Single-cell copy number calling and event history reconstruction

Jack Kuipers[1,2] *, Mustafa Anıl Tuncel[1] *, Pedro Ferreira[1,2], Katharina Jahn[1,2] and Niko Beerenwinkel[1,2]

[1] *Department of Biosystems Science and Engineering, ETH Zurich, Basel, Switzerland*
[2] *SIB Swiss Institute of Bioinformatics, Basel, Switzerland*

Copy number alterations are driving forces of tumour development and the emergence of intra-tumour heterogeneity. A comprehensive picture of these genomic aberrations is therefore essential for the development of personalised and precise cancer diagnostics and therapies. Single-cell sequencing offers the highest resolution for copy number profiling down to the level of individual cells. Recent high-throughput protocols allow for the processing of hundreds of cells through shallow whole-genome DNA sequencing. The resulting low read-depth data poses substantial statistical and computational challenges to the identification of copy number alterations. We developed SCICoNE, a statistical model and MCMC algorithm tailored to single-cell copy number profiling from shallow whole-genome DNA sequencing data. SCICoNE reconstructs the history of copy number events in the tumour and uses these evolutionary relationships to identify the copy number profiles of the individual cells. We show the accuracy of this approach in evaluations on simulated data and demonstrate its practicability in applications to a xenograft breast cancer sample.

Introduction

During tumour progression, cancer cells undergo complex and diverse genomic aberrations leading to heterogeneous cell populations and multiple evolving subclones [1, 2, 3]. Genomic sequencing has been powerfully employed to uncover the heterogeneity across cancer types [4] and to examine the link between tumour diversity and progression [5, 6, 7]. Heterogeneity may also allow the tumour additional ways to evolve resistance under treatment, such that intra-tumour genomic diversity is a cause of relapse and treatment failure [8, 9].

The need for a comprehensive understanding of the composition of each tumour for more precise and effective cancer therapies [10] can be addressed by sequencing tumours at the resolution of individual cells [11, 12]. The small amount of DNA in single cells has to be amplified before sequencing, which leads to characteristic and pronounced noise in the read count data. Specialised phylogenetic methods for single-cell sequencing data accounting for these noise patterns have been developed to detect point mutations and reconstruct the evolutionary history of tumours [13, 14].

In addition to point mutations, cancerous cells often undergo more complex genomic rearrangements, including copy number alterations (CNAs) such as amplifications and deletions. Since the first single-cell DNA sequencing [15], which allowed for the identification of CNAs, rapid progress has led to high-throughput methods [16] that can profile the whole genome of hundreds of single cells. By removing the need for pre-amplification, the protocol of [16] has particularly uniform coverage allowing a resolution of CNAs down to the 100kb scale. Recently, a commercial solution from 10x Genomics has been launched [17], further simplifying and accelerating the processing of hundreds of single cells.

With the availability of whole-genome single-cell sequencing data of this quantity and quality, we are now in the position to study the evolutionary history of CNA events in individual tumours, and further utilise this dependency structure to infer the copy number profiles of single cells. As previously shown for point mutations [18], the evolutionary relationships of tumour cells can be used to boost and

*Contributed equally

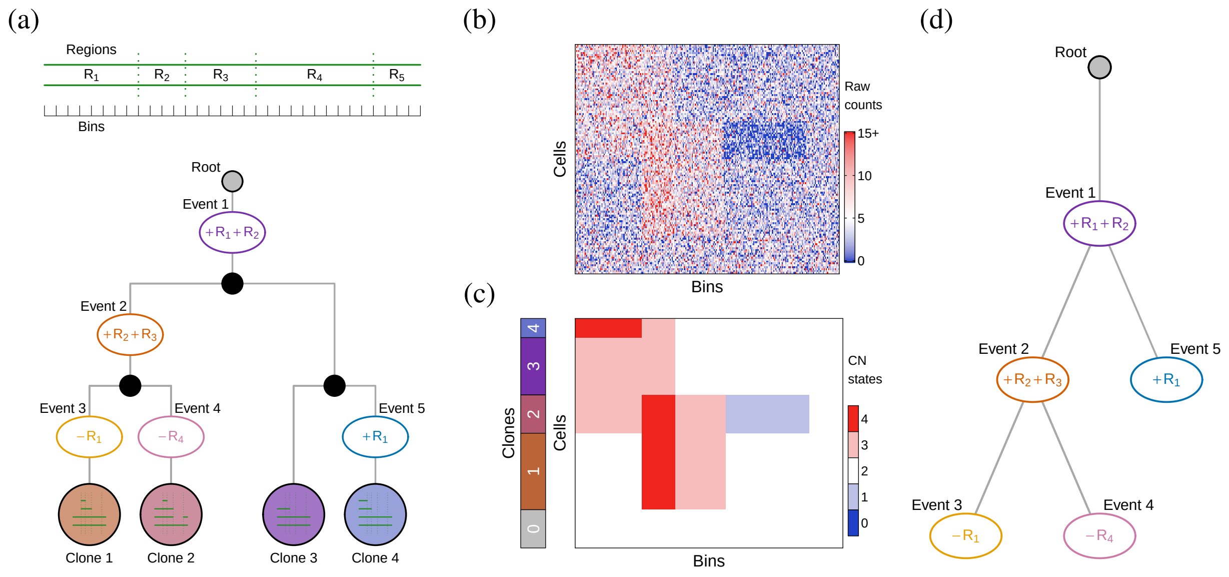

Figure 1: **CNA calling and tree inference.** (a) The genome is partitioned, by the four breakpoints depicted, into the five regions $R_1, \dots, R_5$ (each comprising several bins) that may experience CNAs. During tumour evolution, copy number changes may accumulate along the branches (plus signifies an amplification, minus a deletion) of the cell lineage tree to lead to distinct tumour subclones. The copy number profile of each clone can be obtained by tracing the lineages from the root, or neutral diploid cells. For example, Clone 4 has experienced two CN events, namely Event 1, a gain in a segment spanning regions R_1 and R_2, and Event 5, a further gain in R_1, such that the copy number profile of Clone 4 is $(4, 3, 2, 2, 2)$ across the five regions. (b) From single-cell sequencing we obtain noisy read count data according to the copy number profiles of the clones and their sizes. (c, d) Accounting for lineage relationships between cells and modelling the noise in the data, SCICoNE infers both the copy number calls in each cell (c) and the sequence of the CNAs and their phylogenetic relationships (d).

correct the weak and noisy signal provided by single-cell sequencing reads. However, modelling and inference of event histories is notably more involved in the case of CNAs, as such events may physically overlap, and will more readily reoccur and revert. Thus, the infinite sites assumption [19] cannot be employed as a basis for reconstructing each tumour's evolutionary history, and more complex models are needed. Here, we drop this assumption and develop SCICoNE (https://github.com/cbg-ethz/SCICoNE), a computational method for inferring CNA-based event histories and single-cell copy number profiles that is tailored to the shallow read-depth of present-day whole-genome single-cell sequencing data. We benchmark the performance of SCICoNE, and employ it to examine the copy number history of a xenograft breast cancer sample.

Results

Model overview During tumour evolution, CNAs can occur and accumulate in cancer cell sub-populations. Since all cells in a tumour are related through sharing a common ancestor, CNAs occur on a cell lineage tree and encode the copy number profiles of each clone or cell (Figure 1a). With the whole-genome sequencing of single cells [16, 17], we have read depth data for each cell and each bin along the genome (Figure 1b). After correction for confounders, including read mappability and genomic GC content, for each bin [20, 21], the corrected counts can be assumed to be proportional to the underlying copy number. To segment the genome into regions of consecutive bins which may have experienced CNAs, we develop a dynamic programming approach to detect breakpoints by combining evidence across the individual cells (Methods). For each potential breakpoint, we compare the likelihood of a step change in copy number state to that of a constant model, and then collate the information across cells to obtain the total signal of a copy number change at that genomic position.

For clinical applications, the history of copy number events is often more pertinent than the fully resolved cell genealogy, as the former is sufficient to determine which CNAs are mutually exclusive or co-occurring in the same tumour subclone, and to infer the order of CNA events. We therefore move from the cell or clone lineage tree (Figure 1a) to the CNA tree (Figure 1d). To account for the noise in the sequencing data, we develop a probabilistic model and an MCMC inference scheme for single-cell read counts (Methods). A major difference to MCMC schemes used for reconstructing trees of point mutations [22, 18] is that the infinite sites assumption, which excludes the possibility of multiple mutational hits at the same genomic site, can no longer be made [19]. CNAs may overlap and nest inside each other, so the model developed here allows for arbitrary violations of the infinite sites assumption and arbitrary reoccurrences of amplifications and deletions across different genomic regions.

By jointly estimating the copy number profiles of all cells and their underlying CNA tree, we leverage the evolutionary dependencies among cells for improved copy number calls. The output of SCICoNE comprises (1) the reconstructed tree representing the evolutionary history of all cells (Figure 1d) and (2) the inferred copy number profile for each individual cell (Figure 1c).

Reconstructing copy number trees from tumour data As a first demonstration of the applicability of SCICoNE to shallow whole-genome single-cell sequencing data, we considered a dataset of 260 cells of a triple negative breast cancer xenografted into a mouse (SA501X3F) [16]. The segmentation algorithm of SCICoNE (Methods, with default parameters) revealed 318 candidate breakpoints over the 18,175 bins.

The reconstructed tree (Figure 2 and Figure A1) starts with a large number of CNAs and then branches into several different clones. The first node includes deletions in *TP53* and the *MAPK* genes along with amplification in *PIK3CA*. Notably, several genes including *AKT1*, *ARID1B*, *ESR1* and *NTRK3* show high instability undergoing repeated and alternate amplifications and deletions, highlighting the complex evolutionary history of the sample.

The inferred copy number profiles (Figure 3b) are in good agreement with the normalised counts per bin (Figure 3a), recapitulating the CNAs across cells while accounting for the noise in the raw data and the phylogenetic relationships between the cells. Joint inference of copy number profiles and their evolutionary history provides increased power to separate signal from noise, as emphasised by comparing the raw count and inferred copy number profiles of example cells (Figure 3c). While some small horizontal changes for individual cells visible in the normalised counts (Figure 3a) are filtered out, most changes occurring in a small number of cells are detected (Figure 3b). Overall, we obtain the main CNAs across cells as well as their phylogeny (Figure 2).

To examine the relationship between CNAs and RNA expression changes at the single-cell level, we analysed the xenograft passage SA501X2B which underwent single-cell RNA sequencing [24] and compared the two molecular profiles. By smoothing the RNA signal [25], we observe common strips of over- and underexpression (Figure A2). Comparing to the DNA summarised at the gene level (by averaging over the bins in each of the 6000 expressed genes), we find some agreement between genomic copy number changes and expression levels (Figure A2). However, quite a few of the signals visible in the RNA data, like in chromosome 19, for example, have no basis in the DNA. The concordance and discrepancies between RNA expression and DNA count levels is further emphasised if we cluster the

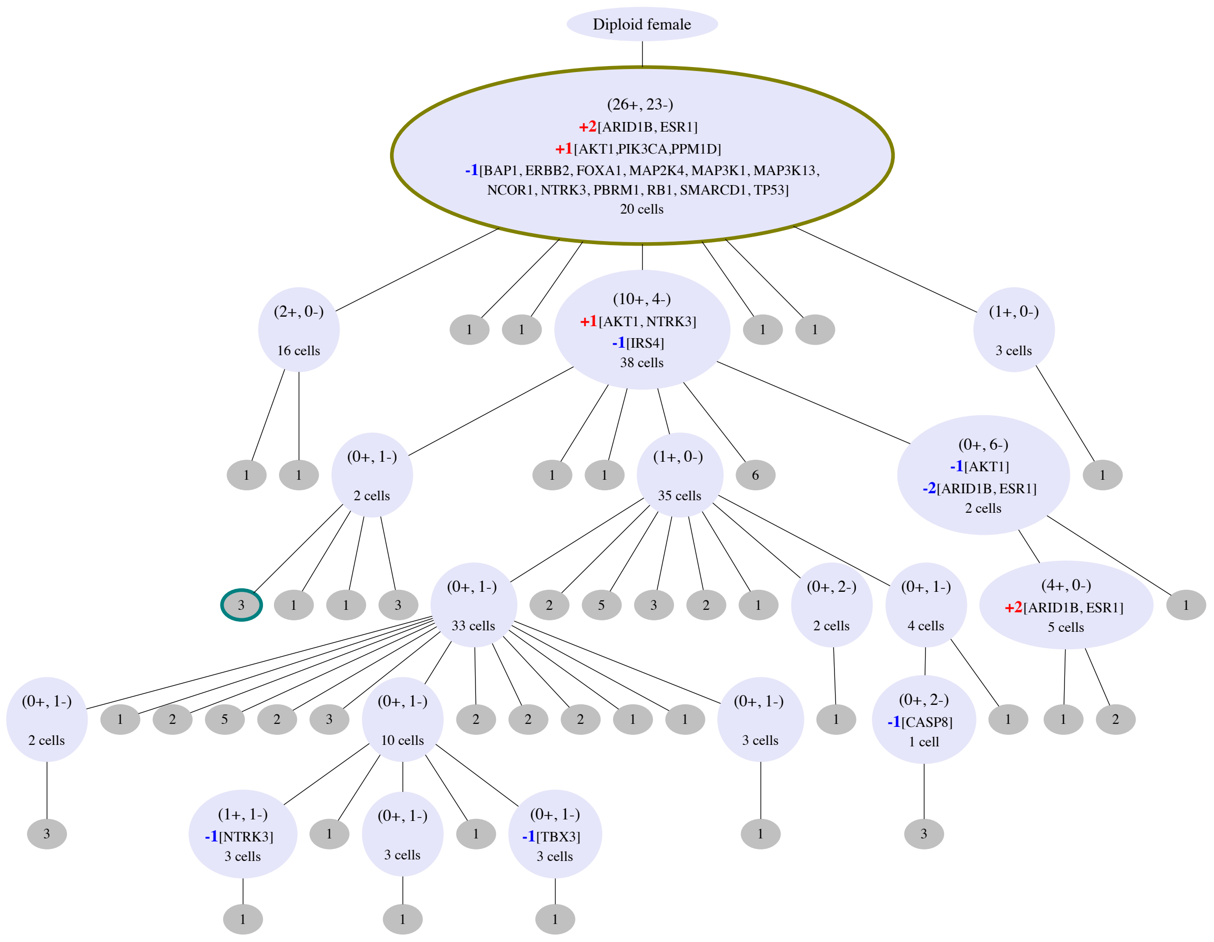

Figure 2: **Inferred tree for 260 cells from a breast xenograft [16].** Inside the nodes of the CNA tree we highlight the total number of amplified or deleted regions (in parentheses), the genes which are affected (among the 33 associated with breast cancer in the COSMIC Cancer Gene Census [23]), including how much they are amplified or deleted, and the number of cells that best attach to each node. The CNAs are not displayed at the grey (leaf) nodes where only the number of cells attached is indicated. Example profiles of cells attaching to the two nodes with coloured borders are displayed in Figure 3c.

cells (Figure A3). The finer structure and clear breakpoints visible in the DNA, which are reflected in the inferred evolutionary history (Figure 2) and copy number profiles (Figure 3) are not well reflected in the RNA (Figure A2). The results confirm on the level of individual cells that expression profiles are not in a strict one-to-one correspondence with copy number profiles and that the latter can be inferred with higher accuracy and in more detail from DNA data.

Benchmarking on simulated data To benchmark SCICoNE we conducted a simulation study (Methods) and compared the performance of SCICoNE to a suite of alternatives. For simulated trees with 20 nodes (Figure 4), we observe a steady increase in accuracy as we model our data in more detail. As a baseline, we cluster the normalised count data using hierarchical clustering or PhenoGraph [27] (Figure 4, gold and silver) and assign cells the averaged profile of their clusters (rounded to integers and centred at diploid). Then we build trees on PhenoGraph clustered data using SCICoNE to leverage information across the clusters and their shared evolutionary history to improve the accuracy of learning the copy number profiles (Figure 4, red and pink). Finally, we perform the full tree inference with SCICoNE on the single-cell data (Figure 4, purple and blue) and observe a strong improvement over

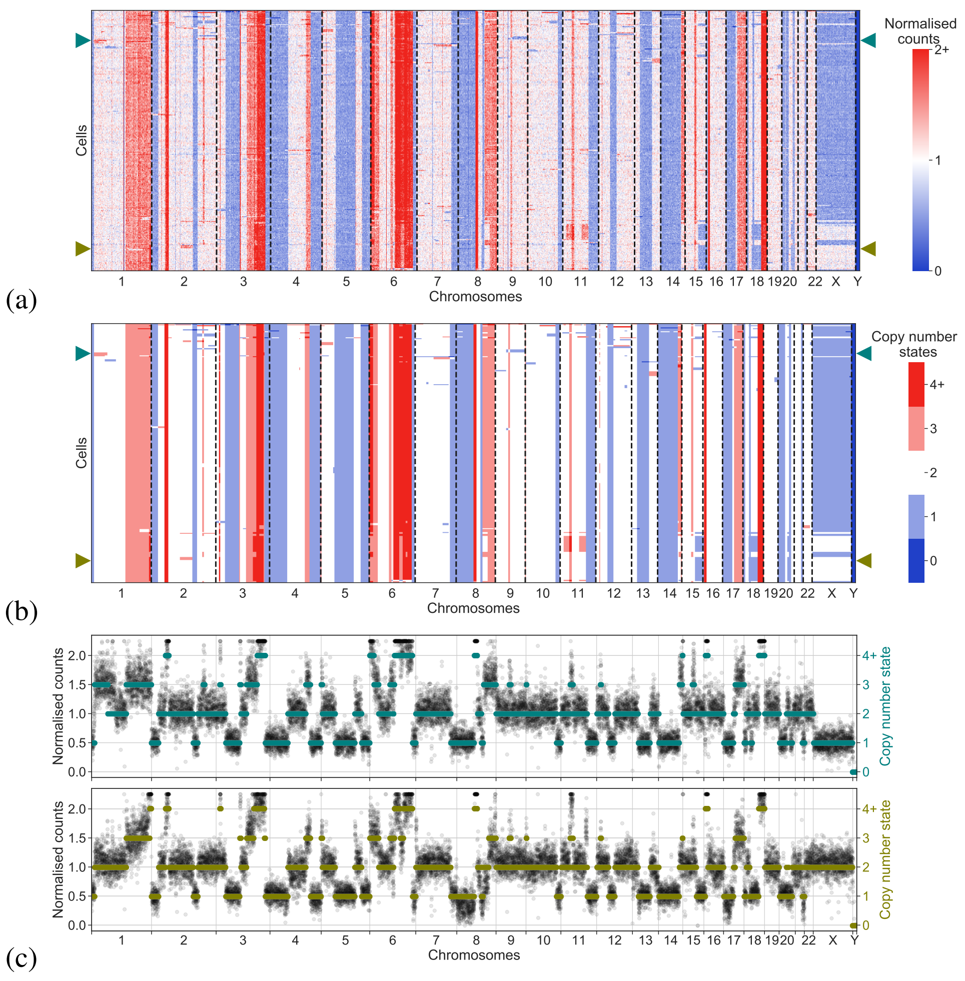

Figure 3: **Inferred copy number profiles for 260 cells from a breast xenograft [16].** (a) Normalised counts per bin. (b) Copy number profiles estimated jointly with the CNA tree in Figure 2. (c) Two examples of raw count data (black dots) and inferred copy number profiles (coloured lines) of the two cells indicated by arrows in the heatmaps of panels a and b.

the alternatives. On the full data, we find that the ‘max’ setting, which finds the best placement of each cell in the CNA tree, offers higher reconstruction quality relative to ‘sum’, which averages over their placements (Methods, Equation (9) and Equation (10)). We therefore used ‘max’ for the breast xenograft data above.

SCICoNE also performs well in comparison to HMMcopy [26] (Figure 4, red) which performs copy number calling per cell with a hidden Markov model, and has been found to have good overall performance in single-cell copy number calling [28]. However, for the simulated data with an average read depth of 2–8 (comparable to 10x Genomics data), HMMcopy performs worse than assuming no CNAs occurred (Figure 4, orange). In contrast, SCICoNE allows us to extract much more accurate copy number profiles than a diploid default or clustering methods (hierarchical and PhenoGraph), along with the evolutionary history encoded in the tree itself.

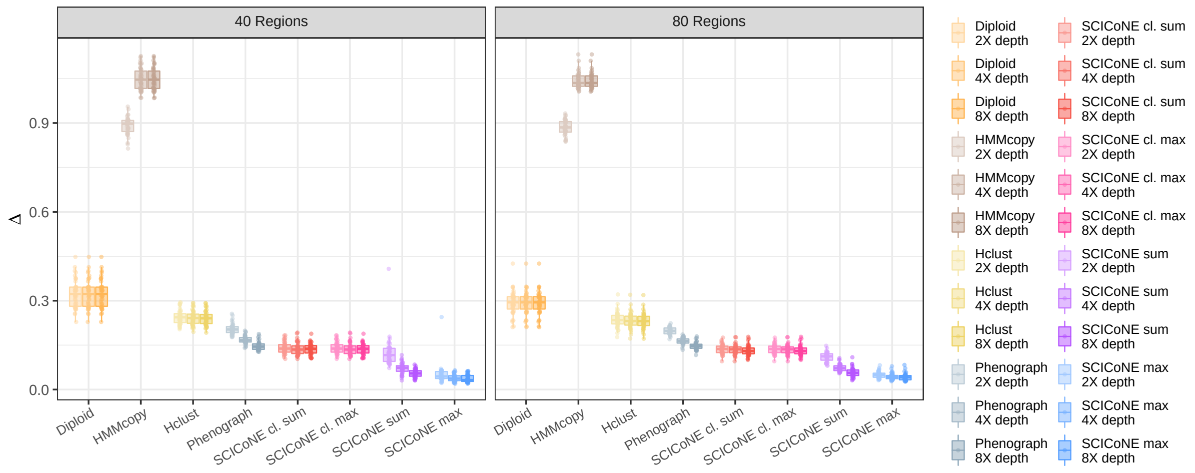

Figure 4: **Comparison of copy number calling for simulated data.** For random trees with 20 nodes, we attached 400 cells and simulated overdispersed read data according to each cell's copy number profile over 10,000 bins. The total number of reads was 20k, 40k and 80k for an average read depth of 2X, 4X and 8X (colour intensity, left to right) for each cell. The maximal number of regions affected by copy number changes was 40 and 80 (panels). The root mean squared difference Δ between the true simulated copy number profiles and the corresponding inferred profiles over all bins and cells is summarised in each box plot, for a neutral diploid profile (orange) and for profiles inferred by HMMcopy [26] (bronze), hierarchical (gold) and PhenoGraph (silver) clustering, as well as SCICoNE on PhenoGraph clustered data (red and pink) and SCICoNE on the full single-cell sequencing data (purple and blue).

Conclusions

For learning the copy number profiles of single cells, sharing information across cells and leveraging their shared evolutionary history boosts our ability to remove noise and accurately call copy numbers. Here we developed a novel phylogenetic framework for this purpose, enabling us to jointly infer the tree and copy number states and obtain better quality profiles. When learning the sequence of CNAs that occur in tumour samples from single-cell sequencing data, the possible overlap and reoccurrences of CNAs need to be accounted for. Our tree model allows regions of the genome to be arbitrarily amplified and deleted, while controlling for model complexity. Simulations demonstrate that our approach is accurate for the challenging task of reconstructing the evolutionary history of tumours and calling copy numbers in individual cells.

For a real data example of 260 xenograft breast cancer cells, we obtained the CNA tree and inferred copy number profiles of the cells and detected the main clonal structure as well as their phylogenetic relationship. Some smaller or weaker changes over small numbers of cells, visible in the normalised data (Figure 3b) were not detected as copy number changes in the inference (Figure 3a) as they were insufficient to justify a more complex model in our framework. Adjusting and relaxing the penalisation for model complexity to detect finer changes may offer further improvements, though learning larger and more complex trees also increase the computation cost of the inference.

To learn the regions in the first place, we developed a dynamic programming approach to combine the evidence of breakpoints across all cells. Compared to methods which work on a per cell basis, our joint inference of breakpoints and then the full probabilistic model of the phylogeny provides a substantial improvement. The combination of bins into regions reduces the possible search space and speeds up the inference, but since the quality of the regions detected directly affects the downstream reconstruction, further improvements in this direction are important. In particular, once a phylogeny has been learnt based on the strongest breakpoints, the corresponding separation of cells into clones can help distinguish noise from signal in the breakpoint detection. Adding and removing breakpoints could also be incorporated as a move into the MCMC scheme itself.

For real data, where GC and mappability corrections are performed to normalise the bins, residual confounding effects still remain which can complicate the breakpoint detection and phylogenetic inference. In particular, the bin corrections depend on the underlying copy number state, indicating that future directions could consider jointly inferring the corrections along with the regions and the phylogeny, although increasing the complexity and computational cost of learning CNA trees.

Understanding and reconstructing the history of copy number events in a tumour could play a key role in predicting response to treatment, especially when resistance arises from adaptive selection of the existing clonal architecture. The combination with single-cell transcriptomics can uncover the interplay between evolutionary pressure and cellular reprogramming [29]. The phylogenetic methods developed here for large-scale, complex and overlapping events could potentially also reconstruct trajectory structures reflected in the transcriptomic profiles of single-cell RNA sequencing. Copy number reconstruction further complements multi-faceted single-cell profiling [30], for example to determine the downstream effects of tumour heterogeneity through evolutionary analyses across cohorts. Accurate copy number calling at the single-cell level, enabled through SCICoNE's joint inference of the evolutionary history and copy number profiles, will enhance single-cell analysis for cancer biology.

Methods

Binning and read count correction Current protocols for copy number detection at single-cell resolution typically employ shallow whole-genome sequencing ($\lesssim$0.1x coverage per cell) [16, 17] which prohibits coverage-based copy number calling at the level of individual loci. Instead one partitions the reference genome into equal sized bins (20–100kb) and counts the reads per bin instead of per locus. The raw read count of a bin, is not only determined by the bin's underlying copy number state, but also by its mappability and GC content. To reduce the bias introduced by these confounders, SCICoNE uses read counts (per bin and per cell) that have been corrected for both effects [20, 21]. With these confounders removed, we now assume that the probability of reads falling into each bin is proportional to the bin's copy number state,

$$p_i^j \propto c_i^j \tag{1}$$

where p_i^j is the probability of a read from cell j falling into bin i, and c_i^j is the copy number state for that bin in that cell.

Breakpoint detection to define copy number regions As copy number changes often affect regions much larger than a typical bin size, we collate neighbouring bins into genomic regions with the same copy number state. To collate the bins into genomic regions, we first detect breakpoints as bin boundaries where the read depth changes across subsets of cells. For this purpose we developed a dynamic programming approach that combines evidence across cells to call the breakpoints (Appendix A). The detection compares a likelihood-based model of a step change in copy number at each bin to a constant copy number model for each cell and then combines the signal over all cells. Bins with the strongest combined signal relative to a noise threshold are classified as breakpoints.

Once the breakpoints have been determined, we collate bins between consecutive breakpoints into regions. For each cell, we sum the counts in all bins belonging to each region to arrive at a count matrix D with entries D_{jk} for each cell j and each region R_k.

The probability of a read falling into region R_k is proportional to the copy number state and the size of the region

$$p_k^j \propto c_k^j r_k \,, \qquad r_k = \text{number of bins in region } R_k \tag{2}$$

where p_k^j is the probability of a read from cell j falling into region k with size r_k, and c_k^j is the copy number state for that region in that cell.

CNA trees For clinical applications and copy number calling, we work with the history of copy number events represented as a CNA tree (Figure 1d). This is in analogy to the mutation trees of SCITE [22], where the tree nodes now encapsulate events corresponding to amplifications or deletions of the regions. The nodes in the tree can have arbitrary degree. We index the event nodes $1, \dots, n$ and additionally label them with the corresponding CNAs. All CNA events are stored in the vector V, such that V_i is the collection of CNAs of vertex i. For the example of Figure 1d, we have $V = (+R_1 + R_2,\, +R_2 + R_3,\, -R_1,\, -R_4,\, +R_2)$. The tree structure T we can store as a parent vector $T = (0, 1, 2, 2, 1)$ where 0 represents the root.

If cell j is attached to event node σ_j we can read off the expected copy number state for each region by tracing all events back to the root for a given tree structure and event vector. In the example of Figure 1a, the attachment vector is $\boldsymbol{\sigma} = (3, 3, 4, 1, 5)$. We denote the expected copy number state of cell j for a given attachment point σ_j as $c_k^j(T, V, \sigma_j)$. Then we have for the probability of the reads of cell j falling into region k

$$p_k^j(T, V, \sigma_j) = \frac{c_k^j(T, V, \sigma_j) r_k}{Z^j} \,, \qquad Z^j = \sum_k c_k^j(T, V, \sigma_j) r_k \tag{3}$$

If a region is entirely deleted so that the copy number state and probability is 0, then we would not expect to see any reads in that region. However, due to mapping errors there may still be some residual reads in that region. To account for this possibility, we instead set the minimum copy number state to $0 < \eta \ll 1$.

Likelihood To assess how well a CNA tree fits our read count data, we define a likelihood model as follows. The data matrix entry D_{jk} stores the number of corrected counts cell j has in region k, with total reads for that cell of $N^j = \sum_k D_{jk}$. Given the probabilities of reads landing in each region, we model the data with a Dirichlet-Multinomial distribution to account for overdispersion

$$D_j \sim \text{Dirichlet-Multinomial}(\nu \boldsymbol{p}^j(T, V, \sigma_j) Z^j, N^j) \tag{4}$$

where ν is the concentration parameter (inverse of the overdispersion). Using Equation (3), the likelihood contribution from cell j is therefore

$$L_j(T, V, \sigma_j) = \frac{\Gamma\left[\nu Z^j\right]}{\Gamma\left[N^j + \nu Z^j\right]} \prod_k \frac{\Gamma\left[D_{jk} + \nu c_k^j(T, V, \sigma_j) r_k\right]}{\Gamma\left[\nu c_k^j(T, V, \sigma_j) r_k\right]}$$

In the large ν limit, the model simplifies to the multinomial distribution

$$L_j(T, V, \sigma_j) = \left[Z^j\right]^{-N^j} \prod_k \left[c_k^j(T, V, \sigma_j)\right]^{D_{jk}} \tag{5}$$

We can compute this likelihood for all cells and all possible attachment points efficiently in total time $O(mn)$ where m is the number of cells and n the number of event nodes. This time complexity is achieved with a tree traversal. During the tree traversal, at each step only the regions in one event node change their state and we only update a limited number of terms in the product. All possible attachment points of a cell can therefore be computed in linear time.

For numerical accuracy, we compute the log-likelihood of all attachment points for each cell relative to the root. At the root, with a given constant ploidy c, we have the score per cell of

$$L_j(T, V, 0) = \frac{\Gamma[\nu Z]}{\Gamma[N^j + \nu Z]} \prod_k \frac{\Gamma[D_{jk} + \nu c r_k]}{\Gamma[\nu c r_k]}, \qquad Z = c \sum_k r_k \tag{6}$$

whereas for large ν the ploidy constant cancels and we simply have

$$L_j(T, V, 0) = \left[\sum_k r_k\right]^{-N^j} \prod_k [r_k]^{D_{jk}} \tag{7}$$

Since the root is common across all tree models, this contribution only varies if ν is changed and is only recomputed in that case. The root does not need to have constant ploidy; for example the sex chromosomes can be set accordingly.

Posterior tree probability With the likelihoods of each cell for each attachment point, we can directly compute the marginal likelihood when we average over all possible attachment points of each cell in the tree

$$P(D \mid T, V) = \frac{1}{(n+1)^m} \prod_{j=1}^{m} \sum_{\sigma_j=0}^{n} L_j(T, V, \sigma_j) \tag{8}$$

with a uniform prior on the attachment points. With a non-informative prior on the tree size, and then a uniform prior also on the trees, the posterior becomes

$$P(T, V \mid D) \propto \frac{P(V \mid T) P'(T)}{(n+1)^{n-1+m}} \prod_{j=1}^{m} \sum_{\sigma_j=0}^{n} L_j(T, V, \sigma_j) \tag{9}$$

where $P'(T)$ is a penalisation term needed to account for combinatorial effects possible for larger trees which we detail in Appendix B, and where we discuss the prior on the event vector in Appendix C.

Since we marginalise out the cell attachments, we focus on the CNA tree and build a scheme to find the tree and vector with the highest posterior score. Not all combinations of trees and event vectors are meaningful. After a copy number state of 0 is reached at any point in the tree, the affected region cannot be regained in any descendant nodes. Also, a tree with a placement of events such that it recreates the exact same genotypes repeatedly in different parts of the tree is redundant, as a simpler model using a smaller tree would fit the data equally well. Biologically such convergent evolution may occur, but as it cannot be distinguished by the data itself, we make the assumption of the more parsimonious model. To exclude the possibilities above we assign any trees with forbidden events a posterior score of 0.

As an alternative to marginalisation, we can also place cells at their maximal attachment position and define the score

$$S(T, V \mid D) = \frac{P(V \mid T)}{(n+1)^{n-1}} \prod_{j=1}^{m} \max_{\sigma_j} L_j(T, V, \sigma_j) \tag{10}$$

which removes the need for correcting for combinatorial effects via $P'(T)$. To distinguish between these two alternatives, we denote inference using the marginalised score, Equation (9), with the term ‘sum’ and inference using the maximised score, Equation (10), with the term ‘max’ in the main text (for example in the simulation plots of Figure 4).

MCMC To sample trees proportionally to $P(T, V \mid D)$ we build an MCMC scheme to move in the space of trees and event vectors. We detail the moves in Appendix D starting with the simple moves of *prune and reattach* and *label swap* employed in SCITE [22] which work for a fixed tree size and fixed set of events at the nodes. We additionally weight the move proposals to account for their computational costs. To change the event vector, we introduce an *add or remove events* move, while to change the tree size we developed the *add or remove node* and *condense or split nodes* moves to cover the full discrete space. To aid moving in such a complex space and finding simpler trees which generate the same set of possible genotypes we also introduce a *genotype preserving prune and reattach* move. Since the set of genotypes is preserved, only the event vector prior needs to be recomputed making this move computationally very efficient, so we score and sample from the whole neighbourhood. For the overdispersion determined by the concentration parameter ν we run a random walk (in log space). For the complete MCMC scheme, at each iteration we first sample a move type with a fixed probability for the six different moves. Each move is equally likely to be chosen apart from the *genotype preserving prune and reattach* which has a relative weight of 0.4.

Maximal search We utilise the MCMC scheme to search for the tree with the highest posterior score, keeping track of the maximally scoring tree encountered. From the best scoring tree discovered, we can compute the best attachment of each cell and hence their corresponding predicted copy number profiles.

To aid finding the best scoring tree and event vectors, we work stepwise. First we cluster the data (using PhenoGraph) and construct the average read counts per cluster. The tree likelihood computation only involves running over each cluster (weighted by the cluster size) rather than each cell, giving a large computational speed up. We run 10 different chains and check for robustness that at least half the chains return trees with similar high score (the log posterior is within 5 of the maximum). Non-robust runs are repeated starting from the highest scoring tree found so far. Once the tree on the clustered data has been determined, we run the scheme on the full data, starting from the cluster tree and with fixed overdispersion set to the value learned for the cluster tree.

Root state Instead of assuming a diploid state at the root, it is easy to start with any other state, for example tetraploid corresponding to a whole genome duplication. Although the average ploidy is not well defined for read depth data, because CNAs are integer, the differences induced can help determine the underlying states. Under the presence of further CNAs, the model comparison of the tree with a diploid root against that with a tetraploid root would potentially allow identification of genome doubling relevant for tumour progression and prognosis [31].

Simulation settings For the simulation we consider a genome consisting of 10,000 bins and different numbers of reads per cell with an average of 2, 4 or 8 reads per bin. This is a relatively low read depth per bin, chosen to make the inference more challenging and comparable to the data from the 10x Genomics protocol with their default bin size of 20kb.

For trees with size $n = 20$, we partitioned the bins into 40 or 80 regions. At each node in the tree we sampled the number of regions involved with a Poisson distribution with parameter $\lambda = 0.1$ and added 1, while for each region we sampled the number of copies with a Poisson distribution with parameter $\lambda = 0.2$ and again added 1 and sampled the sign uniformly. Trees that are not biologically meaningful are rejected and resampled. For each setting, we sampled 40 trees.

For each tree, we sample 400 cells, in line with the typical number of cells targeted with the 10x Genomics protocol for a single sequencing run, and sample their attachment point in the tree uniformly. This provides the true copy number profile of each cell, from which the reads per bin were sampled with a Dirichlet-multinomial with concentration $\nu = 4$.

For inferring the copy number profiles with our MCMC scheme, we first detected the breakpoints automatically (Appendix A with $\nu = 1$) to generate the regions. For our MCMC scheme we set $\eta = 10^{-4}$ and ran chains of length 4,000n, repeatedly until robust results were obtained, to find first the trees on the clustered data and then on the full data. We find the best trees both for the posterior score from the summation of Equation (9) and for the maximisation of Equation (10).

For comparison we include HMMcopy [32, 26] which runs a hidden Markov model on each cell to call the breakpoints and the copy number states, and for which we used the default parameters. HMMcopy returns the inferred copy number states per bin for each cell individually. Ginkgo [21] offers single-cell copy number calling, but is only available as a web-tool unsuitable for multiple simulations so we do not compare against it. Instead we also include a diploid profile and two clustering methods. For the clustering we use both PhenoGraph [27] and hierarchical clustering (with the number of clusters chosen according to the Calinski-Harabasz index) and assigned all cells in each cluster the average read counts of that cluster. For each cluster and its associated cells, we then set the median counts per cell to 2 (by doubling and dividing by the median) assuming that diploid is the most common state and rounded the copy number states to the nearest integer. In the comparison we compute Δ as the root mean squared difference over all cell and bins, between the true copy number profiles and the inferred values.

Code availability SCICoNE has been implemented in C++ and is freely available under a GNU General Public License v3.0 license at https://github.com/cbg-ethz/SCICoNE.

Acknowledgements J.K. was supported by ERC Synergy Grant 609883 (http://erc.europa.eu/), P.F. by SNSF Grant 310030_179518 (http://www.snf.ch) and K.J. was supported by SystemsX.ch RTD Grant 2013/150 (http://www.systemsx.ch/).

References

[1] Yates, L. R. & Campbell, P. J. Evolution of the cancer genome. *Nature Reviews Genetics* **13**, 795–806 (2012).

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

[3] Greaves, M. & Maley, C. C. Clonal evolution in cancer. *Nature* **481**, 306–313 (2012).

[4] Burrell, R. A. & Swanton, C. Re-evaluating clonal dominance in cancer evolution. *Trends in Cancer* **2**, 263–276 (2016).

[5] Maley, C. C. *et al.* Genetic clonal diversity predicts progression to esophageal adenocarcinoma. *Nature Genetics* **38**, 468–73 (2006).

[6] Merlo, L. M. *et al.* A comprehensive survey of clonal diversity measures in Barrett's esophagus as biomarkers of progression to esophageal adenocarcinoma. *Cancer Prevention Research* **3**, 1388–1397 (2010).

[7] Marusyk, A. & Polyak, K. Tumor heterogeneity: causes and consequences. *BBA Reviews on Cancer* **1805**, 105–117 (2010).

[8] Burrell, R. A., McGranahan, N., Bartek, J. & Swanton, C. The causes and consequences of genetic heterogeneity in cancer evolution. *Nature* **501**, 338–345 (2013).

[9] McGranahan, N. & Swanton, C. Biological and therapeutic impact of intratumor heterogeneity in cancer evolution. *Cancer Cell* **27**, 15–26 (2015).

[10] Allison, K. H. & Sledge, G. W. Heterogeneity and cancer. *Oncology* **28**, 772–8 (2014).

[11] Wang, Y. & Navin, N. E. Advances and applications of single-cell sequencing technologies. *Molecular Cell* **58**, 598–609 (2015).

[12] Gawad, C., Koh, W. & Quake, S. R. Single-cell genome sequencing: current state of the science. *Nature Reviews Genetics* **17**, 175–188 (2016).

[13] Kuipers, J., Jahn, K. & Beerenwinkel, N. Advances in understanding tumour evolution through single-cell sequencing. *BBA Reviews on Cancer* **1867**, 127–138 (2017).

[14] Zafar, H., Navin, N., Nakhleh, L. & Chen, K. Computational approaches for inferring tumor evolution from single-cell genomic data. *Current Opinion in Systems Biology* 16–25 (2018).

[15] Navin, N. *et al.* Tumour evolution inferred by single-cell sequencing. *Nature* **472**, 90–94 (2011).

[16] Zahn, H. *et al.* Scalable whole-genome single-cell library preparation without preamplification. *Nature Methods* **14**, 167 (2017).

[17] `https://www.10xgenomics.com/solutions/single-cell-cnv/`.

[18] Singer, J., Kuipers, J., Jahn, K. & Beerenwinkel, N. Single-cell mutation identification via phylogenetic inference. *Nature Communications* **9**, 5144 (2018).

[19] Kuipers, J., Jahn, K., Raphael, B. J. & Beerenwinkel, N. Single-cell sequencing data reveal widespread recurrence and loss of mutational hits in the life histories of tumors. *Genome Research* **27**, 1885–1894 (2017).

[20] Baslan, T. *et al.* Genome-wide copy number analysis of single cells. *Nature Protocols* **7**, 1024 (2012).

[21] Garvin, T. *et al.* Interactive analysis and assessment of single-cell copy-number variations. *Nature Methods* **12**, 1058 (2015).

[22] Jahn, K., Kuipers, J. & Beerenwinkel, N. Tree inference for single-cell data. *Genome Biology* **17**, 86 (2016).

[23] Sondka, Z. *et al.* The COSMIC Cancer Gene Census: describing genetic dysfunction across all human cancers. *Nature Reviews Cancer* **18**, 696–705 (2018).

[24] Campbell, K. R. *et al.* clonealign: statistical integration of independent single-cell RNA and DNA sequencing data from human cancers. *Genome Biology* **20**, 54 (2019).

[25] Tirosh, I. *et al.* Dissecting the multicellular ecosystem of metastatic melanoma by single-cell RNA-seq. *Science* **352**, 189–196 (2016).

[26] Lai, D., Ha, G. & S, S. HMMcopy: Copy number prediction with correction for GC and mappability bias for HTS data (2016). R package version 1.22.0.

[27] Levine, J. H. *et al.* Data-driven phenotypic dissection of AML reveals progenitor-like cells that correlate with prognosis. *Cell* **162**, 184–197 (2015).

[28] Fan, X., Edrisi, M., Navin, N. & Nakhleh, L. Methods for copy number aberration detection from single-cell DNA sequencing data (2019). BioRxiv:696179.

[29] Kim, C. *et al.* Chemoresistance evolution in triple-negative breast cancer delineated by single-cell sequencing. *Cell* **173**, 879–893 (2018).

[30] Irmisch, A. *et al.* The Tumor Profiler Study: Integrated, multi-omic, functional tumor profiling for clinical decision support. *medRxiv:2020.02.13.20017921* (2020).

[31] Bielski, C. M. *et al.* Genome doubling shapes the evolution and prognosis of advanced cancers. *Nature Genetics* **50**, 1189 (2018).

[32] Ha, G. *et al.* Integrative analysis of genome-wide loss of heterozygosity and monoallelic expression at nucleotide resolution reveals disrupted pathways in triple-negative breast cancer. *Genome Research* **22**, 1995–2007 (2012).

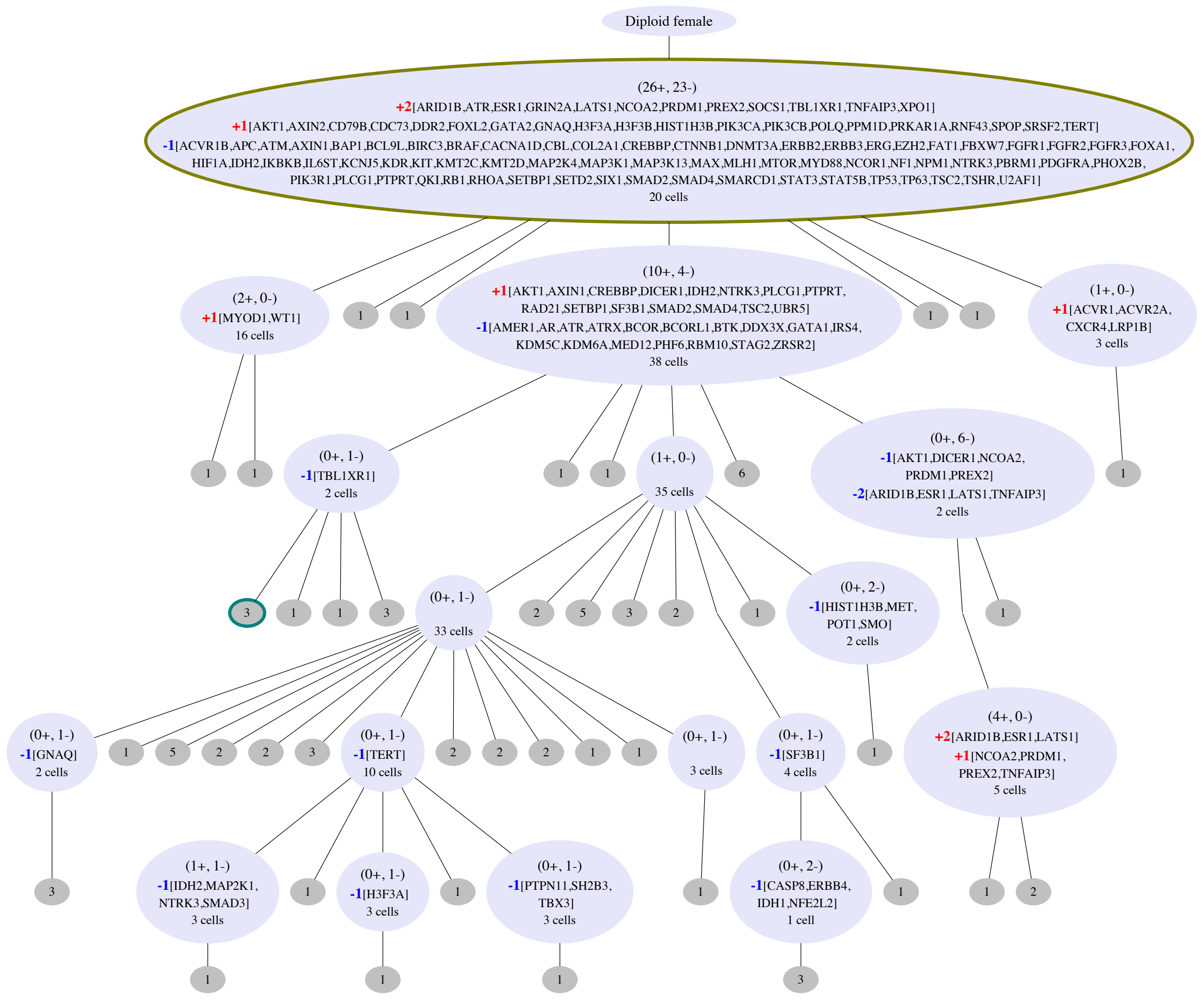
Diploid female
(26+, 23-)
+2[ARID1B,ATR,ESR1,GRIN2A,LATS1,NCOA2,PRDM1,PREX2,SOCS1,TBL1XR1,TNFAIP3,XPO1]
+1[AKT1,AXIN2,CD79B,CDC73,DDR2,FOXL2,GATA2,GNAQ,H3F3A,H3F3B,HIST1H3B,PIK3CA,PIK3CB,POLQ,PPM1D,PRKAR1A,RNF43,SPOP,SRSF2,TERT]
-1[ACVR1B,APC,ATM,AXIN1,BAP1,BCL9L,BIRC3,BRAF,CACNA1D,CBL,COL2A1,CREBBP,CTNNB1,DNMT3A,ERBB2,ERBB3,ERG,EZH2,FAT1,FBXW7,FGFR1,FGFR2,FGFR3,FOXA1, HIF1A,IDH2,IKBKB,IL6ST,KCNJ5,KDR,KIT,KMT2C,KMT2D,MAP2K4,MAP3K1,MAP3K13,MAX,MLH1,MTOR,MYD88,NCOR1,NF1,NPM1,NTRK3,PBRM1,PDGFRA,PHOX2B, PIK3R1,PLCG1,PTPRT,QKI,RB1,RHOA,SETBP1,SETD2,SIX1,SMAD2,SMAD4,SMARCD1,STAT3,STAT5B,TP53,TP63,TSC2,TSHR,U2AF1]
20 cells
(2+, 0-)
+1[MYOD1,WT1]
16 cells
1
1
(10+, 4-)
+1[AKT1,AXIN1,CREBBP,DICER1,IDH2,NTRK3,PLCG1,PTPRT, RAD21,SETBP1,SF3B1,SMAD2,SMAD4,TSC2,UBR5]
-1[AMER1,AR,ATR,ATRX,BCOR,BCORL1,BTK,DDX3X,GATA1,IRS4, KDM5C,KDM6A,MED12,PHF6,RBM10,STAG2,ZRSR2]
38 cells
1
1
(1+, 0-)
+1[ACVR1,ACVR2A, CXCR4,LRP1B]
3 cells
1
1
(0+, 1-)
-1[TBL1XR1]
2 cells
1
1
(1+, 0-)
35 cells
6
(0+, 6-)
-1[AKT1,DICER1,NCOA2, PRDM1,PREX2]
-2[ARID1B,ESR1,LATS1,TNFAIP3]
2 cells
1
3
1
1
3
(0+, 1-)
33 cells
2
5
3
2
1
(0+, 2-)
-1[HIST1H3B,MET, POT1,SMO]
2 cells
1
(0+, 1-)
-1[GNAQ]
2 cells
1
5
2
2
3
(0+, 1-)
-1[TERT]
10 cells
2
2
2
1
1
(0+, 1-)
3 cells
(0+, 1-)
-1[SF3B1]
4 cells
1
(4+, 0-)
+2[ARID1B,ESR1,LATS1]
+1[NCOA2,PRDM1, PREX2,TNFAIP3]
5 cells
3
(1+, 1-)
-1[IDH2,MAP2K1, NTRK3,SMAD3]
3 cells
1
(0+, 1-)
-1[H3F3A]
3 cells
1
(0+, 1-)
-1[PTPN11,SH2B3, TBX3]
3 cells
1
(0+, 2-)
-1[CASP8,ERBB4, IDH1,NFE2L2]
1 cell
1
1
2
1
1
1
3

Figure A1: **Inferred tree for 260 cells from a breast xenograft [16].** An expanded version of Figure 2 showing all the genes from the COSMIC Cancer Gene Census [23], indicating where and by how much they are amplified or deleted

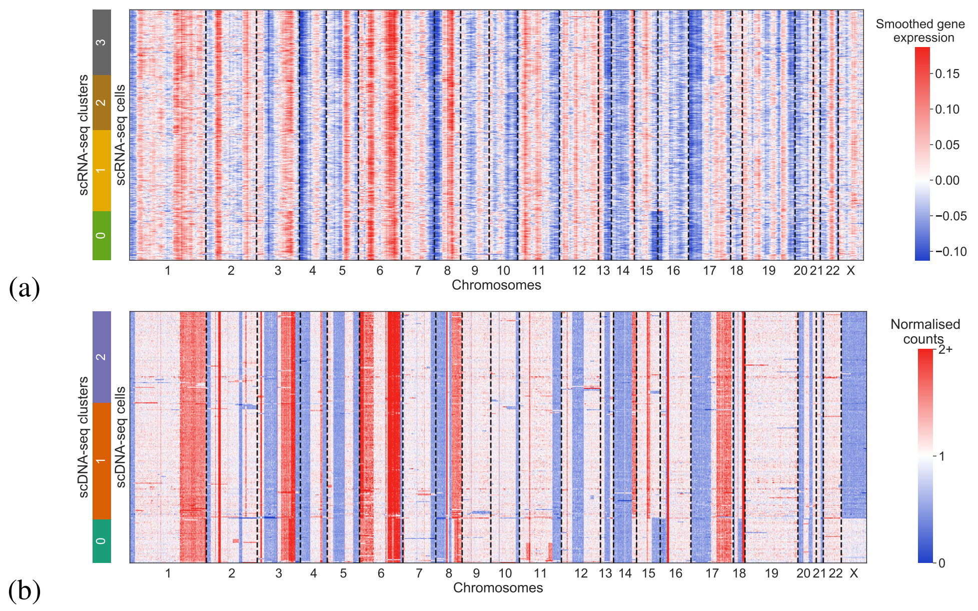

Figure A2: **RNA and DNA levels for breast xenograft cells.** (a) The smoothed expression profiles of 1152 cells from single-cell RNA sequencing [24]. (b) The normalised counts per gene for the expressed genes for 260 cells from single-cell DNA sequencing [16] are displayed for comparison.

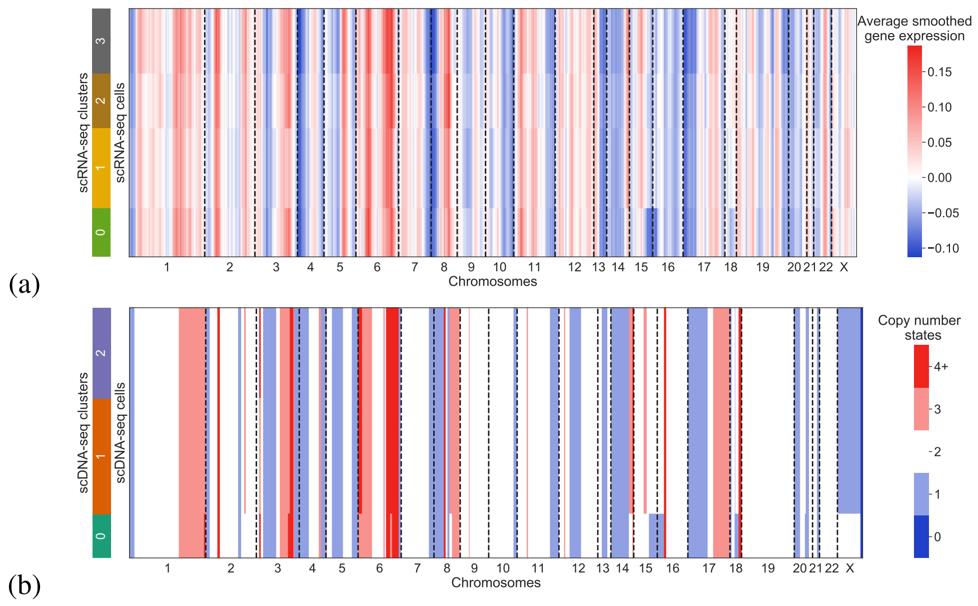

Figure A3: **RNA level and DNA copy number for breast xenograft clusters.** (a) The smoothed expression profiles of the cells in Figure A2a are clustered and the average profile of the cluster displayed. (b) The inferred copy number profiles of clones built from clustering the single-cell DNA sequencing data and learning a tree using SCICoNE (displayed at the gene level to match the normalised counts in Figure A2b). The inferred copy number profiles of the full data are in Figure 3.

Appendices

A Breakpoint detection

For a given bin position ρ, we wish to test if there is a change in read counts in the next bin, across cells. In particular we fix a window size ω, consider the bins from $\rho-\omega+1$ to $\rho+\omega$ and look at the evidence for a breakpoint, and hence a copy number change, after bin ρ.

A.1 Evidence per cell For a given cell j we model the read counts in bin i, z_i^j with a negative Binomial to account for overdispersion. We parametrise in terms of the mean λ and overdispersion parameter ν with a mass function of

$$P(X=z)=\frac{\Gamma(z+\nu)}{z!\Gamma(\nu)}\left(\frac{\lambda}{\lambda+\nu}\right)^{z}\left(\frac{\nu}{\lambda+\nu}\right)^{\nu} \tag{11}$$

If there is no copy number change after bin ρ we would have equal expected counts across all bins in the window, so that the log-likelihood of the observed counts is

$$l(z^j;\lambda,\nu)=\left(\sum_{i=\rho-\omega+1}^{\rho+\omega} z_i^j\right)\left[\log(\lambda)-\log(\lambda+\nu)\right]-2\omega\nu\log(\lambda+\nu) \tag{12}$$

where we ignore constant terms that do not depend on λ. The maximum likelihood, for fixed ν occurs at

$$\lambda^*=\frac{\sum\limits_{i=\rho-\omega+1}^{\rho+\omega} z_i^j}{2\omega} \tag{13}$$

To allow for noise giving unbalanced counts on either side of ρ even without a breakpoint, we fit a linear model for the expected counts

$$l(z^j;\alpha,\beta,\nu)=\sum_{i=\rho-\omega+1}^{\rho+\omega} z_i^j\left[\log(\lambda_i)-\log(\lambda_i+\nu)\right]-2\omega\nu\log(\lambda_i+\nu)\,,\qquad \lambda_i=\alpha+\beta(i-\rho) \tag{14}$$

which we maximise over α and β:

$$(\alpha^*,\beta^*)=\underset{\alpha,\beta}{\operatorname{argmax}}\; l(z^j;\alpha,\beta,\nu) \tag{15}$$

To avoid fitting real copy number changes with the linear model, we bound the slope so the relative change across ρ is less than a quarter by restricting $|\beta|<\frac{1}{4\omega}$.

If there is a breakpoint, we would expect to have different average counts on each side which we model with two mean parameters leading to a log-likelihood of

$$\begin{aligned} l(z^j;\lambda_{\mathrm{L}},\lambda_{\mathrm{R}},\nu) \;=\;& \left(\sum_{i=\rho-\omega+1}^{\rho} z_i^j\right)\left[\log(\lambda_{\mathrm{L}})-\log(\lambda_{\mathrm{L}}+\nu)\right]-\omega\nu\log(\lambda_{\mathrm{L}}+\nu) \\ &+\left(\sum_{i=\rho-\omega+1}^{\rho} z_i^j\right)\left[\log(\lambda_{\mathrm{R}})-\log(\lambda_{\mathrm{R}}+\nu)\right]-\omega\nu\log(\lambda_{\mathrm{R}}+\nu) \end{aligned} \tag{16}$$

This is maximised by the average counts of each side of the potential breakpoint: For robustness and to uncover changes in count level that spread over many bins, we use the robust mean

$$\lambda^*_{\mathrm{L}} = \mathrm{mean}_{\mathrm{r}}\left(z^j_i, i = \rho - \omega + 1, \dots, \rho\right), \qquad \lambda^*_{\mathrm{R}} = \mathrm{mean}_{\mathrm{r}}\left(z^j_i, i = \rho + 1, \dots, \rho + \omega\right) \tag{17}$$

We further enforce a minimum difference between the λ parameters on each side of ρ so that the relative change in copy number is more than a quarter:

$$\frac{|\lambda^*_{\mathrm{L}} - \lambda^*_{\mathrm{R}}|}{\lambda^*} > \frac{1}{4} \tag{18}$$

If the inequality is not satisfied by $\lambda^*_{\mathrm{L,R}}$, we rescale their differences from λ^*.

Finally we compute the difference in maximum log-likelihoods of the two models. We compute this difference per cell for each breakpoint

$$A^j_\rho = l(z^j; \lambda^*_{\mathrm{L}}, \lambda^*_{\mathrm{R}}, \nu) - l(z^j; \alpha^*, \beta^*, \nu) \tag{19}$$

For each cell, the likelihood ratios over bins A^j_ρ may exhibit noise from the underlying count data. However we would expect breakpoints with a copy number change to provide a signal across the bins with a width of the window size ω. To amplify these signals and filter out noise, we perform a low-pass Gaussian filter (with half-gain cut-off frequency corresponding to twice the width ω) with a Fourier transform and replace the A^j_ρ signal per cell by the smoothed version.

A.2 Combining cells Next we compute the logarithm of the combined evidence that the breakpoint occurred in any k of the m cells with the average

$$\Sigma^k_\rho = \log\left[\frac{1}{\binom{m}{k}} \sum_{\boldsymbol{J}\subset\{1,\dots,m\}}^{|\boldsymbol{J}|=k} \mathrm{e}^{\sum_{j\in\boldsymbol{J}} A^j_\rho}\right] \tag{20}$$

This can be computed efficiently using dynamic programming (Algorithm 1). We further combine the evidence for the breakpoint in k of the m cells with the prior of an event affecting k cells in a random binary cell lineage tree [18]:

$$P(k) = \begin{cases} (1-\mu) & k = 0 \\ \mu \frac{\binom{m}{k}^2}{(2k-1)\binom{2m}{2k}} & 1 \leq k \leq m \end{cases} \tag{21}$$

where μ is the prior probability of an event occurring. The posterior probability of the event occurring in k cells is then proportional to $P(k)\mathrm{e}^{\Sigma^k_\rho}$. To rank the breakpoints, we consider the posterior probability of the breakpoint occurring in k^* or more cells

$$P_\rho = \frac{\sum_{k=k^*}^m P(k)\mathrm{e}^{\Sigma^k_\rho}}{\sum_{k=0}^m P(k)\mathrm{e}^{\Sigma^k_\rho}} = 1 - \frac{\sum_{k=0}^{k^*-1} P(k)\mathrm{e}^{\Sigma^k_\rho}}{\sum_{k=0}^m P(k)\mathrm{e}^{\Sigma^k_\rho}} \tag{22}$$

and compute

$$S_\rho = -\log\left[1 - P_\rho\right] = \log\left[\sum_{k=0}^m P(k)\mathrm{e}^{\Sigma^k_\rho}\right] - \log\left[\sum_{k=0}^{k^*-1} P(k)\mathrm{e}^{\Sigma^k_\rho}\right] \tag{23}$$

Algorithm 1 Obtain the sum of breakpoint evidence over all cell subsets

```
Input The vector of penalised relative likelihoods, $A_\rho^j$, $j = 1, \dots, m$
Initialise a vector $v_l$, $l = 0, \dots, m$:
$v_0 = 0$
$v_1 = A_\rho^1$
for $j = 2$ to $m$ do                                          ▷ cells
    for $k = 1$ to $j - 1$ do                                  ▷ size of subset
        $v'_k = \log.\text{add}(v_{k-1} + A_\rho^j, v_k)$      ▷ $\log.\text{add}(a, b) = \max(a, b) + \log\left[1 + \mathrm{e}^{-|a-b|}\right]$
    end for
    $v'_j = v_{j-1} + A_\rho^j$
    Update vector $v_k = v'_k$, $k = 1, \dots, j$
end for
$\Sigma_\rho^k = v_k - \log\binom{m}{k}$, $k = 0, \dots, m$
return Vector of summed combinations: $\Sigma_\rho^k$, $k = 0, \dots, m$
```

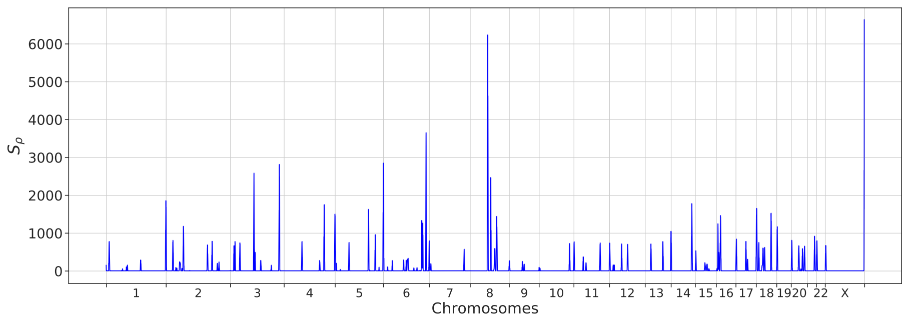

Figure A4: **Breakpoint detection** For each bin we combine the evidence for a breakpoint across the 260 cells from a breast xenograft [16], to arrive at S_ρ. Peaks corresponds to putative breakpoints.

A.3 Peak detection

Plotting S_ρ across the genome we find peaks, with a width of ω when ρ lines up with a breakpoint in a number of cells (Figure A4). Since the peaks possess very different heights, we perform a further log transform, $\tilde{S}_\rho = \log S_\rho$. Since the local median value of $\tilde{S}_\rho$ may depend on the underlying copy number state, we find the breakpoints iteratively. First we divide $\tilde{S}_\rho$ by its median value and find the highest value. If it is above the threshold, we split $\tilde{S}_\rho$ into two parts, exclude a window of size ω from either side and divide the remaining parts by their median. We repeatedly find the maximum value, split the vector excluding a window around the maximum, and divide the regions between the current breakpoints by their median until the maximum in those regions is below the threshold. The threshold has a default value of 3 times the distance between the median and third quartile of the bins not around the current set of breakpoints. We have a default window size of 20.

B Tree penalisation for combinatorial effects

For any tree, assume that the largest cluster of cells of size m_c attach best to the same node k and poorly elsewhere. The likelihood contribution of these cells includes the term

$$\prod_{j=1}^{m_c} \sum_{\sigma_j=0}^{n} L_j(T, V, \sigma_j) \approx \prod_{j=1}^{m_c} L_j(T, V, k) \tag{24}$$

where in the sum we assume that the other attachment points are negligible for the sum. Now imagine we add another node k' to the tree with a very similar genotype to the best attachment point of the cells, and hence very similar likelihoods for the cells. The likelihood term now becomes

$$\prod_{j=1}^{m_c}\sum_{\sigma_j=0}^{n+1} L_j(T',V',\sigma_j) \approx \prod_{j=1}^{m_c} L_j(T,V,k) + L_j(T',V',k') \approx \prod_{j=1}^{m_c} 2L_j(T,V,k) = 2^{m_c}\prod_{j=1}^{m_c} L_j(T,V,k) \tag{25}$$

so that it can be increased by a factor of up to 2^{m_c}. The likelihood contribution can be made arbitrarily large by adding further dummy nodes with similar genotypes. To counteract this effect, we can penalise trees with the factor

$$P'(T) = \frac{1}{2^{nm_c}} \tag{26}$$

so that if we add an additional node and increase n by 1 we obtain an additional factor of 2^{-m_c}.

C Prior on the event vector

For the prior on the event vector V, we simply consider that each amplification or deletion may be selected among the K regions with a choice of sign, leading to a factor of $\frac{1}{2K}$. At each node, we consider the amplification of contiguous regions as a single amplification event, and likewise for deletions. To compute this number, let us list as $\boldsymbol{v}^i$ a vector of the number of copy number events for each region occurring at node i in the tree. For the example in Figure 1c with event vector $V = (+R_1 + R_2,\ +R_2 + R_3,\ -R_1,\ -R_4,\ +R_2)$ we would have $\boldsymbol{v}^2 = (0, +1, +1, 0, 0)$ over the $K = 5$ regions. To count the number of contiguous amplifications and deletions we can simply count when the copy number profile increases as

$$|v^i| = \sum_{k=1}^{K+1} \left(v_k^i - v_{k-1}^i\right) I\left(v_k^i - v_{k-1}^i > 0\right), \qquad v_0^i = v_{K+1}^i = 0 \tag{27}$$

where I is the indicator function. The prior probability for each node is then $\frac{1}{(2K)^{|v^i|}}$.

For the full event vector prior, we may freely permute the ordering of the event vector, along with the corresponding numbering of the event nodes in the tree. We therefore divide by the overcounting factor of $n!$ to arrive at

$$P(V \mid T) = \frac{1}{n!}\prod_{i=1}^{n}\frac{1}{(2K)^{|v^i|}} \tag{28}$$

Note that although the overcounting factor assumes that the elements of the event vector are distinct, we keep the same value in all cases for simplicity.

As a further penalisation, whenever an event has the opposite sign to the cumulative copy number state of the parent node, we subtract α from the log-likelihood, for which we have the default value $\alpha = 10$.

D MCMC moves

Prune and reattach The most basic move to change trees with a fixed event vector is *prune and reattach*. From the current tree T, we propose a new tree T' by uniformly sampling a node (except the root), detaching it and its descendant subtree and then uniformly sampling a new parent from the remainder of the tree including the root. An example of this proposal move is depicted in Figure A5. After the new tree has been proposed, we simply accept the move with probability

$$\rho = \min\left\{1, \frac{P(T',V \mid D)Q(T \mid T')}{P(T,V \mid D)Q(T' \mid T)}\right\} \tag{29}$$

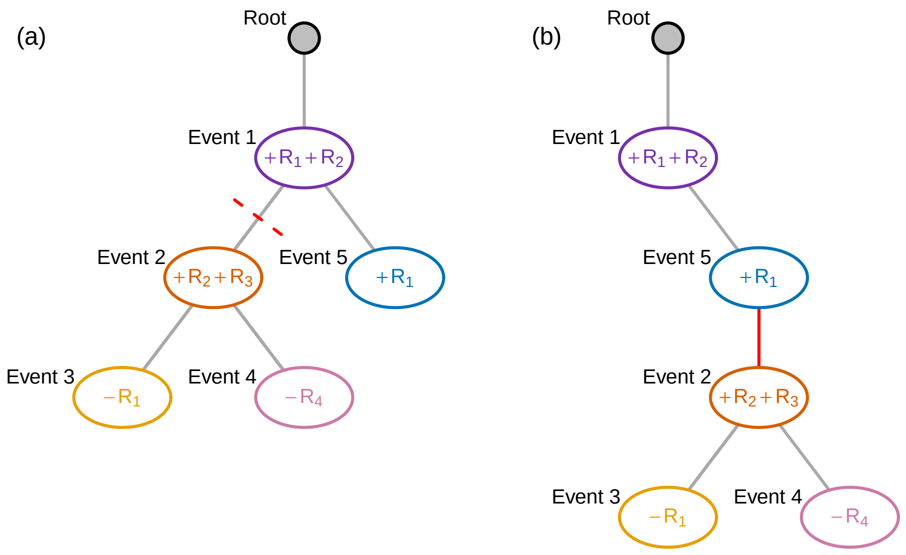

Figure A5: **Prune and reattach.** From the event tree in (a), we propose the new tree in (b) with the *prune and reattach* move by sampling a node (event node 2), detaching it from the tree and sampling a new parent (node 5).

where $Q(T' \mid T)$ is the probability of proposing tree T' when currently at tree T in the chain. Since the move is symmetric, $Q(T' \mid T) = Q(T \mid T')$, the acceptance ratio simplifies to

$$\rho = \min\left\{1, \frac{P(T', V \mid D)}{P(T, V \mid D)}\right\} \tag{30}$$

Since only the detached subtree needs to be rescored, it is cheaper to move smaller subtrees and more of such moves can be processed for the same computational cost. To try and speed up convergence of the MCMC scheme, we can preferentially sample nodes to move which have fewer descendants. We denote by $d_i(T)$ the size of the subtree of T starting with node i (the number of descendants plus 1). Instead of uniformly sampling nodes, we can sample proportionally to the $d_i(T)^{-1}$ to balance the computational cost of different moves.

If we define

$$\zeta(T) = \sum_{i=1}^{n} d_i(T)^{-1} \tag{31}$$

as a normalising constant then the transition probabilities in moving from tree T to T' where node i is selected to detach are then

$$Q(T' \mid T) = \frac{d_i(T)^{-1}}{\zeta(T)}, \qquad Q(T \mid T') = \frac{d_i(T')^{-1}}{\zeta(T')}, \qquad \frac{Q(T \mid T')}{Q(T' \mid T)} = \frac{\zeta(T)}{\zeta(T')} \tag{32}$$

and the acceptance probability becomes

$$\rho = \min\left\{1, \frac{P(T', V \mid D)\zeta(T)}{P(T, V \mid D)\zeta(T')}\right\} \tag{33}$$

Swap node labels A further move is to select two nodes uniformly at random and swap over the events associated with each node. All nodes below either affected node need to be rescored.

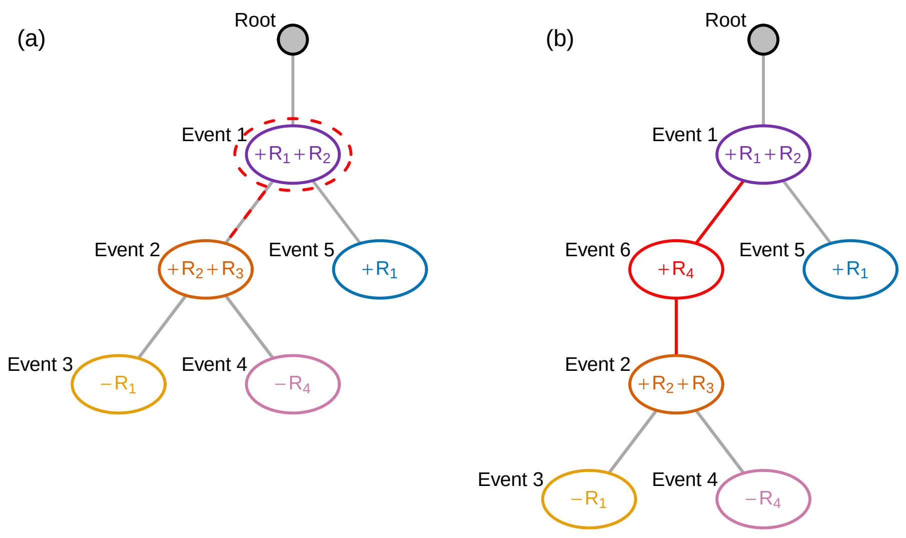

Figure A6: **Add or remove node.** From the tree in (a) [which is the tree in Figure A5a] we sample to add a new node and select event 1 as the new parent. From the two children of event node 1, we select event node 2 to become a child of the newly inserted node. With the node inserted, we sample the events in the new node as an amplification of region 4 to arrive at the tree in (b).

This move is symmetric and hence accepted with probability

$$\rho = \min\left\{1, \frac{P(T', V \mid D)}{P(T, V \mid D)}\right\} \tag{34}$$

However, again we create a weighted version. We define $e_{ij}(T)$ to be the number of nodes affected by the swap. This is the number of descendants of node i plus the number of descendants of node j plus 2, if i and j are in different lineages, or the number of descendants of the ancestor minus the number of descendants of the lower node if they are in the same lineage. Then we can sample the pair (i, j) proportionally to $e_{ij}(T)^{-1}$. Since the tree structure does not change however, this move is also symmetric.

Add or remove events This move changes the event vector, but keeps the tree fixed. We first select a node at random (uniformly). Then we sample the number of regions to be affected from a $\text{Poisson}(\lambda_R) + 1$ and sample this number of (distinct) regions uniformly. Then for each region we sample the number of additional copies from a $\text{Poisson}(\lambda_C) + 1$ and a sign uniformly. If the change leads to all events at the node being completely cancelled (say at node 3 in Figure A5a which only has the event $-R_1$ we sample adding $+R_1$), then we simply reject the move.

This move is symmetric (since the sign accounts for adding and deleting) so the acceptance ratio is

$$\rho = \min\left\{1, \frac{P(T, V' \mid D)}{P(T, V \mid D)}\right\} \tag{35}$$

Setting $\lambda_C = 0$ and $\lambda_R = 0$ makes this move select only a single region, and only allows its number of copies to change by 1.

In the weighted version, we sample nodes proportionally to $d_i(T)^{-1}$ as for the weighted prune and reattach, but since the tree structure does not change, the move is still symmetric.

Add or remove node This move changes both the tree structure and the event vector, as in the example of Figure A6. To define the move, we first consider the possible ways of adding a node. We may place the new node below any of the $(n+1)$ in the current tree including the root. However, when we place the new node below a node with δ children, any of the δ may become children of the new node instead of remaining siblings. There are therefore

$$\chi(T) = \sum_{i=0}^{n} 2^{\delta_i(T)} \tag{36}$$

possible placements of the new node in the tree T where δ_i is the number of children of node i. For the example in Figure A6a then $\delta = (1,2,2,0,0,0)$ and hence $\chi = 13$.

For the new node, we sample the number of regions to be affected with a $\text{Poisson}(\lambda_R) + 1$. For each of the r regions we select from the K regions in total with $\binom{K}{r}$ ways. If $r > K$ we just reject the move for simplicity. Then for each of the selected regions we sample a sign uniformly and a number of copies c from $\text{Poisson}(\lambda_C) + 1$. The transition probability, given that we selected to add a node, to that exact tree with that exact label is then

$$Q_{\text{add}}(T' \mid T) = \frac{\lambda_R^{(r-1)} \mathrm{e}^{-\lambda_R} \lambda_C^{\sum_{j=1}^{r}(c_j-1)} \mathrm{e}^{-r\lambda_C}}{2^r \binom{K}{r} \chi(T)(r-1)! \prod_{j=1}^{r}(c_j-1)!} \tag{37}$$

Since there are many more ways to add a labelled node, than to delete one, we weight the deletion by the corresponding terms in adding the node back. For example, for each node i of the n nodes to delete, we count the number of regions r_i and the number of copies $c_j, j = 1, .., r_i$ and weight that node by the factor

$$w_i(T,V) = \frac{\lambda_R^{(r_i-1)} \mathrm{e}^{-\lambda_R} \lambda_C^{\sum_{j=1}^{r_i}(c_j-1)} \mathrm{e}^{-r_i\lambda_C}}{2^{r_i} \binom{K}{r_i} (r_i-1)! \prod_{j=1}^{r_i}(c_j-1)!} \tag{38}$$

with the sum of these deletion weights, $W(T,V) = \sum_{i=1}^{n} w_i(T,V)$.

For the move, we first sample, with equal probability, whether to delete or add. If delete is chosen then we sample a node from the n available proportionally to the weights $w_i(T,V)$, with probability $\frac{w_i(T,V)}{W(T,V)}$. The sampled node is removed and all of its children are attached to the nodes previous parent. If add is chosen, we sample a node from the $(n+1)$ available including the root proportionally to $2^{\delta_i(T)}$. The new node is inserted as a child of the sampled node. The previous children of the sampled node are each independently randomly assigned to become children of the new inserted node with probability $\frac{1}{2}$, otherwise they remain siblings of the new inserted node. Finally the label of the inserted node is sampled with the Poisson distributions for the number of regions affected, their numbers of copies and uniformly chosen signs for each.

Since the weights for deletion include all the terms corresponding to sampling, for detailed balance we accept the move with a probability of

$$\begin{aligned} \rho_{\text{add}} &= \min\left\{1, \frac{P(T',V' \mid D)\chi(T)}{P(T,V \mid D)W(T',V')}\right\} \\ \rho_{\text{del}} &= \min\left\{1, \frac{P(T',V' \mid D)W(T,V)}{P(T,V \mid D)\chi(T')}\right\} \end{aligned} \tag{39}$$

depending on whether add or delete was selected.

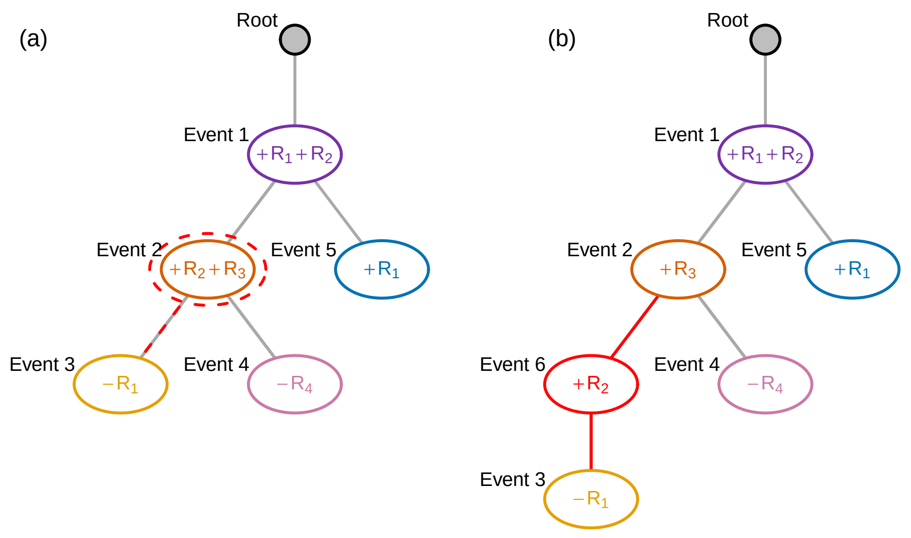

Figure A7: **Condense or split nodes.** From the tree in (a) [which is again the tree in Figure A5a] we sample event node 2 to split. From its two children, we select event node 3 to become a child of the newly split off node which becomes event node 6. With the node placed, we sample how to split the events which were previously at event node 2 across that node and its new child to arrive at the tree in (b).

For weighting the moves also according to their computational cost, we need to further weight each term in the proposal neighbourhood by the cost of updating the score. For each node i to be deleted there are a remaining $(d_i(T) - 1)$ nodes to rescore so we define a new weight (without the minus 1)

$$u_i(T, V) = \frac{w_i(T, V)}{d_i(T)}, \qquad U(T, V) = \sum_{i=1}^{n} u_i(T, V) \tag{40}$$

For adding nodes, say the new parent has δ children, we would need to compute the weights for all 2^δ choices. For simplicity we weight all those choices by assuming that on average only half of its descendants are affected along with the new node. The weights for adding are

$$\xi_i(T) = \frac{2^{\delta_i(T)+1}}{d_i(T) + 1}, \qquad \Xi(T) = \sum_{i=0}^{n} \xi_i(T) \tag{41}$$

and the acceptance probability becomes

$$\begin{aligned} \rho_{\text{add}} &= \min\left\{1, \frac{P(T', V' \mid D)\Xi(T)2d_{i'}(T')}{P(T, V \mid D)U(T', V')(d_i(T) + 1)}\right\} \\ \rho_{\text{del}} &= \min\left\{1, \frac{P(T', V' \mid D)U(T, V)(d_{i'}(T') + 1)}{P(T, V \mid D)\Xi(T')2d_i(T)}\right\} \end{aligned} \tag{42}$$

Condense or split nodes Finally we allow nodes to be split into two, or to combine a parent and child into one, as in the example of Figure A7.

We first consider the move where we split a single node into a parent-child. If we select the node i we would look at the vector $\boldsymbol{v}^i$ of the copy number events at that node. Let's say we have the vector $\boldsymbol{v} = (0, -1, 2, 0, 0)$ which we need to split into two vectors. We sample the splits with a Poisson distribution with parameter λ_S in the following way:

$$\boldsymbol{v}_k^p = \begin{cases} \alpha\beta\Lambda & \boldsymbol{v}_k = 0 \\ \frac{\boldsymbol{v}_k}{2} + \beta\left[\frac{1}{2} + \Lambda\right] & \boldsymbol{v}_k \text{ odd} \\ \frac{\boldsymbol{v}_k}{2} + \beta\Lambda & \boldsymbol{v}_k \text{ even} \end{cases}, \qquad \begin{array}{lcl} \alpha & \sim & \text{Bernoulli}\,(\alpha_0) \\ \beta & \sim & 2\text{Bernoulli}\left(\frac{1}{2}\right) - 1 \\ \Lambda & \sim & \text{Poisson}(\lambda_S) \end{array} \tag{43}$$

where for positions which are 0 in $\boldsymbol{v}$ we allow them to be split into cancelling events with a low probability α_0. This sampling provides the proposed event vector of the parent, while that of the child is fixed by $\boldsymbol{v}^p + \boldsymbol{v}^c = \boldsymbol{v}$. If either vector ends up being completely 0, we reject the move.

For each of the regions we compute the absolute difference c between the number of copies in the new parent vector and the new child vector. The probability of picking each difference is

$$f(c) \underset{\boldsymbol{v}_k \neq 0}{=} \begin{cases} \mathrm{e}^{-\lambda_S} & c = 0 \\ \frac{\lambda_S^{\frac{c-1}{2}}\mathrm{e}^{-\lambda_S}}{2(\frac{c-1}{2})!} & c \text{ odd} \\ \frac{\lambda_S^{\frac{c}{2}}\mathrm{e}^{-\lambda_S}}{2(\frac{c}{2})!} & c \text{ even} \end{cases}, \qquad f(c) \underset{\boldsymbol{v}_k = 0}{=} \begin{cases} (1-\alpha_0) + \alpha_0\mathrm{e}^{-\lambda_S} & c = 0 \\ \alpha_0\frac{\lambda_S^{\frac{c}{2}}\mathrm{e}^{-\lambda_S}}{2(\frac{c}{2})!} & c \text{ even} \end{cases} \tag{44}$$

For the split, we do not consider the root so there are

$$\chi(T, V) = \sum_{i=1}^{n} 2^{\delta_i(T)} \tag{45}$$

ways of arranging splits. The transition probability to that exact tree and exact event vector is

$$Q_{\text{split}}(T', V' \mid T, V) = \frac{\prod_{j=1}^{r} f(c_j)}{\chi'(T, V)} \tag{46}$$

To compensate for this we weight the reverse move of combining a node with its parent by

$$w_i(T, V) = \prod_{j=1}^{r} f(c_j) I(\text{Pa}_i \neq 0) \tag{47}$$

where the c_j are the absolute difference in copy number change in the non-zero regions in the node and its parent. Nodes attached to the root get a weight of 0, which we denote with the indicator function I on the parent Pa_i of node i where 0 represents the root.

The rest of the move follows analogously to the *add or remove node* move, including the weighted version.

Genotype preserving prune and reattach Finally we introduce a variant of *prune and reattach* where instead of retaining the event vector at the pruned node, we retain its genotype of the full copy number profile at that attachment point. When the node is reattached, the event vector of the pruned node is simply updated as the difference in the previous genotype of the pruned node and that of the new attachment point. In the example of Figure A5a the total genotype at event node 2 is $(+R_1 + 2R_2 + R_3)$ while if we were to reattach below event node 5 with genotype $(+R_1 + 2R_2)$ we would simply replace the event vector at event node 2 by $+R_3$. As the copy number states at each event node are preserved, the likelihood of the current and proposed tree are identical and only the proposed event vector prior needs to be computed and compared to the current value to decide on the acceptance of the move.

Since computing the score of a proposal just involves evaluating the event prior, for this move we score the entire neighbourhood of prune and reattach proposals (including the current tree), and sample directly from the neighbourhood.

Changing the overdispersion Alongside tree moves, we also change the overdispersion parameter ν. Since this parameter is positive we work on the log space and propose a new value ν'

$$\log(\nu') = \log(\nu) + \epsilon\,, \qquad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2) \tag{48}$$

using a Gaussian random walk. The standard deviation of the walk can be adapted to the standard deviation of the recent $\log(\nu)$ values of the chain [with a minimum]. The move to ν' is accepted with probability

$$\rho = \min\left\{1, \frac{P(T, V, \nu' \mid D)}{P(T, V, \nu \mid D)}\right\} \tag{49}$$

since the move is symmetric.

Adaptive tempering To speed up the search procedure for the highest scoring tree, we can raise the score to a power $P(T, V \mid D)^\gamma$, and adjust the likelihood landscape by varying γ. For $\gamma > 1$ we amplify the differences between tree scores and spend more time exploring a local neighbourhood, while for $\gamma < 1$ we flatten the landscape and move globally more easily. We vary γ adaptively by keeping track of the acceptance probability of moves in the chain. For every tree move which is accepted we transform $\gamma \to \gamma e^{a(1-a)}$ and for every move that is rejected $\gamma \to \gamma e^{-a^2}$ so that we aim to keep the acceptance probability at a. As a default we set $a = \frac{1}{K}$ as with more regions it is less likely to pick one that improves the tree.

Maximisation instead of summation Instead of the summation of Equation (9), we can target the score defined by attaching cells at the maximally scoring placement using Equation (10) which avoids the need to penalise the combinatorial complexity. In the scheme above, we simply replace $P(T, V \mid D)$ everywhere by $S(T, V \mid D)$ and search for the maximal score S. Since we no longer need to satisfy detailed balance, we simplify the acceptance probability of each move to

$$\rho = \min\left\{1, \frac{S(T', V \mid D)}{S(T, V \mid D)}\right\} \tag{50}$$