Title
Methods for Jointly Calling Somatic Copy Number Alterations in Single Cell Whole Genome Sequencing for Inferring Tumor Phylogenies

Author
Hui, Sandra Camille

Publication Date
2022

Peer reviewed|Thesis/dissertation

Methods for Jointly Calling Somatic Copy Number Alterations in Single Cell Whole Genome Sequencing for Inferring Tumor Phylogenies

by

Sandra Camille Hui

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computational Biology

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Rasmus Nielsen, Chair
Associate Professor Nir Yosef
Associate Professor Dirk Hockemeyer
Associate Professor Hani Goodarzi

Summer 2022

Methods for Jointly Calling Somatic Copy Number Alterations in Single Cell Whole Genome Sequencing for Inferring Tumor Phylogenies

Abstract

Methods for Jointly Calling Somatic Copy Number Alterations in
Single Cell Whole Genome Sequencing for Inferring Tumor Phylogenies

by

Sandra Camille Hui

Doctor of Philosophy in Computational Biology

University of California, Berkeley

Professor Rasmus Nielsen, Chair

Tumors develop after the accumulation of passenger and deleterious driver mutations, including single nucleotide mutations and large scale copy number alterations (CNAs). With the advent of next generation sequencing, large consortia, such as The Cancer Genome Atlas and the International Cancer Genome Consortium, have sequenced thousands of tumor and healthy blood (matched normal) samples from cancer patients. These bulk sequencing studies have yielded an unprecedented amount of information on the molecular level, including showing recurrent molecular signatures across tissue types, as well as patient specific aberrations [1]. While this information can be utilized in clinical settings, the averaging effect of bulk sequencing can obscure rare mutations, which, if not taken into account in treatment, can lead to future relapses. Furthermore, because we cannot ethically obtain longitudinal samples from treatment naive solid tumors to observe how tumors grow and change over time, reconstructing tumor evolutionary phylogenies from samples taken at a single time point is an active area of research.

By treating a collection of tumor cells as individuals in a population, single cell sequencing (SCS) can reveal finer levels of detail of rare events in a tumor population, and allows application of well studied population genetics methods for studying evolution. However, SCS techniques are very noisy due to low amounts of starting DNA. This can manifest as low and uneven coverage across the genome and allelic dropout, making point mutation calling particularly difficult [2]. Although cell dissociation and isolation remain challenges, we aim to computationally address problems and biases specifically introduced from whole genome amplification (WGA), a necessary step because of the small amount of starting material in each cell. Despite these challenges, the number of reads falling in each window along the genome (read depth) can be used to study copy number events.

Here, we present three methods to estimate copy number profiles and cell similarity. In Chapter 2, we present a novel method, SCONCE [3], that accounts for single cell whole

genome sequencing noise and calls copy number events in single cells. SCONCE is based on a Hidden Markov Model that incorporates a Markov process continuous through time, to model the evolutionary history of a tumor, as well as a discrete process along the length of the genome, to estimate copy number alterations from changes in read depth. We show SCONCE outperforms competing methods across a wide range of simulated and published real datasets [4, 5].

In Chapter 3, we present SCONCE2 [6], an expansion on SCONCE to jointly call copy number events across multiple cells and estimate cell similarity. SCONCE2 uses pairs of cells to model evolutionary relationships and estimate joint copy number profiles. By summarizing these joint copy number profiles across multiple cell pairs, SCONCE2 more accurately detects breakpoints and copy number events. Furthermore, SCONCE2 creates a novel cell similarity metric based on pairwise tree branch lengths, which can be used to estimate tumor phylogenies using neighbor-joining. Using a combination of public data [4, 5] and simulations, compared to other methods, we show SCONCE2 more accurately calls copy number profiles, detects breakpoints, and estimates pairwise cell similarity, leading to tumor phylogeny estimates with less error.

Finally, in Chapter 4, we present SCONCEmut, a further expansion on SCONCE and SCONCE2, by utilizing genotype likelihoods. Although calling point mutations in single cell sequencing remains difficult due to noisy and low read depth, estimating the number and types of shared and independent mutations between cells can be incorporated into branch length estimates. Using a range of simulated and real data, we explore a model to jointly estimate mutation counts, using genotype likelihoods, copy number profiles, and tree branch lengths.

In this dissertation, we show SCONCE, SCONCE2, and SCONCEmut can be used to accurately call copy number events and study evolutionary relationships in single cell whole genome tumor sequencing data. When applied to additional datasets, investigators can gain further insight into and understanding of tumor evolution, potentially leading to more effective cancer detection and treatment protocols.

To my family, friends,
loved ones, and possibility models

Contents

Acknowledgments

Thank you first to my advisor, Rasmus Nielsen, and all the past and current members of the Nielsen Lab. Rasmus, thank you for always making time to meet with me, for teaching me to be a better scientist, for your mentorship and suggestions, and for your willingness to explore ways to increase equity and inclusion in your lab. Tyler Linderoth and Maya Lemmon-Kishi, thank you for your past and future stewardship and ongoing troubleshooting of the Nielsen lab servers. Amy Ko, Débora Yoshihara Caldeira Brandt, Diana Aguilar Gómez, and Joana Rocha, thank you for sharing joy, commiseration, cheerleading, and solidarity (such as through all the thermostat adjustments). My time in the Nielsen lab would not have been the same without your friendship, our cubicle, or our community plant menagerie.

Thank you to my qualifying exam and thesis committee members, Nir Yosef, Dirk Hockemeyer, Hani Goodarzi, Priya Moorjani, and Haiyan Huang. I've learned a lot from your rigorous feedback, and my science is better for it.

Thank you to the wonderful students, staff, and faculty in the Center for Computational Biology. Thank you to Rob Tunney and Brooke Rhead for warmly welcoming me before I was even accepted to Berkeley, and for being ongoing sources of advice. Thank you to Carmelle Catamura and Sarah Johnson, for being gracious and enthusiastic mentees, and especially Sarah for doing so much data analysis as a rotation student. My deepest gratitude to Kate Chase and Xuan Quach for always knowing the answers to my logistics questions and for the phenomenal amount of organizational and behind the scenes work you do to keep everything running smoothly and hiccup free.

I would be remiss if I didn't thank the people who started me on this path. Thank you to Olivier Harismendy, for taking a chance on a green undergraduate researcher and for mentoring, supporting, and teaching me so much during my first foray into cancer genomics research. Thank you for introducing me to the many exciting and vast possibilities of computational biology. Thank you to Christine Alvarado, for encouragingly emailing after I completed my first undergraduate computer science course, and for mentoring me through the entirety of my undergraduate teaching career. Thank you also to Camellia Lee, Abby Gallegos, and Niema Moshiri for your friendship, teamwork, and mutual support in the UCSD undergraduate bioinformatics program.

One of the most fulfilling parts of my PhD has been collaborating with some of the many hardworking teams and individuals working to make the UC Berkeley campus environment more inclusive, welcoming, equitable, and supportive. Thank you especially to Bias Busters, the students on the CCB DEI committee, and the Chancellor's Advisory Committee on the Lesbian, Gay, Bisexual, Transgender, and Queer Communities at Cal. Creating and changing institutional structures is hard, slow, exhausting work, and I'm deeply appreciative of your teamwork, energy, fortitude, determination, idealism, and solidarity. Thank you also to Rori Rohlfs, Noah Whiteman, and Peter Sudmant for sharing encouraging words, advice, and optimism. Furthermore, thank you to all of the queer, nonbinary, and trans scientists I've gotten to know for reminding me I'm not alone, and I look forward to building community with so many more of you.

Outside of academia, I've been incredibly fortunate to find, build, and belong to several vibrant communities. Thank you to the percussion section of the Community Women's Orchestra, Suzanne Wallace, Elaine Lin, Kelly O'Donnell, and Nikki Collister, for welcoming me, generously sharing instruments, and creating and exploring music with me. My heart still glows thinking about the wide selection of triangle beaters we got to play with. Extra thanks goes to Serena Le, Amanda Mok, and Katherine Ray for the lovely carpool conversations, and especially for the truly staggering amount of time and work you all put in to develop and implement thoughtful policies to make CWO more inclusive and accessible. Thank you for showing me how to stand up for what I believe in, and more importantly, how to set strong boundaries. Thank you also to Kiku Johnson for your incredibly generous, compassionate, empathetic, and kind guidance and suggestions.

Thank you to all of the wonderful people in Wicked Transcendent Folk. Thank you for vulnerably sharing your wisdom and experiences moving through this cissexist, transphobic, homophobic, heteropatriarchal world, for trusting me to co-create warm, welcoming, and affirming spaces, and for showing me so many expansive and transgressive possibilities for understanding, interrogating, exploring, performing, embodying, and transcending gender. Thank you especially to the group founders, and to my phenomenal co-facilitators, Rafael Matias Langer-Osuna, Holly Dale, Christopher Chagal, and Ren Schiffman. Thank you for your flexibility and for so continuously and generously sharing your time and energy to grow and support this community. Thank you also to Anne Mitchell, for fearlessly leading the peer group program at the Pacific Center. Thank you all for showing me new ways of being, for sharing trans joy and euphoria, and for richly feeding my imagination.

Thank you to Allison Aiken, Scott Mosser, Aimee Jacques, Laura Alie, Cathy Hanville, and Danny Schnittman. Your deep and open listening, care, and support have immeasurably helped me grow into myself and move through the world with more compassion, quiet inner calm, embodiment, and wholeheartedness. Thank you also to Kris Bondoc for the sharpest shadow fades I've ever seen. I always feel so much more at home in myself after seeing you.

Thank you to my amazing cohort. Amanda Mok, despite frequently leading me astray, you've never led me astray when it really matters; thank you for showing me how to stand in one's power. Tal Ashuach, thank you for modeling how to genuinely and fearlessly ask questions and learn from the answers. Nick Everetts, thank you for reminding us to be safe and make good choices, as well as always being ready with a ridiculously small but reliable set of dad jokes. Jared Bennett, thank you for muddling through so many server problems with me. Jordi Silvestre-Ryan, thank you for sharing your truly memorable gradient descent demonstration with us. To the Screaming Unicorns, may you always have exactly as many (or few) tomatoes as you desire, and may your p-values never be negative.

Thank you to my dearest friends for your warmth, support, cheerleading, and friendship. Ace Harlo, thank you for always having an open non judgemental ear, sharing your vulnerabilities and wisdom, and forever delighting me with your Avant-Rainbow art. There's no one else I'd rather go with on a queer fashion adventure. Arden Liao, thank you for being someone I can always turn to, for being such a safe and trusted friend, and for celebrating wins and mourning losses (especially the truly minuscule). Despite the jaded tenor of many

of our conversations, our friendship is truly one of the things that keeps me going, and I look forward to our next soup & noodles, ice cream, and movie night. Isabel Serrano, thank you for sharing your infectious optimism and enthusiasm for science, modeling communicative and caring friendship, and keeping me grounded in things outside of science. I am a better friend and person because of you all, and I'm endlessly grateful.

Thank you to my family, especially my parents and brother, for your endless support and confidence in my abilities. Thank you to my parents for growing with me, for encouraging and supporting me to make the decisions that are best for me, and for teaching me to give back to my communities and make things better for those who come after me. Theo, I'm so excited and thrilled to see you forge your own path, and I look forward to supporting you along the way.

Finally, thank you to Hieu Nguyen, for being a place of love, support, laughter, comfort, empathy, quiet and uncomplicated belonging, exploration, learning, understanding, and deep acceptance. Thank you for showing me how to choose, love, and be myself, for holding me accountable, for sharing your unreserved curiosity, for naming SCONCE2, and for demonstrating new ways of growing and doing relationships. You bring so much joy, compassion, kindness, tender care, thoughtfulness, and vulnerability into my life, and I'm a better person because of you. Thank you.

Chapter 1

Introduction

1.1 Overview of cancer genetics

Cancer develops from an accumulation of deleterious mutations, including single point mutations (base substitutions), small insertions and deletions (indels), changes to molecular markers controlling gene expression (epigenetic modifications), reordering of genomic elements (translocations and genomic rearrangements), and large scale amplifications and deletions of whole chromosomal regions (copy number alterations). These mutations disrupt cell functions controlling cell proliferation, death, and movement, leading to the unchecked cell growth characteristic of cancer, in two ways: by accelerating and promoting cell division (akin to stepping on the gas pedal), and by removing or disabling mechanisms to control and stop cell growth (similar to stepping on the brakes) [7, 8].

In the first scenario, on the DNA level, proto-oncogenes (genes that help cells grow, and are essential for organism growth and development) are mutated into oncogenes, leading to a variety of growth promoting mechanisms. For example, point mutations might make a growth promoting protein more active or long lasting, copy number amplifications might increase the number of copies of oncogenes from two (healthy diploid), or point mutations might lead to increased expression of growth gene products [9–11].

In the second scenario, tumor suppressor genes (genes that slow cell division and growth) are disabled. For example, indels might cause DNA repair proteins to be nonfunctional (leading to a faster accumulation of mutations), copy number deletions might remove chromosomal segments containing one or more copies of tumor suppressor genes, or point mutations might disable a gene necessary for cell adhesion (potentially leading to metastases) [12, 13].

Both proto-oncogenes and tumor suppressor genes play key roles in normal cell growth and organism development. However, healthy cells can evolve into cancerous cells over time as they accumulate and pass down these deleterious driver (causal) mutations to daughter cells through cell division. Eventually, enough unchecked somatic mutations (activating oncogenes and deactivating tumor suppressor genes) are gained to lead to cancerous uncontrolled cell growth. These acquired mutations and their shared history can be leveraged to

better understand the genetics of cancer development [7, 14–16].

1.2 Overview of tumor sequencing

To study cancer genomics, tumors are often sequenced in bulk with a matched healthy sample from the same person, enabling identification of germline (person specific) variants and somatic (cancer specific) mutations. Deleterious germline mutations, which are passed from parent to child, can predispose an individual to a higher risk of developing cancer. In contrast, somatic mutations are acquired throughout a person's lifetime, such as through UV damage, carcinogen exposure, or spontaneous mutations associated with DNA metabolism. In this dissertation, we focus on identifying somatic mutations by examining two aspects of tumor sequencing data: the quantity of reads covering a specific region of the genome (ie, how many reads fall into a genomic window), which can be used to estimate copy number/amount of DNA; and the base content of reads at a given genomic position (ie, which letter, of $\{A, C, G, T\}$, is this site most likely to be), which can be used to call point mutations.

Despite the rapid advances in sequencing technology [17], however, genomic sequencing is not bias or error free. For example, the ratio of G/C bases to A/T bases can bias sequencing results (termed GC bias) [18]. Additionally, some genomic regions are harder to access (such as tightly bundled centromeres) or map (such as highly repetitive regions), thereby reducing sequencing efficiency in these regions (termed mappability bias) [19]. Both GC and mappability biases can directly negatively effect copy number estimates if not properly controlled for. Furthermore, the sequencing process itself can introduce base substitution errors, where the wrong letter is read out [18]. Fortunately, these sequencing errors can be probabilistically accounted for if there is high enough read depth (ie, how many times this base has been independently read), but remain a challenge in areas with low read depth (that is, there aren't enough reads to rule out a sequencing error) [18].

Nonetheless, many tools have been developed to address and model these errors. Bulk sequencing can yield vital information about a tumor's key characteristics and prognosis, and large tumor studies have successfully generated a wealth of data [1, 20–23]. However, the sample preparation process for bulk sequencing requires mixing all cells in a sample together, thereby averaging out rare events and losing cell specific information.

In contrast, single cell sequencing lets one examine cells as a collection of individuals in a population, allowing for adaption and application of well established population genetics principles and tools, identification of rare and cell specific events, and examination of inter-cellular relationships. For example, in a tumor with high intratumor heterogeneity (that is, one tumor with multiple distinct sub populations), some cells may contain rare mutations that will help them evade a particular drug therapy, which, if left unaddressed, could lead to a later relapse or metastasis. However, if these rare mutations are detected via single cell sequencing, treatments can be tailored and combined to target a higher diversity of cells, thereby improving patient outcomes. Additionally, single cell sequencing can be used with

liquid biopsies for noninvasive early cancer detection and treatment monitoring, where individual circulating tumor cells can be extracted from a routine blood draw to identify cancer cells and longitudinally analyze disease progression and treatment response. Furthermore, using population genetics methods, investigators can gain a deeper understanding of the evolutionary processes underlying cancer development, such as estimating changes in mutation and growth rates, comparing tumor evolution models, identifying driver mutations, tracing cancer lineages and trajectories, and inferring evolutionary trees (phylogenies). Finally, using variations on single cell DNA sequencing, such as single cell RNA sequencing (that is, quantifying gene expression), researchers can examine cell specific transcriptional patterns, and changes therein, to highlight functional differences, which can further deepen understanding of cancer evolution and guide personalized therapies [24–30].

However, single cell sequencing comes with a its own set of challenges. In particular, due to the small amount of starting material in one cell (compared to a whole tumor) and necessary whole genome amplification steps, sequencing coverage tends to be sparse and uneven across the genome, with low overall read depth. The limited information about each genomic position makes calling point mutations extraordinarily difficult. Additionally, low read depth can lead to things like allelic drop out, where some alleles or gene copies are unobserved because they were not successfully amplified, leading to incorrect point mutation calls. In contrast, by zooming out and counting the number of reads in a large region, we can gather sufficient information to estimate copy number alterations.

1.3 Overview of SCONCE, SCONCE2, and SCONCEmut

In order to address the challenges presented by low read depth, we first focus on estimating large scale copy number events across the genome in single cells. In Chapter 2, we present our method SCONCE, which uses a principled evolutionary model to call copy number alterations in these data. SCONCE uses matched single cell sequencing of diploid cells as a null model to correct for sequencing biases, namely GC and mappability biases. We applied SCONCE to simulated and previously published real data [4, 5], and show it calls copy number profiles and detects breakpoints with higher accuracy than competing methods.

However, SCONCE does not utilize any shared evolutionary history between cells. In Chapter 3, we expand the SCONCE framework into a new method called SCONCE2, which jointly estimates copy number profiles and evolutionary distances across multiple cells. SCONCE2 estimates pairwise copy number profiles, and produces consensus profiles by summarizing across all cells. In this process, SCONCE2 infers evolutionary relationships between each pair of cells by estimating the amount of shared evolutionary time (branch length) until divergence, as well as the length of time each cell evolved independently. We applied SCONCE2 to a range of simulated and real data [4, 5], and show using information from multiple cells improves upon the copy number call and breakpoint detection accuracy

values from SCONCE. We also apply a well established population genetics method for phylogeny estimation, called neighbor-joining [31, 32], using the estimated evolutionary pairwise branch length relationships to infer more accurate phylogenies than competing methods.

Finally, although definitively calling mutations in low coverage sequencing data is extremely difficult, information can still be gained from estimating the most likely base(s) at a given position (termed the genotype likelihood). In Chapter 4, we further expand upon the SCONCE and SCONCE2 frameworks, by incorporating point mutation data into our estimates of branch lengths. Intuitively, if mutations occur at a constant rate, as the branch length increases, the expected number of mutations on that branch should also increase (that is, there is more time for mutations to occur). We combine this information about mutation counts (and therefore expected branch lengths) with the inferred copy number profiles to jointly estimate branch lengths. Using a wide array of simulated and real data, we explore a model for including estimates of point mutation counts for branch length and phylogeny estimates.

In this dissertation, we present three methods to study somatic copy number alterations in single cell whole genome tumor sequencing. By accurately calling copy number profiles with SCONCE (see Chapter 2, estimating intercellular pairwise evolutionary relationships with SCONCE2 (see Chapter 3), and utilizing point mutations with SCONCEmut (see Chapter 4), investigators can gain deeper insight into cancer evolutionary dynamics, which can potentially lead to more advanced and personalized cancer treatments.

Chapter 2

SCONCE: A method for profiling Copy Number Alterations in Cancer Evolution using Single Cell Whole Genome Sequencing

This chapter was previously published in *Bioinformatics* (2022), and is included as published here. The authors on this paper are:

Sandra Hui[1], Rasmus Nielsen[1,2,3]

1. Center for Computational Biology, University of California, Berkeley, Berkeley, CA, USA
2. Department of Integrative Biology, University of California, Berkeley, Berkeley, CA, USA
3. Department of Statistics, University of California, Berkeley, Berkeley, CA, USA

2.1 Abstract

Motivation: Copy number alterations are a significant driver in cancer growth and development, but remain poorly characterized on the single cell level. Although genome evolution in cancer cells is Markovian through evolutionary time, copy number alterations are not Markovian along the genome. However, existing methods call copy number profiles with Hidden Markov Models or change point detection algorithms based on changes in observed read depth, corrected by genome content, and do not account for the stochastic evolutionary process.
Results: We present a theoretical framework to use tumor evolutionary history to accurately call copy number alterations in a principled manner. In order to model the tumor evolutionary process and account for technical noise from low coverage single cell whole genome sequencing data, we developed SCONCE, a method based on a Hidden Markov Model to analyze read depth data from tumor cells using matched normal cells as negative controls. Using a combination of public data sets and simulations, we show SCONCE accurately decodes copy number profiles, and provides a useful tool for understanding tumor evolution.

2.2 Introduction

In cancerous cells, somatic driver and passenger single nucleotide polymorphisms (SNPs) and copy number alterations (CNAs) accumulate over time. CNAs are extremely common across cancer types [21, 33].

Many large scale cancer studies are done with bulk samples, and many methods and evaluation techniques [34, 35] have been developed to identify copy number alterations in bulk sequencing, especially for low coverage data [36] and tumor heterogeneity deconvolution [37]. However, bulk sequencing averages mutations across many cells and loses the granularity and detail single cell sequencing (SCS) can provide. Single cell sequencing facilitates analyses treating each cell as an individual in a population. However, the SCS process is technically challenging and produces noisy low coverage data, due to challenges such as cell dissociation, small amounts of starting DNA, and non uniform whole genome amplification [38]. Although the rapidly increasing availability of single cell RNA sequencing (scRNA-seq) of tumors can yield insights into tumor subpopulations [39] and relevant biological pathways and processes [40, 41], using scRNA-seq for calling CNAs is limited to areas of the genome that are expressed at the time of sequencing and does not directly measure genomic copy number. However, single cell whole genome DNA sequencing data promises to circumvent these problems, despite the inherent noisiness of the data.

The main components of CNA calling are detecting breakpoints between contiguous regions of the genome with the same copy number and determining the absolute copy number of each region. Previous approaches to calling CNAs using single cells have been based on Hidden Markov Models (HMMs) and change point detection [42]. For example, HMMcopy

use a Hidden Markov Model to segment tumor genomes, normalized by matched normal cells. Although HMMcopy was originally designed for array comparative genomic hybridization data [43, 44], it has been widely used for single cell sequencing data [42, 44].

Another method, CopyNumber [45], was also designed for microarray use. Although CopyNumber detects breakpoints, it does not output absolute copy number calls. One strength of CopyNumber, however, is that it can be run in individual and multi sample modes, where breakpoints are forced to be shared across all samples [45].

A third program, DNAcopy [46, 47], was designed for microarray use, and uses circular binary segmentation to identify breakpoints, but does not output absolute copy number calls [46, 47]. Although DNAcopy was not originally designed for single cell sequencing data, it has been applied to such data sets [4, 48].

A fourth program, AneuFinder [49, 50], which was designed for calling CNAs in whole genome SCS data, uses an HMM [49] or breakpoint detection analysis [50]. To determine absolute copy number, regions are scaled to have integer copy numbers, or so the mean copy number matches a known ploidy (determined by a DNA quantification technique, such as flow cytometry [51]).

Finally, SCICoNE [52] was designed for CNA calling in whole genome SCS data. It uses a likelihood based model to first detect breakpoints shared across cells, and then builds a CNA based tree to determine absolute copy number values [52, 53].

All of these methods require dividing the reference genome into adjacent bins and using bin or cell specific GC and mappability corrections to adjust read counts and mask out "bad" bins that exhibit extremely high or low coverage due to centromeres, telomeres, or highly repetitive regions. None use stochastic models of tumor evolution to do both breakpoint detection and copy number calling. An objective of this paper is to develop models for CNA calling based on explicit models of tumor evolution. The rationale is that the use of such explicit models of evolution might improve inferences similarly to what has been observed in models of molecular evolution used in phylogenetics [54–56].

Because tumor cells evolve forward in time from an ancestral diploid state through mutations that only depend on the current state of the cell, copy number alterations are inherently governed by a (possibly time-inhomogenous) temporal Markov process. However, the read distribution observed along the length of the genome (the spatial process) is not Markovian. To realize this, consider a mutation within a segment of DNA with copy number 4 that reduces the copy number from 4 to 3. When moving from the left to the right along the length of the genome, the copy number would then go from $4 \to 3 \to 4$. There are two transitions (breakpoints) caused by the same single CNA. In many other situations, the rate of mutation from $3 \to 4$ (as in the second breakpoint) might be low, however, because the chromosome previously was in state 4, the rate of transition back from 3 to 4 is in fact high in our example. The process along the length of the genome is not Markovian because copy number alterations may have finite length and each mutation may induce two breakpoints.

Even though the spatial process is not Markovian, the HMM framework is computationally convenient. An aim of this paper is, therefore, to develop Markovian approximations of the spatial process that can be used for inference. We present SCONCE (Single Cell cOpy

Numbers in CancEr), a method based on modeling the temporal Markovian evolutionary process and deriving a best approximating spatial HMM from this process. SCONCE also uses diploid data as a null to model the technical noise in single cell sequencing data and can robustly learn model parameters and detect copy number alterations. We show on simulated data that the method more accurately estimates the copy number states of a cell than previous state-of-the-art methods, and we analyze real data to show that the observations from simulated data are mirrored by similar differences among methods in analyses of real data.

2.3 Theory and Methods

Simulations

In order to robustly evaluate SCONCE, we use two simulation models, one based on treating the genome as a continuous line and modeling copy number alterations as duplications or deletions of line segments (Line Segment Model), and one based on dividing the genome into discrete bins (Binned Model). Of note, the assumptions of these simulation model are more realistic and differ intentionally from the models implemented in SCONCE described in Hidden Markov Model. We simulate data and estimate parameters and copy number calls under different models, to avoid biasing method comparisons towards our method.

Line segment model

In the Line Segment Model, we assume a genome, G, to have a fixed maximal length, L, and be comprised of c orthologous chromosomes. Each chromosome consists of an ordered list of line segments, which have positions that can be mapped back into $[0, L]$. Amplifications create an extra copy of a chromosome or part of a chromosome. Note there is no maximum copy number imposed by this model, and copy number may go to infinity. A deletion in a chromosome erases part, or the entirety, of one or more line segments in a single chromosome. Once deleted, a segment cannot be regained. A worked example is given in Suppl. 2.8.

Rates of amplification and deletion in the Line Segment model: It is assumed that amplifications and deletions initiate at a constant rates φ and δ, respectively, per unit chromosome and per time unit, with lengths drawn from truncated exponential distributions with respective rates τ_a and τ_d, such that the rate at which a particular point in the region is affected by an amplification or deletion is $\frac{\varphi}{\tau_a}$ and $\frac{\delta}{\tau_d}$, respectively.

We assume that amplifications and deletions run from left to right (but by construction the same distribution is obtained if considering the process from right to left), and the truncation occurs when an amplification or deletion extends beyond the end of the chromosome. To remove edge effects, we additionally assume new events can initiate at the left start of each chromosome with the same rates, $\frac{\varphi}{\tau_a}$ and $\frac{\delta}{\tau_d}$. The total genomic rate at which amplifications and deletions occur at any point in time is then $c\left(\frac{\varphi}{\tau_a} + \frac{\delta}{\tau_d}\right) + \mid G \mid (\varphi + \delta)$.

Induced marginal process: The process, as defined here, is a Markov process with state space on the infinite set of all possible genomes. It also induces a marginal continuous time Markov process at each position in the genome, $W_t \in \mathbb{Z}$, with transition rates $q_{ij} = i\frac{\varphi}{\tau_a}$ if $j = i + 1$, $q_{ij} = i\frac{\delta}{\tau_d}$ if $j = i - 1$ and $j \geq 0$, and $q_{ij} = 0$ otherwise, for copy number states i and j. We notice that this is a linear birth-death process with birth rate $\frac{\varphi}{\tau_a}$ and death rate $\frac{\delta}{\tau_d}$.

Binned process

We also consider an alternative and simpler process, termed the *binned process*, where we assume that the genome can be divided into n bins. The state space in each bin is $\mathbb{S} = \{0, 1, 2, ..., k\}$, where k is the maximum copy number.

Amplifications and deletion lengths in the Binned process We assume that the length of amplifications and deletions follows a truncated geometric distribution with parameter p. That is, given that a certain amplification/deletion occurs in bin i, the probability that it extends to the adjacent bin $i + 1$ is $1 - p$. If the copy number in bin i changes by amount s, the copy numbers in bins affected by the same event change from u to $u' = s + u$ if $0 \leq s + u \leq k$, $u' = 0$ if $s + u < 0$, or $u' = k$ if $s + u > k$.

Marginal process of initiation: We model the marginal process of initiation of new CNAs in each bin as a continuous time Markov chain with rate matrix $Q = \{q_{ij}\}$. The total rate of CNA initiation within any of the n bins, at time t, is then $R_t = \sum_{i=1}^{n} \sum_{j \neq Y_i(t)} q_{Y_i(t)j}$, where $Y_i(t)$ is the state in bin i at time t. Notice, that because of the assumption of geometrically distributed lengths of amplifications and deletions, the marginal process in each bin does not follow Q. Only the initiation process of new amplifications and deletions follows Q.

To ensure an approximately constant rate along the length of the chromosome, amplifications and deletions may also initiate immediately to the left of the first bin. Such events occur at a rate of $\frac{R_t}{np}$, and the relative probability of change to state j from state $Y_0(t)$ is given by $q_{Y_0(t)j}$.

Read Depth Simulation

Both the line segment and binned models simulate observed read depth for a given number of genomic windows directly from the simulated genome, G. Read depths are drawn from a user specified negative binomial distribution.

Simulation Data sets

We simulated 5 data sets under the line segment model and 7 data sets under the binned model, in order to generate a variety of types and quantity of copy number events. Each data set had 100 tumor cells and 100 diploid cells, where read counts from diploid cells were

averaged together to form the background model. Full simulation parameter values are given in Suppl. Simulation dataset parameter values.

Hidden Markov Model

In order to detect breakpoints and call absolute copy numbers, we define a Hidden Markov Model along the length of the genome informed by a tumor cell's evolutionary history. We define the state space, $\mathbb{S}$, of the HMM as the integer tumor copy number in a given genomic bin, from 0 up to a user specified k (suggested $k = 10$), and the alphabet as the integer observed tumor read depth in that bin.

Emission probabilities

We model emission probabilities for tumor read counts for each bin with a negative binomial distribution (interpreted here as an overdispersed Poisson). We incorporate the mean diploid read count for each bin into the emission probabilities, in order to normalize for technical noise and sequencing bias. Note that having a matched diploid sample is necessary to account for sequencing errors. We assume the tumor read depth in window i for tumor cell A to be represented by random variable X_{iA}, such that

$$\mathbb{E}(X_{iA}) = \lambda_{iA} = \left(\rho_{iA} \times \frac{\mu_i}{2}\right) \times s_A + \varepsilon \tag{1}$$

$$X_{iA} \sim NegBinom\left(\lambda_{iA}, \sigma^2_{iA} = a\lambda^2_{iA} + b\lambda_{iA} + c\right) \tag{2}$$

where ρ_{iA} is the state in window i for cell A, μ_i is the mean diploid read depth in window i, ε is a constant sequencing error term, s_A is a cell specific library size scaling factor (see Library Size Scaling Factors), and $\{a, b, c\}$ are constants learned from diploid data (see Learning constants $\{a, b, c\}$). We use a quadratic relationship between the mean and variance of read depth in Equation 1, as this approximation best fit real diploid data [4] (see Suppl. Second degree polynomial). Therefore, the emission probability for an observed read depth, x_{iA}, is given by Equation 2.

Joint evolutionary process of two bins forward in time

In Simulations, we described two principled models of CNA evolution. However, neither of these models have the property that they are Markovian along the length of the genome. To construct an approximating process that is Markovian, we will first construct a process jointly affecting two bins. From this description of the joint evolution of two bins, we will then derive the approximating Markov process used as the transition probabilities in the HMM.

Consider two adjacent bins in the genome on one lineage, $(U, V) \in \{(0, 0), (0, 1), \ldots, (k, k)\}$, where U is the copy number in bin i, and V is the copy number in bin $i + 1$. The

copy numbers in these bins change through continuous time evolutionary history according to rate parameters $\{\alpha, \beta, \gamma\}$:

$$\alpha = \text{rate of } \pm 1 \text{ CNA} \tag{3a}$$
$$\beta = \text{rate of any CNA} \tag{3b}$$
$$\gamma = \text{rate of CNAs affecting both } U \text{ and } V \tag{3c}$$

These rates are encoded in a transition rate matrix $\mathbb{Q} = \{q_{(U,V),(U',V')}\}, U, V, U', V' \in \mathbb{S}$, which gives the instantaneous rate of observing a change from (U, V) to (U', V'):

$$q_{(U,V),(U',V')} = \begin{cases} \gamma(\alpha+\beta) & \text{if } (U',V') = \begin{cases}(U+n, V+n)\\(U-n, V-n)\end{cases}, n=1 \\ \gamma\beta & \text{if } (U',V') = \begin{cases}(U+n, V+n)\\(U-n, V-n)\end{cases}, n>1 \\ \alpha+\beta & \text{if } (U',V') = \begin{cases}(U\pm n, V)\\(U, V\pm n)\end{cases}, n=1 \\ \beta & \text{if } (U',V') = \begin{cases}(U\pm n, V)\\(U, V\pm n)\end{cases}, n>1 \\ r_{(U,V)} & \text{if } (U',V') = (U,V) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

We set $r_{(U,V)} = -\sum_{(u',v')\neq(U,V)} q_{(U,V),(u',v')}$ such that all rows sum to 0. As only one event can occur in an infinitesimally small time interval, cases where adjacent bins are simultaneously affected by different events, such as $(U', V') = (U+n, V-n), n > 0$, have instantaneous rate 0. However, notice that any time interval ¿ 0, can contain different changes in adjacent bins.

From this rate matrix $\mathbb{Q}$, the time dependent transition probabilities $\mathbb{P}$ are calculated via the matrix exponential as

$$P_{(U,V),(U',V')}(t) = e^{\mathbb{Q}t} \tag{5}$$

as the solution to the Kolmogorov equations. This gives the probability of observing a transition from (U, V) to (U', V') in evolutionary time t.

Approximating Markovian process along the genome

We convert the forward-in-time process for two bins into an approximating Markov model along the length of the genome with transition probability matrix $\mathbb{M}_t = \{m_{i,i',t}\}$, $i, i' \in \mathbb{S}$, i.e. we identify the one-step transition probability of moving from state i to i' along the genome in a binned process, after a given evolutionary time t. Under the assumption that the cell

has an ancestral diploid state at time $t = 0$, we set $(U, V) = (2, 2)$ and $(U', V') = (i, i')$. To ensure all rows in matrix $\mathbb{M}_t$ sum to 1, we normalize over all states W in $\mathbb{S}$, such that the one-step transition probabilities of the discrete approximating Markov process along the length of the genome are given by

$$m_{i,i',t} = \frac{P_{(2,2),(i,i')}(t)}{\sum_{W \in \mathbb{S}} P_{(2,2),(i,W)}(t)} \tag{6}$$

Given an evolutionary time t, Equation 6 defines the transition matrix for the HMM (described in Hidden Markov Model) along the length of the genome. That is, the HMM transition matrix is fully parameterized by $\{\alpha, \beta, \gamma, t\}$.

The advantage of using a model that approximates a non-Markovian process using an evolutionary time-informed HMM over more generic HMMs is that information about the ancestral diploid state is included in the model specification, allowing, as we will show in the result section, more accurate inference of copy number state. While the model is only an approximation, as it ignores the non-Markovian nature of any realistic model of CNA changes along the genome, we will evaluate it on simulations from the aforementioned more realistic non-Markovian simulation models.

Model Training

The model training has four steps, followed by the copy number profile decoding, shown in Figure 1. We first estimate the emission probability constants in Equation 2, $\{a, b, c\}$, from the diploid data. Second, for each tumor cell, A, we quickly estimate an unconstrained transition matrix, initial probability vector, and library size scaling factor, s_A, using a modification of the Baum-Welch algorithm. Third, the model rate and time parameters, $\{\alpha_A, \beta_A, \gamma_A, t_A\}$, are fit to the estimated transition matrix using least squares. Fourth, the initial estimates for $\{s_A, \alpha_A, \beta_A, \gamma_A, t_A\}$ are refined using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization algorithm to maximize the forward likelihood of the observed tumor read depths, and copy number profiles are produced from the Viterbi decoding.

Negative Binomial Mean and Variance Calculations

The variance and mean of the negative binomial distribution on read depth are related using a second degree polynomial, defined in Equation 2. A second degree polynomial was chosen to maximize the adjusted R-squared value on real diploid data [4], without over-specifying the model (see Suppl. Second degree polynomial).

To determine the constants $\{a, b, c\}$ for a given set of observed diploid data, we calculate the per window expected mean number of reads and variance as specified in Equation 2. Next, we maximize the likelihood of the observed diploid data using the Nelder-Mead method for optimization to find the optimal values of $\{a, b, c\}$. These constants are then used for tumor emission probability calculations (see Suppl. Learning constants $\{a, b, c\}$ for technical optimization details).

Library Size Scaling Factors

Because each sequenced cell will have a different total number of reads, the expected number of reads for each cell needs to be scaled accordingly. Notably, we calculate these cell specific library size scaling factors in a way that accounts for changes in the distribution of reads across the genome caused by CNAs. From Equation 1, let T_A = total # reads in tumor cell A (across n windows), such that

$$\mathbb{E}(T_A) = \sum_{i=1}^{n} \left[\left(\rho_{iA} \times \frac{\mu_i}{2} \right) \times s_A + \varepsilon \right] \tag{7}$$

$$\hat{s}_A = \frac{T_A - n\varepsilon}{\sum_{i=1}^{n} \left(\rho_{iA} \times \frac{\mu_i}{2} \right)} \tag{8}$$

We define ρ_{iA} as copy number in the ith window from cell A's Viterbi decoding path, updated after each iteration of the Baum-Welch algorithm, such that the library size scaling factor estimate continually incorporates changes in estimated copy number across the genome. Initial estimates of s_A are based on the ratio of tumor and average diploid library sizes, and updated according to Equation 8 in subsequent iterations of the Baum-Welch algorithm.

Because s_A estimation can get stuck in local minima, we use several initial estimates of $s_{A,initial,h}; h \in \{1, 2, 3\}$. The first is set to $s_{A,initial,1} = \frac{\text{cell } A \text{ library size}}{\text{average diploid library size}}$. Subsequent starting points are set to $s_{A,initial,2} = 2 \times s_{A,final,1}, s_{A,initial,3} = 4 \times s_{A,final,1}$. Skipped and rarely visited intermediate copy number states (for example, overwhelmingly observing even copy number states genome wide, with odd states observed at 0 or near 0 frequencies) are hallmarks of a local minima for s_A. Estimates of $s_{A,final}$ that display this pattern are excluded, and the $s_{A,final}$ estimate with the highest likelihood is used (see Suppl. Library Size Scaling Factor Selection for further details on filtering $s_{A,final}$ values).

Modified Baum-Welch

We use the standard Baum-Welch algoritm to estimate the transition matrix and initial probability vector, resulting in unconstrained estimates of the transition matrix and initial probability vector. However, we do not use Baum-Welch to directly estimate an emission probability matrix, as emission probabilities are governed by Equation 1, which is only affected by s_A estimates (calculated by Equation 8; see Library Size Scaling Factors).

Next, we fit our model parameters, $\{\alpha_A, \beta_A, \gamma_A, t_A\}$ to the estimated transition matrix using least squares to minimize the sum of squared errors between the Baum-Welch estimated transition matrix and the transition matrix determined by the model parameters.

BFGS Parameter Estimation and Inferring CNAs

Given estimates from previous steps, the parameters $\{s_A, \alpha_A, \beta_A, \gamma_A, t_A\}$ are refined for each tumor cell A, by maximizing the log likelihood (calculated using the Forward Algorithm) us-

ing the BFGS optimization algorithm, an unconstrained quasi-Newton optimization method that approximates the second derivative of the log likelihood by iteratively calculating the gradient [57] (see Suppl. BFGS Technical Details for technical details on BFGS implementation).

Finally, the most likely copy number sequence for each cell is reported using the Viterbi decoding algorithm.

We note that some of the heuristics described in the previous sections could be avoided using a full likelihood estimation using BFGS without the intermediate step of an unconstrained Baum-Welch optimization. However, we find that such optimization is slower, as the Baum-Welch optimization is substantially faster than the BFGS optimization. Additionally, using model parameters fitted to the Baum-Welch results using least squares, without BFGS refinement, results in inaccurate CNA calling, thereby showing the importance of well estimated model parameters (see Suppl. Importance of Model Parameter Accuracy). Finally, using multiple starting points, as described above, was found to be necessary to avoid the optimization getting stuck in local, but not global, optima.

Real Data Preprocessing

We applied SCONCE to two published data sets aligned to hg19 (which was discretized into nonoverlapping 250kb uniform bins using bedtools [58]). The first consists of 34 diploid cells (as determined by cell sorting), and 4 tumor subpopulations (24, 24, 4, and 8 cells, respectively) from one triple negative breast cancer patient [4], a cancer type with prevalent CNAs [59]. The second consists of 10k cells across 5 sections of one triple negative ductal carcinoma sample [5]. Section A was treated as the diploid sample, as determined by [60]. Standard data preprocessing and quality control steps were used to prepare the raw data (see Suppl. Real Data Preprocessing Steps for details). For both real and simulated data sets, we used the averaged diploid cells to calculate the negative binomial distribution constants, $\{a, b, c\}$, and as the matched normal sample to determine the somatic copy number for each tumor cell.

Other methods

In order to evaluate the accuracy of the inference procedure, we compared SCONCE to HMMcopy [43, 44], CopyNumber [45], DNAcopy [46, 47], AneuFinder [49, 50], and SCICoNE [52]. We limited our comparison to methods that have previously been used on the [4] dataset [48], and that, similarly to SCONCE, do not require bam files or SNPs. See 2.8 for full details for running each method.

2.4 Results

GC content and mappability

Because GC content and sequence mappability can bias read distributions, many methods explicitly incorporate corrections for GC content and sequence mappability. However, any technical noise that would affect the tumor sequencing would also affect the diploid sequencing obtained using the same technology, so in SCONCE, these corrections are already directly accounted for in our emission probabilities via the diploid mean.

To verify this, we examined prediction accuracy of expected tumor read counts per window with different amounts of information. For window i, let μ_i be the mean diploid read count, ζ_i be the GC content, and η_i be the mappability from the Duke Uniqueness of 35bp Windows from ENCODE/OpenChrom (UCSC accession wgEncodeEH000325) [61, 62]. For each tumor cell, A, from the previously published data in [4], we predicted the ith window tumor read depth, x_{iA}, using various linear regressions on $\{\mu_i, \zeta_i, \eta_i\}$, then calculated the sum of squared errors (SSE) between predicted and actual tumor read depths. Boxplots of the SSE per cell are shown in Figure 2 and empirical cumulative distribution function (ECDF) plots are shown in Suppl. Figure S4 for A: $x_{iA} \sim \mu_i$, B: $x_{iA} \sim \mu_i + \zeta_i$, C: $x_{iA} \sim \mu_i + \eta_i$, D: $x_{iA} \sim \mu_i + \zeta_i + \eta_i$, E: $x_{iA} \sim \zeta_i$, F: $x_{iA} \sim \eta_i$, G: $x_{iA} \sim \zeta_i + \eta_i$, .

The sum of squared errors remains consistently low across models that incorporate the diploid mean (models A, B, C, and D), and have overlapping ECDF plots, while the SSE increases for models that depend solely on GC content and mappability (models E, F, and G). Because adding the GC content and mappability did not perform significantly differently from the diploid mean alone (two sample KS-test on the cumulative distribution of SSE, $D = 0.033333, p\text{-value} = 1$), we conclude that using the diploid mean is sufficient, and do not add GC or mappability corrections. This conclusion is robust to changes in window size and binning method (ie. uniformly sized bins vs variably sized bins with equal numbers of uniquely mappable bases).

Absolute Copy Number Accuracy

To compare the accuracy of each copy number calling method, we compared the absolute copy number accuracy, scaled copy number accuracy, and breakpoint accuracy across eleven simulated data sets. For brevity, representative simulation data sets are shown in Figure 3, and accuracy results across all simulation sets are shown in Suppl. Figures S5, S6, and S7. Recall that these data sets were simulated under a more realistic non-Markovian model (described in Simulations) that differs from any of the models compared here, including SCONCE. There is, therefore, no reason to presume that the results are particularly biased towards favoring SCONCE because of a match between estimation and simulation model assumptions.

To measure absolute copy number accuracy, we calculated the sum of squared errors (SSE) between true copy number and estimated copy number for each cell and method across

all windows. Because CopyNumber and DNAcopy do not output absolute copy number calls, their results were optimally scaled and shifted to minimize SSE. Additionally, DNAcopy does not output any calls in regions of 0 reads, so these regions were excluded from all SSE calculations for DNAcopy. Overall, SCONCE has similar or lower error rates than AneuFinder, and consistently significantly lower error rate than CopyNumber, DNAcopy, HMMcopy, and SCICoNE (Figure 3).

For example, in Simulation Set H (consisting of many very short and spiky events per cell under the binned model, described in Suppl. Table S2H and Suppl. Figure S1H; Figure 3A), the median SSE for SCONCE is 579.00, 67.00, and 66.50, for $k = 5, 10, 15$, respectively. Of note, because SCONCE cannot call copy numbers above the user specified k, its error rate is significantly higher when the true simulated copy number is greater than k (for example, in the $k = 5$ case). The median SSE values for $k = 10, 15$, however, are lower than the median SSE of 197.00 for AneuFinder. Meanwhile, the median SSE values for HMMcopy and SCICoNE were 2530.50 and 2226.00, respectively, while the scaled median SSE values for CopyNumber (in individual and multisample modes, respectively) were 2510.12 and 2475.81, and scaled median SSE of 1530.52 for DNAcopy.

In Simulation Set G (made of many overlapping CNAS with maximum $k = 8$ under the binned model, described in Suppl. Table S2G and Suppl. Figure S1G; Figure 3B), SCONCE with $k = 10, 15$ outperforms all other methods, with median SSE values of 136.00 and 148.00, respectively. As expected, SCONCE with $k = 5$ has a higher median SSE of 9056.50 as its inference is limited by the maximum k value. Meanwhile, AneuFinder, HMMcopy, and SCICoNE have median SSE values of 222.50, 30378.50, and 27999.00, respectively. DNAcopy and CopyNumber (in individual and multisample modes) have scaled median SSE values of 5265.23, 6841.98, and 6741.10.

In Simulation Set C (consisting of mainly deletions under the line segment model, described in Suppl. Table S1C; Figure 3C), AneuFinder and HMMcopy have scaling problems, while SCONCE does not. The median SSE values for SCONCE are 10.96, 11.09, and 11.11 for $k = 5, 10, 15$, but the median SSE values for AneuFinder and HMMcopy are 6545.67 and 39974.92. Both AneuFinder and HMMcopy tend to incorrectly double copy number estimates. The scaled median SSE values for DNAcopy and CopyNumber (in individual and multisample modes) were 947.50, 200.56, and 194.97. Of note, SCICoNE could not detect any breakpoints in this data set and so could not completely run to produce any copy number profiles.

Scaled Copy Number Accuracy

To check if the differences in median SSE between methods were due to scaling issues, we applied the previously described scaling and shifting procedure to minimize the SSE between true simulated copy number and estimated copy number for all methods. The median SSE values for CopyNumber and DNAcopy did not change here, as their outputs were already scaled and shifted. With this optimal rescaling, SCONCE consistently outperforms or is on par with other methods.

Although the median SSE for SCONCE with $k = 5$ in Simulation Set H (Suppl. Table S2H and Suppl. Figure S1H) decreases from 579.00 to 549.98, rescaling does not address the underlying upper limit on copy number as determined by k (Figure 3D). Similarly, under Simulation Set G (Suppl. Table S2G and Suppl. Figure S1G), rescaling SCONCE with $k = 5$ causes the median SSE to drop from 9056.40 to 5883.34, but it doesn't address same the root problem (Figure 3E). The median SSEs for the other methods for Simulation Set H and G (Figure 3D,E) also decrease, but not significantly.

In contrast, the median SSE values for AneuFinder and HMMcopy for Simulation Set C (Suppl. Table S1C) drops significantly from 6545.67 to 8.18, and from 39974.92 to 3377.62, respectively, while the median SSE for SCONCE changed only slightly, to 10.88, 11.06, and 11.09 for $k = 5, 10, 15$. This shows AneuFinder's high median SSE values for Simulation Set C were due to incorrect scaling, rather than incorrect breakpoint detection and segmentation. However, although HMMcopy's median SSE value dropped by an order of magnitude by from rescaling, the remaining high median SSE value implies other issues remain, such as poor breakpoint detection.

Breakpoint Detection Accuracy

In order to evaluate program accuracy without the confounding factors of absolute or scaled copy number estimates, we compared the breakpoint detection accuracy between each program, by measuring the total distance between true and inferred breakpoints, penalized by the number of inferred breakpoints relative to the number of true breakpoints. Specifically, for each true breakpoint, we calculated the distance to the nearest inferred breakpoint in either direction, and summed this distance across the genome. Because inferring many false positive breakpoints would artificially decrease this breakpoint distance, we defined ω as

$$\omega = \frac{\#\text{inferred breakpoints}}{\#\text{true breakpoints}} \tag{9}$$

such that lowest total breakpoint distance and ω values closest to 1 indicate highest breakpoint detection accuracy.

Across simulation sets, SCONCE consistently has ω values closest to 1 and total breakpoint distances that are lower or on par with other methods. For example, in Simulation Set H (Suppl. Table S2H and Suppl. Figure S1H; Figure 3G), ω values for SCONCE for $k = 5, 10, 15$ all cluster near 1, with median ω values of 0.9302, 0.9321, and 0.9321, respectively. Median ω values for AneuFinder, DNAcopy, CopyNumber (in individual and multisample modes), HMMcopy, and SCICoNE, are 0.7330, 0.7828, 0.5067, 0.5045, 0.3408, and 0.5451, respectively. Additionally, SCONCE has the lowest median total breakpoint distance across $k = 5, 10, 15$ values (1028.0, 1013.5, and 1020.0), while median total breakpoint distances for other programs (in the same order as above) are 3119.5, 3253.5, 9255.5, 9501.0, 13941.0, and 20087.0. Of note, although SCONCE with $k = 5$ had a higher median SSE value than AneuFinder for this data set because many true copy numbers were above

5 (Figure 3A), SCONCE still outperformed AneuFinder in terms of breakpoint detection accuracy.

Furthermore, in Simulation Set G (Suppl. Table S2G and Suppl. Figure S1G; Figure 3H), SCONCE has median ω values closest to 1 for $k = 10, 15$ (0.9499 and 0.9441) and lowest median total breakpoint distances (185.0 and 209.5). However, for $k = 5$, SCONCE is unable to detect additional copy number changes for regions with copy number greater than 5, leading to median $\omega = 0.8908$ and median distance of 1468. AneuFinder, DNAcopy, CopyNumber (in individual and multisample modes), HMMcopy, and SCICoNE had median ω values of 0.9, 0.9889, 0.8983, 0.9278, 0.6034, and 0.9444, respectively, and median total breakpoint distances of 287, 568, 1904.5, 1608, 3834, and 7439.

Additionally, in Simulation Set C (Suppl. Table S1C; Figure 3I), SCONCE (for $k = 5, 10, 15$), AneuFinder, and DNAcopy have similar median ω values of 0.49, 0.49, 0.4911, 0.4828, and 0.431. CopyNumber (in individual and multisample modes) has higher median ω values of 0.649 and 0.6638, while HMMcopy has a median ω value of 0.1798. Despite CopyNumber's better ω values, it has much worse median total breakpoint distances (2629 and 2533) than SCONCE (253, 253, and 251.5 for $k = 5, 10, 15$) and AneuFinder (301). DNAcopy has a similar median total breakpoint distance of 3299, while HMMcopy is an order of magnitude worse, at 23925. Due to the absence of copy number calls, SCICoNE is excluded from this panel. Of note, the similar results between SCONCE and AneuFinder are consistent with AneuFinder performing poorly (Figure 3C) in this setting mostly due to scaling problems. In contrast, these results suggest a combination of scaling and breakpoint errors lead to HMMcopy's poor performance.

Full plots and tables of median ω and median breakpoint distances across all simulation data sets are given in Suppl. Breakpoint Distance and ω Plots and Tables.

Genome wide decodings

By plotting the genome wide copy number profile for a representative cell from each simulation set, we can learn more about the specific differences between methods that lead to differing error rates. For brevity, only genome decodings for SCONCE (with $k = 10$) and AneuFinder are shown in the main text, as AneuFinder consistently performed the best out of other methods (see Suppl. Figure S8 for decodings with other programs and other values of k for SCONCE across all datasets).

SCONCE is more sensitive to small CNAs. For example, for cell 54 in Simulation Set G (described in Suppl. Table S2G and Suppl. Figure S1G; Figure 4A), SCONCE correctly identifies small CNAs that AneuFinder and other methods miss, on chromosomes 10 (right arrow), 11, 12, and 15, ranging in size from 6-13 windows (comparisons to other methods are shown in Suppl. Figure S8G). In one cell from Simulation Set H (described in Suppl. Table S2H and Suppl. Figure S1H; Suppl. Figure S8H), SCONCE has a total breakpoint distance at least one order of magnitude smaller than all other methods and ω value closest to 1. In particular, CopyNumber and SCICoNE only call about half as many breakpoints as necessary, while HMMcopy only calls about a third, resulting in high breakpoint distances

for all three methods. DNAcopy and AneuFinder have similar total breakpoint distances and predict about three quarters of the necessary breakpoints, but still struggle to call small events.

A similar effect plays out in the real data. For example, by examining cell SRR053675 from the [4] dataset in Suppl. Figure S9B, small CNAs (between 5 and 22 250kb windows in length, on chromosomes 9, 10, 12, 13, and 18) are consistently missed by other methods, while SCONCE calls these. Additionally, for the cell with barcode AAACCTGGTTCTTTGT-1 from the [5] data set, shwon in Suppl. Figure S9C, SCONCE detects copy number events on chromosomes 6, 10, 13, 17, 21, and 22 that are not detected by other methods, with sizes ranging from 5 to 20 windows.

SCONCE also calls CNA breakpoints closer to the true breakpoints. In cell 54 from Simulation Set G (Figure 4A), SCONCE detects breakpoints more accurately than AneuFinder (arrows on chromosomes 2, 7, 8, and 10 (left)), with differences ranging from 3 to 35 windows in size. In cell 95 from Simulation Set J (consisting of many overlapping CNAs, with $k = 8$ and uniform initialization matrix under the binned model, described in Suppl. Table S2J and Suppl. Figure S1J; Suppl. Figure S8J), although AneuFinder, DNAcopy, and CopyNumber all have ω values close to 1, they all have higher total breakpoint distance values than SCONCE (with $k = 10, 15$), resulting from erroneously shifting the boundaries of each CNA. HMMcopy is unable to predict enough copy number events, while SCICoNE predicts too many. In both cases, CNAs are predicted in incorrect positions.

As noted before, the value of k must be set high enough to allow a wide enough copy number range in SCONCE. For example, in cell 54 from Simulation Set G (Suppl. Figure S8G), this limitation can be seen in chromosomes 1, 2, 4, 10, and 17-20, where the true copy number reaches a maximum of 8, but SCONCE's copy number estimates are limited to $k = 5$. However, once k is set large enough, SCONCE accurately predicts the true copy number state.

Additionally, in simulations with mostly deletions (Simulation Set C, under the line segment model, described in Suppl. Table S1C), AneuFinder and HMMcopy consistently and incorrectly double the estimated copy number, leading to high SSE values, while SCONCE does not (Figure 3C, Suppl. Figure S8C). Specifically, AneuFinder and HMMcopy mainly call copy numbers of $\{0, 2, 4\}$, instead of $\{0, 1, 2\}$. As in Scaled Copy Number Accuracy, AneuFinder's scaled SSE values dropped, thereby verifying the existence of a scaling problem. In contrast, HMMcopy's remaining large scaled SSE values are caused by not predicting enough CNAs, resulting in high total breakpoint distances and low ω values.

Furthermore, SCONCE considerably outperforms methods like AneuFinder, DNAcopy, HMMcopy, and SCICoNE in regions of 0 tumor read coverage. By using the diploid null model, we are able to separate between true deletions and areas that have missing data due to sequencing noise, and make the most parsimonious calls rather than assuming copy number 0. For example, AneuFinder consistently predicts copy number 0 for centromeres and telomeres, highlighted with arrows in Figure 4B in the centromeres of chromosomes 1, 9, and 16, and in the telomeres of chromosomes 13, 14, 15, 21, and 22. In all panels of Suppl. Figure S9, DNAcopy completely skips telomeres with no tumor coverage, HMMcopy

occasionally predicts copy number 0 for entire chromosomes when one telomere is missing, and SCICoNE inconsistently predicts copy number 0 for centromeres and telomeres. We note that this problem observed in the real data was not contributing to the performance of these methods in the simulated data, as no regions with missing diploid data were simulated.

2.5 Discussion

CNAs are an important driver in cancer evolution, and accurately detecting them on a single cell level can deepen our understanding of tumorigenesis. In this paper, we derive several models of copy number alterations for inference and simulation. We show that using HMMs derived from models of the evolutionary process that generate CNAs, more accurate inferences of CNA could be obtained. The method for inference based on these models, SCONCE, is available as an open source computer package at https://github.com/NielsenBerkeleyLab/sconce.

One limitation of SCONCE is that it requires data from diploid cells sequenced on the same platform as the tumor cells. While this increases accuracy by accounting for platform specific biases and single cell sequencing errors, it also potentially increases sequencing costs to sequence diploid cells, which may not be directly of interest to investigators. However, diploid single cells are often produced incidentally as a by-product of the tumor sequencing strategy. This is, for example, true for the two real data sets analyzed here In such cases there is no extra cost involved in the use of diploid cells for calibration.

Another limitation of SCONCE is that no allele specific or phasing information is used. Incorporating allele frequency and genotype likelihoods of heterozygous sites can increase confidence and clarity in copy number calls, and is the subject of future work.

One of the key strengths of SCONCE over competing methods is its principled Markovian approximation to the copy number process along the length of the genome. This allows for future interpretations and applications of model parameters to understand tumor evolution. Specifically, SCONCE learns transition rate parameters $\{\alpha, \beta, \gamma\}$, time t, and library size scaling factors, and we note that these evolutionary parameters could potentially be used directly for estimating phylogenies.

Compared to other methods, SCONCE has increased sensitivity in calling very small CNAs, particularly those smaller than 5500kb. Additionally, in cells with substantial copy number losses, SCONCE can accurately create copy number profiles without erroneous copy number doublings. This is due to SCONCE's method of estimating library sizes using the Viterbi decoding to account for how changes in the copy number profile necessarily impact the library scaling factor.

Furthermore, because SCONCE uses the averaged diploid data as a null model, in regions with zero tumor read coverage, it can differentiate between genomic loss and sequencing noise, which other methods can not do. In particular, in regions with diploid coverage but no tumor reads, SCONCE calls copy number 0, and in regions without coverage in either the diploid cells or the tumor cell, SCONCE makes the most parsimonious call. This increases CNA

calling accuracy of hard to sequence regions, such as telomeres, centromeres, and repetitive regions.

In conclusion, we present an accurate and principled evolutionary model for calling copy number alterations in single cell whole genome sequencing of tumors, with implications for broader applications.

2.6 Figures

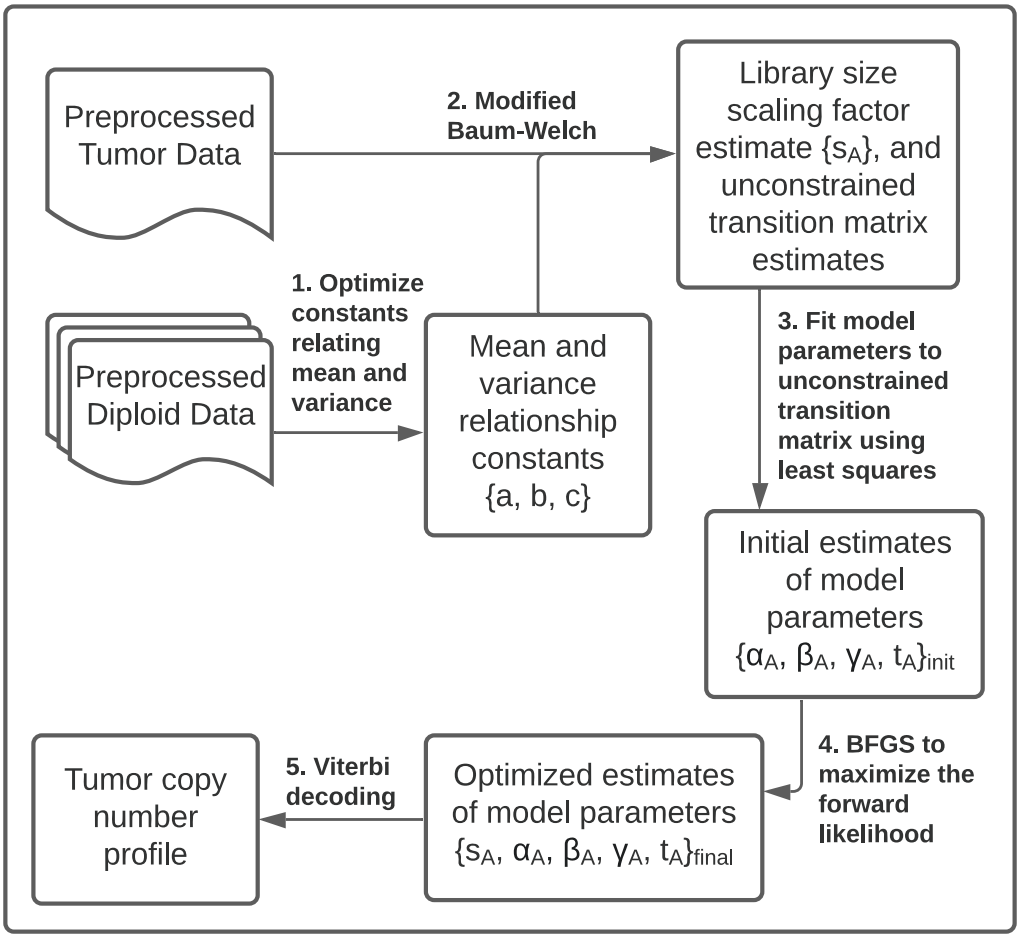


Figure 1: Overview of the SCONCE model training procedure. Tumor and diploid sequencing files must be preprocessed into bed files of read depth per genomic window.

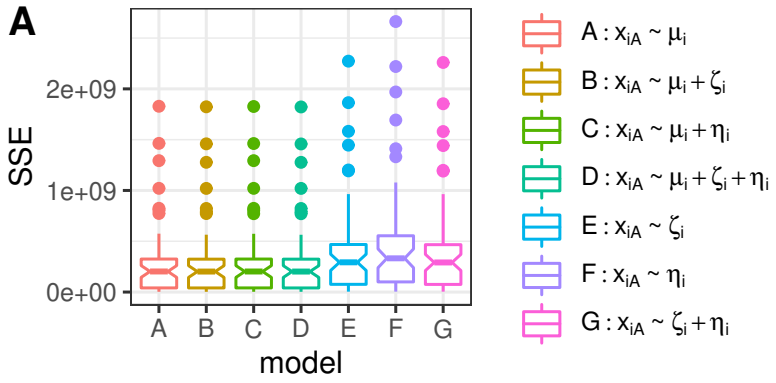


Figure 2: Sum of squared errors (SSE) is shown for each linear regression of observed tumor read depth in window i and cell A (x_{iA}) on mean diploid read depth (μ_i), GC content (ζ_i), and mappability (η_i). SSE is calculated from the differences between the predicted read count and observed read count for each tumor cell in [4] (uniformly sized 250kb bins). No statistically significant difference in error is observed by adding GC or mappability information to the diploid null model.

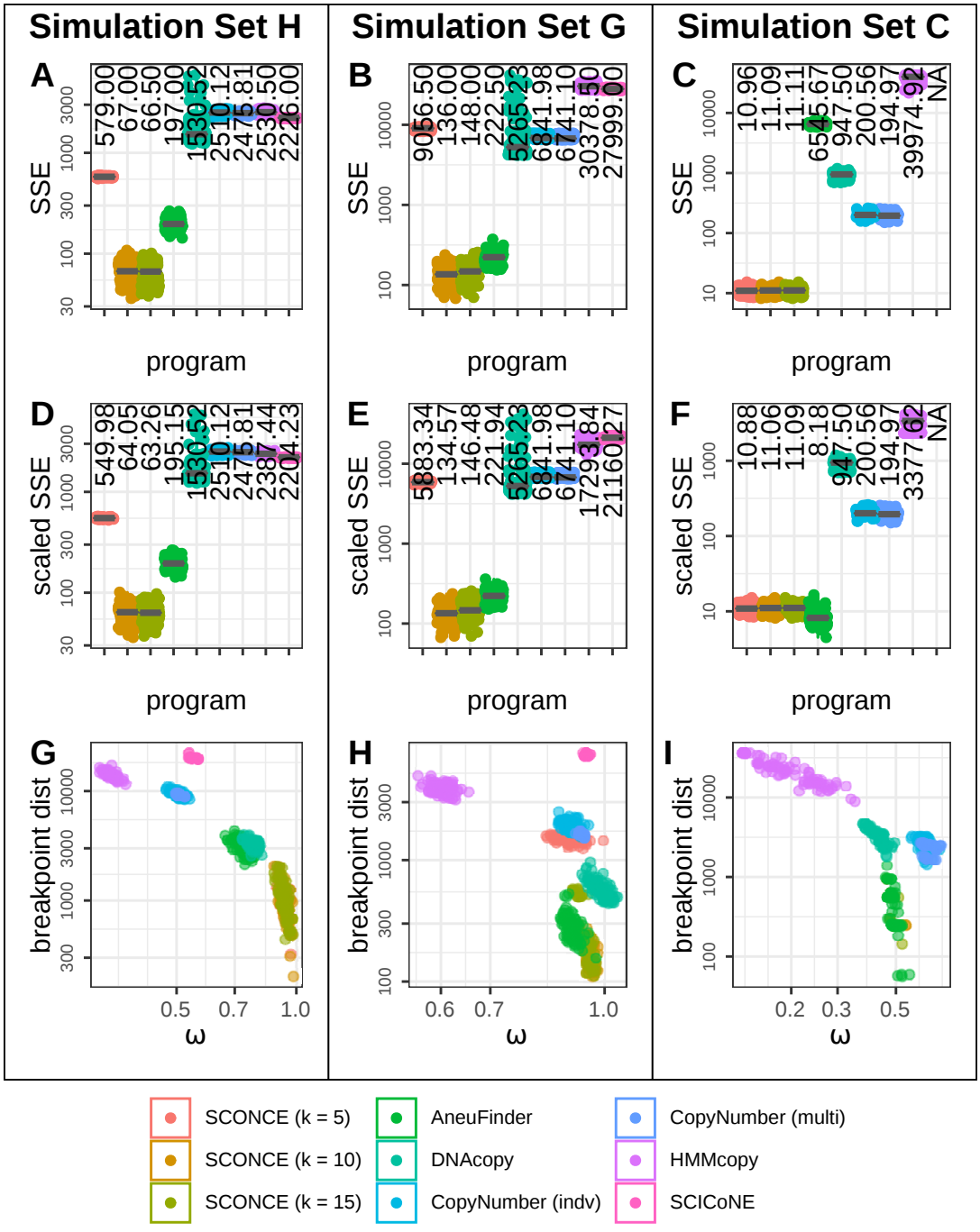

Simulation Set H
Simulation Set G
Simulation Set C
A
B
C
D
E
F
G
H
I
SSE
scaled SSE
breakpoint dist
program
ω
579.00
67.00
66.50
197.00
1530.52
2510.12
2475.81
2530.50
2216.00
9056.50
136.00
148.00
222.50
5265.23
6841.98
6741.10
30378.50
27999.00
10.96
11.09
11.11
654.67
947.50
200.56
194.97
39974.92
NA
549.98
64.05
63.26
195.15
1530.52
2510.12
2475.81
2387.44
2204.23
5883.34
134.57
146.48
221.94
5265.23
6841.98
6741.10
17293.84
21160.27
10.88
11.06
11.09
8.18
947.50
200.56
194.97
3377.02
NA
SCONCE (k = 5)
SCONCE (k = 10)
SCONCE (k = 15)
AneuFinder
DNAcopy
CopyNumber (indv)
CopyNumber (multi)
HMMcopy
SCICoNE

Figure 3: Various accuracy results are shown for three simulation parameter sets, H (consisting of very short and spiky CNAs under the binned model; Suppl. Table S2H and Suppl. Figure S1H), G (many overlapping CNAs with maximum $k = 8$ under the binned model; Suppl. Table S2G and Suppl. Figure S1G), and C (mainly large deletions under the line segment model; Suppl. Table S1C), in the first, second, and third columns, respectively. In the first row, the sum of squared errors (SSE) between simulated ploidy and estimated ploidy is shown across parameter sets. Each dot represents the error for one cell and the median SSE is shown with a gray line and printed at the top of each column. In the second row, the SSE is optimally scaled and shifted for the same data sets for each method to remove errors due to scaling. In the third row, ω (defined in Equation 9) and total breakpoint distance is shown for each method. Each dot represents one cell, colored by method. SCONCE consistently has lower SSE values, ω values closer to 1, and lower total breakpoint distance compared to other methods.

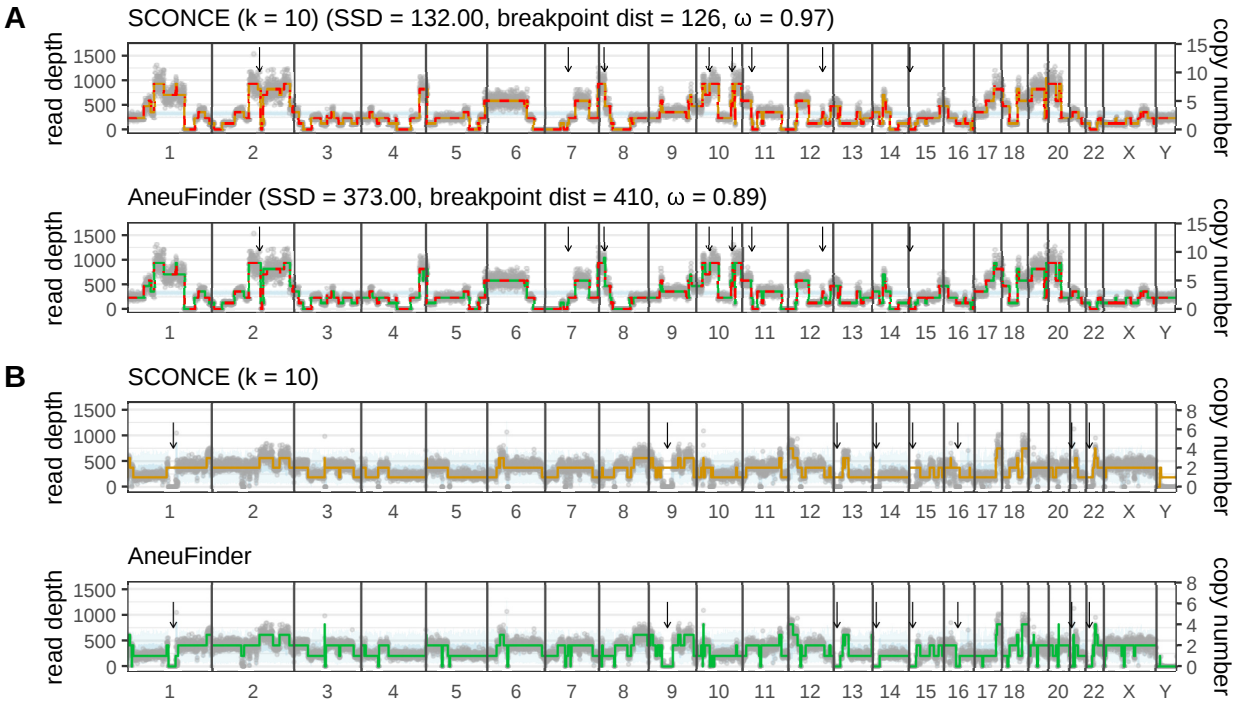


Figure 4: Genome wide copy number decodings are shown for representative cells from simulations and real data. Cell 54 from simulation Set G (many overlapping CNAs with $k = 8$ under the binned simulation model, described in Suppl. Table S2G and Suppl. Figure S1G) is shown in panel A, and cell SRR054570 from [4] is shown in panel B. Genomic window is plotted along the x-axis, per window read depth is shown along the left y-axis, and copy number is plotted along the right y-axis. Black vertical lines denote chromosome boundaries, gray dots represent observed tumor read depth in each window, the red dotted line denotes the true copy number from simulation (where applicable), the light blue line shows the mean diploid read count, the light blue band shows ±1 standard deviation in the diploid read count, and the colored lines denote the copy number decoding from each method. Black arrows highlight regions with differences in CNA calls between SCONCE and AneuFinder. Genome decodings from other methods and additional data sets are shown in Suppl. Additional Genome Traces.

2.7 Appendix

Funding

This work was supported by the National Institutes of Health [R01GM138634-01 to R.N.].

Code Availability

SCONCE is implemented in C++11 and is freely available from https://github.com/NielsenBerkeleyLab/sconce. See Suppl. Code Availability for full details.

2.8 Supplementary Material

Simulations

We provide two simulation models, one based on line segments and one based on bins. The line segment model treats the genome as a line segment to simulate the evolutionary process behind CNAs without assuming any bins, while the binned model divides the genome into discrete bins.

Line Segment Simulation Worked Example

Here we provide a worked example of deletions and amplifications under the line segment model.

Definitions: We define a genome to have a fixed maximal length, L, comprised of a list of chromosomes. A chromosome is an ordered list of line segments,

$$C = \big((b_1, e_1), (b_2, e_2), ..., (b_g, e_g)\big), \quad g \in \mathbb{N}$$

where b_i and e_i are, respectively, the beginning and end positions of the ith chromosomal segments, $b_i < e_i$, $e_i < b_{i+1}$, $b_1 \geq 0$, $e_g \leq L$. Each genome is a set of such chromosomes $G = \{C_1, C_2, ..., C_c\}$, $c \in \mathbb{Z}$. The reference genome in a diploid healthy cell is given by $G_h = \big\{\big((0, L)\big), \big((0, L)\big)\big\}$. The length of a chromosome is $\mid C \mid = \sum_{i=1}^{g}(e_i - b_i)$, and the length of a genome is $\mid G \mid = \sum_{i=1}^{c} \mid C_i \mid$.

Deletions: A deletion removes part of, or the entire segment, for one or more line segments in a single chromosome. For example, a deletion in a chromosome of a healthy diploid cell between positions $d \geq 0$ and $f \leq L$ would results in a new genome $G' = \big\{\big((0, d), (f, L)\big), \big((0, L)\big)\big\}$. If the same chromosome next is hit by a deletion between position i and j with $0 \geq i \leq d$ and $f \geq j \leq L$, the new genome would be $G'' = \big\{\big((0, i), (j, L)\big), \big((0, L)\big)\big\}$. If this chromosome is hit by a deletion with start and end positions 0 and $l < d$, the new genome will then be $G''' = \big\{\big((l, d), (f, L)\big), \big((0, L)\big)\big\}$ and so forth.

Amplifications: An amplification creates an extra copy of part of a chromosome or an entire chromosome. Note there is no maximum ploidy imposed by this model. For example, an amplification starting and ending at positions $m \geq 0$ and $o \leq L$ for in a healthy cell, G_h would result in a genome of composition $G^* = \big\{\big((0, L)\big), \big((0, L)\big)\ \big((m, o)\big)\big\}$. An amplification in the same positions in the first chromosome of genome G' from the previous section would result in $G^{**} = \big\{\big((0, d), (f, L)\big), \big((0, L)\big), \big((m, d), (f, o)\big)\big\}$, if $m < d$ and $o > f$.

Simulation dataset parameter values

For simulations under both the line segment and binned models, all rate parameters were set relative to a genome length of 100. To convert this into the genome was binned into 12,397 bins to match the number of uniform 250kb bins in hg19, the negative binomial parameter r was set to 50, the total number of expected reads, ξ, was set to 4,000,000, and cells

were simulated with even coverage in expectation before accounting for CNAs. Simulation parameter values and dataset descriptions for the line segment and binned models are shown in Table S1 and Table S2, respectively. The rate matrices, Q, for initializing the amplification and deletion processes for the binned model are shown in Supplement Figure S1.

Negative Binomial Mean and Variance Relationship

Second degree polynomial

In order to explore the relationship between λ_{iA} and σ^2_{iA}, we calculated the adjusted R-squared values for three different linear regressions on the mean and variance of the read depth for each window across the diploid cells from [4]:

$$\sigma^2_{iA} = b\lambda_{iA} + c \tag{S1a}$$
$$\sigma^2_{iA} = a\lambda^2_{iA} + b\lambda_{iA} + c \tag{S1b}$$
$$\sigma^2_{iA} = d\lambda^3_{iA} + a\lambda^2_{iA} + b\lambda_{iA} + c \tag{S1c}$$

The regression models in Equation S1 are shown with the mean and variance of per window read depths in Supplemental Figure S2. In panel A, all windows are included in the regressions. In order to ensure windows with extremely low or extremely high mean or variance values were not biasing our analysis, we excluded outlier windows, defined as having a mean or variance value below the 1st quantile or above the 99th quantile, shown in panel B. The adjusted R^2 values are shown in Supplemental Table S3. Going from a linear model, S1a, to a second degree polynomial, S1b increased the adjusted R^2 values in both scenarios, but adding a third degree term, S1c did not substantially change the adjusted R^2 value in either scenario. To avoid overspecifying this model, we used the second degree polynomial.

Learning constants $\{a, b, c\}$

To determine the constants $\{a, b, c\}$, let d_{iA} be the read depth in window i for diploid cell A, such that

$$D = \sum_i \sum_A d_{iA} \tag{S2}$$
$$w_i = \frac{\sum_A d_{iA}}{D} \tag{S3}$$
$$c_A = \frac{\sum_i d_{iA}}{D} \tag{S4}$$
$$\mathbb{E}(d_{iA}) = w_i \times c_A \times D \tag{S5}$$
$$\mathbb{E}(\sigma^2_{iA}) = a\mathbb{E}(d_{iA})^2 + b\mathbb{E}(d_{iA}) + c \tag{S6}$$

where D gives the total number of diploid reads, w_i gives the proportion of total reads in window i, c_A gives the proportion of total reads in diploid cell A, and $\mathbb{E}(d_{iA})$ and $\mathbb{E}(\sigma^2_{iA})$

give the expected mean and variance of the diploid read counts under this second degree polynomial model, respectively.

By parameterizing a Negative Binomial distribution with mean, $\mathbb{E}(d_{iA})$, and variance, $\mathbb{E}(\sigma^2_{iA})$, we can calculate the likelihood of the observed diploid data, for a given set of constants $\{a, b, c\}$. We maximize the likelihood using Nelder-Mead method for optimization to find the optimal values of $\{a, b, c\}$ for the diploid data, and these values are then carried over to the tumor emission probability calculations. See Diploid Cell Processing Calculations and Code for the code.

Library Size Scaling Factor Selection

Because the initial estimate of the library size scaling factor, $s_{A,initial}$, would be erroneously estimated from the Viterbi decoding from an untrained HMM, we instead run Baum Welch from three starting estimates of $s_{A,initial}$.

Additionally, binning the genome into discrete, non-overlapping bins leads to some copy number alterations being split across bin boundaries, as the CNA breakpoints may not align with the bin boundaries. This leads to bins with fractional copy number when CNAs are averaged over the entire bin. Because the HMM is restricted to integer copy numbers, in some cases, the HMM incorrectly attempts to fit these fractional bins to integer copy numbers by doubling all copy numbers and halving the library size scaling factor, since this has a slightly higher forward likelihood than rounding the fractional bins under the true library size scaling factor. For example, given a true copy number sequence of $1 \to 1.6 \to 2$, the decoding $2 \to 3 \to 4, \hat{s}_A = 0.5$ has a slightly higher forward likelihood than the decoding $1 \to 2 \to 2, \hat{s}_A = 1$. This leads to transient bins, where the Viterbi decoding strictly increases or decreases through one state for exactly one bin (ie. observing copy numbers $2 \to 3 \to 4$ across only 3 bins). In datasets with many deletions, this can also lead to steady state distributions where intermediate states are skipped (for example, observing the incorrect steady state distribution $\pi = [0.15, 0, 0.8, 0, 0.05]$ instead of $\pi = [0.15, 0.8, 0.05, 0, 0]$).

To detect and avoid this library size scaling factor misspecification, we use the presence of these transient bins and skipped intermediate states to disqualify library size scaling estimates. Specifically, we first run Baum Welch three times, with different initial values of $\hat{s}_{A,initial}$. For the first run, we set $s_{A,initial,1} = \frac{\text{total \# reads in cell } A}{\text{average \# reads in diploid cells}}$. For the second and third runs, we set $\hat{s}_{A,initial}$ to $\hat{s}_{A,final,1}$ multiplied by 2 and 4, respectively.

Next, we pick the Baum Welch run and $\hat{s}_{A,final}$ estimate that has the highest forward likelihood, unless it has at least five instances of transient states across the genome or has skipped intermediate states. If so, it will be discarded in favor of the run with the next best forward likelihood, if the second best run has no skipped intermediates states, and has less than five instances of transient states or is less than 100 loglikelihood units worse than the best run. This selection procedure corrects for any incorrect estimates of s_A that would lead to ploides of $\{0, 2, 4\}$ instead of $\{0, 1, 2\}$, for example.

BFGS Technical Details

We use the `gsl_multimin_fdfminimizer_vector_bfgs2` function from the GNU Scientific Library (GSL) [63] for BFGS optimization of the loglikelihood from the forward algorithm. Specifically, the likelihood of the observed tumor data is calculated using the forward algorithm and summed across all chromosomes for a given cell, and the HMM is reset to the initial probability vector, defined as the steady state distribution of the Markov chain, at the beginning of each chromosome to maintain chromosomal independence. Here, the central difference approximation is used to calculate the gradient.

To account for instability in small numbers due to machine encoding, the loglikelihood in each position in the sequence is scaled by the maximum of the loglikelihoods in that position, and the final loglikelihood is re-scaled appropriately.

Additionally, BFGS performs an unconstrained optimization, but to have sensible results, we require $\{s_A, \alpha_A, \beta_A, \gamma_A, t_A > 0\}$. As such, we log transform and scale $\{s_A, \alpha_A, \beta_A, \gamma_A, t_A\}$ to constrain the optimization results.

Importance of Model Parameter Accuracy

In order to evaluate the importance of accurately estimating model parameters, we compared the SSE between true and estimated copy number under three different conditions during the model parameter optimization. Specifically, we used the Viterbi decoding to estimate copy numbers at the initial optimization estimate (set arbitrarily to $\{s_A = \frac{\text{total \# reads in cell } A}{\text{average \# reads in diploid cells}}$, $\alpha = 0.1, \beta = 0.01, \gamma = 500, t_A = 0.01\}$), after the Baum Welch and least squares parameter estimation, and after the BFGS parameter estimate refinement (that is, the entire SCONCE pipeline). For Simulation Set A (consisting of many overlapping CNAs, under the line segment), $k = 10$, the median SSE between true and estimated copy numbers dropped from 7383.45 under the initial parameter estimates, to 66.54 after the Baum Welch and least squares steps, and finally to 44.04 after BFGS parameter refinement (Supplemental Figure S3), showing that these model parameters must be well estimated for accurate copy number calls.

Real Data Preprocessing Steps

To prepare the data from [4], reads were trimmed using cutadapt [64] and trimmomatic [65], and low complexity reads were removed using prinseq [66]. Cleaned reads were aligned to hg19 using bowtie2 [67], reads with a q score less than 20 were removed using samtools [68], and duplicates were removed using picard [69].

To prepare the data from [5], a preprocessed bam file for each section was downloaded. These bam files were filtered for reads with q scores greater than or equal to 20 using samtools [68] and split into cell specific bam files using pysam [70].

Finally, bedtools was used to create 250kb uniform windows of hg19, and to calculate cell specific observed read depth per window [58].

Other methods

Scripts to run AneuFinder, HMMCopy, CopyNumber, DNACopy, and SCICoNE are provided on GitHub. Of note, because our simulation model does not incorporate GC or mappability biases into simulated read depths, we did not use GC and mappability corrections on simulated data to avoid overcorrecting, where applicable. For methods that used matched normal samples, we used the averaged diploid read counts. The tool `bedtools intersect` was used to convert large segments to 250kb windows where applicable. Briefly, each method was run as follows.

To run AneuFinder [49, 50] on real data, we ran the `Aneufinder` function with 250,000 binsize, all chromosomes, GC correction, and hg19 assembly. To run AneuFinder on simulated data, we skipped the GC and mappability corrections by running the `findCNVs` function with default parameters (`method="edivisive", R=10, sig.lvl=0.1`). Copy number calls were extracted from the `copy.number` element from the resulting `edivisive` model segments.

With real data, we ran HMMcopy [43, 44] by first doing read correction (`correctReadcount`, default parameters) on both the tumor data and averaged diploid cells, and skipped this step in simulations. We then calculated the log ratio of the normalized tumor and averaged diploid read depths, and the `HMMsegment` function (default parameters) was used to segment each cell. Copy number estimates were extracted from the resulting `state` element -1.

To run CopyNumber [45], the log ratio of the normalized tumor and averaged diploid read depths from HMMcopy preprocessing were used as input. Then, missing data were imputed, using the `constant` method, and the `winsorize` function was used to remove outliers. To run in single sample mode, the `pcf` function was used with parameters `return.est=T, normalize=T, digits=6`, and the exponentiated `estimates` element was extracted for the copy number estimates. To run in multi sample mode, the `multipcf` function was run with parameters `return.est=T, digits=6`, and copy numbers were similarly extracted from the exponentiated `estimates` result.

Similarly, to run DNAcopy [46, 47], the log ratio values of the tumor and diploid data from HMMcopy preprocessing were used as input, and regions with no tumor coverage were removed from these data, as these regions caused DNAcopy to output nonsensical chromosome coordinates. The functions `CNA`, `smooth.CNA`, and `segment`, all with default parameters, were used to segment the genome. Because CopyNumber and DNAcopy do not output absolute copy number calls, we scaled and shifted CopyNumber and DNAcopy results to minimize the sum of squared errors from the true ploidy in simulated datasets to create copy number estimates for comparison purposes. Regions of no tumor coverage were excluded for DNAcopy in these calculations.

To run SCICoNE [52], the subprogram `breakpoint_detection` was run on all tumor data with default parameters, appropriate numbers of cells and bins, and a breakpoints file indicating chromosome boundaries. Next, the script `segment_counts.py` was used to segment the genome, followed by the subprogram `inference` to infer copy number profiles.

GC content and mappability

In order to evaluate the information gain by adding GC content and genome mappability, we compared the empirical cumulative distribution functions across several different linear regression models. As shown in Supplemental Figure S4, no significant difference is seen between models based on the diploid mean alone and models utilizing the diploid mean and mappability.

SSE Error Plots

Unscaled SSE Plots

As in Figures 3A,B,C, for each method, the sum of squared errors between simulated ploidy and inferred ploidy is shown for each parameter set, described in Tables S1 and S2, in Supplemental Figure S5 Of note, in Simulation Set B (whole genome duplication before any CNAs, under the line segment model), SCONCE performs best for $k = 15$, as the duplication results in higher copy number states. Although SCONCE misidentifies some cases here, SCONCE with $k = 15$ still has a lower median SSE than other methods.

Optimally Scaled SSE Plots

As in Figure S5D,E,F, the sum of squared errors between simulated ploidy and estimated ploidy is shown across different parameter sets for each method in Supplemental Figure S6. Here, to eliminate scaling errors, all copy number calls are first scaled and shifted to minimize the sum of squared errors between simulated ploidy and estimated ploidy. Although the overall SSE values decrease with optimal scaling, SCONCE continues to have median SSE values that are lower or on par with other methods.

Breakpoint Distance and ω Plots and Tables

Breakpoint Distance and ω Plots

Similar to Figure 3G, H, and I, total breakpoint distance and ω values (defined in Equation 9) are plotted for all eleven simulation sets across all methods in Supplemental Figure S7.

Breakpoint Distance and ω Tables

For each simulation set, tables are given for median total breakpoint distances (Supplemental Table S4) and median $\omega = \frac{\#\text{inferred breakpoints}}{\#\text{true breakpoints}}$ values (Supplemental Table S5.

Additional Genome Traces

Simulations

Additional genome wide copy number decodings are shown in Supplemental Figure S8 for representative cells across all different simulation conditions, for all evaluated methods. Each panel letter corresponds to the simulation set described in Simulation dataset parameter values. Of note, Simulation Sets H, G, and C appeared in panels {A,D,G}, {B,E,H}, and {C,F,I} in Figure 3, respectively.

Real data

Additional genome wide copy number decodings are shown in Supplemental Figure S9 across methods for representative cells SRR054570 (shown in panel A) and SRR053675 (shown in panel B) from [4].

Furthermore, a genome wide genome decoding for a representative cell, (section C, barcode AAACCTGGTTCTTTGT-1), from [5], is shown in Supplemental Figure S9C.

Code Availability

Diploid Cell Processing Calculations and Code

To create the average diploid read count file, the per window read counts were averaged across all diploid cells. An R script (`avgDiploid.R`) is provided to do this.

The variance for the negative binomial distribution on tumor read counts is determined by a second degree polynomial of the mean (see Negative Binomial Mean and Variance Relationship. The calculations to find optimal values of $\{a, b, c\}$ should be rerun for each dataset, using the provided `fitMeanVarRlnshp.R` script. $\{a, b, c\}$ values from one dataset [4] are supplied as defaults.

SCONCE Prerequisites and Dependencies

SCONCE is implemented in C++11, and requires the Boost C++ Libraries and the GNU Scientific Library (GSL) [63]. SCONCE has been extensively tested on Ubuntu 18.04.6, and is available on GitHub: `https://github.com/NielsenBerkeleyLab/sconce`.

The simulation program for both the line segment and binned models is also available on GitHub. Additional R scripts are provided for intermediate calculations (see Diploid Cell Processing Calculations and Code) and to plot results.

SCONCE runtime and memory requirements

Because k determines the transition matrix dimensions, SCONCE's runtime and memory requirements scale quadratically with k. For example, for Simulation Set A, the median

runtimes for $k = 5, 10, 15$ were 259.30, 2426.05, and 8141.56 seconds (Supplemental Figure S10A). The median maximum memory used was 0.018578, 0.030252, and 0.051274 gigabytes for $k = 5, 10, 15$ (Supplemental Figure S10B). Of note, all simulations were done with the full human genome (hg19). Analyses were run on a server running Ubuntu 18.04.6 with Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz processors and 500 GB RAM.

Supplementary Figures

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.2 & -1 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & -1 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & -1 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & -1 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & -1 \end{bmatrix}$$

(a) Simulation set E

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.01 & -1 & 0.39 & 0.1 & 0.3 & 0.2 \\ 0.1 & 0.2 & -1 & 0.4 & 0.2 & 0.1 \\ 0.1 & 0.2 & 0.25 & -1 & 0.2 & 0.25 \\ 0.2 & 0.1 & 0.2 & 0.3 & -1 & 0.2 \\ 0.1 & 0.3 & 0.2 & 0.2 & 0.2 & -1 \end{bmatrix}$$

(b) Simulation set F

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.21 & -1 & 0.39 & 0.1 & 0.1 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.1 & 0.3 & -1 & 0.3 & 0.1 & 0.1 & 0.05 & 0.025 & 0.025 \\ 0.05 & 0.1 & 0.3 & -1 & 0.25 & 0.2 & 0.05 & 0.025 & 0.025 \\ 0.1 & 0.1 & 0.1 & 0.2 & -1 & 0.2 & 0.1 & 0.1 & 0.1 \\ 0.05 & 0.1 & 0.15 & 0.2 & 0.3 & -1 & 0.1 & 0.05 & 0.05 \\ 0.05 & 0.1 & 0.1 & 0.1 & 0.1 & 0.25 & -1 & 0.2 & 0.1 \\ 0.03 & 0.17 & 0.1 & 0.1 & 0.2 & 0.1 & 0.2 & -1 & 0.1 \\ 0.0 & 0.07 & 0.13 & 0.1 & 0.2 & 0.1 & 0.2 & 0.2 & -1 \end{bmatrix}$$

(c) Simulation set G

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.21 & -1 & 0.39 & 0.1 & 0.1 & 0.05 & 0.05 & 0.05 & 0.05 \\ 0.1 & 0.3 & -1 & 0.3 & 0.1 & 0.1 & 0.05 & 0.025 & 0.025 \\ 0.05 & 0.1 & 0.3 & -1 & 0.25 & 0.2 & 0.05 & 0.025 & 0.025 \\ 0.1 & 0.1 & 0.1 & 0.2 & -1 & 0.2 & 0.1 & 0.1 & 0.1 \\ 0.05 & 0.1 & 0.15 & 0.2 & 0.3 & -1 & 0.1 & 0.05 & 0.05 \\ 0.05 & 0.1 & 0.1 & 0.1 & 0.1 & 0.25 & -1 & 0.2 & 0.1 \\ 0.03 & 0.17 & 0.1 & 0.1 & 0.2 & 0.1 & 0.2 & -1 & 0.1 \\ 0.0 & 0.07 & 0.13 & 0.1 & 0.2 & 0.1 & 0.2 & 0.2 & -1 \end{bmatrix}$$

(d) Simulation set H

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.125 & -1 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & -1 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & -1 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & -1 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -1 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -1 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -1 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -1 \end{bmatrix}$$

(e) Simulation set I

$$\begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.125 & -1 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & -1 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & -1 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & -1 & 0.125 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -1 & 0.125 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -1 & 0.125 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -1 & 0.125 \\ 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & 0.125 & -1 \end{bmatrix}$$

(f) Simulation set J

$$\begin{bmatrix}
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.\overline{09} & -1 & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & -1 & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & 0.\overline{09} & -1 & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & -1 & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & -1 & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & -1 & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & -1 & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & -1 & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & -1 & 0.\overline{09} & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & -1 & 0.\overline{09} \\
0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & 0.\overline{09} & -1
\end{bmatrix}$$

(g) Simulation set K

Figure S1: Amplification and deletion initialization rate matrices for the binned model simulations are shown here for each parameter set.

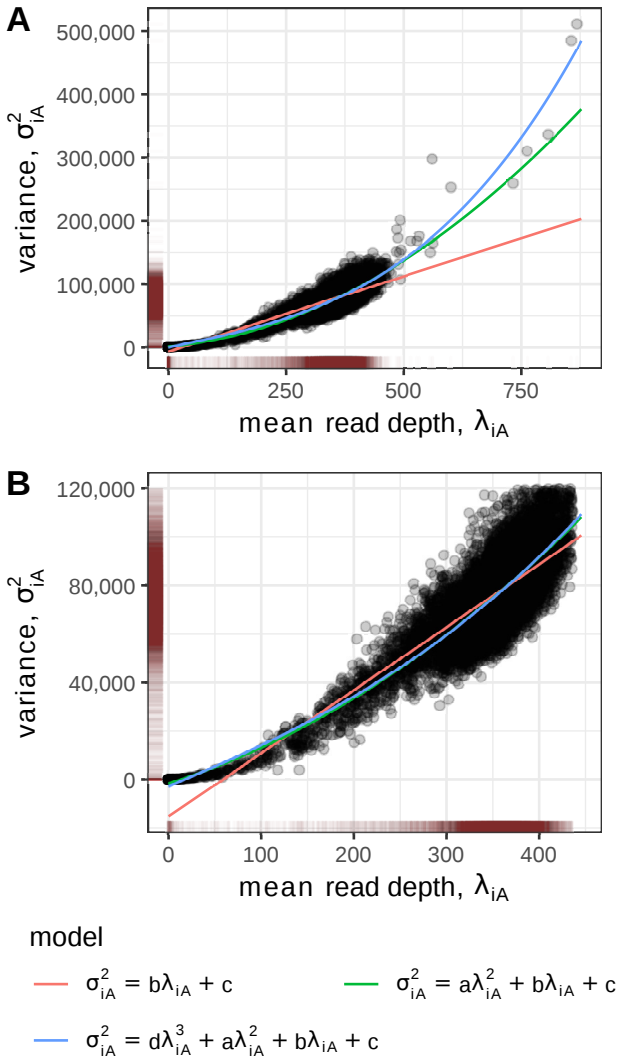


Figure S2: The mean and variance of the read depth for each window in the diploid cells from [4] are shown with three linear regression models. In panel A, all windows are included, and in B, any windows with a mean or variance below the 1st quantile or above the 99th quantile were excluded. Predicted variance values for each mean value are shown for each regression model.

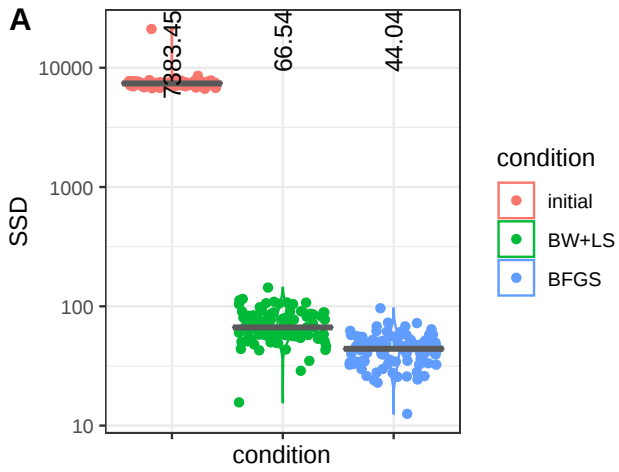


Figure S3: The sum of squared errors between true and estimated copy number is shown for three different conditions (under the initial parameter values, after the Baum Welch and least squares step (BW+LS), and after the BFGS step), for Simulation Set A and $k = 10$.

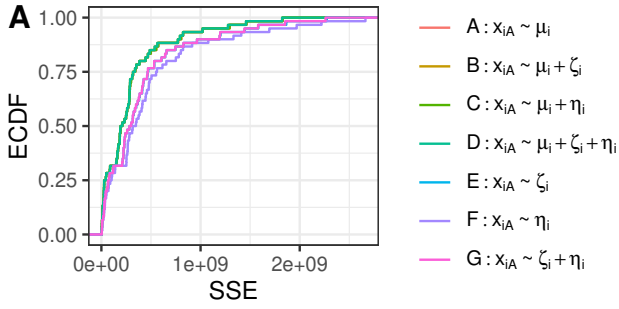


Figure S4: The empirical cumulative distribution function (ECDF) of the sum of squared errors shows no significant differences in adding GC or mappability information. The ECDF lines from models A (diploid mean only), B (diploid mean with GC content), C (diploid mean with mappability), and D (diploid mean with GC and mappability) all lie on top of each other, and the ECDF lines from models E (GC only) and G (GC with mappability) lie on top of each other as well.

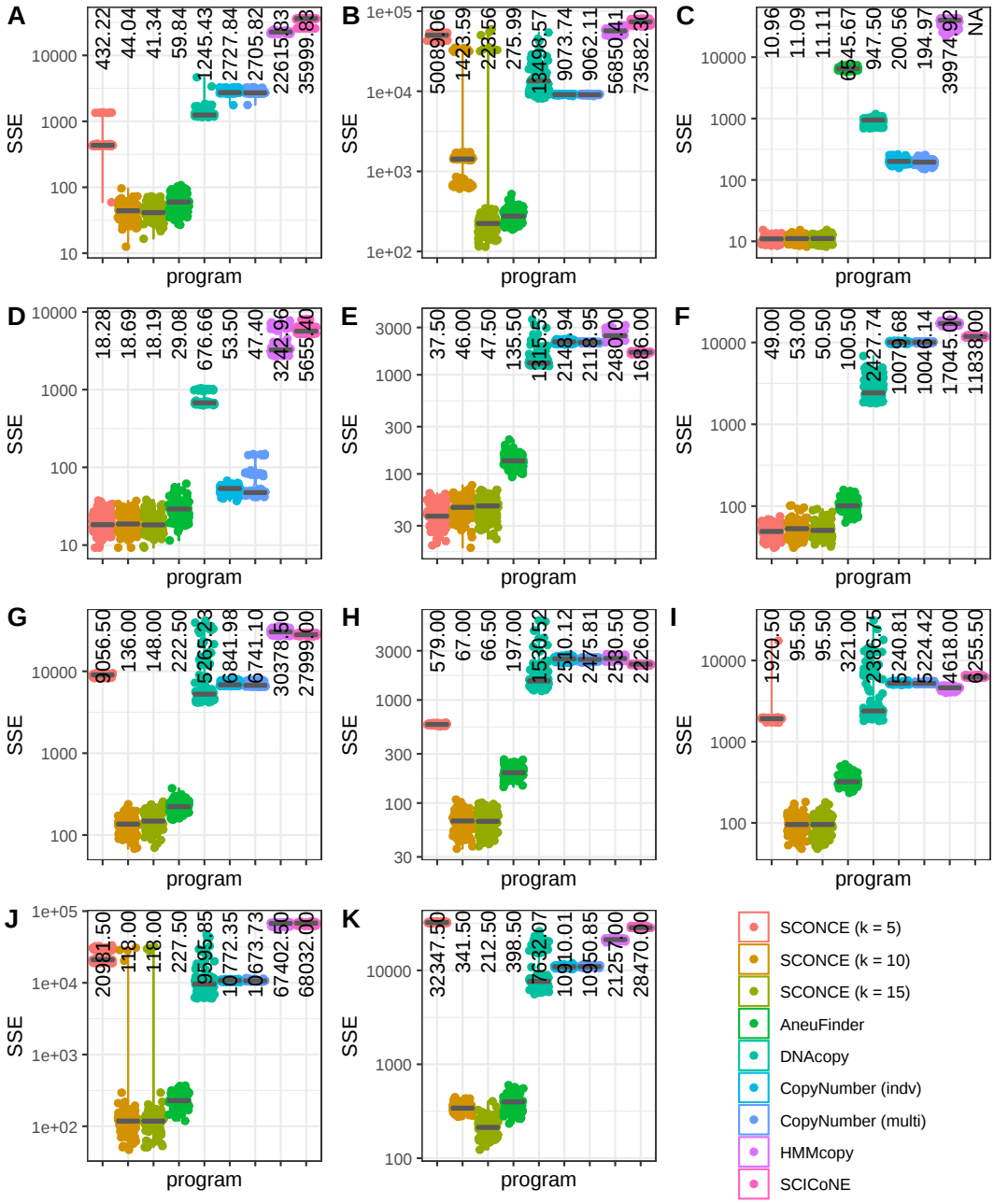


Figure S5: The sum of square errors (SSE) between simulated and estimated copy numbers is shown across parameter sets for each CNA calling program.

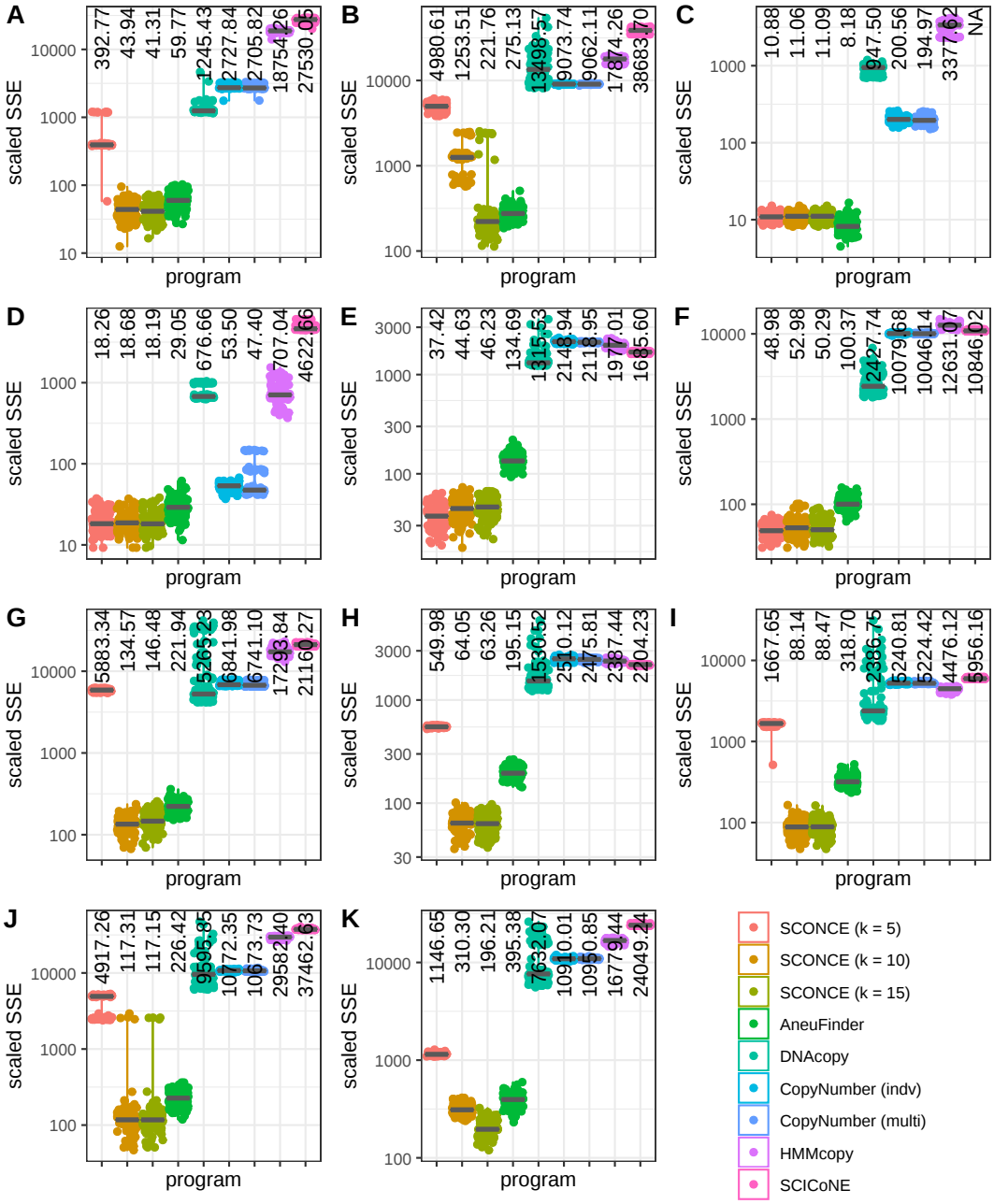


Figure S6: For each CNA program and parameter set, the scaled SSE values between simulated and estimated copy numbers is shown. Of note, all estimated copy numbers have been scaled and shifted to minimize the SSE between simulated and estimated copy numbers, in order to eliminate scaling issues in the estimated CNAs.

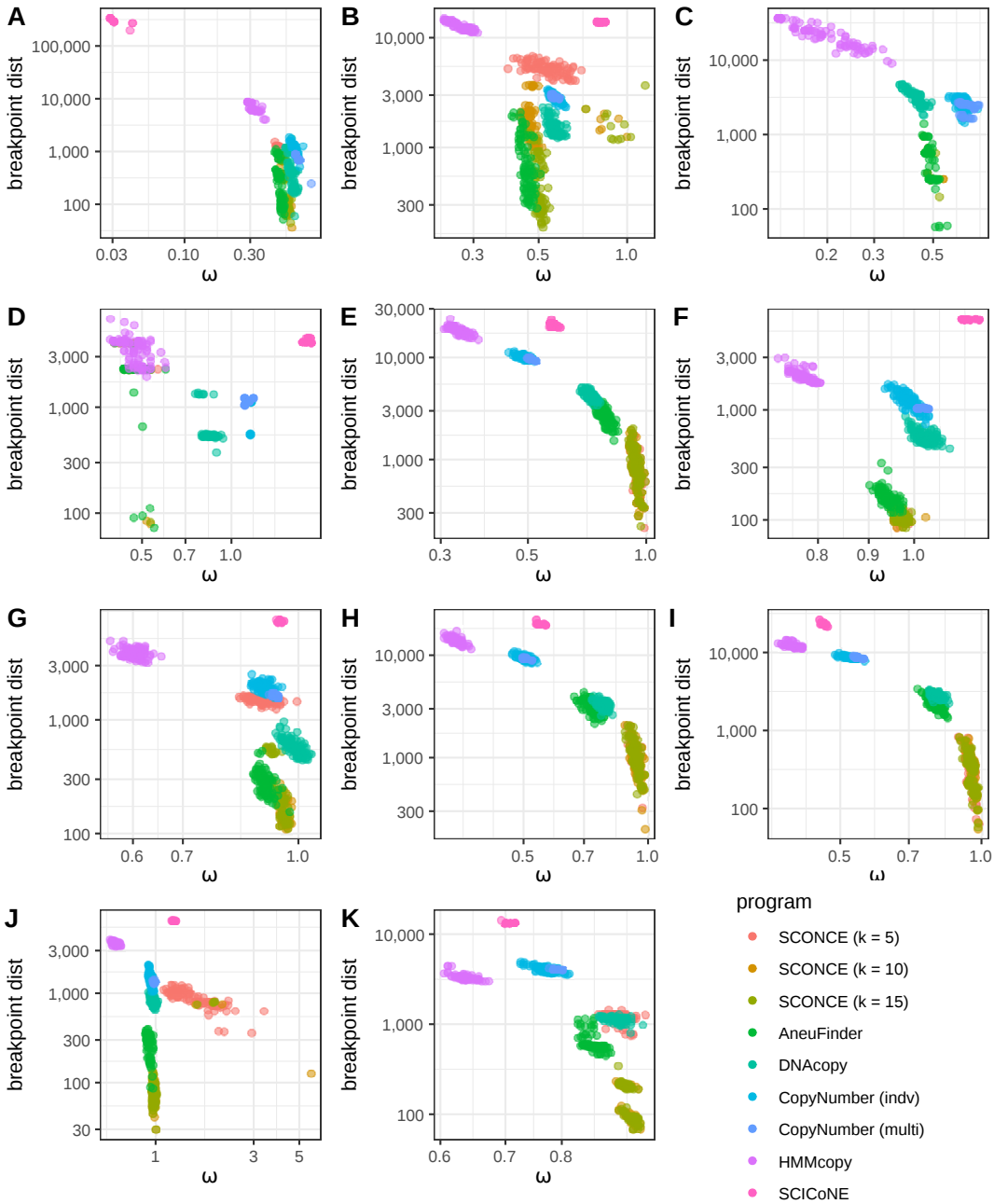
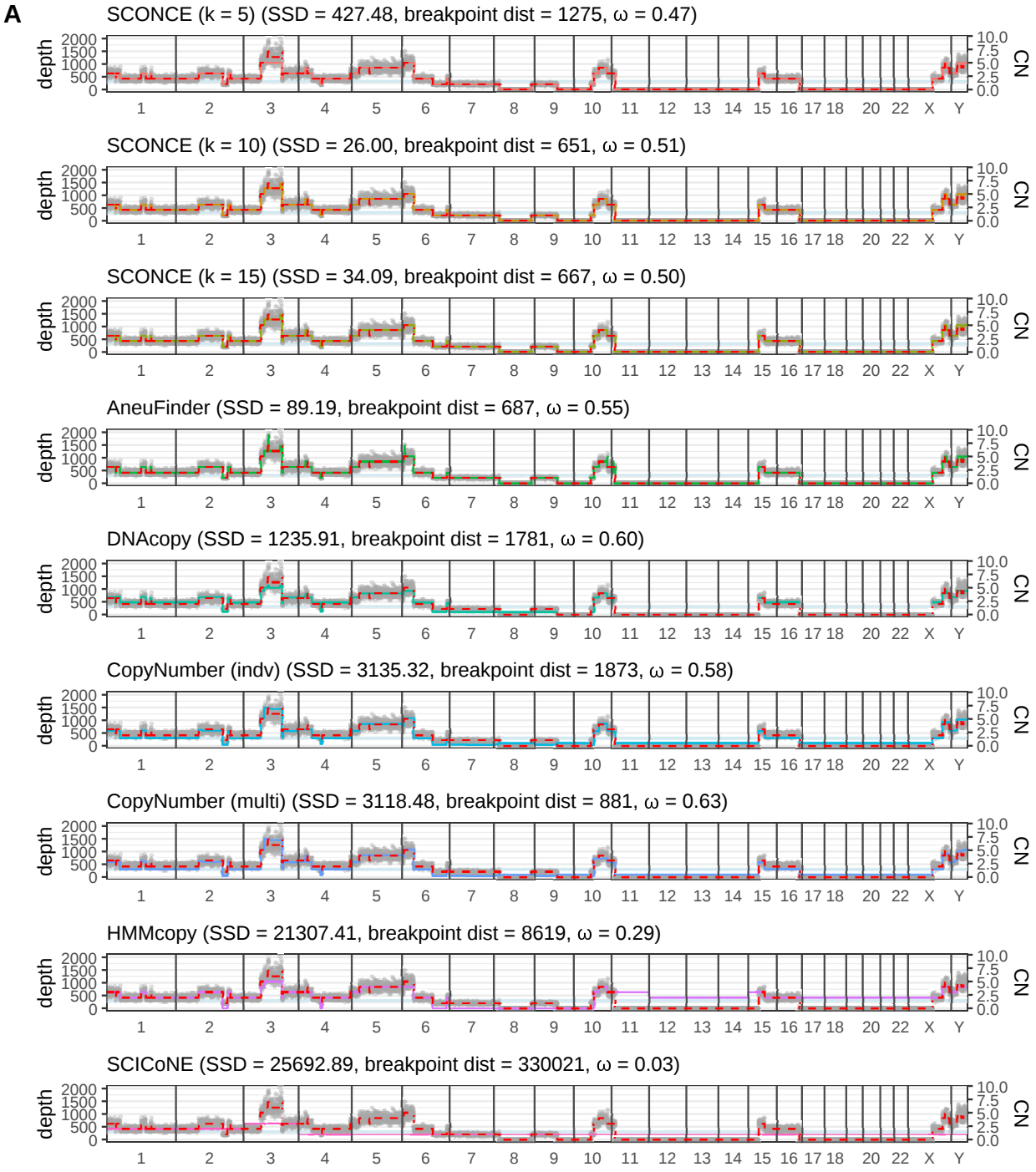
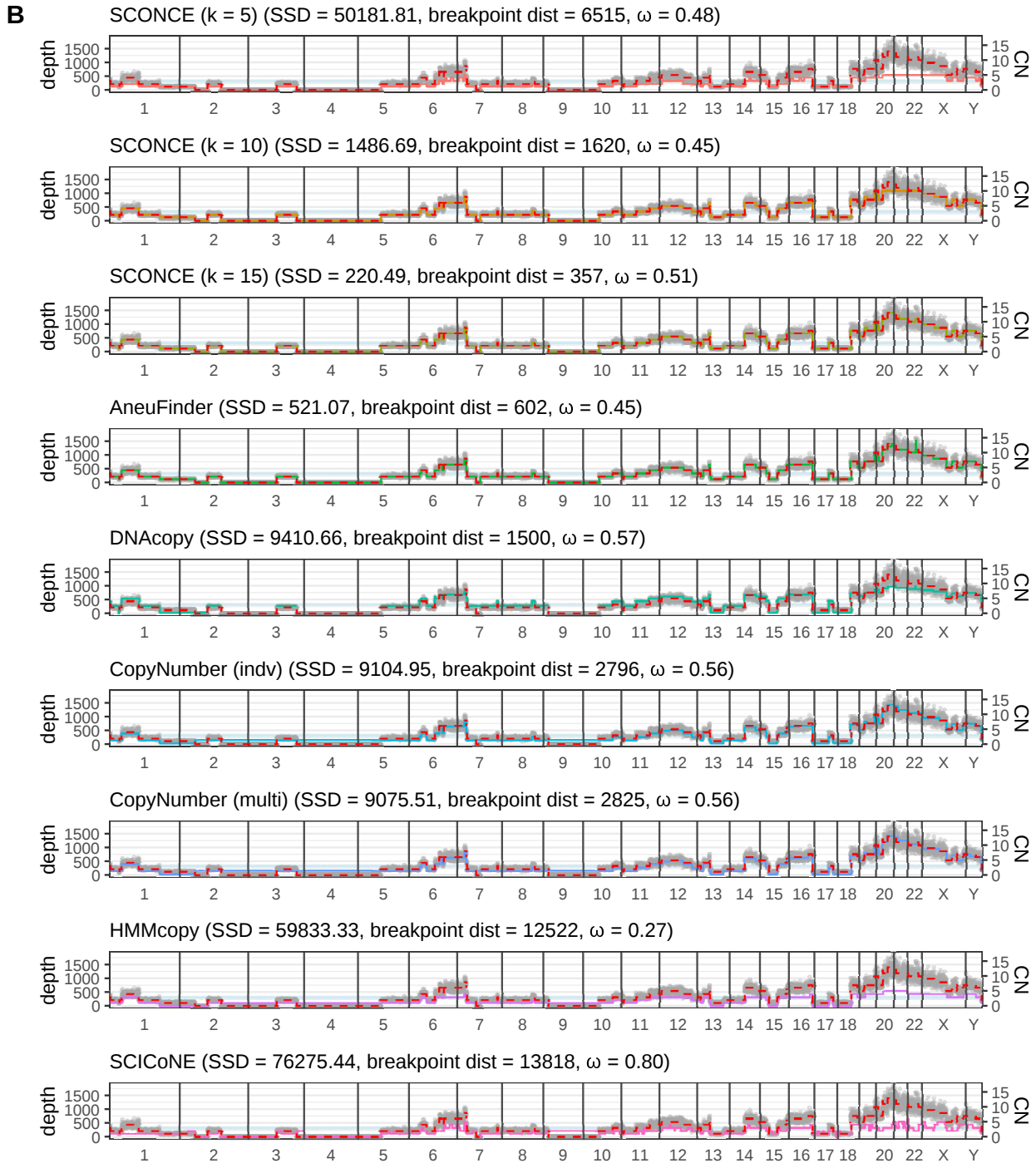


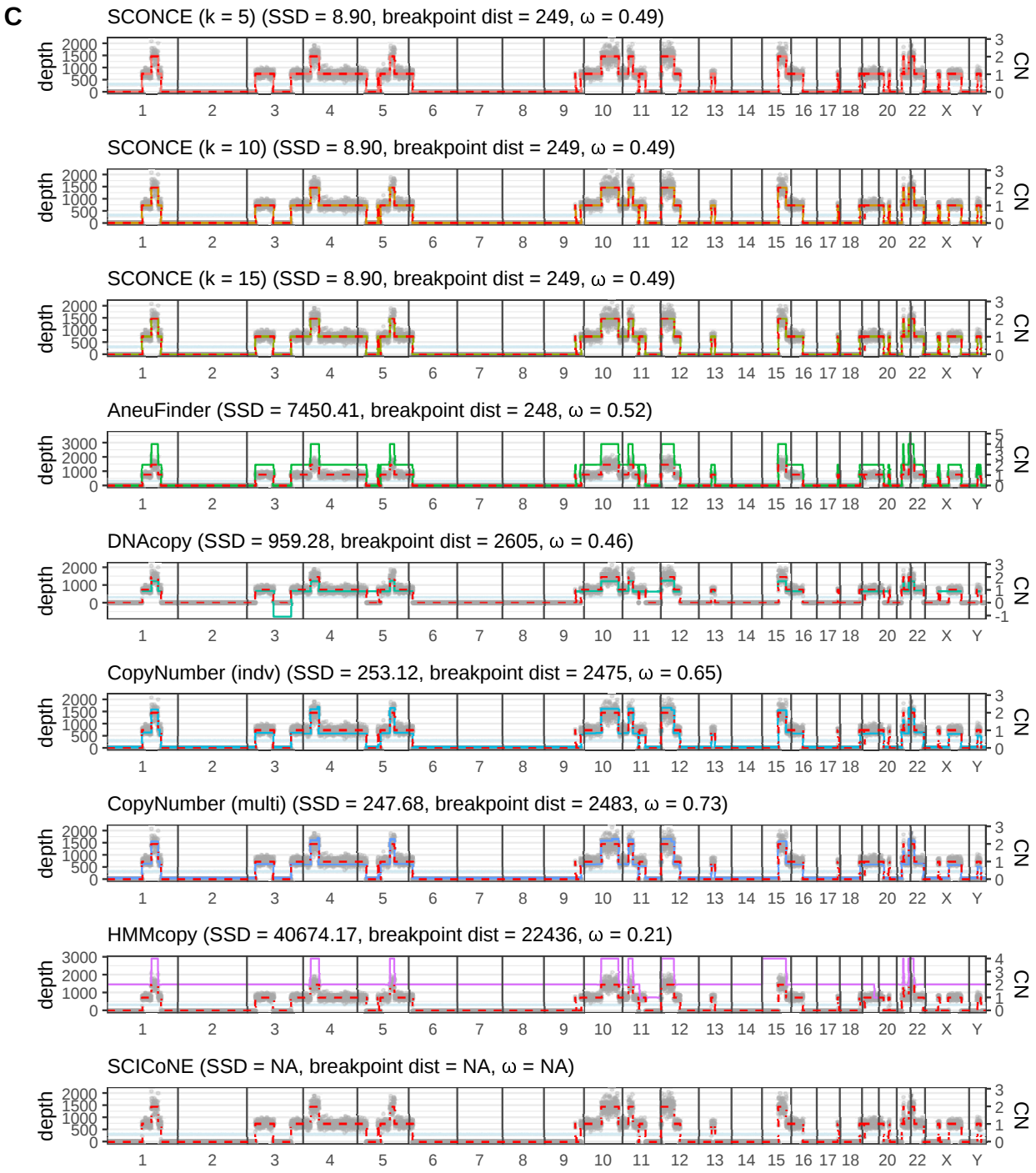
Figure S7: Total breakpoint distance and ω values are plotted for each program and parameter set. Lower total breakpoint distance and ω values closest to 1 represent highest accuracy in detecting breakpoints. Each dot represents one cell in each parameter set, colored by program. Subpanel labels correspond to simulation set names (see Supplement 2.8). With $k = 10, 15$, SCONCE consistently outperforms other methods.



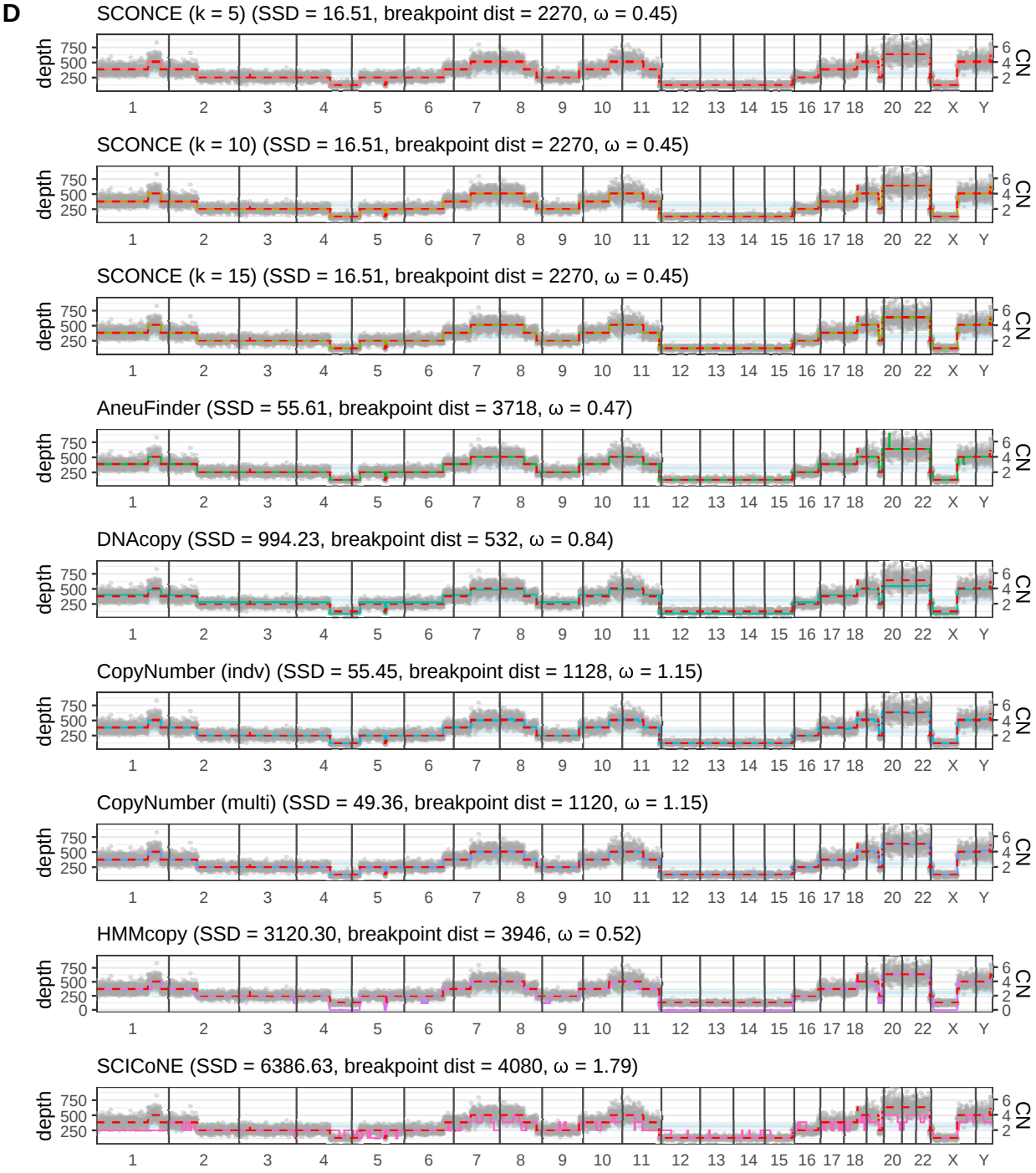
(a) Fig. S8A: Genome wide decoding for cell 14 in Simulation Set A (many overlapping CNAs, under the line segment model).



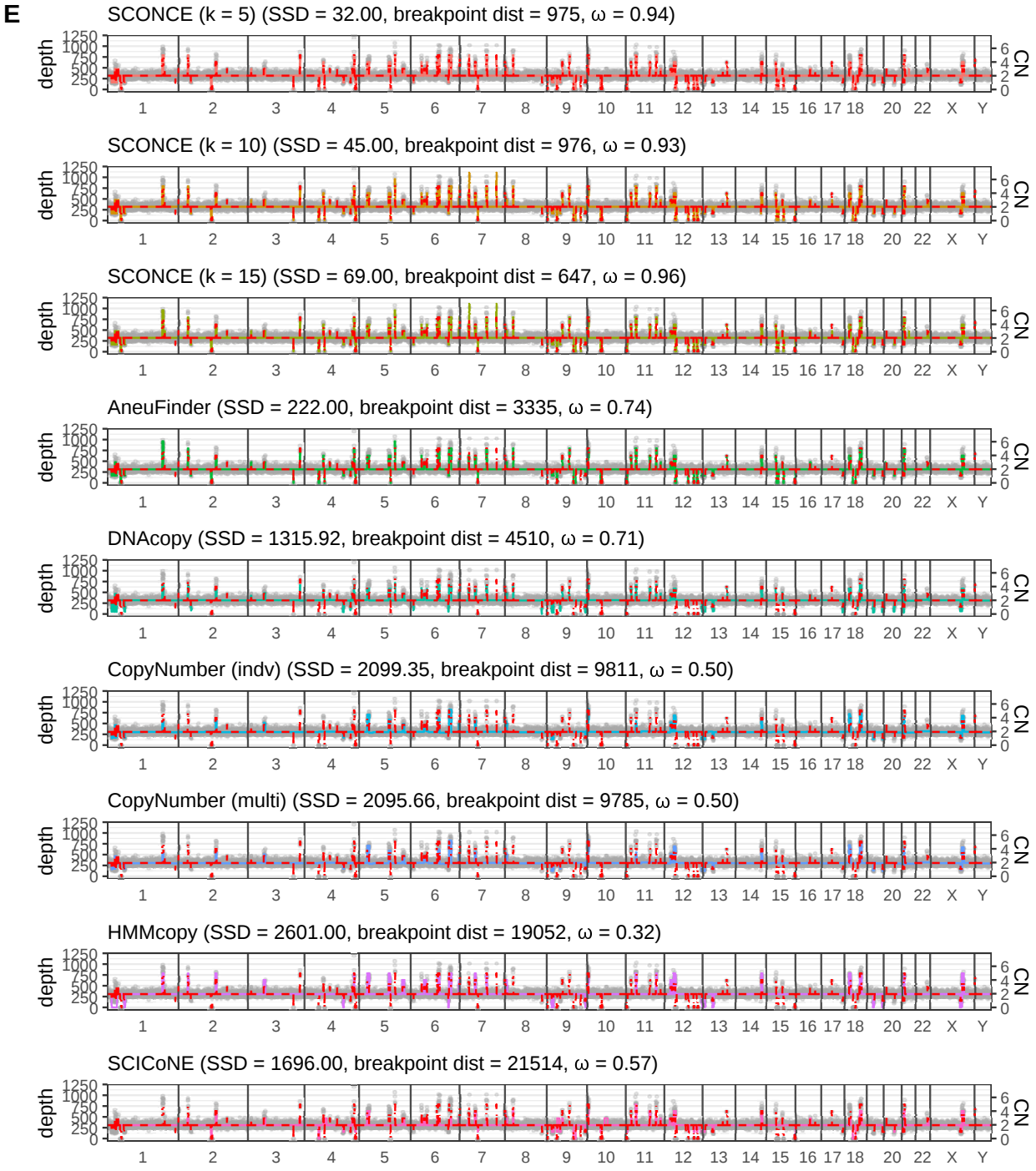
(b) Fig. S8B: Genome wide decoding for cell 48 in Simulation Set B (whole genome duplication before any CNAs, under the line segment model).



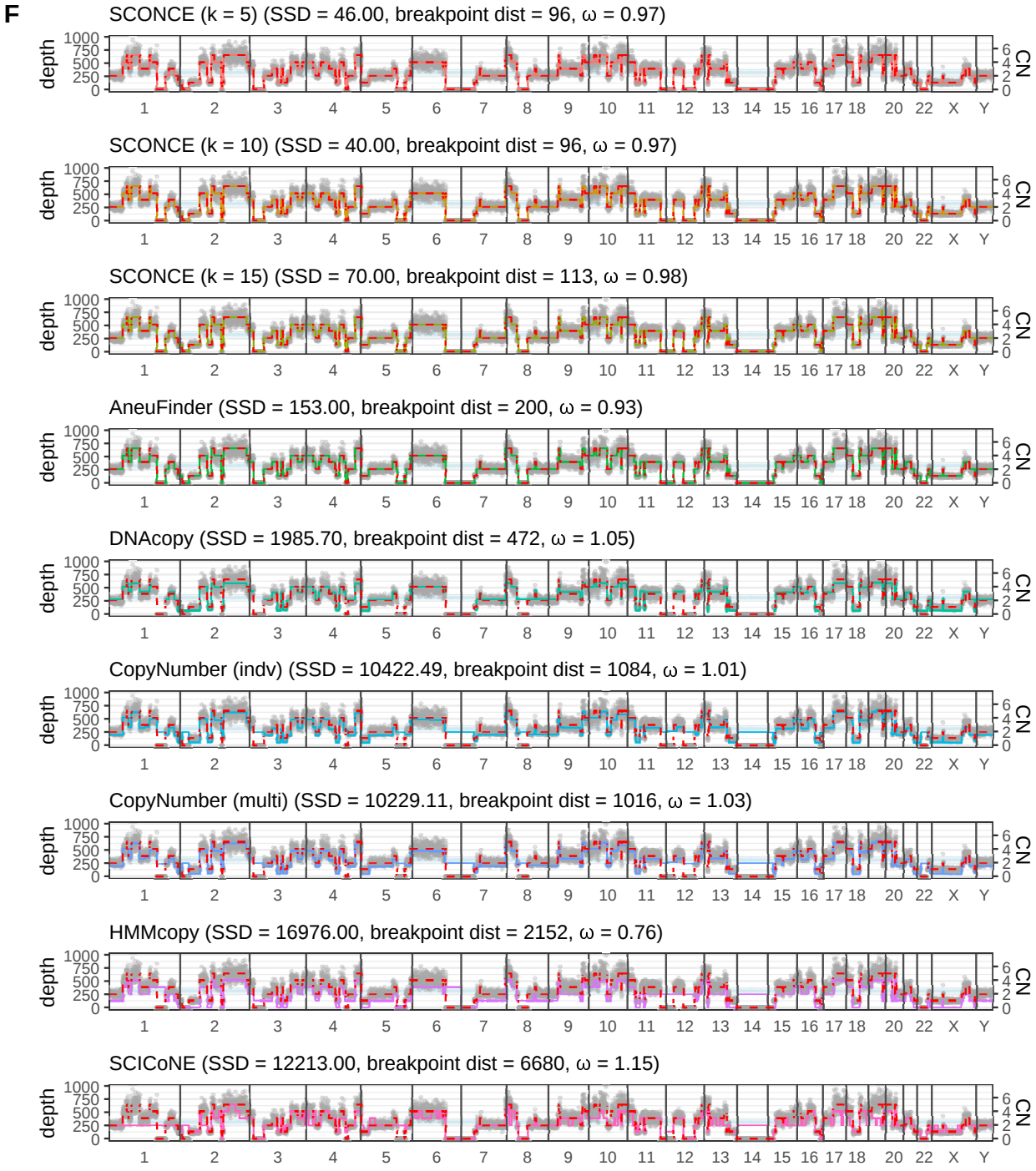
(c) Fig. S8C: Genome wide decoding for cell 5 in Simulation Set C (many large deletions, under the line segment model).



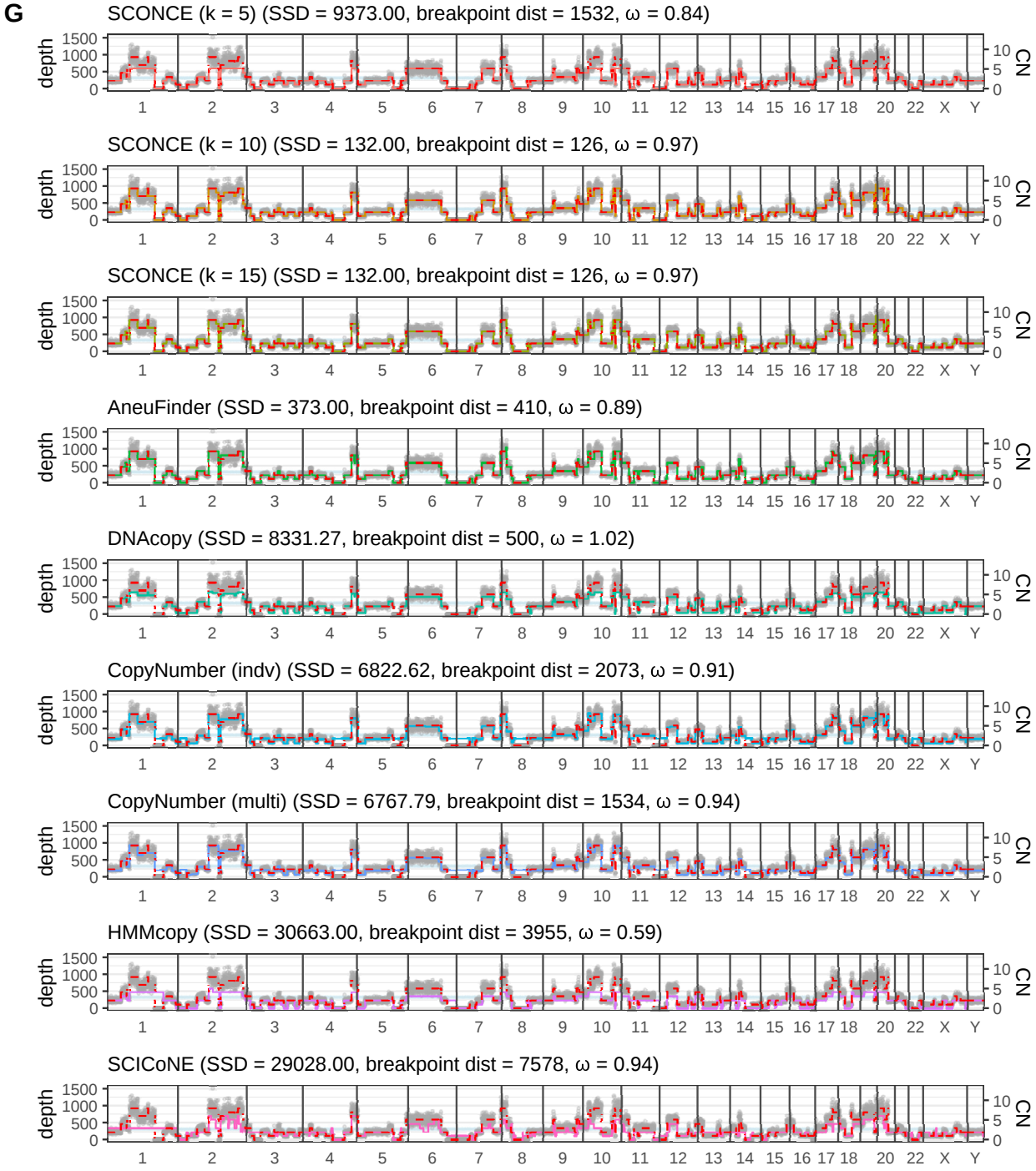
(d) Fig. S8D: Genome wide decoding for cell 38 in Simulation Set D (many large insertions, under the line segment model).



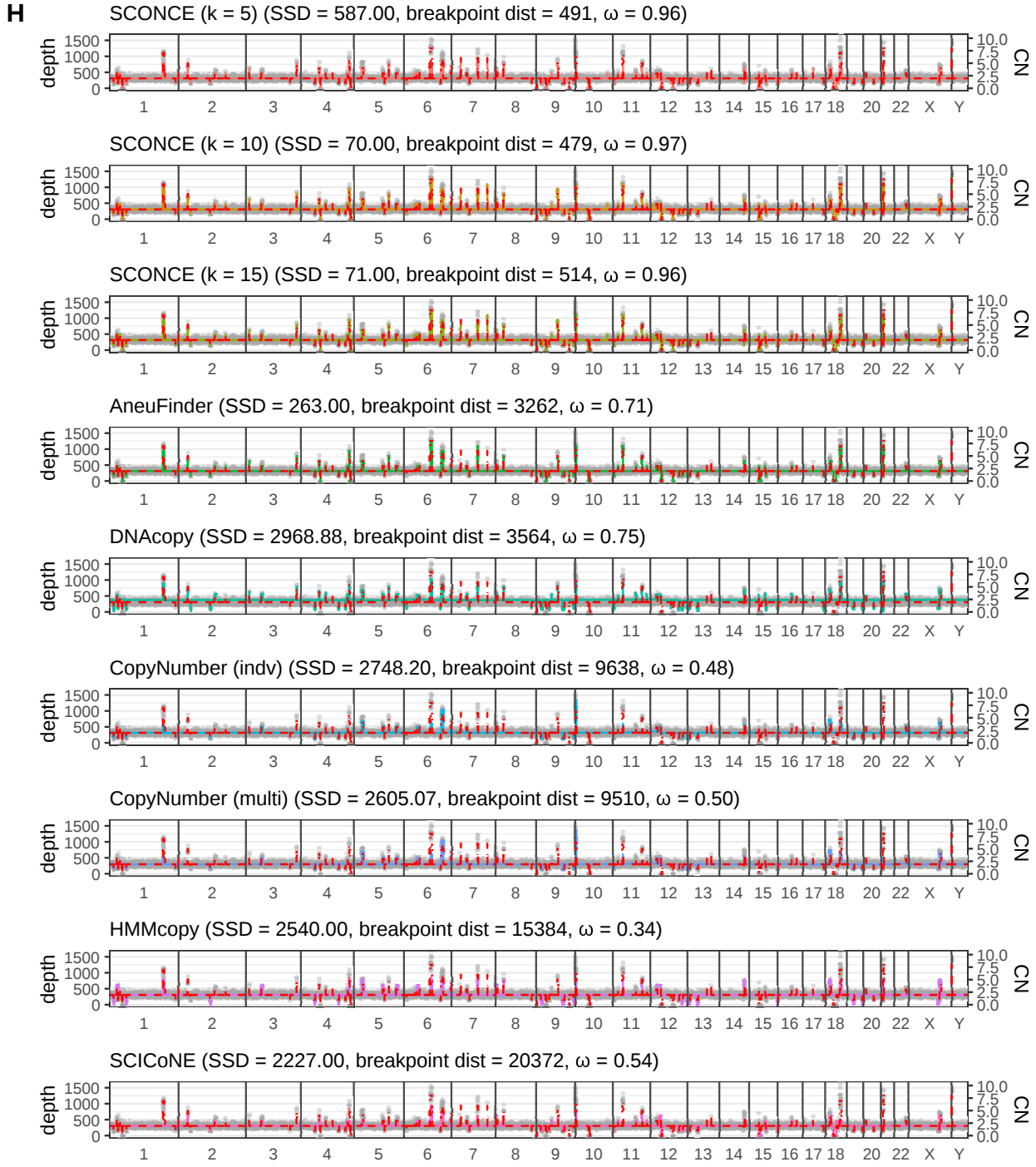
(e) Fig. S8E: Genome wide decoding for cell 51 in Simulation Set E (very short spiky CNAs, with $k = 5$ and uniform initialization matrix, under the binned model).



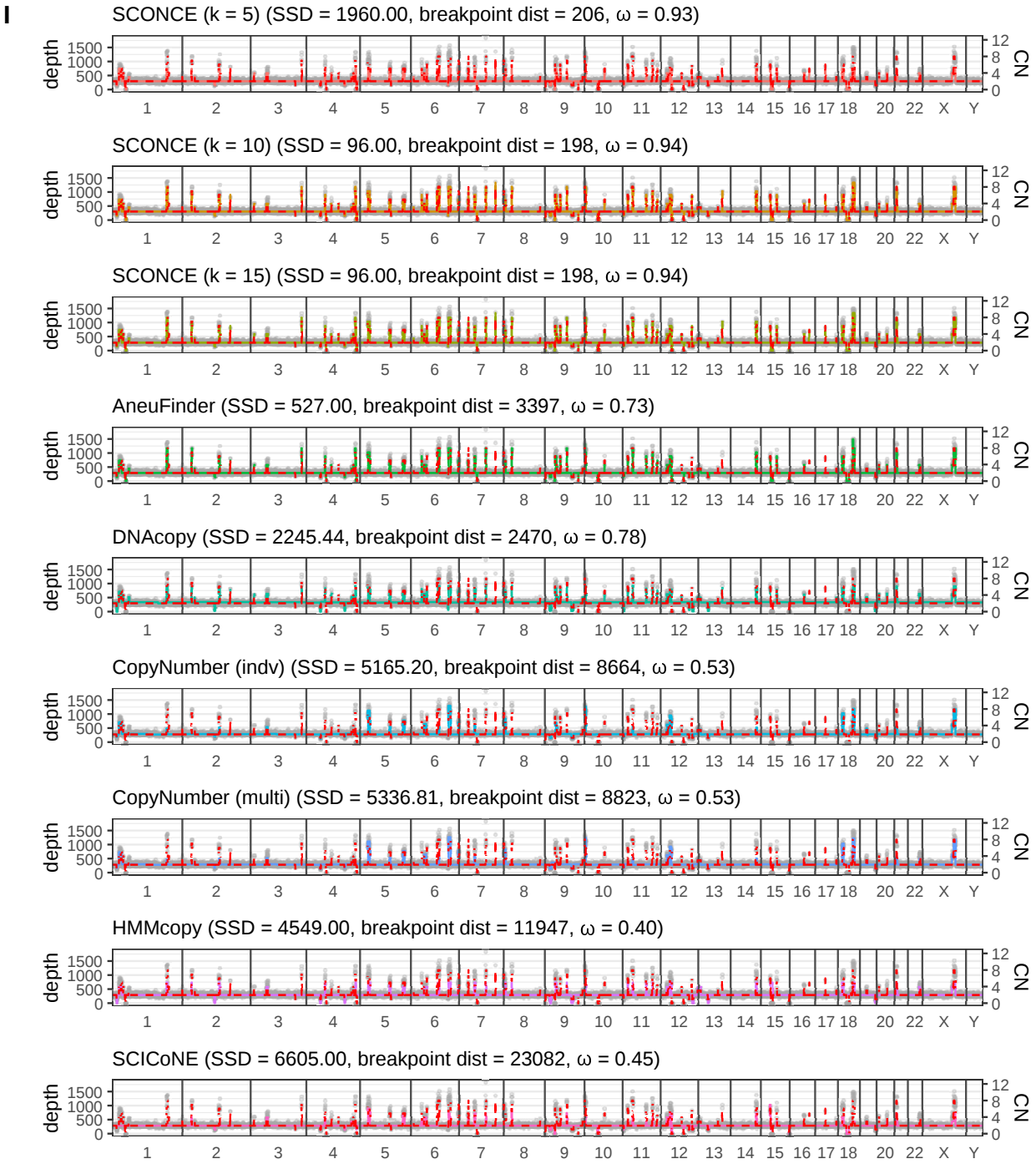
(f) Fig. S8F: Genome wide decoding for cell 3 in Simulation Set F (many overlapping CNAs, with $k = 5$ and nonuniform initialization matrix, under the binned model).



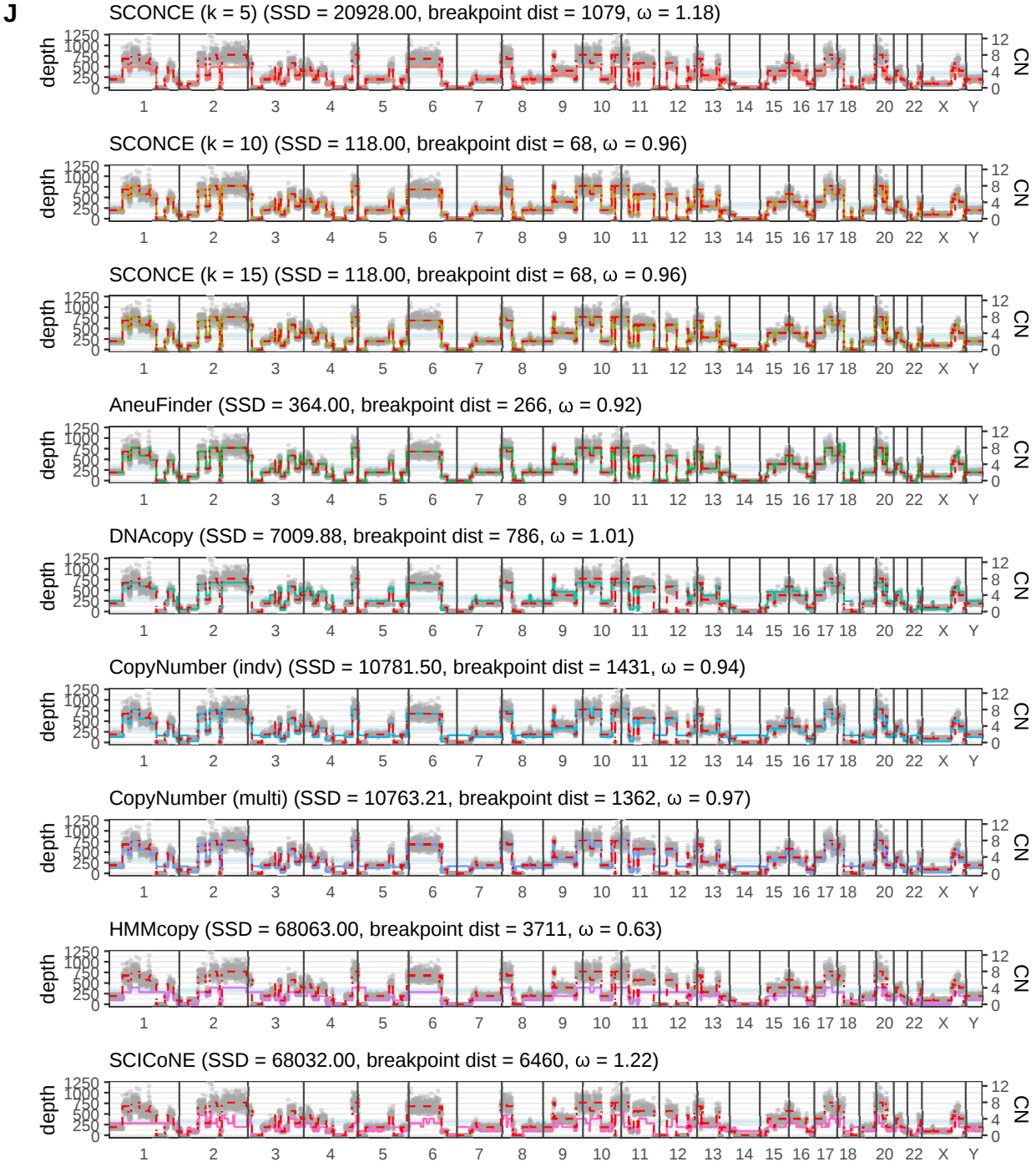
(g) Fig. S8G: Genome wide decoding for cell 1 in Simulation Set G (many overlapping CNAs, with $k = 8$ and nonuniform initialization matrix, under the binned model).



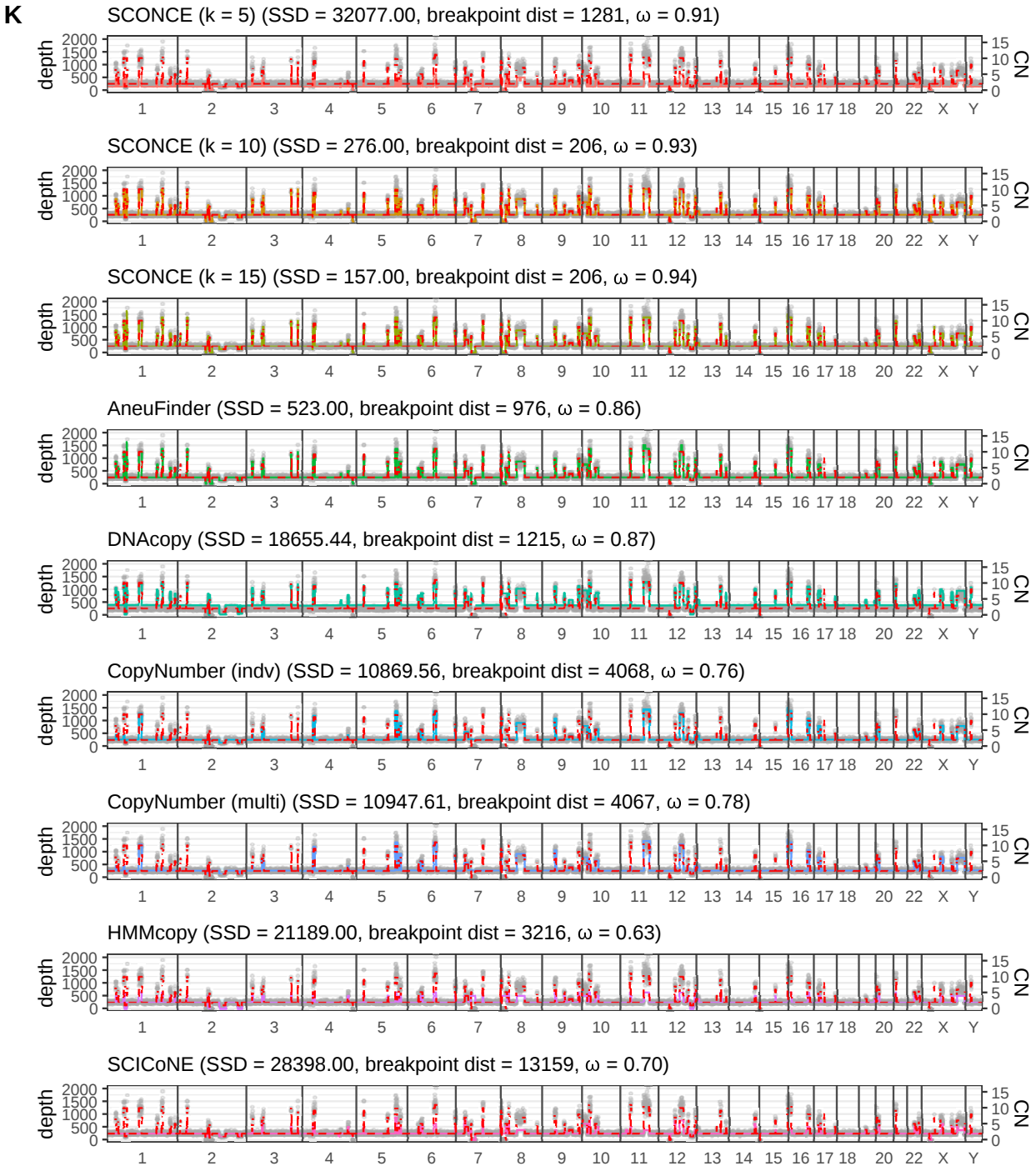
(h) Fig. S8H: Genome wide decoding for cell 29 in Simulation Set H (very short spiky CNAs, with $k = 8$ and nonuniform initialization matrix, under the binned model).



(i) Fig. S8I: Genome wide decoding for cell 20 in Simulation Set I (very short spiky CNAs, with $k = 8$ and uniform initialization matrix, under the binned model).

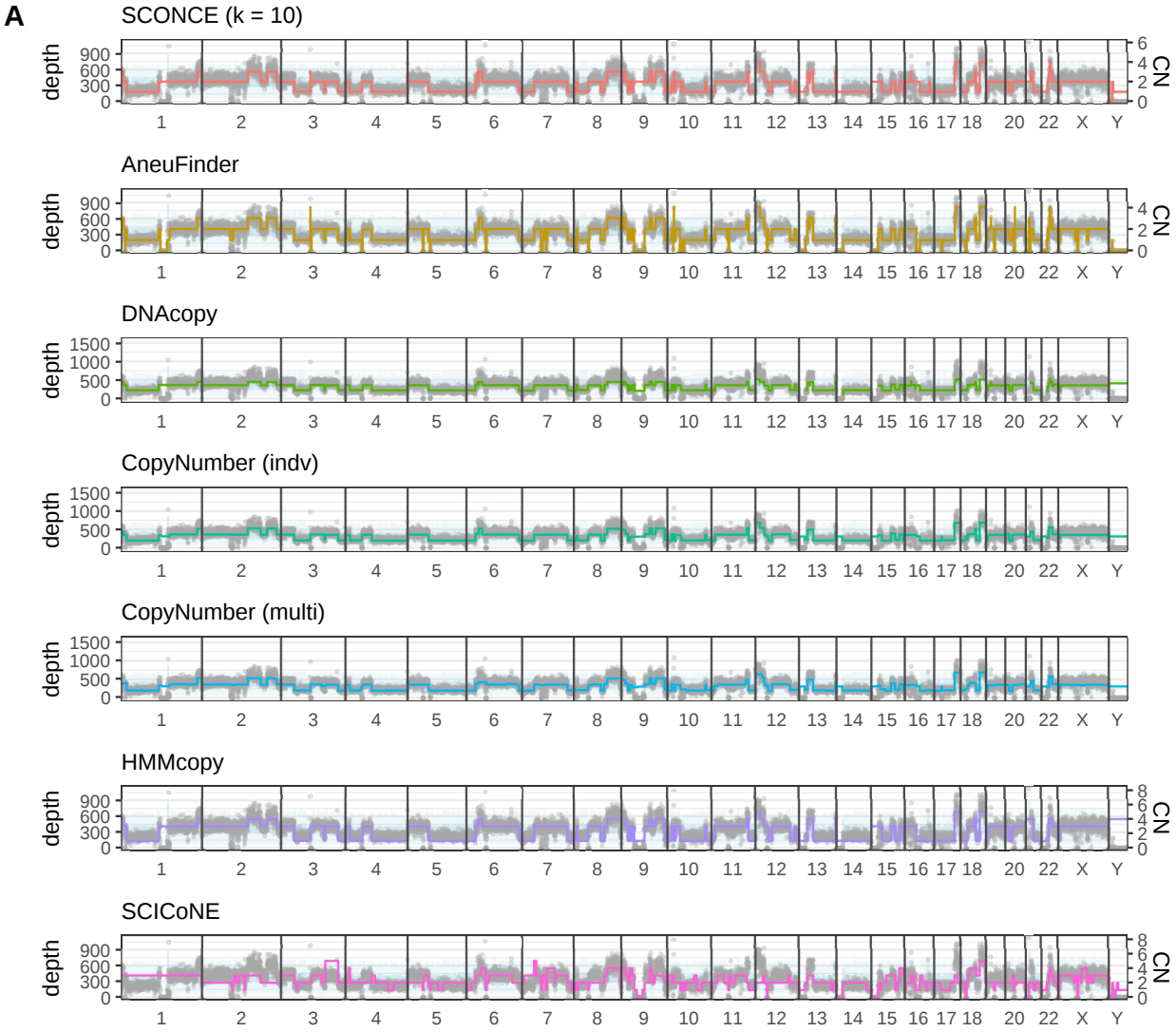


(j) Fig. S8J: Genome wide decoding for cell 95 in Simulation Set J (many overlapping CNAs, with $k = 8$ and uniform initialization matrix, under the binned model).

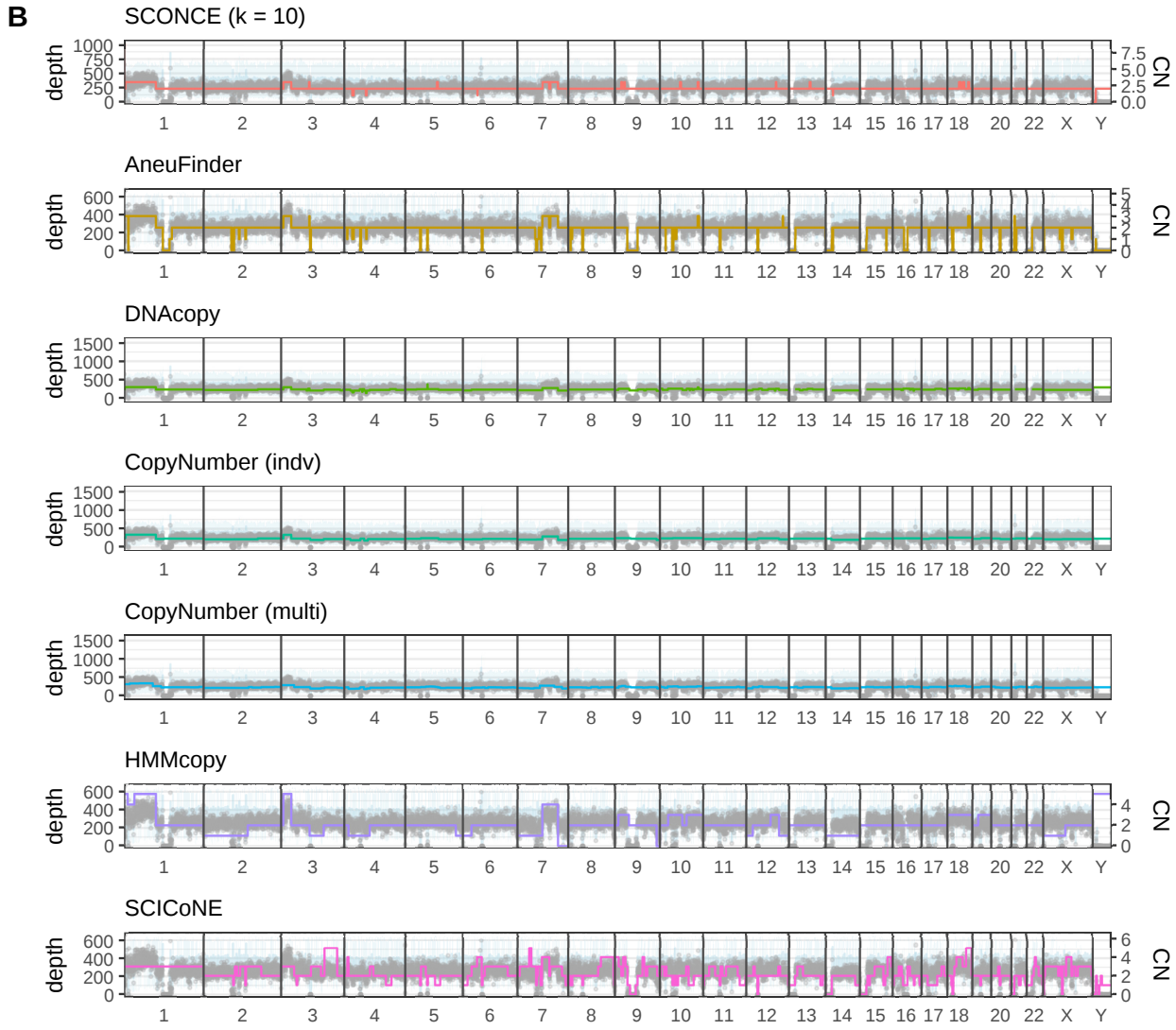


(k) Fig. S8K: Genome wide decoding for cell 29 in Simulation Set K (very short spiky CNAs, with $k = 11$ and uniform initialization matrix, under the binned model).

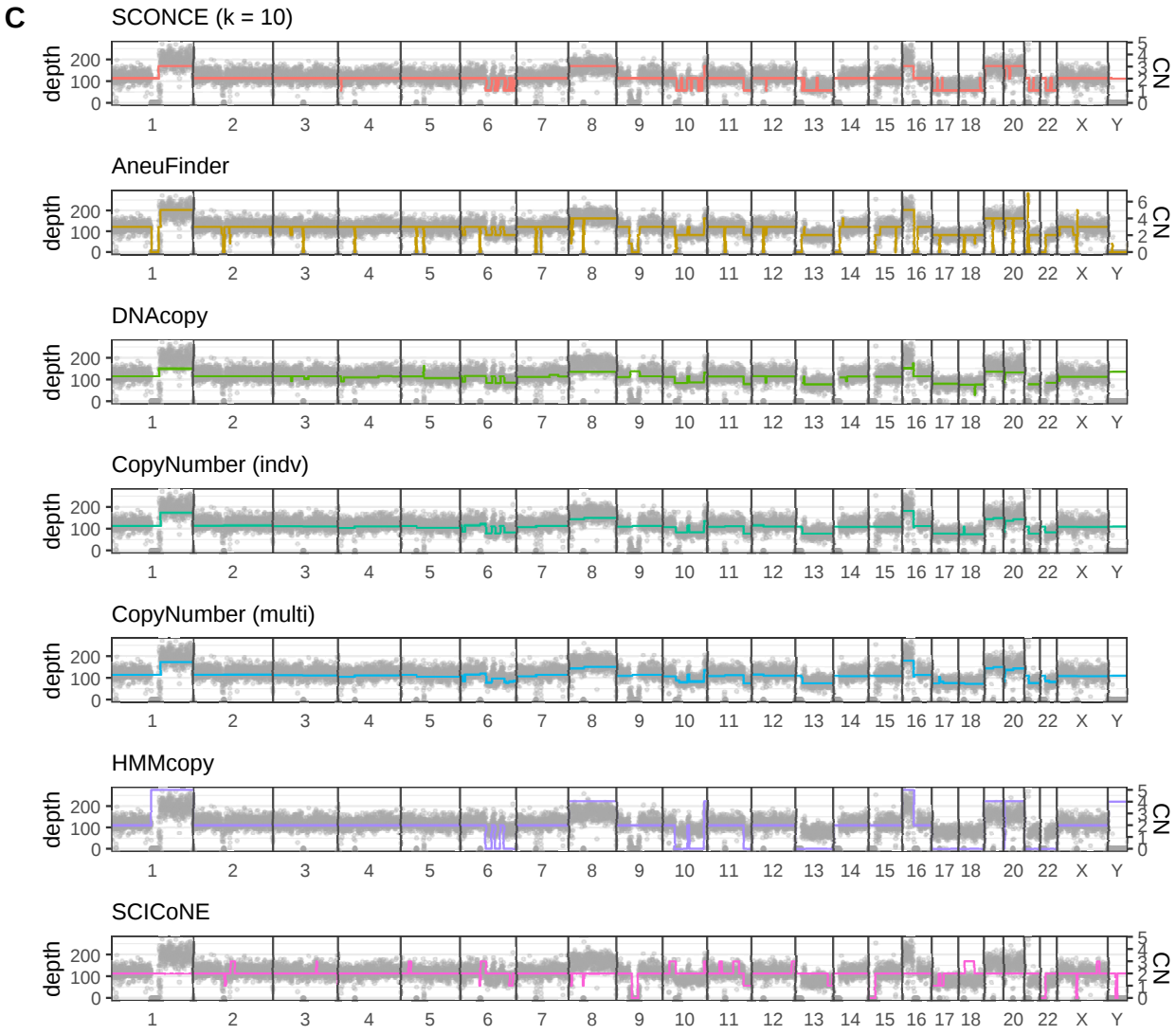
Figure S8: Genome wide decodings are shown for representative cells across simulation sets, where subpanel letters match simulation sets described in Supplement 2.8. As in Figure 4, genomic window is plotted along the x-axis, per window read depth is shown along the left y-axis, and copy number is plotted along the right y-axis. Black vertical lines denote chromosome boundaries, gray dots represent observed tumor read depth in each window, the red dotted line denotes the true copy number from simulation, the light blue line shows the mean diploid read count, the light blue band shows ± 1 standard deviation in the diploid read count, and the colored lines denote the copy number decoding from each method. Across simulation conditions, SCONCE with $k = 10, 15$ consistently decodes the correct copy number state. That is, the value of k must be set high enough to allow the HMM to properly decode the true copy number state. Additionally, SCONCE repeatedly correctly identifies regions with no read coverage, unlike DNAcopy (which does not output anything for regions with no coverage), CopyNumber, and HMMcopy. Furthermore, in panel C, SCICoNE doesn't run andscaling problems are evident for AneuFinder and HMMcopy, while SCONCE correctly calls CNAs.



(a) Fig. S9A: Genome wide decoding for cell SRR054570 from [4].



(b) Fig. S9B: Genome wide decoding for cell SRR053675 from [4].



(c) Fig. S9C: Genome wide decoding is shown for the cell with barcode AAACCTGGTTCTTTGT-1 from section C, from [5].

Figure S9: As in Figure 4 and Supplemental Figure S8, genomic window is plotted along the x-axis, read depth along the left y-axis, and copy number (CN) along the right y-axis, with black vertical lines denoting chromosome boundaries. For each window, gray dots show observed tumor read depth, the light blue line shows the mean diploid read count, and the light blue band shows ± 1 standard deviation in the diploid read count. Colored lines denote the copy number decoding from each method. Note that absolute copy number calls are absent for DNAcopy and CopyNumber, as these methods do not produce absolute copy number calls and could not be optimally scaled as done previously in simulations without ground truth data. SCONCE uses the null diploid model to predict no changes in copy number in hard to sequence and map regions that have no observed diploid or tumor reads (chromosomes 1, 9, 13-16, 21, and 22 in panel A; chromosomes 13, 14, 15, 21, and 22 in panel C), unlike AneuFinder and DNAcopy. Recapitulating the simulation results, SCONCE is more sensitive to small CNAs than all other methods (chromosomes 9, 10, 12, 13, and 18 in panel B; chromosomes 6, 10, 13, 17, 21, and 22 in panel C). Furthermore, in panel C, HMMcopy erroneously predicts copy number 0 for chromosomes 13, 17, 18, 21, and 22, and SCICoNE misses changes in copy number for chromosomes 1, 16, 19, and 20, while SCONCE does not.

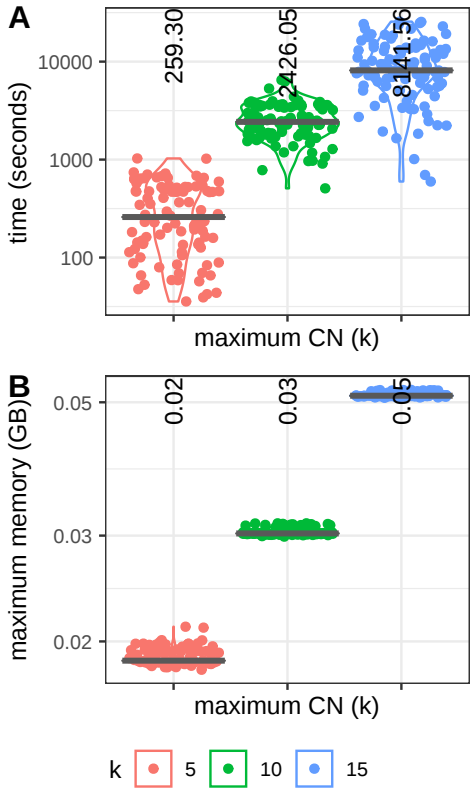


Figure S10: Runtime and memory requirements for SCONCE for Simulation Set A, for different values of k. Each dot represents one cell.

Supplementary Tables

Simulation set	**Short description**	**Max obs copy number**	**Deletion rate, $\frac{\delta}{\tau_d}$**	**Insertion rate, $\frac{\varphi}{\tau_a}$**	**Mean deletion length, τ_d**	**Mean insertion length, τ_a**
A	many overlapping CNAs	8	0.001	0.001	10	10
B	whole genome duplication before any CNAs	13	0.001	0.001	10	10
C	many large deletions	3	0.005	0.0001	5	0.05
D	many large insertions	5	0.0001	0.0005	10	10

Table S1: Description of simulation sets under the line segment model. Mean deletion and insertion lengths are relative to a genome length of 100. All simulations were done under the neutral coalescent, with a tree branch length leading to the root of tree (the ancestral diploid genome) of 1.0. Although the mean CNA length remains constant across simulation sets, the overall size of CNAs is also affected by the deletion and insertion rates, as CNAs tend to overlap. Note, the maximum copy number under the line segment model is not constrained; the maximum observed copy numbers are reported here to demonstrate the range of simulations.

Simulation set	**Short description**	**Max copy number, k**	**Tree branch length, t**	**Geometric p**
E	very short spiky CNAs	5	1.0	0.1
F	many overlapping CNAs	5	100	0.005
G	many overlapping CNAs	8	100	0.005
H	very short spiky CNAs	8	100	0.1
I	very short spiky CNAs	8	100	0.1
J	many overlapping CNAs	8	100	0.005
K	very short spiky CNAs	11	100	0.05

Table S2: Description of simulation sets under the binned model, relative to a genome length of 100. All simulations were done under the neutral coalescent, with a tree branch length leading to the root of tree of 1.0.

Regression Model	**All windows**	**Without outliers**
Eq S1a: $\sigma^2_{iA} = b\lambda_{iA} + c$	0.8331236	0.7568441
Eq S1b: $\sigma^2_{iA} = a\lambda^2_{iA} + b\lambda_{iA} + c$	0.8736199	0.7737766
Eq S1c: $\sigma^2_{iA} = d\lambda^3_{iA} + a\lambda^2_{iA} + b\lambda_{iA} + c$	0.8778322	0.7739496

Table S3: Adjusted R^2 values are shown for each regression model across all windows and across the windows excluding outliers. The second degree polynomial, S1b, $\sigma^2_{iA} = a\lambda^2_{iA} + b\lambda_{iA} + c$ has the lowest adjusted R^2 value without overspecifying the regressions in both data scenarios.

	SCONCE ($k = 5$)	**SCONCE** ($k = 10$)	**SCONCE** ($k = 15$)	**Aneu-Finder**	**DNA-copy**
A	698	210	124	300.5	402
B	5222.5	1558.5	424	584	1437.5
C	253	253	251.5	301	3299
D	2270.5	2270	2270	2278	538
E	799	817.5	809.5	2712.5	4101
F	105	106	103.5	161	555
G	1468	185	209.5	287	568
H	1028	1013.5	1020	3119.5	3253.5
I	353	344	348	2276	2744.5
J	931	72.5	74	298	800
K	1184.5	109	117	569	1155

	CopyNumber (indv)	**CopyNumber (multi)**	**HMMcopy**	**SCICoNE**
A	1026.5	676	6138.5	283578
B	2796.5	2825	12228.5	13786
C	2629	2533	23925	NA
D	1126	1120	3786	4080
E	9910.5	9776	18423.5	21131
F	1199.5	1028	1972.5	6638
G	1904.5	1608	3834	7439
H	9255.5	9501	13941	20087
I	8902	8948	12142	24117
J	1477.5	1304	3616.5	6428
K	4009.5	4067	3235.5	13162

Table S4: Median total breakpoint distance values for each program across simulation datasets.

	SCONCE ($k = 5$)	**SCONCE** ($k = 10$)	**SCONCE** ($k = 15$)	**Aneu-Finder**	**DNA-copy**
A	0.5	0.5309	0.5408	0.5	0.602
B	0.537	0.4731	0.506	0.4549	0.5647
C	0.49	0.49	0.4911	0.4828	0.431
D	0.4677	0.4677	0.4677	0.4531	0.8387
E	0.9446	0.941	0.9453	0.7828	0.7215
F	0.967	0.9671	0.9671	0.9408	1.0259
G	0.8908	0.9499	0.9441	0.9	0.9889
H	0.9302	0.9321	0.9321	0.733	0.7828
I	0.9455	0.95	0.95	0.7928	0.8069
J	1.4412	0.9739	0.9739	0.9221	0.9673
K	0.9105	0.9368	0.9368	0.8632	0.9096

	CopyNumber (indv)	**CopyNumber (multi)**	**HMMcopy**	**SCICoNE**
A	0.6429	0.6735	0.3061	0.0306
B	0.5706	0.5588	0.2778	0.8
C	0.6429	0.6638	0.1798	NA
D	1.1452	1.1452	0.4688	1.7903
E	0.4897	0.5057	0.3333	0.5805
F	0.987	1.0263	0.7745	1.1316
G	0.8983	0.9278	0.6034	0.9444
H	0.5067	0.5045	0.3408	0.5451
I	0.5214	0.5397	0.3973	0.4591
J	0.9477	0.9805	0.6396	1.2288
K	0.7688	0.7842	0.6368	0.7

Table S5: Median ω = #inferred breakpoints/#true breakpoints values for each program across simulation datasets.

Chapter 3

SCONCE2: jointly inferring single cell copy number profiles and tumor evolutionary distances

This chapter is currently under review at *BMC Bioinformatics*, and has been publicly posted to *bioRxiv* (2022). The most recent revision is included here. The authors on this manuscript are:

Sandra Hui[1], Rasmus Nielsen[1,2,3]

1. Center for Computational Biology, University of California, Berkeley, Berkeley, CA, USA
2. Department of Integrative Biology, University of California, Berkeley, Berkeley, CA, USA
3. Department of Statistics, University of California, Berkeley, Berkeley, CA, USA

3.1 Abstract

Background: Single cell whole genome tumor sequencing can yield novel insights into the evolutionary history of somatic copy number alterations. Existing single cell copy number calling methods do not explicitly model the shared evolutionary process of multiple cells, and generally analyze cells independently. Additionally, existing methods for estimating tumor cell phylogenies using copy number profiles are sensitive to profile estimation errors.
Results: We present SCONCE2, a method for jointly calling copy number alterations and estimating pairwise distances for single cell sequencing data. Using simulations, we show that SCONCE2 has higher accuracy in copy number calling and phylogeny estimation than competing methods. We apply SCONCE2 to previously published single cell sequencing data to illustrate the utility of the method.
Conclusions: SCONCE2 jointly estimates copy number profiles and a distance metric for inferring tumor phylogenies in single cell whole genome tumor sequencing across multiple cells, enabling deeper understandings of tumor evolution.

3.2 Background

Cancer evolution is driven by an accumulation of somatic point mutations and large copy number alterations (CNAs) [21, 33]. For example, CNAs can affect the transcriptional landscape via dosage effects [11], and identifying intra tumor heterogeneity and cell specific changes in gene expression is clinically relevant. In particular, recent studies have shown quantifying these transcriptional changes, by measuring tumor specific total mRNA expression, is predictive of disease prognosis and progression across multiple cancer types [71]. In this manuscript, we focus on estimating the underlying copy number alterations using whole genome sequencing, in order to directly study the evolutionary process.

Single cell sequencing can offer a detailed picture of the process of CNA and mutation accumulation that is lost in bulk sequencing, in particular by estimating the phylogenetic relationship among different cell types. A challenge in such efforts is that single cell sequencing data is typically very noisy due to variable and low sequencing depth [38], making accurate genotyping, copy number (CN) calling, and phylogeny estimation difficult. However, as we will show here, by leveraging the shared evolutionary history among cells, jointly calling CNAs across cells can lead to increased accuracy and give information about the evolutionary relationship between cells, thereby leading to improved estimates of tumor phylogenies. Different cells from the same tumor share some of their somatic evolutionary history, and information regarding CNAs, and CNA breakpoints, from one cell can, therefore, inform CNA calling in other cells.

Unfortunately, the commonly used methods for estimating single cell copy number profiles (CNPs), the collection of copy number states across the genome, do not rigorously use this shared information. Instead, most methods, including SCONCE [3] and the commonly-used AneuFinder [49, 50], independently call CNPs. Although SCONCE [3], a copy number

calling method for single cell tumor data, was previously shown to outperform competing methods in absolute copy number and breakpoint detection accuracy [3], it does not utilize any information from shared evolutionary histories between cells. Other methods, such as CopyNumber [45] and SCIcONE [52], jointly call CNPs by forcing breakpoints to be shared across all cells. However, we showed in previous work that SCONCE [3] outperforms these methods as well, despite not analyzing cells jointly. In contrast, WaveDec [72], a method designed to detect shared and cell specific copy number events in copy number arrays and applied to a subset of sequencing data from [4], takes an orthogonal approach by transforming $\log_2$-ratio copy number data into the wavelet space. This transformation allows separation of common/shared and individual CNAs, as shared events are captured by the approximation coefficients and individual events are described by the detail coefficients.

Despite these limitations in copy number calling, several distance metrics for copy number profiles have been developed for estimating tumor phylogenies using algorithms such as neighbor-joining [31, 32]. Commonly used pairwise distance metrics include the Euclidean distance [4, 73], the MEDICC distance described by [74], and the cnp2cnp distance presented by [75]. Although the Euclidean distance is easy to calculate, large and/or overlapping CNAs can artificially inflate this measure, leading to overestimation of dissimilarity. The latter two methods measure distance between two CNPs by attempting to find the minimum number of deletion and amplification events needed to transform one CNP into the other, without allowing regions that are lost to be regained. The MEDICC model is limited to maximum copy number 4 and events that increase or decrease copy number by one, while the cnp2cnp metric relaxes both of these constraints. Cordonnier *et al.* [75] re-implemented the MEDICC algorithm to allow copy numbers greater than 4, and showed that while both the cnp2cnp and MEDICC distances outperform the Euclidean distance for the purpose of phylogeny estimation, cnp2cnp is more accurate on error free data and MEDICC is more accurate on data with errors.

However, none of these methods use explicit evolutionary models of CNAs to provide joint estimates of CNPs and evolutionary distance. Here, we present SCONCE2, an expansion on SCONCE, that further develops SCONCE's underlying tumor evolutionary model to jointly model the CNA process in two cells. SCONCE2 takes advantage of the shared evolutionary history between cells, and produces more accurate single cell CNP estimates and pairwise estimates of the evolutionary distances between cells, by combining information across multiple cells. We show that SCONCE2 estimates more accurate CNPs and tumor phylogenies than competing methods using extensive simulations, and apply it to previously published data from [4, 5] to illustrate its utility.

3.3 Results

To infer the evolutionary history of tumor cells, SCONCE2 models the evolution of pairs of cells. We assume a pair of cells, (A, B), have a partially shared evolutionary history originating from a healthy ancestral diploid cell, D. The shared part of their evolutionary

history is represented in a tree, $\mathcal{T} = [t_1, t_2, t_3]$, by a branch of length t_1, running from an non-tumor diploid cell (D) to an unobserved divergence point, Z. From Z, cells A and B evolve independently, with branch lengths t_2 and t_3, respectively (see Figure 1). A core goal is to estimate this tree and to distinguish between shared evolutionary events and independent cell specific events.

Because the number of pairs of cells grows quadratically with the number of cells, n, full joint maximum likelihood estimation of all parameters can become computationally challenging. We, therefore, first run SCONCE on all cells independently to obtain cell specific estimates of model parameters. We then take the median of the estimates of evolutionary parameters $\{\alpha, \beta, \gamma\}$, corresponding to the rates of different types of copy number events (see One cell continuous time Markov process), to combine the disjoint SCONCE estimates into summary estimates across all cells. Then, for each pair of cells, we estimate branch lengths of tree $\mathcal{T} = [t_1, t_2, t_3]$ using maximum likelihood, and use the Viterbi algorithm to calculate paired decoded copy number profiles. Because each cell appears in $n-1$ pairs, this produces $n-1$ paired CNP estimates per cell. Finally, for each cell, we take the per window mean across each cell's $n-1$ paired CNP estimates to calculate consensus CNPs. This pipeline is described in Detailed SCONCE2 pipeline and illustrated in Figure 2.

By analyzing each cell in the context of multiple pairs, we obtain increased accuracy in copy number calls and breakpoint detection, as well as usable tree branch length estimates. We examine the properties of these estimates on both simulated and real data.

Simulations

In order to rigorously test SCONCE2, we applied it to four simulated datasets and two real datasets, from [4, 5]. We simulated 128 cells on four different tree structures: tree A) is maximally imbalanced and ultrametric, tree B) is perfectly balanced and ultrametric, tree C) is maximally imbalanced and not ultrametric, where internal and terminal branches have uniform length, and tree D) is maximally imbalanced and not ultrametric, where internal branches have equal length and terminal branch lengths decay logarithmically (tree structures shown in Supplementary Figure S1, Additional File 1). Simulated cells from each tree structure were divided into five discrete test subsets of 20 cells each.

Briefly, the simulated genome is modeled as a collection of line segments, where amplifications and deletions occur according to a Markov process and have lengths sampled from a truncated exponential distribution. Copy number events occur within the tree structure, such that ancestral CNAs are propagated to descendent cells. Note, the simulation model is more biologically realistic and intentionally structured to be substantially different from the SCONCE2 inference model, in order to avoid biasing accuracy results to favor our method. We previously described this simulation model in [3], and full simulation details are given in Simulations.

Copy Number and Breakpoint Detection Accuracy

Sum of Squared Error on CNPs

To measure copy number accuracy, we calculated the sum of squared errors (SSE) between the inferred copy number and the true simulated copy number across genomic windows for each cell. To evaluate each step in the SCONCE2 pipeline, we calculated the SSE on copy number profiles generated from individual cell estimation (SCONCE), on profiles from each pair of cells (one pair), and on consensus profiles estimated using three different summary statistics (mean, median, mode) across multiple pairs of cells. We also compared to AneuFinder [49, 50], a commonly used method for single cell copy number calling, and the second-most accurate one, after SCONCE, among methods evaluated in previous work [3]. In all subsequent results, we report summary statistics across all subsets for each tree/simulation set. Recall full simulation descriptions are given in Simulations.

In tree A (maximally imbalanced ultrametric tree; Figure 3A), using pairs of cells had lower SSE than individual cells (SCONCE) alone, with respective median SSE values of 26.01 and 37.31. Furthermore, using the mean had the lowest median SSE of 17.83, with median and mode at 23.68 and 24.04. These SSE values were lower than AneuFinder, which had a median SSE value of 51.78. Similar results for tree B (perfectly balanced ultrametric tree) are shown in Figure 3B, with median SSE values of 28.37, 21.25, 14.74, 17.90, 18.09, and 41.80 for individual cells, single pairs, mean, median, mode, and AneuFinder, respectively. In the same order, the median SSE values for tree C (maximally imbalanced with uniform internal branch lengths) were 73.19, 46.50, 22.20, 32.69, 33.22, and 109.31, and the median SSE values for tree D (maximally imbalanced with logarithmically decaying branch lengths) were 60.13, 40.36, 23.05, 32.00, 33.02, and 76.18 (see Figure 3C, D). Clearly, there is a substantial improvement in accuracy by using pairs of cells instead of individual cells, and this improvement in accuracy is larger if multiple pairs are used.

We note that, as an artifact of the genome binning procedure, true fractional copy numbers may occur from small CNAs completely contained within window boundaries, or from CNAs crossing window boundaries (for example, observing windows with true copy numbers $1 \to 1.25 \to 2$). As such, the mean and median have the lowest SSE values because they allow fractional copy numbers. However, many downstream tools expect integer copy number profiles for single cells, so users may wish to round to the nearest integer or use the mode option.

Breakpoint distance and detection

In order to measure breakpoint detection accuracy, we calculated the genome wide distance between inferred and true breakpoints, penalized by the number of total inferred breakpoints. Specifically, for each simulated breakpoint, we calculated the distance to the nearest inferred breakpoint. Because erroneously inferring breakpoints at every position in the genome would artificially lower this genome wide distance, we also calculated $\omega = \frac{\text{\# inferred breakpoints}}{\text{\# true breakpoints}}$, such that lowest breakpoint distances with ω values closest to 1 indicate greatest accuracy.

In all simulation sets, using the mean consistently had ω values closest to 1, again due to fractional copy number states, as well as lower total breakpoint distance than other methods. Across trees, results from AneuFinder, followed by SCONCE, had the highest breakpoint distances and ω values further from 1. For tree A (ultrametric maximally imbalanced tree; Figure 4A), SCONCE, single pairs, mean, median, mode, and Aneufinder had median distance values of 1167, 1006, 394, 1018.5, 1019.5, and 1172, and median ω values of 0.466, 0.490, 0.921, 0.486, 0.486, and 0.462, respectively. Similarly, for tree D (maximally imbalanced with logarithmically decaying branch lengths, Figure 4D), median distance values were 153.5, 85, 33, 77, 77.5, and 168.5, and median ω values were 0.504, 0.535, 1.007, 0.535, 0.534, and 0.489, in the same order as above. Full median distance and ω values are given in Supplementary Tables S2 and S3, Additional File 2. Similarly to the observations for the CNP estimates, breakpoint detection also improves when using pairs of cells, and improves when estimates from multiple pairs are combined, particularly if combining using the mean. As previously noted in Sum of Squared Error on CNPs, binning the genome can result in fractional copy numbers for some bins. Compared to the median and the mode, the mean is better able to capture these fractional copy number states, resulting in lower breakpoint distances and ω values closer to 1.

Optimal number of pairs to use

Because there are $\binom{n}{2}$ pairs for n cells, averaging over more pairs of cells comes at a computational cost. Furthermore, as we will show, adding too many divergent cells can reduce the accuracy, as including highly divergent cells in the average may increase the noise.

To determine the optimal number of cells to summarize across, we estimated summarized (mean) copy number profiles with increasing numbers of cells. As each cell was added, we calculated the difference in SSE relative to SCONCE (individual cells). Cells were added in three different orderings: most to least similar (i.e., nearest first, as defined by the Euclidean distance between the cells' SCONCE profiles), least to most similar (furthest first), and randomized order. In tree A, the median pairwise Euclidean distance between SCONCE profiles was 88.0625 for the nearest/most similar cells, 138.9585 for the tenth most similar cells, and 205.853 for the least similar cells. Median pairwise Euclidean distances for all datasets are given in Supplemental Table S4, Additional File 2.

In Figure 5, we summarize the change in SSE across κ cells for each tree. Across all trees, SSE improves fastest when adding nearest cells first, and slowest for adding furthest cells first, with the random ordering in between. When adding nearest cells first, the SSE initially sharply decreases, levels off and reaches the largest decrease after approximately 10 cells, and then increases. Specifically, the mean change in SSE when adding nearest cells first reached the greatest decrease in SSE from SCONCE of -20.710, -17.491, -56.185, and -38.570 when $\kappa = 12, 10, 9, 15$ cells for trees A, B, C, and D, respectively. In contrast, when $\kappa = 20$ cells, the change in SSE from SCONCE was -19.721, -16.757, -53.461, and -37.920 for trees A, B, C, and D, consistent with the results shown in Figure 3.

When summarizing over $\kappa < n - 1$ cells, for a given cell, some of the other cells will not be used in that cell's consensus profiles. As a time saving measure, these excluded cell combinations are not analyzed. Therefore, we recommend users summarize over $\kappa = 10$ cells, added in order of most to least similar.

For completeness, SSE and breakpoint detection across parameter sets when summarized over only the nearest 10 cells is shown in Supplementary Figures S2 and S3, Additional File 1, and Supplementary Tables S5 and S6, Additional File 2.

Using multiple cells results in better CNA detection

Plotting true simulated copy number profiles against inferred copy number profiles shows why performance improves when using multiple cells. For example, in Figure 6, SCONCE erroneously combined two breakpoints for cell A, while predicting cell B's breakpoint too far to the left (column labelled SCONCE). However, when analyzed as a pair, a shared breakpoint was inferred (left arrow), and the second breakpoint (right arrow) in cell A was correctly inferred (one pair column). While the shared breakpoint was closer to the true breakpoint, it was not until CNPs are summarized across multiple cells that the breakpoint was called in the correct position. Using the mean results in slightly fuzzier boundaries due to non-integer copy number calls (middle column), which better reflected the true underlying data, while the median and mode (right two columns) result in integer jumps at bin boundaries.

Similar results are observed for real data. For example, in Supplementary Figure S4, Additional File 1, SCONCE missed the left most CNA in cell B (arrow in SCONCE column). When analyzed as a pair, this CNA was detected, and was shared with cell A (left arrows). Additionally, a short deletion was called in both cells (right arrows). However, this is a rare event, as it was averaged out in the mean, median, and mode analyses (arrows in mean, median, and mode columns). Furthermore, in Supplementary Figure S5, Additional File 1, SCONCE did not call a CNA in cell A, but did call a -3 deletion in cell B (arrows in SCONCE column). However, when these two cells were jointly analyzed as a pair, there was enough evidence to call a -1 deletion in both. When summarizing across multiple cells, this deletion continued to be supported (right arrow). Additionally, there was some evidence from joint analyses with other cells that an additional small deletion existed in cell B, but not in cell A (left arrow). However, this small deletion was lost when using the median and mode, although the deletion first identified in the joint analysis of cells A and B remained.

Model Parameter Estimates

For each pair of cells, SCONCE2 estimates the branch lengths for tree $\mathcal{T} = [t_1, t_2, t_3]$ (see Figure 1). From the simulated trees, the corresponding tree branch lengths and node distances can be extracted for each cell pair. Recall the simulation and inference models are intentionally formulated differently to evaluate SCONCE2 in more realistic settings (see Simulations). Because the scaling of $\mathcal{T}$ is different between the simulation and inference models,

we show the R^2 values between true (simulated) and inferred values of $\mathcal{T} = [t_1, t_2, t_3]$, as well as the summed distance $t_2 + t_3$ as a distance metric between two cells.

For tree A (ultrametric, maximally imbalanced), SCONCE2 recovered $\{t_1, t_2, t_3, t_2 + t_3\}$ values with R^2 values of 0.798, 0.35, 0.286, and 0.551 (Figure 7A), respectively. Additionally, for tree D (maximally imbalanced with uniform branch lengths), SCONCE2 had R^2 values of 0.661, 0.564, 0.59, and 0.686, for t_1, t_2, t_3, and $t_2 + t_3$. (Figure 7D).

We note that the sum $t_2 + t3$ has higher R^2 values than those of t_2 or t_3 individually, demonstrating some uncertainty in assigning events to particular branches. Furthermore, because the simulation model generates the number of CNAs from a distribution relating to a Poisson (however, the distribution is not truly Poisson as the size of the genome changes through the simulations), the mean and variance of the number of events increases with branch length in expectation. This increased variance is reflected by the larger range of branch length estimates as branch lengths increase. Nonetheless, as we will show in the next section, SCONCE2 recovers the magnitude of cell relationships sufficiently accurately to allow improved phylogeny estimation.

Phylogeny Estimation

Estimating phylogenies on copy number profiles using neighbor-joining [31, 32] requires a distance metric between cells. Existing metrics include the Euclidean distance [4], and two estimates of the minimum number of CNAs needed to transform one CNP into another: the cnp2cnp metric [75] and the MEDICC distance [74] (here, we use the implementation in the cnp2cnp program [75]). These methods require prior estimation of the CNP. See Running other methods for details on running these programs.

Under the SCONCE2 model, by construction, $t_2 + t_3$ measures the pairwise distance between two cells. To compare these different distance metrics, we first calculated distance matrices using pairwise Euclidean, cnp2cnp, and MEDICC distances on CNPs called by SCONCE (previously showed to be more accurate than other single cell copy number callers [3]), as well pairwise $t_2 + t_3$ estimates. Next, we applied neighbor-joining to estimate phylogenies and computed the Robinson-Foulds (RF) distance [76] between the true trees and the inferred trees. As shown in Figure 8, across parameter sets, the trees inferred from estimates of $t_2 + t_3$ had lower Robinson-Foulds distances than trees inferred from other distance metrics. For example, for tree A (ultrametric, maximally imbalanced), the median RF distances were 27, 27, 29, and 20 for the Euclidean distance, cnp2cnp distance, MEDICC distance, and $t_2 + t_3$ (Figure 8), respectively (see Supplementary Table S7, Additional File 2, for all median Robinson-Foulds distances).

Furthermore, we calculated RF distances from phylogenies based on the Euclidean, cnp2cnp, and MEDICC distances on consensus CNPs and true simulated CNPs (Supplementary Figure S6, Additional File 1). When summarizing over all pairs of cells, using $t_2 + t_3$ consistently had lower median RF distances than other methods on consensus CNPs. For example, in tree A (ultrametric, maximally imbalanced), the median RF distances for phylogenies estimated from mean consensus profiles were 27, 26, and 28 for the Euclidean,

cnp2cnp, and MEDICC distances (Supplementary Figure S6A, Additional File 1). On distances calculated from the true CNPs, $t_2 + t_3$ performed as well or better than the other metrics, with the exception of the cnp2cnp distance in tree A, where the Euclidean, cnp2cnp, MEDICC distances and $t_2 + t_3$ had respective median RF distances of 27, 19, 20, and 20 (Supplementary Figure S6A, Additional File 1). However, under experimental conditions, the true CNP would be unknown. In all other simulation sets, the phylogenies estimated using $t_2 + t_3$ had lower median Robinson-Foulds distances than all other methods.

For completeness, we additionally calculated Robinson-Foulds distances on phylogenies estimated from consensus CNPs from summarizing over the nearest 10 cells. For tree A (ultrametric, maximally imbalanced), the median RF distances on mean consensus CNPs over the nearest 10 cells were 27, 27, and 26 for the Euclidean, cnp2cnp, and MEDICC distances, respectively (Supplementary Figure S7A, Additional File 1). Note that in order to estimate phylogenies using our $t_2 + t_3$ metric, all pairs of cells must be analyzed, and cannot benefit from the time savings of analyzing a selected subset of pairs of cells (see Optimal number of pairs to use). This is a weakness of our method, as analyzing all pairs of cells comes at an increased computational cost.

3.4 Discussion

We present a novel method, SCONCE2, that combines data across single cells in a manner that is grounded in a principled model of stochastic tumor evolution. It jointly calls copy number alterations in single cell sequencing of cancer cells with higher accuracy than competing methods on both simulated and real data. Additionally, SCONCE2 calculates an informative pairwise distance metric that can be used to estimate phylogenies with less error than other methods.

Similar to SCONCE, one weakness of SCONCE2 is the requirement for matched diploid cells in order to normalize GC content and mappability biases. These diploid cells must be sequenced under the same experimental conditions for proper GC content and mappability normalization, and may not be directly of interest to investigators, thereby potentially increasing cost. However, infiltrating diploid cells are often sequenced as a byproduct of single cell sequencing, and can be identified with orthogonal methods, such as cell sorting. For example, in the two datasets analyzed here, no additional sequencing was necessary to purposefully produce matched diploid sequencing data.

Additionally, SCONCE2 does not use SNPs or genotype likelihoods, or do any allelic phasing, to inform copy number calls or $t_2 + t_3$ estimates. Although calling SNPs in low coverage and noisy single cell data is difficult, incorporating genotype likelihoods can add information and increase confidence in these procedures. For example, using the allele frequency in variable single nucleotide sites can support concordant or rule out discordant copy number states. Furthermore, estimating the counts of variable sites on specific branches in $\mathcal{T} = [t_1, t_2, t_3]$ (see Figure 1) can increase confidence in branch length estimates. Adding genotype likelihoods of single nucleotide variants is the subject of future work.

Another weakness of SCONCE2 is that it takes longer to run, relative to other methods. However, if investigators are primarily interested in copy number calling, significant time can be saved by summarizing over a selective subset of pairs of cells (that is, noninformative pairs are not analyzed), described in Optimal number of pairs to use. But, if investigators are interested in estimating phylogenies using our $t_2 + t_3$ metric, all pairwise distances must be estimated to calculate a complete distance matrix (described in Phylogeny Estimation), thereby negating this time saving measure. Because the distance matrix dimensions and number of pairwise comparisons grow quadratically with the number of cells analyzed, the computational complexity and run time cost grows quickly. However, if investigators are interested in both copy number calling and phylogeny estimation using our $t_2 + t_3$ metric, after all pairwise parameter estimates are calculated (the most computationally intensive step), investigators have the flexibility to quickly call consensus copy number profiles over an arbitrary number of pairs. Despite the computational complexity of this model, we propose the increased accuracy of both copy number calls and phylogeny estimation outweighs the increased computational run time cost.

3.5 Conclusions

In conclusion, we present a principled method, SCONCE2, for simultaneously and accurately calling and aggregating copy number profiles across multiple tumor cells, and estimating pairwise evolutionary distances, using single cell whole genome sequencing. This work shows jointly analyzing cells in single cell experiments to leverage their shared evolutionary history increases accuracy in copy number calling and phylogeny estimation, with implications for deepening our understanding of tumor evolution.

3.6 Methods

Evolutionary process modeling

We first review the Markov processes introduced in [3]. Briefly, we assume an evolutionary process that is continuous in time but discrete along the length of the genome. However, notice that this is just an approximation, as the true process along the length of the genome is not Markovian (see Simulations).

One cell continuous time Markov process

The one cell continuous time process from [3] models the copy numbers of two adjacent genomic bins, in positions i and $i+1$ in the genome, on the same lineage (cell) with copy number $U, V \in \mathbb{S}_c = \{0, 1, \ldots, k\}$, respectively, where k is the maximum allowed copy

number. We assume that (U, V) evolve through time with the following rate parameters:

$$\alpha = \text{rate of } \pm 1 \text{ CNA} \tag{1a}$$
$$\beta = \text{rate of any CNA} \tag{1b}$$
$$\gamma = \text{relative rate of CNAs affecting both } U \text{ and } V \tag{1c}$$

which leads to the following instantaneous rate matrix for the joint process for two bins on one lineage: $\mathbb{Q} = \{q_{(U,V),(U',V')}\}$:

$$q_{(U,V),(U',V')} = \begin{cases} \gamma(\alpha+\beta) & \text{if } (U',V') = \begin{cases} (U+n, V+n) \\ (U-n, V-n) \end{cases}, n = 1 \\ \gamma\beta & \text{if } (U',V') = \begin{cases} (U+n, V+n) \\ (U-n, V-n) \end{cases}, n > 1 \\ \alpha+\beta & \text{if } (U',V') = \begin{cases} (U \pm n, V) \\ (U, V \pm n) \end{cases}, n = 1 \\ \beta & \text{if } (U',V') = \begin{cases} (U \pm n, V) \\ (U, V \pm n) \end{cases}, n > 1 \\ r_{(U,V)} & \text{if } (U',V') = (U,V) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

To ensure all rows sum to 0, we set the diagonal elements to the negative row sum, $r_{(U,V)} = -\sum_{(u',v') \neq (U,V)} q_{(U,V),(u',v')}$. Note that $\mathbb{Q}$ defines the instantaneous rate of events such as $(U', V') = (U+n, V-k), n > 0, k > 0$ (i.e., events where (U, V) are changed by different CNAs) to be equal to 0. However, (U, V) can be changed by different CNAs for any evolutionary time interval $t > 0$. Additionally, note we use the sum $\alpha + \beta$ as the rate for events with ± 1 CNA, to allow ± 1 copy number events to have a higher rate than larger magnitude copy number events.

The corresponding probability matrix, $\mathbb{P}$, of time dependent transition probabilities of adjacent bins changes from (U, V) to (U', V') is calculated as the matrix exponential

$$P_{(U,V),(U',V')}(t) = e^{\mathbb{Q}t} \tag{3}$$

where t is evolutionary time.

Two cell evolutionary process expansion

We now extend this single lineage process to describe the joint evolutionary process in two cells. Consider a pair of cells (A, B) and their most recent common ancestor in a tree, $\mathcal{T} = [t_1, t_2, t_3]$, where t_1 denotes the branch length of their shared history and t_2 and t_3 denote the branch lengths from divergence, at unobserved state Z, to cells A and B, respectively (see Figure 1).

Under this tree structure, adjacent bins in cells A and B have a shared evolutionary history for time t_1 from an ancestral diploid state (i.e., $D : (2, 2)$) to an intermediate unobserved state, $Z : (W, Y)$, with associated transition probability $P_{(2,2),(W,Y)}(t_1)$. After divergence, bins in cell A evolve from (W, Y) to $(CN_{iA}, CN_{i+1,A})$ in time t_2 with transition probability $P_{(W,Y),(CN_{iA},CN_{i+1,A})}(t_2)$, where $CN_{iA}, CN_{i+1,A} \in \mathbb{S}_c$ denote copy number in windows i and $i + 1$ for cell A. Similarly, bins in B evolve from (W, Y) to $(CN_{iB}, CN_{i+1,B})$ in time t_3 with transition probability $P_{(W,Y),(CN_{iB},CN_{i+1,B})}(t_3)$.

Approximating discrete Markov process along the genome

Next, we convert these continuous time process transition probabilities for adjacent bins in two cells into the transition probabilities for the approximating discrete Markov process for pairs of cells along the entire length of the genome, further described in Two Cell Hidden Markov Model Description. We do this by expanding the state space to the product space of the state space for each cell, $\mathbb{S}_c$. This expansion of the state space to the joint CN state for two cells is necessary as the correlation structure along the length of the genome prevents the use of standard tree-based dynamic programming algorithms such as Felsenstein's pruning algorithm [54].

The state space, $\mathbb{S}_d$, for this discrete process is composed of pairs (CN_{iA}, CN_{iB}), representing the copy numbers for window i in cell pair (A, B), where $CN_{iA}, CN_{iB} \in \mathbb{S}_c = \{0, 1, \ldots, k\}$ for a fixed maximal copy number k, such that

$$\mathbb{S}_d = \mathbb{S}_c \times \mathbb{S}_c = \{(0, 0), (0, 1), (0, 2), \ldots, (k, k)\} \tag{4}$$

We define matrix $\mathbb{F}(\mathcal{T}) = \{f_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}(\mathcal{T})\}$ as the transition probability of moving from state (CN_{iA}, CN_{iB}) in window i to $(CN_{i+1,A}, CN_{i+1,B})$ in window $i+1$, given evolutionary tree $\mathcal{T} = [t_1, t_2, t_3]$, for cell pair (A, B). Therefore, the matrix $\mathbb{F}(\mathcal{T})$ is defined as

$$\begin{aligned} f_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}(\mathcal{T}) = \sum_{W,Y \in \mathbb{S}_c} \Big(& P_{(2,2),(W,Y)}(t_1) \times \\ & P_{(W,Y),(CN_{iA},CN_{i+1,A})}(t_2) \times \\ & P_{(W,Y),(CN_{iB},CN_{i+1,B})}(t_3) \Big) \end{aligned} \tag{5}$$

which can be used to calculate a transition matrix, $\mathbb{M}(\mathcal{T})$, along the length of the genome for pairs of cells. This is done by dividing the joint probability of the CN state in both cells (A and B) in both bins (i and $i+1$), with the marginal probability of CN state in both cells (A and B) in bin i, i.e., dividing each entry in $\mathbb{F}(\mathcal{T})$ with the corresponding row sum:

$$\mathbb{M}(\mathcal{T}) = \{m_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}\}(\mathcal{T}) \tag{6a}$$

$$m_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}(\mathcal{T}) = \frac{f_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}(\mathcal{T})}{\sum_{(c,d) \in \mathbb{S}_d} f_{(CN_{iA},CN_{iB}),(c,d)}(\mathcal{T})} \tag{6b}$$

We have thereby constructed a process with state space on the copy numbers of pairs of cells, $\mathbb{S}_d$. The matrix $\mathbb{M}(\mathcal{T})$ gives the probabilities of observing transitions from (CN_{iA}, CN_{iB}) in window i to $(CN_{i+1,A}, CN_{i+1,B})$ in window $i+1$, along the genome, for cell pair (A, B), given evolutionary tree $\mathcal{T}$. We also note that the process along the length of the genome is not Markovian, as breakpoints appear in pairs, inducing an inherently non-Markovian correlation structure (see also [3]). However, to facilitate computation, we will approximate the process as a Markovian process with transition probabilities given by $\mathbb{M}(\mathcal{T})$. We note that while this model approximates the evolutionary process and paired nature of breakpoints via the genome wide transition matrix $\mathbb{M}(\mathcal{T})$, it does not explicitly model pairs of breakpoints jointly, potentially leading to unpaired breakpoints. This Markov chain will then be used for inferences in a Hidden Markov Model framework with emission probabilities similar to those described in [3].

Two Cell Hidden Markov Model Description

Expanding on the framework of [3], we define a Hidden Markov Model (HMM) [77–79] to infer copy number across the genome for pairs of tumor cells, using binned read depth data.

Recall that cells A and B are associated with the evolutionary tree, $\mathcal{T}$, shown in Figure 1. The sample space of read data, $\mathbb{A}$, is composed of pairs of observed per window read count values, (x_{iA}, x_{iB}), $x_{iA}, x_{iB} \in \mathbb{N}_0 = \{0, 1, 2, \dots\}$:

$$\mathbb{A} = \mathbb{N}_0 \times \mathbb{N}_0 = \{(0,0), (0,1), (0,2), \dots\} \tag{7}$$

This HMM uses the state space of paired copy numbers, $\mathbb{S}_d$, defined in Equation 4 and the transition matrix $\mathbb{M}(\mathcal{T})$, defined in Equation 6.

Emission probabilities

Assuming conditional independence between cells, the emission probabilities of the HMM are:

$$\mathbb{P}(X_{iA} = x_{iA}, X_{iB} = x_{iB} | CN_{iA}, CN_{iB}) = \mathbb{P}(X_{iA} = x_{iA} | CN_{iA}) \mathbb{P}(X_{iB} = x_{iB} | CN_{iB}) \tag{8}$$

As these probabilities are calculated similarly for cells A and B, we only describe the derivation for cell A (note: we previously described this derivation in [3]).

We assume X_{iA} follows a negative binomial distribution, such that

$$\mathbb{E}(X_{iA}) = \lambda_{iA} = \left(CN_{iA} \times \frac{\mu_i}{2} \right) \times s_A + \varepsilon \tag{9}$$

$$X_{iA} \sim NegBinom\left(\lambda_{iA}, \sigma^2_{iA} = a\lambda^2_{iA} + b\lambda_{iA} + c\right) \tag{10}$$

where

$$CN_{iA} = \text{the copy number in window } i \text{ for cell } A \tag{11a}$$
$$\mu_i = \text{the mean diploid read depth in window } i \tag{11b}$$
$$\varepsilon = \text{constant sequencing error term} \tag{11c}$$
$$s_A = \text{library size scaling factor for cell } A \tag{11d}$$
$$\{a, b, c\} = \text{constants learned from diploid data} \tag{11e}$$

Estimation of constants $\{a, b, c\}$ is described in Initial independent parameter estimation using SCONCE. In the following, we will describe the full SCONCE2 estimation procedure in detail.

Detailed SCONCE2 pipeline

Given binned read depths for tumor and matched diploid cells, joint copy number calling in SCONCE2 takes place in four main steps: 1) independently estimating model parameters and copy number profiles for each cell using SCONCE, 2) combining independent parameter estimates across cells, 3) estimating tree branch lengths for each cell pair, and 4) creating summarized copy number profiles. This process is illustrated in Figure 2.

Initial independent parameter estimation using SCONCE

We first estimate constants $\{a, b, c\}$, defined in Equations 10 and 11e, using maximum likelihood on diploid cells only, as previously described in [3]. We note that most single cell tumor sequencing projects naturally also produce data from non-tumor diploid cells as part of standard sequencing techniques, and that these cells conveniently can be used for standardization [4, 42, 60, 80–84].

In order to obtain initial estimates of all model parameters, we analyze all tumor cells independently through SCONCE, described in detail in [3] and briefly summarized here. This is done to avoid the computational cost of joint estimation for all model parameters across all pairs of cells. The SCONCE pipeline first estimates the transition matrix of an unconstrained CN HMM, with associated library size scaling factor s_A, for each cell using a modified Baum-Welch [85] algorithm. These estimates are then used to obtain initial starting points for each model parameter for an optimization of the likelihood function using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [57]. This results in parameter estimates $\{\hat{s}_A, \hat{\alpha}_A, \hat{\beta}_A, \hat{\gamma}_A, \hat{t}_A\}$, for cell A. Recall s_A is the library size scaling factor, defined as the coverage for the cell relative to the average diploid library size, $\{\alpha, \beta, \gamma\}$ are the instantaneous rates for copy number events, and t_A is the total branch length from the ancestral diploid cell to cell A (see the red block in Figure 2.).

Combining parameter estimates across multiple cells

To analyze shared evolutionary history between n cells, we first combine independent estimates across all cells of the transition rate parameters, $\{\alpha, \beta, \gamma\}$, assumed to be shared among all cells, using the median to form joint estimates:

$$\hat{\alpha} = \text{median}(\hat{\alpha}_A, \hat{\alpha}_B, \ldots, \hat{\alpha}_n) \tag{12a}$$

$$\hat{\beta} = \text{median}(\hat{\beta}_A, \hat{\beta}_B, \ldots, \hat{\beta}_n) \tag{12b}$$

$$\hat{\gamma} = \text{median}(\hat{\gamma}_A, \hat{\gamma}_B, \ldots, \hat{\gamma}_n) \tag{12c}$$

We note that a full joint optimization could possibly lead to better model parameter estimates, especially for highly heterogeneous tumor populations, as the median is not strongly affected by extreme or highly variable individual estimates. However, we opted not to pursue the estimation of such estimates because of considerations of computational efficiency. This is illustrated in the yellow block in Figure 2.

Estimating pairwise tree branch lengths

Next, we estimate parameters of the joint two-cell process for all $\binom{n}{2}$ pairs of cells. The branch lengths of tree $\mathcal{T} = [t_1, t_2, t_3]$, are specific to each pair of cells, and branch length estimates from SCONCE, $\hat{t}$, are used to inform the initial optimization starting point for $\mathcal{T}$. For example, for pair (A, B), the initial branch length estimates, denoted with *, are:

$$t_1^* = \frac{min(\hat{t}_A, \hat{t}_B)}{2} \tag{13a}$$

$$t_2^* = \hat{t}_A - t_1^* \tag{13b}$$

$$t_3^* = \hat{t}_B - t_1^* \tag{13c}$$

For each pair of cells, we use the BFGS algorithm to maximize the forward log likelihood in order to estimate $\mathcal{T}$. To calculate the forward log likelihood of an observed sequence, the HMM is reset into the initial probability vector, defined as the steady state distribution, at the beginning of each chromosome to ensure chromosomal independence.

Because each set of branch lengths, $[t_1, t_2, t_3]$, is specific to each pair of cells, this procedure is trivially parallelizable (see the green block in Figure 2).

Summarized copy number calling

After pairwise branch lengths are estimated, we use the Viterbi algorithm [86, 87] to estimate the most likely joint copy number profile for each pair of cells. If cell A appears in $n-1$ pairs, this results in $n-1$ separate CNP estimates for cell A. In order to calculate a single consensus copy number profile, $CN_{A,consensus}$, we use either the mean (default), median, or mode of the CN in each window among the $n-1$ estimates.

While adding more information to each consensus copy number profile by summarizing across multiple cells initially increases accuracy, summarizing across too many divergent cells is not optimal because more accurate estimates about each cell are obtained using closely related cells than highly divergent cells (Figure 5). Therefore, in order to balance combining data from multiple cells and maintaining cell specificity, the user can also choose to summarize across a subset of the κ nearest neighbors for each cell, instead of all $n-1$ pairs a particular cell appears in. The nearest neighbors for each cell is defined by the Euclidean distance between individual copy number profiles from SCONCE. Then, the consensus copy number profile is calculated only across the κ selected pairs.

Note, because of the genomic binning procedure, true copy number events may be split across bin boundaries or be completely contained within one bin, resulting in bins with non-integer average copy number. Using the mean and median summary functions can result in non-integer copy number calls, which more accurately represent the underlying biology as genomes are not truly organized in discrete bins. However, many downstream tools for single cell analyses require integer copy number profiles, so these values may need to be rounded for downstream analyses.

Simulations

In order to evaluate the accuracy of SCONCE2, we use the Line Segments model from SCONCE [3], which simulates copy number events on a fixed length reference genome as additions or deletions to a collection of line segments, and does not impose a maximum copy number limit. Note that although copy number events change the number and length of line segments, the reference genome length is constant. Additionally, copy number events create pairs of breakpoints at either end of the event, which are explicitly maintained in this simulation model, unlike the approximating discrete Markov process in the SCONCE2 inference model (see Approximating discrete Markov process along the genome), thereby making the simulation model more biologically realistic.

While we previously used neutral coalescent simulations [3]to define trees, we here instead adjust the tree structure and the length of the branch leading to the root (ie, time to first divergence event) to examine a range of highly different tree structures, each including 128 cells. We specify two ultrametric trees with uniform branch lengths of 1/128, where tree A is fully pectinate/maximally imbalanced and tree B is perfectly balanced, and two non ultrametric trees, where tree C has uniform internal and terminal branch lengths of 1/128, and tree D has uniform internal branch lengths of 1/128 and logarithmically decaying terminal branch lengths. These tree structures represent extremes in terms of how balanced the tree is and in terms of deviations from a molecular clock (ultrametric property). Following the definitions of [26], tree A models branching evolution, tree B models neutral evolution, and trees C and D model linear evolution. Under certain conditions, the structure of tree B can also be adjusted to model punctuated evolution [26] if the branch leading to the root is lengthened relative to the internal tree branches, such that more mutations fall on the shared

ancestral/root branch compared to external branches. For illustration, the tree structure for 8 cells is shown for each dataset in Figure S1.

For each tree, the total tree height (longest path from the root to a leaf) was scaled to 1, and the branch leading to the root was set to length 1. Simulated reference genome lengths were set to 100, with amplification and deletion rates and expected lengths shown in Supplementary Table S1, Additional File 2 (note that genomic length units are arbitrary, where expected copy number event lengths are defined relative to the genome length). As previously described in [3], the locations of copy number events follow a Markov process, and the lengths of copy number events follow a truncated exponential distribution.

To simulate read depths across the genome, the human reference genome was divided into 12,397 windows (equalling the number of 250kb non overlapping uniform windows in hg19), and the number of reads falling into each window was simulated from a negative binomial distribution with parameter $r = 50$. This results in files listing genomic window coordinates and number of reads observed in that window, for every simulated cell (similar to output from `bedtools coverage` [58] on real data). The total expected number of reads for each cell was set to 4,000,000 (322.7 expected reads per window) to approximate the observed number of reads per 250kb window (mean 316.1, median 351.2) in diploid cells from [4]. Note the actual number of total observed reads in each cell is random.

In order to ensure tree B was a perfectly balanced binary tree and to be consistent between tree structures, read depths for 128 tumor cells and 100 diploid cells were simulated for each tree. Read depth across diploid cells was averaged per window for each tree. Tumor cells from each tree were divided into five non overlapping subsets of 20 cells to create test sets. Although healthy cells were shared for each analysis run, each test set was otherwise analyzed independently from other test sets from the same tree.

All parameter files used to generate simulations are available on GitHub, along with examples of simulated data.

Real data preprocessing

We applied SCONCE2 to two published single cell breast cancer datasets, from [4] and [5], a cancer type known for their frequent CNAs [59]. Both of these datasets were processed as previously described [3]. Briefly, for the [4] dataset, we trimmed reads using cutadapt [64] and trimmomatic [65], removed low complexity reads with prinseq [66], aligned reads to hg19 using bowtie2 [67], removed reads with q scores less than 20 using samtools [68], and removed PCR duplicates using picard [69]. For the [5] dataset, we split downloaded preprocessed bam files into cell specific bam files using pysam [70], and removed reads with q scores less than 20 using samtools [68]. Finally, we used `bedtools coverage` to count per window read depth for each cell [58]. Cells previously and orthogonally identified as diploid cells in [4] served as the matched normal. For the [5] dataset, cells from subset A were used as the diploid samples, as previously described [60].

Running other methods

For benchmarking, we limit our comparisons to other copy number only methods (that is, no SNP or phasing information is used): SCONCE [3] and AneuFinder [49, 50] for copy number accuracy, and the cnp2cnp [75] and MEDICC [74] distances for phylogeny building.

Briefly, we ran AneuFinder with default parameters, with the exception of skipping GC and mappability corrections to avoid overcorrecting, as we did not include GC or mappability bias in our simulations. To benchmark SCONCE2's copy number calling, we first ran SCONCE [3] with default parameters (`k=10`). To run AneuFinder [49, 50], we skipped the GC and mappability corrections steps to avoid over correcting, as our simulation model does not include GC or mappability biases. We directly ran AneuFinder's `findCNVs` function (default parameters: `method="edivisive", R=10, sig.lvl=0.1`). We extracted copy number calls from the resulting the `copy.number` element, and used `bedtools intersect` [58] to split large segments into 250kb windows.

To evaluate SCONCE2's $t_2 + t_3$ distance metric in phylogeny estimation, we compared to the cnp2cnp distance [75] and the MEDICC distance [74]. To run cnp2cnp, we first converted and rounded called CNPs into fasta files, then ran cnp2cnp in matrix mode with default parameters (`-m matrix -d any`). Because the cnp2cnp metric depends on the input sample ordering and is not symmetric, we repeated this process on the reversed sample ordering, and summed the two resulting distance matrices to make a symmetric metric. To calculate the MEDICC distance, we used the ZZS implementation of the MEDICC algorithm in the cnp2cnp program to remove the maximum copy number limit of 4 in the original MEDICC software, and ran it on the same fasta files (`-m matrix -d zzs`). Full scripts to run other methods are provided on GitHub (`runAneufinder.sh` and `runCnp2cnp.sh`).

Phylogeny estimation and Robinson-Foulds distance calculations

To estimate phylogenies, distance matrices were first read into R [88]. Next, we applied neighbor-joining [31, 32], implemented in the `ape` [89] package, to each distance matrix to estimate phylogenies.

To calculate Robinson-Foulds distances between inferred trees and true trees, we first used the `read.tree` function in the `ape` [89] package to read true trees in Newick format into R. Next, we used the `treedist` function from the `phangorn` [90, 91] package to calculate the Robinson-Foulds distance. Full scripts to estimate phylogenies from distance matrices and calculate Robinson-Foulds distances between phylogenies are available on GitHub (`readTreeBranches.R` and `plotRFdist.R`).

3.7 Appendix

Abbreviations

- BFGS: Broyden–Fletcher–Goldfarb–Shanno algorithm [57]

- CN: copy number
- CNA: copy number alteration(s)
- CNP: copy number profile
- HMM: hidden Markov model [77–79]
- RF distance: Robinson-Foulds distance [76]
- SNP: single nucleotide polymorphism
- SSE: sum of squared errors

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Availability of data and materials

SCONCE2 is implemented in C++11, requires the Boost C++ Libraries (developed on v1.65.1)and the GNU Scientific Library (developed on v2.4) [63], has been developed and tested on Ubuntu 18.04.6, and is freely available from https://github.com/NielsenBerkeleyLab/sconce2. The simulation program (written in C) and corresponding parameter files, all R scripts (developed on R v4.1.2) [88] needed to preprocess diploid data (avgDiploid.R and fitMeanVarRlnshp.R), and R plotting scripts (readBedFilesPairs.R, readTreeBranches.R, plotBetterBoundaries.R, plotDiminishingReturnsNumPairs.R, plotIllustrativeTrees.R, plotRFdist.R, plotSSEandBreakpointPairs.R, plotTreeBranchCorrelation.R) are also on GitHub. Plotting scripts require the R packages `ape` [89], `cowplot` [92], `ggplot2` [93], `ggtree` [94–96], `grid` [88], `gtools` [97], `phangorn` [90, 91], plyr [98], `reshape2` [99], `scales` [100], and `stringr` [101].

We analyzed two previously published real datasets. Data from [4] is available at the Sequence Read Archive (SRA) under accession number SRR054616. The 10x dataset from [5] is available at https://cf.10xgenomics.com/samples/cell-dna/1.1.0/breast_tissue_aggr_10k/breast_tissue_aggr_10k_web_summary.html.

Competing interests

The authors declare that they have no competing interests.

Funding

This work was supported by the National Institutes of Health [R01GM138634-01 to R.N.]. The funding body played no role in the design of the study and collection, analysis, and interpretation of data, or in writing the manuscript.

Authors' contributions

SH designed and implemented the SCONCE2 pipeline, and performed all data analysis. RN conceived the study and developed the simulation program. SH and RN wrote the manuscript. All authors read and approved the final manuscript.

3.8 Figures

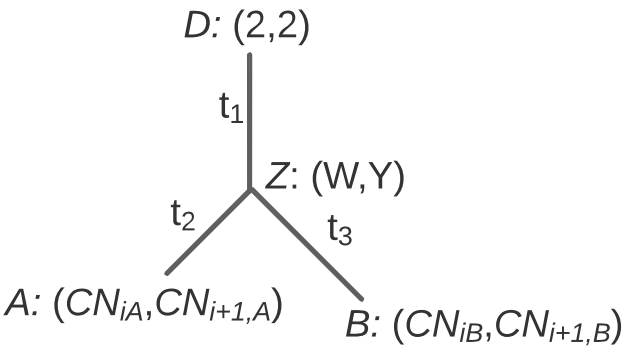


Figure 1: Pairwise tree structure, showing the tree, $\mathcal{T} = [t_1, t_2, t_3]$, between the pair of cells A and B, where the branch with length t_1 represents their shared evolutionary history from an ancestral diploid cell, D, before diverging at the unobserved state Z. The branches with lengths t_2 and t_3 show independent evolution to cells A and B, respectively. Copy number in adjacent bins along the genome is shown in parentheses.

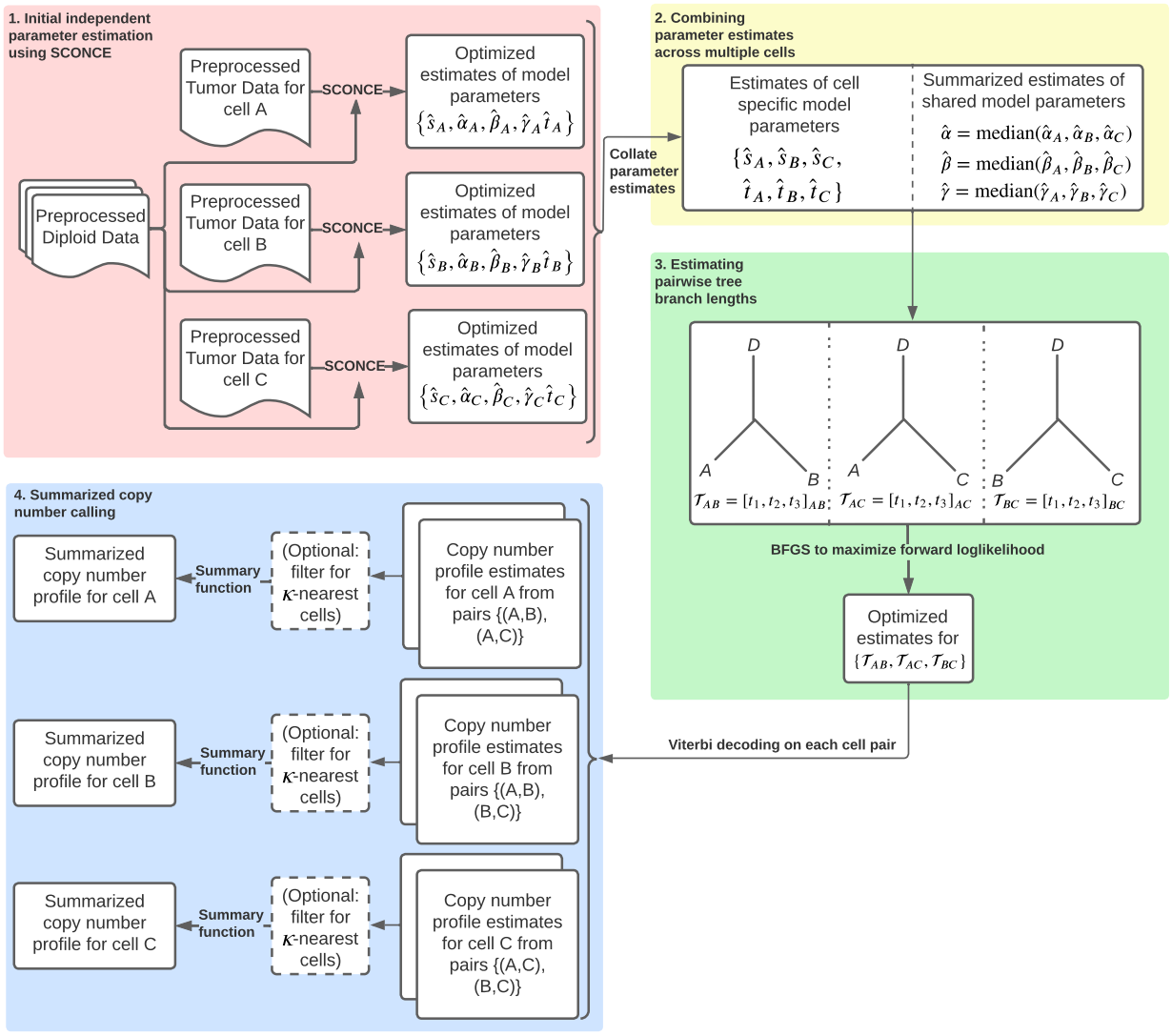


Figure 2: Detailed flowchart of the SCONCE2 pipeline. We demonstrate the pipeline with cell triplet $\{A, B, C\}$, without loss of generality. Each tumor cell is initially independently analyzed through SCONCE, which gives parameter estimates and copy number profiles for each cell (red box). These parameter estimates are then summarized (yellow box), and branch lengths for tree $\mathcal{T} = [t_1, t_2, t_3]$ are estimated for each pair of cells (green box). Finally, for each cell, paired copy number profiles are summarized into a consensus copy number profile (blue box). Each step is fully described in Detailed SCONCE2 pipeline.

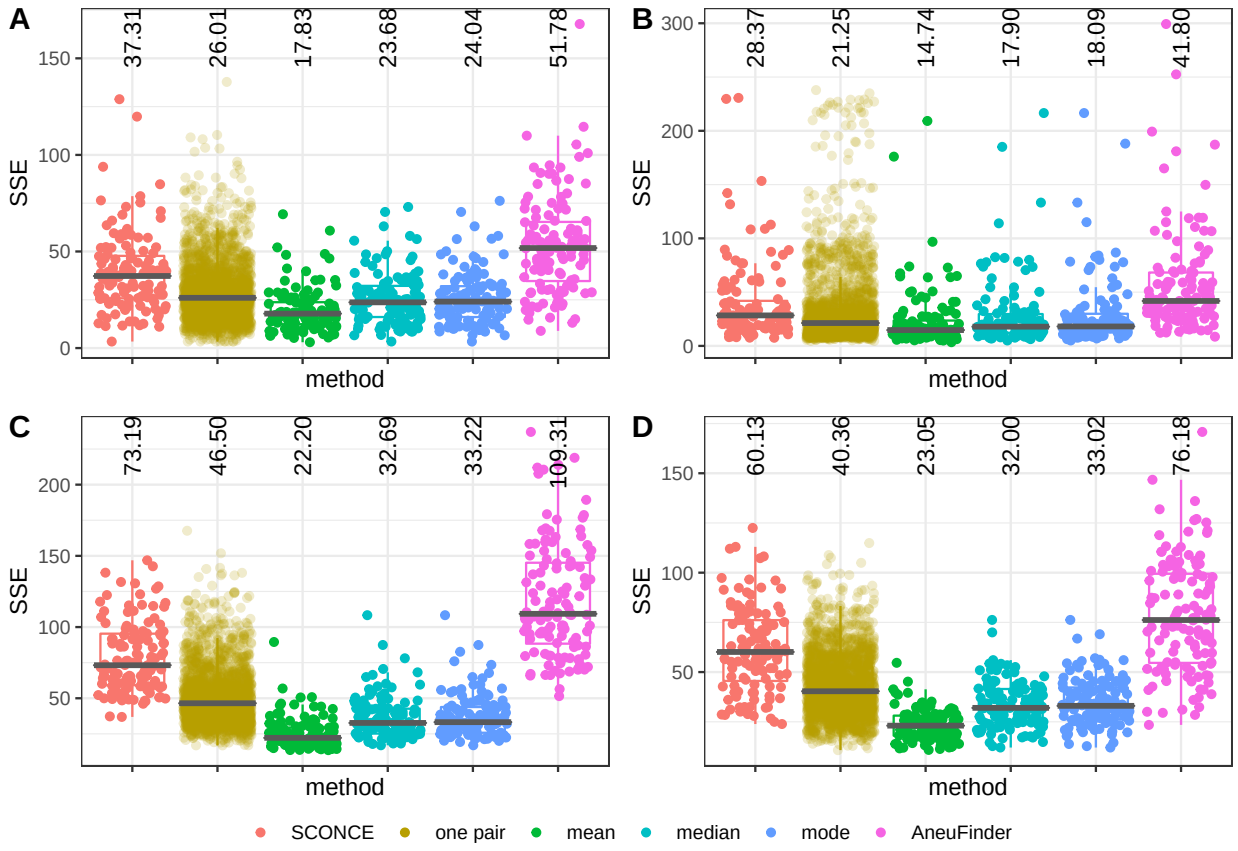


Figure 3: Boxplots of genome wide sum of squared error (SSE) between true simulated copy number profiles and inferred copy number profiles, across methods. SSE results are shown for cell specific CNPs from SCONCE (independent cell inference); joint inference on each pair of cells (one pair); summary functions across all pairs of cells (mean, median, and mode); and AneuFinder. Different methods are shown across the x-axis, and SSE is shown on the y-axis. Median SSE for each method is printed at the top of each column. Each dot represents one cell (note, in "one pair", each cell appears multiple times), and the median SSE is printed at the top of each column. Panel letters correspond to tree labels A (maximally imbalanced ultrametric tree), B (perfectly balanced ultrametric tree), C (maximally imbalanced non ultrametric tree with uniform branch lengths), and D (maximally imbalanced non ultrametric tree with logarithmically decaying branch lengths). SSE results are consistently lower when using data from multiple cells.

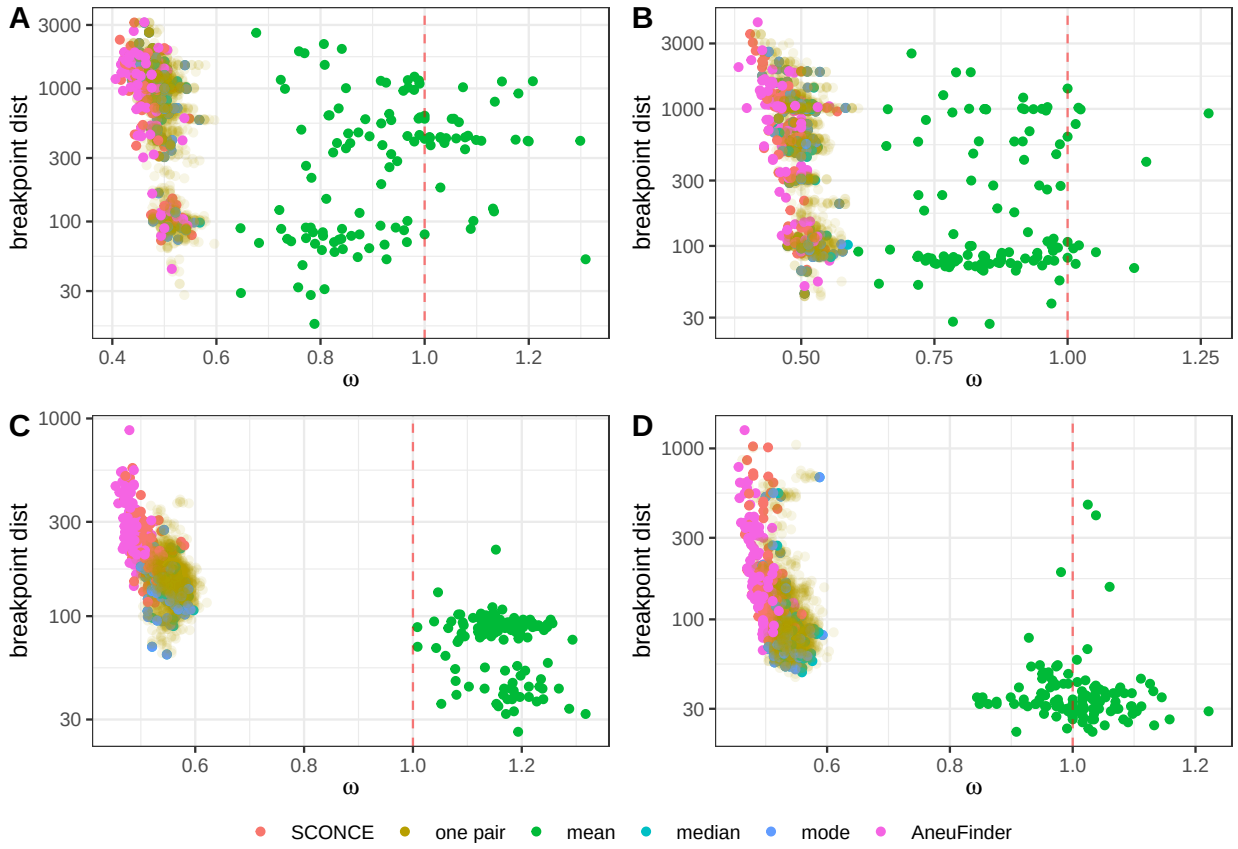


Figure 4: Breakpoint detection accuracy results across methods. Total distance to nearest breakpoint is shown on the y-axis, and $\omega = \frac{\#\text{ inferred breakpoints}}{\#\text{ true breakpoints}}$ is shown along the x-axis. Each dot represents one cell, colored by method. Methods with the highest breakpoint detection accuracy cluster near $\omega = 1$ (vertical red dotted line) and have lowest total breakpoint distance. Each panel corresponds to a simulation set (A-D). In all simulation sets, using the mean, median, and mode have lower total breakpoint distance than independent cell analyses. Furthermore, using the mean results in ω values closest to 1, as it is able to infer fractional copy numbers.

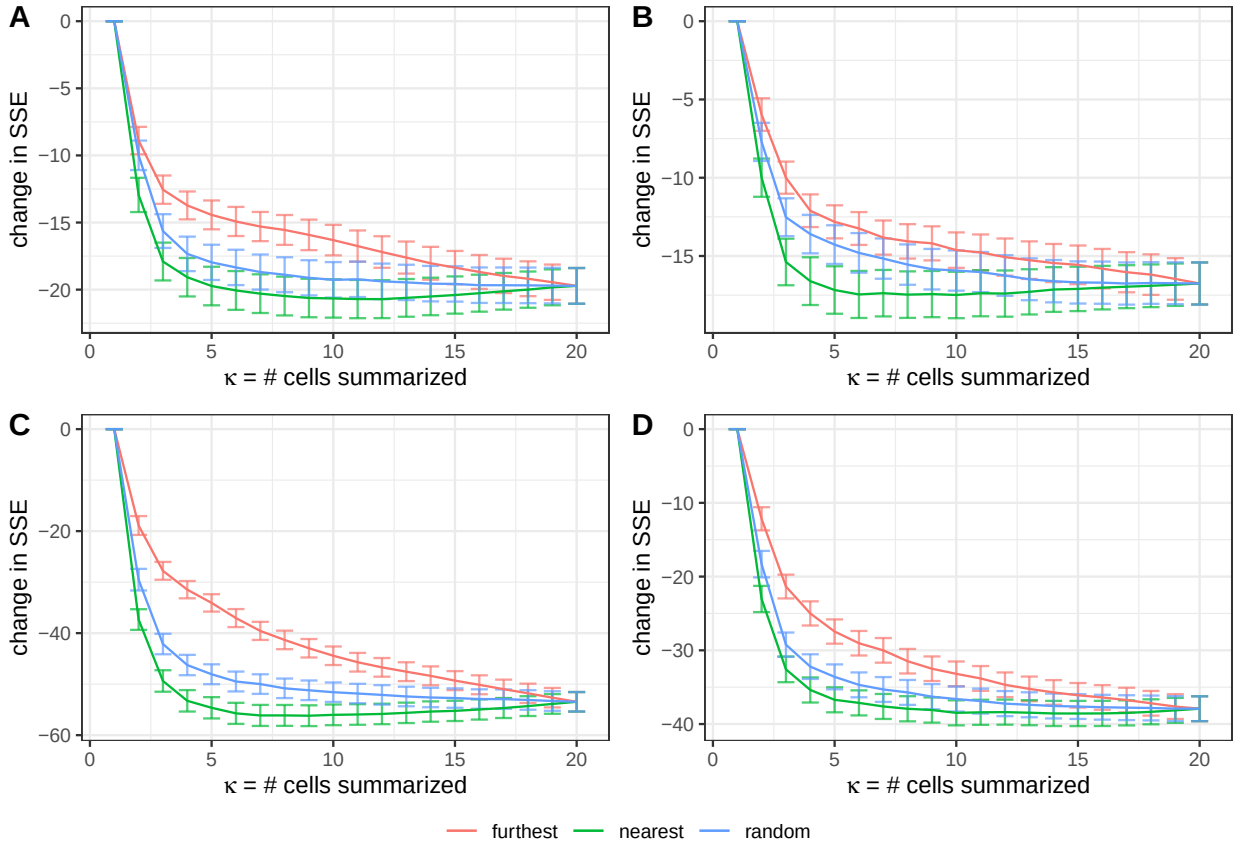


Figure 5: Change in SSE relative to SCONCE, as more cells are added to the consensus (mean) analysis. Number of cells, κ, in the joint analysis is shown along the x-axis, and change in SSE (relative to SCONCE) is shown on the y-axis. Each line shows the mean change in SSE across cells, with error bars showing ± 1 standard deviation. Colors show cell ordering, where furthest denotes adding the least similar cells first (i.e., furthest distance as defined by the Euclidean distance on SCONCE profiles) and nearest denotes adding the most similar cells first. Across all datasets and cell orderings, SSE quickly drops as more cells are added, with adding the nearest cells (green line) showing the fastest improvement. However, for this ordering, the decrease in SSE levels off after approximately 10 cells are added, and then slightly rises as more cells are added, due to rare copy number events getting averaged out.

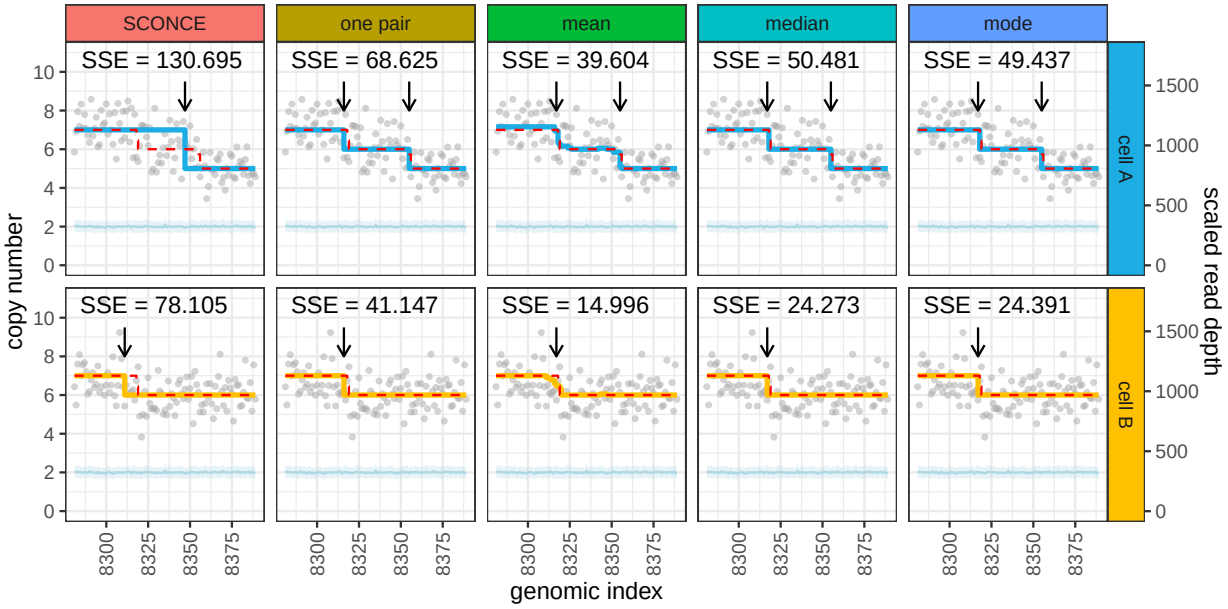


Figure 6: Jointly calling CNAs and summarizing data across multiple cells results in more accurate boundary detection. Inferred copy number profiles and read depth data are shown for two cells (A: 111, B: 59) simulated from tree C (maximally imbalanced, uniform branch lengths). Genomic index shows 110 250kb windows along the x-axis, while the y-axis shows copy number (left) and read depth (right). Gray dots show per window read depth, the light blue line and band show the mean and variance of the diploid read depth, the dotted red line shows the true simulated copy number, and the blue and yellow lines show the inferred copy number calls for each cell. Arrows denote inferred breakpoints, and SSE values are listed for each subpanel. Subpanels show results from different copy number calling methods: SCONCE (independent analysis); one pair (analysis of cells A and B as a pair); mean, median, and mode (consensus calls from summarizing paired CNPs for cells A and B across all relevant pairs of cells). Breakpoint detection accuracy increases as more cells are included in the joint analysis.

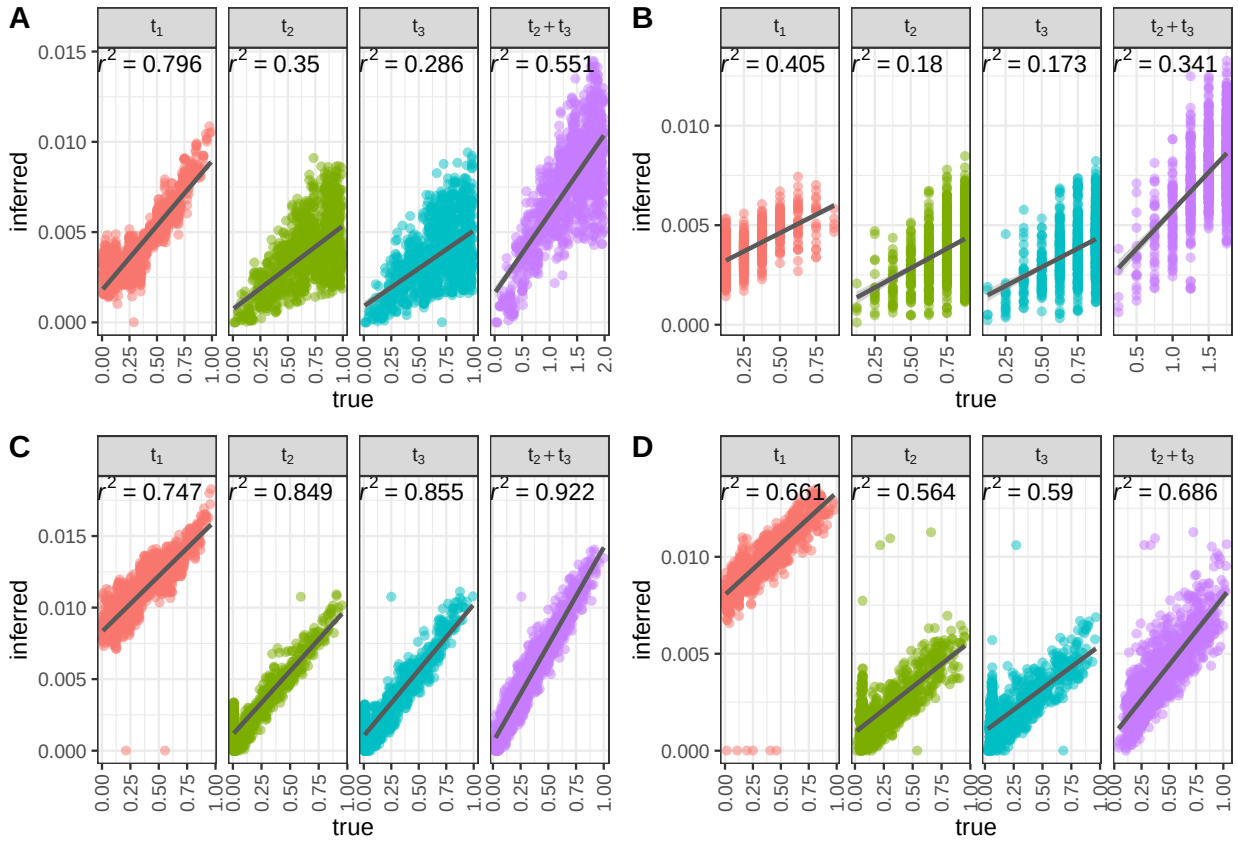


Figure 7: Correlation between true branch lengths and estimated branch lengths across simulation sets. Each dot represents one pairwise branch length estimate, with true node distance on the x-axis and estimated branch length on the y-axis. R^2 values from a linear regression on branch length (dark gray lines) is shown for each subpanel. Across all simulation scenarios (panels A-D correspond to trees A-D), SCONCE2 consistently predicts t_1 and $t_2 + t_3$ with high R^2 values.

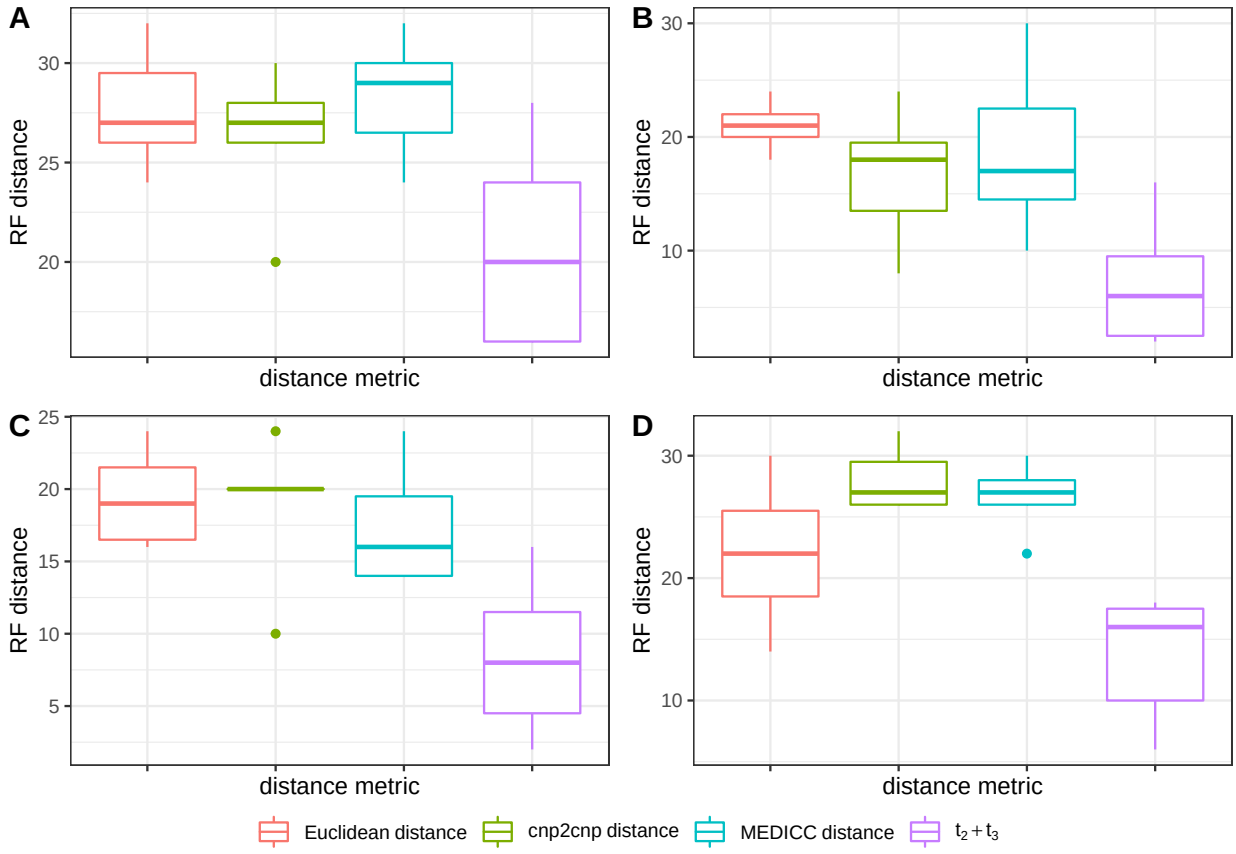


Figure 8: Robinson-Foulds (RF) distances between true and inferred phylogenies. Phylogenies were estimated using neighbor-joining on $t_2 + t_3$ estimates, and on the Euclidean, cnp2cnp, and MEDICC distances between true CNPs and CNPs inferred from SCONCE. Methods are shown along the x-axis, while Robinson-Foulds distances are shown on the y-axis. Across multiple parameter sets (panels A-D correspond to trees A-D), using $t_2 + t_3$ estimates resulted in a lower Robinson-Foulds distance from the simulated tree, relative to all other inferred phylogenies.

3.9 Supplementary Material

Supplementary Notes

Tree illustrations

We simulated 128 cells from four distinct tree structures, described in Simulations. Small 8 cell trees are shown for illustration purposes in Supplementary Figure S1, Additional File 1. Tree A is ultrametric and maximally imbalanced, tree B is ultrametric and perfectly balanced, tree C is not ultrametric and maximally imbalanced with uniform branch lengths, and tree D is not ultrametric and maximally imbalanced with logarithmically decaying branch lengths.

SSE and breakpoint distance accuracy over the nearest 10 cells

Across parameter sets, the median sum of squared error (SSE) between the true and inferred copy number profile for each cell is lowest when each cell's consensus profile is calculated by summarizing across the nearest 10 cells, rather than over all 20 in each subset, as rare events aren't averaged out. The nearest cells are determined by the lowest Eucliden distance on each cell's SCONCE profile. Similar to Figure 3, SSE is shown across parameter sets for CNPs from SCONCE (independent inference); from each pairwise analysis; summarized across pairs using mean, median, and mode; and from AneuFinder in Supplementary Figure S2, Additional File 1.

Additionally, similar to Figure 4, we show breakpoint detection accuracy in Supplementary Figure S3, Additional File 1 when consensus profiles are calculated by summarizing over each cell's nearest 10 cells. Median breakpoint distance and $\omega = \frac{\text{\# inferred breakpoints}}{\text{\# true breakpoints}}$ values are given in Supplementary Tables S5 and S6, Additional File 2.

Median Breakpoint Distance and ω Values

To evaluate breakpoint detection accuracy, we summed the distance from each true breakpoint to its nearest inferred breakpoint. We define $\omega = \frac{\text{\# inferred breakpoints}}{\text{\# true breakpoints}}$, such that highest accuracy is indicated by low breakpoint distance and ω close to 1. Median breakpoint distance and ω values are given across all analyzed cells for each tree in Supplementary Tables S2 and S3, Additional File 2. Similarly, median breakpoint distance and ω values when consensus methods only summarize across the 10 nearest cells are given in Supplementary Tables S5 and S6, Additional File 2.

Robinson-Foulds Distance Values

SCONCE2 estimates branch lengths for tree $\mathcal{T} = [t_1, t_2, t_3]$ for every pair of cells. To evaluate the usefulness of $t_2 + t_3$ as a distance metric for phylogeny building, we first calculated distance matrices using $t_2 + t_3$. For comparison, we calculated distance matrices using several different distance metrics (Euclidean distance, cnp2cnp, and MEDICC) at different points in

the SCONCE2 pipeline (after SCONCE (independent inference); on summarized CNPs from the mean, median, and mode). As a sanity check, we also calculated distance matrices on the true simulated CNPs (i.e., removing all noise from the copy number inference process). Next, we applied neighbor-joining to these distance matrices and calculated the Robinson-Foulds (RF) distance between each resulting tree and the true simulated tree. For each cell subset in each tree, RF distances are plotted in Supplementary Figure S6, Additional File 1, and median RF distances are given in Supplementary Table S7, Additional File 2.

Using the Euclidean distance is remarkably stable to variations in CNPs. Both the cnp2cnp and MEDICC metrics had the highest median RF distances for SCONCE, and the lowest on the true CNPs, with distances from mean, median, and mode lying between those two extremes. Across all parameter sets, using $t_2 + t_3$ resulted in median RF distances lower than other methods, with the exceptions of using the true CNPs. However, it stands to reason that one would not have access to noiseless copy number profiles in experimental conditions.

Additionally, we calculated the above distance matrices and ran neighbor-joining on CNPs where consensus profiles were summarized across only the nearest 10 cells. However, analyzing only the nearest 10 cells means some cell pairs get skipped, and therefore do not have an $t_2 + t_3$ estimate. To address this, we only calculated RF distances on trees estimated from other metrics.RF distances are plotted in Figure S7, and median RF distances are given in Supplementary Table S8, Additional File 2.

Detailed SCONCE2 pipeline

The SCONCE2 pipeline, shown in Figure 2, consists of four main steps: 1) Tumor cells are first independently analyzed through SCONCE to estimate model parameters (for cell A, $\{s_A, \alpha_A, \beta_A, \gamma_A, t_A\}$) and cell specific copy number profiles. 2) Shared parameter estimates, $\{\alpha, \beta, \gamma\}$, are then summarized across all cells using the median. 3) The branch lengths in tree $\mathcal{T} = [t_1, t_2, t_3]$ are then independently estimated for each pair of cells. Finally, the Viterbi decoding is used to calculate copy number profiles for each pair of cells, and 4) cell specific copy number profiles are summarized across pairs. Optionally, only the κ nearest neighbor cells (defined by the Euclidean distance on the SCONCE copy number profiles) are summarized for a given cell.

Supplementary Figures

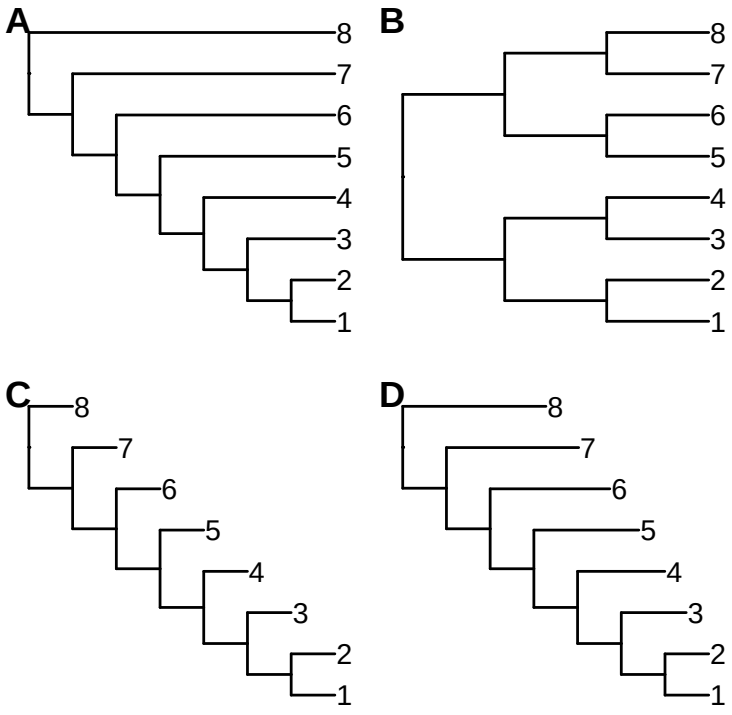


Figure S1: Tree structure for each simulated data set. Although each tree contained 128 cells, only 8 cells for each tree is shown for brevity.

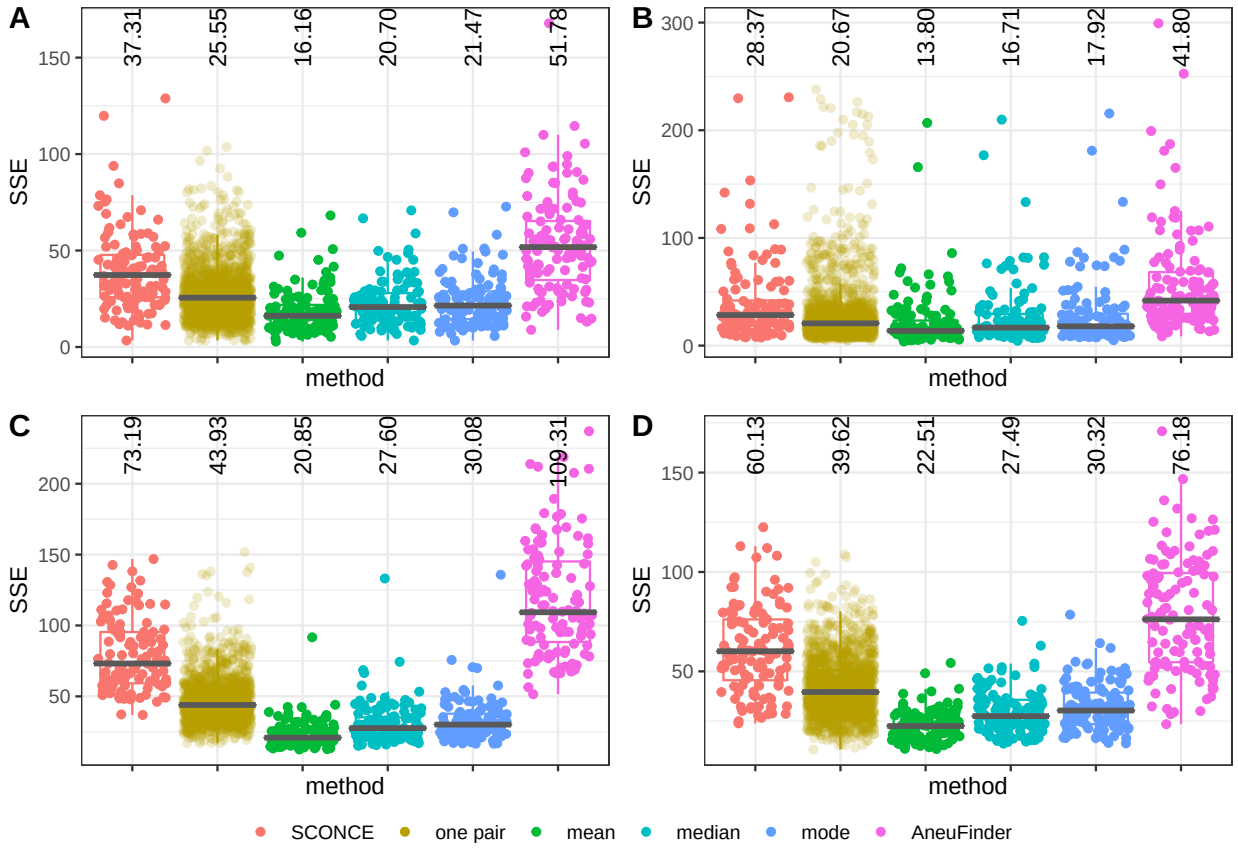


Figure S2: Boxplots of SSE across parameter sets and CNP calling method. Here, for summary methods (mean, median, and mode), each cell's consensus CNP is calculated over its nearest 10 cells. Each dot shows one cell, CNP calling method is on the x-axis, and SSE is on the y-axis, with median SSE values printed at the top of each column. The median SSE is indicated with a black bar and printed at the top of each column. Across parameter sets, calculating consensus calls results in lower SSE than both competing methods and summarizing over all pairs of cells (Figure 3).

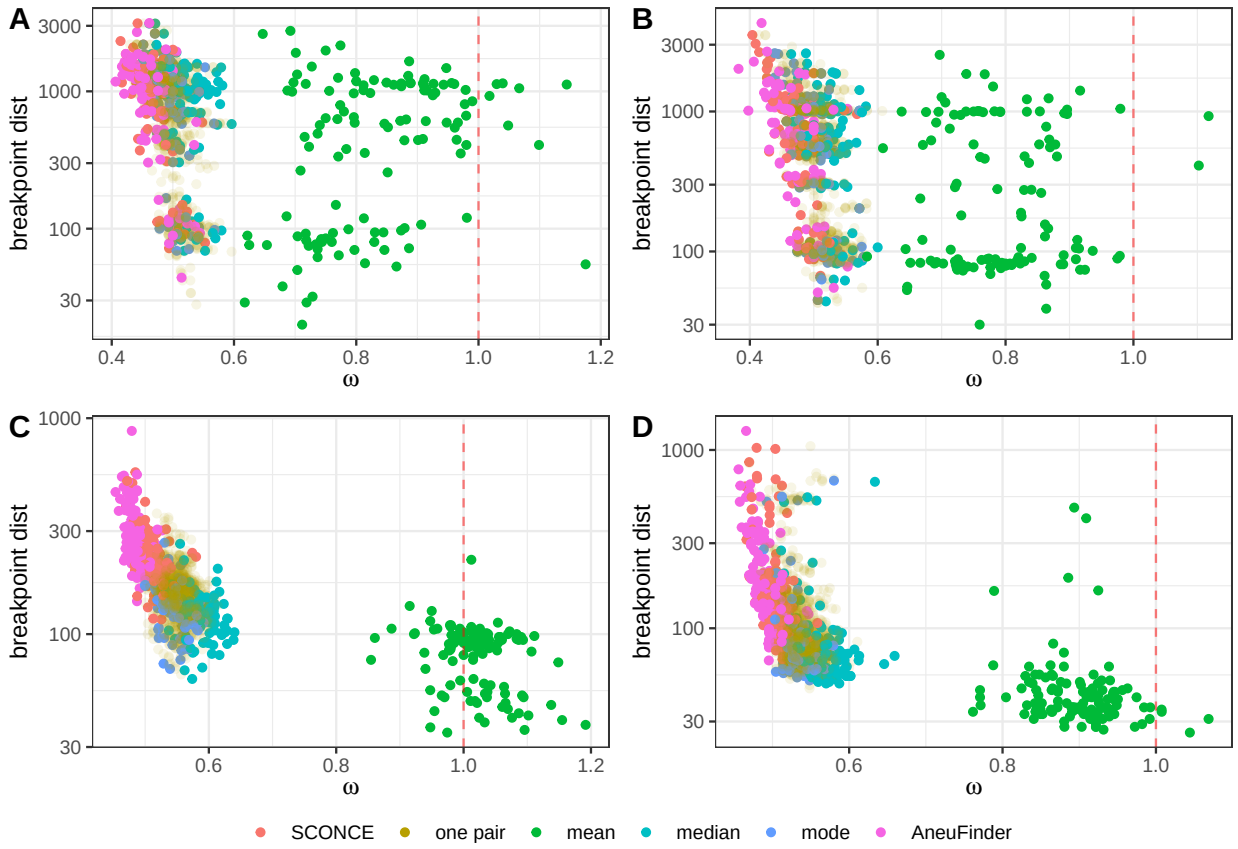


Figure S3: Breakpoint detection accuracy across parameter sets and CNP calling methods, where consensus CNPs are summarized over each cell's nearest 10 cells. Each dot represents one cell and each color represents on method, with $\omega = \frac{\#\text{ inferred breakpoints}}{\#\text{ true breakpoints}}$ on the x-axis and breakpoint distance on the y-axis. Using multiple cells consistently outperforms individual analyses.

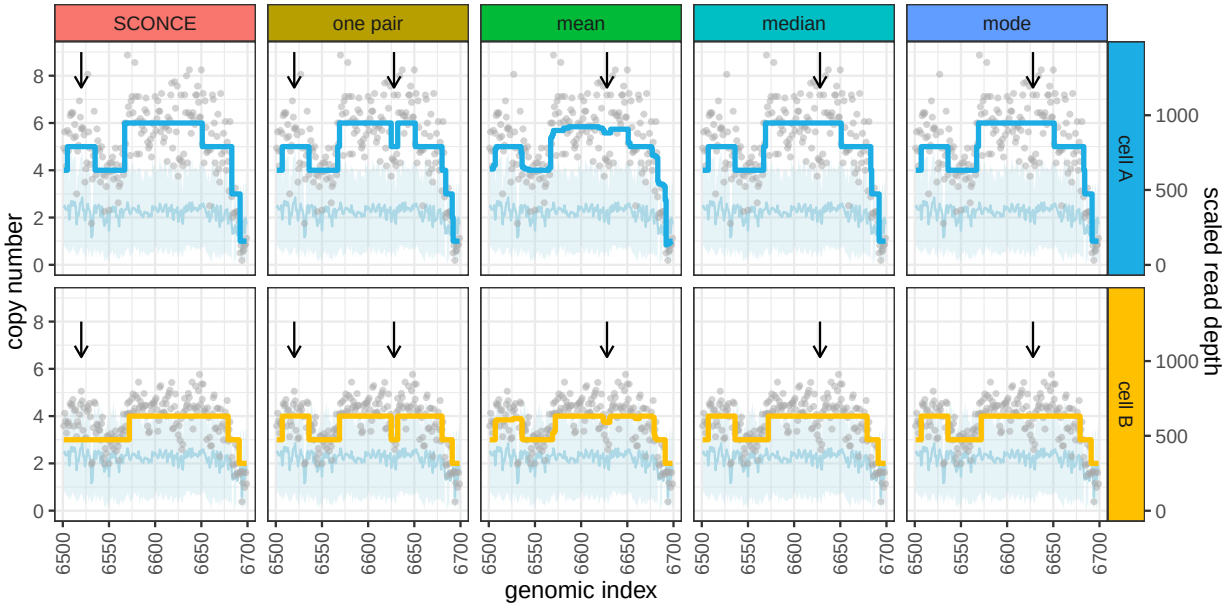


Figure S4: Improved CNA detection using multiple cells in real data [4]. Copy number calls and read depths are shown across methods for two cells (cell A: SRR054596, cell B: SRR054609, from [4]) for genomic windows 6500-6700 (x-axis). Genomic windows are 250kb across hg19 and numbered sequentially. Each dot shows scaled read depth (right y-axis, relative to diploid read depth) in one window, the light blue line and band show the mean and variance of the diploid read depth data, and colored lines show inferred copy number (left y-axis). Arrows denote differences in copy number calls between methods.

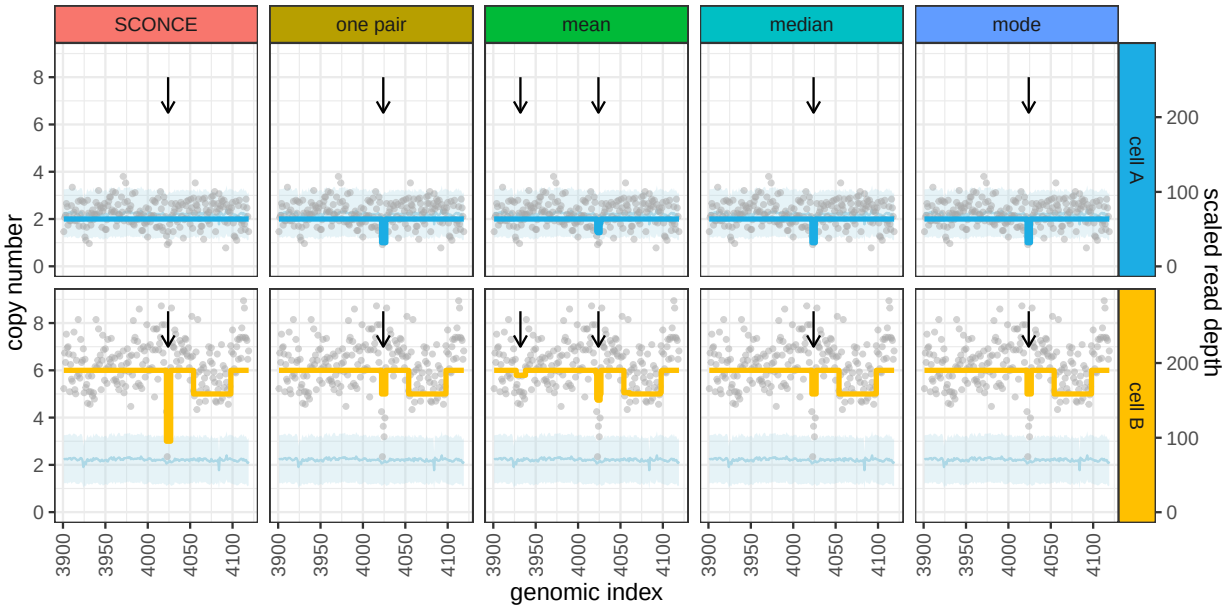


Figure S5: Improved CNA detection using multiple cells in real data [5]. Copy number calls and read depth are shown for two cells (cell A: cell 1538 with barcode TAGCCGGCAAGAACTA-1 from segment D, cell B: cell 1360 with barcode GCATACAAGTAACCCT-1. from segment B, from [5]) across methods in genomic windows 3900-4120 (x-axis). Windows are numbered sequentially across hg19 and are 250kb. Each dot shows per window read depth (right y-axis, scaled to the average diploid read depth), the light blue line and band show the mean and variance of the averaged diploid read depth data, and colored lines show inferred copy number (left y-axis). Arrows denote differences in copy number calls between methods.

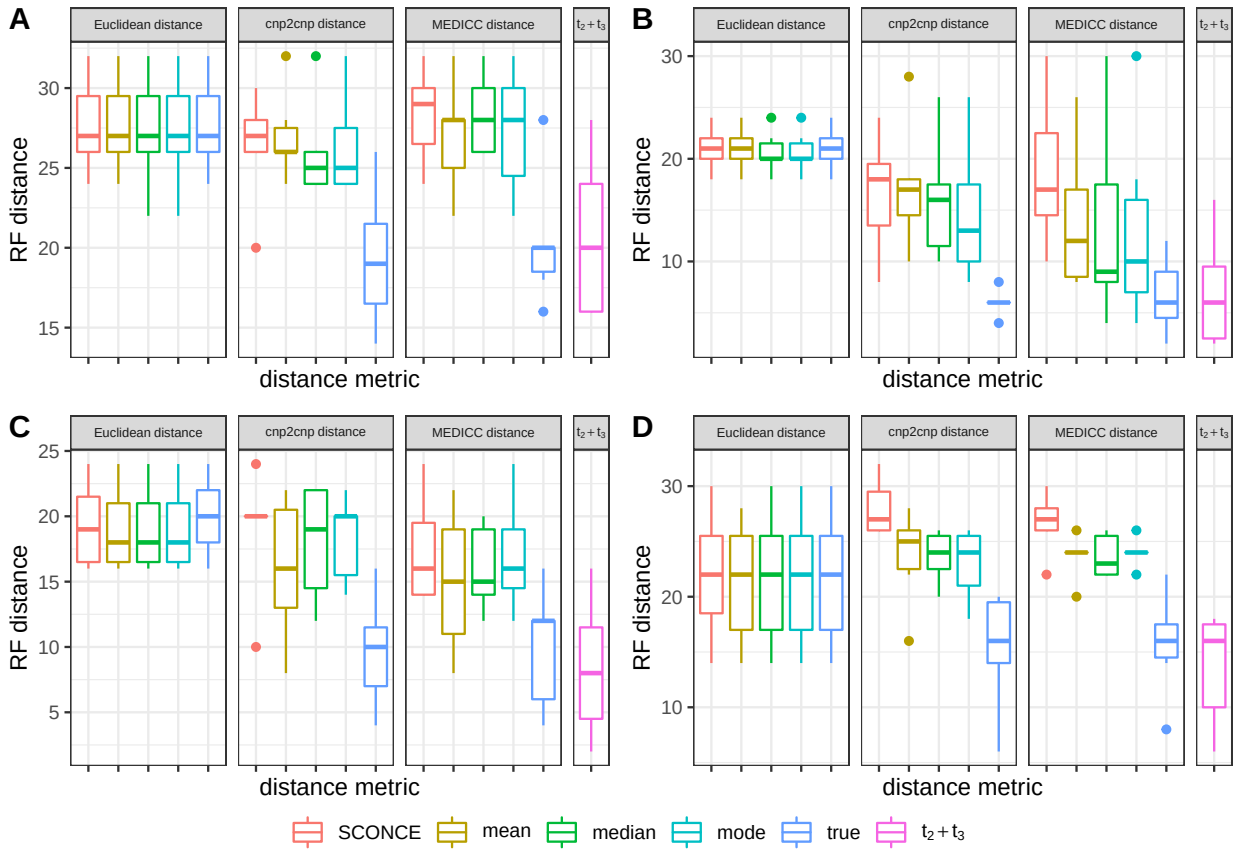


Figure S6: Robinson-Foulds (RF) distances for each simulation set (trees A-D) for different distance metrics. Distance metrics (Eucliden distance, cnp2cnp distance, MEDICC distance, $t_2 + t_3$) are grouped for each panel, and underlying CNPs sources are colored along the x-axis. Using $t_2 + t_3$ outperforms all other distance metrics on inferred copy number profiles.

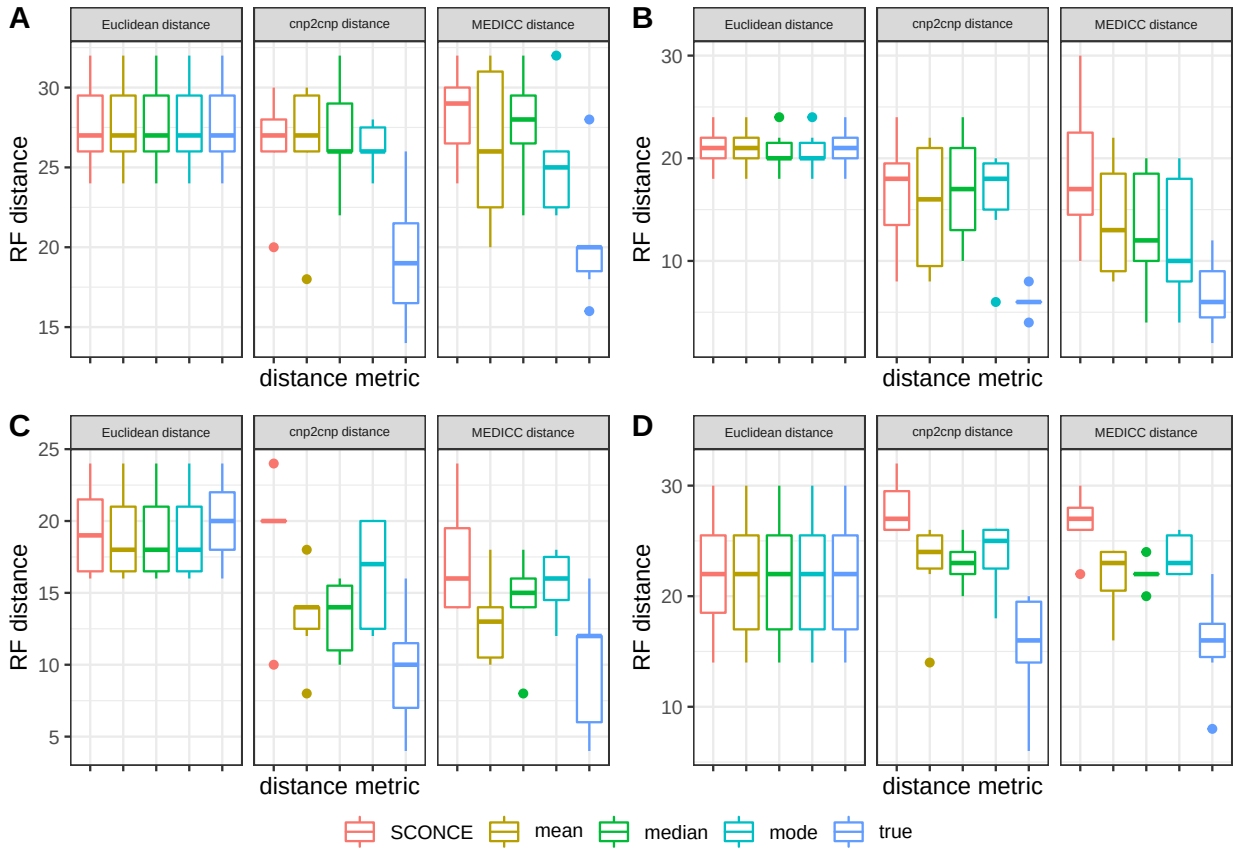


Figure S7: Robinson-Foulds distances for each simulation set (trees A-D) for different distance metrics, where consensus CNPs are summarized over each cell's 10 nearest neighbors. As in Supplementary Figure S6, distance metrics are grouped for each panel, and underlying CNPs types are colored along the x-axis. Note our $t_2 + t_3$ metric is excluded here; as a time saving measure, any pairs that aren't part of the nearest 10 of any cell (e.g., two very distant cells) are not analyzed, leading to missing data in the pairwise distance matrices.

Supplementary Tables

Simulation set/ tree	**Short description**	**Deletion rate, $\frac{\delta}{\tau_d}$**	**Insertion rate, $\frac{\varphi}{\tau_a}$**	**Mean deletion length, τ_d**	**Mean insertion length, τ_a**
A	maximally imbalanced, ultrametric	0.025	0.025	10	10
B	perfectly balanced, ultrametric	0.025	0.025	10	10
C	maximally imbalanced, not ultrametric, all branches have equal length	0.05	0.065	10.5	8
D	maximally imbalanced, not ultrametric, terminal branch lengths decay logarithmically	0.05	0.065	10.5	8

Table S1: Description of simulation sets under the line segment model, relative to a genome length of 100. A range of tree structures were chosen to test both typical and edge cases.

	SCONCE	**one pair**	**mean**	**median**	**mode**	**AneuFinder**
A	1167	1006	394	1018.5	1019.5	1172
B	1009.5	796.5	91.5	959	959	1016.5
C	206	153	84	142	142	263
D	153.5	85	33	77	77.5	168.5

Table S2: Full median breakpoint distance values for all trees and methods.

	SCONCE	**one pair**	**mean**	**median**	**mode**	**AneuFinder**
A	0.4663	0.49	0.921	0.4862	0.4856	0.4623
B	0.4762	0.5	0.8643	0.4941	0.4897	0.4744
C	0.5134	0.5507	1.1667	0.5467	0.5436	0.4851
D	0.504	0.5349	1.0072	0.5346	0.5338	0.4885

Table S3: Full $\omega = \frac{\text{\# inferred breakpoints}}{\text{\# true breakpoints}}$ values for all trees and methods.

	nearest/most similar	**10th nearest**	**furthest/least similar**
A	88.0625	138.9585	205.853
B	87.0804	135.2585	208.149
C	26.9814	84.32675	127.957
D	38.76815	72.9452	112.81

Table S4: Median pairwise Euclidean distances between SCONCE profiles for each dataset. Median distances between the nearest/most similar cells, 10th nearest cells, and furthest/least similar cell are shown.

	SCONCE	**one pair**	**mean**	**median**	**mode**	**AneuFinder**
A	1167	1014	591	1009	1027	1172
B	1009.5	858	121	958.5	959.5	1016.5
C	206	152	90	130.5	135	263
D	153.5	84	38	70	76	168.5

Table S5: Full median breakpoint distance values for all trees and methods, where consensus methods summarize over the nearest 10 cells only.

	SCONCE	**one pair**	**mean**	**median**	**mode**	**AneuFinder**
A	0.4663	0.4907	0.8186	0.5155	0.4929	0.4623
B	0.4762	0.5	0.7833	0.5191	0.4941	0.4744
C	0.5134	0.5493	1.02	0.5786	0.5462	0.4851
D	0.504	0.5344	0.9027	0.5639	0.5304	0.4885

Table S6: Full $\omega = \frac{\text{\# inferred breakpoints}}{\text{\# true breakpoints}}$ values for all trees and methods, where consensus methods summarize over the nearest 10 cells only.

	SCONCE	**mean**	**median**	**mode**	**true**	$\boldsymbol{t_2 + t_3}$
Euclidean distance	27	27	27	27	27	NA
cnp2cnp distance	27	26	25	25	19	NA
MEDICC distance	29	28	28	28	20	NA
$t_2 + t_3$	NA	NA	NA	NA	NA	20

(a) Tree A

	SCONCE	**mean**	**median**	**mode**	**true**	$\boldsymbol{t_2 + t_3}$
Euclidean distance	21	21	20	20	21	NA
cnp2cnp distance	18	17	16	13	6	NA
MEDICC distance	17	12	9	10	6	NA
$t_2 + t_3$	NA	NA	NA	NA	NA	6

(b) Tree B

	SCONCE	**mean**	**median**	**mode**	**true**	$\boldsymbol{t_2 + t_3}$
Euclidean distance	19	18	18	18	20	NA
cnp2cnp distance	20	16	19	20	10	NA
MEDICC distance	16	15	15	16	12	NA
$t_2 + t_3$	NA	NA	NA	NA	NA	8

(c) Tree C

	SCONCE	**mean**	**median**	**mode**	**true**	$\boldsymbol{t_2 + t_3}$
Euclidean distance	22	22	22	22	22	NA
cnp2cnp distance	27	25	24	24	16	NA
MEDICC distance	27	24	23	24	16	NA
$t_2 + t_3$	NA	NA	NA	NA	NA	16

(d) Tree D

Table S7: Median Robinson-Foulds distances across different distance metrics on different CNPs for each simulation set.

	SCONCE	**mean**	**median**	**mode**	**true**
Euclidean distance	27	27	27	27	27
cnp2cnp distance	27	27	26	26	19
MEDICC distance	29	26	28	25	20

(a) Tree A

	SCONCE	**mean**	**median**	**mode**	**true**
Euclidean distance	21	21	20	20	21
cnp2cnp distance	18	16	17	18	6
MEDICC distance	17	13	12	10	6

(b) Tree B

	SCONCE	**mean**	**median**	**mode**	**true**
Euclidean distance	19	18	18	18	20
cnp2cnp distance	20	14	14	17	10
MEDICC distance	16	13	15	16	12

(c) Tree C

	SCONCE	**mean**	**median**	**mode**	**true**
Euclidean distance	22	22	22	22	22
cnp2cnp distance	27	24	23	25	16
MEDICC distance	27	23	22	23	16

(d) Tree D

Table S8: Median Robinson-Foulds distances across different distance metrics on different CNPs for each simulation set, where consensus CNPs are summarized over only the nearest 10 cells. Note that because summarizing over only the nearest 10 cells does not guarantee all pairs of cells will be jointly analyzed, no $t_2 + t_3$ values are given here, as estimating phylogenies using $t_2 + t_3$ requires a complete distance matrix.

Chapter 4

SCONCEmut: joint copy number profile and evolutionary distance inference with point mutations

4.1 Abstract

Using single cell whole genome sequencing, new insights can be gained into the tumor evolutionary process by estimating intercellular evolutionary relationships. These relationships can be estimated by analyzing shared and cell specific somatic point mutations and large copy number alterations, and single cell sequencing can help distinguish cell specific events from shared ones. However, single cell sequencing results in low and uneven read coverage across the genome, making point mutation calling difficult. Although many methods exist for single cell copy number calling and phylogeny estimation, most do not consider point mutation data. Here, we explore an expanded model rooted in previous work on copy number calling and phylogeny estimation in single cell tumor sequencing, called SCONCEmut. SCONCEmut infers copy number profiles and cell evolutionary distances, by jointly estimating counts of point mutations and pairwise branch lengths. We benchmark SCONCEmut on simulated data, and show its application in a previously published breast cancer dataset.

4.2 Introduction

Cancer arises from an accumulation of somatic copy number alterations (CNAs) and point mutations. Identifying these somatic mutations and how they affect tumor progression and evolution are key goals of studying cancer genomics. One common approach to studying tumor evolution is to estimate tumor phylogenies using these somatic mutation data.

Large pan-cancer studies of whole genome bulk tumor sequencing have yielded massive amounts of data about thousands of tumors, characterized mutation signatures, and identi-

fied recurring mutations [22]. Numerous methods have been developed to analyze this type of bulk data to estimate tumor phylogenies, using a mixture of point mutations and copy number alterations [102–108]. However, due to the averaging effect of bulk sequencing, rare mutations, cell specific information, and direct connections between point mutations and copy number states are lost. For example, a mutation with an allele frequency of 0.25 might be due to a homozygous mutation in a quarter of the sequenced cells, or due to a mutation on one allele, in a region with copy number 4, in all cells.

In contrast, because single cell sequencing maintains cell individuality, copy number and point mutation relationships are retained. Furthermore, well established population genetics methods for estimating phylogenies, such as neighbor-joining [31, 32], can be applied by treating single cells as individuals in a population.

However, addressing uncertainty and error in single cell sequencing remains an active area of research. One of the main challenges of using single cell whole genome sequencing is the unsolved issue of calling somatic mutations using sparse data. That is, the extremely low amount of starting DNA and subsequent low and uneven sequencing depth, approximately 0.1-0.5x mean read depth [109, 110], for each position makes accurate variant calling difficult. Additionally, the necessary genome amplification steps during library preparation can introduce their own biases and errors [109, 110]. For example, if an error occurs in an early PCR cycle during library preparation (such as a point mutation or allelic dropout at a heterozygous site), this error will be exponentially propagated.

One approach to incorporate uncertainty in point mutation detection is to calculate genotype likelihoods [111, 112], rather than attempting to call variants. This allows extraction of useful information about somatic mutations and their allele frequencies, while accounting for low read depth and sequencing error, and does not simply relying on observing a sufficiently high quantity of high quality reads at each genomic site. However, with limited read depth, reliably differentiating between true variants and sequencing error remains an unsolved problem.

Nevertheless, many methods exist to estimate tumor phylogenies from single cell sequencing, by inferring CNAs alone [6, 74, 75, 81, 113], by inferring point mutations alone [114–119], or by inferring point mutations with orthogonally determined copy number profiles [84, 120, 121]. However, no peer reviewed methods exist to jointly infer CNAs and point mutations to estimate phylogenies in single cell sequencing. To our knowledge, only COMPASS [35], which is designed for amplicon data instead of whole genome sequencing, and $BiTSC^2$ [122], both preprints, use copy number and point mutations jointly to estimate phylogenies.

Instead of taking called genotypes for input, COMPASS [35] directly analyzes read counts. However, because COMPASS is designed for amplicon data from the Tapestri platform with recommended 60-80x coverage [123, 124], it doesn't need to contend with the same low coverage issues as single cell whole genome sequencing. Additionally, COMPASS first estimates phylogeneies with point mutations only, then searches for nodes with differences in coverage to identify CNAs and refine the tree estimate. Although this allows COMPASS to detect subclonal CNAs, it also limits CNA detection to sites with point mutation support.

In contrast, $BiTSC^2$ [122] is designed specifically for single cell whole genome sequencing,

and takes in matrices of total and derived/somatic allele read counts, instead of genotype calls. $BiTSC^2$ uses a Zero Inflated Poisson distribution to model low read depth, alleleic dropout, and sequencing error. To detect CNAs, $BiTSC^2$ takes in genomic segments (ie, regions likely to have the same copy number), pre-called by other methods, to define CNA boundaries, and uses point mutations to support CNA calls. If these segments are unknown, $BiTSC^2$ can also label every locus or bin as its own bin, but this results in lower accuracy, as $BiTSC^2$ cannot infer copy number segments on its own. Furthermore, because $BiTSC^2$ relies on point mutations to detect CNAs, it does not detect CNAs in segments with low point mutation counts.

Here, we explore a novel method, SCONCEmut, that expands on our previous work [3, 6] by incorporating genotype likelihoods of point mutations into tree branch estimation, based on a stochastic model of tumor evolution. In this process, SCONCEmut accounts for sequencing error and models low read depth using a Negative Binomial distribution for CNAs and a BetaBinomal distribution for point mutations. SCONCEmut first estimates pairwise point mutation counts, then infers accurate copy number profiles across multiple cells and estimates tree branch lengths. However, the tree branch length estimates, useful for phylogeny estimation, from SCONCEmut do not present an improvement on previous work [6]. Here, we explore SCONCEmut's performance on simulated and previously published data [4], consider sources of modeling error and bias, and suggest future directions and improvements.

4.3 Methods

Modeling the tumor evolutionary process using CNAs

In previous work [3, 6], we introduced a Markov process that approximates the stochastic tumor evolutionary process by incorporating two processes: one that is continuous through evolutionary time, and one that is discrete along the length of the genome. We briefly review this model here, before presenting the novel incorporation of genotype likelihoods.

Continuous Time Evolutionary Model

First, we model how the copy numbers, U and V, in two adjacent genomic bins change over time using a continuous time process. Specifically, for bins i and $i+1$, the copy number changes from (U, V) to (U', V') according to the following rate parameters:

$$\alpha = \text{rate of } \pm 1 \text{ CNA} \tag{1a}$$

$$\beta = \text{rate of any CNA} \tag{1b}$$

$$\gamma = \text{relative rate of CNAs affecting both } U \text{ and } V \tag{1c}$$

These rates are encoded in instantaneous rate transition matrix $\mathbb{Q} = \{q_{(U,V),(U',V')}\}$:

$$q_{(U,V),(U',V')} = \begin{cases} \gamma(\alpha+\beta) & \text{if } (U',V') = \begin{cases} (U+n, V+n) \\ (U-n, V-n) \end{cases} , n = 1 \\ \gamma\beta & \text{if } (U',V') = \begin{cases} (U+n, V+n) \\ (U-n, V-n) \end{cases} , n > 1 \\ \alpha+\beta & \text{if } (U',V') = \begin{cases} (U \pm n, V) \\ (U, V \pm n) \end{cases} , n = 1 \\ \beta & \text{if } (U',V') = \begin{cases} (U \pm n, V) \\ (U, V \pm n) \end{cases} , n > 1 \\ r_{(U,V)} & \text{if } (U',V') = (U,V) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Diagonal elements of $\mathbb{Q}$ are set to the row sums, $r_{(U,V)} = -\sum_{(u',v') \neq (U,V)} q_{(U,V),(u',v')}$, such that each row sums to 0. We define elements of $\mathbb{Q}$ as instantaneous rates, such that two separate events, such as $(U',V') = (U-n, V+k), n > 0, k > 0$, can only occur for time intervals $t > 0$.

The time dependent transition matrix $\mathbb{P}(t) = \{p_{(U,V),(U',V')}(t)\}$, representing the probability of the copy number in two bins evolving from (U,V) to (U',V') in evolutionary time t, is calculated using the matrix exponential:

$$P_{(U,V),(U',V')}(t) = e^{\mathbb{Q}t} \tag{3}$$

Discrete Process Approximation

Next, we convert this continuous-in-evolutionary-time model to the approximating discrete process along the length of the genome. We model the copy number of two cells, A and B, at window i, as CN_{iA} and CN_{iB}, respectively, where $CN_{iA}, CN_{iB} \in \mathbb{S}_c = \{0, 1, 2, ..., k\}$ for a constant maximum copy number k. The state space of this discrete process therefore consists of pairs of copy numbers:

$$\mathbb{S}_d = \mathbb{S}_c \times \mathbb{S}_c = \{(0,0), (0,1), (0,2), \ldots, (k,k)\} \tag{4}$$

We then incorporate the tree $\mathcal{T} = [t_1, t_2, t_3]$ shown in Figure 1. Specifically, the copy numbers in bins with indices i and $i+1$ evolve together from an ancestral diploid cell, D, for time t_1, until divergence at an unknown intermediate state $Z : (W, Y)$. From Z, cell A evolves for time t_2 to state $(CN_{iA}, CN_{i+1,A})$, and cell B evolves for time t_3 to state $(CN_{iB}, CN_{i+1,B})$. Because state Z is unobserved, we sum over all values of $W, Y \in \mathbb{S}_c$. The probability of

observing this evolution is defined in matrix $\mathbb{F}(\mathcal{T}) = \{f_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}(\mathcal{T})\}$:

$$f_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}(\mathcal{T}) = \sum_{W,Y \in \mathbb{S}_c} \Big(P_{(2,2),(W,Y)}(t_1) \times P_{(W,Y),(CN_{iA},CN_{i+1,A})}(t_2) \times P_{(W,Y),(CN_{iB},CN_{i+1,B})}(t_3) \Big) \tag{5}$$

By dividing by the corresponding row sums, matrix $\mathbb{F}(\mathcal{T})$ can be normalized into the transition matrix $\mathbb{M}(\mathcal{T}) = \{m_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}(\mathcal{T})\}$:

$$m_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}(\mathcal{T}) = \frac{f_{(CN_{iA},CN_{iB}),(CN_{i+1,A},CN_{i+1,B})}(\mathcal{T})}{\sum_{(c,d)\in\mathbb{S}_d} f_{(CN_{iA},CN_{iB}),(c,d)}(\mathcal{T})} \tag{6}$$

This matrix, $\mathbb{M}(\mathcal{T})$, describes the probability of observing a transition between state (CN_{iA}, CN_{iB}) to state $(CN_{i+1,A}, CN_{i+1,B})$ along the genome, given evolutionary tree $\mathcal{T}$. Thus, we have derived a Markov process that approximates the evolutionary process through time and across the length of the genome (previously described in [3, 6]).

Two Cell Hidden Markov Model

We return to the Hidden Markov Model (HMM) [77–79] introduced in [6]. In this HMM, pairs of read depths and copy number states ($\mathbb{S}_d$, defined in Equation 4) make up the alphabet and state space, respectively, for any cell pair (A, B). The transition matrix is given by $\mathbb{M}(\mathcal{T})$ (defined in Equation 6).

Assuming conditional independence between cells A and B, the emission probabilities for this HMM are defined as:

$$\mathbb{P}(X_{iA} = x_{iA}, X_{iB} = x_{iB} | CN_{iA}, CN_{iB}) = \mathbb{P}(X_{iA} = x_{iA} | CN_{iA}) \mathbb{P}(X_{iB} = x_{iB} | CN_{iB}) \tag{7}$$

As the emission probability derivation is identical for cells A and B, we derive the emission probabilities for generic cell j only. We define random variable X_{ij} to model the observed number of reads falling in window i for cell j. We assume X_{ij} follows a Negative Binomial distribution:

$$\mathbb{E}(X_{ij}) = \lambda_{ij} = \left(CN_{ij} \times \frac{\mu_i}{2} \right) \times s_j + \varepsilon \tag{8}$$

$$X_{ij} \sim NegBinom\left(\lambda_{ij}, \sigma^2_{ij} = a\lambda^2_{ij} + b\lambda_{ij} + c\right) \tag{9}$$

where

$$CN_{ij} = \text{the copy number in window } i \text{ for cell } j \tag{10a}$$

$$\mu_i = \text{the mean diploid read depth in window } i \tag{10b}$$

$$\varepsilon = \text{constant sequencing error term} \tag{10c}$$

$$s_A = \text{library size scaling factor for cell } j \tag{10d}$$

$$\{a, b, c\} = \text{constants learned from diploid data} \tag{10e}$$

Note, these emission probabilities were previously described in [3, 6], and estimation of constants $\{a, b, c\}$ is described in SCONCE and SCONCE2 review. Recall these methods require matched diploid single cells, in order to use averaged diploid read depth as a null model for sequencing errors and GC and mappability biases.

Branch Length Estimation Using Somatic Point Mutations

In [3, 6], we estimated branch lengths, shown in Figure 1, by optimizing the forward loglikelihood of the observed copy number data, $\ell_{CNA}(\mathcal{T})$, where $\mathcal{T} = [t_1, t_2, t_3]$ is a tree consisting of one pair of cells. This results in branch length estimates based on paired copy number profiles. Here, in order to add point mutations, we instead optimize

$$\ell(\mathcal{T}) = \ell_{CNA}(\mathcal{T}) + \ell_{mut}(\mathcal{T}) \tag{11}$$

where $\ell_{mut}(\mathcal{T})$ is the mutation loglikelihood of tree $\mathcal{T}$ (see Estimating branch lengths from mutation counts), using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [125]. After this optimization, the Viterbi decoding is used to infer pairwise copy number profiles, which are then summarized into cell specific consensus profiles using the mean, median, or mode (following [6]). An overview of this pipeline is shown in Supplementary Figure S1, with further details in Pipeline details.

Model assumptions

In order to define $\ell_{mut}(\mathcal{T})$, we make a number of simplifying assumptions. We use the infinite sites model, such that mutations can only arise once and are inherited by daughter cells, and assume mutations occur according to a constant Poisson process. Additionally, we assume that given a copy number state for position i, CN_i, all somatic allele frequencies are equally likely with probability $\frac{1}{CN_i}$, and that given an alleleic state, cell specific read counts are conditionally independent. Furthermore, we assume any unobserved sites contain the ancestral/germline allele. Finally, we assume constant sequencing error rates and biallelic variable sites. Limitations of these assumptions are considered in the Discussion.

Estimating branch lengths from mutation counts

Using the tree structure defined in Figure 1, we define three types of pairwise somatic mutations for cell pair (A, B). We define $S_i = (I_{iA}, I_{iB})$ as the mutation status at site i, where I_{iA}, I_{iB} are indicator functions denoting presence or absence of a somatic mutation in cells A and B, respectively:

$$S_i \in \mathbb{S}_m = \begin{cases} (1,1) & \text{somatic mutation occurred on } t_1\text{, observed in cell } A \text{ and } B \\ (1,0) & \text{somatic mutation occurred on } t_2\text{, observed in cell } A \text{ only} \\ (0,1) & \text{somatic mutation occurred on } t_3\text{, observed in cell } B \text{ only} \\ (0,0) & \text{germline SNP} \end{cases} \tag{12}$$

We define random variable X_b as the count of mutations on branch b, for $b \in \{1, 2, 3\}$, such that the expected number of mutations is proportional to the branch length:

$$\mathbb{E}(X_b) = \varphi t_b \tag{13}$$

$$\mathbb{P}(X_b = x; \varphi t_b) = \frac{(\varphi t_b)^x e^{-\varphi t_b}}{x!} \tag{14}$$

where φ is a scaling factor that relates the rate of CNA events and point mutation events (see Jointly estimating Beta Binomial Parameters).

Using Equation 14, we define the loglikelihood of tree $\mathcal{T}$ from point mutations, $\ell_{mut}(\mathcal{T})$, as the sum of the loglikelihoods of the mutation counts on each branch:

$$\ell_{mut}(\mathcal{T}) = \sum_{b=1}^{3} \log\left(\mathbb{P}(X_b = x_b; \varphi t_b)\right) \tag{15}$$

Estimating mutation counts from observed read data

In order to estimate the mutation count, X_b, on branch b, we first define random variable $D_{ij} = (a_{ij}, n_{ij})$ as an ordered pair representing the collection of read data at genomic position i for cell j, where a_{ij} is the number of reads mapping to the derived/somatic allele, and n_{ij} is the total number of reads mapped to this site. We define random variable $D_i = (D_{iA}, D_{iB})$ to represent all observed read count data for cells A and B at site i, such that $\mathbb{P}(D_i|S_i = (I_{iA}, I_{iB}))$ gives the probability of the observed data, for site i in cells A and B, given the underlying mutation status, S_i.

To calculate the total loglikelihood of mutation counts $\{X_1, X_2, X_3\}_{AB}$, we sum the following across all N_{AB} sites observed in cells A and B:

$$\begin{aligned}\ell(\{X_1 = x_1, X_2 = x_2, X_3 = x_3\}_{AB}) = \sum_{i=1}^{N_{AB}} \log \Bigg[&\mathbb{P}(D_i|S_i = (1,1))\frac{x_1}{N_{AB}} + \\ &\mathbb{P}(D_i|S_i = (1,0))\frac{x_2}{N_{AB}} + \\ &\mathbb{P}(D_i|S_i = (0,1))\frac{x_3}{N_{AB}} + \\ &\mathbb{P}(D_i|S_i = (0,0))\frac{N_{AB} - x_1 - x_2 - x_3}{N_{AB}}\Bigg]\end{aligned} \tag{16}$$

such that each $\mathbb{P}(D_i|S_i = (I_{iA}, I_{iB}))$ term is scaled by the proportion of mutations of that type/on the corresponding branch.

Likelihood of observed read data, given mutation status S_i

In order to calculate the likelihood of the observed read data given the underlying mutation status S_i, $\mathbb{P}(D_i|S_i = (I_{iA}, I_{iB}))$, we assume conditional independence between cells, such that

the probability of the observed read data D_i for site i is

$$\mathbb{P}(D_i = (D_{iA}, D_{iB})|S_i = (I_{iA}, I_{iB})) = \mathbb{P}(D_{iA}|I_{iA})\mathbb{P}(D_{iB}|I_{iB}) \tag{17}$$

where $\mathbb{P}(D_{iA}|I_{iA})$ and $\mathbb{P}(D_{iB}|I_{iB})$ are the probabilities of the observed read data at site i for cells A and B, respectively. As $\mathbb{P}(D_{iA}|I_{iA})$ and $\mathbb{P}(D_{iB}|I_{iB})$ are equivalent, we derive $\mathbb{P}(D_{ij}|I_{ij})$ for generic cell j going forward.

The probability of the observed read data at site i, $\mathbb{P}(D_{ij}|I_{ij})$, depends on the allele frequency, f, of the derived/somatic allele, calculated from the number of somatic alleles and total number of alleles (ie, the copy number state). The true value of an allele frequency is $f = \frac{s}{CN} = \frac{s}{s+g}$, where s and g are the number of somatic and germline alleles, respectively, at any given position. However, this true allele frequency is unknown. To model this, we define $\mathbb{P}(D_{ij}|I_{ij})$ as

$$\mathbb{P}(D_{ij}|I_{ij} = 1) = \sum_{0 \leq g < CN_{ij}} \mathbb{P}(D_{ij}|f, \omega, g, CN_{ij})\mathbb{P}(g|CN_{ij}) \tag{18}$$

$$\mathbb{P}(D_{ij}|I_{ij} = 0) = \mathbb{P}(D_{ij}|f, \omega, g = CN_{ij}, CN_{ij}) \tag{19}$$

where

$$g = \text{true ancestral/germline allele count} \tag{20a}$$

$$s = \text{true derived/somatic allele count} = CN_{ij} - g \tag{20b}$$

$$f = \frac{s}{g+s}(1-\varepsilon) + \frac{g}{g+s}\varepsilon \tag{20c}$$

$$\omega = \text{overdispersion parameter} \tag{20d}$$

$$\varepsilon = \text{sequencing error term} \tag{20e}$$

$$CN_{ij} = \text{inferred copy number state at site } i \text{ for cell } j \tag{20f}$$

$$\mathbb{P}(g|CN_{ij}) = \frac{1}{CN_{ij}} \tag{20g}$$

This allows us to account for uncertainty in somatic allele frequencies, f, given the inferred copy number state, by considering all possible values of f.

In order to calculate $\mathbb{P}(D_{ij}|f, \omega, \ell, CN_{ij})$, recall D_{ij} is defined as the ordered pair (a_{ij}, n_{ij}), representing the number of reads mapping to the derived allele, a_{ij}, and the total number of reads, n_{ij}, at site i for cell j. Similar to the formulation presented in [126], D_{ij} follows a BetaBinomial distribution:

$$D_{ij} \sim BetaBinomial(n_{ij}, \alpha = f\omega, \beta = (1-f)\omega) \tag{21}$$

such that

$$\mathbb{P}(D_{ij} = (a_{ij}, n_{ij})|f, \omega, g, CN_{ij}) = \binom{n_{ij}}{a_{ij}} \frac{B(a_{ij} + \alpha, n_{ij} - a_{ij} + \beta)}{B(\alpha, \beta)} \tag{22a}$$

$$= \binom{n_{ij}}{a_{ij}} \frac{B(a_{ij} + f\omega, n_{ij} - a_{ij} + (1-f)\omega)}{B(f\omega, (1-f)\omega)} \tag{22b}$$

where $B(x, y)$ is the beta function and f is calculated from g and CN_{ij} (see Equation 20c). By design, this formulation results in the expected value of D_{ij} as:

$$\mathbb{E}(D_{ij}) = \frac{n_{ij}\alpha}{\alpha + \beta} \tag{23a}$$

$$= \frac{n_{ij} f\omega}{f\omega + (1 - f)\omega} \tag{23b}$$

$$= n_{ij} f \tag{23c}$$

that is, the product of the total number of reads and the allele frequency at a given position. This allows us to marginalize out all possible somatic allele frequencies, f, for an inferred copy number state, CN_{ij}, to calculate the probability of read data D_{ij}.

Note, we assume all allele frequencies have equal probability and weight them accordingly in Equation 20g, which may erroneously give too much weight to high allele frequencies. Additionally, due to low read depth, few sites are observed in both cells A and B. Therefore, if site i is observed in cell A but not in cell B, we assume cell B matches the ancestral allele, and set $\mathbb{P}(D_{iB}|I_{iB})$ to 1, such that the data from site i may still be utilized for cell A (and vice versa). This assumption can bias our estimates towards longer t_2 and t_3 branch lengths (see the Discussion for further analysis).

Simulations

To rigorously test SCONCEmut, we applied it to 7 simulated datasets and a published dataset from [4]. Briefly, we expanded the line segment simulation model previously presented in [3, 6] to add point mutations. Mutations follow an infinite sites model, that is, any given mutation occurs only once on one copy/allele. Mutations occur along the tree structure, such that mutations are propagated to all descendent cells. Copy number events can change the allele frequency of a mutation, but there are no other mechanisms for allele frequency changes.

For every mutation, the simulation model outputs genomic coordinate, the count of each allele (ancestral/germline and derived/somatic), and the number of reads aligning to each allele. For each genomic bin, the simulation model outputs the genomic coordinate range, the true copy number state, and the number of reads falling into that bin.

We simulated read depth and point mutation data for 4 distinct tree structures, representing the extremes of possible tree structures and different cancer evolutionary models [26]. Specifically, tree A is fully pectinate/maximally imbalanced and ultrametric, tree B is perfectly balanced and ultrametric, tree C is maximally imbalanced and not ultrametric (uniform branch lengths), and tree D is maximally imbalanced and not ultrametric (logarithmically decaying external branch lengths). Trees E, G, and F have the same structure as tree A (maximally imbalanced and ultrametric), and vary in overall read depth. We simulated 128 tumor cells and 100 matched diploid cells for each tree, and test sets of 20 cells were sampled from each tree. Full simulation details are given in Simulations with Mutations, and simulation parameters are shown in Supplementary Table S1.

Detecting Point Mutations in Real Data

One of the main challenges of identifying point mutations in single cell sequencing is the low read depth, making it difficult to separate amplification and sequencing errors from true variants. In order to identify a set of high confidence somatic point mutations from [4], we used the diploid cells as a null model to estimate sequencing error.

First, we identified sites likely to be sequencing error in diploid cells, based on loglikelihood ratios for variability and dbSNP membership, following [127]. Then, for both diploid and tumor cells, we applied a number of filters based on number of cells a mutation appeared in and average mutation coverage. Specifically, we removed any mutations that were observed in fewer than 5 cells or had fewer than 3.5 reads on average per observation. We note this filtering approach biases our mutation set towards variants shared between cells (that is, more likely to be on branch t_1).

Next, we used a BetaBinomial distribution to estimate ω, the overdispersion parameter, on diploid sites likely to be sequencing errors. We then used this estimate, $\hat{\omega}$, to perform loglikelihood ratio tests for variability in tumor cells (relative to the diploid cells), such that sites likely to be variable were labeled somatic mutations (see Identifying point mutations in real data for full details).

4.4 Results

Copy number and breakpoint detection accuracy

In order to evaluate the copy number calling accuracy of SCONCEmut, we calculated the sum of squared errors (SSE) across the genome between the simulated and inferred copy numbers for each cell. Across simulated datasets, compared to SCONCE2, adding mutations did not significantly change the median SSE values. In tree A (maximally imbalanced and ultrametric), the median SSE for SCONCE2 was 18.96, 21.74, and 21.74, for mean, median, and mode, respectively, and 18.96, 21.98, and 22.02 for SCONCEmut (Figure 2A). For tree D (maximally imbalanced and not ultrametric, with logarithmically decaying external branch lengths), in the same order, SCONCE2 had median SSE values of 60.00, 69.50, and 72.97, while SCONCEmut had median SSE values of 59.04, 68.12, and 69.76 (Figure 2D). This result is not surprising, as the mutation counts do not directly affect the copy number profiles. As in [6], using the mean consensus profile results in the biggest decrease in SSE from SCONCE, compared to using the median or mode.

To evaluate breakpoint detection accuracy, we calculated the total breakpoint distance by summing the distance to the nearest inferred breakpoint for each simulated breakpoint, genome wide. Additionally, to penalize erroneously inferring an excess of breakpoints to bring down total breakpoint distance, we calculated $\omega = \frac{\#\text{ inferred breakpoints}}{\#\text{ true breakpoints}}$, where ω values closest to 1 indicate greatest accuracy. Similarly to SSE, compared to SCONCE2, the total breakpoint distances did not change significantly for SCONCEmut, and the majority of

differences were due to relatively small changes in ω values. Differences between SCONCE2 and SCONCEmut breakpoint distance and ω values are shown in Figure 3, where lines connect cell specific results from SCONCE2 (circles) and SCONCEmut (triangles). In tree D (maximally imbalanced and not ultrametric, with logarithmically decaying external branch lengths), the median breakpoint distances were 102, 194.5, and 196 for mean, median, and mode for SCONCE2, and 103, 198, and 199.5 for SCONCEmut (Figure 3D). In the same order, the median ω values for SCONCE2 were 1.0108, 0.5404, and 0.5361, and were 0.973, 0.5458, and 0.5378 for SCONCEmut for tree D. Full median breakpoint distances and ω values are given in Supplementary Tables S2 and S3.

Recovery of mutation counts

In addition to estimating branch lengths, SCONCEmut also estimates the number of mutations on each branch in $\mathcal{T} = [t_1, t_2, t_3]$. In Figure 4, we show the R^2 values between observed and inferred number of mutations on each branch, for all cell pairs. In tree A (maximally imbalanced and ultrametric), SCONCEmut estimates the number of mutations on branches t_1, t_2, and t_3 with R^2 values of 0.689, 0.754, and 0.738, respectively (Figure 4A). Similarly, in tree C (maximally imbalanced and not ultrametric, with uniform branch lengths), R^2 values were 0.769, 0.939, and 0.912 for branches t_1, t_2, and t_3 (Figure 4C).

However, estimations of mutation counts are not consistently correlated with branch length. In Figure 5, we show the R^2 values between number of mutations and branch length, where the number of mutations is split into three categories: the true number of mutations simulated for each branch (determined by presence/absence of derived alleles in one or both cells), the number of observed mutations (determined by presence/absence of reads mapping to the derived allele in one or both cells), and the number of inferred mutations (estimated from SCONCEmut). In tree A (maximally imbalanced and ultrametric), although the number of inferred mutations was positively correlated with the number of observed mutations (Figure 4A), the number of inferred mutations was negatively correlated with branch length (Figure 5A, row 3). Because the number of observed mutations is also negatively correlated with branch length (Figure 5A, row 2) while the true number of mutations was positively correlated with branch length (Figure 5A, row 1), this suggests the mutation count inference is limited by low read depth (that is, many mutations are not observed).

This is further supported by comparing the true number of point mutations and number of observed mutations. In the left most column of Figure 6, we show the R^2 values between the number of observed mutations and the true number of mutations. Tree A (maximally imbalanced and ultrametric) has R^2 values of 0.319, 0.0006, and 0.0011 for branches t_1, t_2, and t_3 (Figure 6A, left column). In the center column, we compare the number of inferred mutations to the true number of mutations. In the same order, tree A has R^2 values of 0.19, 2.55e-5, and 4.24e-6 (Figure 6A, center column). For ease of comparison, the right column of Figure 6 shows the correlation between the observed and inferred numbers of mutations from Figure 4. Across parameter sets, the R^2 values between the inferred and observed numbers

of mutations (Figure 6, right column) was higher than the R^2 values with the true number of mutations (Figure 6, center column).

Tree branch length estimation

Similarly to SCONCE2, SCONCEmut estimates pairwise tree branch lengths, $\mathcal{T} = [t_1, t_2, t_3]$, for all cells. In order to evaluate tree branch length recovery, we show the R^2 values between true branch lengths and inferred branch lengths, for all cell pairs in Figure 7. Using mutations (ie, SCONCEmut), the R^2 values for t_1, t_2, t_3, and $t_2 + t_3$ in tree C (maximally imbalanced and not ultrametric, with uniform branch lengths) were 0.776, 0.656, 0.597, and 0.178 (Figure 7C). In contrast, the R^2 values for the same branches and tree were 0.785, 0.812, 0.841, and 0.854 from SCONCE2. In tree G (maximally imbalanced, ultrametric), which had higher read depth, the R^2 values from SCONCEmut were 0.907, 0.427, 0.501, and 0.824, and 0.946, 0.777, 0.807, and 0.854 from SCONCE2 (Figure 7G). This demonstrates that although SCONCEmut can partially capture the underlying tree structure and higher read depth improves branch length estimates, SCONCEmut is unable to consistently outperform SCONCE2.

Furthermore, R^2 values of estimates for $t_2 + t_3$ from SCONCEmut in trees A, B, C, and D were all significantly worse than t_1 values, indicating higher confidence in the length of ancestral branch compared to the derived branches. In contrast, R^2 values for $t_2 + t_3$ were on par with those for t_1 in trees E, F, and G (increased read depth). Consistent with results from Recovery of mutation counts, the noise in point mutations on branches t_2 and t_3 due to low read depth obscure the underlying tree structure in trees A, B, C, and D.

Accuracy of phylogeny estimation using neighbor-joining

To reconstruct the tumor's evolutionary history, we estimated phylogenies using neighbor-joining [31, 32] with our $t_2 + t_3$ pairwise distance metric, previously shown to outperform competing methods in [6]. Additionally, because the R^2 values for t_1 estimates were higher than those for $t_2 + t_3$ for all trees (Figure 7), we also estimated phylogenies using t_1. As t_1 measures similarity (ie, shared history) between cells, we converted it to a distance metric as follows:

$$dist(\hat{t}_{1j}) = 1 - \frac{\hat{t}_{1j}}{\max_n(\hat{t}_{1n})} \tag{24}$$

for all n $\hat{t}_{1i}$ estimates.

We compared our branch length distance metrics to three metrics based on copy number profile similarity: the Euclidean distance as used in [4, 73], the cnp2cnp metric [75], and the MEDICC distance [74] (reimplemented by [75]). Each of these distance metrics was run on copy number profiles estimated by SCONCE, SCONCE2, and SCONCEmut (consensus profiles summarized via the mean), as well as on the true simulated copy number profiles, and phylogenies were estimated by neighbor-joining.

We then compared inferred phylogenies with true phylogenies by calculating the Robinson-Foulds (RF) distance [76], shown in Figure 8. In all cases, the $t_2 + t_3$ metric from SCONCE2 had lowest RF distances than other methods, and outperformed the $dist(t_1)$ metric in all cases except for tree D (maximally imbalanced and not ultrametric, with logarithmically decaying external branch lengths), where the RF distance was 10 for $dist(t_1)$ and 16 for $t_2 + t_3$ (Figure 8D). However, SCONCEmut had higher RF distances than SCONCE2 in all cases, except for the $t_2 + t_3$ metric in tree C (maximally imbalanced and not ultrametric, with uniform branch lengths), where the RF distance was 14 for SCONCE2 and 12 for SCONCEmut (Figure 8C). While using $t_2 + t_3$ estimates from SCONCE2 consistently outperforms other methods, neither $dist(t_1)$ nor $t_2 + t_3$ from SCONCEmut reliably has lower RF distances than other methods.

Real data

Using point mutations identified in Detecting Point Mutations in Real Data, we applied SCONCEmut to a 20 cell subset of previously published data [4]. To evaluate the improvement upon SCONCE and SCONCE2, in Figure 9, we highlight differences between copy number calls from SCONCE (independent copy number calls, shown in purple), SCONCE2 (joint CN calls across multiple cells, shown in pink), and SCONCEmut (joint CN calls across multiple cells, using mutations, shown in blue) for cell SRR053672, where SCONCE2 and SCONCEmut consensus profiles were created using the mean. In particular, both SCONCE and SCONCE2 identify a copy number loss (left arrow) that SCONCEmut does not. Upon examination, 8 reads map to the germline allele at 5 sites in this region (that is, the point mutation evidence suggests no copy number changes in this region). In contrast, SCONCE2 calls a copy number gain that SCONCE2 and SCONCEmut do not (right arrow). However, SCONCE2 and SCONCEmut make the same exact call, suggesting no information was gained by including point mutations in this region. Indeed, only 1 read maps to the germline allele in this region, implying one read is not sufficient to gain any evidence for copy number calls.

4.5 Discussion

We present an exploration of a method, SCONCEmut, that expands upon previous work [3, 6] by incorporating point mutations. SCONCEmut jointly estimates copy number profiles, somatic point mutation counts, and pairwise branch lengths in tumor single cell whole genome sequencing data. Although it accurately calls copy number alterations and breakpoint distances, it does not present an improvement over SCONCE2 [6] in estimating branch lengths or phylogenies. We now discuss model limitations, possible sources of error, and future improvements.

As in SCONCE [3] and SCONCE2 [6], SCONCEmut requires matched diploid data that must be sequenced with the same procedure as the tumor data, in order to do GC and

mappability correction. While investigators may be interested in matched normal sequencing to differentiate between germline and somatic point mutations, investigators might not be directly interested in using single cell sequencing to study germline variants due to the increased cost, compared to bulk sequencing. Nonetheless, as a by-product of single cell sequencing, diploid cells are often sequenced as in [4, 5], and can be identified using other methods such as cell sorting.

Additionally, SCONCEmut only infers the counts of mutations on each branch, and is unable to identify which mutations fall on a particular branch, which could provide useful information. For example, identifying the mutations that fall on the ancestral t_1 branch could highlight potential driver mutations. Additionally, SCONCEmut assumes a constant rate of mutations, but the rate of mutation accumulation can increase as the mutational load increases (for example, if DNA repair mechanisms are knocked out) [128]. Although accurate mutation calling remains challenging due to the low sequencing depth in single cell whole genome sequencing, identifying sites of interest and accounting for changes in mutation rate would be improvements to the SCONCEmut model.

Another weakness of SCONCEmut is it models germline variants and somatic mutations using presence/absence indicator variables. Because germline heterozygous sites are not explicitly modeled, SCONCEmut is unable to detect loss of heterozygosity events, a well known phenomenon in breast cancers [129]. Additionally, using indicator variables assumes all sites are biallelic and does not consider sequence composition, such as GC content (which is known to affect mutation rates [130]). Furthermore, if a site is observed in only one cell of a pair, the unobserved cell is assumed to be germline at that site. This can incorrectly inflate our estimates for t_2 and t_3 branch lengths. Explicitly modeling heterozygous sites, sequence content, and missing data is the subject of future work, and could improve estimates from SCONCEmut.

Furthermore, in Equation 20g, we assume all possible derived allele frequencies have equal weight. While this is a helpful simplifying assumption, this results in two unmodeled factors. First, allele frequency is dependent on the age of the point mutation. That is, under the infinite sites model, point mutations at a particular site can only affect one allele and occur once, and require subsequent copy number changes to change the allele frequency. Therefore, point mutations occurring earlier in evolutionary time (ex, on branch t_1) are more likely to experience subsequent copy number amplifications, simply because they have more time to accumulate overlapping copy number events. This could lead to a higher allele frequency than point mutations occurring closer to the leaves of the tree (ex, on branches t_2 or t_3), which have less time to be hit with a subsequent copy number event, making them more likely to have allele frequency $\frac{1}{CN_i}$ for site i. Secondly, allele frequency is dependent on mutation effect and evolutionary dynamics. For example, driver mutations may quickly cause an increase in copy number and allele frequency, which can also affect nearby passenger mutations [131]. One possible approach is to give high derived allele frequencies less weight in Equations 18 and 20g, by using $\mathbb{P}(g = CN_i - s|CN_i) = \left(\frac{1}{CN_i}\right)^s$ instead of $\mathbb{P}(g = CN_i - s|CN_i) = \left(\frac{1}{CN_i}\right)$ (following [122]). Robustly modeling these effects is the subject of future work.

Another possible source of error in SCONCEmut lies in potential optimization problems and failures (see Pipeline details for optimization details), as evidenced by persistently poor tree branch length estimation in simulation sets with higher read counts (trees E, G, and F). Higher read counts should decrease uncertainty and error in mutation count estimation because as read depth increases, all simulated mutations become observed, as seen in the left column of panels F and G in Figure 6 (that is, there is no allelic dropout). However, estimates of mutation counts and, therefore, tree branch lengths, do not improve accordingly. One example of an optimization problem is in trees E, G, and F, the joint optimization for φ, the scaling factor between CNA and point mutations, and ω, the BetaBinomial overdispersion parameter (see Equation S1), failed to move from the initial starting point. That is, instead of using parameter set specific maximum likelihood estimates for φ and ω in downstream optimization steps, common constants (optimization starting points) were used for trees E, G, and F. This could lead to incorrect point mutation count and tree branch length estimates, thereby propagating this error throughout all subsequent analyses. One way to address this problem is to restart the optimization procedure in several different starting points, to avoid getting stuck in local maxima. Another debugging step would be to remove potential confounding variables, such as simulated sequencing error. Furthermore, one could explore estimating tree branch lengths using true/known point mutation counts, rather than estimates, to rule out snowballing point mutation count errors. Despite best efforts to identify and address these optimization errors, some persist, and are the subject of future work.

There are also a number of limitations in our analysis of real data from [4]. First, in order to remove sites with little information, we applied minimum cell and average read depth filters. While this reduces noise, it also biases the mutations analyzed towards those more likely to be on the shared ancestral t_1 branch, and is prone to removing rare point mutations/singletons, which may be of interest to investigators using single cell sequencing. Additionally, the loglikelihood ratio cutoff in Identifying variable sites in tumor data was chosen to give a reasonably sized input dataset, but does not have rigorous statistical backing. Furthermore, we estimated overdispersion, ω, on diploid data (that is, we assumed uniform copy number 2 across the genome). However, because overdispersion might change based on copy number, applying it to sequencing data with copy numbers other than 2 can introduce error when calculating logliklihood ratios (see Identifying variable sites in tumor data). In future studies, high resolution bulk sequencing could be used to identify and filter for high confidence variable sites and rigorously calibrate sequencing error in single cell sequencing data.

4.6 Conclusions

In conclusion, we present an exploration of a model for incorporating somatic point mutations into joint copy number inference and evolutionary distances estimation between cells. SCONCEmut is able to accurately estimate copy number calls and somatic mutation counts,

but does not yet improve upon evolutionary distances estimates from previous work [6]. This work shows that jointly estimating both copy number alterations and counts of somatic mutations can capture novel aspects of cancer evolution, but further work is necessary to improve branch length estimates in this model.

4.7 Appendix

Code Availability

SCONCEmut is implemented in `C++11` and is freely available on GitHub at https://github.com/NielsenBerkeleyLab/SCONCEmut. SCONCEmut requires the `Boost C++ Libraries` (developed on v1.71) and the `GNU Scientific Library` (developed on v2.5) [63], and was developed and tested on Ubuntu 20.04.4. The simulation program, written in `C`, and corresponding parameter files are available on GitHub. Plotting scripts and some of the real data analysis were done in `R` (developed on v4.2.0) [88], and additionally require the `R` packages `ape` [89], `cowplot` [92], `ggplot2` [93], `ggtree` [94–96], `grid` [88], `gtools` [97], `phangorn` [90, 91], plyr [98], `reshape2` [99], `scales` [100], and `stringr` [101].

We analyzed one previously published real dataset from [4], available at the Sequence Read Archive under accession number SRR054616. Pipeline scripts for processing this data are available on GitHub, and require `samtools` (developed on v1.11 with `htslib` v1.11) [68], `bedtools` (developed on v2.27.1) [58], `bedops` (developed on v2.4.35) [132], and `python3` (developed on v3.8.10).

Acknowledgements

We thank Sarah Johnson for performing extensive data analysis on sequencing data from [4], and developing key scripts for identifying point mutations in these data (see Identifying point mutations in real data).

4.8 Figures

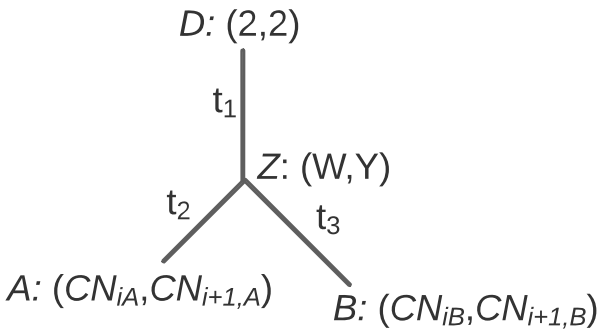


Figure 1: Tree structure for SCONCEmut. Tree $\mathcal{T} = [t_1, t_2, t_3]$ is shown with copy numbers in windows i and $i + 1$ in parentheses. Within this tree structure, cells A and B evolve together from ancestral diploid cell D for time t_1 before divergence at unobserved point Z, with unknown copy state (W, Y). Cells A and B continue to independently evolve for times t_2 and t_3 into copy number states $(CN_{iA}, CN_{i+1,A})$ and $(CN_{iB}, CN_{i+1,B})$, respectively.

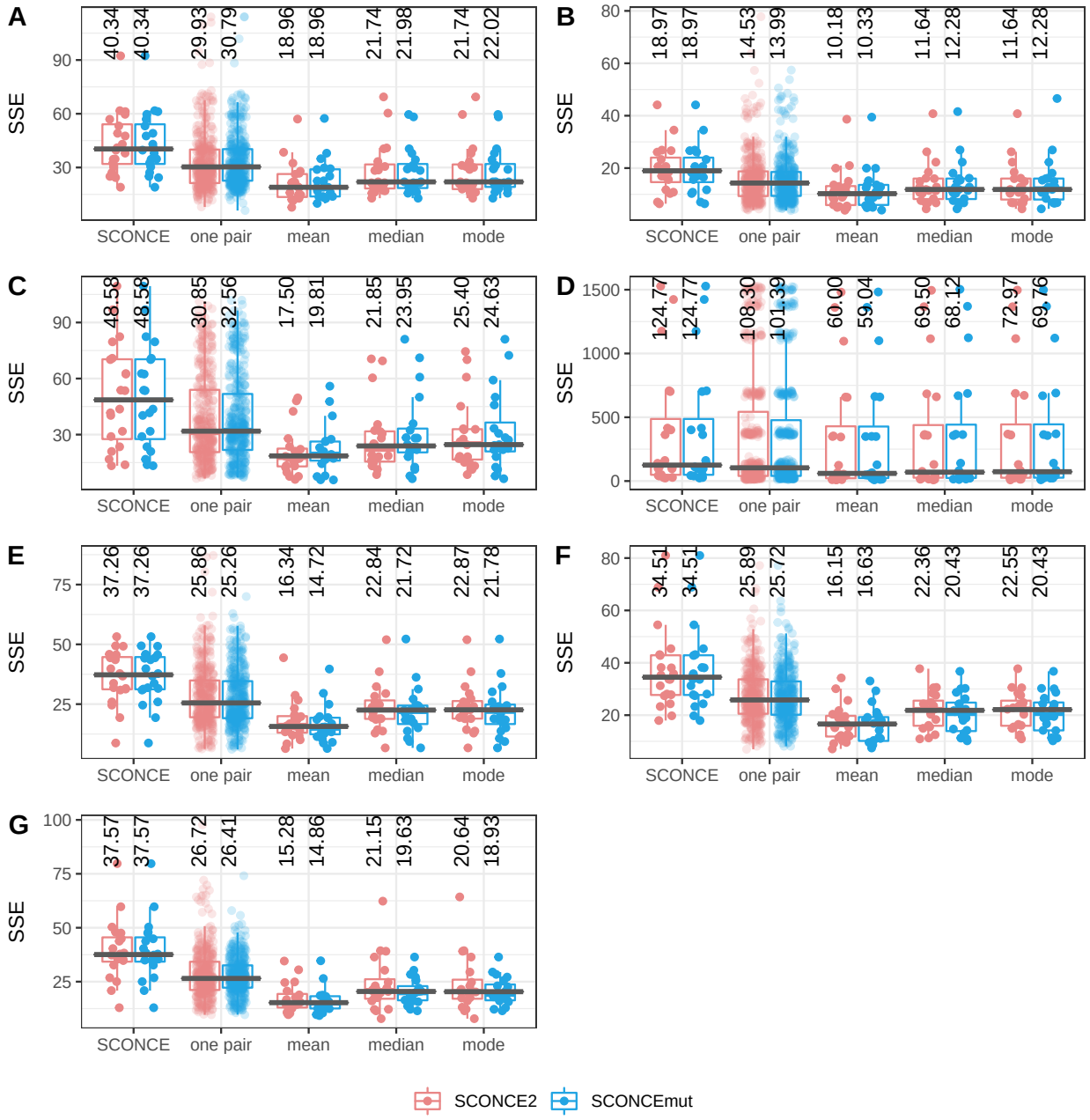

A
B
C
D
E
F
G
SSE
SCONCE
one pair
mean
median
mode
40.34 40.34 29.93 30.79 18.96 18.96 21.74 21.98 21.74 22.02
18.97 18.97 14.53 13.99 10.18 10.33 11.64 12.28 11.64 12.28
48.58 48.58 30.85 32.56 17.50 19.81 21.85 23.95 25.40 24.63
124.77 124.77 108.30 101.39 60.00 59.04 69.50 68.12 72.97 69.76
37.26 37.26 25.86 25.26 16.34 14.72 22.84 21.72 22.87 21.78
34.51 34.51 25.89 25.72 16.15 16.63 22.36 20.43 22.55 20.43
37.57 37.57 26.72 26.41 15.28 14.86 21.15 19.63 20.64 18.93
SCONCE2
SCONCEmut

Figure 2: Boxplots of sum of squared errors (SSE) across simulation sets and methods. The genome wide SSE was calculated between the simulated (true) and inferred copy number profiles. Different copy number calling methods are shown along the x-axis, with SSE on the y-axis. Results shown here include SCONCE (independent cell estimation); mean, median, and mode consensus CNPs from SCONCE2 (no point mutations; pink); and mean, median, and mode consensus CNPs from SCONCEmut (with point mutations; blue). Panel letters correspond to simulation sets described in Supplementary Table S1. Each dot represents one cell, with median SSE shown as a black horizontal bar and at the top of each column. SSE results are consistent across SCONCE2 and SCONCEmut.

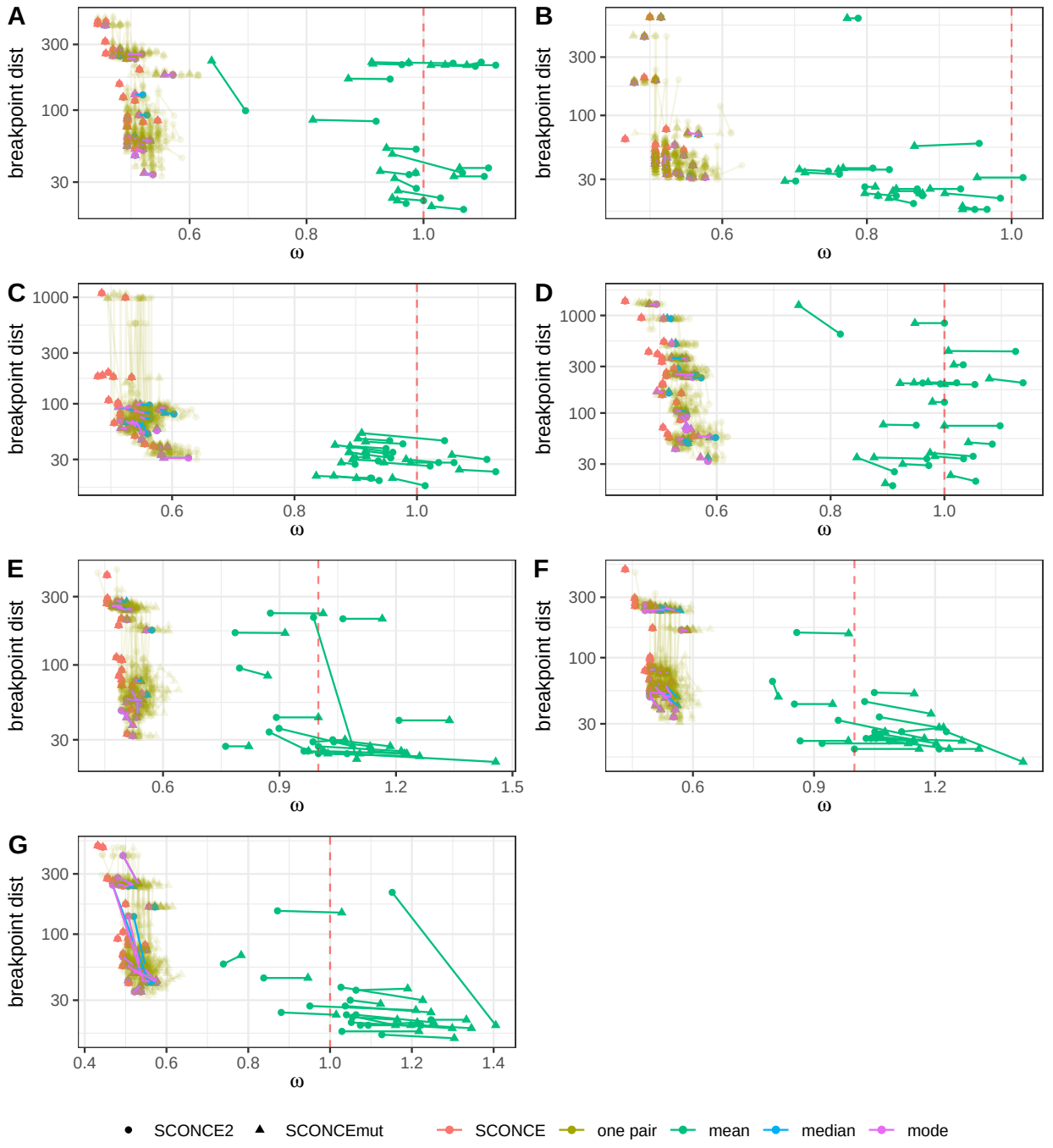

A
B
C
D
E
F
G
breakpoint dist
ω
SCONCE2
SCONCEmut
SCONCE
one pair
mean
median
mode

Figure 3: Breakpoint distance accuracy across simulation sets and methods. Breakpoint distances, shown on the y-axis, are calculated by summing the distance from each true breakpoint to its nearest inferred breakpoint. $\omega = \frac{\text{\# inferred breakpoints}}{\text{\# true breakpoints}}$ values are shown on the x-axis, where ω values closest to 1 (red dashed line) indicate highest accuracy. Each dot represents one cell, with colors indicating summarizing method, and subpanel letters correspond to simulation sets described in Supplementary Table S1. Breakpoint distance accuracy is consistent between SCONCE2 (circles) and SCONCEmut (triangles), with lines connecting cell specific results between SCONCE2 and SCONCEmut.

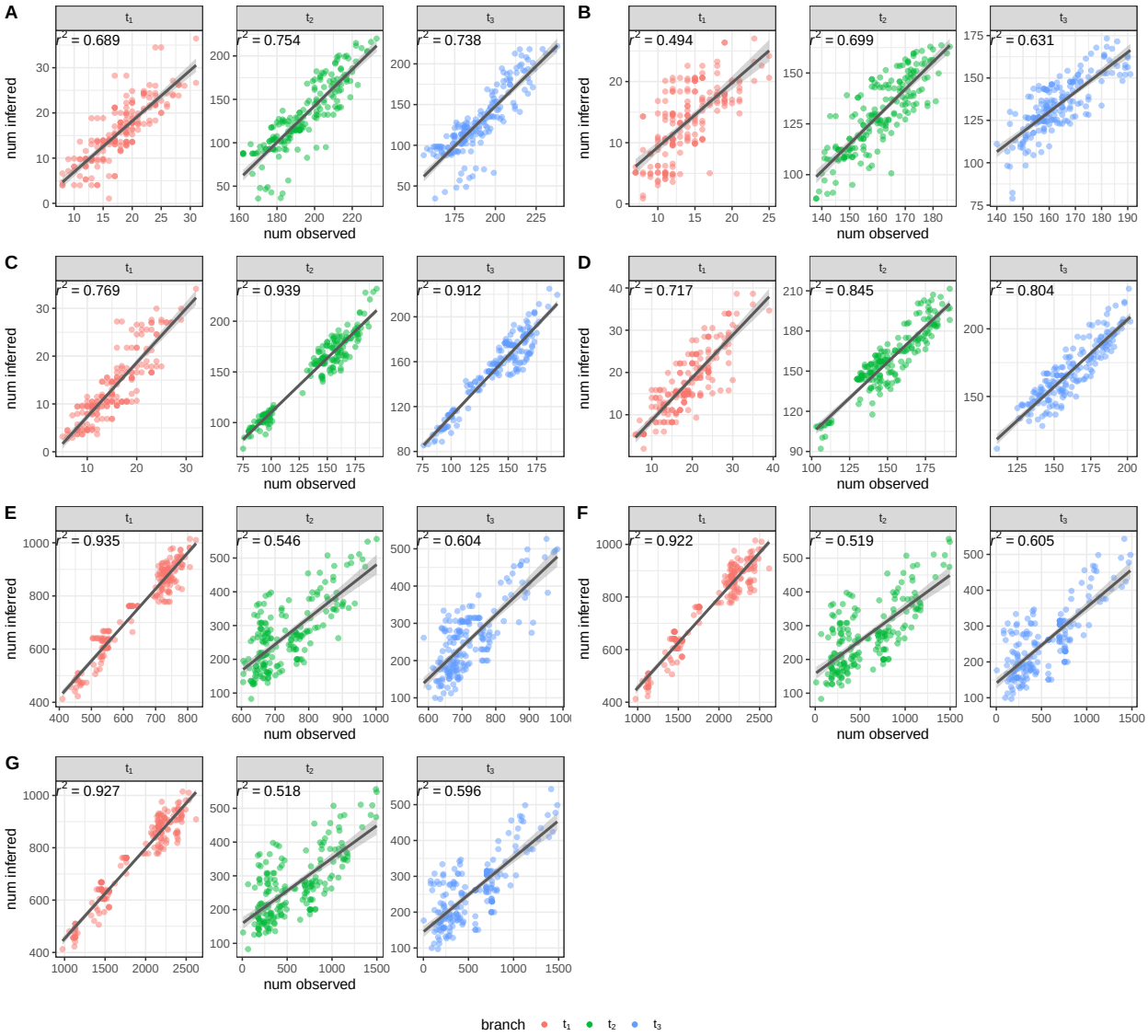
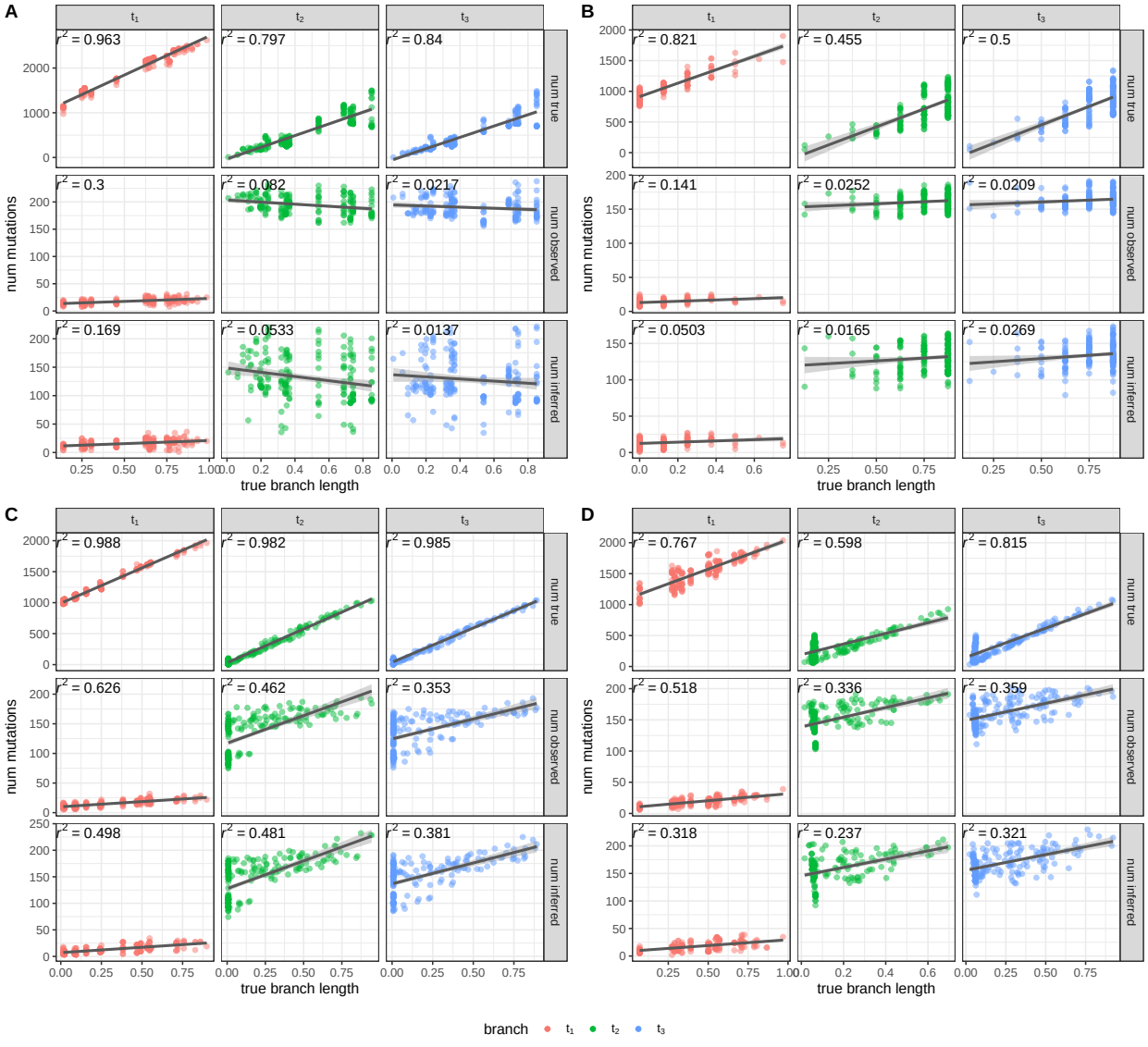


Figure 4: Correlation between branch specific counts of inferred and observed point mutations, for each simulation set. Counts of observed point mutations (ie, mutations with non-zero read counts), for branches t_1, t_2, t_3, are shown on the x-axis, with inferred mutation counts for each branch on the y-axis. Each dot denotes one mutation count estimate, and R^2 values and a linear regression line are shown for each branch specific mutation count. Figure subpanel letters correspond to simulation set letters (see Supplementary Table S1). In all simulation sets, the number of inferred mutations positively correlates with the number of observed mutations.


A
B
C
D
t1
t2
t3
num true
num observed
num inferred
num mutations
true branch length
r2 = 0.963
r2 = 0.797
r2 = 0.84
r2 = 0.3
r2 = 0.082
r2 = 0.0217
r2 = 0.169
r2 = 0.0533
r2 = 0.0137
r2 = 0.821
r2 = 0.455
r2 = 0.5
r2 = 0.141
r2 = 0.0252
r2 = 0.0209
r2 = 0.0503
r2 = 0.0165
r2 = 0.0269
r2 = 0.988
r2 = 0.982
r2 = 0.985
r2 = 0.626
r2 = 0.462
r2 = 0.353
r2 = 0.498
r2 = 0.481
r2 = 0.381
r2 = 0.767
r2 = 0.598
r2 = 0.815
r2 = 0.518
r2 = 0.336
r2 = 0.359
r2 = 0.318
r2 = 0.237
r2 = 0.321
branch t1 t2 t3

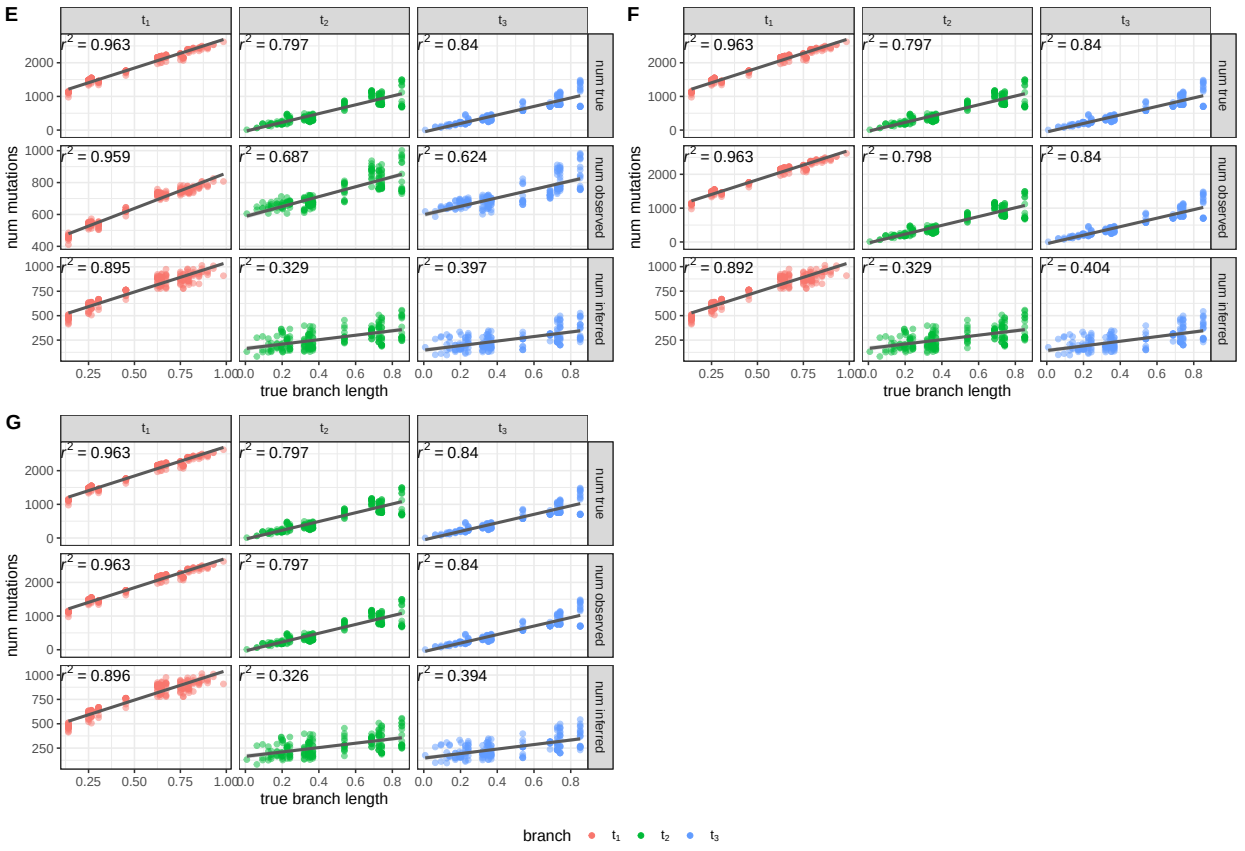
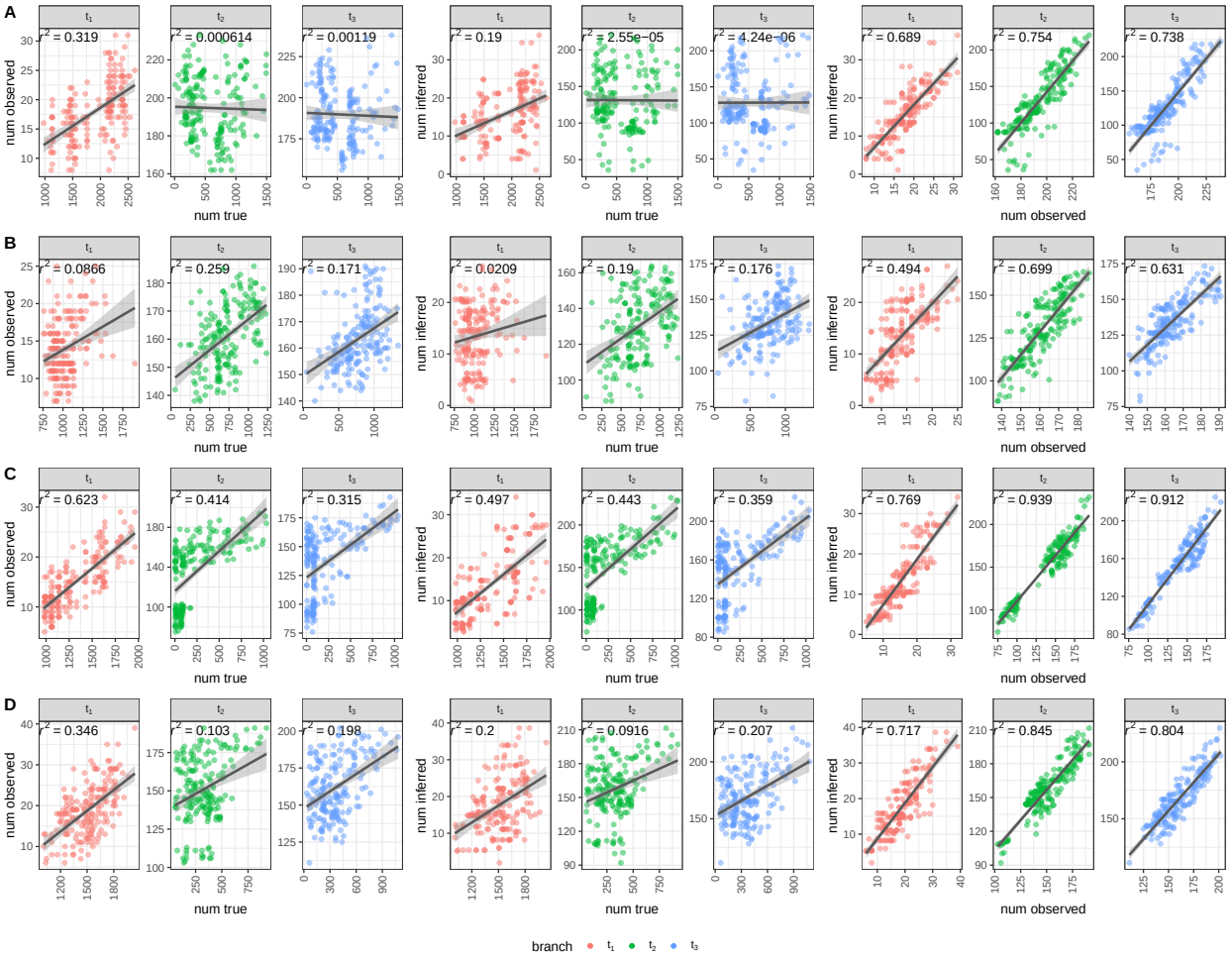


Figure 5: Correlation between point mutation counts and branch lengths for each simulation set. For each simulation set (subpanel labels correspond to simulation sets described in Supplementary Table S1), the true number of mutations (top row), the number of observed mutations (center row), and the number of inferred mutations (bottom row) is shown on the y-axis, while true branch lengths for t_1, t_2, and t_3 are shown on the x-axis. R^2 values between mutation counts and branch lengths are shown for each combination. For all parameter sets, the true numbers of mutations positively correlates with branch lengths. However, the numbers of observed and inferred mutations have weaker correlations due to low read depth.


A
B
C
D
num observed
num inferred
num true
num observed
branch t_1 t_2 t_3

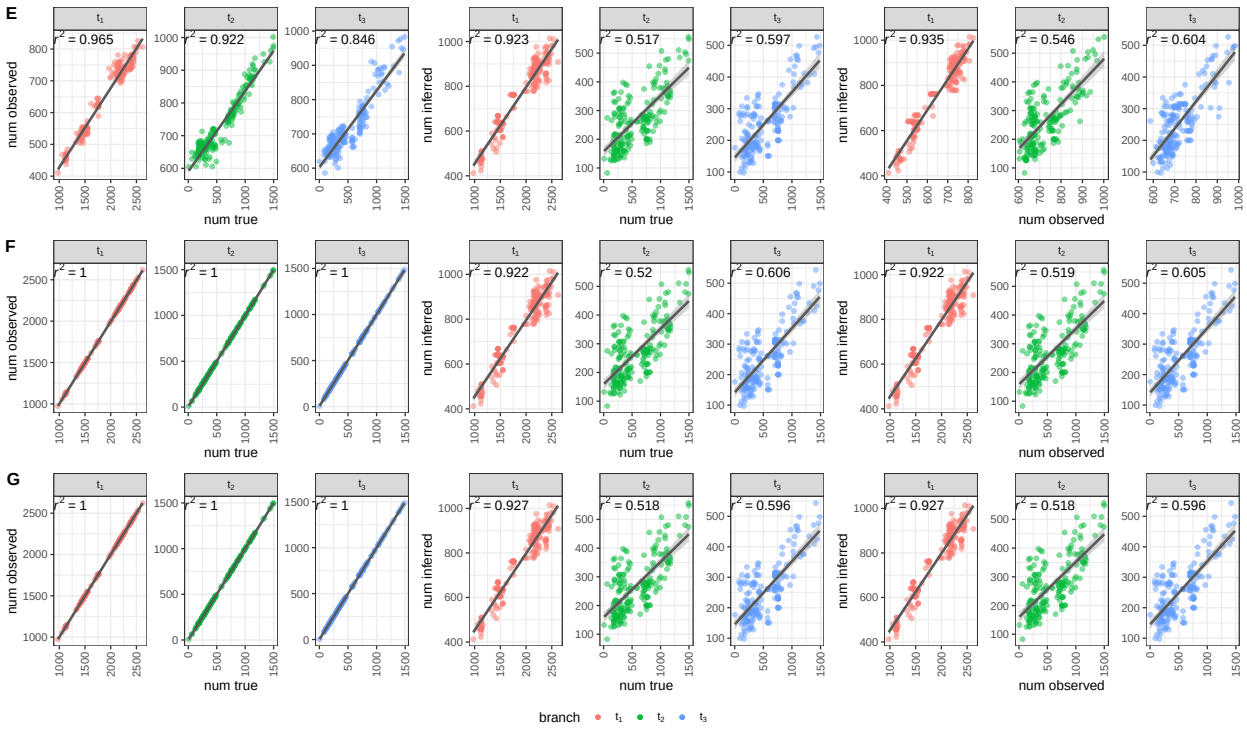
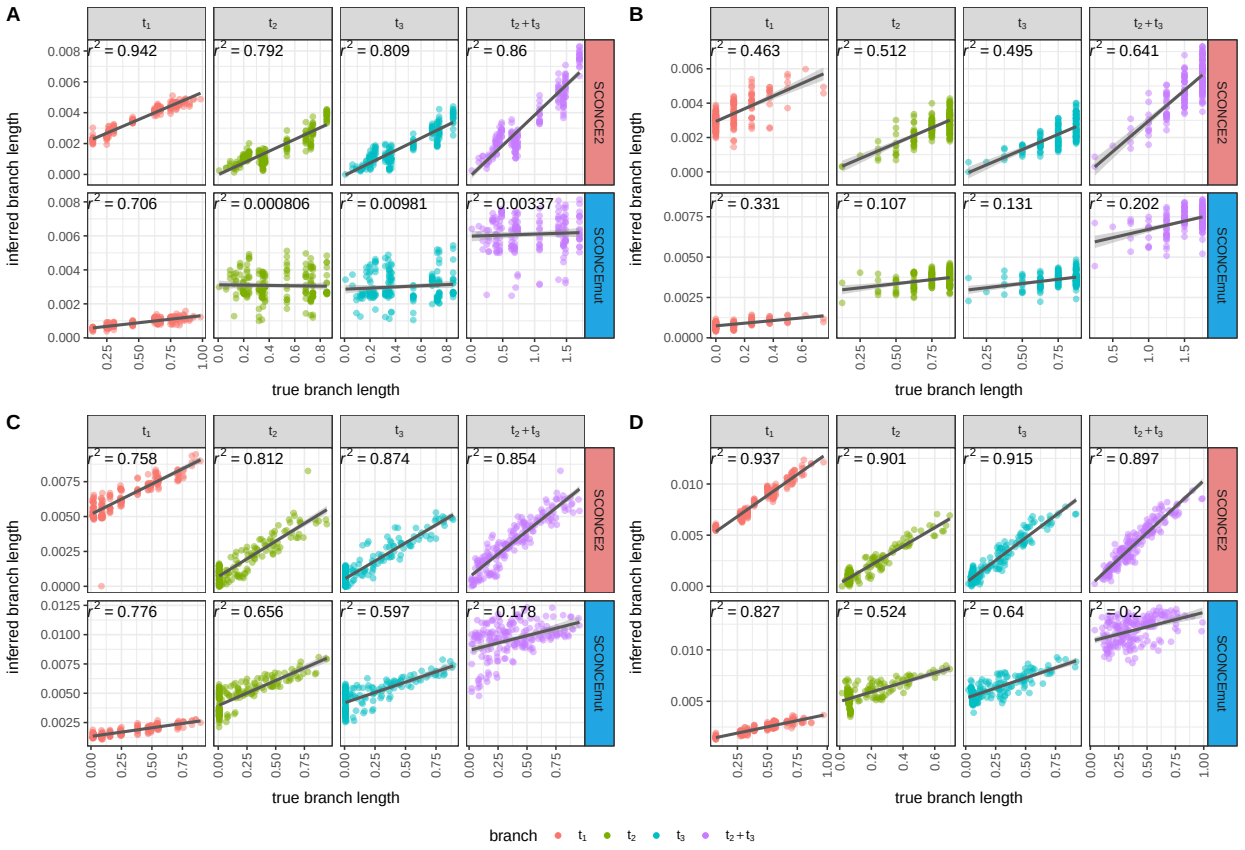


Figure 6: Correlation between branch specific true, observed, and inferred counts of point mutations, for each simulation set. Each row (subpanel letter) corresponds to parameter sets described in Supplementary Table S1. In the three left most columns, the number of observed mutations (ie, mutations that have non-zero reads mapping to them; y-axis) is compared to the true number of mutations (x-axis). In the center three columns, the number of inferred mutations from SCONCEmut (y-axis) is compared to the true number of mutations (x-axis). For comparison, the three right most columns show the comparison of numbers of inferred (y-axis) and observed (x-axis) point mutations (previously shown in Figure 4). Each dot represents one mutation count value, separated by branch (colors), and R^2 values are shown for each subpanel. Across simulation sets, the number of inferred point mutations best correlates with the number of observed mutations, and is limited by the number of observed mutations.


A
B
C
D
t1
t2
t3
t2+t3
SCONCE2
SCONCEmut
inferred branch length
true branch length
r2 = 0.942
r2 = 0.792
r2 = 0.809
r2 = 0.86
r2 = 0.706
r2 = 0.000806
r2 = 0.00981
r2 = 0.00337
r2 = 0.463
r2 = 0.512
r2 = 0.495
r2 = 0.641
r2 = 0.331
r2 = 0.107
r2 = 0.131
r2 = 0.202
r2 = 0.758
r2 = 0.812
r2 = 0.874
r2 = 0.854
r2 = 0.776
r2 = 0.656
r2 = 0.597
r2 = 0.178
r2 = 0.937
r2 = 0.901
r2 = 0.915
r2 = 0.897
r2 = 0.827
r2 = 0.524
r2 = 0.64
r2 = 0.2
branch

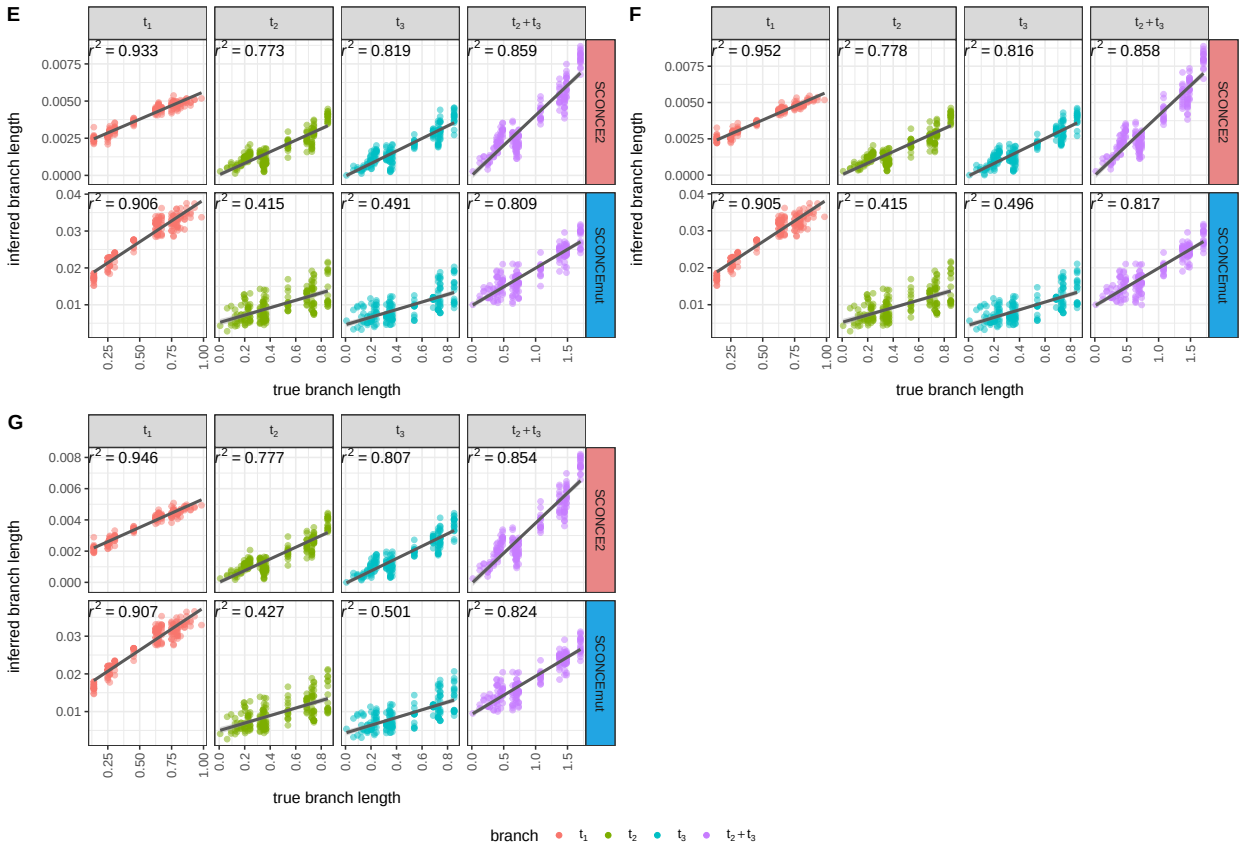


Figure 7: Correlation between inferred and true tree branch lengths for each simulation set. Subpanel letters correspond to parameter set labels in Supplementary Table S1. True tree branch lengths, for branch lengths $t_1, t_2, t_3, t_2 + t_3$, are shown on the x-axis, with inferred branch lengths on the y-axis. Each dot represents on branch length estimate. The top row of each subpanel shows estimates from SCONCE2 (pink label), and the bottom row of each subpanel shows estimates from SCONCEmut (blue label). As true and inferred branch lengths are scaled differently, R^2 values and a linear regression are shown for each branch length. Although SCONCEmut is able to partially capture the underlying tree structure, SCONCE2 performs better.

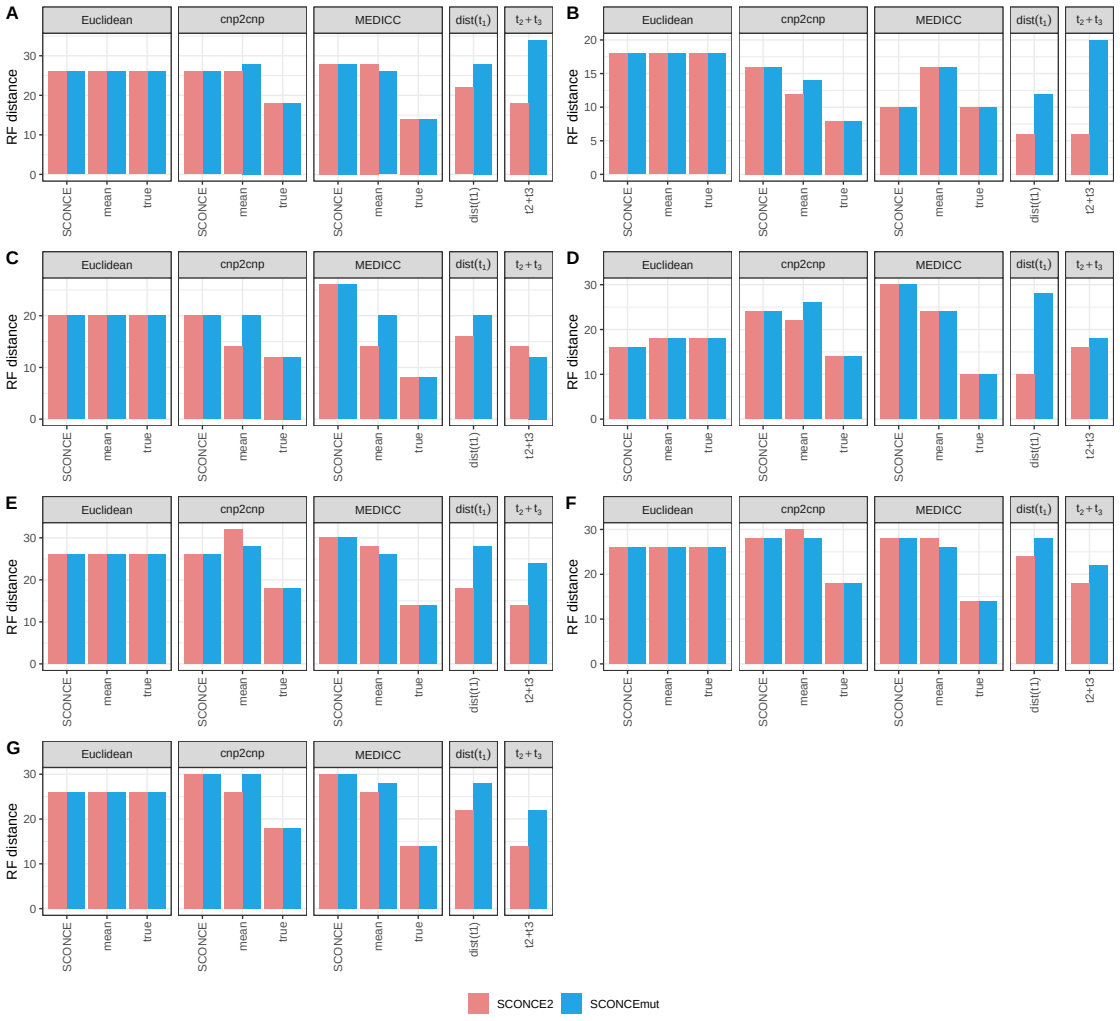


Figure 8: Barplots of Robinson-Foulds (RF) distances between true trees and trees inferred via neighbor-joining with different distance metrics, for each simulation set. Distance metrics based on copy number profile similarity (Euclidean, cnp2cnp [75], and MEDICC [74] distances) were computed on CNPs from SCONCE, SCONCE2 (mean consensus profiles), and SCONCEmut (mean consensus profiles), as well as on true CNPs. These are shown on the x-axis, along with the $dist(t_1)$ and $t_2 + t_3$ metrics from SCONCE2 and SCONCEmut. RF distances are shown on the y-axis for estimates from SCONCE2 (pink) and SCONCEmut (blue). While phylogenies inferred from $t_2 + t_3$ estimates from SCONCE2 consistently have lower or equivalent RF distances than other methods, phylogenies estimated by SCONCEmut do not show a marked or consistent improvement over SCONCE2 or other methods.

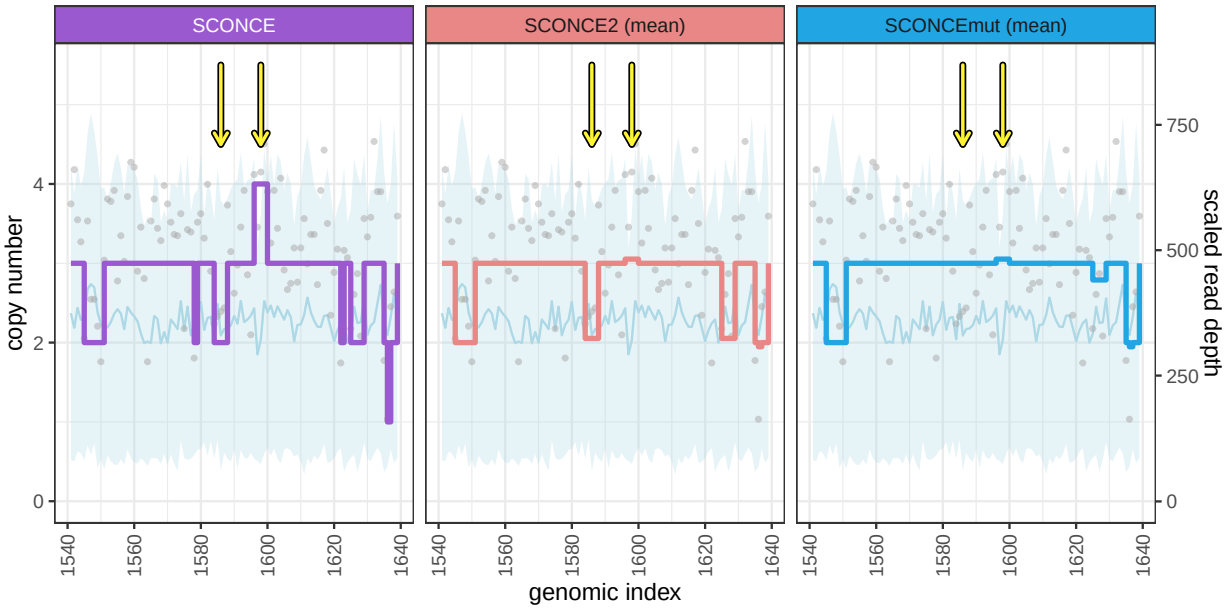


Figure 9: Comparison of genome traces for cell SRR053672 from [4] using SCONCE, SCONCE2, and SCONCEmut. For both SCONCE2 and SCONCEmut, the mean was used to generate consensus profiles for this cell. Genomic window is shown on the x-axis, with read depth on the right y-axis and inferred copy number on the left y-axis. Each dot represents the observed read depth for a given window, and the light blue line shows the mean diploid read depth, ±1 SD (light blue band). Colored lines indicate the copy number profile inferred by each method. Yellow arrows highlight differences between SCONCE (purple), SCONCE2 (pink), and SCONCEmut (blue).

4.9 Supplementary Material

Pipeline details

A graphical overview for the SCONCEmut pipeline is shown in Supplementary Figure S1, where white boxes show input data, pink boxes show steps from SCONCE [3] and SCONCE2 [6] (described in SCONCE and SCONCE2 review), and blue boxes show novel expansions for incorporating point mutations. Each step is detailed in the following sections.

SCONCE and SCONCE2 review

Similar to the pipeline presented in [6], we first estimate constants $\{a, b, c\}$ (see Equation 10e) using diploid data only. We then use SCONCE [3] to do independent parameter estimation on all τ tumor cells, resulting in independent estimates for library size scaling factors, CNA rate parameters, and branch lengths for each cell. The shared rate parameter estimates are summarized using the median, and branch lengths are reestimated for each cell, using these summary rate parameters. This is shown in pink boxes 1 and 2 in Supplementary Figure S1.

Jointly estimating Beta Binomial Parameters

Next, we jointly estimate ω, the overdispersion parameter for the BetaBinomial distribution (defined in Equation 22b), and φ, the scaling factor between CNA and point mutation event rates (defined in Equation 14), across all τ individual tumor cells. To do this, we use the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [125] to optimize the joint likelihood function

$$\ell(\varphi, \omega) = \sum_{j=1}^{\tau} \log \mathbb{P}(X_j = x_j; \varphi t_j) \tag{S1}$$

using a two step optimization procedure. First, for every proposed ω value, we estimate the mutation count, X_j, for the j'th individual cell, by optimizing the following loglikelihood function using the Nelder-Mead simplex optimization algorithm [133]:

$$\mathbb{P}(X_j = x_j|\omega) = \sum_{i=1}^{N_j} \log \left[\mathbb{P}(D_{ij}|S_{ij} = 1, \omega)\frac{x_j}{N_j} + \mathbb{P}(D_{ij}|S_{ij} = 0, \omega)\frac{N_j - x_j}{N_j} \right] \tag{S2}$$

where N_j is the total number of sites observed in cell j. Note, this is the one cell version of the function used to calculate the numbers of mutations in a two cell tree $\mathcal{T}$, defined in Equation 16. Recall ω is only used in calculating the likelihood of the observed read data in the BetaBinomial probability function defined in Equation 22b. This step is trivially parallelizable, as each cell is analyzed independently of each other.

Once mutation counts are estimated for each cell, we sum the loglikelihood of these mutation counts across all cells using Equations 14 and S1:

$$\ell(\varphi, \omega) = \sum_{j=1}^{\tau} \log \mathbb{P}(X_j = x_j; \varphi t_j) \tag{S3a}$$

$$= \sum_{j=1}^{\tau} \log \left[\frac{(\varphi t_j)^{x_j} e^{-\varphi t_j}}{x_j!} \right] \tag{S3b}$$

where t_j is the reestimated branch length estimate for cell j from SCONCE [3] after rate parameter summarizing (see SCONCE and SCONCE2 review).

By maximizing Equation S1 using BFGS, we jointly estimate both ω and φ across all individual tumor cells. This is shown in blue box 3 in Supplementary Figure S1.

Branch length estimation

Using these estimates of φ and ω, we next estimate the number of point mutations, $\{X_1, X_2, X_3\}$, on each tree branch in $\mathcal{T}$, for every pair of cells, by maximizing the summed genome wide loglikelihood of the number of point mutations on each branch (defined in Equation 16) using the Nelder-Mead simplex optimization algorithm [133].

Finally, for every pair of cells, we estimate tree $\mathcal{T}$ by using BFGS to maximize the summed loglikelihood of the tree from copy number alterations, $\ell_{CNA}(\mathcal{T})$, and from point mutations, $\ell_{mut}(\mathcal{T})$ (see Equation 11), shown in blue box 4 in Supplementary Figure S1. The Viterbi decoding on the trained model is used to output the pairwise copy number profiles, which are then summarized into consensus profiles for each cell (pink boxes 5 and 6 in Supplementary Figure S1).

Simulations with Mutations

We simulated 7 different datasets on 4 different tree structures, where the tree structures were chosen to cover the extremes of possible trees and a variety of evolutionary models (such as those described in [26]). Tree A was ultrametric and maximally imbalanced/fully pectinate; tree B was ultrametric and perfectly balanced; tree C was maximally imbalanced and not ultrametric, with uniform branch lengths; and tree D was maximally imbalanced and not ultrametric, with logarithmically decaying terminal branch lengths. To evaluate the effect of read depth on SCONCEmut, we simulated three additional trees, E, F, and G, with the same structure and parameter values as tree A (ultrametric and maximally imbalanced), with the exception of the expected total number of reads per cell (4e7, 4e8, and 4e9, respectively, instead of 4e6, see Supplementary Table S1). The length of each internal branch was set to 1/128 for trees A, C, D, E, F, and G, and to 1/8 for tree B. The length of the branch leading to the root (ie, time to first divergence from ancestral diploid) was set to 1.

For each tree, 128 tumor and 100 diploid cells were simulated, and 20 cells were sampled from trees A, B, C, and D to form test sets. In order to isolate the effect of read depth on

inference accuracy, test sets for trees E, G, and F were formed using the same 20 cell labels sampled from tree A. The genome was divided into 12,397 bins (to match the number of uniform 250kb non-overlapping windows in hg19), and the error rate was set to 1e-3. The simulated genome length was set to 100 (arbitrary units), with amplification and deletion rates ($\frac{\varphi}{\tau_a}$ and $\frac{\delta}{\tau_d}$) and lengths (τ_a and τ_d, relative to the simulated genome length), and expected number of point mutations per time unit per genome length unit, θ, given in Supplementary Table S1. The read length was set to 4.032735e-6 (relative to the simulated genome length of 100) to match the probability of 100bp reads covering one of 2,479,706,951 mappable bases in hg19 (as determined by the Duke Uniqueness of 35bp Windows from ENCODE/OpenChrom (UCSC accession wgEncodeEH000325) [61, 62], to match data from [4]).

The expected read depth across the genome was set to be uniform, in the absence of CNAs. Reads were sampled from a Negative Binomial distribution, with parameter $r = 50$ and 4e6 expected total number of reads for trees A, B, C, and D, and 4e7, 4e8, 4e9 for trees E, F, and G, respectively.

Identifying point mutations in real data

We identified variable tumor sites (ie, somatic point mutations) in real data from [4] with a loglikelihood ratio test using the BetaBinomial distribution defined in Equation 22b. Error and overdispersion were modeled from the diploid cells, described in the following sections. Full scripts for this analysis are provided on GitHub.

Preprocessing sequencing data

Sequencing data was downloaded from [4], and preprocessed as previously described [3, 6], using standard tools. Briefly, reads were trimmed and cleaned using `cutadapt` [64] and `trimmomatic` [65], low complexity reads were removed using `prinseq` [66], reads were aligned to hg19 with `bowtie2` [67], reads with q scores less than 20 were removed using `samtools` [68], and PCR duplicates were removed using `picard` [69]. Diploid cells were identified using orthogonal cell sorting as described in [4].

Identifying the major germline allele

All diploid cells were pooled and converted into an mpileup file using `samtools` [68], and the major germline allele was found in the pooled diploid sample. Then, for each individual diploid and tumor cell, the number of reads mapping to the germline/ancestral (defined by the major allele in the pooled diploid sample) and somatic/derived alleles were counted genome wide. The script to do this (`find_major_allele.py`) is available on GitHub.

Modeling overdispersion using the diploid data

To estimate ω, the overdispersion parameter defined in Equation 22b, we first identified observations likely to be sequencing error in the diploid data. To do this, we filtered for sites that were not likely to be variable and were not present in dbSNP (that is, sites not known to be variable in humans).

To identify non variable sites in the diploid data, we first estimated the allele frequency, f_{gij}, for major germline allele $g \in \{A, C, G, T\}$, for each site i in each cell j:

$$\hat{f}_{gij} = \min\left\{\frac{n_{gij} - \varepsilon \times n_{ij}}{(1-2\varepsilon)n_{ij}}, 1\right\} \tag{S4}$$

where n_{ij} and n_{gij} are the total number of reads and number of reads mapping to the major germline allele, respectively, and $\varepsilon = 5e-3$ is the sequencing error rate. The log likelihood function assumed to derive this expression is:

$$\begin{aligned}\ell(f_{gij}) = n_{gij}\log[(1-\varepsilon)f_{gij} + \varepsilon(1-f_{gij})] + \\ (n_{ij} - n_{gij})\log[(\varepsilon f_{gij} + (1-\varepsilon)(1-f_{gij})]\end{aligned} \tag{S5}$$

Then, we calculated the following loglikelihood ratio:

$$LLR_{diploid,ij} = \ell(\hat{f}_{gij}) - \ell(f_{gij} = 1) \tag{S6a}$$

$$\begin{aligned}= n_{gij}\left[\log(\varepsilon + f_{gij} - 2\varepsilon f_{gij}) - \log(1-\varepsilon)\right] - \\ (n_{ij} - n_{gij})\left[\log(\varepsilon) - \log(1-\varepsilon - f_{gij} + 2\varepsilon f_{gij})\right]\end{aligned} \tag{S6b}$$

Summed across d diploid cells, this gives

$$LLR_{diploid,i} = \sum_{j=1}^{d} LLR_{diploid,ij} \tag{S7}$$

such that high values of $LLR_{diploid,i}$ indicate high evidence for site i to be variable within the diploid cells, while low values of $LLR_{diploid,i}$ indicate low evidence for variability.

We then evaluated the number of sites by dbSNP membership (build 155) [134]. The fraction of sites in dbSNP above a given loglikelihood ratio cutoff grows quickly, with 95% of all diploid sites appearing in dbSNP above a LLR cutoff of 33 (blue vertical line in Figure S2A). In contrast, the total number of sites above a given LLR cutoff drops precipitously, from 447,672,658 sites in dbSNP (blue) and 869,879,075 sites not in dbSNP (pink) at a LLR cutoff of 0, to 12691 and 604 sites, respectively, above a LLR cutoff of 33 (see Figure S2B). We then selected all sites with $LLR_{diploid,i}$ values of 32 or less that were not present in dbSNP (that is, sites that are likely to contain sequencing errors, following [127]), resulting in 869,878,372 sites.

Because single cell data is so sparse, we additionally removed sites that had data from very few cells or had very low average coverage per observation. Each of these filtered

sites was observed in 2.47439 diploid cells on average, with an average of 1.75195 reads per observation. To filter out non informative sites due to low sequencing depth, we removed any sites that were observed in fewer than 5 cells ($\approx 2 \times 2.47439$) or had fewer than 3.5 ($\approx 2 \times 1.75195$) reads per observation on average, resulting in 3,959,722 sites.

Next, we estimated the BetaBinomial overdispersion parameter, ω, by maximizing the following likelihood function using the Brent algorithm for one dimensional optimization [135]:

$$\ell(\omega) = \sum_{j=1}^{d} \sum_{i=1}^{N} \mathbb{P}(D_{ij} = (a_{ij}, n_{ij}) | f, \omega) \tag{S8a}$$

$$= \sum_{j=1}^{d} \sum_{i=1}^{N} \binom{n_{ij}}{a_{ij}} \frac{B(a_{ij} + f\omega, n_{ij} - a_{ij} + (1-f)\omega)}{B(f\omega, (1-f)\omega)} \tag{S8b}$$

where

$$d = \#\text{ of diploid cells} \tag{S9a}$$

$$N = \#\text{ of sites} \tag{S9b}$$

$$f = 1 - \varepsilon \tag{S9c}$$

$$\varepsilon = \text{sequencing error} \tag{S9d}$$

resulting in $\hat{\omega} = 8.68008$.

Note this is the same distribution described in Equation 22b. Here, however, f, the major germline allele frequency, is set to $1 - \varepsilon$, under the null hypothesis that the sites under consideration are not variable (that is, any allele frequency not equal to 1 is due to sequencing error). The full script (`fitBetaBinomNull.R`) is available on GitHub.

Identifying variable sites in tumor data

To identify sites in tumor cells that are likely to be somatic point mutations, we removed any sites that were observed in fewer than 5 cells or had fewer than 3.5 average reads per observation (ie, the same cutoffs as in the diploid data), or had only reads mapping to the ancestral allele (ie, no variability in the observed tumor data), resulting in 433,947 sites. We note requiring sites to have read data in multiple cells biases our results, by selecting sites more likely to be on the shared t_1 branch, but we chose to keep this filter to reduce the number of spurious somatic mutations with very little evidence.

Next, for each site i, we estimated the tumor somatic allele frequency, f_i, by maximizing the following loglikelihood function (first defined in Equation 22b), again using the Brent

algorithm [135]:

$$\ell(f_i) = \sum_{j=1}^{\tau} \log\left[\mathbb{P}(D_{ij} = (a_{ij}, n_{ij}) | f_i, \hat{\omega})\right] \tag{S10a}$$

$$= \sum_{j=1}^{\tau} \log\left[\binom{n_{ij}}{a_{ij}} \frac{B(a_{ij} + f_i\hat{\omega}, n_{ij} - a_{ij} + (1-f_i)\hat{\omega})}{B(f_i\hat{\omega}, (1-f_i)\hat{\omega})}\right] \tag{S10b}$$

where

$$\tau = \#\text{ of tumor cells} \tag{S11a}$$

$$\hat{\omega} = 8.68008 \text{ (from Modeling overdispersion using the diploid data)} \tag{S11b}$$

We note using $\hat{\omega}$ from Modeling overdispersion using the diploid data has limitations, as $\hat{\omega}$ was estimated using diploid (ie, copy number of 2) data. However, changes in copy number could lead to differences in overdispersion, thereby potentially biasing or invalidating this analysis. Further modeling of overdispersion in non-diploid data could be accomplished by comparing bulk and single cell sequencing, where bulk tumor sequencing could be used to robustly identify variable sites. Then, overdispersion on these known variable sites could be estimated in a copy number specific manner.

Using the estimates $\hat{f}_i$ and Equation S10, we then calculate the following loglikelihood ratio for each site i:

$$LLR_{tumor,i} = \ell(\hat{f}_i) - \ell(f_i = 0) \tag{S12}$$

All sites with LLR_{tumor_i} values greater than 3 were carried forward, resulting in 39,180 somatic sites. 3136.97 sites were observed, on average, in each cell, with an average of 10.0847 sites with reads aligning to the derived allele.

Finally, cell specific lists of somatic sites and binned read depths were processed through SCONCEmut. The full program (`fitBetaBinomTumor.cpp`) to calculate $LLR_{tumor,i}$ values for all i is provided on GitHub.

Supplementary Figures

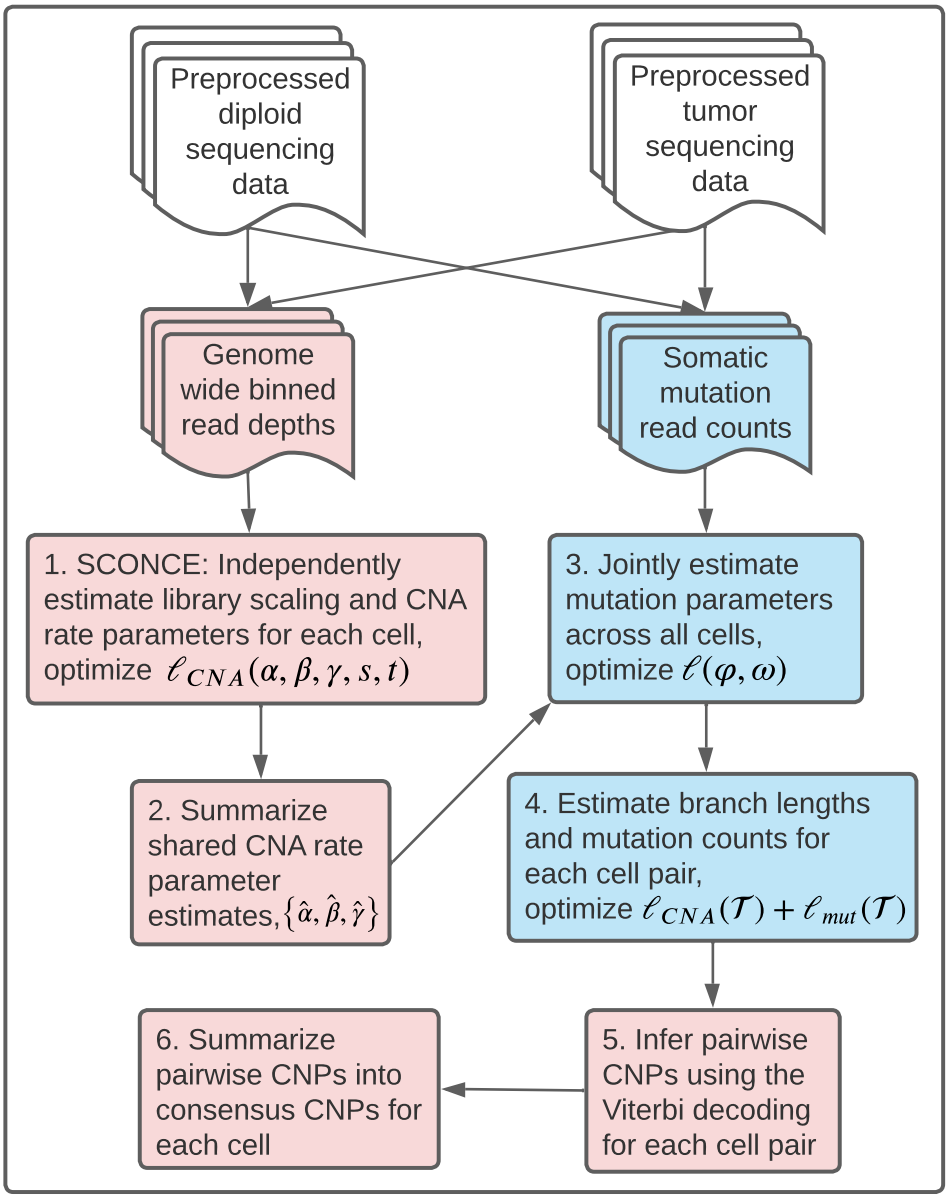


Figure S1: Pipeline overview for SCONCEmut. White boxes indicate input data, pink boxes denote steps previously described in SCONCE2 [6], and blue boxes denote steps where point mutation data has been added for SCONCEmut. Investigators must first clean and align diploid and tumor sequencing data, count the number of reads falling into each genomic bin, and count the number of reads aligning to each allele for all somatic point mutations. SCONCEmut starts by calling SCONCE [3] on each cell to independently estimate library size scaling and CNA rate parameters by optimizing the forward loglikelihood given by the copy number HMM (box 1). Summary estimates of CNA rate parameters are calculated with the median across cells (box 2). Next, mutation parameters φ and ω are jointly estimated across all cells (box 3). Then, branch lengths and mutation counts are estimated for each pair, by optimizing $\ell(\mathcal{T}) = \ell_{CNA}(\mathcal{T}) + \ell_{mut}(\mathcal{T})$ (box 4). Finally, the Viterbi decoding is used to estimate pairwise copy number profiles (CNPs) for each cell (box 5), and consensus CNPs are called for each cell using a summary statistic (mean, median, or mode; box 6).

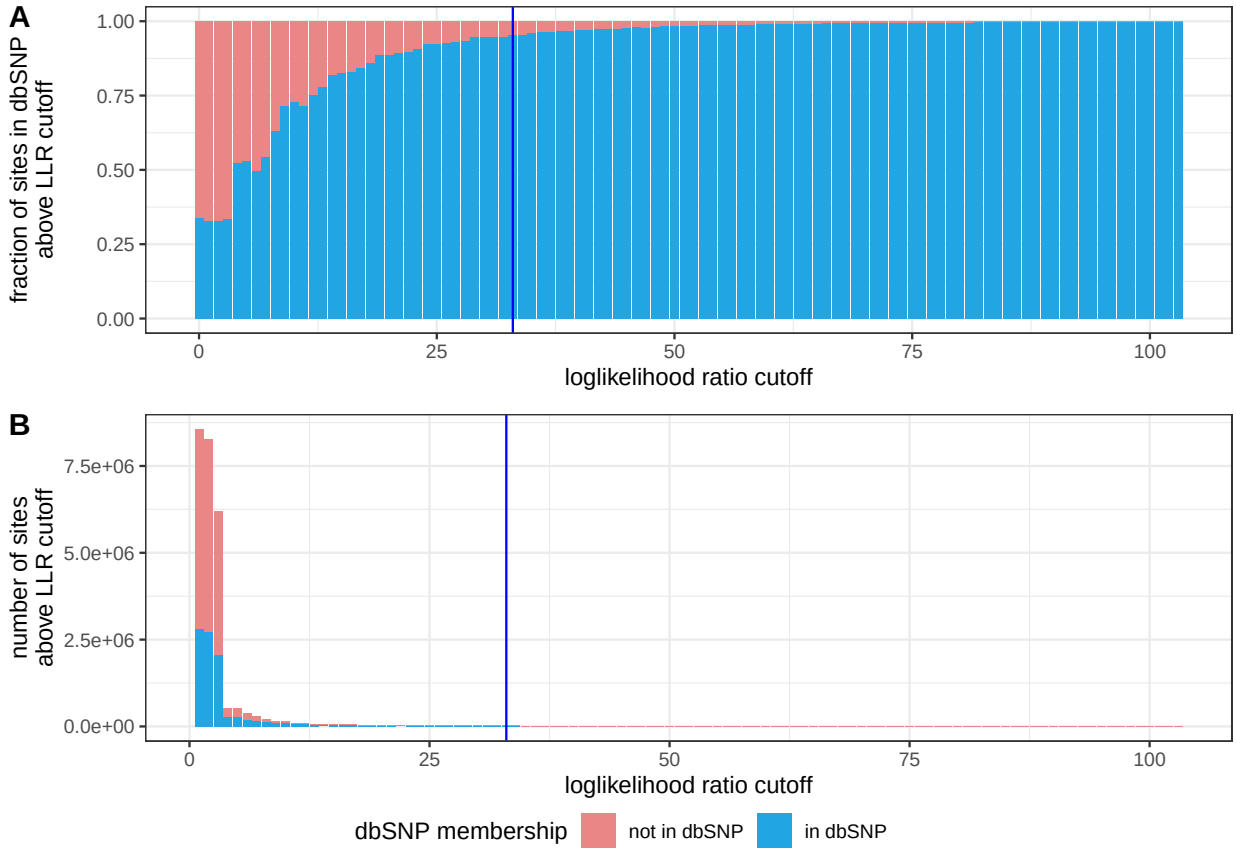


Figure S2: Fraction and number of sites by dbSNP membership and loglikelihood ratio for site variability in diploid data from [4]. Each bar represents the fraction or number of sites above a given loglikelihood ratio (LLR) along the x-axis, where loglikelihood ratios are calculated using Equation S7 across all diploid cells. Color indicates membership in dbSNP (version 155), where pink means a site is not in dbSNP and blue means a site is present in dbSNP. The vertical dark blue line shows the LLR cutoff of 33, where 95% of sites are in dbSNP. All sites not in dbSNP (pink) with $LLR < 33$ were used to fit the overdispersion parameter ω in Equation S8b. Panel A shows the fraction of sites, while panel B shows the total number of sites. Note that for plot scaling reasons, the first column ($LLR = 0$) in panel B is excluded, where 447,672,658 sites were in dbSNP and 869,879,075 sites were not (1,317,551,733 sites total).

Supplementary Tables

Simulation set/ tree	**Short description**	**Deletion rate, $\frac{\delta}{\tau_d}$**	**Amplification rate, $\frac{\varphi}{\tau_a}$**	**Mean deletion length, τ_d**	**Mean amplification length, τ_a**	**Expected number of point mutations, θ**	**Expected total number of reads per cell**
A	maximally imbalanced, ultrametric	0.02	0.02	8	8	8	4e6
B	perfectly balanced, ultrametric	0.02	0.02	8	8	8	4e6
C	maximally imbalanced, not ultrametric, all branches have equal length	0.055	0.055	7.5	6	8	4e6
D	maximally imbalanced, not ultrametric, terminal branch lengths decay logarithmically	0.055	0.055	7.5	6	8	4e6
E	maximally imbalanced, ultrametric	0.02	0.02	8	8	8	4e7
F	maximally imbalanced, ultrametric	0.02	0.02	8	8	8	4e8
G	maximally imbalanced, ultrametric	0.02	0.02	8	8	8	4e9

Table S1: Description of simulation parameters, relative to a genome length of 100 (arbitrary units), for all simulated trees. Simulation values and tree structures were chosen to demonstrate a wide variety of possible trees.

	SCONCE	**one pair**	**mean**	**median**	**mode**
A	120.5	92.5	45	88.5	88.5
B	50.5	44	25	41.5	41.5
C	92.5	70	29.5	61.5	64
D	220	231.5	102	194.5	196
E	109	76	29.5	64	64.5
F	83.5	71	25.5	64.5	64.5
G	89	67	23.5	61.5	61.5

(a) Median breakpoint distances from SCONCE2

	SCONCE	**one pair**	**mean**	**median**	**mode**
A	120.5	94	50.5	89.5	89.5
B	50.5	44	25	44	44
C	92.5	71	31	66.5	68.5
D	220	232	103	198	199.5
E	109	70.5	27	59.5	60
F	83.5	66.5	22.5	64.5	64.5
G	89	61.5	21	55	55.5

(b) Median breakpoint distances from SCONCEmut

Table S2: Median breakpoint distances for each simulation set and method.

	SCONCE	**one pair**	**mean**	**median**	**mode**
A	0.493	0.5105	0.9938	0.5072	0.507
B	0.5081	0.5231	0.8666	0.5231	0.5231
C	0.5155	0.5596	0.9564	0.5613	0.5526
D	0.5062	0.5395	1.0108	0.5404	0.5361
E	0.4928	0.519	0.987	0.5129	0.5129
F	0.4937	0.52	1.0452	0.5185	0.5099
G	0.4938	0.52	1.0507	0.52	0.5067

(a) Median ω values from SCONCE2

	SCONCE	**one pair**	**mean**	**median**	**mode**
A	0.493	0.5067	0.9529	0.5064	0.5002
B	0.5081	0.5224	0.8333	0.5231	0.5231
C	0.5155	0.5524	0.9067	0.5505	0.5449
D	0.5062	0.5378	0.973	0.5458	0.5378
E	0.4928	0.5316	1.1027	0.5201	0.5201
F	0.4937	0.5362	1.1543	0.5339	0.5313
G	0.4938	0.5432	1.2116	0.5368	0.5321

(b) Median ω values from SCONCEmut

Table S3: Median $\omega = \frac{\#\text{ inferred breakpoints}}{\#\text{ true breakpoints}}$ for each simulation set and method.

Bibliography

1. The Cancer Genome Atlas Research Network *et al.* The Cancer Genome Atlas Pan-Cancer analysis project. *Nature genetics* **45,** 1113–20. ISSN: 1546-1718. http://dx.doi.org/10.1038/ng.2764 (Oct. 2013).
2. Gawad, C., Koh, W. & Quake, S. R. Single-cell genome sequencing: current state of the science. *Nature Reviews Genetics* **17,** 175–188. ISSN: 1471-0056. http://www.nature.com/articles/nrg.2015.16 (Mar. 2016).
3. Hui, S. & Nielsen, R. SCONCE: a method for profiling copy number alterations in cancer evolution using single-cell whole genome sequencing. *Bioinformatics.* ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/38/7/1801/6515610 (Jan. 2022).
4. Navin, N. *et al.* Tumour evolution inferred by single-cell sequencing. *Nature* **472,** 90–94. ISSN: 0028-0836. http://www.nature.com/doifinder/10.1038/nature09807 (Apr. 2011).
5. 10x Genomics. *Breast Tissue nuclei sections A-E (v1, 84x100)* May 2019. https://cf.10xgenomics.com/samples/cell-dna/1.1.0/breast_tissue_aggr_10k/breast_tissue_aggr_10k_web_summary.html.
6. Hui, S. & Nielsen, R. SCONCE2: jointly inferring single cell copy number profiles and tumor evolutionary distances. *bioRxiv,* 2022.05.12.491742. https://www.biorxiv.org/content/10.1101/2022.05.12.491742v1 (May 2022).
7. Vogelstein, B. & Kinzler, K. W. The multistep nature of cancer. *Trends in Genetics* **9,** 138–141. ISSN: 0168-9525 (Apr. 1993).
8. Baylin, S. B. & Jones, P. A. Epigenetic Determinants of Cancer. *Cold Spring Harbor Perspectives in Biology* **8.** ISSN: 19430264. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5008069/ (Sept. 2016).
9. Klein, G. & Klein, E. Oncogene activation and tumor progression. *Carcinogenesis* **5,** 429–435 (1984).
10. Haluska, F. G., Tsujimoto, Y. & Croce, C. M. Oncogene Activation by Chromosome Translocation in Human Malignancy. *Annual Review of Genetics* **21,** 321–345. ISSN: 00664197. https://doi.org/10.1146/annurev.ge.21.120187.001541 (Nov. 1987).

11. Upender, M. B. *et al.* Chromosome Transfer Induced Aneuploidy Results in Complex Dysregulation of the Cellular Transcriptome in Immortalized and Cancer Cells. *Cancer Research* **64,** 6941–6949. ISSN: 0008-5472. https://aacrjournals.org/cancerres/article/64/19/6941/511822/Chromosome-Transfer-Induced-Aneuploidy-Results-in (Oct. 2004).
12. Levine, A. J. The Tumor Suppressor Genes. *Annual Review of Biochemistry* **62,** 623–651. ISSN: 00664154. https://doi.org/10.1146/annurev.bi.62.070193.003203 (Nov. 1993).
13. Wang, L. H., Wu, C. F., Rajasekaran, N. & Shin, Y. K. Loss of Tumor Suppressor Gene Function in Human Cancer: An Overview. *Cellular Physiology and Biochemistry* **51,** 2647–2693. ISSN: 1015-8987. https://www.karger.com/Article/FullText/495956 (Jan. 2018).
14. Knudson, A. G. Hereditary cancer: Two hits revisited. *Journal of Cancer Research and Clinical Oncology* **122,** 135–140. ISSN: 0171-5216. http://link.springer.com/10.1007/BF01366952 (Mar. 1996).
15. Knudson, A. G. Two genetic hits (more or less) to cancer. *Nature reviews. Cancer* **1,** 157–62. ISSN: 1474-175X. http://dx.doi.org/10.1038/35101031 (Nov. 2001).
16. Vogelstein, B. *et al.* Cancer Genome Landscapes. *Science* **339,** 1546–1558. http://www.sciencemag.org/content/339/6127/1546.abstract (Mar. 2013).
17. Hu, T., Chitnis, N., Monos, D. & Dinh, A. Next-generation sequencing technologies: An overview. *Human Immunology* **82,** 801–811. ISSN: 0198-8859. https://doi.org/10.1016/j.humimm.2021.02.012 (Nov. 2021).
18. Dohm, J. C., Lottaz, C., Borodina, T. & Himmelbauer, H. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Research* **36,** 105. ISSN: 0305-1048. https://academic.oup.com/nar/article/36/16/e105/6334555 (Sept. 2008).
19. Schwartz, S., Oren, R. & Ast, G. Detection and Removal of Biases in the Analysis of Next-Generation Sequencing Reads. *PLOS ONE* **6,** e16685. ISSN: 1932-6203. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0016685 (2011).
20. Tomczak, K., Czerwińska, P. & Wiznerowicz, M. The Cancer Genome Atlas (TCGA): an immeasurable source of knowledge. *Contemporary oncology (Poznan, Poland)* **19,** 68–77. ISSN: 1428-2526. http://www.ncbi.nlm.nih.gov/pubmed/25691825 (2015).
21. Gerstung, M. *et al.* The evolutionary history of 2,658 cancers. *Nature 2020 578:7793* **578,** 122–128. ISSN: 1476-4687. https://www.nature.com/articles/s41586-019-1907-7 (Feb. 2020).
22. The ICGC/TCGA Pan-Cancer Analysis of Whole Genomes Consortium *et al.* Pan-cancer analysis of whole genomes. *Nature 2020 578:7793* **578,** 82–93. ISSN: 1476-4687. https://www.nature.com/articles/s41586-020-1969-6 (Feb. 2020).

23. Ganini, C. *et al.* Global mapping of cancers: The Cancer Genome Atlas and beyond. *Molecular Oncology* **15,** 2823–2840. ISSN: 1878-0261. https://doi.org/10.1002/1878-0261.13056 (Nov. 2021).
24. Navin, N. E. Cancer genomics: one cell at a time. *Genome biology* **15,** 452. ISSN: 1474760X. https://genomebiology.biomedcentral.com/articles/10.1186/s13059-014-0452-9 (Aug. 2014).
25. Wang, Y. & Navin, N. E. Advances and Applications of Single-Cell Sequencing Technologies. *Molecular Cell* **58,** 598–609. ISSN: 1097-2765 (May 2015).
26. Davis, A., Gao, R. & Navin, N. Tumor evolution: Linear, branching, neutral or punctuated? *Biochimica et Biophysica Acta (BBA) - Reviews on Cancer* **1867,** 151–161. https://www.sciencedirect.com/science/article/pii/S0304419X17300197 (Apr. 2017).
27. Ellsworth, D. L. *et al.* Single-cell sequencing and tumorigenesis: improved understanding of tumor evolution and metastasis. *Clinical and Translational Medicine 2017 6:1* **6,** 1–19. ISSN: 2001-1326. https://link.springer.com/article/10.1186/s40169-017-0145-6 (Apr. 2017).
28. Ren, X., Kang, B. & Zhang, Z. Understanding tumor ecosystems by single-cell sequencing: promises and limitations. *Genome Biology 2018 19:1* **19,** 1–14. ISSN: 1474-760X. https://genomebiology.biomedcentral.com/articles/10.1186/s13059-018-1593-z (Dec. 2018).
29. Ramón y Cajal, S. *et al.* Clinical implications of intratumor heterogeneity: challenges and opportunities. *Journal of Molecular Medicine (Berlin, Germany)* **98,** 161. ISSN: 14321440. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7007907/ (Feb. 2020).
30. Bai, X., Li, Y., Zeng, X., Zhao, Q. & Zhang, Z. Single-cell sequencing technology in tumor research. *Clinica Chimica Acta* **518,** 101–109. ISSN: 0009-8981. https://doi.org/10.1016/j.cca.2021.03.013 (July 2021).
31. Saitou, N. & Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution* **4,** 406–425. ISSN: 0737-4038. https://academic.oup.com/mbe/article/4/4/406/1029664 (July 1987).
32. Studier, J. A. & Keppler, K. J. A note on the neighbor-joining algorithm of Saitou and Nei. *Molecular Biology and Evolution* **5,** 729–731. ISSN: 0737-4038. https://academic.oup.com/mbe/article/5/6/729/1044339 (July 1988).
33. Beroukhim, R. *et al.* The landscape of somatic copy-number alteration across human cancers. *Nature 2010 463:7283* **463,** 899–905. ISSN: 1476-4687. https://www.nature.com/articles/nature08822 (Feb. 2010).
34. Salcedo, A. *et al.* A community effort to create standards for evaluating tumor subclonal reconstruction. *Nature Biotechnology 2020 38:1* **38,** 97–107. ISSN: 1546-1696. https://www.nature.com/articles/s41587-019-0364-z (Jan. 2020).

35. Smolander, J. *et al.* Evaluation of tools for identifying large copy number variations from ultra-low-coverage whole-genome sequencing data. *BMC Genomics 2021 22:1* **22,** 1–15. ISSN: 1471-2164. https://link.springer.com/articles/10.1186/s12864-021-07686-z (May 2021).

36. Poell, J. B. *et al.* ACE: absolute copy number estimation from low-coverage whole-genome sequencing data. *Bioinformatics* **35,** 2847–2849. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/35/16/2847/5265327 (Aug. 2019).

37. Xiao, Y. *et al.* FastClone is a probabilistic tool for deconvoluting tumor heterogeneity in bulk-sequencing samples. *Nature Communications 2020 11:1* **11,** 1–11. ISSN: 2041-1723. https://www.nature.com/articles/s41467-020-18169-2 (Sept. 2020).

38. Kashima, Y. *et al.* Single-cell sequencing techniques from individual to multiomics analyses. *Experimental & Molecular Medicine* **52.** ISSN: 1226-3613. https://doi.org/10.1038/s12276-020-00499-2 (Sept. 2020).

39. Patel, A. P. *et al.* Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science* **344,** 1396–1401. ISSN: 0036-8075. https://science.sciencemag.org/content/344/6190/1396 (June 2014).

40. Tirosh, I. & Suvà, M. L. Deciphering Human Tumor Biology by Single-Cell Expression Profiling. *https://doi.org/10.1146/annurev-cancerbio-030518-055609* **3,** 151–166. https://doi.org/10.1146/annurev-cancerbio-030518-055609 (Mar. 2019).

41. Suvà, M. L. & Tirosh, I. Single-Cell RNA Sequencing in Cancer: Lessons Learned and Emerging Challenges. *Molecular Cell* **75,** 7–12. ISSN: 1097-2765. https://doi.org/10.1016/j.molcel.2019.05.003 (July 2019).

42. Mallory, X. F., Edrisi, M., Navin, N. & Nakhleh, L. Methods for copy number aberration detection from single-cell DNA-sequencing data. *Genome Biology* **21,** 208. ISSN: 1474-760X. https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-02119-8 (Dec. 2020).

43. Shah, S. P. *et al.* Integrating copy number polymorphisms into array CGH analysis using a robust HMM. *Bioinformatics* **22,** e431–e439. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/22/14/e431/228318 (July 2006).

44. Lai, D., Ha, G. & Shah, S. *HMMcopy: Copy number prediction with correction for GC and mappability bias for HTS data* 2019. https://bioconductor.org/packages/release/bioc/html/HMMcopy.html.

45. Nilsen, G. *et al.* Copynumber: Efficient algorithms for single- and multi-track copy number segmentation. *BMC Genomics 2012 13:1* **13,** 1–16. ISSN: 1471-2164. https://bmcgenomics.biomedcentral.com/articles/10.1186/1471-2164-13-591 (Nov. 2012).

46. Olshen, A. B., Venkatraman, E. S., Lucito, R. & Wigler, M. Circular binary segmentation for the analysis of array-based DNA copy number data. *Biostatistics* **5,** 557–572. ISSN: 1465-4644. https://academic.oup.com/biostatistics/article-lookup/doi/10.1093/biostatistics/kxh008 (Oct. 2004).

47. Venkatraman, E. S. & Olshen, A. B. A faster circular binary segmentation algorithm for the analysis of array CGH data. *Bioinformatics* **23,** 657–663. ISSN: 1367-4803. https://doi.org/10.1093/bioinformatics/btl646 (Mar. 2007).

48. Baslan, T. *et al.* Genome-wide copy number analysis of single cells. *Nature Protocols* **7,** 1024–1041. http://www.nature.com/doifinder/10.1038/nprot.2012.039 (May 2012).

49. Bakker, B. *et al.* Single-cell sequencing reveals karyotype heterogeneity in murine and human malignancies. *Genome Biology* **17,** 115. http://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0971-7 (Dec. 2016).

50. Taudt, A. S. *Hidden Markov models for the analysis of next-generation-sequencing data* PhD thesis (University of Groningen, Groningen, Oct. 2018). https://research.rug.nl/en/publications/hidden-markov-models-for-the-analysis-of-next-generation-sequenci.

51. Gao, R. *et al.* Punctuated copy number evolution and clonal stasis in triple-negative breast cancer. *Nature Genetics* **48,** 1119–1130. ISSN: 1061-4036. http://www.nature.com/articles/ng.3641 (Oct. 2016).

52. Kuipers, J., Tuncel, M. A., Ferreira, P., Jahn, K. & Beerenwinkel, N. Single-cell copy number calling and event history reconstruction. *bioRxiv,* 2020.04.28.065755. https://www.biorxiv.org/content/10.1101/2020.04.28.065755v1 (Apr. 2020).

53. 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. ISSN: 1088-9051. https://genome.cshlp.org/content/27/11/1885 (Nov. 2017).

54. Felsenstein, J. Journal of Molecular Evolution Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach. *J Mol Evol* **17,** 368–376. https://link.springer.com/content/pdf/10.1007/BF01734359.pdf (1981).

55. Yang, Z. Maximum-likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. *Molecular biology and evolution* **10,** 1396–1401. ISSN: 0737-4038. https://pubmed.ncbi.nlm.nih.gov/8277861/ (1993).

56. Yang, Z. Maximum Likelihood Phylogenetic Estimation from DNA Sequences with Variable Rates over Sites: Approximate Methods. *J Mol Evol* **39,** 306–314. https://doi.org/10.1007/BF00160154 (1994).

57. Fletcher, R. in *Practical Methods of Optimization* Second, 44–79 (John Wiley & Sons, Ltd, Chichester, West Sussex England, May 2000).

58. Quinlan, A. R. & Hall, I. M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics (Oxford, England)* **26,** 841–2. ISSN: 1367-4811. http://www.ncbi.nlm.nih.gov/pubmed/20110278 (Mar. 2010).

59. Li, Z. *et al.* Comprehensive identification and characterization of somatic copy number alterations in triple-negative breast cancer. *International Journal of Oncology* **56,** 522–530. ISSN: 1019-6439. https://www.spandidos-publications.com/10.3892/ijo.2019.4950 (Feb. 2020).

60. 10x Genomics. *Application Note - Assessing Tumor Heterogeneity with Single Cell CNV* 2018. https://pages.10xgenomics.com/rs/446-PBO-704/images/10x_AN026_SCCNV_Assessing_Tumor%20Heterogeneity_digital.pdf.

61. Dunham, I. *et al.* An integrated encyclopedia of DNA elements in the human genome. *Nature 2012 489:7414* **489,** 57–74. ISSN: 1476-4687. https://www.nature.com/articles/nature11247 (Sept. 2012).

62. Derrien, T. *et al.* Fast Computation and Applications of Genome Mappability. *PLOS ONE* **7,** e30377. ISSN: 1932-6203. https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0030377 (Jan. 2012).

63. Galassi, M. *et al. GNU Scientific Library Reference Manual* 2006. https://www.gnu.org/software/gsl/.

64. Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal* **17,** 10–12. http://journal.embnet.org/index.php/embnetjournal/article/view/200/479 (May 2011).

65. Bolger, A. M., Lohse, M. & Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics (Oxford, England)* **30,** 2114–20. ISSN: 1367-4811. http://www.ncbi.nlm.nih.gov/pubmed/24695404 (Aug. 2014).

66. Schmieder, R. & Edwards, R. Quality control and preprocessing of metagenomic datasets. *Bioinformatics* **27,** 863–864. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/27/6/863/236283 (Mar. 2011).

67. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nature Methods* **9,** 357–359. ISSN: 1548-7091. http://www.nature.com/articles/nmeth.1923 (Apr. 2012).

68. Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics (Oxford, England)* **25,** 2078–9. ISSN: 1367-4811. https://academic.oup.com/bioinformatics/article/25/16/2078/204688 (Aug. 2009).

69. The Broad Institute. *Picard: A set of command line tools (in Java) for manipulating high-throughput sequencing (HTS) data and formats such as SAM/BAM/CRAM and VCF.* 2021. http://broadinstitute.github.io/picard/.

70. Heger, A., Jacobs, K. & et al. *pysam* 2021. https://github.com/pysam-developers/pysam.

71. Cao, S. *et al.* Estimation of tumor cell total mRNA expression in 15 cancer types predicts disease progression. *Nature Biotechnology 2022,* 1–10. ISSN: 1546-1696. https://www.nature.com/articles/s41587-022-01342-x (June 2022).

72. Cai, H. *et al.* WaveDec: A wavelet approach to identify both shared and individual patterns of copy-number variations. *IEEE Transactions on Biomedical Engineering* **65,** 353–364. ISSN: 15582531. https://doi.org/10.1109/tbme.2017.2769677 (Feb. 2018).

73. Schwartz, R. & Schäffer, A. A. The evolution of tumour phylogenetics: principles and practice. *Nature Reviews Genetics 2017 18:4* **18,** 213–229. ISSN: 1471-0064. https://www.nature.com/articles/nrg.2016.170 (Feb. 2017).

74. Schwarz, R. F. *et al.* Phylogenetic Quantification of Intra-tumour Heterogeneity. *PLOS Computational Biology* **10,** e1003535. ISSN: 1553-7358. https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003535 (2014).

75. Cordonnier, G. & Lafond, M. Comparing copy-number profiles under multi-copy amplifications and deletions. *BMC Genomics* **21,** 1–12. ISSN: 14712164. https://link.springer.com/article/10.1186/s12864-020-6611-3 (Apr. 2020).

76. Robinson, D. F. & Foulds, L. R. Comparison of phylogenetic trees. *Mathematical Biosciences* **53,** 131–147. ISSN: 0025-5564. https://doi.org/10.1016/0025-5564(81)90043-2 (Feb. 1981).

77. Baum, L. E. & Petrie, T. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics* **37,** 1554–1563. ISSN: 00034851. http://www.jstor.org/stable/2238772 (1966).

78. Baum, L. E. & Eagon, J. A. An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society* **73,** 360–363. ISSN: 0002-9904. https://doi.org/10.1090/S0002-9904-1967-11751-8 (1967).

79. Baum, L. E., Petrie, T., Soules, G. & Weiss, N. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics* **41,** 164–171. ISSN: 00034851. http://www.jstor.org/stable/2239727 (1970).

80. Wang, X., Chen, H. & Zhang, N. R. DNA copy number profiling using single-cell sequencing. *Briefings in Bioinformatics* **19,** 731–736. ISSN: 1467-5463. https://academic.oup.com/bib/article/19/5/731/2968315 (Sept. 2018).

81. Wang, R., Lin, D. Y. & Jiang, Y. SCOPE: A Normalization and Copy-Number Estimation Method for Single-Cell DNA Sequencing. *Cell Systems* **10,** 445–452. ISSN: 2405-4712. https://doi.org/10.1016/j.cels.2020.03.005 (May 2020).

82. Lähnemann, D. *et al.* Eleven grand challenges in single-cell data science. *Genome Biology 2020 21:1* **21,** 1–35. ISSN: 1474-760X. https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-1926-6 (Feb. 2020).

83. Casasent, A. K. *et al.* Multiclonal Invasion in Breast Tumors Identified by Topographic Single Cell Sequencing. *Cell* **172,** 205–217. ISSN: 0092-8674. https://doi.org/10.1016/j.cell.2017.12.007 (Jan. 2018).

84. Zaccaria, S. & Raphael, B. J. Characterizing allele- and haplotype-specific copy numbers in single cells with CHISEL. *Nature Biotechnology* **39,** 207–214. ISSN: 15461696. https://doi.org/10.1038/s41587-020-0661-6 (Feb. 2021).

85. Rabiner, L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE* **77,** 257–286. ISSN: 15582256. http://dx.doi.org/10.1109/5.18626 (1989).

86. Viterbi, A. J. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory* **13,** 260–269. ISSN: 15579654. https://doi.org/10.1109/TIT.1967.1054010 (1967).

87. Forney, G. D. The Viterbi Algorithm. *Proceedings of the IEEE* **61,** 268–278. ISSN: 15582256. http://dx.doi.org/10.1109/PROC.1973.9030 (1973).

88. R Core Team. *R: A Language and Environment for Statistical Computing* Vienna, Austria, 2021. https://www.R-project.org/.

89. Paradis, E. & Schliep, K. ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. *Bioinformatics* **35,** 526–528. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/35/3/526/5055127 (Feb. 2019).

90. Schliep, K. P. phangorn: phylogenetic analysis in R. *Bioinformatics* **27,** 592–593. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/27/4/592/198887 (Feb. 2011).

91. Schliep, K., Potts, A. J., Morrison, D. A. & Grimm, G. W. Intertwining phylogenetic trees and networks. *Methods in Ecology and Evolution* **8,** 1212–1220. ISSN: 2041-210X. https://onlinelibrary.wiley.com/doi/full/10.1111/2041-210X.12760 (Oct. 2017).

92. Wilke, C. O. *cowplot: Streamlined Plot Theme and Plot Annotations for 'ggplot2'* 2020. https://CRAN.R-project.org/package=cowplot.

93. Wickham, H. *ggplot2: Elegant Graphics for Data Analysis* 2016. https://ggplot2.tidyverse.org.

94. Yu, G., Smith, D. K., Zhu, H., Guan, Y. & Lam, T. T. Y. ggtree: an r package for visualization and annotation of phylogenetic trees with their covariates and other associated data. *Methods in Ecology and Evolution* **8,** 28–36. ISSN: 2041-210X. https://onlinelibrary.wiley.com/doi/full/10.1111/2041-210X.12628 (Jan. 2017).

95. Yu, G., Lam, T. T. Y., Zhu, H. & Guan, Y. Two Methods for Mapping and Visualizing Associated Data on Phylogeny Using Ggtree. *Molecular Biology and Evolution* **35,** 3041–3043. ISSN: 0737-4038. https://academic.oup.com/mbe/article/35/12/3041/5142656 (Dec. 2018).

96. Yu, G. Using ggtree to Visualize Data on Tree-Like Structures. *Current Protocols in Bioinformatics* **69,** e96. ISSN: 1934-340X. https://onlinelibrary.wiley.com/doi/full/10.1002/cpbi.96 (Mar. 2020).

97. Warnes, G. R., Bolker, B. & Lumley, T. *gtools: Various R Programming Tools* 2021. https://CRAN.R-project.org/package=gtools.

98. Wickham, H. The Split-Apply-Combine Strategy for Data Analysis. *Journal of Statistical Software* **40,** 1–29. ISSN: 1548-7660. https://www.jstatsoft.org/index.php/jss/article/view/v040i01 (Apr. 2011).

99. Wickham, H. Reshaping Data with the reshape Package. *Journal of Statistical Software* **21,** 1–20. ISSN: 1548-7660. https://www.jstatsoft.org/index.php/jss/article/view/v021i12 (Nov. 2007).

100. Wickham, H. & Seidel, D. *scales: Scale Functions for Visualization* 2020. https://CRAN.R-project.org/package=scales.

101. Wickham, H. *stringr: Simple, Consistent Wrappers for Common String Operations* 2019. https://CRAN.R-project.org/package=stringr.

102. Oesper, L., Mahmoody, A. & Raphael, B. J. THetA: inferring intra-tumor heterogeneity from high-throughput DNA sequencing data. *Genome Biology* **14,** R80. ISSN: 1465-6906. http://genomebiology.biomedcentral.com/articles/10.1186/gb-2013-14-7-r80 (2013).

103. Ha, G. *et al.* TITAN: inference of copy number architectures in clonal cell populations from tumor whole-genome sequence data. *Genome research* **24,** 1881–93. ISSN: 1549-5469. http://www.ncbi.nlm.nih.gov/pubmed/25060187 (Nov. 2014).

104. Roth, A. *et al.* PyClone: statistical inference of clonal population structure in cancer. *Nature Methods* **11,** 396–398. ISSN: 1548-7091. http://www.nature.com/articles/nmeth.2883 (Apr. 2014).

105. Deshwar, A. G. *et al.* PhyloWGS: Reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. *Genome Biology* **16,** 1–20. ISSN: 1474760X. https://genomebiology.biomedcentral.com/articles/10.1186/s13059-015-0602-8 (Feb. 2015).

106. Urrutia, E., Chen, H., Zhou, Z., Zhang, N. R. & Jiang, Y. Integrative pipeline for profiling DNA copy number and inferring tumor phylogeny. *Bioinformatics* **34,** 2126–2128. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/34/12/2126/4838234 (June 2018).

107. Zaccaria, S., El-Kebir, M., Klau, G. W. & Raphael, B. J. Phylogenetic Copy-Number Factorization of Multiple Tumor Samples. *https://home.liebertpub.com/cmb* **25,** 689–708. ISSN: 10665277. https://www.liebertpub.com/doi/10.1089/cmb.2017.0253 (July 2018).

108. Satas, G., Zaccaria, S., El-Kebir, M. & Raphael, B. J. DeCiFering the elusive cancer cell fraction in tumor heterogeneity and evolution. *Cell Systems* **12,** 1004–1018. ISSN: 2405-4712. https://pubmed.ncbi.nlm.nih.gov/34416171/ (Oct. 2021).

109. Hou, Y. *et al.* Comparison of variations detection between whole-genome amplification methods used in single-cell resequencing. *GigaScience* **4.** ISSN: 2047217X. https://academic.oup.com/gigascience/article/4/1/s13742-015-0068-3/2707556 (Dec. 2015).

110. Chen, D. Y. *et al.* Comparison of single cell sequencing data between two whole genome amplification methods on two sequencing platforms. *Scientific Reports 2018 8:1* **8,** 1–11. ISSN: 2045-2322. https://www.nature.com/articles/s41598-018-23325-2 (Mar. 2018).

111. Nielsen, R., Paul, J. S., Albrechtsen, A. & Song, Y. S. Genotype and SNP calling from next-generation sequencing data. *Nature Reviews Genetics 2011 12:6* **12,** 443–451. ISSN: 1471-0064. https://www.nature.com/articles/nrg2986 (May 2011).

112. Zafar, H., Wang, Y., Nakhleh, L., Navin, N. & Chen, K. Monovar: single-nucleotide variant detection in single cells. *Nature Methods 2016 13:6* **13,** 505–507. ISSN: 1548-7105. https://www.nature.com/articles/nmeth.3835 (Apr. 2016).

113. Garvin, T. *et al.* Interactive analysis and assessment of single-cell copy-number variations. *Nature Methods* **12,** 1058–1060. ISSN: 1548-7091. http://www.nature.com/articles/nmeth.3578 (Nov. 2015).

114. Jahn, K., Kuipers, J. & Beerenwinkel, N. Tree inference for single-cell data. *Genome Biology* **17,** 1–17. ISSN: 1474760X. https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0936-x (May 2016).

115. Ross, E. M. & Markowetz, F. OncoNEM: Inferring tumor evolution from single-cell sequencing data. *Genome Biology* **17,** 1–14. ISSN: 1474760X. https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-0929-9 (Apr. 2016).

116. Zafar, H., Tzen, A., Navin, N., Chen, K. & Nakhleh, L. SiFit: inferring tumor trees from single-cell sequencing data under finite-sites models. *Genome Biology* **18,** 178. ISSN: 1474-760X. http://genomebiology.biomedcentral.com/articles/10.1186/s13059-017-1311-2 (Dec. 2017).

117. Singer, J., Kuipers, J., Jahn, K. & Beerenwinkel, N. Single-cell mutation identification via phylogenetic inference. *Nature Communications 2018 9:1* **9,** 1–8. ISSN: 2041-1723. https://www.nature.com/articles/s41467-018-07627-7 (Dec. 2018).

118. Zafar, H., Navin, N., Chen, K. & Nakhleh, L. SiCloneFit: Bayesian inference of population structure, genotype, and phylogeny of tumor clones from single-cell genome sequencing data. *Genome Research* **29,** 1847–1859. ISSN: 1088-9051. https://genome.cshlp.org/content/29/11/1847 (Nov. 2019).

119. Wu, Y. Accurate and efficient cell lineage tree inference from noisy single cell data: the maximum likelihood perfect phylogeny approach. *Bioinformatics* **36,** 742–750. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/36/3/742/5555811 (Feb. 2020).

120. El-Kebir, M. SPhyR: tumor phylogeny estimation from single-cell sequencing data under loss and error. *Bioinformatics* **34,** i671–i679. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/34/17/i671/5093218 (Sept. 2018).

121. Satas, G., Zaccaria, S., Mon, G. & Raphael, B. J. SCARLET: Single-Cell Tumor Phylogeny Inference with Copy-Number Constrained Mutation Losses. *Cell Systems* **10,** 323–332. ISSN: 2405-4712. https://doi.org/10.1016/j.cels.2020.04.001 (Apr. 2020).

122. Chen, Z., Gong, F., Wan, L. & Ma, L. BiTSC2: Bayesian inference of Tumor clonal Tree by joint analysis of Single-Cell SNV and CNA data. *bioRxiv,* 2020.11.30.380949. https://www.biorxiv.org/content/10.1101/2020.11.30.380949v1 (Dec. 2020).

123. Mission Bio. Performance of the Tapestri ® Platform for Single-Cell Targeted DNA Sequencing. https://missionbio.com/wp-content/uploads/2019/10/WhitePaper_MissionBio_TapestriPlatform_RevA.pdf (Oct. 2019).

124. Mission Bio. *What sequencing coverage is recommended?* June 2021. https://support.missionbio.com/hc/en-us/articles/360044243193-What-sequencing-coverage-is-recommended-.

125. Fletcher, R. *Practical Methods of Optimization* Second. https://doi.org/10.1002/9781118723203 (John Wiley & Sons, Ltd, Chichester, West Sussex, England, May 2000).

126. Sollier, E., Kuipers, J., Takahashi, K., Beerenwinkel, N. & Jahn, K. Joint copy number and mutation phylogeny reconstruction from single-cell amplicon sequencing data. *bioRxiv,* 2022.01.06.475205. https://www.biorxiv.org/content/10.1101/2022.01.06.475205v2 (Apr. 2022).

127. Tarabichi, M. *et al.* A practical guide to cancer subclonal reconstruction from DNA sequencing. *Nature Methods 2021 18:2* **18,** 144–155. ISSN: 1548-7105. https://www.nature.com/articles/s41592-020-01013-2 (Jan. 2021).

128. Ruan, Y., Wang, H., Chen, B., Wen, H. & Wu, C. I. Mutations Beget More Mutations—Rapid Evolution of Mutation Rate in Response to the Risk of Runaway Accumulation. *Molecular Biology and Evolution* **37,** 1007–1019. ISSN: 0737-4038. https://academic.oup.com/mbe/article/37/4/1007/5645182 (Apr. 2020).

129. Curtis, C. *et al.* The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature 2012 486:7403* **486,** 346–352. ISSN: 1476-4687. https://www.nature.com/articles/nature10983 (Apr. 2012).

130. Chen, C., Qi, H., Shen, Y., Pickrell, J. & Przeworski, M. Contrasting determinants of mutation rates in Germline and Soma. *Genetics* **207,** 255–267. ISSN: 19432631. https://academic.oup.com/genetics/article/207/1/255/5930699 (Sept. 2017).

131. Beerenwinkel, N., Schwarz, R. F., Gerstung, M. & Markowetz, F. Cancer Evolution: Mathematical Models and Computational Inference. *Systematic Biology* **64,** e1–e25. ISSN: 1063-5157. https://academic.oup.com/sysbio/article/64/1/e1/2848310 (Jan. 2015).

132. Neph, S. *et al.* BEDOPS: high-performance genomic feature operations. *Bioinformatics* **28,** 1919–1920. ISSN: 1367-4803. https://academic.oup.com/bioinformatics/article/28/14/1919/218826 (July 2012).

133. Nelder, J. A. & Mead, R. A Simplex Method for Function Minimization. *The Computer Journal* **7,** 308–313. ISSN: 0010-4620. https://academic-oup-com.libproxy.berkeley.edu/comjnl/article/7/4/308/354237 (Jan. 1965).

134. Sherry, S. T. *et al.* dbSNP: the NCBI database of genetic variation. *Nucleic Acids Research* **29,** 308–311. ISSN: 0305-1048. https://academic.oup.com/nar/article/29/1/308/1116004 (Jan. 2001).

135. Brent, R. P. *Algorithms for Minimization without Derivatives* (ed Forsythe, G.) ISBN: 0130223352 (Prentice-Hall, Englewood Cliffs, New Jersey, 1973).