Not ready for production

CONIPHER: a computational framework for scalable phylogenetic reconstruction with error correction

Kristiana Grigoriadis[1,2,3,#], Ariana Huebner[1,2,3,#], Abigail Bunkum[1,4,5,#], Emma Colliver[2,#], Alexander M. Frankell[1,2,#], Mark S. Hill[2], Kerstin Thol[1,3], Nicolai J. Birkbak[1,2,6,7,8], Charles Swanton[1,2,9,*], Simone Zaccaria[1,5,*], Nicholas McGranahan[1,3,*]

1. Cancer Research UK Lung Cancer Centre of Excellence, University College London Cancer Institute, London, UK
2. Cancer Evolution and Genome Instability Laboratory, The Francis Crick Institute, London, UK
3. Cancer Genome Evolution Research Group, Cancer Research UK Lung Cancer Centre of Excellence, University College London Cancer Institute, London, UK
4. Cancer Metastasis Lab, University College London Cancer Institute, London, UK
5. Computational Cancer Genomics Research Group, University College London Cancer Institute, London, UK
6. Department of Molecular Medicine, Aarhus University Hospital, Aarhus, Denmark
7. Department of Clinical Medicine, Aarhus University, Aarhus, Denmark
8. Bioinformatics Research Centre, Aarhus University, Aarhus, Denmark
9. Department of Oncology, University College London Hospitals, London, UK

# These authors contributed equally: Kristiana Grigoriadis, Ariana Huebner, Abigail Bunkum, Emma Colliver, Alexander M. Frankell

* These authors jointly supervised this work

Correspondence to: Charles Swanton, Simone Zaccaria, Nicholas McGranahan

EDITORIAL SUMMARY: CONIPHER: a computational framework for accurately inferring subclonal structure and the phylogenetic tree from multi-sample tumour sequencing, accounting for both copy number alterations and mutation errors.

PROPOSED TWEET: CONIPHER: a computational framework for scalable phylogenetic reconstruction with error correction from tumours

PROPOSED TEASER: CONIPHER reconstructs tumour evolutionary history

KEY POINTS:

- CONIPHER is a computational framework for accurately inferring subclonal structure and phylogenetic relationships from multi-sample tumour sequencing, accounting for both copy number alterations and mutation errors.

- Benchmarking analyses on simulations show that CONIPHER outperforms similar methods, and in particular scales to a large number of tumour samples and clones. This enables automated phylogenetic analysis which can be effectively applied to large sequencing datasets generated with different technologies.

Abstract

Intra-tumour heterogeneity provides the fuel for the evolution and selection of subclonal tumour cell populations. However, accurate inference of tumour subclonal architecture and reconstruction of tumour evolutionary history from bulk DNA sequencing data remains challenging. Frequently, sequencing and alignment artefacts are not fully filtered out from real cancer somatic mutations and errors in the identification of copy number alterations or complex evolutionary events (e.g. mutation losses) affect the estimated cellular prevalence of mutations. Together, such errors propagate into the analysis of mutation clustering and phylogenetic reconstruction. In this paper we present a new computational framework, CONIPHER (COrrecting Noise In PHylogenetic Evaluation and Reconstruction), that accurately infers subclonal structure and phylogenetic relationships from multi-sample tumour sequencing, accounting for both copy number alterations and mutation errors. CONIPHER has been used to reconstruct subclonal architecture and tumour phylogeny from 421 multi-sample tumours with high-depth whole-exome sequencing (WES) from the TRACERx421 dataset, as well as 126 primary-metastatic cases. CONIPHER outperforms similar methods on simulated datasets, and in particular scales to a large number of tumour samples and clones, while completing in under 1.5 hours on average. As such, CONIPHER enables automated phylogenetic analysis which can be effectively applied to large sequencing datasets generated with different technologies. CONIPHER can be run with basic knowledge of bioinformatics, and R and bash scripting languages.

Key papers using this protocol

- Frankell *et al.* The evolution of lung cancer and impact of subclonal selection in TRACERx (doi.org/10.1038/s41586-023-05783-5)
- Al Bakir *et al.* The evolution of non-small cell lung cancer metastases in TRACERx (doi.org/10.1038/s41586-023-05729-x)
- Martinez-Ruiz *et al.* Genomic–transcriptomic evolution in lung cancer and metastasis (doi.org/10.1038/s41586-023-05706-4)
- Abbosh *et al.* Tracking early lung cancer metastatic dissemination in TRACERx using ctDNA (doi.org/10.1038/s41586-023-05776-4)
- Karasaki *et al.* Evolutionary characterization of lung adenocarcinoma morphology in TRACERx (doi.org/10.1038/s41591-023-02230-w)

Introduction

Cancer is an evolutionary process[1], in which the heritable accumulation of somatic mutations results in the formation of heterogeneous subpopulations of cancer cells, referred to as intra-tumour heterogeneity (ITH)[2]. Most cancer evolution studies quantify ITH from DNA sequencing data by identifying the unique complements of somatic mutations that are carried by these different subpopulations of cells, or 'subclones'. Accurately reconstructing the genomic profile of each subclone, and inferring the evolutionary hierarchy between the subclones present in a tumour is important, not only for studying the biology of the disease trajectory, but because a tumour subclone harbouring a treatment-resistant genomic variant could have important clinical implications, and could be used to guide therapeutic decision making[3].

In recent years, progress in next-generation sequencing technology and computational methodology has revealed significant ITH in several cancer types[4]. However, a single tumour tissue biopsy sample may contain a mixture of many thousands of heterogeneous normal and cancer cells, making the full deconvolution of subclonal populations and their phylogenetic ordering from bulk DNA sequencing challenging. While single-cell sequencing techniques are promising approaches providing unprecedented resolution to cancer evolutionary analysis, they remain highly specialised techniques with various technical and financial challenges that limit their application to large cohorts of tumour samples, particularly in clinical settings. In fact, the most recent and large cancer sequencing studies such as the TRACERx[5] and PCAWG[6] studies still rely on bulk sequencing. Therefore, accurate and automatic pipelines for tumour evolutionary analysis from bulk sequencing data, especially multi-region and multi-site datasets, still represent an important unmet need.

Typically, subclonal reconstruction algorithms leverage the observed variant allele frequency (VAF) of single-nucleotide mutations measured from aligned DNA sequencing reads in order to quantify the prevalence of somatic events[7]. Due to the presence of somatic copy number alterations (SCNAs) and normal cell admixtures, the VAF alone is not an accurate estimator of the population frequency of the variant. Therefore, most existing algorithms apply different approaches to correct the VAF for tumour purity and SCNAs to infer estimates of the cancer cell fraction (CCF) of a mutation, which defines the proportion of cancer cells in the sample that carry the mutation[8].

To reconstruct clonal evolution, computational methods cluster together mutations with similar CCFs in all samples sequenced into ‘subclonal clusters’, under the assumption that they are likely present in a similar set of cells and that they represent a clonal expansion at a similar evolutionary time point[8]. Then, by nesting subclonal cluster CCFs based on evolutionary principles for constraining lineage relationships, algorithms seek to infer the evolutionary ordering of clusters and reconstruct the full tumour phylogenetic tree[2] (Table 1). Such principles include the ‘sum condition’[9], (sometimes referred to as the ‘pigeonhole principle’)[10] which states that the CCF of a parental cluster must be greater than or equal to the sum of its daughter cluster CCFs in all tumour samples, and ‘crossing rule’, which states that for two subclonal clusters A and B, if CCF(A) > CCF(B) in at least one tumour sample, and CCF(A) < CCF(B) in one or more distinct tumour samples, then A and B must be on distinct branches of the phylogenetic tree[11,1211].

Three key challenges make the accurate estimation of mutation CCFs from bulk sequencing data, assigning mutations to clusters, and inferring evolutionary ordering between mutation clusters non-trivial.

First, errors in both mutation and copy number calling (e.g., sequencing artefacts, misalignments, etc) may result in errors in the estimated CCFs and, hence, in the identification of false mutation clusters that do not reflect true biological signals. For example, subclonal SCNAs undetected by copy-number calling algorithms can result in a genomically clustered group of mutations having a distinct CCF which reflects the copy number event and not the true underlying prevalence of the mutations. Unless explicitly removed, such clusters will be propagated and will impact the phylogenetic tree reconstruction. However, most of the existing algorithms that cluster mutations and reconstruct tumour phylogenetic trees assume that the input data is error free[13], either in terms of SNVs[9], SCNAs[14], or both[15]. Thus, a cluster resulting from mutation or SCNA errors will be given equal weight to a bona-fide mutation cluster which might erroneously impact the reconstruction of the tumour phylogenetic tree.

Second, SCNAs can result in the loss of mutations when genomic segments that contain the locus of their mutated alleles are deleted[8]. Mutation losses violate the commonly enforced infinite sites assumption (i.e., the assumption in which mutations occur at most once at a particular genomic locus and cannot be lost by reversion mutation[13]). When analysing these lost mutations, their CCFs will appear lower than the CCFs of the other mutations that represent the same clonal expansion (i.e. that are part of the same edge of the tumour phylogenetic tree). Hence, accounting for mutation losses is important for inferring the correct mutation cell fraction. In this paper, we refer to the fraction of cancer cells that either carry a mutation, or whose ancestors carried the mutation before mutation loss, as the phylogenetic cancer cell fraction (PhyloCCF)[8]. This concept has been introduced and used in previous studies[2].

Finally, most current subclonal reconstruction methods are limited in their ability to accurately cluster and construct phylogenetic trees based on large multi-sample studies. In particular, to account for SCNAs during the estimation of CCFs from the observed VAFs, some phylogenetic reconstruction algorithms aim to jointly model the evolution of SNVs and SCNAs[10].

However, due to the complexity of these models, these algorithms do not scale to the high numbers of mutations found in the whole-genome and whole-exome sequencing studies[16], and neither to the large number of tumour samples sequenced in recent multi-sample tumour studies[2].

To address previous limitations, we develop CONIPHER (COrrecting Noise In PHylogenetic Evaluation and Reconstruction), a novel algorithm to automatically reconstruct subclonal mutation clusters, tumour phylogeny and subclone cell proportions from bulk sequencing data and account for uncertainty. CONIPHER is characterised by three novel features that address key challenges in phylogenetic reconstruction described above: (1) an approach to remove biologically improbable clusters that either are driven by likely-erroneous mutations or by subclonal SCNAs, (2) a method to correct for complex evolutionary events, including mutation losses[8], and (3) an efficient extension of previous and new approaches that allows CONIPHER to scale to a high number of primary tumour samples per patient.

Despite the rich literature on tumour phylogeny reconstruction[14], how features of the inferred tumour phylogenies relate to the biology of tumour growth, in terms of selection, mutation rates and rates of chromosomal instability, remains unclear. This protocol enables a user-friendly, straightforward computational framework for analysis of tumour phylogenies in R, including calculation of subclone proportions in each tumour sample. CONIPHER has been used to automatically reconstruct the tumour phylogenetic trees for 421 patients with non-small cell lung cancers (NSCLC) with primary and metastatic disease in the recent TRACERx421 study[5,17,18].

Development of the protocol

Automated tumour phylogenetic reconstruction from bulk DNA sequencing of tumours with a large number of mutations enables an in depth analysis of tumour evolution. To accurately reconstruct the tumour phylogenetic tree we posit that it is imperative to account for mutation losses and erroneously clustered mutations. Correct tree reconstruction will affect interpretation of downstream analyses of evolutionary relationships between specific driver mutations, and inference of metastatic seeding and dissemination patterns. Hence, we created CONIPHER to process and construct tumour phylogenetic trees for 432 tumours from 421 patients with NSCLC from the TRACERx lung cohort[5]. We will now first outline the CONIPHER method before showing that CONIPHER outperforms previous algorithms on simulations.

Overview of the CONIPHER method

CONIPHER takes as input processed mutation data from bulk DNA sequencing (for example using Mutect2[19] and Varscan[20]), as well as SCNAs, purity and ploidy, which can be computed by existing and well established methods, such as ASCAT[21], HATCHet[22], Sequenza[23], and Battenberg[24]. We report recommended mutation preprocessing steps in Supplementary Methods 1. CONIPHER subsequently performs mutation clustering, followed by tumour phylogeny reconstruction, and finally computes subclone proportions (Figure 1). Below, we describe an overview of the method. We provide a more detailed explanation of the method in Supplementary Methods 2, including statistical tests performed and exact values of the parameters and thresholds.

Subclonal mutation clustering. The first stage in CONIPHER is the estimation of PhyloCCFs and clustering of somatic mutations (Figure 1a-d). This stage can be broken down into four main components, which were designed with attention given to minimise the error introduced at each subsequent stage. First, copy number preprocessing of every mutation is performed (Figure 1a), in which the PhyloCCF of every mutation is calculated, by transforming the measured VAF by expected mutation copy number and tumour purity to compute the CCF metric[25], and taking into account both clonal and subclonal SCNAs[2]. Secondly, a pre-clustering stage is implemented to split mutations in distinct groups, such that each group only contains mutations that are clearly present or clearly absent in the same set of tumour samples (Figure 1b, Supplementary Methods

2 – Section 2.1)[5]. Similar to recent methods[26], this step prevents the mixing of these mutations in the same cluster, an error that has been observed for most existing mutation clustering algorithms[26]. In addition, insertion/deletion mutation (indel) VAFs are corrected, if indel calls are (optionally) provided as input (as detailed in the Procedure, Supplementary Methods 2 – Section 2.1). Thirdly, CONIPHER applies Dirichlet clustering using the PyClone algorithm (v.0.13.1[7]) to each group of mutations separately to identify the candidate mutation clusters (Figure 1c). Finally, post-processing and quality control is performed on the inferred mutation clusters (Figure 1d). First, clusters that appear to be driven by copy number loss are removed[2]. Subsequently, mutation clusters are removed that comprise a small number of mutations (user-defined) and pairs of subclonal clusters are merged if their difference is driven solely by a subclonal copy number correction (Figure 1d, Supplementary Methods 2 – Section 2.1).

Phylogenetic tree building. The second and main stage of CONIPHER is reconstruction of the tumour phylogenetic tree. This stage takes the output from the previously performed mutation clustering as input, namely, inferred assignments of mutations to mutation clusters, and mutation PhyloCCF estimates. Notably, this stage is compatible with mutation clustering performed from other methods. The phylogenetic tree building stage can be broken down into four main components: cluster nesting, growing the tree, enumerating the solution space of alternative phylogenies, and computing subclone proportions.

Mutation cluster nesting. First, 95% confidence intervals are computed to obtain estimates for average PhyloCCF values for each mutation cluster identified in the clustering stage, in each tumour sample (Figure 1e, Supplementary Methods 2 – Section 2.2). Secondly, two one-sided tests are performed comparing PhyloCCF values between every possible pair of clusters in each tumour sample, in order to determine whether one cluster could potentially be nested within the other (Figure 1f). The truncal cluster is assigned as the cluster that can nest all other clusters (Figure 1f). A test is additionally performed to check whether each cluster could be classified as subclonal within any given tumour sample, or whether it is indistinguishable from the truncal cluster (Supplementary Methods 2 – Section 2.2)[5].

In order to prevent artefactual mutation clusters from being assigned to a branch of the phylogenetic tree, the genomic positions of mutations within each cluster are inspected. If all mutations in a cluster are less evenly distributed across chromosomes than would be expected based on the distribution of mutations across chromosomes in the truncal cluster, the cluster is deemed as potentially copy number driven and therefore removed from subsequent analysis. Notably, in the TRACERx primary NSCLC cohort[5], we verify that the mutations removed with clusters that do not fit the phylogenetic tree are consistent with truncal mutations subject to copy number loss, as evidenced by mutational signature distributions (Supplementary Methods 3, Supplementary Figure 1). Cluster nesting is summarised as a nesting matrix and can be represented as an ancestral graph (Figure 1f).

Growing the phylogenetic tree. Then, the ancestral graph is pruned to attempt to produce a tree structure with no cycles (Figure 1g). This method favours a more linear tree topology structure, as opposed to a more branched structure. Subsequently, clusters are removed from the tree that are the cause of the following issues: (i) cycles in the tree, or (ii) CCFs of tree branches at each tree level exceeding a user-defined threshold (by default a CCF buffer of 10% is used, Supplementary Methods 2 – Section 2.3). Clusters are removed such that the fewest mutations possible are removed from the phylogenetic tree. This stage returns one ‘default’ tumour phylogenetic tree.

Growing the forest. After identifying the default tree, our algorithm enumerates all possible alternative phylogenies that fit the identified cluster nesting structure of the pruned ancestral graph (Figure 1h). First, all combinations of clusters are identified that could be moved to descend from a different parental node, without causing graph cycles (i) or tree-level issues (ii) as described above (Supplementary Methods 2 – Section 2.4). All possible phylogenetic trees are provided as output.

After all potential trees are identified, tree branches, or edges, that are common to all trees are classified as “consensus” branches, conversely, branches that are found in only a subset of trees are classified as “non-consensus” branches.

CONIPHER additionally provides two methods for summarising the solution space of multiple phylogenetic trees per tumour (Figure 1i). First, CONIPHER computes the tree(s) that generates the lowest amount of nesting error, which we term the sum condition error (SCE). Secondly, CONIPHER computes the tree(s) comprising branches, or tree edges, most commonly shared amongst alternative trees in the solution space, by computing the edge probability. We describe the calculation of the SCE and edge probability metrics in Supplementary Methods 2 – Section 2.4.

Computing subclone proportions. Finally, CONIPHER automatically computes the proportion of cells in each tumour sample belonging to each genomically homogeneous subclone, or the “subclone proportions'', based on the inferred default tree and tumour phylogeny with lowest SCE (Figure 1j, Supplementary Methods 2 – Section 2.5). Notably, subclone proportions will sum to 1 in each tumour sample and will only correspond to the mutation cluster PhyloCCF in the case of terminal nodes on the phylogenetic tree. This enables an analysis of recent subclonal expansions in a tumour, which was found to be prognostic in our companion manuscript[5].

Benchmarking and evaluating the performance of CONIPHER

A realistic simulation framework for tumour evolution. We benchmarked the performance of CONIPHER using a simulation framework introduced within the TRACERx421 study, that comprises generated tumour phylogenies, mutation clusters, and related bulk sequencing data[5]. Ground truth simulations were designed to model the evolution of genetic variants frequently observed in NSCLC, including somatic SNVs, truncal/subclonal SCNAs, and truncal/subclonal whole genome doubling (WGD) events. In particular, the simulation framework models the effect of such SCNAs and WGD events on the overlapping SNVs, thus resulting in SNV mutation losses or changes in SNV multiplicity (i.e., number of copies harbouring the SNV). A cohort of 150 simulated tumours was used to benchmark CONIPHER (Simulated Dataset 1, Supplementary Methods 4): 50 simulated tumours with 2-3 samples per tumour (low category), 50 simulated tumours with 4-7 samples per tumour (medium category), and 50 simulated tumours with >7 samples per tumour (high category), totalling a collection of 150 simulated tumours. Full mathematical details of the simulation framework are reported in Supplementary Methods 4 – Section 4.1 and our companion paper[5]. In Simulated Dataset 1 an erroneous cluster not fitting the ground truth tree topology was introduced. An analogous dataset comprising 150 simulated tumours with no erroneous cluster was also generated (Simulated Dataset 2, Supplementary Methods 4 - Section 4.2). Finally, a third simulated dataset comprising 36 simulated tumours was generated in the same way as Simulated Dataset 1, with varying sequencing coverage (Simulated Dataset 3, Supplementary Methods 4 - Section 4.2).

Commented [1]: As you refer to three datasets in Supplementary Method 4, I just want to confirm that I have understood correctly. Do these 150 datasets correspond to what you call dataset 1 in Supplementary Method 4? ie is the benchmarking data presented in figure 2 and the discussion below is based entirely on dataset 1.

A further small point for clarification: the results you describe in Supplementary Method 4 seem to be based on dataset 1. Section 4.4 describes results from dataset 3, but I can only find mention of dataset 2 in the legend to Supplementary Figure 3, panel b. However, I can't find a citation to this panel in the text.

Commented [2]: Thanks. Yes, that is correct - the 150 datasets here are dataset 1. We have now updated the terminology to refer to each simulated tumour case as a 'simulated tumour' instead of 'simulated dataset', and the three simulated cohorts are referred to as 'Simulated Dataset 1', etc. I hope this makes it clearer.

Supplementary Figure 3b is cited further below, in the subsection titled 'Multiple alternative tree solutions'. We've additionally added a sentence in Supplementary Methods - Section 4.4 to cite the Supplementary Figures 3a and 3b.

Comparison of CONIPHER with current state-of-the-art methods. Based on the ground truth simulations generated using the simulation framework[5], we compared CONIPHER for reconstructing tumour subclonal mutation clusters and inferring tumour phylogeny with five current state-of-the-art approaches (Figure 2). Specifically, we compared our clustering method with PyClone, as well as our clustering and phylogenetic tree building method with PhyloWGS[10], LICHeE[15], CITUP[13] and Pairtree[27], with each of these methods only comprising a subset of the features introduced in CONIPHER (Table 1). Overall, CONIPHER is able to identify mutation clusters (Figure 2a) and reconstruct tumour phylogenies (Figure 2b) with higher accuracy than other methods (Supplementary Methods 4 – Section 4.2), and obtains consistently high performance with sequencing coverage >50x (validated on Simulated Dataset 3; Supplementary Methods 4 – Section 4.3, Supplementary Figure 2). When providing the true mutation clusters, we additionally demonstrated a similar improvement in performance when benchmarking only the tree building method of CONIPHER with previous methods that provide the same feature, i.e., LICHeE, CITUP, and Pairtree (Figure 2).

Commented [3]: Given that you do not name datasets 1 or 2 above, I think it could be confusing to refer to dataset 3.

Commented [4]: We have updated this

Table 1. Comparison table of phylogenetic tree inference methods.

Method	**Date**	**Mutation presence / absence assignment**	**Mutation loss inference**	**Subclonal SCNA correction**	**SNV error correction**	**Scalable**	**Multiple solutions**	**Clone proportions calculation**
CITUP[13]	2015	N	N	N	N	Y	Y	N
PhyloWGS[10]	2015	N	Y	Y	N	N	Y	N
LICHeE[15]	2015	Y	N	N	Y	Y	Y	N
PASTRI[9]	2017	N	N	N	N	Y	Y	N
Pairtree[27]	2022	N	N	Y	N	Y	Y	N
CONIPHER	2023	Y	Y	Y	Y	Y	Y	Y

Table comparing functionalities offered by various state-of-the-art mutation clustering and phylogenetic reconstruction algorithms.

Scalability of method. We compared the scalability of CONIPHER with the other methods by using the simulated tumours with increasing numbers of samples and tumour clones (low, medium, and high categories, Simulated Dataset 1; see above). We found that CONIPHER and Pairtree were able to infer tumour phylogeny for every simulated tumour, whereas the other methods failed to run or complete the reconstruction for 12-98% of simulated tumours within the time frame allowed (8 hours). In particular, PhyloWGS was unable to complete tumour phylogenetic reconstruction on any of the simulated tumours in the medium or high category and only able to reconstruct 3/50 trees in the low category (Figure 2c).

Presence-absence informed clustering. Distinguishing whether mutations are absent or present in certain samples is an important feature in certain applications, for example when assessing the presence of mutations in primary tumour samples compared to metastases[17]. As such, we explicitly compared the performance of CONIPHER's mutation clustering to other methods, to evaluate how differences in the clustering would affect the downstream phylogenetic tree analysis (Figure 2d). We found that CONIPHER and LICHeE had the highest mutation presence precision in every tumour sample. In particular, the presence-absence classification stage in CONIPHER led to improved mutation presence precision in the high category, compared to the other methods for which performance decreased with larger simulations. We note that these results suggest the improved accuracy of CONIPHER to distinguish presence/absence of mutations in certain samples compared to the similar method previously introduced by LICHeE.

Measuring mutation losses. Losses of mutations due to CNAs have been observed to be frequent in cancer[5,8,17]. As CONIPHER is one of the few methods that takes these events into consideration and applies related corrections, we assessed the impact of this feature on the accuracy of mutation clustering by evaluating the sensitivity in the identification of truncal mutations. Truncal mutations that are affected by subclonal losses might impact downstream mutation clustering and related phylogenetic analysis if not taken into account[8]. Consistent with this expectation, we found that methods that do not account for mutation loss, such as CITUP and Pairtree, had a lower truncal sensitivity in all simulation categories (Figure 2e). We also observed that when running Pairtree with CONIPHER clustering, the truncal sensitivity was greatly improved, thereby indicating that Pairtree was not directly accounting for mutation losses (Figure 2e). Clustering performance may directly impact the truncal sensitivity independently of tree building, so we also evaluated the performance of each tree building method on the set of ground truth simulated clusters per simulation (Figure 2f & 2g). We found that CITUP failed in all 150/150 (100%) instances, which we hypothesise is due to the inability to account for mutation loss. Pairtree and LICHeE were able to identify the correct truncal mutation cluster in 83/150 (55%) and 84/150 (56%) of the simulated instances respectively, compared with CONIPHER that was best able to account for mutation loss and correctly identified the truncal mutation cluster in 141/150 (94%) of ground truth instances.

Accurate error removal. Bulk DNA sequencing data may contain a significant degree of error; however, most existing methods for phylogenetic reconstruction ignore the presence of errors and noisy mutations in the input data (Table 1). To mitigate the impact of errors, CONIPHER aims to identify mutation clusters driven by sequencing noise, and removes these. We evaluated the extent to which CONIPHER correctly identifies and removes mutational sequencing noise by injecting an artefactual cluster in the simulated tumours (Simulated Dataset 1), and comparing the number of simulations in which the artefact cluster is removed (Figure 2h). Notably, the artefact cluster is not necessarily incompatible with the tree structure (i.e. it was not necessarily biologically implausible). CONIPHER was able to accurately identify and remove error-driven mutations in 77/150 simulated tumours (51%), compared to LICHeE that removed noisy clusters in 3/150 simulated tumours (2%), and CITUP and Pairtree which did not identify the noisy clusters in any instances. For simulations with a low number of samples per tumour, CONIPHER also often failed to remove the erroneous cluster (38/50 simulated tumours). In these cases, many ‘error clusters’ still fit the tree, without the need to remove any mutations. By contrast, for simulations with a high number of samples, the erroneous cluster was correctly identified in 38/50 simulated tumours (76%).

Multiple alternative tree solutions. Most existing methods provide multiple solutions for the reconstruction of tumour phylogenies and rank these solutions according to their likelihood, or to some objective score. We thus used the ground truth simulations to assess whether the tree ranking of CONIPHER allows the identification of the true tree as a high-rank solution. To do this, we measured whether phylogenetic tree solutions with higher mutation descendant accuracy gave better performing sum condition error (SCE) and edge probability metric scores. We observed that for simulated tumour cases for which CONIPHER identified more than one potential tree structure, the alternative trees that were reconstructed with the highest mutation descendant accuracy had lower SCE scores compared to less accurate alternative phylogenetic trees (Supplementary Figure 3a, Supplementary Methods 4 – Section 4.4). Evaluating the performance of the CONIPHER tree building stage on the set of ground truth clusters from Simulated Dataset 2 (a simulated dataset with no mutation loss and no error-driver mutations, Supplementary Methods 4 – Section 4.2), we also observed that the inferred edges that were present in the ground truth tree were shared amongst a larger number of alternative tree solutions than edges not present in the ground truth tree (Supplementary Figure 3b). Finally, we observed that the highest ranking tree solutions based on the SCE and edge probability metrics had a higher descendant accuracy than alternative tree solutions (Supplementary Figure 3c, Supplementary Methods 4 – Section 4.4).

Realistic reconstruction of tumour evolutionary history. We assessed the impact of the different performance of the benchmarked methods, including CONIPHER, CITUP, LICHeE and Pairtree, by comparing their results when applied on the sequencing data previously generated for CRUK0063, a metastatic case from the TRACERx421 cohort from our companion study[17] (Supplementary Figure 4). We found CONIPHER produced the most realistic reconstruction of mutation clusters and tumour phylogeny, as supported by a reasonable assignment of the truncal cluster (compared with LICHeE and Pairtree) (Supplementary Figure 4a, b) and separating of mutations by presence and absence (compared with CITUP) (Supplementary Figure 4c, Supplementary Methods 5). Step-by-step reconstruction of the evolutionary history of CRUK0063 using CONIPHER is detailed in the Procedure.

Advantages and limitations of CONIPHER

CONIPHER performs mutation clustering and phylogenetic tree building from processed bulk DNA sequencing data. This can be from bulk whole genome sequencing (WGS), whole exome sequencing (WES) or a targeted sequencing approach. It is highly scalable and can reconstruct tumour phylogenies from tumours with many samples and many clusters in a time frame of the order of minutes. CONIPHER assigns mutations to the phylogenetic tree more accurately than other state-of-the-art methods and in particular improves the quality of the mutations assigned to the tree, by taking into account biological constraints in order to remove error-driven signal. CONIPHER for phylogenetic tree building is compatible with input from mutation clustering performed using other methods and automatically computes subclone proportions in each tumour sample.

However, CONIPHER does have limitations. CONIPHER does not currently support raw sequencing data as input and requires processed data from bulk DNA sequencing. In particular, we assume that mutation and copy number calling algorithms have been applied to the raw sequencing data.

Required expertise

CONIPHER is straightforward to implement from the command line, using basic knowledge of Linux/Unix syntax. CONIPHER output is in both human readable form (.tsv files) and additionally .RDS objects for use in the R programming language. Knowledge of scripting languages would be helpful for users who wish to use CONIPHER output for downstream analyses; however, non-experts in bioinformatics should be able to run CONIPHER using the command line only to obtain mutation clustering and tumour phylogenies with correct input data. The current implementation of CONIPHER is written in the R programming language.

Experimental design

The CONIPHER Procedure is composed of two main stages: a clustering stage (Steps 1 - 3) and a tree building stage (Step 4) (Figure 3). The clustering stage is optional, and can be replaced by a mutation clustering method of the user's choice. At each stage, output directories are generated containing both data and summary plots (Boxes 1 and 2). Both stages can be run from the command line. Alternatively, both stages can be run with a wrapper end-to-end; that is, the clustering stage automatically generates output that is taken as input to the tree building stage (see below). Both clustering and tree building stages can also be run in an interactive R session, either separately, or end-to-end (see the Github page, https://github.com/McGranahanLab/CONIPHER, for further details).

Preprocessing input data

Preprocessing of mutations. Somatic mutation calling and filtering should be carried out by the user, before input to CONIPHER. The details on the mutation preprocessing steps used in the TRACERx study can be found in Supplementary Methods 1 and our companion manuscript[5].

Preprocessing of `input.tsv`. Our protocol requires as input one file, `input.tsv`, that is a mutation table containing information about each point mutation in each tumour sample sequenced (Figure 4). This input table can be used as input for both clustering and tree building stages, with specific column names required for each stage. We provide a complete description of all columns required in `input.tsv` for CONIPHER clustering and tree building stages in the input table in Table 2.

The CONIPHER input table `input.tsv` is in long format, with a new row for each mutation, for each tumour sample sequenced (Figure 4). Mutation clustering takes as input the genomic position of every mutation in every tumour sample, the copy number at the genomic position of each mutation (`COPY_NUMBER_A`, `COPY_NUMBER_B`), and an estimate of the tumour purity (or aberrant cell fraction, `ACF`) and ploidy `(PLOIDY`) within each sample (Figure 4, pink box). Columns `COPY_NUMBER_A` and `COPY_NUMBER_B` can represent the major and minor copy number alleles, respectively, or alternatively, phased copy number values can be used. Tree building takes the same table as input, with additional columns required (green box, Figure 4): mutation cluster assignments (`CLUSTER`), estimates of the PhyloCCF (`CCF_PHYLO`) and observed CCF (`CCF_OBS`), and mutation copy number estimates for each mutation in each sample (`MUT_COPY`). These data and table columns are generated automatically by the clustering stage (Figure 3).

Optionally, an additional column (`MUT_TYPE`) can be included in `input.tsv` with a flag indicating the mutation type. Currently, there are two mutation types supported: SNV (`MUT_TYPE=="SNV"`) or an indel (`MUT_TYPE=="INDEL"`) (Table 2).

Preprocessing of `input_seg.tsv`. Optionally, a copy number segmentation file `input_seg.tsv` can be provided as input (see example CRUK0063[17] in PROCEDURE), which is used in the clustering stage to generate a copy number plot across the genome with overlaid mutation copy numbers. This table is in long format, with a new row for one copy number segment in one tumour sample. The first column `SAMPLE` describes the tumour sample identifier. Columns

`CHR`, `STARTPOS` and `ENDPOS` indicate the genomic segment. Columns `COPY_NUMBER_A` and `COPY_NUMBER_B` indicate the copy number of the major and minor alleles, respectively. These values can be integer copy number or raw fractional copy number.

Conventions

In our companion manuscripts[5,17], the naming convention is to refer to "tumour regions" when referring to multiple distinct bulk samples taken from one tumour. In this manuscript we instead refer to the more general term "tumour samples" (`SAMPLE`) when referring to any sample with available sequencing data to be processed through CONIPHER. Chromosome names can be either with or without 'chr' prefix (e.g. '1' or 'chr1'). Chromosomes X and Y are ignored in this procedure.

Execution of full pipeline

An example wrapper script to run both stages of the pipeline end-to-end (`wrapper_conipher.sh`) is available to download from the CONIPHER-wrapper GitHub page (https://github.com/McGranahanLab/CONIPHER-wrapper). This wrapper is designed to be run for one case in the analysis cohort. A description of how to run each CONIPHER step individually is detailed in the Procedure below, in which both the CONIPHER clustering and tree building wrappers are run on processed WES data from a patient with metastatic disease from the TRACERx421 cohort, case CRUK0063[17].

MATERIALS

EQUIPMENT

- Data files are required for each tumour in the analysis cohort (`input.tsv`, optionally `input_seg.tsv`) as described in section Input data of the Experimental Design.
- A standard computer system with a Linux or macOS operating system is required to run CONIPHER from the command line. CONIPHER can be run using access to Conda. Details can be found in Software Requirements.
- Programme source code is publicly available for our CONIPHER R package at https://github.com/McGranahanLab/CONIPHER, and for our CONIPHER clustering and tree building wrapper at https://github.com/McGranahanLab/CONIPHER-wrapper.

EQUIPMENT SETUP

Hardware requirements

Memory requirements depend on whether the input data is from whole exome or whole genome sequencing data. It is recommended to run the method using at least 8GB memory.

Software requirements

Access to a high performance computing (HPC) system is recommended for tumours with a large number of samples and mutations, but CONIPHER clustering and tree building can also be run on a local machine. CONIPHER clustering relies on the PyClone algorithm[7] and therefore needs to be run within a Conda environment (see instructions in section Installation). If only running CONIPHER tree building, the CONIPHER package can be installed directly in R (3.6.1 <= version < 4.2).

Installation

CONIPHER code repository can be downloaded from GitHub and installed using Bioconda. We have created an R package for CONIPHER clustering and tree building with full package installation and interactive run instructions at (https://github.com/McGranahanLab/CONIPHER). We have additionally created a Github repository with a CONIPHER wrapper to run both

CONIPHER clustering and tree building end-to-end from the command line (https://github.com/McGranahanLab/CONIPHER-wrapper/). Instructions for creating the Conda environment required to run clustering and tree building are detailed below and in the `README.md` file in the CONIPHER-wrapper Github repository.

- To install the CONIPHER Bioconda package, open the terminal and run the following command:

```
conda create -n conipher -c conda-forge -c bioconda pyclone conipher
```

- To install the CONIPHER-wrapper, navigate to a desired directory and run the following command:

```
git clone git@github.com:McGranahanLab/CONIPHER-wrapper.git
```

Table 2. Description of required and optional columns in input.tsv.

Column name	**Column description**	**Required for clustering input**	**Required for tree building input**
`CASE_ID`	Tumour case identifier	Required	Required
`SAMPLE`	Sample identifier	Required	Required
`CHR`	Chromosome identifier	Required	Required
`POS`	Genomic position of mutation in chromosome	Required	Required
`REF`	Reference allele nucleotide	Required	Required
`ALT`	Alternative nucleotide	Required	Required
`REF_COUNT`	No. reads of reference allele	Required	Required

`VAR_COUNT`	No. reads of variant allele	Required	Required
`DEPTH`	Sequencing depth	Required	Required
`COPY_NUMBER_A`	Copy number at mutation position, allele A	Required	Required
`COPY_NUMBER_B`	Copy number at mutation position, allele B	Required	Required
`ACF`	Aberrant cell fraction/purity in tumour sample	Required	Required
`PLOIDY`	Tumour ploidy in tumour sample	Required	Required
`MUT_TYPE`	Optional flag for mutation type (either “SNV” or “INDEL”)	Optional	Not required
`MUT_COPY`	Mutation copy number/multiplicity	Not required	Required
`CCF_PHYLO`	Mutation PhyloCCF	Not required	Required
`CCF_OBS`	Mutation CCF	Not required	Required
`CLUSTER`	Mutation cluster assignment	Not required	Required

Columns from left to right: (1) input.tsv column name, (2) description of column input data, (3) requirement of column for input into clustering, and (4) requirement of column for input into tree building.

PROCEDURE

Stage 1: Mutation clustering - TIMING: 10 min - 6 hrs

! CRITICAL. The tumour identifier in column `CASE_ID` and tumour sample identifier in column `SAMPLE` must include a prefix character string common to all patients in the cohort, for example prefix `'CRUK'` in the toy case `CRUK0000` (Figure 4). The input table should be in tab-separated format (`input.tsv`), should have no additional column with row names or numbers, and should have no quotation marks for character string entries.

! CRITICAL. In cases of multiple genomically distinct tumours detected within one patient, CONIPHER should be implemented separately for each tumour (Supplementary Note 1, Considering patients input to CONIPHER with multiple genomically distinct tumours)**.**

1| Prepare `input.tsv` file. An example `input.tsv` for TRACERx case CRUK0063 is shown below. The case CRUK0063 has WES data available for 5 primary tumour samples (`CRUK0063_SU_T1.R3 - CRUK0063_SU_T1.R7`) and two metastatic samples (`CRUK0063_SU_FLN1 - CRUK0063_BR_T1.R1`):

```
CASE_ID         SAMPLE   CHR      POS      REF      ALT      REF_COUNT        VAR_COUNT        DEPTH
       COPY_NUMBER_A    COPY_NUMBER_B      ACF      PLOIDY   MUT_COPY CCF_PHYLO        CCF_OBS  CLUSTER
CRUK0063        CRUK0063_BR_T1.R11         1854811  C        G        406      0        406      3        0
       0.19     2.85     0        0        0        1
CRUK0063        CRUK0063_SU_FLN1 1         1854811  C        G        111      26       137      2        1
       0.16     3.44     2.51     1.33     1        1
CRUK0063        CRUK0063_SU_T1.R31         1854811  C        G        222      0        222      2        1
       0.12     3        0        0        0        1
CRUK0063        CRUK0063_SU_T1.R41         1854811  C        G        155      43       198      2        1
       0.26     3.65     1.9      0.949    1        1
CRUK0063        CRUK0063_SU_T1.R51         1854811  C        G        184      43       229      2        1
       0.25     3.83     1.71     0.857    1        1
CRUK0063        CRUK0063_SU_T1.R61         1854811  C        G        205      42       247      2        1
       0.14     3.64     2.59     1.3      1        1
CRUK0063        CRUK0063_SU_T1.R71         1854811  C        G        177      32       209      2        1
       0.13     3.6      2.5      1.27     1        1
CRUK0063        CRUK0063_BR_T1.R11         2525963  -        A        105      25       130      3        0
       0.19     2.85     2.58     1.13     1        4
```

When running the clustering and tree building pipeline for a cohort of tumours, it is recommended to save the input and output in a distinct directory for each tumour case `${CASE_ID}`, for example:

```
inputTSV="${CASE_ID}/input.tsv"

outDir="${CASE_ID}/"
```

2| (Optional) Prepare `input_seg.tsv` file. This table can optionally be provided as input to create an across-genome copy number plot. This file describes the estimated copy number across the genome for each tumour sample. An example `input_seg.tsv` for case CRUK0063 is shown below:

```
SAMPLE CHR      STARTPOS ENDPOS   COPY_NUMBER_A    COPY_NUMBER_B
CRUK0063_SU_T1.R3       1        1154343  24194770 1.84     0.819
CRUK0063_SU_T1.R4       1        1154343  24194770 2.06     0.987
CRUK0063_SU_T1.R5       1        1154343  24194770 2.03     1.02
CRUK0063_SU_T1.R6       1        1154343  24194770 1.99     0.969
CRUK0063_SU_T1.R7       1        1154343  24194770 1.97     0.995
CRUK0063_BR_T1.R1       1        1154343  24194770 3.28     0.264
CRUK0063_SU_FLN1        1        1154343  24194770 1.89     0.843
CRUK0063_SU_T1.R3       1        24200891 24201115 0.802    0
```

! CRITICAL STEP. For file `input_seg.tsv`, the tumour sample identifiers in column `SAMPLE` and chromosome identifiers in column `CHR` should correspond to those in `input.tsv`.

3| Run mutation clustering for one patient with the following command, specifying inputs for the parameters of file names: —`patient`, --`out_dir`, --`input_tsv_loc`, and optionally --`input_seg_tsv_loc`:

```
Rscript run_clustering.R --case_id ${CASE_ID} --out_dir ${outDir} --
input_tsv_loc ${inputTSV}
```

A full description of all parameters available for the clustering stage can be found in Table 3. For anticipated outputs see Box 1 and Figure 5.

Table 3. Description of parameters as input into CONIPHER clustering stage.

Parameter	**Parameter description**	**Parameter data type (in R)**	**Default value**
--case_id	A tumour case identifier	Character	Please specify

--out_dir	File path to desired output directory	Character	Please specify
--input_tsv_loc	File path to input mutation table in correct format	Character	Please specify
--input_seg_tsv_loc	File path to a copy number segment table used for plotting only	Character	Optionally specify (default = NULL)
--subclonal_copy_correction	Should subclonal copy number correction be used?	Logical	TRUE
--only_truncal_subclonal_copy_correction	Should only truncal subclonal copy number correction be used?	Logical	TRUE
--pyclone_yaml_loc	File path to template yaml file for PyClone (to specify Dirichlet clustering parameters). If not specified, the default CONIPHER yaml file is used	Character	Optionally specify (default = NULL)
--min_cluster_size	Minimum number of mutations required in a cluster to be included in the analysis	Integer	5
--multiple_test_correction	Should multiple testing correction be applied for the copy number correcting mutations?	Logical	TRUE
--clean_clusters	Should the clusters be cleaned and merged?	Logical	TRUE
--clonal_cutoff	Lower threshold of PhyloCCF to be considered truncal	Double	0.9
--propClonal_thresh ol d	Proportion of mutations in a cluster that need to be considered truncal in order to merge back into the trunk	Double	0.25
--fix_absentCCFs	Should PhyloCCF of absent mutations be set to 0?	Logical	TRUE
--burn_in	Burn in for Dirichlet Process	Integer	1000
--seed	Seed for PyClone	Integer	1024

--nProcs	Number of cores allocated to run script in parallel	Integer	1

Commented [6]: We have decided to remove this

Deleted: Wrapper script filename: run_clustering.R.

Box 1: ANTICIPATED CLUSTERING OUTPUT

Running the clustering stage will output the following files in patient-specific directory "${CASE_ID}/Clustering/":

OUTPUT DATA:

- <CASE_ID>.SCoutput.CLEAN.tsv. This is a mutation table in the same format as input.tsv, including columns for: mutation cluster assignments (CLUSTER); mutation cell fraction estimates, including the PhyloCCF (CCF_PHYLO) and observed CCF (CCF_OBS); and mutation copy number estimates for each mutation in each sample (MUT_COPY). Additionally, there is a column mutation_id, which is an identifier for the mutation in the form: <CASE_ID>:<CHR>:<POS>:<REF>:<ALT>. By convention, cluster names are integers, ordered by the number of mutations assigned to that cluster (so the cluster with the largest number of mutations will be labelled as CLUSTER==1, and so forth).
- <CASE_ID>.SCoutput.FULL.tsv. This is a mutation table in the same format as the file <CASE_ID>.SCoutput.CLEAN.tsv described above, except with one additional column: CLEAN, which is a logical flag indicating whether this mutation was deemed 'dirty' and removed (CLEAN==FALSE), or deemed 'clean' and kept (CLEAN==TRUE). The subset of this table based on CLEAN==TRUE is identical to the table <CASE_ID>.SCoutput.CLEAN.tsv.
- <CASE_ID>.removed.muts.txt. This is a mutation table containing the mutations that were removed during the clustering stage. Each row is a new mutation. Tumour sample-specific information is found in columns that begin with <SAMPLE>.*. NOTE: this file will not be generated if no mutations are removed.

OUTPUT PLOTS:

- `<CASE_ID>_pyclone_cluster_assignment_copynumber_clean.pdf`. This output is a series of across-genome plots of each mutation plotted at its genomic position (x-axis) against its mutation copy number (y-axis), coloured by the cluster it was assigned to in mutation clustering. Each new row shows a new tumour sample. If the `input_seg.tsv` file was additionally provided, the copy number of each segment will be plotted: black indicates allele A, while green indicates allele B. The first page of the pdf displays all mutations from every cluster. The subsequent pages display the same segment copy number information for each sample, with mutations from only one cluster overlaid. Histograms on the right hand side of cross-genome plots (on all pages except the first page) indicate the frequency of mutations at each copy number value. An example of sample 5 from page 1 of the pdf for case CRUK0063 is shown in Figure 5a.
- `<CASE_ID>.removedCPN.muts.pdf`. This output plot is identical to the above, except restricting to only mutations that were removed during the mutation clustering post-processing step. Each mutation is coloured by the cluster it was assigned to in mutation clustering. Each new row shows a new tumour sample. Histograms on the right hand side indicate the frequency of mutations at each integer copy number value. NOTE: this file will not be generated if no mutations are removed. An example of samples CRUK0063_BR_T1.R1 and CRUK0063_SU_FLN1 for case CRUK0063 are shown in Figure 5b.
- `<CASE_ID>.heatmap.pdf`. This output shows a heatmap of presence/absence of each mutation (rows) in each tumour sample (columns). The colour bar on the left indicates removed mutations (blue) and kept mutations (yellow).
- `<CASE_ID>.cluster.ccf.heatmap.pdf`. This output shows a heatmap of the inferred PhyloCCF of each mutation (rows) in each tumour sample (columns). The colour bar on the left indicates the assigned cluster.

- `<CASE_ID>.pyclone_cluster_assignment_phylo_clean.pdf`. This output shows a scatter plot of the PhyloCCF of each (non-removed) mutation between each pair of samples. Each mutation is coloured by the assigned cluster. An example of one pair of samples from case CRUK0063 is shown in Figure 5c.

[Production: end of Box 1]

Stage 2: Phylogenetic tree building - TIMING: 1 min - 1 hrs

4| Run tree building for one patient with one of the following commands:

- If running CONIPHER tree building from CONIPHER clustering output:

```
Rscript run_treebuilding.R --input_tsv_loc
${outDir}"/Clustering/"${CASE_ID}".SCoutput.CLEAN.tsv" --out_dir
${outDir} --prefix CRUK
```

- If running CONIPHER tree building directly from an `input.tsv` file:

```
Rscript run_treebuilding.R --input_tsv_loc ${inputTSV} --out_dir
${outDir} --prefix CRUK
```

A full description of all parameters for the tree building stage can be found in Table 4. For anticipated outcomes see Box 2 and Figures 6 and 7. Guidance on exploring the mutations removed during the CONIPHER tree building stage can be found in Supplementary Note 2. Additional output produced by CONIPHER includes data for analysis in R, described in the Supplementary Note 3.

Commented [7]: Your Supplementary Notes are currently callout out of sequence. I therefore suggest moving this citation to Supplementary Note 3 from the Troubleshooting section to here.

Commented [8]: Thanks, we have now renumbered these and checked they are cited in the right order.

Table 4. Description of parameters as input into CONIPHER tree building stage.

Parameter	**Parameter description**	**Paramete r data type (in R)**	**Default value**
–input_tsv_loc	File path to input mutation table in correct format	Character	Please specify
–out_dir	File path to desired output directory	Character	Please specify
–prefix	Tumour identifier prefix	Character	Please specify
–ccf_buffer	PhyloCCF buffer allowance for testing tree level issue	Integer	10
–pval_cutoff	P-value cut off for testing tree level issue	Double	0.01
–use_boot	Should bootstrapping be used to compute confidence intervals?	Logical	TRUE
–merge_clusters	Should similar clusters be merged if possible?	Logical	TRUE
–correct_cpn_clusters	Should clusters driven by copy number errors be removed?	Logical	TRUE
–adjust_noisy_cluster s	Should noisy clusters be adjusted?	Logical	FALSE
–adjust_noisy_cluster s_prop	What is the minimum proportion of mutations required to be present in a sample to avoid cluster adjustment?	Double	0.05
–min_ccf	What is the minimum CCF threshold to consider a mutation as present in a sample?	Double	0.01
–min_cluster_size	What is the minimum number of mutations required in a cluster to be included in analysis?	Integer	5

–multi_trees	Should alternative plausible tumour phylogenies be explored?	Logical	TRUE

Wrapper script filename: run_treebuilding.R.

! CRITICAL STEP. NOTE: CONIPHER tree building implements its own cluster merging process in addition to cluster merging in the CONIPHER clustering stage (Supplementary Methods 2). By default similar clusters are merged if possible (`merge_clusters==TRUE`) and bootstrapped confidence intervals are used (`use_boot==TRUE`). These settings are recommended.

! CRITICAL STEP. If running tree building only, it is required that all columns in the input file `${inputTSV}` are present. NOTE: if an alternative clustering method is used, which does not output an estimate of PhyloCCF as well as observed CCF per mutation, the column `CCF_PHYLO` should be manually added to the input table, with identical entries to column `CCF_OBS`.

Box 2: ANTICIPATED TREE BUILDING OUTPUT

Running the tree building stage will output the following files in patient-specific directory `“${CASE_ID}/Trees/”`:

OUTPUT DATA:

- `allTrees.txt`. This is a text file containing all potential inferred phylogenetic trees, in the format below. This file can be parsed into any scripting language for further analysis.

```
### 11 trees
# tree 1
2       1
8       3
21      4
1       5
…
# tree 2
2       8
8       21
2       1
17      20
…
```

The first row of the file indicates how many alternative phylogenies were detected by the tree building algorithm. Each alternative tumour phylogeny number `X` begins with a header: `# tree X`. For each tree, each new row of `allTrees.txt` is a tree branch, or edge, connecting a pair of distinct clusters. The first column indicates the parental node; the second column indicates the child node.

! CRITICAL. Tree number 1 (`# tree 1`) always refers to the default tree generated by the tree building algorithm.

- `alternativeTreeMetrics.txt`. This is a tab-delimited text file containing summary metrics of all alternative phylogenetic trees, whereby each row of the table indicates one alternative tree (`treeID`).

```
treeID sum_condition_error      SCE_ranking      lowest_SCE       edge_probability_score
       edge_probability_ranking  highest_edge_probability
1      2.46     1        Lowest SCE tree  -13.8    1        Highest edge probability tree
2      2.67     2        Alternative tree -13.8    1        Highest edge probability tree
3      2.85     4        Alternative tree -28.2    6        Alternative tree
4      3.19     8        Alternative tree -17.1    3        Alternative tree
5      2.8      3        Alternative tree -18.3    5        Alternative tree
6      3.09     6        Alternative tree -14.9    2        Alternative tree
7      3.47     11       Alternative tree -17.1    3        Alternative tree
8      3.05     5        Alternative tree -18.3    5        Alternative tree
9      3.36     9        Alternative tree -14.9    2        Alternative tree
10     3.47     10       Alternative tree -18.2    4        Alternative tree
11     3.19     7        Alternative tree -18.2    4        Alternative tree
```

The `treeID` column value directly corresponds to the alternative tree number in the full alternative tree list `allTrees.txt`. The second column `sum_condition_error` gives the sum condition error value for that tree, and subsequent column `SCE_ranking` is an ordering of the trees from lowest error (`SCE_ranking==1`) to highest. Correspondingly, `lowest_SCE` is a binary flag to indicate whether this tree had the lowest error (`'Lowest SCE tree'`) or not (`'Alternative tree'`). Similarly, the fourth column `edge_probability_score` gives the edge probability score for that tree, and subsequent column `edge_probability_ranking` is an ordering of the trees from highest edge probability (`edge_probability_ranking==1`) to lowest. Column `highest_edge_probability` is a binary flag to indicate whether this tree had the maximal edge probability score (`'Highest edge probability tree'`) or not (`'Alternative tree'`). Any ties within either `SCE_ranking` or `edge_probability_ranking` are labelled with the same rank.

- `clusterInfo.txt`. This is a tab-delimited text file containing a table detailing information about each mutation cluster, whereby each row of the table indicates one cluster (`clusterID`) in one tumour sample (`SAMPLE`), as shown below.

```
clusterID truncal        treeClust        cpnRemClust      nMuts    SAMPLE   meanCCF  CCF_CI_low
          CCF_CI_high    clonality        clone_proportions_default
1         FALSE  TRUE    FALSE    174     CRUK0063_SU_T1.R30       0        0        absent   0
1         FALSE  TRUE    FALSE    174     CRUK0063_SU_T1.R499      96.4     102      clonal   0
1         FALSE  TRUE    FALSE    174     CRUK0063_SU_T1.R599      96.9     101      clonal   0
1         FALSE  TRUE    FALSE    174     CRUK0063_SU_T1.R698      95       100      clonal   28
1         FALSE  TRUE    FALSE    174     CRUK0063_SU_T1.R7100     96.5     103      clonal   0
1         FALSE  TRUE    FALSE    174     CRUK0063_SU_FLN1 103     99.2     107      clonal   0
1         FALSE  TRUE    FALSE    174     CRUK0063_BR_T1.R10       0        0        absent   0
2         TRUE   TRUE    FALSE    91      CRUK0063_SU_T1.R3101     95.3     106      clonal   0
```

The cluster name in `clusterID` matches the cluster names input into tree building (in either `<CASE_ID>.SCoutput.CLEAN.tsv` or `input.tsv`). The second column `truncal` indicates whether this cluster was assigned to be the truncal cluster of the phylogenetic tree. Only one unique cluster will be assigned to be truncal. The third column `treeClust` indicates whether the cluster was assigned to a branch of the phylogenetic tree (`treeClust==TRUE`). If a cluster was identified as erroneous due to being composed of biologically implausible mutations only, column `treeClust` will have a value of `FALSE`. If the cluster was identified as erroneous due to subclonal copy number alterations undetected during clustering, column `treeClust` will have a value of `FALSE` and `cpnRemClust` will have a value of `TRUE`. Column nMuts describes the number of SNVs assigned to that cluster. The columns `meanCCF`, `CCF_CI_low`, and `CCF_CI_high` describe the distribution of PhyloCCF values for all mutations in that `clusterID` in that `SAMPLE`. Column `clonality` describes whether that `clusterID` in that `SAMPLE` was classified as being either: absent, subclonal or clonal within that sample (Supplementary Methods 2). Finally, column `clone_proportions_default` describes the subclone proportion of that `clusterID` in that `SAMPLE`, computed from the default phylogenetic tree (tree 1).

- `cloneProportionsMinErrorTrees.txt`. This is a tab-delimited text file containing subclone proportion tables in long format from only phylogenetic trees with the lowest

SCE. Each row corresponds to one `clusterID` from one `treeID`. In example CRUK0063 below, the lowest SCE tree was the default tree (tree 1). Values in the table indicate the subclone proportion of the subclone resulting from that `clusterID` within that sampled tumour sample (column). For each `treeID`, columns should sum to 100.

```
CRUK0063_SU_T1.R3       CRUK0063_SU_T1.R4CRUK0063_SU_T1.R5CRUK0063_SU_T1.R6CRUK0063_SU_T1.R7CRUK0063_SU_FLN1
        CRUK0063_BR_T1.R1clusterID        treeID
0       0       0       28      0       0       0       1       1
0       0       0       0       0       0       0       2       1
6.65    0       0       0       0       0       0       3       1
0       0       0       0       0       0       38      4       1
0       0       0       0       0       5.92    0       5       1
0       25      0       27      25      0       0       6       1
0       0       0       23      1       0       0       7       1
0       0       0       0       0       0       0       8       1
57      0       0       0       0       0       0       9       1
17      0       0       0       0       0       0       10      1
0       0       0       0       0       0       37      11      1
0       6.68    11.5    0       0       0       0       12      1
0       0       0       0       0       0       13      15      1
0       0       0       0       0       0       12      16      1
0       17.8    0       0       18.3    0       0       17      1
0       0       0       0       0       33.1    0       18      1
0       0       0       0       0       61      0       19      1
0       0       25.5    0       0       0       0       20      1
19.4    0       0       0       0       0       0       21      1
0       0       14      0       27      0       0       22      1
0       50.5    0       22      28.7    0       0       23      1
0       0       49      0       0       0       0       24      1
```

- `subclonalExpansionScoreMinErrorTrees.txt`. This is a tab-delimited text file containing subclonal expansion scores (column `subclonal_expansion_score`) for each tumour sample (column `sample`) computed from the phylogenetic tree(s) with the lowest SCE (column `treeID`). The subclonal expansion score is computed as the maximum PhyloCCF of all terminal (leaf) nodes present in that tumour sample. NOTE: for multi-sample tumour cases, there may exist a sample with no terminal node present, in which case the subclonal expansion score for this sample is set to 0. The tumour level subclonal expansion score is taken as the maximum subclonal expansion score across tumour samples.

```
sample      subclonal_expansion_score       subclonal_expansion_score_tumour   treeID
CRUK0063_SU_T1.R30.568    0.619    1
CRUK0063_SU_T1.R40.509    0.619    1
CRUK0063_SU_T1.R50.493    0.619    1
CRUK0063_SU_T1.R60.505    0.619    1
CRUK0063_SU_T1.R70.619    0.619    1
CRUK0063_SU_FLN1 0.613    0.619    1
```

```
CRUK0063_BR_T1.R10.366    0.619    1
```

- `consensusBranches.txt`. This is a text file containing all branches (parent-child pairs) of the phylogenetic tree that were identified to be present across all alternative phylogenies, as shown in example CRUK0063 below. First column: parent node; second column: child node.

```
1      18
1      5
10     9
12     6
…
```

- `consensusRelationships.txt`. This is a text file containing all ancestor-descendent pairs that were identified to be present across all alternative phylogenies, as shown in example CRUK0063 below. First column: ancestral node; second column: descendent node.

```
1      12
1      17
1      18
1      19
…
```

- `treeTable.tsv`. This is a tab-separated mutation table in the format of `input.tsv`, except with extra columns: `originalCLUSTER` and `treeCLUSTER`. `originalCLUSTER` indicates the cluster ID this mutation was assigned to in the clustering stage (and will correspond to column `CLUSTER` in the `input.tsv` to the tree building stage). `treeCLUSTER` indicates the final cluster name the mutation is assigned to after treebuilding. Note: `originalCLUSTER` and `treeCLUSTER` are identical, except in cases of cluster merging (Supplementary Methods 2).

```
CASE_ID  SAMPLE   CHR      POS      REF      ALT      REF_COUNT                  VAR_COUNT         DEPTH  COPY_NUMBER_A
         COPY_NUMBER_B     ACF      PLOIDY   MUT_COPY CCF_PHYLO                  CCF_OBS           originalCLUSTER
         treeCLUSTER

CRUK0063 CRUK0063_SU_T1.R31         1854811  C        G                          222               0      222      2
         1        0.12     3        0        0        0                          1                 1
```

Commented [9]: The spacing between the columns here is uneven. Please confirm that this is how you want it to be reproduced, otherwise, please adjust the spacing.

Commented [10]: This is because it's a tab separated table copied in, with column names that are different lengths. Is there a way to get the spacing to be consistent?

```
CRUK0063 CRUK0063_SU_T1.R31        2525963  -        A                          301              2       303      2
         1        0.12     3       0.14     0        0.11                       4                4

CRUK0063 CRUK0063_SU_T1.R31        6311357  A        G                          473              46      519      2
         1        0.12     3       1.54     0.833    1                          2                2

CRUK0063 CRUK0063_SU_T1.R31        11845281 G        C                          287              0       287      2
         1        0.12     3       0        0        0                          1                1

CRUK0063 CRUK0063_SU_T1.R31        19683908 G        C                          133              0       133      2
         1        0.12     3       0        0        0                          1                1

CRUK0063 CRUK0063_SU_T1.R31        24082766 C        T                          145              0       146      2
         1        0.12     3       0        0        0                          1                1

CRUK0063 CRUK0063_SU_T1.R31        25784848 -        T                          458              0       458      2
         1        0.12     3       0        0        0                          1                1

CRUK0063 CRUK0063_SU_T1.R31        27105550 C        T                          312              4       316      2
         1        0.12     3       0.219    0.219    0.22                       21               21
```

OUTPUT PLOTS:

- `pytree_and_bar.pdf`. The left side of the output shows a barplot of the mean estimated PhyloCCF values of each mutation cluster (rows) in each sample (columns), with a bootstrap computed 95% confidence interval (Figure 1, Supplementary Methods 2). If the cluster was classified as 'clonal' within that tumour sample, the corresponding bar has a black outline (for example, bars for truncal cluster 2 have a black box in every tumour sample). The right hand side of the figure shows the inferred default phylogenetic tree. Each node pie chart corresponds to the same mutation cluster shown in the barplot, whereby each piece of the pie corresponds to a tumour sample sampled and is shaded by the mean PhyloCCF of that mutation cluster in that sample. The numbers of mutations per cluster are shown, as well as clusters identified as comprising erroneous mutations and removed (right side). If copy number driven clusters are removed, these are indicated on the right-hand side in the middle. Tree removed clusters are indicated on the bottom right-hand side. Tree branches that are black indicate this branch is a consensus branch and was found to be present in all alternative phylogenies. Grey branches indicate non-consensus branches. An example is shown in Figure 6.

- `pytree_multipletrees.pdf` displays all alternative phylogenetic trees identified in the tree building procedure. Black branches indicate consensus branches and grey branches indicate non-consensus branches. An example is shown in Figure 7.

[Production: end of Box 2]

?TROUBLESHOOTING

A troubleshooting table is provided (Table 5)..

Table 5. Troubleshooting.

Procedure Step	**Problem**	**Possible Reason**	**Possible Solution**
Step 3 | Clustering	Error message: `‘Sample IDs do not match between input_tsv and input_seg_tsv’`	This means that the sample ID character strings do not match in the two input files. `input_tsv` and `input_seg_tsv`’	Ensure the sample identifiers in the `input_seg.tsv` file are identical to the sample identifiers in the `input.tsv` file.
Step 4 | Tree building	Error message: `’No prefix specified. Please indicate a prefix for the current tumour case.’`	This means no tumour sample prefix was specified when running the tree building wrapper.	To fix the error, indicate a character string that represents the tumour case prefix using flag `--prefix` when running the tree building stage from command line.
Step 4 | Tree building	Error message: `‘Incorrect prefix specified. Please input the correct prefix for the current tumour case.’`	This means a prefix was specified that does not match a string within the values in the CASE_ID or SAMPLE columns in the input table.	To fix the error, input the correct prefix using flag `--prefix`.

Timing

Step 1, Data preprocessing: 5 min

Step 2, Data preprocessing (optional): 5 min

Step 3, Mutation clustering: 10 min - 6 hrs

Step 4, Phylogenetic tree building: 1 min - 1 hrs

We report average time to run CONIPHER mutation clustering and tree building stages on simulated tumours with varying numbers of samples and clones in Supplementary Note 4. For tumours with a large number of samples or mutations run time may be longer.

Commented [11]: This Note is called out of sequence. Please either include an earlier citation (before Note 2), or renumber you Notes. Alternatively, given that the note is essentially just a citation to Supplementary Table 1, you could remove the note entirely and just keep the table (although this would still need the remaining notes to be renumbered).

Commented [12]: Thanks, we have updated this - see response to comment above

ANTICIPATED RESULTS

A successful completion of the procedure results in, at least, the following output files:

CLUSTERING STAGE:

- `<CASE_ID>.SCoutput.CLEAN.tsv`
- `<CASE_ID>.removed.muts.txt` (if mutations were removed)
- `<CASE_ID>.pyclone_cluster_assignment_copynumber_clean.pdf`
- `<CASE_ID>.removedCPN.muts.pdf` (if mutations were removed)
- `<CASE_ID>.heatmap.pdf`
- `<CASE_ID>.cluster.ccf.heatmap.pdf`
- `<CASE_ID>.pyclone_cluster_assignment_phylo_clean.pdf`

TREEBUILDING STAGE:

- `allTrees.txt`
- `alternativeTreeMetrics.txt`
- `cloneProportionsMinErrorTrees.txt`
- `clusterInfo.txt`
- `cloneProportionsMinErrorTrees.txt`
- `subclonalExpansionScoreMinErrorTrees.txt`
- `consensusBranches.txt`
- `consensusRelationships.txt`
- `treeTable.tsv`
- `pytree_and_bar.pdf`
- `pytree_multipletrees.pdf`

A detailed description of the anticipated output files for clustering is given in Box 1 and Figure 5 and for tree building in Box 2 and Figures 6 and 7. Additional files produced during the Procedure are listed in Supplementary Note 3.

Code availability

Code to run the CONIPHER clustering and tree building wrapper functions can be found with documentation and run examples on the Github page https://github.com/McGranahanLab/CONIPHER-wrapper. The source code for the CONIPHER R package can be found on the Github page https://github.com/McGranahanLab/CONIPHER. The simulation framework can be found on the Github page https://github.com/zaccaria-lab/TRACERx_simulation_tool.

Commented [13]: Many thanks for the suggestion. We think the most convenient way for people to download the package and stay up to date with the latest version of CONIPHER is by hosting the code on GitHub. However we are looking into how we can get citable link of the current release version. Would it be ok to update this during the proof stage?

Commented [14]: Adding citations during proofing could be a problem if it means adding to the reference list (in case this requires renumbering references). If you cannot get a citable link, what you have here is good.

Commented [15]: If this is ok, then we would prefer to keep the github link as the main source of package installation/usage. Thanks

Data access statement

The Whole Exome Sequencing data (from the TRACERx study) used during this study has been deposited at the European Genome–phenome Archive (EGA), which is hosted by The European Bioinformatics Institute (EBI) and the Centre for Genomic Regulation (CRG) under the accession codes EGAS00001006494; access is controlled by the TRACERx data access committee. Details on how to apply for access are available on the linked page.

Acknowledgements

The TRACERx study (Clinicaltrials.gov no: NCT01888601) is sponsored by University College London (UCL/12/0279) and has been approved by an independent Research Ethics Committee (13/LO/1546). TRACERx is funded by Cancer Research UK (C11496/A17786) and coordinated through the Cancer Research UK and UCL Cancer Trials Centre which has a core grant from CRUK (C444/A15953). We gratefully acknowledge the patients and relatives who participated in the TRACERx study. We thank all site personnel, investigators, funders and industry partners that supported the generation of the data within this study.

This work was supported by the Francis Crick Institute that receives its core funding from Cancer Research UK (CC2041), the UK Medical Research Council (CC2041), and the Wellcome Trust (CC2041). This work was also supported by the Cancer Research UK Lung Cancer Centre of Excellence, the CRUK City of London Centre Award (C7893/A26233) and the UCL Experimental Cancer Research Centre.

Author contributions

K.G., A.H., E.C., A.M.F., K.T., N.J.B. and N.M. helped develop the protocol and wrote the manuscript. A.B. and S.Z. created the simulations, performed the benchmarking and wrote the manuscript. M.S.H. helped with bioinformatics pipeline development. C.S., S.Z. and N.M. jointly designed and supervised the study and helped write the manuscript.

Competing interests

A.M.F. is co-inventor to a patent application to determine methods and systems for tumour monitoring (PCT/EP2022/077987).

N.J.B. is a co-inventor to a patent to identify responders to cancer treatment (PCT/GB2018/051912).

C.S. acknowledges grant support from AstraZeneca, Boehringer-Ingelheim, Bristol Myers Squibb, Pfizer, Roche-Ventana, Invitae (previously Archer Dx Inc - collaboration in minimal residual disease sequencing technologies), and Ono Pharmaceutical. He is an AstraZeneca Advisory Board member and Chief Investigator for the AZ MeRmaiD 1 and 2 clinical trials and is also Co-Chief Investigator of the NHS Galleri trial funded by GRAIL and a paid member of GRAIL's Scientific Advisory Board. He receives consultant fees from Achilles Therapeutics (also SAB member), Bicycle Therapeutics (also a SAB member), Genentech, Medicxi, Roche Innovation Centre – Shanghai, Metabomed (until July 2022), and the Sarah Cannon Research Institute. ad stock options in Apogen Biotechnologies and GRAIL until June 2021, and currently has stock options in Epic Bioscience, Bicycle Therapeutics, and has stock options and is co-founder of Achilles Therapeutics. C.S. holds patents relating to assay technology to detect tumour recurrence (PCT/GB2017/053289); to targeting neoantigens (PCT/EP2016/059401),

identifying patent response to immune checkpoint blockade (PCT/EP2016/071471), determining HLA LOH (PCT/GB2018/052004), predicting survival rates of patients with cancer (PCT/GB2020/050221), identifying patients who respond to cancer treatment (PCT/GB2018/051912), US patent relating to detecting tumour mutations (PCT/US2017/28013), methods for lung cancer detection (US20190106751A1) and both a European and US patent related to identifying insertion/deletion mutation targets (PCT/GB2018/051892) and is co-inventor to a patent application to determine methods and systems for tumour monitoring (PCT/EP2022/077987).

N.M. has received consultancy fees and has stock options in Achilles Therapeutics. N.M. holds European patents relating to targeting neoantigens (PCT/EP2016/ 059401), identifying patient response to immune checkpoint blockade (PCT/ EP2016/071471), determining HLA LOH (PCT/GB2018/052004), predicting survival rates of patients with cancer (PCT/GB2020/050221).

References

1. Greaves, M. & Maley, C. C. Clonal evolution in cancer. *Nature* **481**, 306–313 (2012).
2. Jamal-Hanjani, M. *et al.* Tracking the Evolution of Non-Small-Cell Lung Cancer. *N. Engl. J. Med.* **376**, 2109–2121 (2017).
3. Maley, C. C. *et al.* Genetic clonal diversity predicts progression to esophageal adenocarcinoma. *Nat. Genet.* **38**, 468–473 (2006).
4. Gerstung, M. *et al.* The evolutionary history of 2,658 cancers. *Nature* **578**, 122–128 (2020).
5. Frankell, A. M. *et al.* The evolution of lung cancer and impact of subclonal selection in TRACERx. *Nature* **616**, 525–533 (2023).
6. Dentro, S. C. *et al.* Characterizing genetic intra-tumor heterogeneity across 2,658 human cancer

genomes. *Cell* **184**, 2239–2254.e39 (2021).

7. Roth, A. *et al.* PyClone: statistical inference of clonal population structure in cancer. *Nat. Methods* **11**, 396–398 (2014).
8. Satas, G., Zaccaria, S., El-Kebir, M. & Raphael, B. J. DeCiFering the Elusive Cancer Cell Fraction in Tumor Heterogeneity and Evolution. *bioRxiv* 2021.02.27.429196 (2021).
9. Satas, G. & Raphael, B. J. Tumor phylogeny inference using tree-constrained importance sampling. *Bioinformatics* **33**, i152–i160 (2017).
10. Deshwar, A. G. *et al.* PhyloWGS: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. *Genome Biol.* **16**, 35 (2015).
11. Dentro, S. C., Wedge, D. C. & Van Loo, P. Principles of Reconstructing the Subclonal Architecture of Cancers. *Cold Spring Harb. Perspect. Med.* **7**, (2017).
12. Tarabichi, M. *et al.* A practical guide to cancer subclonal reconstruction from DNA sequencing. *Nat. Methods* **18**, 144–155 (2021).
13. Malikic, S., McPherson, A. W., Donmez, N. & Sahinalp, C. S. Clonality inference in multiple tumor samples using phylogeny. *Bioinformatics* **31**, 1349–1356 (2015).
14. El-Kebir, M., Oesper, L., Acheson-Field, H. & Raphael, B. J. Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. *Bioinformatics* **31**, i62–70 (2015).
15. Popic, V. *et al.* Fast and scalable inference of multi-sample cancer lineages. *Genome Biol.* **16**, 91 (2015).
16. Gundem, G. *et al.* The evolutionary history of lethal metastatic prostate cancer. *Nature* **520**, 353–357 (2015).
17. Al Bakir, M. *et al.* The evolution of non-small cell lung cancer metastases in TRACERx. *Nature* **616**, 534–542 (2023).
18. Martínez-Ruiz, C. *et al.* Genomic-transcriptomic evolution in lung cancer and metastasis. *Nature* **616**, 543–552 (2023).
19. Benjamin, D. *et al.* Calling Somatic SNVs and Indels with Mutect2. *bioRxiv* 861054 (2019) doi:10.1101/861054.
20. Koboldt, D. C. *et al.* VarScan 2: somatic mutation and copy number alteration discovery in

cancer by exome sequencing. *Genome Res.* **22**, 568–576 (2012).

21. Van Loo, P. *et al.* Allele-specific copy number analysis of tumors. *Proc. Natl. Acad. Sci. U. S. A.* **107**, 16910–16915 (2010).

22. Zaccaria, S. & Raphael, B. J. Accurate quantification of copy-number aberrations and whole-genome duplications in multi-sample tumor sequencing data. *Nat. Commun.* **11**, 4301 (2020).

23. Favero, F. *et al.* Sequenza: allele-specific copy number and mutation profiles from tumor sequencing data. *Ann. Oncol.* **26**, 64–70 (2015).

24. Nik-Zainal, S. *et al.* The life history of 21 breast cancers. *Cell* **149**, 994–1007 (2012).

25. Dentro, S. C. *et al.* Characterizing genetic intra-tumor heterogeneity across 2,658 human cancer genomes. *Cell* **184**, 2239–2254.e39 (2021).

26. Myers, M. A., Satas, G. & Raphael, B. J. CALDER: Inferring Phylogenetic Trees from Longitudinal Tumor Samples. *Cell Syst* **8**, 514–522.e5 (2019).

27. Wintersinger, J. A. *et al.* Reconstructing Complex Cancer Evolutionary Histories from Multiple Bulk DNA Samples Using Pairtree. *Blood Cancer Discov* **3**, 208–219 (2022).

Figure Legends

Figure 1. Overview of the CONIPHER clustering and tree building methods. **a.** Minor subclonal copy number alterations affecting the locus of mutations are corrected for, e.g. a gain affecting two mutations on chr3q of allele B (green line) in tumour sample S3. **b.** Mutations are grouped based on presence/absence within each tumour sample. **c.** Dirichlet clustering[7] is performed separately on each group of mutations determined in the previous step. This returns an estimate of the PhyloCCF of each mutation and its assigned cluster. **d.** Finally mutation copy number post-processing is applied to correct for errors propagated through mutation clustering. Clusters are removed if they are evidenced to be driven by copy number loss, and if subclonal copy number correction has created an additional subclonal cluster, this subclonal cluster's mutation PhyloCCF values are un-corrected and merged with the most similar cluster. **e.** The mean (bar plot, value) and 95% confidence intervals (black line) of the distribution of mutation PhyloCCFs for each inferred mutation cluster in each sample are computed. The total number of mutations per cluster are shown in brackets. Pie chart shading represents the mean CCF of that cluster in each sample. **f.** For each pair of clusters, PhyloCCF distributions are compared using two one-sided Wilcoxon tests, to test whether one cluster can be nested within the other. The truncal cluster is defined as that cluster which can nest all other clusters. The chromosomal

Commented [16]: Regarding your supplementary materials, currently you have your supplementary figures and table embedded in your supplementary methods and notes. Given that you have citations to your supplementary figures in your main text, I think it would be clearer if you separated them out eg the supplementary materials should be presented in the order: Methods, Notes, Figures, Table.

Commented [17]: Thanks, we have now moved the supplementary figures to the end of the supplementary materials, but kept the singular supplementary table in place (in Supplementary Note, Section 4- Expected run time).

distributions of all mutations in a cluster are checked and clusters are removed if all mutations are localised, indicating a missed copy number alteration. All clusters are additionally tested against the truncal cluster to determine cluster clonality in each tumour sample. This step returns a nesting matrix and ancestral graph. **g.** The ancestral graph is pruned to create a phylogenetic tree structure. Issue clusters that either (i) create cycles in the graph, or (ii) result in PhyloCCF values that sum to more than the user-defined threshold per tree level are identified and removed, to return a default phylogenetic tree. **h.** Clusters on the default tree are permuted to determine all possible alternative trees that do not cause issues (i) or (ii). **i.** Alternative phylogenetic trees are ranked according to two metrics, 1) trees generating the smallest amount of Sum Condition Error (SCE) (average amount of CCF error generated at each tree level), and 2) trees with the highest edge probability (trees comprising edges which appear most frequently across the solution space of alternative trees). **j.** Subclone proportions based on each mutation cluster on the tree are computed for the default tree and tree with lowest SCE.

Figure 2. Benchmarking CONIPHER against current state-of-the-art methods. The performance of CONIPHER was evaluated against existing methods on 150 simulated tumours (Simulated Dataset 1 – see Supplementary Methods 4). All existing methods were run using default parameters. Three categories of simulations were generated with different numbers of samples - low (2-3 samples), medium (4-7 samples), and high (>7 samples). **a.** CONIPHER clustering performance is compared against PyClone using the mutation clustering Adjusted Rand Index (ARI). **b**. CONIPHER tree building performance is compared against LICHeE, CITUP, and Pairtree using mutation descendent accuracy. **c.** Scalability of combined CONIPHER clustering and tree building is compared with LICHeE, CITUP, PhyloWGS and Pairtree. Bar plots indicate the success status of simulations run in the time frame allowed (8 hours). Opaque coloured bars indicate successfully completed simulations, transparent coloured bars indicate failed simulations, and grey bars indicate simulations that did not complete execution within the set time. **d.** Mutation presence precision is computed to compare presence/absence of mutations in CONIPHER, LICHeE, CITUP, and Pairtree. **e.** Truncal sensitivity is computed for truncal mutations to assess performance in the presence of mutation losses between CONIPHER, LICHeE, CITUP, Pairtree, and Pairtree tree building run with CONIPHER clustering. **f.** Run success of tree building methods based on the ground truth simulated clusters. **g.** Truncal sensitivity computed from tree building methods run on the ground truth simulated clusters. **h.** Detection of error-driven clusters is compared between CONIPHER, LICHeE, CITUP, and Pairtree using noisy cluster sensitivity. The box plots represent the upper and lower quartiles (box limits), the median (centre line) and the vertical bars span 1.5x the interquartile range).

Commented [18]: A few of your plots have error bars. What do they represent? Similarly, what range do the boxes in your box plots represent?

Commented [19]: Thanks, we have added this at the end of the figure legend

Figure 3. Method workflow. CONIPHER is composed of two stages that are run sequentially: mutation clustering and tree building. The tree building stage is compatible with other clustering methods.

Figure 4. Example input table for clustering and tree building: input.tsv. The clustering stage only considers the columns in the pink box. The tree building stage considers all columns in both pink and green boxes. This toy tumour example CRUK0000 has two sequenced tumour samples CRUK0000_R1 and CRUK0000_R2.

Figure 5. Example panels from CONIPHER clustering stage output. a. Panel of estimated mutation copy number of each mutation from each cluster in one sample from CRUK0063_pyclone_cluster_assignment_copynumber_clean.pdf. Mutations are coloured

according to their assigned cluster. The y-axis corresponds to the copy number, and the x-axis to the position along the genome. **b.** Panel of estimated mutation copy number of every mutation removed from the clustering stage in metastatic samples CRUK0063_BR_T1.R1 and CRUK0063_SU_FLN1 from CRUK0063.removedCPN.muts.pdf. Mutations are coloured according to their assigned cluster. The y-axis corresponds to the copy number, and the x-axis to the position along the genome. **c**. Scatter plot comparing estimated PhyloCCF of all mutations in two tumour samples CRUK0063_BR_T1.R1 and CRUK0063_SU_T1.R3 in CRUK0063.pyclone_cluster_assignment_phylo_clean.pdf. Mutations are coloured by their assigned cluster.

Figure 6. Example pytree_and_bar.pdf for case CRUK0063. This plot is produced in the tree building stage, and contains the estimated PhyloCCF bar plots for each cluster in each tumour sample (left), the inferred default tree structure (middle) and the number of mutations in each cluster kept in the tree (right top) and removed from the tree (right bottom).

Figure 7. Example pytree_multipletrees.pdf for case CRUK0063. This output plot contains all potential alternative phylogenetic trees inferred by CONIPHER in the tree building stage. The alternative trees are plotted by row, i.e. the top left tree plot is alternative tree # 1 (the default tree), to the right of this is alternative tree # 2, and so on.