to the cell population sizes. For randomized sampling, we select up to five subclones for each sample. For localized sampling, there can be up to five subclones from the same distinct subtree, exactly one subclone from a neighboring subtree, and some fraction of normal cells to represent normal contamination (the fraction is randomly selected to be from 0% up to 20% of the sample). To determine how the performance of LICHeE degrades as the number of SSNVs located in CNV regions grows, we ran several experiments with increasing numbers of CNV events. In particular, we set $P_{\text{CNV}} = [0, 0.1, 0.18]$, which resulted in a total of 0%,

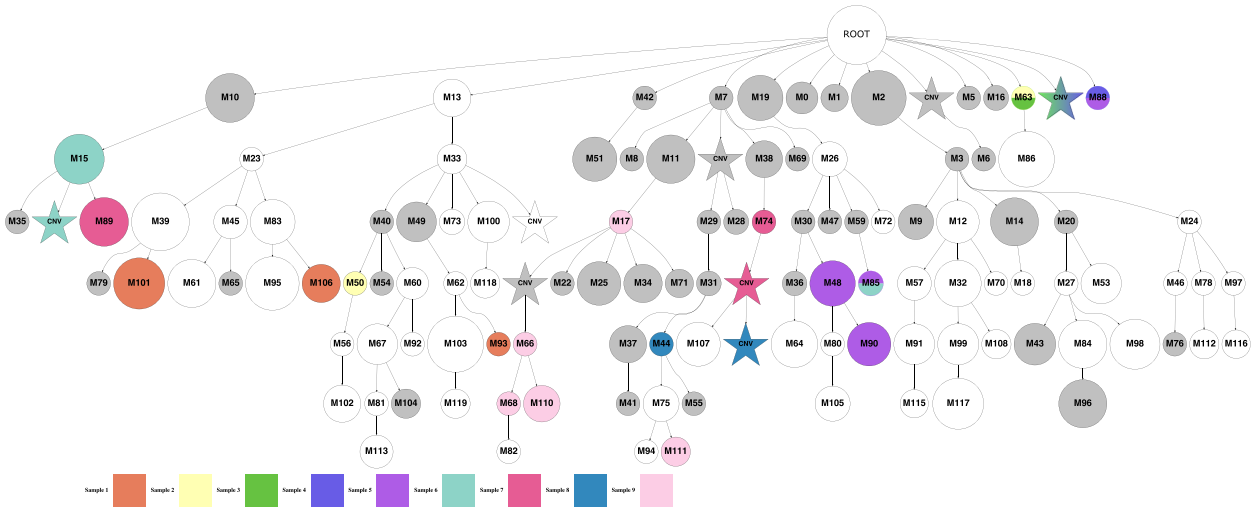

Figure 5 Example of a small simulated lineage tree (with SSNV and CNV events) with localized sampling (nine tumor samples in total indicated by different colors). Each node represents a cell population that contains every mutation in its lineage. Gray indicates dead cell populations. CNV, copy number variant; SSNV, somatic single nucleotide variant.

65%, and 80% of SSNVs being affected by CNVs in the collected samples, respectively. Given the sample cell population counts, we can compute the VAF of each SSNV in this sample. In the absence of CNVs, the VAF of a given SSNV is simply the fraction of the cells containing the SSNV out of all the cells in the sample. When CNVs are present, we count the number of haplotypes containing the SSNV and the reference allele across all cells of the sample. Given an SSNV, we consider the chain of mutations present in each cell population affecting its genome position. Let H_P^v be the number of haplotypes containing the SSNV in population P and H_P^r be the number of haplotypes containing the reference allele. The VAF of a given SSNV M is then:

$$\mathrm{VAF}(M) = \frac{\sum_{P=1}^{N} P_n \cdot H_P^v}{\sum_{P=1}^{N} P_n \cdot \left(H_P^v + H_P^r\right)},$$

where N is the total number of selected populations and P_n is the number of cells selected from population P.

Finally, given the true VAFs of each SSNV, we add sampling and sequencing noise to each value. In particular, to simulate reads covering each SSNV position, we sample the VAFs from the binomial distribution $B(n, p)$, where p is the true VAF of each SSNV and n is the total simulated read coverage (100×, 1,000×, and 10,000×); the generated frequencies have a mean p and variance $p(1-p)/n$. We simulate a Q30 (1 in 1,000) base call sequencing accuracy.

Given the simulated VAFs, we first assessed the performance of LICHeE in SSNV calling. Table 1 presents LICHeE's sensitivity in calling mutations across the samples (that is, the number of SSNVs with correctly identified sample presence patterns out of all the simulated SSNVs) given true VAF values and for coverages of 100×, 1,000×, and 10,000×. As expected, higher coverage results in higher sensitivity. The method achieves 94% to 99% sensitivity across all the experiments (more detailed results are reported in Additional file 3).

Next we compared the topology of the reported lineage trees and the simulated trees and measured LICHeE's accuracy in reconstructing the ancestor–descendant and sibling relationships. For every pair of mutations in the simulated tumor hierarchy, we checked if LICHeE preserved the relationship between them in the generated top tree. More specifically, we checked for the following two types of violations: 1) if an ancestor–descendant relationship was inverted or became a sibling relationship, and 2) if a sibling relationship became an ancestor–descendant relationship. Tables 2 and 3 summarize the accuracy of these metrics (more detailed results are reported in Additional file 3). Since LICHeE may remove nodes from the network representing non-robust SSNV groups if no valid trees are found during the search (as described in 'Materials and methods'), not all of the simulated mutations will be present in the final tree. We report the percentage of the SSNVs and of simulated ancestor–descendant (AD) and sibling (Sib) mutation pairs that are present in the tree. As expected, the number of mutations present in the trees decreases with higher numbers of input samples, lower coverage, and the presence of CNVs (which can significantly alter the VAF values, causing the violation of the VAF ordering constraints). In particular, with a low coverage of 100× and 15 samples, only 81% of SSNVs are preserved in the tree. Therefore, LICHeE is best applied to data with higher coverage when the

Table 1 Sensitivity of assignment to somatic single nucleotide variant groups on simulated data

Number of samples	Number of simulated SSNVs	Coverage			
		True VAF (+CNV)	10,000× (+CNV)	1,000× (+CNV)	100× (+CNV)
5	26.6 (42.2)	99.2 (96.9)	98.9 (96.7)	99.1 (96.7)	97.2 (95.5)
	25 (44.5)	99.5 (97.4)	99.1 (97.3)	99.3 (97.4)	97.4 (95.9)
10	43.9 (89.7)	99.2 (97.8)	98.5 (97.5)	98.5 (97.7)	95 (96)
	40.7 (88.6)	99.5 (97.5)	99.1 (97.3)	99 (97.3)	96.3 (95.9)
15	53.6 (131.1)	98.8 (96.9)	98.2 (96.7)	97.9 (96.5)	94.2 (95.4)
	51 (136.1)	99.1 (97.6)	98.7 (97.4)	98.5 (97.2)	94.5 (95.8)

Values indicate the number of correctly assigned SSNVs out of the total number of SSNVs collected in each experiment (number of simulated SSNVs). Results are shown for 5, 10, and 15 samples given true VAFs, 10,000×, 1,000×, and 100× coverage data obtained with localized (top row in each pair) and randomized (bottom row) sampling, from trees without CNVs and with approximately 80% of SSNVs in CNV regions (in parentheses). All values are averaged over 100 simulated trees. CNV, copy number variant; SSNV, somatic single nucleotide variant; VAF, variant allele frequency.

number of input samples is high. For instance, with a higher coverage of 1,000×, 91% to 94% of SSNVs and 83% to 88% of mutation pairs are present in the trees with 15 samples (although this metric drops with the presence of CNVs to the ranges 91% to 92% for SSNVs and 66% to 70% for SSNV pairs).

Since LICHeE groups mutations with the same presence patterns across samples and similar VAFs, only the mutations occurring in a different set of samples or with significantly different VAFs will be placed in distinct nodes of the tree. We report the percentage of such ancestor–descendant pairs (AD-Ord) in Tables 2 and 3. We then evaluate how many ordered mutations preserved the correct ancestor–descendant relationship (AD-Corr). Across all the experiments without CNVs, we get 99% to 100% correctness (with less than 1% of pairs being in reversed order). We see 92% to 96% correctness in experiments with 80% of SSNVs located in CNV regions and 1,000× coverage. AD mutations that were not ordered or grouped in the same node, will be siblings in the reconstructed tree (AD →Sib). We find that the vast majority of such mutation pairs involve private mutations, whose placement in the tree is usually under-constrained (that is, multiple tree nodes can serve as ancestors to such mutation groups). Finally, we can see that the reverse violation of sibling mutations being placed into ancestor–descendant nodes (Sib→AD) is also very rare (up to 7% across all the experiments). Therefore, we conclude that the trees reconstructed by LICHeE provide highly accurate ordering of the mutations in its nodes.

Conclusions

LICHeE has been designed to automatically infer cell lineages of multiple tumor samples and the sample subclone decomposition. Our analysis shows that LICHeE is highly effective in reconstructing the phylogenies and uncovering the heterogeneity of previously published datasets and in simulations, improving not only upon traditional tree-building methods, but also on recent developments specialized for cancer data. Currently LICHeE works with deep-sequencing data that provide VAF estimates with low variance (as well as on CP values that can be obtained from existing tools and can correct for CNVs, LOH, and sample purity). SSNV data obtained from deep whole-genome sequencing, targeted resequencing of informative SSNVs, or exome sequencing in tumors with a high degree of somatic SSNVs present in exomes, should be appropriate inputs to LICHeE. Several directions for future work are open: extension of this method to lower coverage whole-genome sequencing data and incorporation of aneuploidies and large CNVs directly into the model.

Materials and methods

Grouping and clustering somatic single nucleotide variants

When partitioning SSNVs into groups based on sample occurrence, each SSNV is first associated with a binary sample profile denoting its presence or absence in each sample. Given S samples from an individual, the *binary profile* is defined as a binary sequence of length S where the ith bit is set to 1 if this SSNV is called in the ith sample, and is 0 otherwise (for example, given five samples, an SSNV with the profile 01011 is called in samples 2, 4, and 5). SSNVs with the same profile are assigned to the same SSNV group (for example, a group with the profile 01101 will contain all the SSNVs occurring in samples 2, 3, and 5). The group with the profile consisting entirely of 1s will contain SSNVs that occur in all the samples and are, therefore, germ-line variants (assuming that the sample set includes a normal control sample).

Table 2 Topological ancestor–descendant and sibling relationship reconstruction on simulated data

Samples	Cov	Trees	% SSNVs	% AD	% AD-Ord	% AD-Corr	% AD→Sib (−priv)	% Sib	%Sib-Corr	% Sib→AD (−priv)
5	t-VAF	94	98.9	99	41.5	100	30.2 (2.6)	98.2	83.2	7.3 (1.5)
		98	99.5	99.5	40.4	100	26.3	99.2	84.8	6.6 (0.9)
	10K	95	98.8	98.7	40.4	99.9	30.5 (2.5)	97.8	82.5	7.7 (1.4)
		98	99.2	98.7	39	100	27.4	98.8	84.7	7 (1.3)
	1K	95	98.6	97.7	39.7	100.0	30.5 (2.3)	97.1	82.8	7.2 (1.3)
		98	99.4	99.5	38.1	99.7	29.2	98.9	83.9	7.5 (2.2)
	100	97	96.8	93.2	38.7	99.8	31.4 (3.5)	93.1	81.8	7.9 (1.7)
		97	97.3	94.8	35.2	99.7	30.1	95.3	83.7	7.2 (2.2)
10	t-VAF	97	97.5	96.2	58.8	100	43.1 (2.2)	96.9	94.4	3.3 (0.2)
		95	97.7	96.5	58.2	100	38.1 (1.9)	96.7	93.5	4.2 (1.4)
	10K	98	96.5	94.4	57.3	100	44.9 (4)	95.2	94	3.6 (0.4)
		96	97	96.1	56.5	100	39.8 (2.3)	96.2	93.3	4.3 (1.2)
	1K	98	97.4	96.4	57.5	99.9	45.2 (3.9)	96.8	94.1	3.7 (0.4)
		96	97	95.7	57.4	99.9	38.7 (2.4)	96	92.7	4.8 (1.5)
	100	93	90.8	75.7	52.3	99.7	38.6 (4.4)	78.0	93.9	3.4 (0.7)
		91	90.5	80.4	47.9	99.5	40.3 (5.7)	83.3	92.8	4.4 (1.5)
15	t-VAF	99	91.7	85	63.1	100	40.8 (2.2)	88	96.6	2 (0.2)
		96	92.1	87.5	61.7	100	41.5 (2)	89.1	96.9	2 (0.3)
	10K	98	92.3	85.3	61.2	100	44.3 (3.5)	87.6	96.2	2.4 (0.3)
		100	90.6	83.6	59.1	100	41.8 (2.6)	85.8	96.6	2.1 (0.3)
	1K	93	94.7	83.5	61.5	99.9	42.7 (4.3)	85.4	96.1	2.6 (0.6)
		100	91.8	85.4	59.1	100	43.5 (2.9)	87.8	96.7	2.1 (0.3)
	100	92	81.1	55.2	48.4	99.4	37.4 (5)	61.3	96.1	2.1 (0.3)
		98	81.8	57.1	46.1	99.8	36.5 (2.2)	63.1	96.2	2.2 (0.4)

Results are shown for 5, 10, and 15 samples given true VAFs, 10,000×, 1,000×, and 100× coverage data (without CNVs) obtained with localized (top row in each pair) and randomized (bottom row) sampling. All values are averaged over the number of reconstructed trees (Trees) out of 100 simulated trees. The following metrics are presented: SSNVs present in the tree (% SSNVs), ancestor–descendant pairs of mutations in the tree (AD), ordered AD pairs (AD-Ord), correctly ordered AD pairs (AD-Corr), unordered AD pairs reconstructed as siblings (AD→Sib) (with and without private mutation nodes), sibling pairs of mutations in the tree (Sib), correctly reconstructed sibling pairs (Sib-Corr), sibling pairs reconstructed as AD (Sib→AD) (with and without private mutation nodes).
AD, ancestor–descendant; CNV, copy number variant; Corr, correctly ordered; Cov, coverage; Ord, ordered; priv, private mutation nodes; Sib, sibling; SSNV, somatic single nucleotide variant; t-VAF, true variant allele frequency; VAF, variant allele frequency.

The SSNV binary profiles can be passed as input or computed from SSNV VAFs as follows. First two hard VAF thresholds, T_{present} and T_{absent}, are used to determine if an SSNV is robustly present or absent from a sample. An SSNV profile is classified as robust if its VAF is above or below the two thresholds, respectively, and if at least a minimum number of other robust SSNVs (default set to 1) have the same binary profile. All other SSNVs are considered non-robust and are assigned to a group as follows. Given a non-robust SSNV m, its VAF across samples can either fall below (marked 0), above (marked 1), or in between (marked*) the thresholds T_{present} and T_{absent}, resulting in a profile such as 01*11. The candidate groups, to which m can be assigned, must have an identical profile in all the samples that are marked 0 or 1 (for example, for profile 01*11, two valid candidate groups are 01111 and 01011). Since m can be assigned to more than one target group, we consider that the group containing a robust SSNV that is most similar in VAF to m is the best candidate. The following metric is used to compute the similarity between two SSNVs m and n:

$$\text{sim}_{mn} = \sum_{i \in \text{samples}} \frac{\min(m.\text{VAF}_i, n.\text{VAF}_i)}{\max(m.\text{VAF}_i, n.\text{VAF}_i)}, \quad (1)$$

where $m.\text{VAF}_i$ is the VAF of m in sample i. If the maximum similarity is higher than a given threshold, m is assigned to the group of $\text{argmax}_{n \in \text{Candidates}} \text{sim}_{mn}$. Unassigned non-robust SSNVs will form new profile groups.

Table 3 Topological ancestor–descendant and sibling relationship reconstruction on simulated data in the presence of copy number variants

Samples	Trees	% SSNVs	% AD	% AD-Ord	% AD-Corr	% AD→Sib (−priv)	% Sib	% Sib-Corr	% Sib→AD (−priv)
5	95	96.2	89.1	14.4	93.7	20 (0.4)	92.6	80.8	2.4 (0)
	98	97.42	91.96	10.74	96.1	0.20	94.85	80.46	2 (0.02)
10	91	94.8	77.1	14.6	93.5	24.9 (0.6)	90.9	91.8	1.3 (0)
	92	94.55	78.97	11.95	93.1	0.41	90.5	91.7	1.1 (0.01)
15	97	91.8	65.9	11.1	92.5	23.3 (0.4)	85.5	94.6	0.7 (0)
	94	92.9	69.6	10.8	94.5	24.2 (0.35)	87.5	94.8	0.6 (0.01)

Results are shown for 5, 10, and 15 samples given 1,000× coverage data obtained with localized (top row in each pair) and randomized (bottom row) sampling with approximately 80% of SSNVs in CNV regions. All values are averaged over the number of reconstructed trees.
AD, ancestor–descendant; CNV, copy number variant; Corr, correctly ordered; Cov, coverage; Ord, ordered; priv, private mutation nodes; Sib, sibling; SSNV, somatic single nucleotide variant.

We minimize the number of such new groups by formulating this task as a set cover problem. In particular, let X be the set of all unassigned SNVs. Denote Y as the set of subsets of X, where each subset represents mutations that can be assigned to the same potential target group. The target groups are a list of all possible binary profiles that the SSNVs can be assigned to, obtained by substituting all *'s by 0 or 1. We want to find the minimum number of target groups (that is, the smallest subset of Y), which cover all mutations in X. This problem is known to be NP-complete and searching for the exact solution is not feasible. Instead we apply the standard greedy algorithm by choosing (at each stage) the target group that covers the largest number of non-robust SSNVs. For SSNVs whose targets are not supported by any other mutations, we convert each * to 1 or 0 depending on whether the VAF is closest to T_{present} or T_{absent}, respectively.

Once the SSNVs are partitioned into groups, the SSNVs of each group are further clustered based on their sample VAFs. Each SSNV group is associated with a matrix M of VAFs of size $n \times s$, where n is the number of SSNVs in this group and s is the number of represented samples (for example, $s = 3$ for a group with the profile 0110001). The expectation-maximization clustering algorithm for

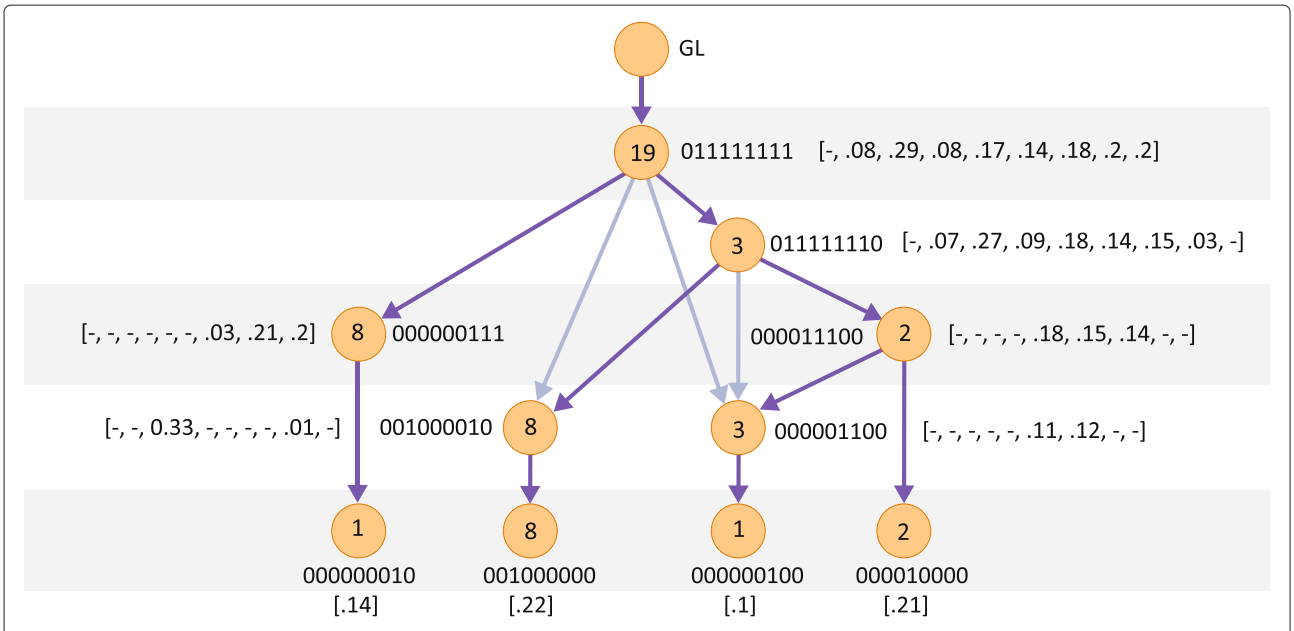

Figure 6 Evolutionary constraint network for patient EV007 with 8 tumor samples and 55 SSNVs. Each node is associated with the binary profile of its corresponding SSNV group, the VAF centroid vector, and the number of SSNVs assigned to it. The edges represent the potential precedence relationships between the node SSNVs. The spanning tree reported for patient EV007 is highlighted (see 'Results and discussion'). GL, germ line; SSNV, somatic single nucleotide variant; VAF, variant allele frequency.

Gaussian mixture models is run on the resulting VAF matrix M using [43]. The result is a set of SSNV clusters with an associated VAF centroid vector, $\overline{\text{VAF}}$. To handle the high variance in the VAF data due to noise, some of the resulting clusters are eliminated (based on size) or collapsed with neighboring clusters, based on the distance between their centroid vectors.

Evolutionary constraint network construction

Given the clusters of each SSNV group, we construct an evolutionary constraint network to capture valid evolutionary timing relationships between the mutations of each cluster pair. The network is a DAG, where each node corresponds to an SSNV cluster (except the root, which represents the germ line) and each edge between two nodes, $(u \rightarrow v)$ denotes that parent node u could be an evolutionary predecessor of child node v (that is, that SSNVs in cluster u could have 'happened before' SSNVs in cluster v). In particular, an edge $(u \rightarrow v)$ is added only if the nodes satisfy the following two constraints $\forall i \in$ samples (which guarantee that the network will be acyclic):

$$\begin{array}{l} (1)\ \ u.\overline{\text{VAF}}_i \geq v.\overline{\text{VAF}}_i - \epsilon_{uv} \text{ and} \\ (2)\ \ \text{if } u.\overline{\text{VAF}}_i = 0, \quad v.\overline{\text{VAF}}_i = 0, \end{array} \tag{2}$$

where ϵ_{uv} is the VAF noise error margin (note, the error margin is the maximum of the sum of the standard errors for sample i of the two clusters and a configurable parameter). In the resulting network, each node will have at least one parent (since all nodes can be connected to the root). We avoid checking all node pairs, by organizing the nodes into levels according to the Hamming weight (that is, the number of 1's) of the binary group profile to which they belong. Nodes that are in the same level and have conflicting binary profiles cannot satisfy the above constraints. Nodes from different levels can only be connected such that the node in the higher level is the parent of the node in the lower level. Finally, for nodes that are in the same level and have the same binary profile, the edge is added in the direction that minimizes VAF_{ERR}, where:

$$\text{VAF}_{\text{ERR}_{u\rightarrow v}} = \sum_{i \in \text{ samples}} \mathbf{1}_{\text{ERR}_{u\rightarrow v}}(i) \cdot (v.\overline{\text{VAF}}_i - u.\overline{\text{VAF}}_i)^2 \tag{3}$$

with the indicator function:

$$\mathbf{1}_{\text{ERR}_{u\rightarrow v}}(i) = \begin{cases} 1, & v.\overline{\text{VAF}}_i > u.\overline{\text{VAF}}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Figure 6 illustrates the constraint network produced for the dataset of ccRCC patient EV007 (described in 'Results and discussion').

Phylogenetic tree search

By the constraint network construction, a valid lineage tree T of the SSNV clusters (that is, a tree that does not violate the three constraints of the perfect phylogeny model) must be a spanning tree of the network that satisfies the following requirement $\forall$ nodes $u \in T$:

$$\forall i \in \text{samples} \ : \sum_{v \text{s.t.} (u\rightarrow v)\in T} v.\overline{\text{VAF}}_i \leq u.\overline{\text{VAF}}_i + \epsilon. \tag{5}$$

That is, the sum of the VAF centroids of all the children must not exceed the centroid of the parent. We use inequality here since our method does not require all the true lineage branches to have been observed. To tolerate noise in the VAF data, we relax the constraint by allowing the sum of children VAFs to exceed the parent by an error margin ϵ. Given such a valid lineage tree, each sample can then be decomposed into subpopulations by enumerating all the paths in the tree starting with the germ-line root and ending in the last node containing mutations in that sample.

The problem of finding all such trees is equivalent to the problem of finding *all spanning trees* of the constraint network DAG for which Eq. (5) holds. We have extended the Gabow and Myers spanning tree search algorithm [44] to generate all such spanning trees. The original algorithm generates all spanning trees of a directed graph using backtracking and an efficient bridge edge detection method based on depth-first search (DFS). Our extension consists of enforcing Eq. (5) during the tree search by terminating the expansion of a given tree and backtracking as soon as an edge violating Eq. (5) is added (see Algorithm 1). Same as the original algorithm, this search runs in $O(|V| + |E| + |E|N)$ time, where $|V|$ is the number of nodes, $|E|$ is the number of edges, and N is the number of spanning trees in the network. While the runtime of the program depends on the number of spanning trees in the constraint network, in practice the search is very fast (taking of the order of a few seconds). However, since, in theory, it is possible for the algorithm to take longer on datasets that result in constraint networks with many spanning trees, we provide a bound on the maximum number of lineage trees to generate to avoid searches that are too long. Similarly, we also have a high bound on the number of calls to the GROW procedure in Algorithm 1. We expect these scenarios to be very rare in typical validation datasets. To reduce the search space, we also optionally constrain the placement of private mutation nodes in the constraint network to their closest valid predecessors.

Algorithm 1 Finding All Lineage Trees

```
1: Initialization: f ← empty list, L ← null // stores the last tree output
2: procedure LINEAGE TREE SEARCH(N) // N is a constraint network rooted at r
3:     Tree t ← new empty Tree
4:     t.ADDNODE(r)
5:     add all edges (r → v) ∈ N to f
6:     GROW(t)
7: procedure GROW(t, N)
8:     if t contains all the nodes in N then
9:         L ← t
10:        output L
11:    else
12:        s ← empty list
13:        b ← false
14:        while (not b and f not empty) do
15:            // e defined as (e.From → e.To)
16:            Edge e ← f.REMOVELAST()
17:            Node v ← e.To
18:            t.ADDNODE(v)
19:            t.ADDEDGE(e.From → v)
20:            // return true if Eq. (5) is satisfied for node e.From
21:            if t.CHECKCONSTRAINT(e.From) then
22:                add all edges (v → w), w ∉ t to f
23:                remove all edges (w → v), w ∈ t from f
24:                GROW(t)
25:                if number of returned trees > max_trees return
26:                remove all edges (v → w), w ∉ t from f
27:                add all edges (w → v), w ∈ t to f
28:            t.REMOVEEDGE(e.From → e.To)
29:            N.REMOVEEDGE(e.From → e.To)
30:            s.ADD(e)
31:            if ∃ an edge (w → v) s.t. w not a descendant of v in L then
32:                b ← false
33:            else b ← true
34:        for all edges e starting from the end of s do
35:            remove e from s, add e to f, add e to N
```

The above search algorithm will find all spanning trees for which Eq. (5) holds locally at each individual node; however, it is possible that to satisfy this constraint, the centroid values are deviated (within ϵ) in a globally inconsistent way. Therefore, to enforce consistency, we apply one additional requirement to the trees returned by the search. In particular, we formulate the following quadratic programming (QP) problem to find a set of $e_{v,i}$ such that for every sample i and node v:

$$\begin{aligned} &\text{minimize } \textstyle\sum_v \sum_i e_{v,i}^2 \\ &\text{subject to} \\ &\textstyle\sum_{v \text{ s.t.}(u \to v) \in T} \left(v.\overline{\text{VAF}_i} + e_{v,i} \right) \leq u.\overline{\text{VAF}_i} + e_{u,i} \text{ and} \\ &|e_{v,i}| \leq \epsilon, e_{v,i} \leq v.\overline{\text{VAF}_i} \end{aligned} \tag{6}$$

If no solution exists, the tree is considered invalid. Since multiple valid lineage trees can be generated, we rank them using the resulting $e_{v,i}$ solution, which corresponds to how well each tree fits the VAF data. The top-ranking tree will be the tree with the minimum sum of squared deviations: $\sum_v \sum_i e_{v,i}^2$. Since for certain networks the total number of valid lineage trees can be large, it is impractical to run a QP program on each tree. Therefore, we first rank the trees based on the sum of squared deviations computed locally at each node, and then run QP on the resulting top k trees.

Multiple lineage trees can support a dataset equally well since the placement of certain nodes in the tree may be ambiguous using perfect phylogeny SSNV ordering constraints only (especially when not all lineage branches have been observed). In these cases, more sophisticated custom evaluation criteria would be required to rank the trees. We do not address defining such criteria in this work. Instead we report all the produced trees to the user ranked by their associated score (the number of trees reported is configurable) and provide a GUI to allow the user to easily explore the differences in the topologies (see 'Visualization' for details). We expect that under any optimality criteria, the suboptimal trees can also represent signals in the data of potential biological significance.

It is also possible that no valid lineage trees are found during the search. This can happen if the noise error margin is too narrow or the network contains nodes corresponding to SSNV groups with a misclassified binary profile. In this case, the network is adjusted and searched again. Currently the network adjustment procedure will remove one by one all the nodes that belong to non-robust SSNV groups (smallest nodes first). Other adjustments, such as increasing the noise error margin, are also possible but not currently implemented.

Visualization

The constraint network and the phylogenetic trees can be visualized and interacted with in GUI form. The JUNG graph library [45] is used to generate the resulting graphs. When visualizing lineage trees, each input sample appears as a leaf in the tree and is connected to the nodes that contain SSNVs present in the sample. By clicking on the nodes of a tree, it is possible to obtain additional information about each node. The information displayed about an internal (non-sample) tree node consists of its binary group profile, its cluster centroid and standard deviation vectors, and the list of the SSNVs in its cluster (these

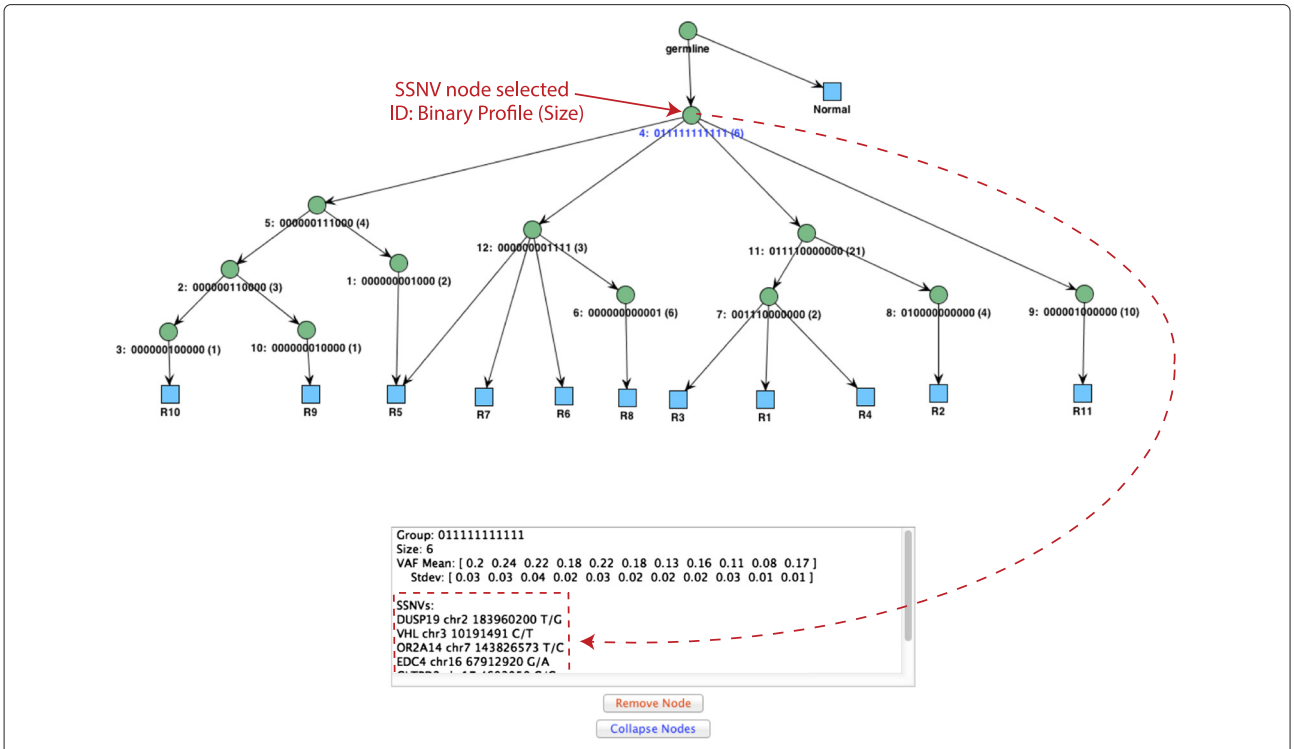

Figure 7 LICHeE graphical user interface output for top lineage tree of ccRCC patient RK26 with node selection. Information is displayed about the SSNV members of the selected node. ccRCC, clear-cell renal cell carcinoma; SSNV, somatic single nucleotide variant; Stdev, standard deviation; VAF, variant allele frequency.

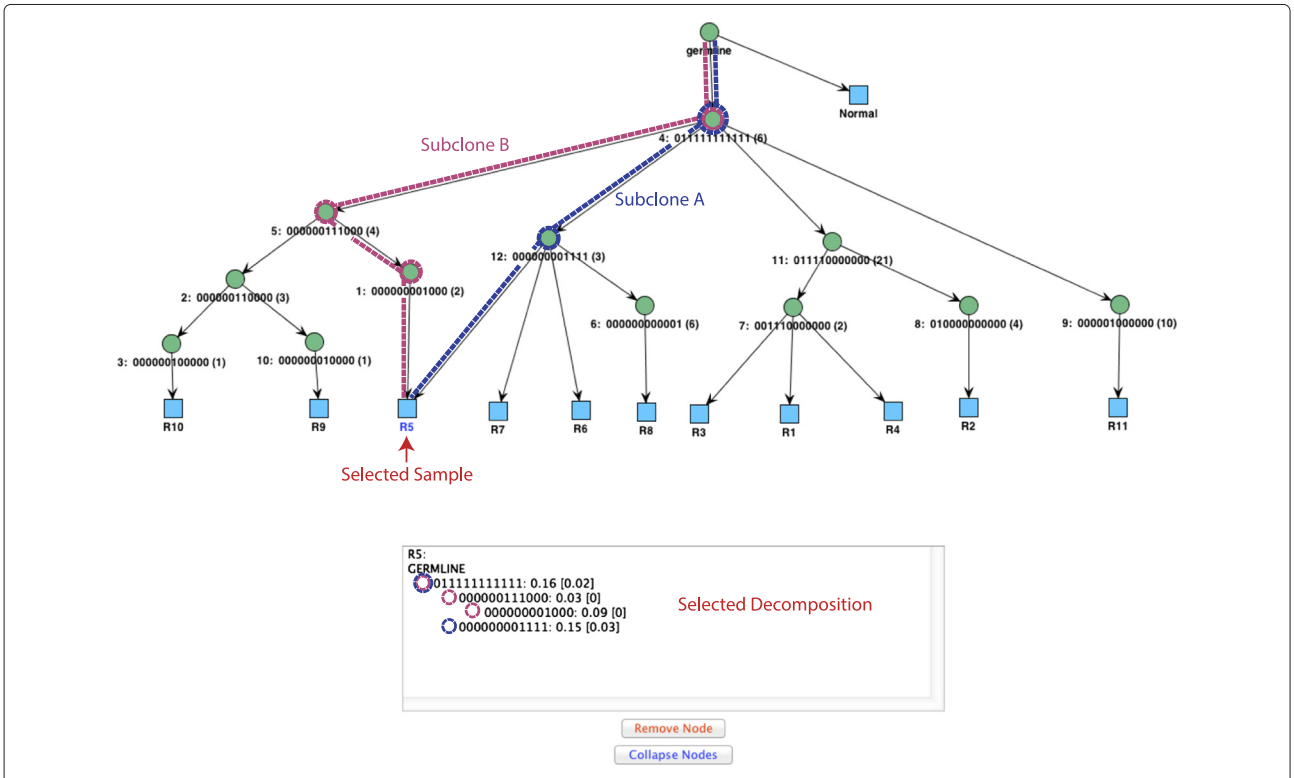

Figure 8 LICHeE graphical user interface output for top lineage tree of ccRCC patient RK26 with sample selection. Information is displayed about the sample composition. ccRCC, clear-cell renal cell carcinoma.

SSNVs can be annotated with information from public databases such as COSMIC, TCGA, etc.). If a sample leaf node is clicked, the information displayed consists of the lineage of this specific sample obtained by doing a DFS traversal of the tree starting with the germ-line root. Finally, the user can rearrange the nodes in the tree, as well as remove nodes and collapse nodes (provided they are clusters of the same group). See Figures 7 and 8 for several examples. In addition to the GUI, the program reports the number of trees found and the score of the highest-ranking tree. The user can control how many trees to display.

Implementation

The LICHeE algorithm was implemented in Java. It is open source and freely available online at [33].

Additional files

Additional file 1: Evaluation of Phylosub. Performance evaluation of the Phylosub [29] program on the ccRCC dataset.

Additional file 2: Experimental details of the ccRCC, HGSC, and xenoengraftment comparison. It includes a description of the dataset and LICHeE parameter settings.

Additional file 3: Detailed simulation results.

Abbreviations

AD: ancestor–descendant; ccRCC: clear-cell renal cell carcinoma; CNV: copy number variant; Corr: correctly ordered; Cov: coverage; CP: cell prevalence; DAG: directed acyclic graph; GUI: graphical user interface; HGSC: high-grade serous ovarian cancer; Ord: ordered; QP: quadratic programming; Sib: sibling; SSNV: somatic single nucleotide variant; VAF: variant allele frequency.

Competing interests

SB is a founding advisor of DNAnexus Inc, and member of the Scientific Advisory Boards of 23andMe and Eve Biomedical.

Authors' contributions

VP, RS, and SB developed the method. VP implemented the LICHeE program and simulator and drafted the manuscript. RS, SB, DK, and IH contributed to the manuscript. VP, RS, and SB performed the evaluation and analysis. All authors were involved in discussions. All authors read and approved the final manuscript.

Acknowledgements

The authors would like to thank Arend Sidow for valuable discussions and Aaron C Abajian for contributing to the simulation studies. VP was supported by the Stanford-KAUST grant. RS and IH were also supported by Natural Sciences and Engineering Research Council of Canada (NSERC) Postdoctoral Fellowships. DK was supported by an STMicroelectronics Stanford Graduate Fellowship. This work was funded by a grant from KAUST to SB.

Author details

[1]Department of Computer Science, Stanford University, Stanford, CA, USA. [2]Department of Pathology, Stanford University School of Medicine, Stanford, CA, USA.

Received: 20 December 2014 Accepted: 7 April 2015
Published online: 06 May 2015

References

1. Yates LR, Campbell PJ. Evolution of the cancer genome. Nat Rev Genet. 2012;13:795–806.
2. Marusyk A, Almendro V, Polyak K. Intra-tumour heterogeneity: a looking glass for cancer?. Nat Rev Cancer. 2012;12:323–34.
3. Bamford S, Dawson E, Forbes S, Clements J, Pettett R, Dogan A, et al. The cosmic (catalogue of somatic mutations in cancer) database and website. Br J Cancer. 2004;91:355–8.
4. Collins FS, Barker AD. Mapping the cancer genome. Sci Am. 2007;296:50–7.
5. Fisher R, Pusztai L, Swanton C. Cancer heterogeneity: implications for targeted therapeutics. Br J Cancer. 2013;108:479–85.
6. Schuh A, Becq J, Humphray S, Alexa A, Burns A, Clifford R, et al. Monitoring chronic lymphocytic leukemia progression by whole genome sequencing reveals heterogeneous clonal evolution patterns. Blood. 2012;120:4191–6.
7. Newburger DE, Kashef-Haghighi D, Weng Z, Salari R, Sweeney RT, Brunner AL, et al. Genome evolution during progression to breast cancer. Genome Res. 2013;23:1097–108.
8. Carter SL, Cibulskis K, Helman E, McKenna A, Shen H, Zack T, et al. Absolute quantification of somatic DNA alterations in human cancer. Nat Biotechnol. 2012;30:413–21.
9. Campbell PJ, Pleasance ED, Stephens PJ, Dicks E, Rance R, Goodhead I, et al. Subclonal phylogenetic structures in cancer revealed by ultra-deep sequencing. Proc Natl Acad Sci. 2008;105:13081–6.
10. Nik-Zainal S, Van Loo P, Wedge DC, Alexandrov LB, Greenman CD, Lau KW, et al. The life history of 21 breast cancers. Cell. 2012;149:994–1007.
11. Shah SP, Roth A, Goya R, Oloumi A, Ha G, Zhao Y, et al. The clonal and mutational evolution spectrum of primary triple-negative breast cancers. Nature. 2012;486:395–9.
12. Roth A, Khattra J, Yap D, Wan A, Laks E, Biele J, et al. Pyclone: statistical inference of clonal population structure in cancer. Nat Methods. 2014;11:396–8.
13. Oesper L, Mahmoody A, Raphael BJ. THetA: inferring intra-tumor heterogeneity from high-throughput DNA sequencing data. Genome Biol. 2013;14:R80.
14. Ha G, Roth A, Khattra J, Ho J, Yap D, Prentice LM, et al. Titan: inference of copy number architectures in clonal cell populations from tumor whole-genome sequence data. Genome Res. 2014;24:1881–93.
15. Hajirasouliha I, Mahmoody A, Raphael BJ. A combinatorial approach for analyzing intra-tumor heterogeneity from high-throughput sequencing data. Bioinformatics. 2014;30:78–86.
16. Ding L, Ley TJ, Larson DE, Miller CA, Koboldt DC, Welch JS, et al. Clonal evolution in relapsed acute myeloid leukaemia revealed by whole-genome sequencing. Nature. 2012;481:506–10.
17. Landau DA, Carter SL, Stojanov P, McKenna A, Stevenson K, Lawrence MS, et al. Evolution and impact of subclonal mutations in chronic lymphocytic leukemia. Cell. 2013;152:714–26.
18. McFadden DG, Papagiannakopoulos T, Taylor-Weiner A, Stewart C, Carter SL, Cibulskis K, et al. Genetic and clonal dissection of murine small cell lung carcinoma progression by genome sequencing. Cell. 2014;156:1298–311.
19. Campbell PJ, Yachida S, Mudie LJ, Stephens PJ, Pleasance ED, Stebbings LA, et al. Patterns and dynamics of genomic instability in metastatic pancreatic cancer. Nature. 2010;467:1109–13.
20. Yachida S, Jones S, Bozic I, Antal T, Leary R, Fu B, et al. Distant metastasis occurs late during the genetic evolution of pancreatic cancer. Nature. 2010;467:1114–17.
21. Gerlinger M, Horswell S, Larkin J, Rowan AJ, Salm MP, Varela I, et al. Genomic architecture and evolution of clear cell renal cell carcinomas defined by multiregion sequencing. Nat Genet. 2014;46:225–33.
22. Green MR, Gentles AJ, Nair RV, Irish JM, Kihira S, Liu CL, et al. Hierarchy in somatic mutations arising during genomic evolution and progression of follicular lymphoma. Blood. 2013;121:1604–11.
23. de Bruin EC, McGranahan N, Mitter R, Salm M, Wedge DC, Yates L, et al. Spatial and temporal diversity in genomic instability processes defines lung cancer evolution. Science. 2014;346:251–6.
24. Hajirasouliha I, Raphael BJ. Reconstructing mutational history in multiply sampled tumors using perfect phylogeny mixtures. In: Algorithms in bioinformatics. Berlin, Heidelberg: Springer; 2014. p. 354–67.
25. Salari R, Saleh SS, Kashef-Haghighi D, Khavari D, Newburger DE, West RB, et al. Inference of tumor phylogenies with improved somatic mutation discovery. J Comput Biol. 2013;20:933–44.

26. Gerlinger M, Rowan AJ, Horswell S, Larkin J, Endesfelder D, Gronroos E, et al. Intratumor heterogeneity and branched evolution revealed by multiregion sequencing. N Engl J Med. 2012;366:883–92.
27. Bashashati A, Ha G, Tone A, Ding J, Prentice LM, Roth A, et al. Distinct evolutionary trajectories of primary high-grade serous ovarian cancers revealed through spatial mutational profiling. J Pathol. 2013;231:21–34.
28. Qiao Y, Quinlan AR, Jazaeri AA, Verhaak RG, Wheeler DA, Marth GT, et al. Subcloneseeker: a computational framework for reconstructing tumor clone structure for cancer variant interpretation and prioritization. Genome Biol. 2014;15:443.
29. Jiao W, Vembu S, Deshwar AG, Stein L, Morris Q. Inferring clonal evolution of tumors from single nucleotide somatic mutations. BMC Bioinformatics. 2014;15:35.
30. Deshwar AG, Vembu S, Yung CK, Jang GH, Stein L, Morris Q, et al. PhyloWGS: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. Genome Biol. 2015;16:35.
31. Malikic S, McPherson AW, Donmez N, Sahinalp CS. Clonality inference in multiple tumor samples using phylogeny. Bioinformatics. 2015;31:1349–56.
32. Eirew P, Steif A, Khattra J, Ha G, Yap D, Farahani H, et al. Dynamics of genomic clones in breast cancer patient xenografts at single-cell resolution. Nature. 2014;518:422–6.
33. LICHeE Github Repository. http://viq854.github.io/lichee/.
34. Gusfield D. Efficient algorithms for inferring evolutionary trees. Networks. 1991;21:19–28.
35. Gerstung M, Beisel C, Rechsteiner M, Wild P, Schraml P, Moch H, et al. Reliable detection of subclonal single-nucleotide variants in tumour cell populations. Nat Commun. 2012;3:811.
36. Yost S, Alakus H, Matsui H, Schwab R, Jepsen K, Frazer K, et al. Mutascope: sensitive detection of somatic mutations from deep amplicon sequencing. Bioinformatics. 2013;29:1908–9.
37. Larson DE, Harris CC, Chen K, Koboldt DC, Abbott TE, Dooling DJ, et al. Somaticsniper: identification of somatic point mutations in whole genome sequencing data. Bioinformatics. 2012;28:311–17.
38. Koboldt DC, Zhang Q, Larson DE, Shen D, McLellan MD, Lin L, et al. Varscan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. Genome Res. 2012;22:568–76.
39. Saunders CT, Wong WS, Swamy S, Becq J, Murray LJ, Cheetham RK, et al. Strelka: accurate somatic small-variant calling from sequenced tumor–normal sample pairs. Bioinformatics. 2012;28:1811–17.
40. Van Loo P, Nordgard SH, Lingjærde OC, Russnes HG, Rye IH, Sun W, et al. Allele-specific copy number analysis of tumors. Proc Natl Acad Sci. 2010;107:16910–15.
41. Greaves M, Maley CC. Clonal evolution in cancer. Nature. 2012;481:306–13.
42. Yap TA, Gerlinger M, Futreal PA, Pusztai L, Swanton C. Intratumor heterogeneity: seeing the wood for the trees. Sci Transl Med. 2012;4:127ps10.
43. Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH, et al. The WEKA data mining software: an update. ACM SIGKDD Explorations Newsl. 2009;11:10–18.
44. Gabow HN, Myers EW. Finding all spanning trees of directed and undirected graphs. SIAM J Comput. 1978;7:280–7.
45. O'Madadhain J, Fisher D, Smyth P, White S, Boey YB. Analysis and visualization of network data using JUNG. J Stat Soft. 2005;10:1–35.