Haplotype-aware segmentation dramatically increases accuracy of homologue-specific somatic copy number calling

Oliver Priebe[1,2,4], Conor Messer[1], Claudia Chu[1], Julian Hess[1*], Gad Getz[1,2,3*]

[1]Broad Institute of Massachusetts Institute of Technology and Harvard, Cambridge, 02142, MA, USA.
[2]Cancer Center and Department of Pathology, Mass General Hospital, Boston, 02115, MA, USA.
[3]Harvard Medical School, Boston, 02115, MA, USA.
[4]Current Address: Serinus Biosciences, New York, 10016, NY, USA.

*Corresponding author(s). E-mail(s): jhess@broadinstitute.org; gadgetz@broadinstitute.org;
Contributing authors: opriebe@broadinstitute.org; cmesser@broadinstitute.org; cchu@broadinstitute.org;

Abstract

Reconstruction of tumor evolutionary phylogenies is essential for understanding mechanisms of resistance and metastasis in cancer. Phylogenetic inference methods fundamentally require accurate estimates of subclonal homologue-specific absolute copy number (CN), which are used to predict cancer cell fractions of mutations and deconvolve tumor evolution. Highly polyclonal metastatic samples have incredibly complex karyotypes, and calls from current CN calling methods are not accurate enough for phylogenetic reconstructions. Some methods increase accuracy by using a large panel of normal (PoN) samples closely matched to the tumor sample type, but such large PoN samples are unavailable for samples preserved by the increasingly common formalin-fixed, paraffin-embedded (FFPE) fixation method. Here, HapASeg leverages haplotype phasing information to greatly increase accuracy in calling homologue-specific CN, outperforming current methods at all karyotypic complexities, on all sample types (WGS, WES, fresh-frozen, and FFPE) without relying on any PoN correction.

Keywords: Cancer Genomics, sCNA, FFPE

1 Main

The deadliest and least understood aspect of cancer is its clonal evolution under selective pressure. Although we have a comprehensive understanding of growth mechanisms of primary tumors and thus have a wide range of first-line therapies, many of these therapies are only efficacious initially, quickly becoming ineffective as they select for resistant subpopulations of cells, ultimately yielding a completely resistant and thus incurable tumor.

Little is known about the driving mechanisms by which these selective pressures shape the evolution of the tumor. In order to find drivers of clonal evolution, we must first be able to reconstruct the clonal phylogenies within each patient's tumors and metastases. Doing this from bulk DNA sequencing data is a complex process that requires decomposing multiple tumor samples from a patient into their individual clonal cell populations, and then assigning those clones to branches in a tree.

One of the main prerequisites to clonal decomposition from bulk sequencing data is accurate characterization of allele-specific somatic copy number alterations (sCNAs) at the subclonal level. Current sCNA methods are sufficient for calling events in simple karyotypes for relatively pure, fresh frozen samples (Fig. 2d). However, in samples that are more complex, contain low tumor purity, or are formalin-fixed, paraffin-embedded (FFPE), current sCNA methods are inaccurate (Fig. 1c-f). Many studies of resistance and metastasis are retrospective, relying on archived FFPE samples, which is much more cost-effective than cryopreservation [1]. sCNA methods that perform poorly on FFPE samples severely limit the scope of tumor evolutionary studies.

We thus developed HapASeg, a novel allelic sCNA caller that we qualitatively and quantitatively show to rectify the inaccuracies of current state-of-the-art methods, especially in FFPE samples, yielding sCNA profiles of requisite high accuracy for subsequent clonal decomposition, enabling comprehensive future study of resistance and metastases.

1.1 Overview of sCNA calling

An allele-specific sCNA caller uses high-throughput sequencing data to call contiguous genomic regions (i.e. segments) of the same allelic copy ratio (ACR), the proportion of DNA at a given locus coming from either the maternal or paternal homologue, relative to the total genomic mass of the sample. This is computed by multiplying total copy ratio (TCR) — the proportion of the total number of DNA molecules at the locus — and allelic imbalance — the fraction of those molecules from the maternal homologue versus the paternal homologue. Ideally, all called segments should correspond to true copy number alterations, with no spurious segments originating from noise in the sample preparation and sequencing process.

Allelic sCNA calling methods infer TCR from the total number of tumor sequencing reads aligned to a genomic interval spanning the SNP site (sequencing coverage), and allelic imbalance (AI) from the fraction of non-reference reads in the tumor at germline sites heterozygous in the normal (Alternate Allele Fraction, "AAF"). These methods segment both the AAF and coverage, combining the two into a joint allelic copy ratio segmentation. Most sCNA callers first segment the coverage to obtain a

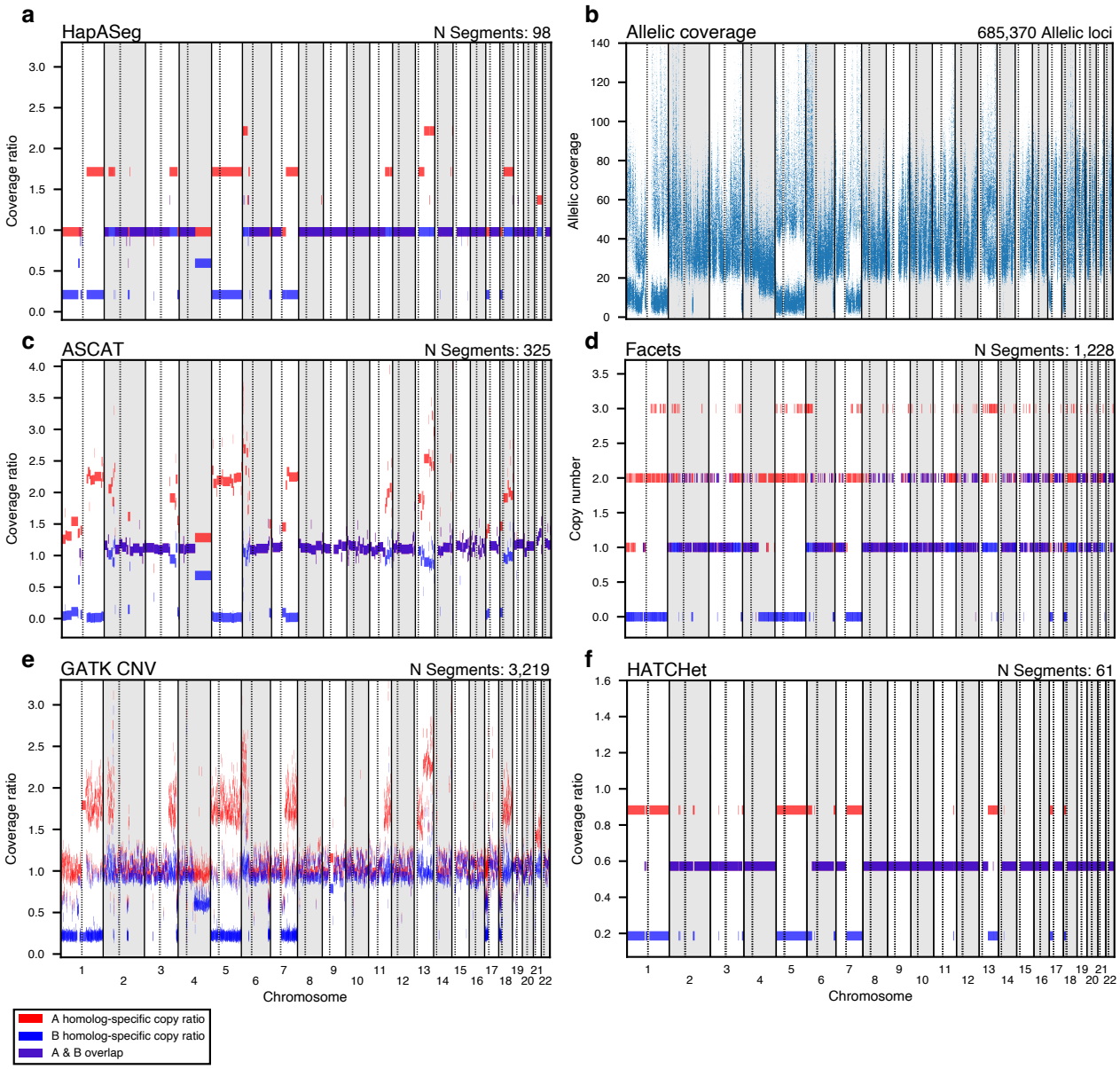

Fig. 1 FFPE samples present difficulties for current state-of-the-art somatic CNA calling methods. Comparison of raw data and sCNA caller results on whole genome sequencing of FFPE Richter's transformed CLL sample CH1003. **b**, Allelic coverage for sample CH1003. For each 2 kilobase genomic interval containing at least one confidently heterozygous SNP, the allelic coverage is computed by multiplying the coverage observed in the interval by the Alternate Allele Frequency (and 1-AAF) of the heterozygous SNPs observed in the interval. **c-f**, Segmentation results for four widely used sCNA methods: ASCAT, Facets, GATK CNV, and HATCHet. Minor allele segments colored in blue, major allele segments in red. **a**, Somatic CNA segmentation results from HapAseg. The ground truth copy number profile is not known for this sample, meaning comparisons in method effectiveness must be qualitative.

TCR profile, then look at AAF within each TCR segment for evidence of AI segments. However, TCR segmentation is confounded by systematic fluctuations in sequencing coverage due to both biological and technical confounders. For example, early replicating regions have higher coverage than late replicating regions [2]; exceptionally GC rich or GC poor regions have lower coverage than regions of intermediate GC content [3]. Fluctuations exceeding the model's expected uncertainty will produce false positive segments (oversegmentation). Some methods [4–7] attempt to regress out fluctuations via covariates like replication timing and GC content, but cannot handle variability not predicted by these covariates. Furthermore, these methods have difficulty fitting the regression model when there is high variability in the coverage signal due to an abundance of sCNA events.

Other methods rely on a panel of normal samples (PoN) to empirically model these fluctuations, assuming that in a normal diploid sample they are solely due to noise [6, 8–10]. PoN-based methods have the advantage of not needing to fit any regression parameters, since the noise profiles of the PoN samples have the same scale as the noise of the tumor, but this approach is only applicable if the characteristics of the PoN samples are closely matched to those of the tumor sample. In particular, an effective PoN should match the sequencing technology, library type (PCR-free/PCR-plus whole genome, whole exome bait set), sample type (fresh frozen, FFPE, cell-free DNA) of the tumor cohort under investigation. Assembling a PoN that matches all of these qualities is not always feasible, nor even always possible. For example, as of this writing, there is no publicly available set of whole genome FFPE normal samples.

Even if an FFPE PoN existed, it is not guaranteed to be effective at denoising. Formalin crosslinks chromatin and DNA [11], hampering the ability to purify it for sequencing at chromatin-rich loci, and thereby causing sequencing coverage in FFPE tumors to vary as a function of tumor-specific chromatin state, which may differ dramatically from normals in the PoN. In order to effectively denoise FFPE coverage, it must be robustly regressed out without a PoN.

1.2 sCNA detection with HapASeg

HapASeg combines several innovative techniques into a novel method for sCNA detection that delivers accurate sCNA detection in FFPE and fresh frozen samples without the need for a PoN. First, rather than using inherently noisy TCR segmentation as its primary signal, HapASeg first obtains an AI segmentation profile by segmenting AAF, a cleaner signal, since variabilities in total sequencing depth affect both alleles equally. Indeed, the two known AAF-specific factors, capture bait bias and aligner bias, cause negligible biases towards the reference allele ($< 3\%$, $< 1\%$, respectively) that can easily be corrected for.

HapASeg uses the clean AI segmentation profile to initialize the TCR segmentation. This is highly advantageous, as breakpoints of AI segments coincide almost perfectly with breakpoints of TCR segments, allowing HapASeg's regression to robustly model coverage variability from true sCNA events. Once coverage has been adequately denoised, HapASeg performs coverage segmentation within each AI segment to infer TCR, and then combines the two signals to compute allele-specific copy ratio.

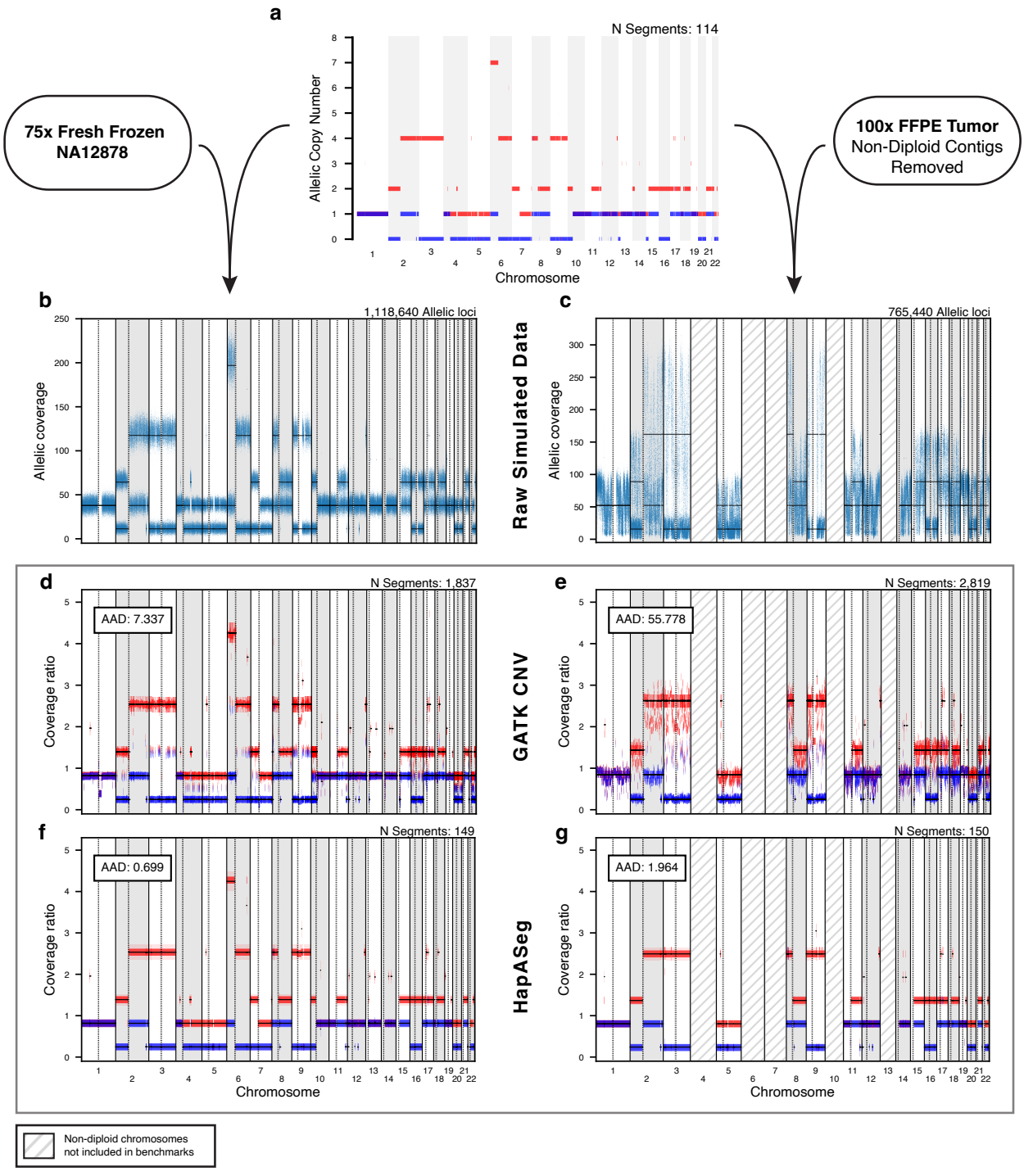

Fig. 2 Simulating tumor samples with known karyotypes enables quantitative benchmarking of sCNA methods. a, Simulated tumor karyotype. **b**, Simulated allelic coverage derived from PCR-free whole genome sequencing of NA12878 and the simulated karyotype from a. **c**, Simulated allelic coverage derived from whole genome sequencing of Richter's sample CH1022 and the simulated karyotype from a. Only chromosomes that are confidently diploid were used for the simulations (non-diploid chromosomes indicated with hatches). **d,e**, sCNA results from GATK CNV on the simulated fresh frozen and FFPE data, respectfully. Segments called by the method in dark red and blue. Ground truth copy levels for each allele derived from the simulated karyotype and preservation specific sequencing data are depicted with a black line at the mean and light boxes denoting the 95% confidence interval. Average Absolute Difference (AAD) between the segments called by the method and the ground truth in addition to the number of segments are listed above. **f,g**, sCNA results from HapASeg on the simulated fresh frozen and FFPE data, respectfully.

HapASeg uses a variety of covariates in this regression, many of which are well-known, e.g., GC content, replication timing, WES target lengths. To address FFPE-induced noise, HapASeg uniquely uses coverage from Formaldehyde Assisted Isolation of Regulatory Elements (FAIRE) sequencing performed on a variety of normal and cancer cell lines [12]. FAIRE-seq uses formalin to crosslink DNA with bound chromatin and then sequences only accessible DNA, which emulates the same coverage irregularities observed in FFPE tumor samples. HapASeg's robust regression combined with its unique covariates not only obviates the need for a PoN, but actually outperforms it, since FAIRE profiles include a variety of tissue types whose chromatin profiles more closely resemble those found in tumors. HapASeg is also much easier to run relative to PoN-based methods, since PoN generation can be a highly involved process at all steps, from sourcing suitable normal samples to computationally assembling them into a panel.

Since HapASeg relies on an accurate AI segmentation, it leverages germline SNP phasing to increase its AI segmentation power. Most other methods perform allelic segmentation on the raw AAF f, however, there is a 50% chance that the non-reference allele lies on the A homolog versus the B homolog, meaning raw AAF values will cluster around two values, f_{A} and $f_{\mathrm{B}} = 1 - f_{\mathrm{A}}$. Methods segmenting raw AAF must therefore fit a constrained two component mixture model to each AI segment, halving the statistical power to discern AI segments relative to a single component model. Although this poses little issue when the statistical noise around f is less than the difference between components, it becomes problematic when noise obscures the two components, as is the case for low coverage samples, clonal sCNAs in a low purity sample, and/or subclonal sCNAs with low cancer cell fraction (CCF). Thus, rather than segmenting the AAF, HapASeg uses imputed SNP phasing [13] to obtain a priori knowledge of which reads support the A homolog, whether those reads be reference or non-reference, thereby allowing it to fit a single component model, alleviating the aforementioned shortcomings.

2 Results

2.1 Qualitative Results

We first qualitatively illustrate the issues plaguing current sCNA calling methods on FFPE tumor samples. We applied HapASeg and four other widely-used sCNA methods (GATK4-CNV [6], ASCAT [4], FACETS [14], and HATCHet [15]) to a set of 16 FFPE Richter's transformed CLL samples [16, 17] of various states of degradation and tumor purities. We found that only HapASeg produced viable sCNA calls; other methods' calls were either over-segmented (Fig. 1c, 1e) or often missed alterations (Fig. 1d, 1f) that were evident in the raw allelic coverage data (Fig. 1b). These error modes were particularly distinct in samples with higher levels of degradation, but were present in even the high quality samples (Supplementary Fig. 1). HapASeg's calls are biologically reasonable: most allelic copy ratio segments occur at uniformly spaced levels, corresponding to clonal integer copy states; segments occuring between the clonal components (e.g. chr4q, chr21q) occur at consistent distances away from the clonal integer levels, indicating they belong to the same subclone. Although this

qualitatively illustrates that HapASeg performs well across a variety of real FFPE samples versus other callers that produce clearly unusable results, we cannot make any quantitative judgments with respect to accuracy due to the lack of a ground truth. We thus developed a robust simulation-based benchmarking platform to compare the accuracy of HapASeg to competing methods.

2.2 Quantitative benchmarking

Many sCNA detection methods have been developed, but few attempts have been made to systematically assess their performance across the range of karyotype complexities, sample preparations, and other sequencing modalities. This is primarily due to the fact that there does not exist a set of tumor samples with known ground truth copy number profiles at all possible event sizes and subclonal fractions. Thus, benchmarking approaches that use real tumor samples can merely either i) qualitatively assess relative differences between methods (Sequenza [5], CNV Radar [10], SEQC-II [18]) and cannot quantitatively benchmark against a comprehensive ground truth, or ii) only assess performance on a limited subset of the karyotype assayed by orthogonal means (e.g., FISH [ASCAT [4], CNV Radar [10]]; or ddPCR; [Falcon [19]]). On the other hand, benchmarking simulated tumors yields a perfect ground truth for karyotypes of arbitrary complexity. This not only allows us to absolutely quantify a method's performance at every single genomic locus, but also allows us to stratify our assessment based on event type, e.g., subclonal focal events. Previous approaches using simulated tumors have failed to robustly assess sCNA calling accurately due to a reliance on overly simplistic models in their simulations, which fail to capture the noise and biases of real sequencing data (HATCHet [15], CloneHD [20]), or have been limited in the number and types of events that they could simulate (TITAN [7], Falcon [19], ReMiXT [21]). To better evaluate the accuracy of HapASeg with respect to other state of the art methods, we developed a new simulation-based benchmarking approach called CNV-Suite, which addresses the previous pitfalls.

CNV-Suite can simulate any karyotype, defined by a set of allele and clone-specific copy number events (e.g., gains, losses, whole-genome doubling, loss-of-heterozygosity, chromothripsis). Recall that there are two primary signals that allele-specific sCNA detection algorithms leverage from sequencing data: the AI at germline heterozygous variant sites in the tumor and the TCR inferred from tumor sequencing coverage. CNV-Suite simulates both of these components by taking known diploid samples (both fresh-frozen and FFPE) and directly scaling their real allele counts and read coverage according to the synthetic tumor karyotype and desired purity (Fig. 4; extended methods). Crucially, this approach maintains the intrinsic noise of the sequencing data, making the simulated tumors indistinguishable from real tumors with the same sample characteristics as their underlying diploid progenitors. The simulated tumor allele count and read coverage data can then be passed directly to each method under evaluation; its results can then be directly compared to the ground truth.

We used CNV-Suite to create a comprehensive benchmarking panel composed of 50 synthetic tumor karyotypes, each consisting of up to three subclones and a randomized selection of copy events. Event lengths were randomly sampled from the reported distribution of sCNAs found in real cancers [22] (See Methods for details).

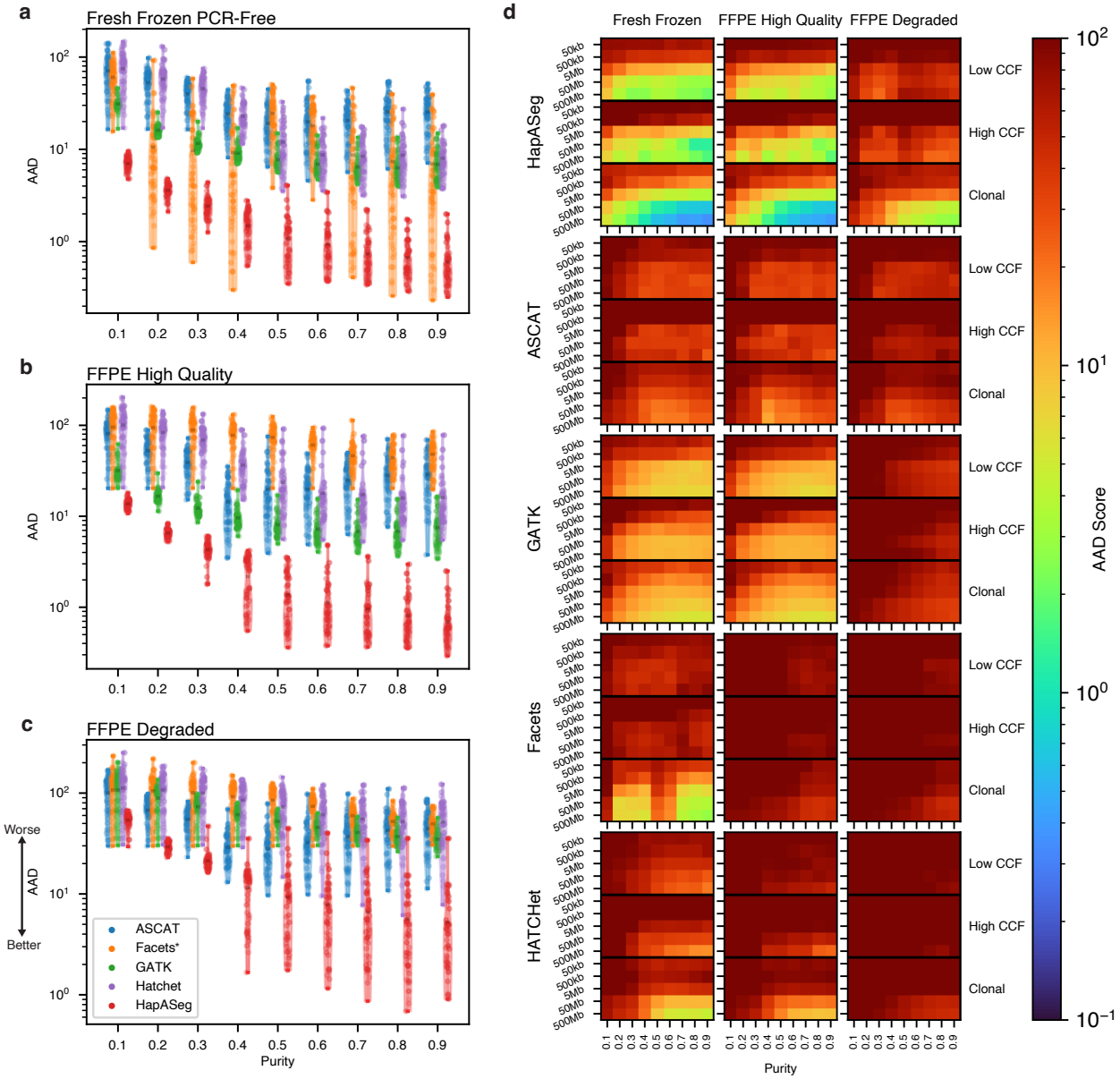

Fig. 3 Quantitative benchmarking panel results a-c, Violin plots for each of the three sample preparation modalities comparing the AAD scores for the five sCNA methods analyzed across the 50 simulated tumor karyotypes in the comprehensive benchmarking panel (See Supplementary Fig. 2 for simulated karyotypes). Note the y-axis for AAD is in log scale. Dashes within violin plots denote the mean and range. **d**, AAD heatmap examines the accuracy of the sCNA methods stratified by purity, ground truth event length and the cancer cell fraction (CCF) of the simulated CNA segments (clonal: ccf=1, low: ccf < 0.7, high: ccf ≥ 0.7). *Facets only returns absolute copy number results, which are used here for computing AAD scores.

The number of events and the clone for each event were also chosen randomly in each sample, resulting in a total of 4,637 simulated sCNA events across a range of simulated karyotype complexities (Extended Data Figure 1). We then simulated tumor coverage/allele counts for each karyotype using three different underlying progenitor samples — fresh-frozen, high-quality FFPE, and highly degraded FFPE, at purities from 10%-90% in 10% increments, for a total of 50 x 3 x 9 = 1,350 simulated tumors.

To measure the accuracy of a model's sCNA calls to the ground truth, we compute the average absolute distance (AAD) between the allelic copy ratio for each sCNA segment generated by the method against the allelic copy ratio in the ground truth karyotype, with each distance weighted by the segment length. Since the units and magnitude of the output segment copy levels vary between each method, and also vary within each method across purities, we rescale all of the results to a common reference before calculating AAD scores (S.1, Methods).

This allowed us to dissect the accuracy of each model for each sample type by ground truth event length and clonal cell fraction (Fig. 3d). HapASeg was able to consistently outperform all other methods for both fresh frozen and FFPE samples across the entire range of purities, event sizes, and clonal fractions, often by an order of magnitude (Fig. 3a-c). Some methods such as FACETS aggressively smooth the segmentation profile, which can result in higher scores for certain simulated samples, but is detrimental on average. HapASeg particularly excelled relative to other methods for the high-quality FFPE sample, which will facilitate future studies leveraging the enormous archives of paraffin slides that have thus far remained largely inaccessible for precise genomic characterization. We also benchmarked simulated WES tumors using the same panel and found that HapAseg had similar accuracy to other methods, suggesting that current methods have reached a theoretical limit of detection with this sequencing technique (Fig. S.2).

3 Discussion

In this work we present HapASeg, a novel homologue-specific sCNA caller that leverages genomic covariates and haplotype phasing to accurately model noisy samples with complex karyotypes without requiring a PoN. We demonstrate HapASeg's superior sCNA calling ability both qualitatively, using FFPE Richter's transformed CLL tumors, and quantitatively, via CNV-Suite, an extensive open source benchmarking platform that we developed for comparing sCNA callers with simulated tumors. In our simulation experiments we found that HapASeg sCNA calls were more accurate than the four state of the art methods we benchmarked against, often by an order of magnitude. We demonstrated that the improved sCNA calls for HapASeg were consistent across sample purity, event lengths and sequencing modality, suggesting that HapASeg can offer enhanced resolution of the landscape of copy number alterations. The improved calling accuracy of HapASeg in FFPE samples is also poised to unlock a vast supply of archived tumor samples which had previously inscrutable sCNA profiles. These newly unlocked samples, combined with the increased accuracy of sCNA calls in other sequencing types, will enable tumor phylogenetic reconstruction at unprecedented precision, offering a new lens to understand mechanisms of metastasis and resistance in cancer.

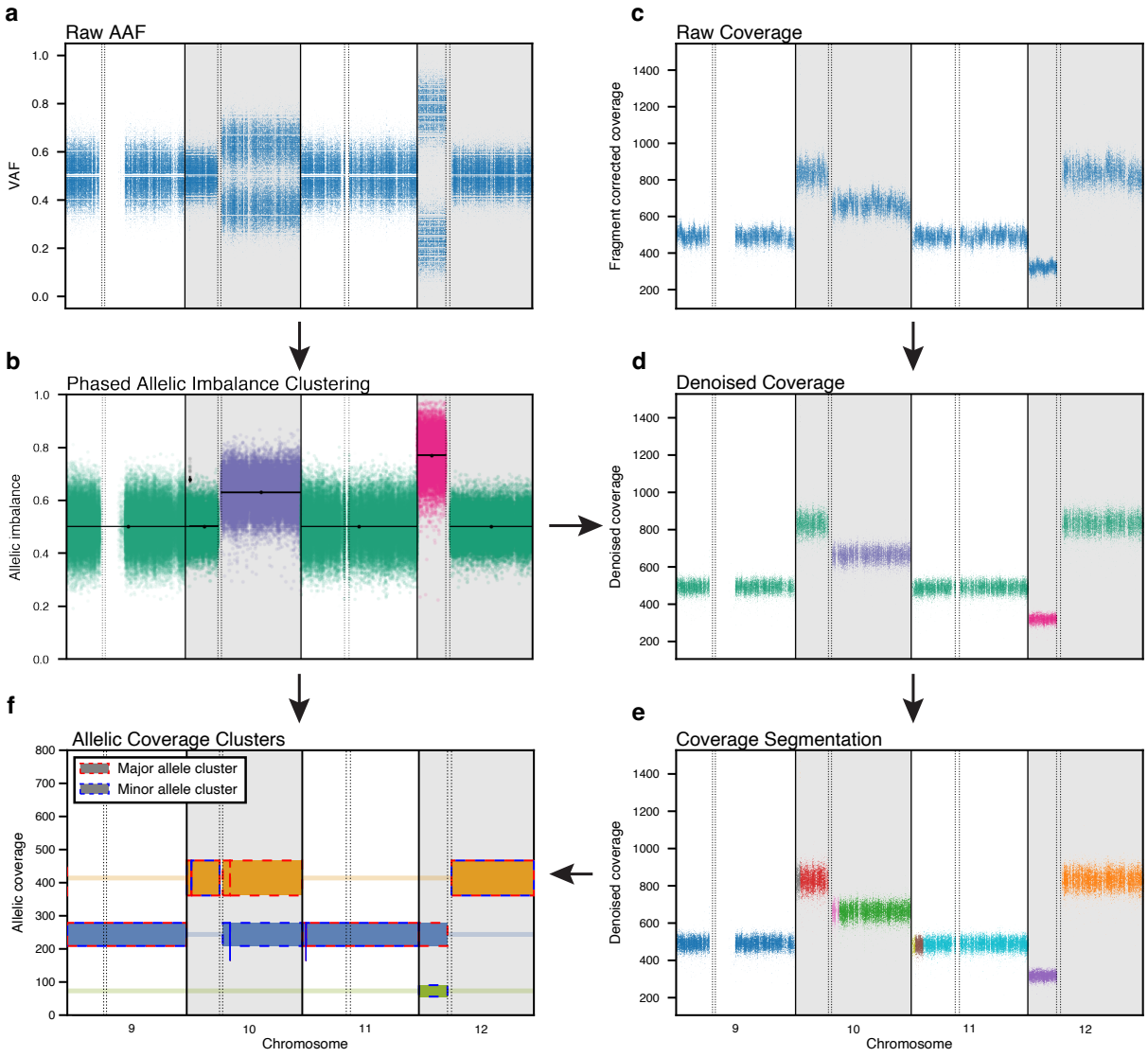

Fig. 4 HapAseg method schematic. **a**, Allelic imbalance is computed at confidently heterozygous SNP sites. **b**, Phased allelic imbalance is segmented and clustered across the genome. SNP sites are colored by allelic imbalance cluster. **c**, Raw fragment length corrected coverage is computed in genomic intervals. **d**, Initial coverage segments are created using the breakpoints inferred by the allelic imbalance clusters and coverage covariates are applied to regress out known biases in coverage. The color of each point corresponds to the allelic cluster that it overlaps. **e**, corrected coverage with each initial coverage segment is further segmented using the log-normal poisson model. **f**, Allelic imbalance segments and coverage segments are combined to create final, allele-specific coverage segments. Each allelic coverage cluster is depicted with colored rectangles, with heights corresponding to the inferred variances.

4 Methods

4.1 AI Segmentation

4.1.1 SNP processing

For each SNP coordinate in a reference panel of phased germline SNPs (e.g., 1000 Genomes [23]), we extract the read count pileups from the tumor/normal alignments for each allele at that coordinate in the panel. We use MuTect [24] for this purpose, since it is efficient, has a robust set of read filters, and most importantly, counts alleles on the fragment level (i.e., read pair level), rather than the single read level. This is critical for sample types with short fragments where reads in a pair may overlap (e.g., FFPE or ctDNA); double counting alleles in overlapping reads will distort downstream results.

We genotype whether a site is heterozygous in the normal sample via the following procedure. Let n_{Nalt_i} and n_{Nref_i} be the normal alternate and reference allele counts in the pileup for the ith SNP site. Then, assuming the fraction of alternate reads f is beta distributed, i.e.

$$f \sim \text{Beta}(n_{\text{Nalt}_i} + 1, n_{\text{Nref}_i} + 1),$$

SNP site i is considered heterozygous if the absolute posterior log odds ratio

$$|\log \Pr(f \geq 0.5) - \log \Pr(f \leq 0.5)| < 2.5.$$

2.5 was empirically chosen because it yields a 90% true positive rate for het site genotyping (assuming uniformly distributed Q30 error rates), irrespective of sequencing coverage. Because we only consider SNPs in a curated reference panel (that thus reside in genomic regions free of sequencing artifacts), the false positive rate is vanishingly small.

In order to properly impute phasing, we also must call homozygous SNPs in the normal, since phasing imputation relies on proper genotyping for all SNPs in the sample. Any SNP in the panel confidently close to $f = 1$ is considered homozygous, according to the criterion $\Pr(f \geq 0.95) > 0.7$.

We impute phasing using Eagle2 [13], given the SNP genotypes and the reference panel. Eagle2 assigns each allele in each het site to either the A or B haplotype. Let n_{Talt_i} and n_{Tref_i} be the tumor alt/ref read counts for the ith SNP site, and a_i, b_i the corresponding haplotype-specific tumor read counts for SNP i. If the alternate allele is phased to the A haplotype, then $a_i = n_{\text{Talt}_i}$ and $b_i = n_{\text{Tref}_i}$; if the alternate allele is phased to the B haplotype, then $a_i = n_{\text{Tref}_i}$, $b_i = n_{\text{Talt}_i}$. We define the haplotypic imbalance of SNP i as $f_i = a_i/(a_i + b_i)$.

4.1.2 Segmentation MCMC

Index SNPs ordered by genomic position from $i = 1 \ldots N_s$. Let the interval set $\mathcal{I}_j = [s_j, e_j)$ comprise all SNP indices falling between s_j and $e_j - 1$. Let f_j be the haplotypic imbalance of $\mathcal{I}_j$. Assuming only aleatoric uncertainty (i.e., f_j is a random

variable whose stochasticity comes only from binomial sampling), the likelihood of f_j for interval $\mathcal{I}_j$ is

$$\begin{aligned}\mathcal{L}(f_j|\mathcal{I}_j) &= \prod_{i\in\mathcal{I}_j} f_j^{a_i}(1-f_j)^{b_i} \\ &= f_j^{\sum_{i\in\mathcal{I}_j} a_i}(1-f_j)^{\sum_{i\in\mathcal{I}_j} b_i} \\ &\equiv f_j^{A_j}(1-f_j)^{B_j}\end{aligned}$$

where A_j and B_j are the total read counts respectively assigned to homologs A and B within interval $\mathcal{I}_j$. Marginalizing out f_j yields the marginal likelihood of the interval,

$$\begin{aligned}\mathcal{L}(\mathcal{I}_j) &= \int_0^1 df_j\, f_j^{A_j}(1-f_j)^{B_j} \\ &= \beta(A_j+1, B_j+1).\end{aligned}$$

Finding an optimal segmentation is equivalent to finding an optimal partitioning of all SNPs, i.e. the optimal set of intervals $\{\mathcal{I}_1 = [0, e_1), \mathcal{I}_2 = [s_2, e_2), \ldots, \mathcal{I}_N = [s_N, e_N)\}$, where $s_j = e_{j-1}\forall j$. The likelihood of a given segmentation is the product of the marginal likelihoods of each interval,

$$\begin{aligned}\mathcal{L}(\mathcal{I}_1, \ldots, \mathcal{I}_N) &= \prod_{j=1}^{N} \mathcal{L}(\mathcal{I}_j) \\ &= \prod_{j=1}^{N} \beta(A_j+1, B_j+1).\end{aligned}$$

Unfortunately, for N total SNPs, the total number of possible partitions is 2^N, which is intractably large for $\sim 10^6$ SNPs in a whole genome, or $\sim 10^4$ SNPs in a whole exome.

Luckily, the space of partitions is amenable to stochastic optimization via MCMC sampling. Initially, each SNP belongs to its own segment. At random, we pick two adjacent segments and probabilistically merge them according to the Metropolis criterion:

$$\Pr(\underbrace{[s_j, e_j), [s_{j+1}, e_{j+1})}_{S} \to \underbrace{[s_j, e_{j+1})}_{S^*}) = \min\left\{1, \frac{\mathcal{L}(S^*)q(S|S^*)}{\mathcal{L}(S)q(S^*|S)}\right\}, \tag{4.a}$$

where marginal likelihoods are

$$\mathcal{L}(S^*) = \beta(\underbrace{\textstyle\sum_{\{i|i\in[s_j,e_{j+1})\}} a_i}_{A_{[s_j,e_{j+1})}} + 1, \underbrace{\textstyle\sum_{\{i|i\in[s_j,e_{j+1})\}} b_i}_{B_{[s_j,e_{j+1})}} + 1) \tag{4.b}$$

$$\mathcal{L}(S) = \beta(A_{[s_j,e_j)} + 1, B_{[s_j,e_j)} + 1) \times \beta(A_{[s_{j+1},e_{j+1})} + 1, B_{[s_{j+1},e_{j+1})} + 1). \tag{4.c}$$

We can also split segments of length > 1. Rather than picking a breakpoint within the segment at random, we probabilistically tailor our proposal distribution. For each SNP $k \in [s_j, e_j)$, we compute

$$\mathcal{L}_j(k) = \beta(A_{[s_j,k)} + 1, B_{[s_j,k)} + 1) \times \beta(A_{[k,e_j)} + 1, B_{[k,e_j)} + 1), \tag{4.d}$$

and propose picking k as the breakpoint with probability $p_j(k) = \mathcal{L}_j(k) / \sum_k \mathcal{L}_j(k)$. Our proposal distributions are

$$q(S|S^*) = \frac{p_j(k)}{N} \qquad q(S^*|S) = \frac{1}{N-1}$$

where N is the total number of segments at the current MCMC iteration.

The overall MCMC procedure is:

1. Initialize each SNP to belong to its own segment.
2. With equal probability, pick whether to attempt a merge or split this iteration.
3. If we choose to merge, uniformly pick segment j at random from 1 to $N-1$. Probabilistically merge it with segment $j+1$ with probability defined in (4.a).
4. If we choose to split, uniformly pick j at random from 1 to N, propose a breakpoint within j via (4.d), and accept the proposal with probability defined as the reciprocal of (4.a). Proposing a breakpoint at the end of the segment is equivalent to not accepting any proposal at all and leaving the chain as-is.
5. We run the chain until it has burned-in, which we infer by detecting whether the overall likelihood oscillates around a maximum, rather than monotonically increasing. We then run the chain for another few thousand iterations, and save the maximum likelihood segmentation.

The MCMC is extremely fast, since each iteration only involves computing partial sums in (4.b), (4.c), and (4.d) which can be memoized between iterations. In theory, storing all possible partial sums would require $O(N^2)$ space, but in practice we only ever need blocks close to the diagonal, so the partial sum matrix can be stored sparse, with space complexity $O(\bar{N}_s \bar{B}^2) \ll O(N^2)$, where $\bar{N}_s$ is the average number of segments and $\bar{B}$ the average length of each segment. The MCMC is also efficiently parallelized: the most compute-intensive phase is at the beginning when each SNP belongs to its own segment. Blocks of adjacent SNPs are burned-in in parallel, which are then concatenated post burnin to sample the final segmentation posterior. During this step, each chromosome arm can be run in parallel, since we do not expect reliable phasing across the centromere.

4.1.3 Reference bias correction

The alternate allelic fraction at a SNP site can be biased from an overrepresentation of reference reads, especially in capture-based whole exome sequencing, since baits only target the reference genome and thus preferentially bind fragments supporting the reference. Small amounts of reference bias may also manifest in whole genome sequencing due to aligners' slightly diminished ability to map reads containing mismatches.

We correct for reference bias empirically. Recall that a SNP assigned to haplotype A with haplotypic imbalance $f = a/(a+b)$ means that a is the read count of the alternate allele and b the read count of the reference allele; a SNP assigned to haplotype B has a as the reference count and b as the alternate count. Thus, for SNPs in segment S_j, overall haplotypic imbalance across all SNPs assigned to A is distributed

$$\begin{aligned} f_{\mathrm{A},j} &\sim \mathrm{Beta}(\textstyle\sum_{i \in \mathrm{A} \cap S_j} a_i + 1, \sum_{i \in \mathrm{A} \cap S_j} b_i + 1) \\ &\sim \mathrm{Beta}(\textstyle\sum_{i \in \mathrm{A} \cap S_j} n_{\mathrm{Talt}_i} + 1, \sum_{i \in \mathrm{A} \cap S_j} n_{\mathrm{Tref}_i} + 1). \end{aligned}$$

Similarly, the haplotypic imbalance across all SNPs assigned to haplotype B is

$$\begin{aligned} f_{\mathrm{B},j} &\sim \mathrm{Beta}(\textstyle\sum_{i \in \mathrm{B} \cap S_j} a_i + 1, \sum_{i \in \mathrm{B} \cap S_j} b_i + 1) \\ &\sim \mathrm{Beta}(\textstyle\sum_{i \in \mathrm{B} \cap S_j} n_{\mathrm{Tref}_i} + 1, \sum_{i \in \mathrm{B} \cap S_j} n_{\mathrm{Talt}_i} + 1). \end{aligned}$$

In the absence of reference bias, if we had infinite SNPs assigned to A and B, the absolute difference between f_A and f_B would approach zero; $\lim_{|A|\to\infty,|B|\to\infty} |f_{\mathrm{A}} - f_{\mathrm{B}}| = 0$. In the presence of reference bias, all reference allele counts are scaled by some constant factor $r_b < 1$,

$$\begin{aligned} f_{\mathrm{A},j}(r_b) &\sim \mathrm{Beta}(\textstyle\sum_{i \in \mathrm{A} \cap S_j} n_{\mathrm{Talt}_i} + 1, r_b \sum_{i \in \mathrm{A} \cap S_j} n_{\mathrm{Tref}_i} + 1) \\ f_{\mathrm{B},j}(r_b) &\sim \mathrm{Beta}(r_b \textstyle\sum_{i \in \mathrm{B} \cap S_j} n_{\mathrm{Tref}_i} + 1, \sum_{i \in \mathrm{B} \cap S_j} n_{\mathrm{Talt}_i} + 1). \end{aligned}$$

Immediately after burn-in, we perform a grid search over $r_b \in [0.7, 1]$, and minimize the mean absolute difference across segments, weighted by the number of SNPs per segment, $n(S_j)$:

$$r_b = \arg\min_{r_b} \frac{\sum_j |f_{\mathrm{A},j}(r_b) - f_{\mathrm{B},j}(r_b)| \times n(S_j)}{\sum_j n(S_j)}.$$

We compute the absolute difference via Monte Carlo (i.e., sample the beta distributions and take the mean difference between samples). We then scale all alternate allele counts by r_b, and perform all subsequent allelic segmentation operations with scaled reference allele counts.

4.2 Allelic segmentation refinement

Imputed phasing cannot yield perfectly consistent homolog assignments from telomere-to-telomere, due to the fact that the genotype of an individual being imputed will be a combination of the haplotypes present in the reference panel. Thus, homolog assignments will at best be consistent relative only to a contiguous haplotype block within the imputed individual, and may even experience orientation switches within a haplotype block due to meiotic recombination events. SNP haplotype assignments are generally consistent relative to each other within an average of 14.4Mb, or about 14 centimorgans (see Appendix E for details). This means that a long AI segment will

typically be split into several components alternating between f and $1-f$. We must correct for this phase orientation switching in post-processing ¡TODO: add figure, jointly with next section¿.

To address this, we refine our optimum initial segmentation, jointly performing phase orientation correction and clustering of non-adjacent segments ¡TODO: add figure¿ using a Dirichlet process (DP) MCMC. Clustering non-adjacent segments improves sensitivity by increasing the probability that the model produces extremely focal segments supported by few SNPs if their allelic imbalance is consistent with other non-adjacent segments. We refer to this procedure as the Allelic Dirichlet Process, or ADP.

We initialize each non-phase corrected segment into its own cluster, yielding clusters indexed as $\mathbf{K} = \{1 \dots n_{\text{seg}}\}$. We formulate the MCMC as a Gibbs sampler: at each MCMC iteration, we choose a segment S, unassign it from the cluster it is currently part of, and probabilistically join a currently existing cluster, or open a new cluster, conditioned on the configuration of all other segments. Let A_S and B_S correspond to the total segmental read counts of homologs A and B, and $\tilde{A}_k = \sum_{s \in C_k} A_s$ and $\tilde{B}_k = \sum_{s \in C_k} B_s$ be the total read counts of A and B across all segments assigned assigned to cluster $k \in \mathbf{K}$.

The probability that S joins C_k is proportional to the overall likelihood of the system, which is the product of (i) the marginal likelihood of S combined with C_k,

$$\mathcal{L}(S \cup C_k) = \beta(\tilde{A}_k + A_S + \mathfrak{a} + 1, \tilde{B}_k + B_S + \mathfrak{b} + 1) \tag{4.e}$$

and (ii) the marginal likelihood of every other cluster $\{C_\kappa \mid \kappa \neq k\}$,

$$\mathcal{L}(\{C_\kappa \mid \kappa \neq k\}) = \prod_{\kappa \neq k} \mathcal{L}(C_\kappa) = \prod_{\kappa \neq k} \beta(\tilde{A}_\kappa + \mathfrak{a} + 1, \tilde{B}_\kappa + \mathfrak{b} + 1).$$

This yields

$$\begin{aligned} p(S \cup C_k) &\propto \mathcal{L}(S \cup C_k)\mathcal{L}(\{C_\kappa \mid \kappa \neq k\})p_{\text{DP}}(S \cup C_k) \\ &\propto \frac{\mathcal{L}(S \cup C_k)\mathcal{L}(\{C_\kappa \mid \kappa \in \mathbf{K}\})}{\mathcal{L}(C_k)} p_{\text{DP}}(S \cup C_k), \end{aligned}$$

since $\prod_{\kappa \neq k} \mathcal{L}(C_\kappa) = \prod_{\kappa \in \mathbf{K}} \mathcal{L}(C_\kappa)/\mathcal{L}(C_k)$. The term in the numerator does not depend on k and thus is a constant that can be cancelled, leaving us with

$$\begin{aligned} p(S \cup C_k) &\propto \frac{\mathcal{L}(S \cup C_k)}{\mathcal{L}(C_k)} p_{\text{DP}}(S \cup C_k) \\ &\propto p(S|C_k) p_{\text{DP}}(S \cup C_k), \end{aligned}$$

where $p_{\text{DP}}(S \cup C_k)$ is the DP prior on C_k, which is based on the number of SNPs in both C_k and S. This is an extension of the standard DP prior, since it accounts not only for the number of entities in the cluster being joined but also the number of entities being moved. See Appendix B for details.

S can also create a new cluster ($k = |\mathbf{K}|+1$) using the above formalism, by setting $\tilde{A}_k$ and $\tilde{B}_k$ to 0, i.e.

$$\mathcal{L}(C_{|\mathbf{K}|+1} \cup S) = \beta(A_S + \mathfrak{a} + 1, B_S + \mathfrak{b} + 1)$$
$$\mathcal{L}(C_{|\mathbf{K}|+1}) = \beta(\mathfrak{a} + 1, \mathfrak{b} + 1).$$

4.2.1 Phasing correction

We probabilistically orient each segment's haplotypic imbalance to $f \geq 0.5$; in other words, we want the number of copies assigned to the A haplotype to always be greater or equal to the number of copies assigned to the B haplotype. The probability that a segment's phasing orientation is properly oriented is the probability that the total number of alternate reads assigned to the A haplotype is higher than the total number of alternate reads assigned to the B haplotype. We sum the alt and ref counts across all SNPs in segment S, segregated by whether the SNP is assigned to A or B:

$$A_{\text{alt}} = \sum_{i \in \text{A}} n_{i,\text{alt}} \qquad B_{\text{alt}} = \sum_{i \in \text{B}} n_{i,\text{alt}}$$
$$A_{\text{ref}} = \sum_{i \in \text{A}} n_{i,\text{ref}} \qquad B_{\text{ref}} = \sum_{i \in \text{B}} n_{i,\text{ref}}.$$

The haplotype-specific alternate allele fractions are beta distributed,

$$f_A \sim \text{Beta}(A_{\text{alt}} + \mathfrak{a} + 1, A_{\text{ref}} + \mathfrak{b} + 1)$$
$$f_B \sim \text{Beta}(B_{\text{alt}} + \mathfrak{a} + 1, B_{\text{ref}} + \mathfrak{b} + 1)$$

and thus probability that a segment is oriented "properly" is given by the difference

$$p(\phi) := \Pr[f_A - f_B > 0].$$

When computing the marginal likelihood of joining an existing cluster, we jointly consider the phasing orientation by computing *both* phasing orientations of the segment being moved in (4.e), i.e.,

$$\mathcal{L}(C_k \cup S | \phi = 0) = \beta(\tilde{A}_k + A_S + \mathfrak{a} + 1, \tilde{B}_k + B_S + \mathfrak{b} + 1)$$
$$\mathcal{L}(C_k \cup S | \phi = 1) = \beta(\tilde{A}_k + B_S + \mathfrak{a} + 1, \tilde{B}_k + A_S + \mathfrak{b} + 1),$$

yielding the joint probability

$$p(S, k, \phi) = p(S|C_k, \phi)p(\phi)p(C_k \cup S).$$

At each MCMC iteration, the new cluster assignment k and phasing orientation ϕ for segment S are drawn from the joint posterior

$$p(k, \phi | S) = \frac{p(S|C_k, \phi)p(\phi)p(C_k \cup S)}{\sum_{k \in \mathbf{K}} \sum_{\phi \in \{0,1\}} p(S|C_k, \phi)p(\phi)p(C_k \cup S)}. \tag{4.f}$$

4.2.2 Joint segmentation and clustering

If two adjacent segments join the same cluster, they are subsequently considered as a single segment. This greatly increases the efficiency of the DP to reach an equilibrium state, but can also lead the MCMC down an irreversible path. To mitigate this, we also allow segments to be probabilistically split, in a manner identical to the splitting procedure of the initial segmentation MCMC (4.d). In this case, we enumerate the likelihoods of all possible split points and probabilistically draw accordingly, and then randomly choose either the right or left side of the split segment as a candidate to reassign in the DP.

Finally, we account for local segmentation to prevent the DP from generating too many clusters. It is possible that by chance, a genomic region's haplotypic imbalance is slightly more similar to a distant segment than to its immediate neighbors ¡TODO: add figure¿, and would be assigned to the same cluster as the distant segment, even though it truly belongs in the same segment with its neighbors. Clustering that is naïve to local segmentation would thus tend to oversegment. To mitigate this, we add a penalty term to the ADP that encourages the segment being moved to be in the same cluster as its neighbors.

Given segment S_i being moved, and its immediate neighbors S_{i-1} and S_{i+1} respectively assigned to clusters k_{i-1} and k_{i+1}, we compute the following likelihoods:

$$p(S|k_{i-1}, k_{i+1}, \phi = 0) \propto \begin{cases} \beta(A_{i-1} + A_i + A_{i+1} + \mathfrak{a} + 1, & k_{i-1} = k_i = k_{i+1} \\ \quad B_{i-1} + B_i + B_{i+1} + \mathfrak{b} + 1) & \\ \quad \beta(A_{i-1} + A_i + \mathfrak{a} + 1, B_{i-1} + B_i + \mathfrak{b} + 1) & k_{i-1} = k_i \neq k_{i+1} \\ \times\, \beta(A_{i+1} + \mathfrak{a} + 1, B_{i+1} + \mathfrak{b} + 1) & \\ \quad \beta(A_{i+1} + A_i + \mathfrak{a} + 1, B_{i+1} + B_i + \mathfrak{b} + 1) & k_{i-1} \neq k_i = k_{i+1} \\ \times\, \beta(A_{i-1} + \mathfrak{a} + 1, B_{i-1} + \mathfrak{b} + 1) & \\ \quad \beta(A_i + \mathfrak{a} + 1, B_i + \mathfrak{b} + 1) & k_{i-1} \neq k_i \neq k_{i+1} \\ \times\, \beta(A_{i-1} + \mathfrak{a} + 1, B_{i-1} + \mathfrak{b} + 1) & \\ \times\, \beta(A_{i+1} + \mathfrak{a} + 1, B_{i+1} + \mathfrak{b} + 1) & \end{cases}.$$

(Analogous likelihoods are computed for $\phi = 1$, by swapping A_i and B_i in each expression.) The appropriate term is added to (4.f) depending on the upstream and downstream segments' cluster assignments.

Adding the locality penalty, the joint Gibbs sampler posterior becomes

$$p(k, \phi|S) = \frac{p(S|C_k, \phi)p(S|k_{i-1}, k_{i+1}, \phi)p(\phi)p(C_k \cup S)}{\sum_{k \in \mathbf{K}} \sum_{\phi \in \{0,1\}} p(S|C_k, \phi)p(S|k_{i-1}, k_{i+1}, \phi)p(\phi)p(C_k \cup S)}.$$

As with the initial segmentation MCMC, we use the ADP as a stochastic optimizer, running it until it reaches an equilibrium and then selecting the maximum likelihood MCMC sample.

The final refined allelic segmentation comprises contiguous regions of SNPs assigned to the same ADP cluster at the maximum likelihood sample. Formally, given SNPs ordered by genomic position indexed from $i = 1 \ldots N$, denote the SNP-specific

cluster assignments as $k_1 \dots k_N$. Denote the jth refined segment as

$$S_j = \{i \mid i \in [s_j, e_j), k_i = \kappa_j \forall i\}, \qquad s_j = e_{j-1} \forall j. \tag{4.g}$$

4.3 Total Copy Ratio Segmentation

The final result of the ADP yields an optimal segmentation based on allelic imbalance (AI) alone. However, this is only half of the information needed for inferring allelic copy ratio, and in turn absolute allelic copy number. We also need the total copy ratio (TCR) at each SNP. This is inferred from the overall genomic coverage in intervals spanning the SNPs—uniform windows for whole genomes, individual exons for whole exomes—which in the absence of systematic coverage biases would be proportional to the total amount of genomic mass within each interval, and thus the TCR of that interval.

However, because systematic biases do exist, we cannot simply segment the coverage *de novo* to obtain a TCR profile that is independent of the AI segmentation. Instead, we leverage the fact that AI segments frequently correspond perfectly to TCR segments, since the mapping between AI and TCR is nearly unique at the segment level; it is rare that an AI segment comprises multiple distinct TCR states (see Appendix A). Nonetheless, it is sometimes possible, so we use the AI segmentation to establish a strong prior foundation on the TCR segment intervals.

We leverage this prior by initializing all TCR segments in 1:1 correspondence with AI segments. This lets us fit an initial regression model to remove coverage noise correlated with known genomic covariates. We then perform an additional round of segmentation on the residuals of the regression, capturing TCR segments within AI segments that have degenerate AI→TCR mappings.

4.3.1 Coverage processing

We calculate the fragment length-normalized coverage for each window $w_i = [s_i, e_i)$ where s_i, e_i are genomic start and end coordinates for interval i. While other CNV callers simply compute coverage as the number of sequencing fragments in the window, this fails to account for the fact that different genomic windows may have different fragment length distributions. Total genomic mass of a window is proportional to the total number of sequencing bases in the window, not the total number of fragments, since longer fragments contribute more genomic mass than shorter fragments. Thus, for each w_i, we sum the lengths of all fragments overlapping the window and divide by the average fragment length in the entire sample to obtain fragment length-normalized coverage. Coverage window intervals can be defined arbitrarily, but for the analyses described herein we use uniform 2 kilobase windows for WGS data, and exon capture intervals for WES data.

4.3.2 Coverage denoising

We assign each coverage window that overlaps a SNP to its corresponding AI segment, as defined in (4.g). For whole genomes, we do not consider coverage windows that do not overlap SNPs, since assigning an allelic copy ratio state to a window requires

joint knowledge of its AI state and its TCR state. There are many more coverage bins than SNPs (the SNP reference panel yields between 5×10^5 and 10^6 SNPs per genome, depending on how closely the patient's ethnicity matches members of the reference panel; there are approximately 1.6×10^6 2kb bins in a human genome), so it is possible for a highly focal event to not overlap any SNPs, but still be distinguishable via coverage alone. We do not consider such events, since their allele-specific copy number states would be ambiguous, even if their TCRs are resolvable. However, whole exomes only have $\approx 3 \times 10^4$ SNPs, which we found insufficient to robustly regress out the higher coverage noise relative to whole genomes. For exomes, we therefore also consider coverage windows within 10kb of a SNP, and impute their AI state based on the neighboring SNP.

Given the jth allelic segment S_j containing SNPs $j_1, \ldots, j_{N_j}$, denote the corresponding coverage segment containing overlapping windows as $\Omega_j = \{w_{i(j_1)}, \ldots, w_{i(j_{N_j})}\}$. Note that $|\Omega_j| \leq |S_j|$, since some coverage windows may contain multiple SNPs. Let $\vec{c}_i$ be covariates for w_i. We fit a Poisson generalized linear model (GLM) across all coverage windows:

$$w_i|\vec{c}_i, j_i \sim \text{Pois}(\lambda_i|\vec{c}_i, j_i) \qquad \log \lambda_i|\vec{c}_i, j_i = \mu_{j_i} + \vec{\beta} \cdot \vec{c}_i,$$

whose model likelihood is

$$\mathcal{L}(\{\mu_j\}, \vec{\beta}|\{w_i\}, \{\vec{c}_i\}, \{j_i\}) = \prod_i \text{Pois}(w_i; \exp(\mu_{j_i} + \vec{\beta} \cdot \vec{c}_i)).$$

This likelihood is log-convex and can thus be globally optimized by Newton-Raphson, to obtain maximum likelihood parameters $\{\hat{\mu}_j\}$ and $\hat{\vec{\beta}}$. Fitting coverage segment-specific means $\{\mu_j\}$ allows the slope parameter $\vec{\beta}$ to only reflect coverage variability due to covariates, and not due to true copy alternations. An accurate estimate of $\vec{\beta}$ is critical for subsequent coverage segmentation.

We use the following covariates:

GC content Regions of both high and low GC content have depressed coverage, due to PCR inefficiencies during library prep [3]. High GC content makes it difficult to melt DNA duplexes, and low GC content makes it difficult to anneal primers. We found that the relationship between coverage and GC content is close to quadratic, and thus we add two covariates for GC content (fraction of C or G bases in window, f_{GC}, and f_{GC}^2) to accommodate a second-order model.

Replication timing Early replicating regions have higher coverage [2] We use replication timing tracks at the Xkb resolution, derived from ... TODO:Fill in

FAIRE-seq As the name implies, Formaldehyde Assisted Isolation of Regulatory Elements (FAIRE) [12] is an assay originally developed to identify regions of open chromatin that uses formalin to crosslink DNA to bound chromatin, and then purifies and sequences the non-crosslinked DNA. FAIRE-seq was performed in 37 cell lines comprising various normal tissues and cancers. We found that a linear combination of their coverage profiles can closely match the coverage spikes due to open chromatin in FFPE tumor samples.

Matched normal coverage Leveraging the matched normal coverage as a covariate can remove balanced germline copy number events, which would not be discernible from a lack of heterozygous SNPs close to 50% variant allele fraction. If the tumor and normal share similar sequencing characteristics, this can further suppress systematic noise not captured by the other covariates.

Capture target length Since coverage windows in whole exomes are the exon capture intervals, window-specific coverage will be proportional to the length of the exon.

4.4 Coverage segmentation

Once we have an accurate estimate of $\hat{\vec{\beta}}$ to denoise coverage, we perform an additional round of segmentation within each allelic segment. To compute the likelihood of sub-segments, we use a log-normal Poisson (LNP) GLM

$$w_i|\vec{c}_i \sim \text{Pois}(\lambda_i|\vec{c}_i) \qquad \log \lambda_i|\vec{c}_i = \mu + \hat{\vec{\beta}} \cdot \vec{c}_i + \sigma\epsilon_i \qquad \epsilon_i \sim \mathcal{N}(0,1),$$

which is robust to the observed overdispersion of coverage levels. The LNP has advantages in flexibility and interpretability when compared to other overdispersed count models such as the negative binomial.

Given some allelic segment with corresponding coverage segment $\Omega_j = \{w_{s_j}, \dots w_{e_j+1}\}$ spanning the interval $[s_j, e_j)$, the marginal likelihood of Ω_j containing no breakpoints is

$$\begin{aligned}\mathcal{L}([s_j, e_j)) &:= \int_{M,\Sigma} d\mu\, d\sigma\, \mathcal{L}(\mu, \sigma | \{w_i \mid j \in [s_j, e_j)\}, \{\vec{c}_i \mid j \in [s_j, e_j)\}, \hat{\vec{\beta}}) \\ &= \int_{M,\Sigma} d\mu\, d\sigma \prod_{i=s_j}^{e_j-1} \int_{-\infty}^{\infty} d\epsilon_i\, \text{Pois}(w_i; \exp(\mu + \hat{\vec{\beta}} \cdot \vec{c}_i + \sigma\epsilon_i)) \times \mathcal{N}(\epsilon_i; 0, 1).\end{aligned} \tag{4.h}$$

Marginal likelihoods for arbitrary disjoint sub-segmentations with breakpoints at $\{b_1, \dots, b_N\}$, $s_j < b_1 < \cdots < b_i < b_{i+1} < \cdots < e_j$ are given by

$$\mathcal{L}(\{b_1, \dots, b_N\}) = \mathcal{L}([s_j, b_1)) \times \cdots \times \mathcal{L}([b_i, b_{i+1})) \times \mathcal{L}([b_{i+1}, b_{i+2})) \times \cdots \times \mathcal{L}([b_N, e_j)),$$

with the limits of the product in (4.h) modified accordingly for each term.

This lets us define an MCMC similar to the one used for allelic segmentation. Initially, all coverage windows within Ω_j belong to the same sub-segment. We probabilistically choose a coverage window within Ω_j with probability $q(S|S^*)$ (defined in Section 4.4.2) and propose to split, accepting the split with Metropolis probability

$$\Pr(\underbrace{[s_j, e_j)}_{S} \to \underbrace{[s_j, b_1), [b_1, e_j))}_{S^*} = \min\left\{1, \frac{\mathcal{L}([s_j, b_1)) \times \mathcal{L}([b_1, e_j))}{\mathcal{L}([s_j, e_j))} \times \frac{q(S|S^*)}{q(S^*|S)}\right\}. \tag{4.i}$$

If the split is accepted, further splits within $[s_j, b_1)$ or $[b_1, e_j)$ (and additional recursions therein) are computed analogously. We can also choose to join segments that had been previously split, by randomly picking two adjacent segments and probabilistically joining them with the reciprocal of the probability defined in (4.i).

Note that the likelihood of a single coverage bin is undefined, so we restrict segments to contain at least two coverage windows. This yields a theoretical limit of detection of 4kb for WGS and two adjacent exons for WES.

4.4.1 Marginal likelihood computation

Both integrals in (4.h) are analytically intractable. To compute the inner integral

$$I_i := \int_{-\infty}^{\infty} d\epsilon_i \ \text{Pois}(w_i; \exp(\mu + \hat{\vec{\beta}} \cdot \vec{c}_i + \sigma\epsilon_i)) \times \mathcal{N}(\epsilon_i; 0, 1)$$

we use Gaussian quadrature restricted to a neighborhood around the peak of the integrand. Although in principle Hermite quadrature matches the domain of integration, the integrand is so sharply peaked that we need an impractically large number of Hermite polynomial roots to accurately approximate the integral.

To compute the outer integral

$$\int_{M,\Sigma} d\mu \, d\sigma \prod_{i=s_j}^{e_j - 1} I_i$$

we use Laplace approximation.

4.4.2 Choosing segment split points

Because coverage segments can potentially comprise thousands of windows, enumerating the entire set of all possible split points would be too slow. Instead, to limit the search space, we probabilistically determine split candidates with a heuristic that convolves two adjacent boxcar kernels ("change kernel") of varying widths $W = \{10, 50, 100, 250\}$ across all coverage windows in the segment being split, and take the absolute difference $d_{W,i}$ of both convolutions at each window i.

For each W, we store any position i whose difference $d_{W,i}$ exceeds the 98th percentile of all differences for that kernel width. This defines an initial set of candidate split points. However, the optimal split points as found by the change kernel heuristic may not correspond to the best split points based based on their marginal likelihood. Hence, we scan around each split point, computing the split marginal likelihood of neighboring points until the split marginal likelihood falls below 7 log units below the maximal split marginal likelihood.

This procedure outputs a subset of most likely split points $b_1 \dots b_N$, with associated marginal likelihoods $\mathcal{L}_1 \dots \mathcal{L}_N$. We probabilistically pick a split to propose

proportional to the candidate marginal likelihoods,

$$p_i = \frac{\mathcal{L}_i}{\sum_{j=1}^{N} \mathcal{L}_j}.$$

4.4.3 LNP prior

To both avoid over-segmentation in the due to noise and to make the LNP optimization more robust, we impose a prior on its parameters. We choose the normal inverse gamma distribution, since although it is not conjugate to the LNP, it is conjugate to the normal and Poisson distributions. For LNP parameters μ and σ, we respectively have normal and inverse gamma priors

$$\begin{aligned} \mu \,|\, \sigma,\, \mu_{prior},\, \lambda &\sim \mathcal{N}(\mu_{prior}, \sigma^2/\lambda) \\ \sigma^2 \,|\, \alpha,\, \beta &\sim \Gamma^{-1}(\alpha, \beta) \end{aligned}$$

where Γ^{-1} denotes the inverse gamma distribution. From this, we derive the joint normal inverse gamma PDF

$$\mathcal{N}\Gamma^{-1}(\mu, \sigma) = \frac{\sqrt{\lambda}}{\sigma\sqrt{2\pi}} \left(\frac{1}{\sigma^2}\right)^{\alpha+1} \exp\left(-\frac{2\beta + \lambda(\mu - \mu_{prior})^2}{2\sigma^2}\right)$$

We find empirically that setting $\alpha = 1 \times 10^{-5}$, $\beta = 4 \times 10^{-3}$ works well across the benchmarking samples we have tested. We also make the prior on μ uninformative by setting $\lambda = 1 \times 10^{-10}$ and $\mu_{prior} = \sum_{w \in \Omega_j} \log(w)/|\Omega_j|$, i.e., the mean of the log coverage windows.

4.5 Allelic coverage clustering

At this stage we have found optimal AI and TCR segmentations. Each coverage window w_i is jointly associated with an AI segment index a_i and TCR segment index t_i; denote this tuple $u_i = (a_i, t_i)$. Our initial set of allelic copy ratio (ACR) segments comprises contiguous sets of coverage windows with identical tuples. Formally, the jth ACR segment Ξ_j comprises all coverage windows such that

$$\Xi_j = \{i \mid u_i = u_{i-1} \forall i\}.$$

The overall AI α_j of Ξ_j is given by the beta distribution

$$\alpha_j | \{a_i\}_j, \{b_i\}_j \sim \text{Beta}(\textstyle\sum_{i \in \Xi_j} a_i + 1, \sum_{i \in \Xi_j} b_i + 1)$$

where a_i and b_i are the total read counts within coverage window w_i assigned to the A and B homologs, respectively. Likewise, the overall TCR τ_j of Ξ_j is given by the log-normal Poisson distribution

$$\tau_j | \{w_i\}_j, \{\vec{c}_i\}_j, \hat{\vec{\beta}} \sim \text{LNP}(\mu_j, \sigma_j | \{w_i \mid i \in \Xi_j\}, \{\vec{c}_i \mid i \in \Xi_j\}, \hat{\vec{\beta}})$$

where μ_j and σ_j are found by fitting the LNP regression described in Section 4.4 to the coverage bins and covariates assigned to Ξ_j.

The product of random variables α_j and τ_j yields two homolog-specific ACR (HSACR) distributions for Ξ_j, $\xi_{j,\mathrm{A}}$ for the A homolog and $\xi_{j,\mathrm{B}}$ for the B homolog:

$$\xi_{j,\mathrm{A}} \sim \alpha_j \times \tau_j \qquad \xi_{j,\mathrm{B}} \sim (1-\alpha_j) \times \tau_j. \tag{4.j}$$

Since human genomes only have two homologs, the allelic imbalance of the B allele is by definition $1-\alpha_j$. For n_Ξ total ACR segments, there are thus $2n_\Xi$ total HSACR segments. In regions of allelic balance ($\hat{\alpha}_j = 0.5$), the HSACR segments overlap.

These HSACR segments are suitable for use by downstream tools (e.g. ABSOLUTE, PhylogicNDT), but they can be further refined by combining segments with identical levels of HSACR but with distinct AI and TCR levels. For example, a triploid region with 2 copies of homolog A and 1 copy of homolog B has AI and TCR levels distinct from a region of copy-neutral loss-of-heterozygosity (2 copies of homolog A, 0 copies of homolog B), even though both share a common level of ACR for homolog A (2 copies in both regions). Highly focal regions supported by few coverage windows/SNPs have intrinsically high measurement uncertainty; by combining them with other regions of the same HSACR, we increase our ability to definitively assign ACR states to them. To do this, we employ an additional round of Dirichlet process clustering, which we call the Allelic Coverage Dirichlet Process (ACDP).

4.5.1 ACDP model

For each pair of HSACR segments $\xi_{j,\mathrm{A}}$ and $\xi_{j,\mathrm{B}}$, we compute the product of random variables in (4.j) via a Monte Carlo simulation, since there is no closed form expression for the product of beta and LNP random variables. Each ACR segment consists of $n_j = |\xi_j|$ coverage bins, so we draw n_j samples from the corresponding beta and LNP distributions and compute the product of each draw. Denote these sets of samples

$$\tilde{\boldsymbol{\xi}}_{j,\mathrm{A}} = \{\tilde{\alpha}_i\tilde{\tau}_i \mid \tilde{\alpha}_i \sim \alpha_j, \tilde{\tau}_i \sim \tau_j, i \in \{1 \dots n_j\}\}$$
$$\tilde{\boldsymbol{\xi}}_{j,\mathrm{B}} = \{(1-\tilde{\alpha}_i)\tilde{\tau}_i \mid \tilde{\alpha}_i \sim \alpha_j, \tilde{\tau}_i \sim \tau_j, i \in \{1 \dots n_j\}\}.$$

We found that this product of random variables is nicely approximated by a normal distribution, allowing us to fit a Dirichlet process Gaussian clustering model across all HSACR segments. The likelihood of each HSACR segment is given by a normal distribution,

$$\mathcal{L}(\mu,\lambda|\underbrace{\tilde{\boldsymbol{\xi}}_{j,\mathrm{A}}}_{\tilde{\mathbf{x}}_j}) = \prod_{i=1}^{n_j} \mathcal{N}(\underbrace{\tilde{\alpha}_i\tilde{\tau}_i}_{\tilde{x}_i};\mu,\lambda) \qquad \mathcal{L}(\mu,\lambda|\tilde{\boldsymbol{\xi}}_{j,\mathrm{B}}) = \prod_{i=1}^{n_j} \mathcal{N}((1-\tilde{\alpha}_i)\tilde{\tau}_i;\mu,\lambda) \tag{4.k}$$

with a normal-gamma prior on the mean and precision parameters,

$$p(\mu,\lambda;\mu_0,\kappa_0,\alpha_0,\beta_0) = \mathcal{NG}(\mu,\lambda;\mu_0,\kappa_0,\alpha_0,\beta_0)$$

$$= \mathcal{N}(\mu; \mu_0, (\kappa_0\lambda)^{-1})\mathcal{G}(\lambda; \alpha_0, \text{rate} = \beta_0)$$

where $\mathcal{G}$ denotes the gamma distribution. We have prior-weighted likelihood

$$p(\tilde{\mathbf{X}}_j|\mu, \lambda, \underbrace{\mu_0, \kappa_0, \alpha_0, \beta_0}_{\mathcal{H}}) = \left[\prod_{i=1}^{n_j} \mathcal{N}(\tilde{x}_i; \mu, \lambda)\right] \mathcal{NG}(\mu, \lambda; \mathcal{H}), \tag{4.l}$$

whose marginal likelihood has closed form,

$$\begin{aligned} p(\tilde{\mathbf{X}}_j|\mathcal{H}) &= \int_{M,\Lambda} d\mu\, d\lambda\, p(\tilde{\mathbf{X}}_j|\mu, \lambda, \mathcal{H}) \\ &= \frac{\Gamma(\alpha_n)}{\Gamma(\alpha_0)} \frac{\beta_0^{\alpha_n}}{\beta_n^{\alpha_0}} \sqrt{\frac{\kappa_0}{\kappa_n} \frac{1}{(2\pi)^{n_j}}} \end{aligned} \tag{4.m}$$

where Γ denotes the gamma function, with

$$\mu_n = \frac{\kappa_0\mu_0 + n_j\langle\tilde{\mathbf{X}}_j\rangle}{\kappa_0 + \langle\tilde{\mathbf{X}}_j\rangle} \qquad \kappa_n = \kappa_0 + n_j$$

$$\alpha_n = \alpha_0 + n_j/2 \qquad \beta_n = \beta_0 + \frac{1}{2}\sum_{i=1}^{n_j}(x_i - \langle\tilde{\mathbf{X}}_j\rangle)^2 + \frac{\kappa_0 n_j(\langle\tilde{\mathbf{X}}_j\rangle - \mu_0)^2}{2(\kappa_0 + n_j)}.$$

4.5.2 Hyperparameters

We employ a prior because there is some latent noise in the system not reflected in the data likelihood. The normal-gamma distribution is conjugate to the joint posterior on (μ, λ), making hyperparameter interpretation intuitive—the prior distribution is equivalent to a likelihood with κ_0 observations of μ with sample mean μ_0, with precision estimated from 2α observations with sample mean μ_0 and sum of squared deviations 2β. We empirically set α_0 to the average number of data points $\langle n_j\rangle$ across all HSACR segments and $\beta_0 = \sum_{i=1}^{n}(x_i - \bar{x})^2/2$ to the average sum of squared deviations for each set of segment data points. Since we lack prior knowledge for what the mean value of a cluster should be, we leave the prior on μ uninformative.

4.5.3 MCMC procedure

The closed-form nature of the marginal likelihood for a set of HSACR segments makes optimizing the ACDP clustering model amenable to MCMC sampling in a manner similar to the allelic clustering model defined in Section 4.2. We initialize each HSACR segment in its own cluster, yielding clusters indexed as $\mathbf{K} = \{1 \ldots 2n_\Xi\}$. At each MCMC step, we pick an HSACR segment at random, unassign it from its current cluster, and probabilistically join it with an existing cluster, or open a new cluster.

The probability that an HSACR segment $H = \tilde{\boldsymbol{\xi}}$ joins cluster C_k is proportional to

$$p(H \cup C_k|\mathcal{H}) \propto \frac{\mathcal{L}(H \cup C_k)}{\mathcal{L}(C_k)} p_{\text{DP}}(H \cup C_k)$$

$$\equiv p(H|C_k, \mathcal{H}) p_{\mathrm{DP}}(H \cup C_k)$$

where

$$\mathcal{L}(H \cup C_k) = p(\tilde{\boldsymbol{\xi}} \cup \{\tilde{x}_i \mid i \in C_k\}|\mathcal{H})$$
$$\mathcal{L}(C_k) = p(\{\tilde{x}_i \mid i \in C_k\}|\mathcal{H}),$$

with marginal likelihoods defined in (4.m), HSACR sets $\tilde{\boldsymbol{\xi}}$ and $\{\tilde{x}_i\}$ defined in (4.k), and DP prior $p_{\mathrm{DP}}(H \cup C_k)$ defined in Appendix B. H can also create a new cluster by setting $C_k = \emptyset$, which makes $p(H|\emptyset, \mathcal{H})$ fall back on the normal-gamma prior in (4.l).

To speed up the MCMC, we also propose merging entire clusters. At random, we pick a cluster C_j, unassign all segments therein, and in a manner similar to above, compute probabilities and underlying marginal likelihoods of joining every other cluster,

$$\begin{aligned} p(C_j \cup C_k) &\propto p(C_j|C_k, \mathcal{H}) p_{\mathrm{DP}}(C_j \cup C_k) \\ \mathcal{L}(C_j \cup C_k) &= p(\{\tilde{x}_i \mid i \in C_j\} \cup \{\tilde{x}_i \mid i \in C_k\}|\mathcal{H}). \end{aligned} \tag{4.n}$$

Allowing clusters to merge can get the MCMC stuck in a local optimum, since it only requires a single move to merge clusters but potentially many moves to entirely reverse a merge. To mitigate this, we propose splitting clusters. Let $\mathbf{H}_s = \{H_{j_1}, \ldots, H_{j_{|C_j|}}\}$ be the HSACR segments in cluster C_j sorted by their means, i.e., $\langle\{\tilde{x}|\tilde{x} \in H_{j_k}\}\rangle \leq \langle\{\tilde{x}|\tilde{x} \in H_{j_{k+1}}\}\rangle$. We calculate the marginal likelihoods of bifurcating $\mathbf{H}_s$ at every possible index i,

$$\mathcal{L}_i := p\left(\bigcup_{k=1}^{i} H_{j_i}|\mathcal{H}\right) \times p\left(\bigcup_{k=i+1}^{|C_j|} H_{j_i}|\mathcal{H}\right),$$

construct a probability distribution proportional to the marginal likelihoods, and draw from that distribution to propose a bifurcation point. We then pick a random half of the bifurcation, unassign the segments on that half, and merge them with another clustering according to the probabilities defined in (4.n). We observed that even with the a well calibrated DP prior (see B), the influence of the prior in large ACDP clusters, would often overpower smaller subclonal clusters, forcing them to merge erroneously. To mitigate this, we implemented a procedure to distinguish clonal HSACR segments from subclonal segments based on AI described in C, and leverage it to run the ACDP separately on each class.

4.6 Qualitative Assessment on Real Tumors

We downloaded 16 tumor BAMs aligned to hg19 from [17]. We first realigned each of the BAMs to hg38 using bwa-mem [25] and samtools [26]. We then ran every sample through each of the 5 competing tools, using the raw data collection steps and method settings as described in 4.7. Note that sample CH1022 failed to complete HATCHet due to a segfault during phasing that could not be resolved. All other sample-method pairs ran to completion. The raw results for each algorithm can be found in the

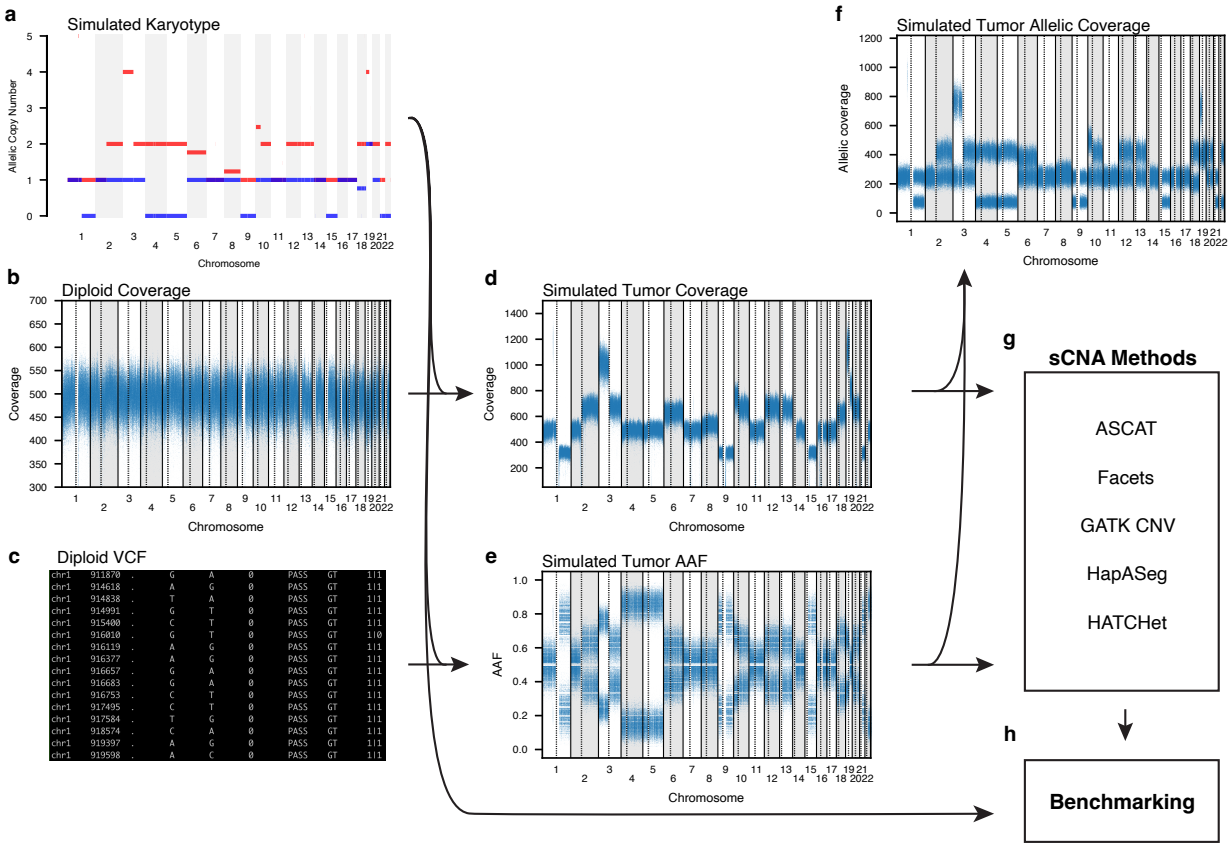

Fig. 5 HapAseg benchmarking panel schematic **a**, Simulated tumor karyotype is randomly generated with biologically realistic characteristics. **b**, Coverage is computed at 2kb genomic intervals in confidently diploid, normal tissue. **c**, Genotypes and phasing information are calculated for the normal tissue. **d**, Simulated tumor coverage is generated by scaling the coverage level in the confidently diploid coverage to the TCR level of the simulated karyotype. **e**, allele counts at heterozygous sites are generated by scaling the allele counts in the diploid tissue by the homologue specific copy number designated by the simulated karyotype. **f**, The simulated TCR is multiplied by the average AAF in each covered bin to compute the simulated allelic coverage for visualization purposes. **g**, State of the art sCNA methods receive the simulated tumor coverage and AAF as input. **h**, The results from the sCNA methods are compared to the simulated karyotype and AAD scores are computed for quantitative benchmarking.

available benchmarking data while the segfiles from each method can be found in the extended data.

4.7 Benchmarking

We sought to rigorously benchmark HapASeg against other state-of-the-art methods across a wide range of karyotypic complexities, tumor purities, sequencing modalities

and sample types. To this end, we developed a suite of tools for simulating tumor coverage and variant profiles from arbitrary copy number profiles called CNV-Suite. The source code for this package can be found on github. Prior approaches to benchmarking CNV callers on simulated data fail to reflect the complex noise and correlation structure of real sequencing coverage [7, 15, 19–21], leading to biased results. In contrast, CNV-Suite takes variant counts and coverage profiles calculated from real diploid samples as input, and linearly scales these values based on the desired copy number profile to create realistic simulated tumor count data and ground-truth segmentation files. Using CNV-Suite, we benchmarked CNV calling performance across 5 methods: HapASeg, ASCAT, Facets, HATCHet and GATK; nine tumor purities: 0.1, 0.2 ... 0.9; four sample types: fresh frozen whole genome, high quality FFPE whole genome, heavily degraded FFPE whole genome and fresh frozen TWIST whole exome; and 50 simulated tumor profiles. For each of these 1800 combinations, we created simulated input data files for each of the five competing methods and compared the accuracy of their resulting calls to the simulation ground truth.

4.7.1 Simulating Tumor Samples

Each copy number calling method fundamentally requires a allele counts file for each of the SNPs in the tumor of interest and a coverage file for each of its coverage windows. Typically, each method generates these files by processing sequencing reads from a Tumor-Normal pair. For our simulations we perform the BAM processing step one time on a reference sample for each sample type, and create simulated tumor counts based on these references. By reducing compute-intensive BAM processing steps to one run per sample per method and re-using those raw data for many simulated profile, we are able to greatly expand the number of benchmarking samples. All sequencing data used for benchmarking were aligned to the hg38 reference genome.

Each simulated tumor sample is based on sequencing data from a real, reference diploid sample. For the fresh frozen whole genome we use the Illumnia platinum sequencing of NA12878 as our reference while for the whole exome sequencing we use a Twist capture library of NA12878 from the Broad Genomics Platform benchmarking efforts. For the FFPE samples, we use two FFPE Richter's Transformed CLL WGS from a previous study (CH1022, CH1032) [16, 17]. Sample CH1022 is not fully diploid, however we limited the simulations to chromosomes that we identified as confidently diploid. For all of the samples, we computed variant counts and coverage profiles for the diploid normal samples using each method's preferred or provided BAM-processing modules, ensuring that each method got simulated data with its expected format and potential biases. Allele counts are calculated at the variant sites of NA12878. NA12878 was chosen for its complete phase information and abundance of sequencing data. The complete phasing information for each variant is required to properly simulate allele specific copy number changes.

The simulated copy profiles were generated by first randomly sampling hyperparameters for the number of subclones (0-2), the number of arm and focal events, and proportion of clonal event for the sample. Copy number events were then randomly added to a particular allele (maternal or paternal) on the initially diploid copy profile. Focal events follow the previously reported distributions of event lengths, however,

we also add a number of hyper-focal events of length < 150kb in order to benchmark each method's ability to call short segments. Each profile data file and a illustration of its absolute copy profile is available in Extended Data.

To create simulated input files for each (method, profile, sample type, purity) combination we first find the total allele counts and coverage counts from the files generated by running the particular method's pre-processing functions on diploid sample type sequencing files. Then, to simulate coverage at a given clonal loci Cov_ℓ with X_A maternal copies and X_B paternal copies we simply scale the diploid coverage Cov_d to the coverage implied by the purity p and the total copy state $\tau = X_A + X_B$, i.e.

$$\text{Cov}_\ell = \text{Cov}_d(1-p) + p \cdot \frac{\tau \cdot \text{Cov}_d}{2}$$

For loci with subclones at subclonal fractions $f_1, f_2, \ldots f_i$ the coverage is scaled by the weighted allelic copies of each of the clones $X_A^{f_i}, X_B^{f_i}$ and the clonal copies $X_A^c + X_B^c$ where $\tau^z = X_A^z + X_B^z$ is the total copy number of clone z.

$$\text{Cov}_\ell = \text{Cov}_d\left[(1-p) + \frac{p[\tau^c(1-\sum_i f_i) + \sum_i f_i \tau^i]}{2}\right]$$

The allele counts A_ℓ, B_ℓ at locus ℓ with T_ℓ total allele counts are similarly computed as

$$A_\ell = \left\lfloor T_\ell \cdot \frac{(1-p) + p\big[X_A^c(1-\sum_i f_i) + \sum_i f_i X_A^{f_i}\big]}{2(1-p) + p\big[\tau^c(1-\sum_i f_i) + \sum_i f_i \tau^{f_i}\big]} \right\rfloor, B_\ell = T_\ell - A_\ell$$

4.7.2 Normal samples

Some methods such as HATCHet require allele counts and coverage from a normal sample in order to run, while each of the other methods can optionally take a normal sample as input. In an attempt to benchmark each method on an equal playing field, we decided to let every method use a matched normal sample. For the matched samples we located sequencing data from alternative libraries of the same tissue, when possible. For the fresh frozen WGS we used a BAM from a separate library of NA12878 made available in a recent publication [27] and for the fresh frozen exome we used a separate TWIST sequencing of NA12878. Clinical FFPE samples generally do not have paired FFPE normal tissue and instead use fresh frozen blood normals, which renders the normal coverage less useful to regress out coverage noise. To reflect the difficulty of copy number calling on FFPE clinical samples in our benchmarking study we used matched fresh frozen blood normals for each of the FFPE clinical samples. To generate allele counts and coverage for each matched normal sample, we ran the same preprocessing steps as for the simulated tumor samples.

4.7.3 Panel of Normals

GATK CNV can optionally use a panel of normals in place of a matched normal to regress out coverage artifacts. We created a WGS PoN using 50 random samples from the 1000 genomes stage 3 PCR-free whole genome cohort and a WES PoN using 6

TWIST WES samples sequenced at Broad GP. FFPE samples were given the 1kG WGS PoN since to the best of our knowledge no WGS FFPE PoN is available. We did not see a noticeable performance improvement for GATK between PoN normalization and normal sample normalization. Unless otherwise specified, all GATK CNV analyses were done using the corresponding PoN.

4.7.4 Running competing tools

All benchmarking methods were run using the open-source wolF workflow management software. The scripts for running each of the tools are available in the benchmarking folder of the github repository. In general we ran each tool using its default configuration. Details of the preprocessing steps, run commands and options used for each method can be found in D.

4.7.5 Comparing Results

We wish to fairly compare the copy number calls from each of the competing methods across the range of simulated purities. Only a subset of the methods in our benchmarking set return absolute copy number estimates, however, all methods besides Facets compute allelic coverage estimates. We therefore chose to evaluate methods by comparing their allelic coverage estimates to the ground truth simulation profile. In particular, we developed the Average Absolute Difference (AAD) score to evaluate the quality of coverage estimates given a ground truth coverage profile (see S.1 for illustration). The AAD score is defined as the length weighted absolute difference in allelic coverage across the autosome between the estimated and ground truth samples. In particular, let the called segments for a particular simulated copy profile e at purity p and method m be denoted as $S_{(e,p,m)} = \{s_1, s_2 \dots s_i\}$ and the ground truth segments as $G_{(e,p)} = \{s_1, s_2 \dots s_j\}$ where $s_k = (\text{start}_k, \text{end}_k, \mu_k^{\text{major}}, \mu_k^{\text{minor}})$. We can compute the intersection of these segment intervals $I_{(e,p,m)} = \{t_1, t_2 \dots t_\ell\}, t_\ell = (\text{start}_\ell, \text{end}_\ell, \mu_{S,\ell}^{\text{major}}, \mu_{S,\ell}^{\text{minor}}, \mu_{G,\ell}^{\text{major}}, \mu_{G,\ell}^{\text{minor}})$ and compute the AAD score as

$$\text{AAD}^{\text{naive}}_{(e,p,m)} = \frac{\sum_\ell \left(\text{end}_\ell - \text{start}_\ell\right) \left[\left| \mu_{S,\ell}^{\text{major}} - \mu_{G,\ell}^{\text{major}} \right| + \left| \mu_{S,\ell}^{\text{minor}} - \mu_{S,\ell}^{\text{minor}} \right| \right]}{\sum_\ell \left(\text{end}_\ell - \text{start}_\ell\right)}$$

This comparison is complicated by the fact that methods we are comparing use varying units of coverage. Moreover, the ground truth coverage profiles at different purities will have varying levels of average coverage, and hence varying AAD scales (e.g. a duplicated region miscalled as diploid will had a much higher absolute coverage difference in a sample with 0.9 purity than one with 0.1). To normalize comparisons across methods and purities we find the optimal affine transformation $f(x) = ax + c$ which minimizes the AAD score between the intersection of a given output segmentation file $S_{(e,p,m)}$ and the ground truth segments at 0.7 purity $G_{(e,0.7)}$ for the same profile e.

That is,

$$\mathrm{AAD}_{(e,p,m)} = \min_{f} \frac{\sum_{\ell} \left(\mathrm{end}_{\ell} - \mathrm{start}_{\ell}\right) \left[\left| f(\mu_{S,\ell}^{\mathrm{major}}) - \mu_{G,\ell}^{\mathrm{major}} \right| + \left| f(\mu_{S,\ell}^{\mathrm{minor}}) - \mu_{S,\ell}^{\mathrm{minor}} \right| \right]}{\sum_{\ell} \left(\mathrm{end}_{\ell} - \mathrm{start}_{\ell}\right)}$$

For the heatmap result plots we use the same optimal AAD scores described here, however we classify the segments based on the ground truth segment length of the original ground truth segment, rather than the length of the intersected segment.

Supplementary Figures

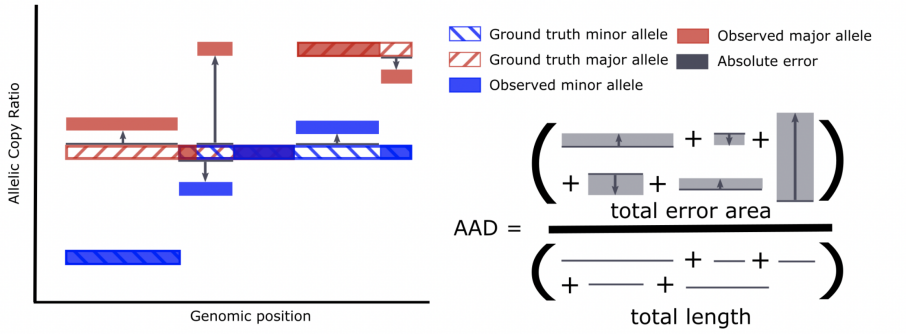

Fig. S.1 AAD calculation schematic. The average absolute difference is computed by taking the length-weighted average of the absolute differences between the ground truth copy ratios and the method estimated copy ratio for both alleles.

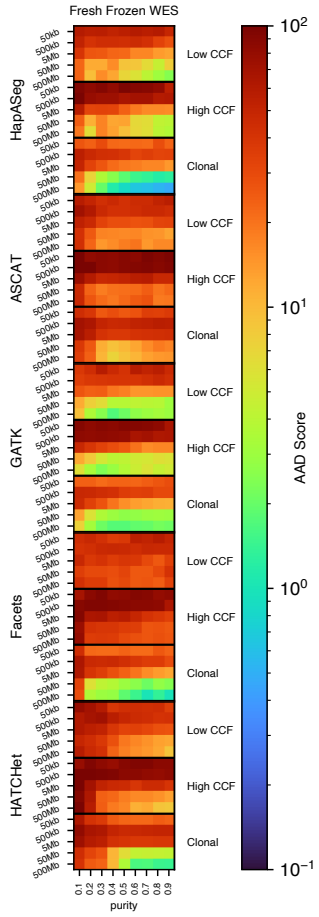

Fig. S.2 Fresh frozen WES benchmarking results heatmap. AAD heatmap examines the accuracy of the sCNA methods stratified by purity, ground truth event length and the (CCF) of the simulated CNA segments (clonal: ccf=1, low: ccf < 0.7, high: ccf ≥ 0.7).

Data availability

Richter's transformed CLL samples along with matched blood normals were obtained from Parry et al. [16] and realigned to hg38. WGS sequencing data for NA12878 used for benchmarking are available at the NCI Sequence Read Archive (SRA) under accession number SRX3666165 [27] and from the Illumina platinum truthset, available at the European Nucleotide Archive (ENA) under accession code ERR194147 [16]. WES TWIST data for NA12878 were generated by the Broad Institute Genomics Platform and are available at XXX TODO:FILL. The simulated karyotype profiles,

along with the associated simulated tumor data and method results can be downloaded from .

Code availability

HapASeg is maintained and available for use at . Code for generating the figures in this publication can be found in . The benchmarking simulations in this study relied on tools from CNV-Suite which is available at .

Acknowledgements

This work was funded in part by the National Cancer Institute Genomic Data Analysis Network grant XXX

Author information

These authors contributed equally: Oliver Priebe and Julian Hess

Broad Institute of MIT and Harvard, Cambridge, MA, USA
Oliver Priebe, Conor Messor, Claudia Chu, Julian Hess, Mendy Miller, Gad Getz

Center for Cancer Research, Massachusetts General Hospital, Boston, MA, USA
Gad Getz

Harvard Medical School, Boston, MA, USA
Gad Getz

Department of Pathology, Massachusetts General Hospital, Boston, MA, USA
Gad Getz

Author contribution

J.H. and G.G. conceived the methodology. O.P. and J.H developed the method. O.P. performed analysis of the Richter's samples. O.P., J.H. and C.M. developed the benchmarking simulation framework. O.P. performed and analyzed the benchmarking simulations. O.P., J.H., M.M., and G.G. prepared the manuscript and figures.

Corresponding Authors

Correspondence to Julian Hess or Gad Getz.

Declarations

G.G. is an inventor on patent applications related to MSMuTect, MSMutSig, MSIDetect and POLYSOLVER; and is a founder and consultant of and holds privately held equity in Scorpion Therapeutics. The other authors declare no competing interests. O.P. is an employee at and privite equity holder in Serinus Biosciences.

References

[1] Kokkat, T. J., Patel, M. S., McGarvey, D., LiVolsi, V. A. & Baloch, Z. W. Archived formalin-fixed paraffin-embedded (FFPE) blocks: A valuable underexploited resource for extraction of DNA, RNA, and protein. *Biopreserv. Biobank.* **11**, 101–106 (2013).

[2] Koren, A. *et al.* Genetic variation in human DNA replication timing. *Cell* **159**, 1015–1026 (2014).

[3] Benjamini, Y. & Speed, T. P. Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Res.* **40**, e72 (2012).

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

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

[6] Van der Auwera, G. A. & O'Connor, B. D. *Genomics in the Cloud: Using Docker, GATK, and WDL in Terra* ("O'Reilly Media, Inc.", 2020).

[7] Ha, G. *et al.* TITAN: inference of copy number architectures in clonal cell populations from tumor whole-genome sequence data. *Genome Res.* **24**, 1881–1893 (2014).

[8] Gao, G. F. *et al.* Tangent normalization for somatic copy-number inference in cancer genome analysis. *Bioinformatics* **38**, 4677–4686 (2022).

[9] Goyal, A. *et al.* Ultra-fast next generation human genome sequencing data processing using DRAGENTM bio-IT processor for precision medicine. *Open J. Genet.* **07**, 9–19 (2017).

[10] Soong, D. *et al.* CNV radar: an improved method for somatic copy number alteration characterization in oncology. *BMC Bioinformatics* **21**, 98 (2020).

[11] Hoffman, E. A., Frey, B. L., Smith, L. M. & Auble, D. T. Formaldehyde crosslinking: a tool for the study of chromatin complexes. *J. Biol. Chem.* **290**, 26404–26411 (2015).

[12] Giresi, P. G., Kim, J., McDaniell, R. M., Iyer, V. R. & Lieb, J. D. FAIRE (Formaldehyde-Assisted isolation of regulatory elements) isolates active regulatory elements from human chromatin. *Genome Res.* **17**, 877–885 (2007).

[13] Loh, P.-R. *et al.* Reference-based phasing using the haplotype reference consortium panel. *Nat. Genet.* **48**, 1443–1448 (2016).

[14] Shen, R. & Seshan, V. E. FACETS: allele-specific copy number and clonal heterogeneity analysis tool for high-throughput DNA sequencing. *Nucleic Acids Res.* **44**, e131 (2016).

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

[16] Parry, E. M. *et al.* Evolutionary history of transformation from chronic lymphocytic leukemia to richter syndrome. *Nat. Med.* **29**, 158–169 (2023).

[17] Klintman, J. *et al.* Genomic and transcriptomic correlates of richter transformation in chronic lymphocytic leukemia. *Blood* **137**, 2800–2816 (2021).

[18] Mercer, T. R., Xu, J., Mason, C. E., Tong, W. & MAQC/SEQC2 Consortium. The sequencing quality control 2 study: establishing community standards for sequencing in precision medicine. *Genome Biol.* **22**, 306 (2021).

[19] Chen, H., Bell, J. M., Zavala, N. A., Ji, H. P. & Zhang, N. R. Allele-specific copy number profiling by next-generation DNA sequencing. *Nucleic Acids Res.* **43**, e23 (2015).

[20] Fischer, A., Vázquez-García, I., Illingworth, C. J. R. & Mustonen, V. High-definition reconstruction of clonal composition in cancer. *Cell Rep.* **7**, 1740–1752 (2014).

[21] McPherson, A. W. *et al.* ReMixT: clone-specific genomic structure estimation in cancer. *Genome Biol.* **18**, 140 (2017).

[22] Beroukhim, R. *et al.* The landscape of somatic copy-number alteration across human cancers. *Nature* **463**, 899–905 (2010).

[23] 1000 Genomes Project Consortium *et al.* A global reference for human genetic variation. *Nature* **526**, 68–74 (2015).

[24] Cibulskis, K. *et al.* Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. *Nat. Biotechnol.* **31**, 213–219 (2013).

[25] Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM (2013).

[26] Danecek, P. *et al.* Twelve years of SAMtools and BCFtools. *Gigascience* **10** (2021).

[27] Marks, P. *et al.* Resolving the full spectrum of human genome variation using Linked-Reads. *Genome Res.* **29**, 635–645 (2019).

[28] Browning, S. R. & Browning, B. L. Haplotype phasing: existing methods and new developments. *Nat. Rev. Genet.* **12**, 703–714 (2011).

[29] Ceballos, F. C., Hazelhurst, S. & Ramsay, M. Assessing runs of homozygosity: a comparison of SNP array and whole genome sequence low coverage data. *BMC Genomics* **19**, 106 (2018).

Appendix A Total copy number as function of allelic imbalance

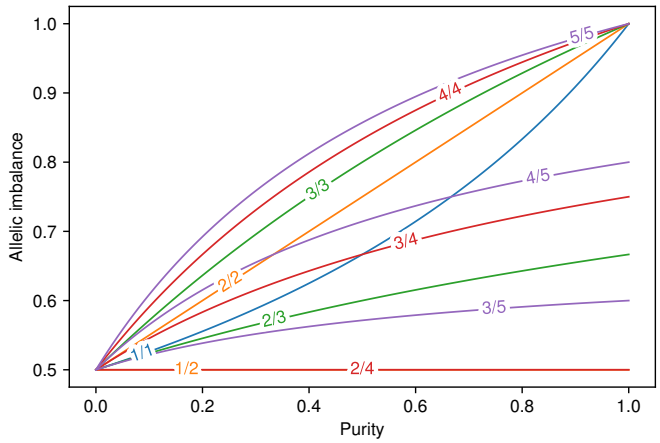

Fig. A1 Allelic imbalances as a function of tumor purity for different clonal A allele counts and tumor ploidies, for ploidies from 1 . . . 5. Each line is labeled as (A allele count)/(total tumor ploidy), e.g. 1/2 corresponds to a diploid tumor; 2/2 corresponds to copy-neutral LoH. Note that the curves rarely intersect, indicating that total copy number is nearly a well-defined function of allelic imbalance. However, note that this function is not perfectly well-defined: multiple allelic imbalances can map to the same ploidy (where lines intersect), especially when subclonal sCNAs exist (not shown).

Here, we demonstrate that allelic imbalance is a nearly well-defined function of total copy ratio. Allelic imbalance f of a heterozygous germline SNP is a function of its allele-specific copy number on one of the two homologs n_A, the total copy number τ, and the fraction of tumors cells α.

$$\begin{aligned} f(n_A, \tau, \alpha) &= \frac{n_A\alpha + 1(1-\alpha)}{\tau\alpha + 2(1-\alpha)} \\ &= \frac{1 + \alpha(n_A - 1)}{2 + \alpha(\tau - 2)} \end{aligned} \tag{1}$$

The numerator is the total number of copies containing the allele; the fraction $1-\alpha$ of normal cells are diploid and thus always have one copy of the allele, while the fraction α of tumor cells have n_A copies of the allele. The denominator is the total number of copies; diploid normal cells all have two copies, while tumor cells all have τ copies.

We plot f in Figure A1 for all purities at a variety of allele-specific copy numbers. We see that with a few exceptions (detailed below), the curves never intersect, demonstrating that allelic imbalance is nearly a well-defined function of total copy number, and thus ideal to use as a strong prior for total copy segmentation.

For all allele-specific copy numbers with $\tau \leq 5$, the exceptions are:

1. At any purity, a balanced diploid ($n_A = 1$, $\tau = 2$) region has the same allelic imbalance ($\frac{1}{2}$) as a balanced tetraploid (genome doubled) region ($n_A = 2/\tau = 4$), but will have differing total copy ratios respectively corresponding to total ploidies of 2 and 4. Thus, a 1/2 region immediately adjacent to a 2/4 region would not be distinguishable from allelic imbalance segmentation alone. Indeed, this will be the case for any balanced region.
2. At purity = 0.5, a haploid (1/1) region has the same allelic imbalance ($\frac{2}{3}$) as a tetraplod (3/4) region.
3. At purity = 1/3, a region of copy-neutral loss-of-heterozygosity (2/2) has the same allelic imbalance ($\frac{2}{3}$) as a pentaploid (4/5) region.
4. At purity = 2/3, a haploid region and a pentaploid region (4:1) both have imbalance 0.7
5. At purity = 100%, all loss-of-heterozygosity regions have the same allelic imbalance (1).

Furthermore, ambiguities only arise if degenerate AI/TCR pairings occur in immediately adjacent segments (e.g., a balanced diploid segment next to a balanced tetraploid segment). Non-adjacent segments

Note that more ambiguities can arise if subclones are present. Let the fraction of the ith subclone be s_i; $\mathbf{s} = \{s_1, \ldots, s_N\}$, $\sum_{i=1}^{N} s_i = 1$. Then n_A and τ in (1) become a weighted average of allelic copies over subclones

$$n'_A(\mathbf{s}) = \sum_{i=1}^{N} s_i n_{A_i} \qquad \tau'(\mathbf{s}) = \sum_{i=1}^{N} s_i \tau_i$$
$$f(n'_A(\mathbf{s}), \tau'(\mathbf{s}), \alpha) = \frac{1 + \alpha(n'_A(\mathbf{s}) - 1)}{2 + \alpha(\tau'(\mathbf{s}) - 2)}$$

(In a region with no subclones, $s_1 = 1$; $\sum_{i=2}^{N} s_i = 0$, recovering (1).)

We show in figure ¡x¿ the allelic imbalance levels for varying mixtures of $\mathcal{T}$ and $\mathcal{S}$ populations with differing copy states. Observe the additional complexity the subclone presents, for the mixture of populations at a given loci can result in nearly any allelic imbalance level.

Appendix B Dirichlet Process details

A Dirichlet Process (DP) is a commonly used prior for mixture models with variable numbers of components. It is a prior on the number of datapoints in each component. The DP prior can be derived by starting with a mixture model with a fixed number of components (K), with a symmetric Dirichlet-categorical prior on the assignment probabilities of each datapoint.

$$\begin{aligned} x_i | \{\theta_k\}_{k=1}^K, c_i &\sim p(\theta_{c_i}) \\ c_i | \{p_k\}_{k=1}^K &\sim \text{Cat}(p_1, \ldots, p_K) \\ \{p_k\} &\sim \text{SymDir}(\alpha/K). \end{aligned}$$

We typically use a Gibbs sampler when sampling this DP mixture model via MCMC; we take one datapoint, remove it from the cluster it is currently assigned to, and compute the probability of assigning it to all other clusters, conditioned on the assignments of every other datapoint. This probability (that the ith datapoint is assigned to component $\tilde{c}$, given the assignments of all other data points) is

$$p(c_i = \tilde{c} | \underbrace{c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_N}_{\mathbf{c}_{-i}}) = \frac{p(\overbrace{c_1, \ldots, c_i = \tilde{c}, \ldots, c_N}^{\mathbf{c}})}{p(\mathbf{c}_{-i})},$$

by definition of conditional probability. Without loss of generality, let $i = 1$ and $\tilde{c} = 1$, since the ordering of data points and clusters is arbitrary. The numerator is the Dirichlet-Categorical likelihood, after marginalizing out $\mathbf{p}$:

$$\begin{aligned} p(\mathbf{c}) &= \int d\mathbf{p}\ \mathrm{Cat}(c_1 = 1|\mathbf{p}) \left[\prod_{i=2}^{N} \mathrm{Cat}(c_i|\mathbf{p})\right] \mathrm{SymDir}(\mathbf{p}|\alpha/K) \\ &\propto \int d\mathbf{p}\, p_1^{1+n_1+\alpha/K-1} \prod_{k=2}^{K} p_k^{n_k+\alpha/K-1}. \end{aligned}$$

where n_1 is the number of points already in cluster 1, i.e. not counting x_1. This integral has closed form—it's the multivariate beta function:

$$\begin{aligned} \int d\mathbf{p}\, p_1^{1+n_1+\alpha/K-1} \prod_{k=2}^{K} p_k^{n_k+\alpha/K-1} &= \frac{\Gamma(1 + n_1 + \alpha/K) \prod_{k=2}^{K} \Gamma(n_k + \alpha/K)}{\Gamma(1 + \sum_{k=1}^{K} n_k + \alpha/K)} \\ &= \frac{\Gamma(1 + n_1 + \alpha/K) \prod_{k=2}^{K} \Gamma(n_k + \alpha/K)}{\Gamma(N + \alpha)}. \end{aligned} \tag{2}$$

.

The likelihood in the denominator is similar:

$$\begin{aligned} p(\mathbf{c}_{-i}) &= \int d\mathbf{p} \left[\prod_{i=2}^{N} \mathrm{Cat}(c_i|\mathbf{p})\right] \mathrm{SymDir}(\mathbf{p}|\alpha/K) \\ &\propto \int d\mathbf{p}\, p_1^{n_1+\alpha/K-1} \prod_{k=2}^{K} p_k^{n_k+\alpha/K-1} \\ &= \frac{\prod_{k=1}^{K} \Gamma(n_k + \alpha/K)}{\Gamma(N + \alpha - 1)}. \end{aligned} \tag{3}$$

Note the -1 term in the denominator, since we have one fewer datapoint.

Putting (2) and (3) together, we have

$$\frac{p(\mathbf{c})}{p(\mathbf{c}_{-i})} = \frac{\Gamma(1 + n_1 + \alpha/K)\Gamma(N + \alpha - 1)}{\Gamma(n_1 + \alpha/K)\Gamma(N + \alpha)} = \frac{n_1 + \alpha/K}{N + \alpha - 1} = \frac{n_{\tilde{c}} + \alpha/K}{N + \alpha - 1}. \tag{4}$$

where the last equality arises from the fact that we took $\tilde{c} = 1$ without loss of generality. If the number of clusters is variable, we take the limit $K \to \infty$, yielding

$$\lim_{K\to\infty} p(c_i = \tilde{c}|c_1, \ldots, c_{i-1}, c_{i+1}, \ldots, c_N) = \frac{n_{\tilde{c}}}{N + \alpha - 1}$$

The probability of opening a new cluster is

$$p_{\text{new}} := p(c_i \neq c_k \forall k \leq \kappa | \mathbf{c}_{-i})$$

where κ is the current number of clusters. Note that

$$p(c_i \neq c_k \forall k \leq \kappa | \mathbf{c}_{-i}) = p(c_i = c_k \forall k > \kappa | \mathbf{c}_{-i}),$$

and that $n_k = 0 \forall k > \kappa$. Plugging into (4), the probability of opening a new cluster becomes

$$\begin{aligned} p_{\text{new}} = p(c_i = c_k \forall k > \kappa | \mathbf{c}_{-i}) &= \lim_{K\to\infty} \sum_{k=\kappa+1}^{K} \frac{\alpha/K}{N + \alpha - 1} \\ &= \lim_{K\to\infty} (K - (\kappa + 1)) \frac{\alpha/K}{N + \alpha - 1} \\ &= \frac{\alpha}{N + \alpha - 1}. \end{aligned} \tag{5}$$

What if we are moving more than one datapoint at once, as is the case when we combine segments containing multiple SNPs? In that case, we have

$$p(\underbrace{\{c_{z_i}\}_{i=1}^{M}}_{\mathbf{c}_{\text{move}}} = \tilde{c} | \underbrace{\mathbf{c} \setminus \mathbf{c}_{\text{move}}}_{\mathbf{c}_{-\text{move}}}) = \frac{p(\overbrace{c_1, \ldots, c_{z_1} = \tilde{c}, c_{z_2} = \tilde{c}, \ldots, c_{z_M} = \tilde{c}, \ldots, c_N}^{\mathbf{c}})}{p(\mathbf{c}_{-\text{move}})}.$$

The math proceeds similarly as when moving a single datapoint. Again, WLOG, let $z_1, \ldots, z_M = 1, \ldots, M$ and $\tilde{c} = 1$. Then

$$\begin{aligned} p(\mathbf{c}) &= \int d\mathbf{p} \left[\prod_{i=1}^{M} \text{Cat}(c_i = 1|\mathbf{p}) \right] \left[\prod_{i=M+1}^{N} \text{Cat}(c_i|\mathbf{p}) \right] \text{SymDir}(\mathbf{p}|\alpha/K) \\ &\propto \int d\mathbf{p}\, p_1^{M+n_1+\alpha/K-1} \prod_{k=2}^{K} p_k^{n_k+\alpha/K-1} \\ &= \frac{\Gamma(M + n_1 + \alpha/K) \prod_{k=2}^{K} \Gamma(n_k + \alpha/K)}{\Gamma(N + \alpha)} \end{aligned} \tag{6}$$

and

$$
\begin{aligned}
p(\mathbf{c}_{-\text{move}}) &= \int d\mathbf{p} \left[\prod_{i=M+1}^{N} \text{Cat}(c_i|\mathbf{p}) \right] \text{SymDir}(\mathbf{p}|\alpha/K) \\
&\propto \int d\mathbf{p}\, p_1^{n_1+\alpha/K-1} \prod_{k=2}^{K} p_k^{n_k+\alpha/K-1} \\
&= \frac{\prod_{k=1}^{K} \Gamma(n_k + \alpha/K)}{\Gamma(N+\alpha-M)}. \qquad (7)
\end{aligned}
$$

The ratio of (6) and (7) is

$$
\frac{p(\mathbf{c})}{p(\mathbf{c}_{-\text{move}})} = \frac{\Gamma(M + n_1 + \alpha/K)\Gamma(N+\alpha-M)}{\Gamma(n_1+\alpha/K)\Gamma(N+\alpha)}. \qquad (8)
$$

To further simplify this, recall that

$$
\frac{\Gamma(x-y)}{\Gamma(x)} = \frac{1}{\prod_{i=1}^{y}(x-i)} \qquad \frac{\Gamma(x+y)}{\Gamma(x)} = \prod_{i=0}^{y-1}(x+i) \qquad x \in \mathbb{R}^+, y \in \mathbb{Z}^+,
$$

so (8) becomes

$$
\begin{aligned}
p(\mathbf{c}_{\text{move}}|\mathbf{c}_{-\text{move}}) &= \lim_{K\to\infty} \frac{\Gamma(M + n_{\tilde{c}} + \alpha/K)\Gamma(N+\alpha-M)}{\Gamma(n_{\tilde{c}}+\alpha/K)\Gamma(N+\alpha)} \\
&= \frac{\prod_{i=0}^{M-1}(n_{\tilde{c}}+i)}{\prod_{i=1}^{M}(N+\alpha-i)}.
\end{aligned}
$$

In practice, it is computationally simpler to use the expression in (8) (after taking the limit $K \to \infty$), since numerical computing packages provide efficient functions for evaluating $\log\Gamma(x)$ directly.

The probability of opening a new cluster is thus

$$
\begin{aligned}
p(\{c_{z_i}\}_{i=1}^{M} = c_k \forall k > \kappa | \mathbf{c}_{-\text{move}}) &= \lim_{K\to\infty} \sum_{k=\kappa+1}^{K} \frac{\Gamma(M+\alpha/K)\Gamma(N+\alpha-M)}{\Gamma(\alpha/K)\Gamma(N+\alpha)} \\
&= \frac{\Gamma(N+\alpha-M)}{\Gamma(N+\alpha)} \lim_{K\to\infty} \sum_{k=\kappa+1}^{K} \frac{\Gamma(M+\alpha/K)}{\Gamma(\alpha/K)} \\
&= \frac{\Gamma(N+\alpha-M)}{\Gamma(N+\alpha)} \lim_{K\to\infty} (K-(\kappa+1)) \prod_{i=0}^{M-1} (\alpha/K+i) \\
&= \frac{\Gamma(N+\alpha-M)}{\Gamma(N+\alpha)} \alpha\Gamma(M).
\end{aligned}
$$

Unlike in the case of moving a single datapoint at a time, we cannot use the probabilities $p(\mathbf{c}_{\text{move}}|\mathbf{c}_{-\text{move}})$ as-is. The event space of $p(\mathbf{c}_{\text{move}})$ comprises all possible assignments of each datapoint in $\mathbf{c}_{\text{move}}$, with probabilities normalized with respect to this event space. For example, suppose we are moving M datapoints $c_{z_1}, c_{z_2}, \ldots, c_{z_M}$. The event space is the Cartesian product $\lim_{K\to\infty}[\{c_{z_1} = k\}_{k=1}^{K} \times \{c_{z_2} = k\}_{k=1}^{K} \times \cdots \times \{c_{z_M} = k\}_{k=1}^{K}]$. However, we are only interested in the event space for which k is constant across all datapoints, i.e. $\lim_{K\to\infty}\{c_{z_1} = k, c_{z_2} = k, \ldots, c_{z_M} = k\}_{k=1}^{K}$. We must re-normalize our probabilities, like so:

$$\begin{aligned} p'(\{c_{z_i}\}_{i=1}^{M} = \tilde{c}|\mathbf{c}_{-\text{move}}) &\equiv p(\{c_{z_i}\}_{i=1}^{M} = \tilde{c}|\mathbf{c}_{-\text{move}}, c_{z_1} = c_{z_2} = \cdots = c_{z_M}) \\ &= \frac{p(\{c_{z_i}\}_{i=1}^{M} = \tilde{c}|\mathbf{c}_{-\text{move}})}{\sum_{k=1}^{\kappa} p(\{c_{z_i}\}_{i=1}^{M} = c_k|\mathbf{c}_{-\text{move}}) + p_{\text{new}}}. \end{aligned}$$

Appendix C ACDP Implementation Details

C.1 Distinguishing clonal vs. subclonal events

Most sCNAs are clonal, so most HSACR segments will be assigned to a handful of ACDP clusters corresponding to the clonal states of $0, 1, 2, \ldots$ copies of each allele. The Dirichlet process prior favors a uniform distribution of cluster sizes and penalizes small clusters, which often forces small clusters corresponding to subclonal copy states to merge with nearby clonal clusters, resulting in a loss of subclonal calling accuracy. To mitigate this, we separate clonal and subclonal HSACR segments and run the ACDP clustering on them separately. We separate clonal from subclonal segments using a two-step process. The first step relies solely on AI, which provides an extremely clean signal but does not unambiguously indicate HSACR. The second step looks at HSACR, which is somewhat noisier due to TCR being noisier than AI, but lacks the ambiguity of AI.

C.2 AI step

A clonal segment has allelic imbalance

$$\alpha(p, n_a, \tau) = \frac{1 + p(n_a - 1)}{2(1 - p) + p\tau}$$

where $p \in [0, 1]$ is the purity of the sample, $\tau \in \mathbb{Z}^+$ is the clonal ploidy of the segment, and $n_a \in \{0, \ldots, \lfloor \tau/2 \rfloor\}$ is the clonal copy number of homolog A for the segment.

The likelihood of the ith ACR segment Ξ_i is

$$\mathcal{L}_i(p, \tau, n_a|A_i, B_i) = \text{Beta}(\alpha(p, n_a, \tau); A_i + 1, B_i + 1)$$

where A_i and B_i are the read counts assigned to the A and B homologs within the Ξ_i. For a given purity p^*, we maximize for each segment

$$\hat{\tau}_i, \hat{n}_{a,i} = \arg\max_{\tau, \text{n}_\text{a}} \mathcal{L}_i(\tau, n_a|A_i, B_i, p^*)$$

over all $\tau \in \{0 \dots 7\}$, $n_a \in \{0 \dots 3\}$. We then maximize the overall likelihood

$$\mathcal{L}(p^*|\{A\},\{B\}) = \prod_i \mathcal{L}_i(\hat{\tau}_i, \hat{n}_{a,i}|A_i, B_i, p^*)$$

over all purity values $p^* \in [0,1]$ with step size 10^{-4}, yielding optimal purity

$$\hat{p}^* = \arg\max_{\mathrm{p}^*} \mathcal{L}(p^*|\{A\},\{B\}).$$

Given $\hat{p}^*$, we sort the ACR segments by their genomic mass (length times ploidy) $\Xi_{s_1}, \dots, \Xi_{s_N}$ and compute the cumulative log-likelihood

$$\mathcal{L}_{\mathrm{cum}}(k) = \sum_{i=1}^{k} \log \mathcal{L}_{s_i}(\hat{p}^*, \hat{n}_{a,s_i}, \hat{\tau}_{s_i}|A_{s_i}, B_{s_i}) \qquad k \leq s_N$$

and find the elbow point $\iota(k)$ of this curve as a threshold to identify segments that are confidently clonal. These segments undergo a full run of the ACDP, generating clusters $\mathbf{C}^* = \{C_1^* \dots C_{|\mathbf{C}^*|}^*\}$. We assume that each cluster corresponds to a distinct clonal allelic copynumber state. These will serve as a seed to identify additional clonal segments based on their full ACR values.

C.3 HSACR step

Recall that HSACR has units of sequencing coverage, since an HSACR segment ξ is the product of the segment's AI (which is a dimensionless scale factor) and its TCR (which HapASeg measures in sequencing coverage). We must associate allelic coverage levels with discrete clonal allelic copynumber states.

We begin by taking all clusters $\mathbf{C}^*$ from the previous step, and parameterize the mean normal likelihood associated with each cluster C_k^* in terms of purity $\hat{p}^*$, allelic copynumber n_k, and coverage scale factor ϕ,

$$\mathcal{L}_k(n_k, \phi|\{\tilde{x} \mid i \in C_k^*\}, \hat{p}^*, \sigma_k) = \prod_{i \in C_k^*} \mathcal{N}(\tilde{x}_i; \mu(n_k, \phi, \hat{p}^*), \sigma_k),$$

$$\mu(n_k, \phi, \hat{p}^*) = \phi(1-\hat{p}^*) + n_k \hat{p}^*.$$

σ_k is simply the cluster's empirical standard deviation $\sqrt{\mathrm{Var}(\{\tilde{x}_i | i \in C_k^*\})}$, and is not a free parameter.

We find the optimal scale factor $\hat{\phi}$ by performing a grid search similar to the one used to find optimal purity $\hat{p}^*$, by optimizing each C_k^* for a given ϕ with respect to $n_k \in \{0 \dots 5\}$,

$$\hat{n}_k = \arg\max_{\mathrm{n_k}} \mathcal{L}_k(n_k|\{\tilde{x} \mid i \in C_k^*\}, \hat{p}^*, \sigma_k, \phi),$$

and then optimizing across all clusters in $\mathbf{C}^*$ with respect to ϕ,

$$\hat{\phi} = \arg\max_{\phi} \prod_{k=1}^{|\mathbf{C}^*|} \mathcal{L}_k(\phi | \{\tilde{x} \mid i \in C_k^*\}, \hat{p}^*, \hat{n}_k, \sigma_k).$$

We then asses the clonality of clusters in ξ_{clonal} by comparing the Gaussian likelihood of data points

$$p(C_\ell | \mu^{m^*}_{\eta^*, p^*}, \sigma_\ell) = \sum_{x_i \in \mathcal{R}(C_\ell)} \log \mathcal{N}(x_i; \mu^{m^*}_{\eta, p^*}, \sigma_\ell)$$

where m^* is the most likely allelic copy level for cluster ℓ with a null model

$$p_0(C_\ell) = \sum_{x_i \in \mathcal{R}(C_\ell)} \log U(x_i; \min(\mathcal{R}(C_\ell)), \max(\mathcal{R}(C_\ell))$$

Which is a uniform distribution with the same domain as the data points in D_{C_ℓ}. Clusters where $p(C_\ell | \mu^{m^*}_{\eta^*, p^*}, \sigma_\ell) > p_0(C_\ell)$ and which comprise greater than 1% of the total coverage windows are labeled as confidently clonal, denoted $\mathfrak{C}$. We then construct our final set of clonal $\mathfrak{S}_{clonal}$ and sub-clonal segments $\mathfrak{S}_{subclonal}$ by initially setting $\mathfrak{S}_{clonal} = \mathcal{S}(\mathfrak{C})$ and $\mathfrak{S}_{subclonal} = \mathcal{S}(\xi_{clonal} \setminus \mathfrak{C})$. We assign the remaining segments that were initially deemed non-clonal $S_j, S_j \notin \mathcal{S}(\xi_{clonal})$ to $\mathfrak{S}_{clonal}$ or $\mathfrak{S}_{subclonal}$ using the same null likelihood approach as in (REF), that is,

$$S_j \to \mathfrak{S}_{clonal} \quad \text{iff} \quad p(S_j) > p_0(S_j)$$

With our models now defined on the segment level and Gaussian likelihood model is defined using $\mathfrak{C}$, i.e.

$$p(S_j) = \max_{C_w} \sum_{x_i \in S_j} \log \mathcal{N}(x_i; \overline{\mathcal{R}(C_w)}, \sqrt{\text{Var}(\mathcal{R}(C_w))}, C_w \in \mathfrak{C}$$

$$p_0(S_j) = \sum_{x_i \in \mathcal{R}(S_j)} \log U(x_i; \min(\mathcal{R}(S_j), \max(\mathcal{R}(S_j))$$

With $\mathfrak{S}_{clonal}$ and $\mathfrak{S}_{subclonal}$ in hand, we finally run ACDP clustering on each segment set independently, and combine the resulting clusters from the draw from each chain after burnin to produce our final results.

Appendix D Benchmarking Details

D.1 Running Competing Methods

D.2 ASCAT

We installed ASCAT v3.1.1 from the github repository along with its dependencies into a docker container available in the google container repository. ASCAT only uses allele

count data to perform copy number calling, and provides its own basic allele counting method. We tested the difference between allele counts at NA12878 variant sites from Mutect1 and ASCAT prepare.HTS and found very strong agreement. We chose to move forward with the Mutect counts since these counts were already generated for use by HapASeg. Following our protocol for HapASeg, we removed sites with greater than 5% MAPQ0 reads or greater than 10% of reads being filtered by Mutect standard filters in order to remove likely sequencing artifacts. We then filtered on the ASCAT's WGS loci list as recommended in their documentation. The matched normal allele counts were also computed using Mutect and were merged with the passing "tumor" sites. Allele counts for the simulations were generated using the procedure described above and the resulting counts were converted to LogR and AAF format for input to ASCAT. We ran ASCAT as its documentation recommended with default settings. This included correcting the the LogR file using the GC and RT correction files provided in their repo. The raw allelic coverage estimates for downstream analysis were taken from the segments_raw.txt ASCAT output file.

D.3 Facets

We installed Facets v0.6.2 from the github repository along with its dependencies into a docker container available in the google container repository. Facets only uses allele count data to perform copy number calling, and provides its own basic allele counting method. We tested the difference between allele counts at NA12878 variant sites from Mutect1 and Facets SNP Pileup and found very strong agreement. We chose to move forward with the Mutect counts since these counts were already generated for use by HapASeg. Following our protocol for HapASeg, we removed sites with greater than 5% MAPQ0 reads or greater than 10% of reads being filtered by Mutect standard filters in order to remove likely sequencing artifacts. We then filtered on the Facets WGS loci list from dbSNP as recommended in their documentation. The matched normal allele counts were also computed using Mutect and were merged with the passing "tumor" sites. Allele counts for the simulations were generated using the procedure described above and the resulting counts were converted to the snp counts format for Facets. We ran Facets as its documentation recommended with default settings of minimum snp depth of 10 and critical value of 150.

Facets does not return, or in fact even compute, raw allele specific coverage as part of its algorithm. Instead, it returns estimated allelic copy number that it computes after fitting a comb. We therefore use these smoothed estimates, which can be found in the facets segfile generated by the `(emcncf` function, for downstream comparison.

D.4 GATK

For our GATK benchmarking analyses we followed the procedure on the GATK CNV tutorial. We used the publicly available `broadinstitute/gatk:4.0.1.1` docker container, which was also used in the tutorials, for all of our GATK analyses. We used the standard public hg38 WGS calling intervals file with zero padding for our WGS preprocessing and the TWIST target intervals with a 250bp padding for WES. We

excluded the sex chromosomes for all analyses since their analysis would require generating sex specific PoNs. Allele counts were computed using CollectAllelicCounts on the set of NA12878 variants and coverage was computed with CollectFragmentCounts using the recommended interval size of 1kb. Simulated tumor allele counts and coverage were generated with the procedure described above and underwent PoN and GC correction using the DenoiseReadCounts function. These denoised counts were then used as input for copy number calling by the ModelSegements function, which was run with default options. Downstream results were calculated from the `modelFinal.seg` segmentation file.

To generate the PoN files for denoising we first ran CollectFragmentCounts on each of the panel BAMs using the same WGS/WES specific interval list as described above. The fragment counts were then aggregated into a PoN using the CreateReadCount-PanelOfNormals function with the minimum interval median percentile option set to 5 and all other options set to default. The PoN files are available in the benchmarking data bucket.

D.5 HapASeg

Allele counts and coverage counts were generated using Mutect on the NA12878 variants and covcollect with 2kb intervals. Mutect allele counts were processed to remove artifacts based on MAPQ and Mutect default filtering, but were only filtered for het site status if the normal reference sample was derived from NA12878, since the FFPE normals have a separate set of variants. While the het site filtering is not essential, since the simulated tumor allele counts are based on the total read depth at the variant depths rather than the individual allele counts, we felt it was appropriate to use the same het site filtering when possible in order to provide HapASeg with a representative number of het sites. The coverage counts were normalized by the average fragment length as described in (REF coverage MCMC) and passed with the allele counts to the tumor simulation procedure.

The resulting simulated files were passed to HapASeg and the method was run using the default settings. We ran our downstream analyses on both the ACDP clustered `hapaseg_segfile.txt` and the unclustered `hapaseg_skip_acdp_segfile.txt` in order to evaluate the accuracy of each strategy over the range of simulated sample characteristics.

D.6 HATCHet

We installed HATCHet v1.1.1 from github repository along with dependencies in a docker container publicly available on google container repository. We genotyped each variant site in NA12878 using the `genotype-snps` function using the default parameters. Allele counts and coverage counts were computed using the `count-alleles` and `count-reads` functions using default settings, respectively. HATCHet also optionally performs phasing to improve calling accuracy. We enabled phasing on the genotype files for all samples. The phased SNP result files were saved for each sample type for use during copy number calling. The allele count and read count files were passed as inputs to the tumor simulation procedure. The simulated counts were then passed

to `combine-counts` to merge coverage bins and then to `cluster-bins` to perform the copy number calling. Allelic coverage estimates were extracted from the resulting clustered bins and cluster segments files. We used the cluster estimates assigned to each segment rather than the segment specific estimates, as is presented in the method's final result plots. These allelic coverage estimates were used for downstream comparisons.

Appendix E Miscellaneous calculations

E.1 Expected length before phase switch

Eagle2 reports a 0.5% error rate [13], which is defined as the number of phase switches required to obtain the true haplotype, normalized by the number of possible switch error postions, which is simply the number of homozygous markers in the individual[28]. There are on average 404,700 homozygous SNPs recently reported by [29]. This results in an expected switch error every 14.2Mb.