ClonEvol Supplementary Methods & Discussion

Contents

Supplementary Methods  2

- Variant clustering and cellular prevalence as input to ClonEvol  2
- Bootstrap resampling  2
- Determining founding clone at sample level and model of clonal seeding  3
- Pruning clonal evolution trees and merging intra-tumor samples  3
- ClonEvol implementation  4
  - Heuristic approach to bootstrapping  4
  - Weighted bootstrapping  4
  - Non-parametric and parametric bootstrapping  5
  - Tree enumeration algorithm  5

Simulation experiments  7

- Ground truth cancer cell fraction and clonal mixtures  7
- Random tree generation  7
- Read count simulation  7
- Incorporation of noise  8
- Non-small and small cluster sizes and total number of variants  8
- Clonal seeding evaluation  8
- Running ClonEvol, LICHeE, Canopy, and PhyloWGS  8

Supplementary Discussion  9

References  11
Supplementary Methods

Variant clustering and cellular prevalence as input to ClonEvol

ClonEvol uses the variant clusters identified by clustering methods and variant allele frequencies (VAFs) to perform clonal ordering. Two prominent approaches in clonal evolution studies are: (i) using only diploid heterozygous variants (variants in regions without copy number alteration), hence excluding copy-altered variants; and (ii) including copy-altered variants. There exist dedicated tools for each approach. When only diploid heterozygous variants are used, VAFs can be estimated as the ratio of the variant read count and total read count and clustering can be performed by tools such as sciClone [1]. When copy-altered variants are included, clustering should be performed using copy-number aware tools such as Pyclone [2], and copy number corrected VAFs can be obtained by dividing the CCFs estimated by such tools by two.

Bootstrap resampling

The CCF of a clone is formulated as the difference of the CCF of its cluster and the total CCFs of the clusters corresponding to its direct subclones, as in Eq. (3), rewritten here as follows.

\[
CCF(\text{clone } Y) = CCF(\text{cluster } Y) - \sum_{Xi \in \text{all direct subclones of } Y} CCF(\text{cluster } X_i)
\]  

(S1)

We estimate the CI of the CCF of clone Y in Eq. (S1) via a bootstrap resampling approach [3]. In each resampling step, a random sample of variants (of the same number as in the cluster) is drawn with replacement from each cluster corresponding to the clones \(X_i\) or \(Y\). Following variant resampling, VAFs of the variants are used to calculate the mean VAF of each cluster and the CCF of each cluster is calculated as twice the mean or median of the resampled VAF of that cluster. We denote by \(CCF(\text{clone } Y)\) the estimate of the CCF for clone \(Y\) in bootstrap sample \(j\). After performing a large number of \(M\) bootstraps, we next calculate a two-sided CI of the CCF of clone \(Y\), by taking sample quantiles of the set of bootstrap estimates \(\{CCF(\text{clone } Y)\}\):

\[
CI(\text{CCF(clone } Y)) = [Q(\text{CCF}_j(\text{clone } Y), \alpha/2), Q(\text{CCF}_j(\text{clone } Y), 1 - \alpha/2)]
\]  

(S2)

where \(Q(a,b)\) is the \(b*100\) percentile of \(a\). Similarly, two one-sided CIs of the CCF of clone \(Y\) are calculated as follows.

\[
CI^- (\text{CCF(clone } Y)) = (-\infty, Q(\text{CCF}_j(\text{clone } Y), 1 - \alpha)]
\]  

(S3a)
Inverting the one-sided CIs of the CCFs in Eqs. (S3a) and (S3b) gives the probability that the bootstrap CCF of clone Y is negative \( p_{neg}(Y) \) or positive \( p_{pos}(Y) \) as follows.

\[
p_{neg}(Y) = p(CCF(\text{clone } Y) < 0) = \frac{\text{count}(CCF(\text{clone } Y) < 0)}{M}
\]

\[
p_{pos}(Y) = p(CCF(\text{clone } Y) > 0) = \frac{\text{count}(CCF(\text{clone } Y) > 0)}{M}
\]

We use \( p_{neg}(Y) \) as the probability that the clonal ordering of parental clone Y violates the sum rule. The probability that clone Y obeys the sum rule is calculated as \( 1 - p_{neg}(Y) \).

Determining founding clone at sample level and model of clonal seeding

A critical question in clonal evolution study is to determine the seeding clones and models of cancer seeding across samples. For example, one may be interested in whether a metastasis is seeded by one (monoclonal seeding) or more (polyclonal seeding) clones from a primary tumor. A clone (often a subclone) in the primary tumor that is also a lone founding clone of the metastasis sample indicates monoclonal seeding from the former to the latter samples, while at least two clones present in both primary and metastasis indicates polyclonal seeding between them (eg. from primary to metastasis). Therefore, to address this, we need to: (i) determine the presence of a clone in a sample; and (ii) identify whether a clone is a subclone or a founding clone of a sample. Using the bootstrap approach, we can answer both questions probabilistically. First, we define that a clone \( Y \) is present in a sample with a probability \( p_{pos}(Y) \) defined in Eq. (S3b). Second, a clone is considered a subclone in a sample only when at least one of its ancestor clones is found to be present in the sample. The initial founding clones of a sample are determined as the first clones to have positive CCFs in a breadth-first traversal of the clonal evolution tree. Subsequently, any clones that are present in multiple samples are also considered founding clones involved in the clonal seeding between those samples. After founding clones and subclones are identified, the models of seeding can be inferred using the above logic.

Pruning clonal evolution trees and merging intra-tumor samples

Obeying clonal ordering constraints (ie., sum and cross rules) does not guarantee a unique model will be identified. This is often due to the presence of low prevalence clones private to a sample that can be freely placed as the children of clones with higher cluster cellular prevalence (without violating the sum rule). Moreover, the cross rule does not apply to private subclones. However, the evolution of the private subclones found in only a single sample do not often affect
inter-sample clonal relationships or clonal seeding between them. When placing the private subclones onto the clonal evolution tree changes the interpretation of presence/absence of the parental clone shared between samples, it may affect clonal seeding interpretation. As such, relying on the ability to interpret seeding clones, ClonEvol compares trees differing only in the placement of private subclones to determine if they exhibit the same seeding patterns, hence collapsing them to a single tree by pruning the private subclones. Moreover, by comparing clonal evolution trees between samples, ClonEvol is also able to identify redundancies within sets of samples and merge those that do not offer additional information. Lastly, as multi-region sequencing is becoming a common practice to study intra-tumor heterogeneity, ClonEvol offers an option to merge multi-region samples into one single sample that better represents the whole tumor than a single biopsy. Therefore, pruning clonal evolution tree and merging samples can vastly simplify the analysis of cases with a large number of samples.

ClonEvol implementation

Heuristic approach to bootstrapping

Bootstrapping is a time-consuming task as it requires repeated samplings of the variants from the clusters to calculate the CCFs. Due to the high number of potential clonal orderings that need to be evaluated, it is critical to reduce the time involved in bootstrapping to ensure this approach remains practical. To achieve this, ClonEvol employs a heuristic approach that only samples the variants once to precalculate the mean of the cluster CCFs (each sample bootstrapping is presented as a matrix of M x N, with M = number of bootstraps, N = number of clusters). In each subsequent sum rule evaluation of a clonal ordering, the precalculated means of the cluster CCFs are used.

Weighted bootstrapping

The cellular prevalence estimates can be more accurate when tumors are sequenced to a greater depth. Therefore, ClonEvol also supports weighted models in which preference is given to variants with greater depth when they are resampled. A simple weighted model is used in which the weight is calculated as the ratio of the depth of the variant and the total depth of all variants within a cluster. The weight of each variant is used as the probability of selecting that variant when resampling (non-parametric bootstrap) or used in fitting the distributions (parametric bootstrap).

Non-parametric and parametric bootstrapping

ClonEvol provides both non-parametric and parametric approaches for resampling. The non-parametric bootstrap samples the variants with replacement directly from a cluster, and uses
their VAFs to estimate the CI of CCF. In contrast, the parametric approach samples variants from a distribution that is first fit using the read counts or VAFs of the variants within the cluster. ClonEvol supports several distributions that have been previously used to model read counts and VAFs in the literature including the binomial, beta, beta-binomial, and normal distributions [1,2]. The parameters for the binomial and normal distributions are estimated using the maximum likelihood approach [4]. Beta and beta-binomial distributions are fit using the method of moments [4]. In the weighted parametric models, the weighted sample mean and standard deviation are used in the estimators. Parameter estimations and random sample generation were performed using built-in R functions [5].

Tree enumeration algorithm

ClonEvol uses a recursive algorithm to enumerate the tree space given the clones. Each tree is grown from the root representing the founding clone. While growing a tree, the clonal ordering (involving a parental clone and its direct subclones) is evaluated and if the sum rule is probabilistically violated (as defined above, having high \( p_{neg} \)), the tree is terminated. Individual samples’ clonal evolution trees are grown separately, which are subsequently compared and superimposed across samples to derive the consensus trees (see Pseudocode 1).
Pseudocode 1. ClonEvol tree enumeration

# Grow and evaluate all clonal evolution trees for a sample rooted at clone 1
# clones are contiguous integers starting from 1
# This algorithm finds the parent for every clone among
# other clones excluding the ancestors of the clone (to prevent looping,
# hence, if every node has a parent assigned, the result will be a tree)
# ancestors(i, tree): the ancestor clones of clone i in tree
# pneg(i, tree, sample): probability that CCF(clone i) < 0 in sample, given tree
# updateCCF(i, tree, sample): calculate CCF(clone i) in sample, given tree

Function EnumerateTrees(sample, maxPneg):
    # sample's tree involves only clones whose cluster is present in sample (pClones)
    pClones = {c ∈ clones | CCF(cluster c | sample) > 0}
    forest = NULL

    # Grow and evaluate all possible trees stemming from tree
    # tree should already include clones 1,2,...,i-1
    # tree may be incomplete and disconnected because not all clones are included yet

    Function Grow(tree, i):
        If i > length(pClones):
            # all clones are in tree
            forest = {forest,  tree}
        Else:
            If i == 1:
                nextTree = {NULL -> 1} # root clone 1 is the first node of nextTree
                # calculate CCF(clone 1) using sum rule given nextTree
                nextTree = updateCCF(1, nextTree, sample)
                Grow(nextTree, 2)
            Else:
                # Allow each of other clones, excluding i and ancestors of i,
                # to be the parent of clone i, and evaluate sum rule for the parent
                For j in (pClones - {i, ancestors(i, tree)}):
                    nextTree = {tree, j -> i} # add branch from j to i to tree
                    # continue growing tree only if sum rule is not probabilistically violated
                    If pneg(j, nextTree, sample) <= maxPneg: # sum rule not violated
                        nextTree = updateCCF(j, nextTree, sample)
                        nextTree = updateCCF(1, nextTree, sample)
                        Grow(nextTree, i + 1)

            Grow(emptyTree, 1)
        Return(forest)

    # Order clones for all samples, construct consensus trees
    Function InferClonalEvolutionTrees(samples, maxPneg):
        For sample In samples:
            trees(sample) = EnumerateTrees(sample, maxPneg)
        ConsensusTrees = MatchAndSuperimposeTreesAcrossSamples(trees(all samples))
        Return(ConsensusTrees)
Simulation experiments

Simulation was implemented in R [5]. The simulation code is available at https://github.com/hdng/clonevol/, under “simulations” directory.

Ground truth cancer cell fraction and clonal mixtures

To evaluate clonal ordering performance and clonal seeding of ClonEvol, we generated 24 datasets (Table S1), each represents 100 random trees using two predefined clonal mixtures (A and B) in which the CCF of the clones were predefined to mimic intra and inter tumor heterogeneity observed in patient data (i.e. each sample had several clones present, some are shared between samples and some are private to only one sample).

Random tree generation

There were seven clones mixed in four samples (one primary and three metastasis) in clonal mixture A (Table S2) and 10 clones mixed in six samples (two primary and four metastasis) in clonal mixture B (Table S3). For each ground truth clonal mixture, 100 trees were randomly generated by growing a tree starting with the root (root was always clone 1 assumed to have arisen from germline cell), and a random clone is chosen and added to the tree as a child of a randomly chosen clone among those that were already included in the tree, until all clones were included. Given a random tree, the ground truth CCF of the clonal marker variants corresponding to a clone were back-calculated from the predefined CCFs of the clones in the clonal mixture by aggregating the CCFs of all descendent clones and itself using the sum rule in Eq. (1), up the tree. VAFs were then calculated by dividing the CCFs of the marker variants by two, and used to generate variant read counts.

Read count simulation

Once ground truth VAFs were identified, variant and reference read counts were generated following a binomial distribution with the assumption that the reads were randomly drawn from 100,000 reads representing the ground truth ratio of variant and reference fragments (ie. the number of variant reads were VAF*100000 and the number of reference reads were (1-VAF)*100,000). The number of reads to draw (sequencing depth) followed the Poisson distribution with mean depth = 100x. Our approach is similar to a previous approach [6], with the addition of noise described below.

Incorporation of noise

We incorporated several noise sources into the simulation including 1% sequencing error rate and up to 25% rate of small (undetectable) copy number variation. The sequencing error allows
1% of the reads (randomly chosen) to have the variant or reference base to be misread as one of the other three bases. To simulate undetectable copy number variations, we allowed 1%, 10%, or 25% of the variants to have a small gain or loss of the variant allele such that the site copy numbers randomly fell between 1 and 3, thereby deviating their VAFs from representing the ground-truth cellular prevalence (either over- or under-estimating the cellular prevalence).

Non-small and small cluster sizes and total number of variants
To evaluate performance of the bootstrap approach in various cluster sizes, we simulated both non-small and small cluster sizes (Table S1, S2, S3). In the non-small cluster size setting, the total variants were 400 and 1000 for datasets A and B, with the number of variants per cluster ranging between 30-100 (average ~57) and 40-300 (average ~100). In the small cluster setting, the number of variants per cluster ranged between 12-40 (average ~23) and 10-60 (average ~20) for datasets A and B, respectively.

Clonal seeding evaluation
To evaluate the performance of ClonEvol in clonal seeding prediction, we constrained the parental-child relationship to the clones such that the seeding clones for the metastasis samples were predefined. In datasets A, the parental-child constraints defined that metastasis M1 was seeded by clone 2 (monoclonal), metastasis M2 was seeded by clone 3 (monoclonal), and metastasis M3 was seeded by clones 2 and 3 (polyclonal); all from the primary tumor sample P (Table S2). In datasets B, the parental-child constraints defined that metastasis M1 was seeded by clone 1 (monoclonal), metastasis M2 was seeded by clone 2 (monoclonal), metastasis M3 was seeded by clones 1 and 3 (polyclonal), and metastasis M4 was seeded by clones 1 and 2; all from the primary tumor samples P1 or P2 (Table S2).

Running ClonEvol, LICHeE, Canopy, and PhyloWGS
ClonEvol was run using the unweighted non-parametric approach with 1000 bootstraps and \( p_{neg} = 0.95, \ p_{pos} = 0.95 \). LICHeE, Canopy and PhyloWGS were all run using the default or recommended parameters in the user manuals. In particular, LICHeE was run using parameters “-n 0 -s 100 -minClusterSize 5 -e 0.1”, Canopy was run with parameters “max.simrun = 10000, min.simrun = 1000, writeskip = 200”, and PhyloWGS was run with parameters “-B 1000 -s 2500 -i 5000”. We also provided Canopy with the number of optimal clones that was equal the number of clusters. Because PhyloWGS did not take preclustered variants as input, we created a meta variant for each cluster by aggregating the read counts from all variants within the cluster and provide the meta variants to PhyloWGS.

To compare the overall performance of the methods, we evaluated only trees that had all clones defined in the ground truth positioned (complete trees). LICHeE often failed to construct
the complete trees and attempted to guess the most erroneous clusters and remove them from the tree construction until a tree was found. This vastly altered the structure of the trees, especially when internal nodes were removed. We discarded these trees in our comparison.

Lastly, Canopy and PhyloWGS often reported trees with multiple clusters merged together in one node. In contrast to LICHeE’s incomplete trees, we considered those trees complete trees as all clones were positioned on the trees. However, when multiple clusters were merged, it indicated that Canopy and PhyloWGS were unable to distinguish them as distinct clones and therefore could not resolve their orders. To score trees involving merged clusters, we randomized the linear orders of the corresponding clones to generate separate trees whose each clone was represented by a distinct node, and calculated the average score of the those trees.

**Supplementary Discussion**

ClonEvol is designed to handle the uncertainty and errors that affect the cellular prevalence estimates from sequencing data. We have demonstrated that ClonEvol was effective in handling the errors. However, there were still cases where ClonEvol failed to infer the consensus clonal evolution trees. This could be due to extreme errors present in the data. Several sources of such errors include incorrect clustering, low sample quality, low sequencing depth, small number of variants, and various other errors in data analysis. Under such extreme conditions, ClonEvol does not output the consensus tree but still constructs individual sample trees. Manual investigation of individual samples’ trees could help identify the potential sources of error and reconstruct partial consensus trees.

We have also demonstrated that ClonEvol performance was robust to various sources of statistical uncertainty including random error, sequencing error and small random copy number variation. However, ClonEvol seeding prediction appeared to decrease when the number of variants within a cluster decreased. This is likely due to the decreased performance of the bootstrap approach on small sample sizes. A shared variance strategy in the parametric bootstrap approach can be employed in which the variance estimated from larger clusters can be used to guide estimation of the variance of small clusters. On the other hand, small clusters are often considered unreliable in clonal evolution reconstruction unless deep coverage can be obtained. Sometimes a small cluster could be formed from the outliers of a large cluster and thus is not a valid cluster. Therefore, we recommend users to investigate such clusters carefully and remove or recluster the variants if needed prior to running ClonEvol.

Our simulation also showed that ClonEvol predicted a higher number of trees than LICHeE and PhyloWGS. Multiple probable trees are indeed an intrinsic issue with clonal ordering that many tools face primarily due to the flexibility in ordering low frequency subclones. For example, it may not be possible to distinguish between a model in which a
low-frequency subclone is descended directly from a parental clone and a second model in which it is a subclone arising from a subclone and hence a grandchild of the parental clone. ClonEvol has a scoring scheme to select the tree with highest probability (as described in the methods), similar to the other tools. However, the ClonEvol approach is to output all likely models (in contrast to outputting only a certain number of top models) and to encourage users to further refine them. Additionally, ClonEvol prunes trees such that the clonal seeding interpretation is not changed from the original, unpruned tree and produces one tree per clonal seeding model. As we have shown in the result, clonal seeding was accurately reconstructed although the trees had mismatch with the correct tree. In multi-sample clonal evolution studies, identifying the seeding clones and patterns between samples is probably the most important question. Additional data such as more samples or single cell sequencing data can be used and are sometimes required to unambiguously resolve the correct, unique model as done previously, eg. AML31 case [7].

The ClonEvol tree enumeration algorithm is recursive and attempts to evaluate all possible trees. This could be a time-consuming task. Based on our experience with multi-region, multi-organ sequencing studies in hematological and solid cancers, ClonEvol often produces results within minutes. We have analyzed as many as ~10,000 variants, 18 samples, and 20 clones, and ClonEvol finished in less than an hour. Our reanalysis of the breast cancer case with 17 samples and 8 clones only took two minutes due to smaller number of clones. We have implemented a heuristic approach for bootstrap resampling that takes linear time in the number of variants. Hence, the major factor that affects the running time is the number of trees that must be evaluated, which is dependent on the number of clones and the number of samples. Several tricks can be employed to further reduce ClonEvol running time. First, the clonal orderings that violated the sum rule (and all of its family orderings, ie. orderings with additional children) can be recorded and thus excluded when a new tree is growth. This can be done within individual sample trees or between samples’ trees. When applied between samples, this is equivalent to simultaneously evaluating the sum rule and cross rule. Second, samples can be prioritized by complexity levels such that the samples with fewer clones will be analyzed first to establish a smaller set of clonal orderings that has to be matched by subsequent samples. Lastly, sum rule evaluation can be bypassed for clonal orderings that obviously violate the sum rule (eg. by a hard and large threshold cutoff using the mean of the $CCFs$).

The increasing use of deep sequencing allows for the detection of low frequency subclones thereby presenting a new challenge. Often many clonal lineages are consistent with the data due to the flexibility in the ordering of low frequency subclones. These are further confounded by the growing quantity of samples used to study intra- and inter-tumor heterogeneity. ClonEvol alleviates the necessity of considering a large number of models by using a reduced representation that is shared across models and that consistently captures important biological findings. In particular, upon determining the seeding/founding clones of the samples, ClonEvol can prune subclones that are private to a single sample without changing the
founding clones or, hence, the seeding models. As demonstrated in our patient data analysis, this collapses otherwise distinct models to a single, consistent model to facilitate biological interpretation.

References