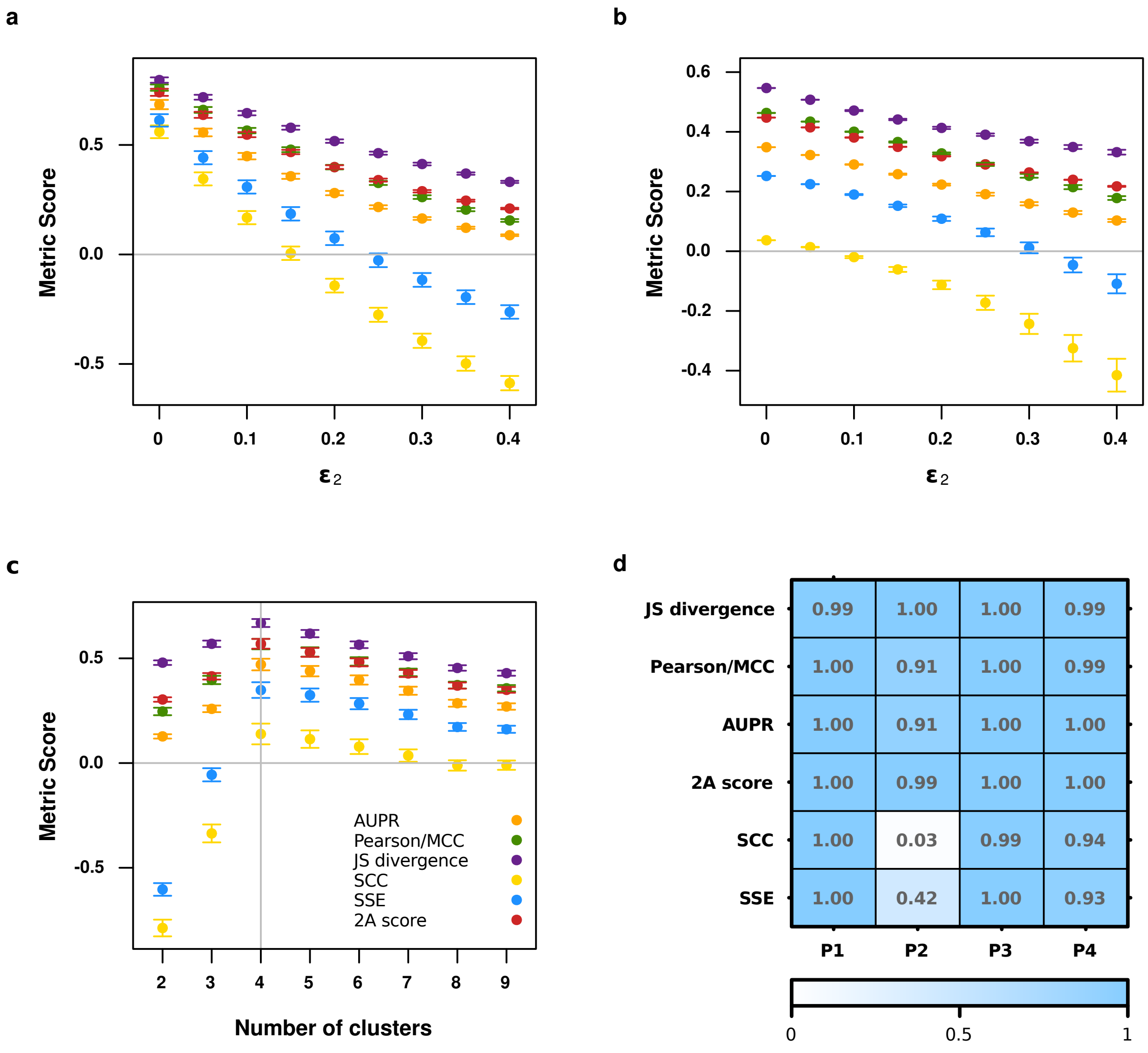
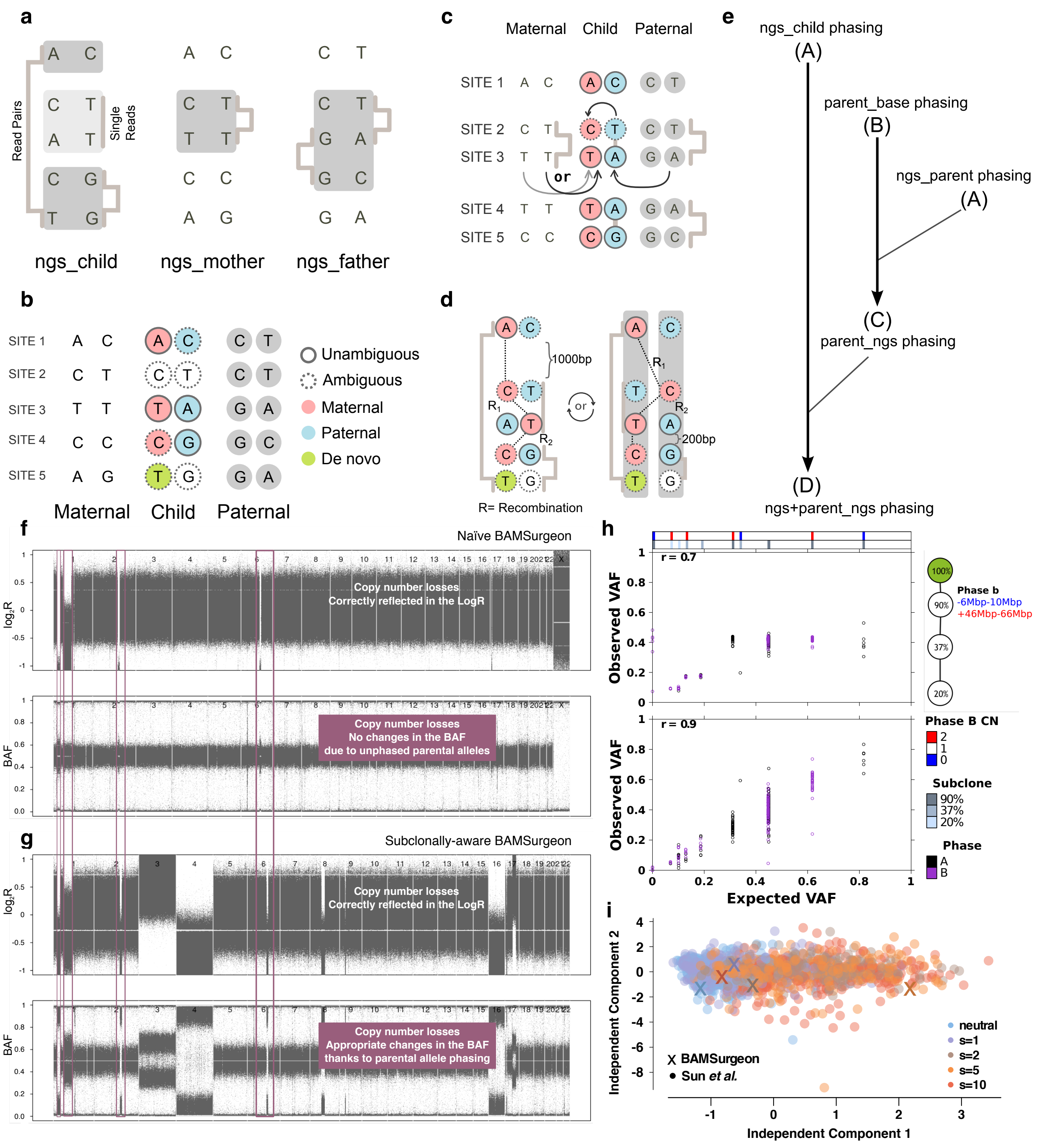


Supplementary Figure 1

Behaviour of Scoring Metrics

(a) The mean score for each candidate mutation cluster assignment score (SC2) metric considered with an increasing proportion of mutations assigned to wrong useful clusters with bars indicating standard error (n=24). **(b)** The mean score and standard error for each candidate SC2 metric considered with an increasing proportion of mutations in noise clusters (n=3). **(c)** The mean score and standard error for each candidate SC2 metric considered as the number of predicted clusters increases (n=27). The true number of clusters (four) is marked by the vertical line. Excess clusters retain correct co-clustering and are subsets of the true clusters. **(d)** For each potential SC2 scoring metric, the proportion of simulation runs that satisfied each of the four desirable metric properties for a given simulation parameter setting (n=2 x 105, 8 x 105, 2.8 x 106, 4.3 x 106 for P1-P4, respectively). Each property is tested by fixing all but one of the simulation parameters and then looking at the effect of changing the fourth parameter on the metric score.



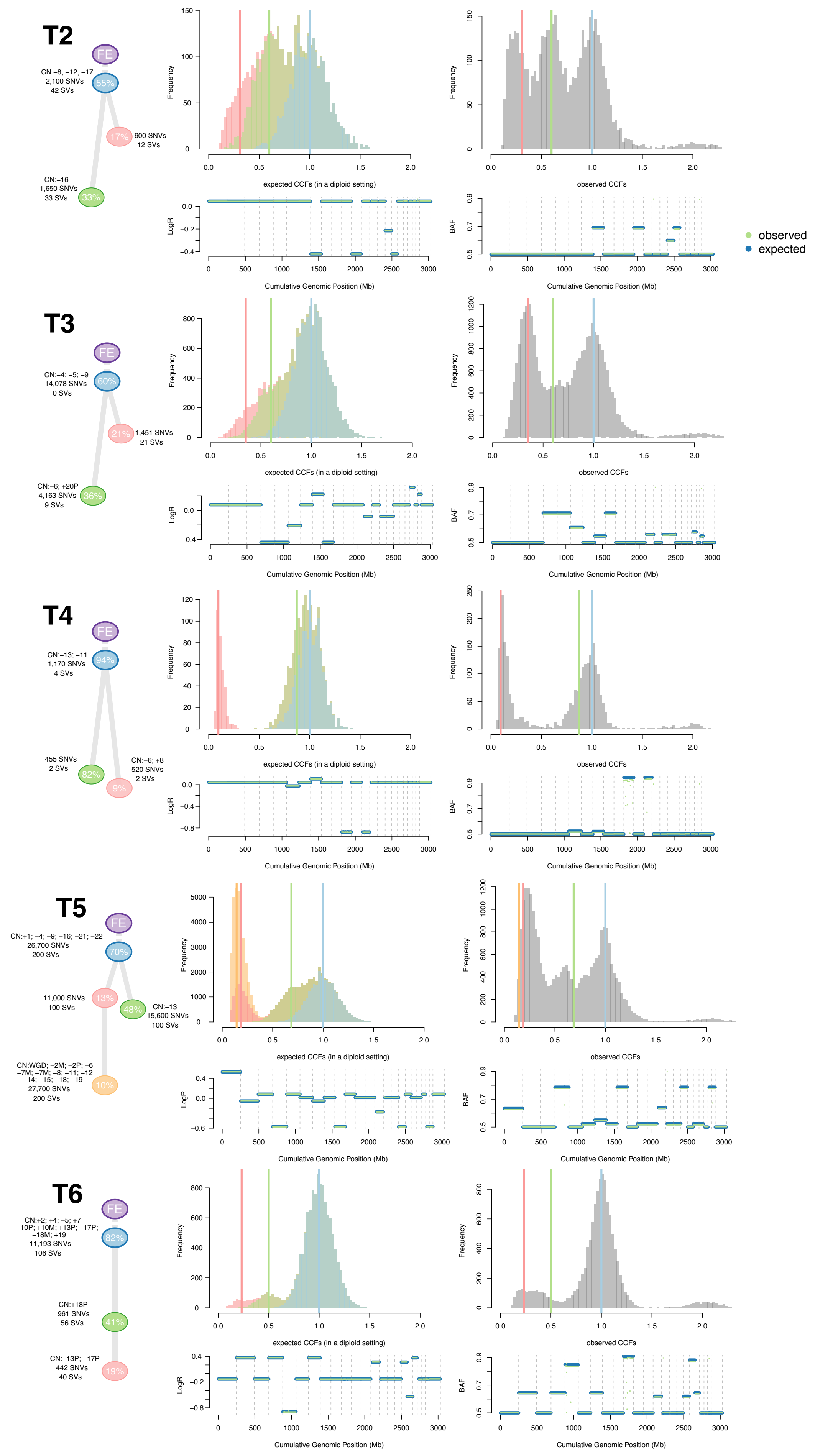
Supplementary Figure 2

Simulating BAMs with subclonal structure

(a) Example of the PhaseTools algorithm constructing an extended phase set from four heterozygous sites by leveraging NGS and parent phasing. ngs_phasing of 5 heterozygous sites in the child and the corresponding NGS-phased sites in the mother and father, shown with informative NGS fragments. Heterozygote variants boxed together represent phase sets. There is not enough information to construct a single phase set. **(b)** parent_base phasing uses parental genotypes to assign parent of origin to the 5 hets in the child. Hets 2 and 5 remain unresolved while heterozygotes 1, 3, and 4 show at least one unambiguous parent of origin. **(c)** parent_ngs phasing extends parent_base phasing with parental NGS

(continued)

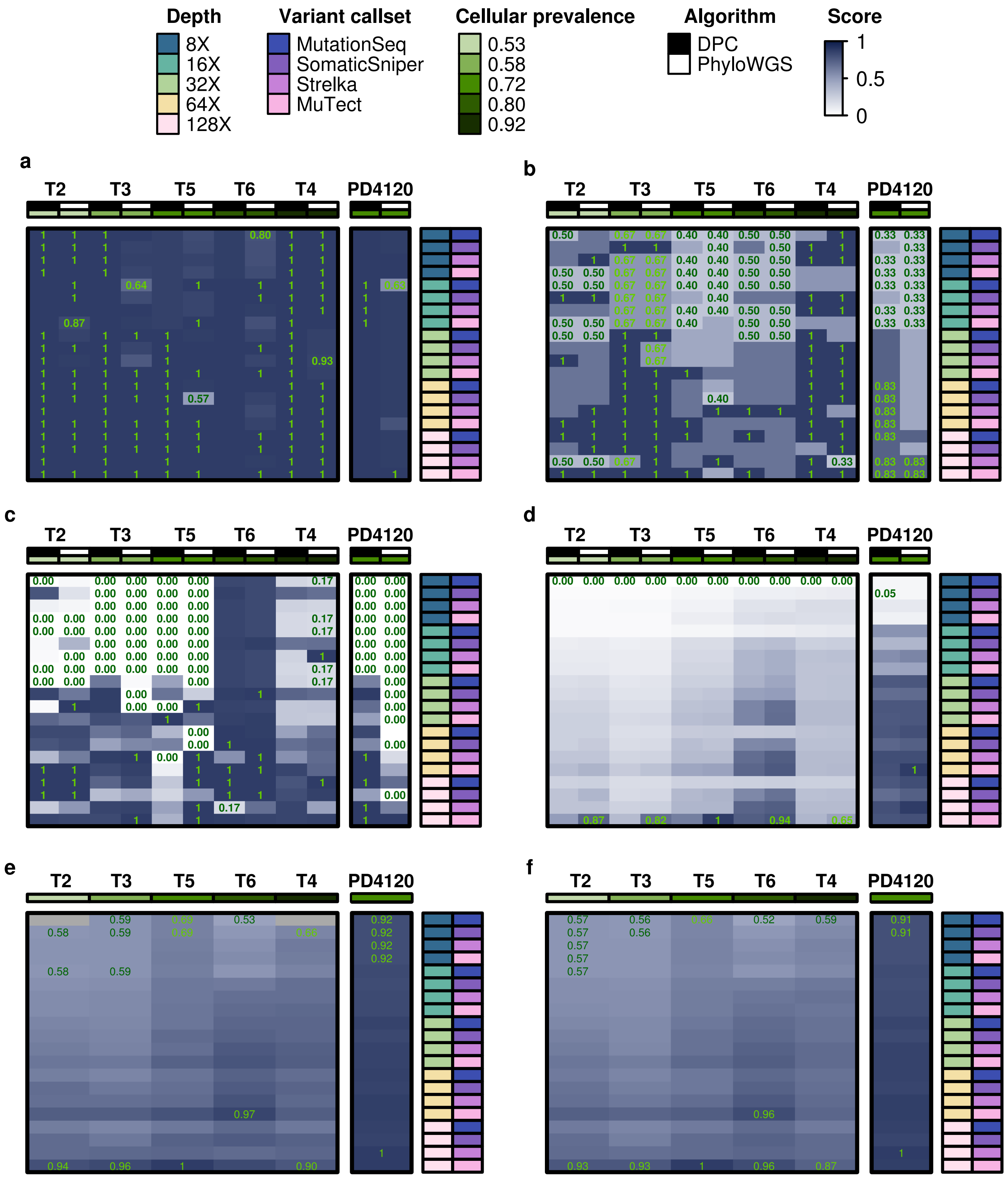
fragments from ngs_phasing. The linked NGS fragment in sites 2 and 3 (T, T) of the maternal genotype is not informative as site 3 is homozygous, however the linked NGS fragment in sites 2 and 3 (T and A) of the paternal genotype is heterozygous and therefore informative. The phasing proposed by ngs_father of sites 3 and 4 (GG/AC) contradicts parent of origin information in hets 3 and 4 (A and G). This event is recognised as a pre-meiosis recombination event in the child and the ngs_father phasing is ignored.**(d)** ngs+parent_ngs phasing extends ngs_child phasing with parent_ngs, giving priority to ngs_child phasing. NGS fragments such as hets 2 and 3 (T and T) take precedent over any phasing assigned by parent_ngs phasing see hets 2 and 3 (C and T) and indicate probable recombination events (shown with diagonal lines). Two possible sets of recombination events are shown. The proximity between phased heterozygotes determines which recombination events are most probable. Here, the recombination events shown on the right are selected, as recombination between sites 1 and 2 is more likely than recombination between sites 3 and 4, as sites 1 and 2 are further apart. The final phase sets are shown. **(e)** Schematic of phase-set reconstruction. Priority is given to procedures on the left. **(f)** Genome-wide logR and BAF tracks from a simulated BAM using the original "naive" BAMSurgeon. **(g)** Same as **(a)** with the new proposed BAMSurgeon pipeline and additional whole-chromosomal events. **(h)** Comparison of expected and observed VAFs for SNVs on chromosome 17 of a tumour simulated with the "naive" BAMSurgeon (top panel) and the new pipeline (bottom panel) with Pearson correlations (n=317). As naive BAMSurgeon does not simulate allele-specific gains and losses, copy number alterations in one allele do not produce the expected allele-frequency changes for each phase. Tumour structure showing a clonal deletion and duplication on the chromosome is specified in the top right panel. Allele-specific copy number events for phase B (phase A has no copy number alterations) and the subclone where they occur are shown in the top heatmap. Expected VAFs were calculated by summing CNA-adjusted CCF of the subclones were the SNV was present, adjusting for the frequency of the SNV in those subclones and standardizing by the total CNA-adjusted depth in that region. **(i)** Independent component analysis of the intra-tumour heterogeneity metrics presented in Sun et al. on n=1,366 3D simulated training tumours (Sun et al. simulator) and 5x3-region BAMs derived from 5 3D test tumours with SNVs simulated with increasing selection *s* (Sun et al. simulator) using BAMSurgeon. The 5 regions were classified correctly using an SVM predictor (Supplementary Note 2).


T2
FE
CN:−8; −12; −17
2,100 SNVs
42 SVs
55%
17%
600 SNVs
12 SVs
CN:−16
1,650 SNVs
33 SVs
33%
T3
FE
CN:−4; −5; −9
14,078 SNVs
0 SVs
60%
21%
1,451 SNVs
21 SVs
CN:−6; +20P
4,163 SNVs
9 SVs
36%
T4
FE
CN:−13; −11
1,170 SNVs
4 SVs
94%
455 SNVs
2 SVs
82%
CN:−6; +8
520 SNVs
2 SVs
9%
T5
FE
CN:+1; −4; −9; −16; −21; −22
26,700 SNVs
200 SVs
70%
11,000 SNVs
100 SVs
13%
48%
CN:−13
15,600 SNVs
100 SVs
CN:WGD; −2M; −2P; −6
−7M; −7M; −8; −11; −12
−14; −15; −18; −19
27,700 SNVs
200 SVs
10%
T6
FE
CN:+2; +4; −5; +7
−10P; +10M; +13P; −17P;
−18M; +19
11,193 SNVs
106 SVs
82%
CN:+18P
961 SNVs
56 SVs
41%
CN:−13P; −17P
442 SNVs
40 SVs
19%
Frequency
expected CCFs (in a diploid setting)
observed CCFs
LogR
BAF
Cumulative Genomic Position (Mb)
observed
expected

Supplementary Figure 3

Real and Simulated Subclonal Structures

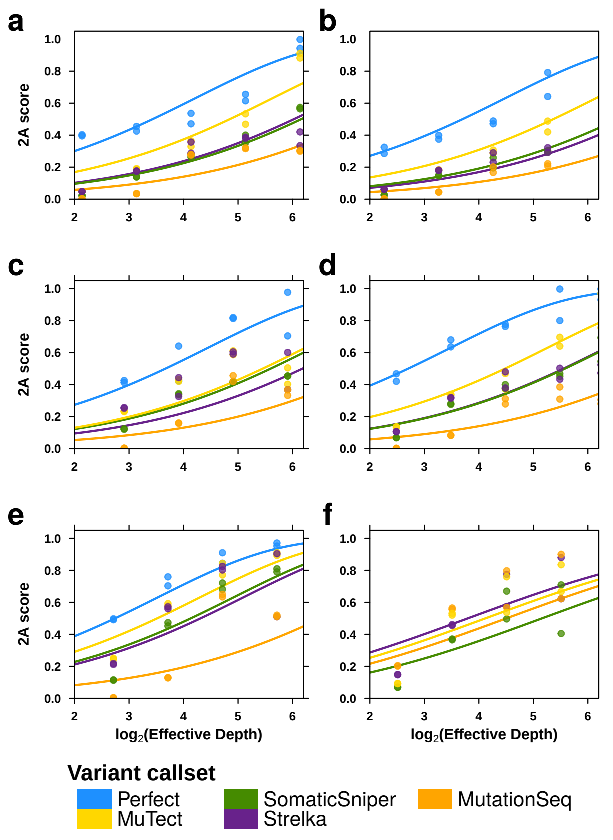
True subclonal structures of the simulated tumours (T2, T3, T4, T5 and T6) that were simulated with their desired and observed variant allele frequency histograms and logR profiles. In each panel, we show the phylogenetic tree, inspired by published reconstructed tumours, and the mutations associated with each (sub)clone. The top figures compared expected cancer cell fractions of the SNVs under a diploid setting, against the inferred cancer cell fractions from the simulated data. T5, for which the inferred purity is over-estimated due to the limitations of the copy number detection algorithm to identify subclonal whole genome duplication, shows an observed space that departs from the expected. The bottom figures compare the observed and expected BAF and logR of the genomic segments identified by the copy number detection algorithm.


Depth
8X
16X
32X
64X
128X
Variant callset
MutationSeq
SomaticSniper
Strelka
MuTect
Cellular prevalence
0.53
0.58
0.72
0.80
0.92
Algorithm
DPC
PhyloWGS
Score
1
0.5
0
a
b
c
d
e
f
T2
T3
T5
T6
T4
PD4120

Supplementary Figure 4

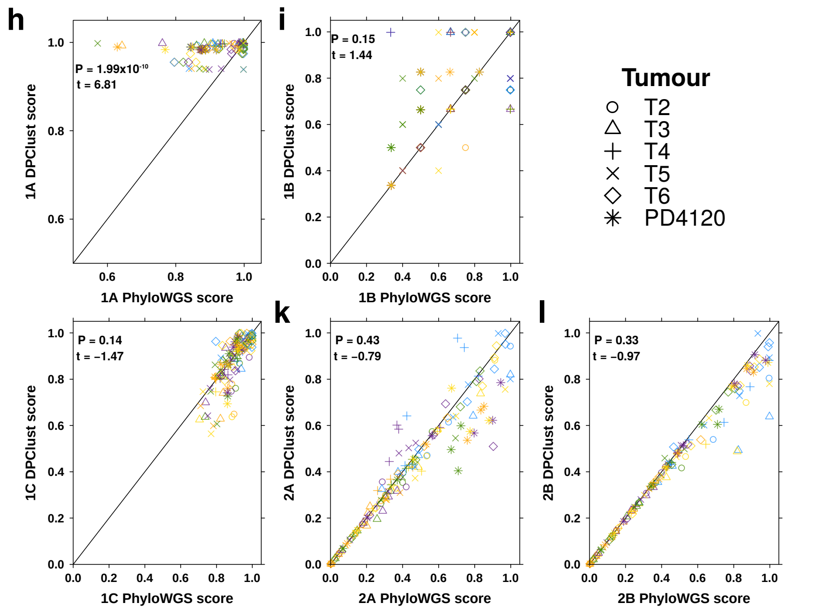
Subclonal Reconstruction Scores

Subclonal reconstruction scores based on the five tumours with each somatic SNV detection algorithm-depth-algorithm combination and down-sampled CNAs (n=290). All scores are normalised to the score of the best performing algorithm using perfect calls at the full tumour depth. Scores exceeding this baseline likely represent noise or overfitting and were capped at 1. **(a)** Scores for 1A are uniformly high. **(b)** Scores for SC1B improve with depth and but not continuous as the metric reflects a true proportion. **(c)** Clonal fraction is low below 32X but rapidly increases at higher depths, with some inter-tumour variability. **(d,f)** Scores for SC2B **(d)** and SC3B **(f)** closely mirror those of SC2A and SC3A **(e)**, respectively.



g

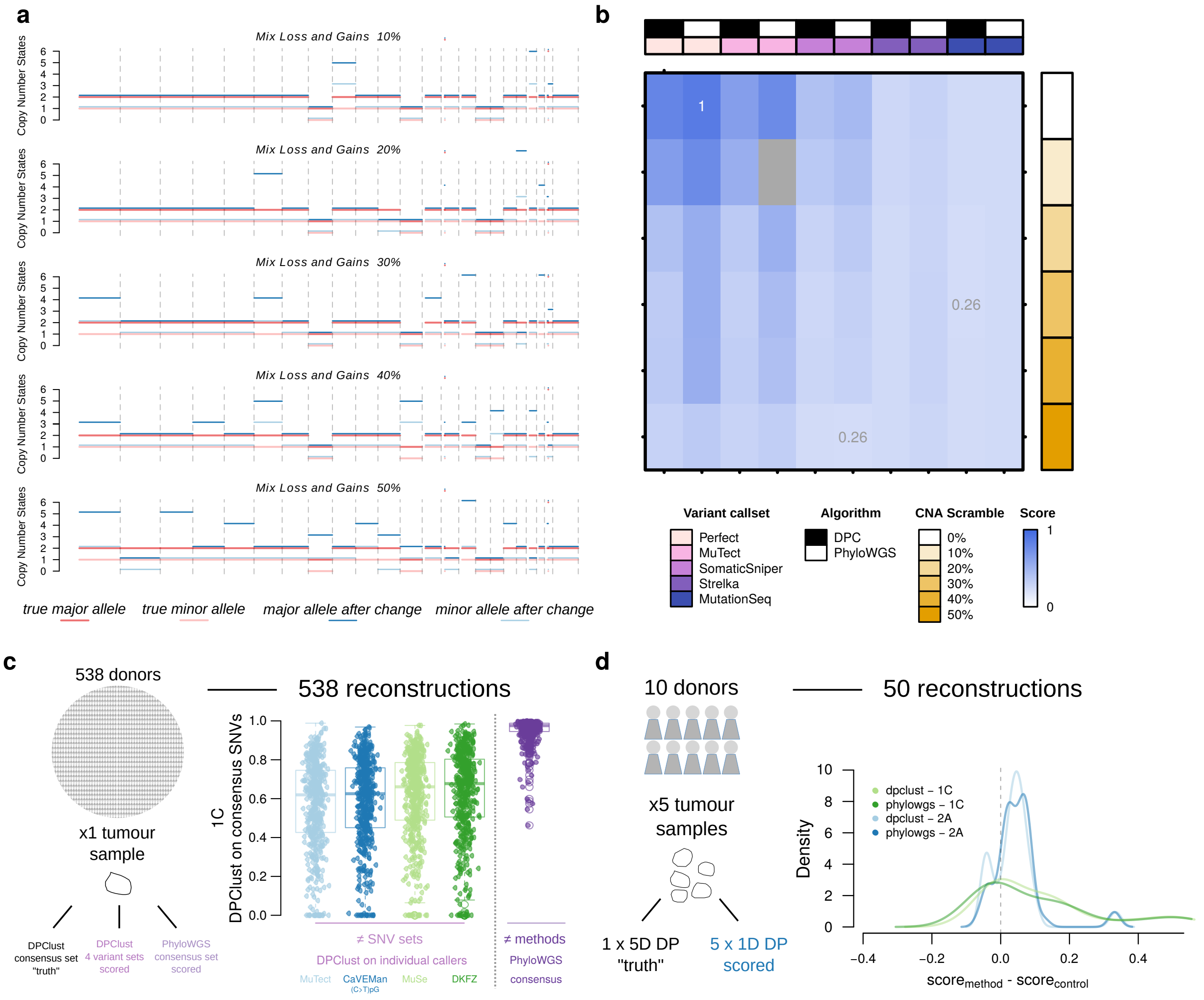
PhyloWGS	DPClust
Assigns CNAs to clusters	Does not assign CNAs to clusters
Estimates multiplicity of mutations during Markov chain Monte–Carlo (MCMC)	Uses most likely multiplicity of each mutation during MCMC
Reconstructs phylogenies during MCMC	Reconstructs phylogenies after MCMC
Estimates purity from SNVs and CNAs	Estimates purity from CNAs



Supplementary Figure 5

Covariates of Scoring Accuracy

SC2A score increases with effective depth for all tumours but the effect of the somatic SNV detection algorithm depends on the tumour. **(a)** T2 **(b)** T3 **(c)** T4 **(d)** T5 **(e)** T6 **(f)** real PD4120. **(g)** A summary of the differences between DPClust and PhyloWGS. PhyloWGS incorporates CNAs and phylogenetic structure into the generative model sampled through Markov chain Monte-Carlo (MCMC). **(h-l)** Comparison of subclonal reconstruction scores for each sub-challenge using PhyloWGS (x-axis) and DPClust (y-axis). Somatic SNV detection algorithms are coded by colour and tumours are coded by symbol. P-values and effect size for a t-test between algorithms on score are shown (n=580 for each sub-challenge).



Supplementary Figure 6

Effects of Noise in CNA Detection on Reconstruction Accuracy

Effect of proportion of CNAs scrambled on a representative tumour (T2) at 128x sequencing coverage. **(a)** Allele specific copy number profiles as increasing proportions of CNAs are randomly assigned to a wrong copy number state. **(b)** Effect of increasing the proportion of wrongly assigned CNAs on SC2a scores at 128x for different somatic SNV detection algorithms and SRC algorithms. **(c)** Deriving an imperfect "truth" from real data and scoring runs against it. We took data from 538 donors from the PCAWG study41 and executed DPClust on different sets of SNVs: a consensus set of SNVs was used as "truth" and three individual somatic SNV detection pipelines (MuTect, DKFZ and Sanger on mutations in a (C>T)pG context). We also executed PhyloWGS on the consensus sets. We then scored SC1C for each run against the "truth". **(d)** We took data from 10 donors from the PCAWG study with at least 5 tumour regions or metastases sequenced and ran DPClust in 5 dimensions to derive a "truth" from it. We scored each one-dimensional DPClust run on individual region against the "truth" derived from the multi-dimensional run for SC1C and SC2A and compared them against scores obtained from randomised 1C and 2A inputs (Online Methods). Upper and lower box limits show the 25th and 75th quartile, respectively and the line indicates the median. Whiskers extend to 1.5 x the length of the box.

Supplementary Table 2: Genaralized linear models for each subchallenge.

β	Estimate 1A	Std. Error 1A	P-value 1A	Estimate 1B	Std. Error 1B	P-value 1B	Estimate 1C	Std. Error 1C	P-value 1C	Estimate 2A	Std. Error 2A	P-value 2A	Estimate 2B	Std. Error 2B	P-value 2B	Estimate 3A	Std. Error 3A	P-value 3A	Estimate 3B	Std. Error 3B	P-value 3B
Intercept	4.42	0.15	1.30×10^{-196}	1.76	0.13	3.73×10^{-39}	1.23	0.05	3.77×10^{-133}	0.46	0.08	6.91×10^{-9}	0.34	0.07	5.54×10^{-6}	1.30	0.08	1.18×10^{-61}	1.37	0.08	6.39×10^{-71}
Effective Depth	0.25	0.04	3.78×10^{-10}	0.34	0.09	1.14×10^{-4}	0.25	0.01	6.08×10^{-147}	0.56	0.02	4.72×10^{-254}	0.58	0.02	1.78×10^{-294}	0.31	0.01	1.52×10^{-98}	0.31	0.01	2.43×10^{-118}
T3	-0.16	0.13	2.14×10^{-1}	-0.26	0.13	3.97×10^{-2}	-0.02	0.07	8.23×10^{-1}	-0.11	0.11	2.90×10^{-1}	-0.14	0.10	1.60×10^{-1}	-0.00	0.11	9.94×10^{-1}	-0.06	0.11	5.50×10^{-1}
T4	-0.04	0.13	7.66×10^{-1}	-0.42	0.13	9.45×10^{-4}	-0.22	0.07	1.45×10^{-3}	-0.00	0.11	9.81×10^{-1}	0.02	0.10	8.45×10^{-1}	0.09	0.11	4.27×10^{-1}	-0.02	0.11	8.76×10^{-1}
T5	-0.44	0.13	5.15×10^{-4}	-0.62	0.12	1.34×10^{-7}	-0.01	0.07	9.09×10^{-1}	0.37	0.11	6.53×10^{-4}	0.45	0.10	1.80×10^{-5}	0.41	0.12	6.35×10^{-4}	0.34	0.12	3.65×10^{-3}
T6	-0.72	0.12	5.26×10^{-9}	-0.37	0.13	3.69×10^{-3}	0.07	0.07	2.98×10^{-1}	0.48	0.11	1.36×10^{-5}	0.63	0.11	1.89×10^{-9}	0.50	0.12	4.55×10^{-5}	0.41	0.12	5.79×10^{-4}
PhyloWGS	-0.42	0.08	1.47×10^{-7}	-0.09	0.08	2.64×10^{-1}	0.07	0.02	5.38×10^{-4}	0.05	0.03	1.28×10^{-1}	0.14	0.03	1.25×10^{-6}						
Downsampling	-0.18	0.08	2.20×10^{-2}	0.07	0.08	3.67×10^{-1}	-0.02	0.02	3.03×10^{-1}	0.00	0.03	9.97×10^{-1}	-0.00	0.03	9.49×10^{-1}	-0.01	0.03	8.25×10^{-1}	-0.02	0.03	4.39×10^{-1}
MuTect	-0.10	0.13	4.28×10^{-1}	-0.33	0.12	8.46×10^{-3}	-0.23	0.07	5.87×10^{-4}	-0.63	0.11	3.38×10^{-9}	-0.66	0.10	4.87×10^{-11}	-0.41	0.10	6.77×10^{-5}	-0.48	0.10	1.35×10^{-6}
SomaticSniper	-0.11	0.13	4.03×10^{-1}	-0.42	0.15	4.47×10^{-3}	-0.22	0.07	1.14×10^{-3}	-0.99	0.11	7.96×10^{-20}	-1.02	0.10	3.34×10^{-23}	-0.67	0.10	1.86×10^{-11}	-0.74	0.10	1.78×10^{-14}
Strelka	-0.25	0.12	4.16×10^{-2}	-0.59	0.12	3.52×10^{-7}	-0.32	0.07	2.90×10^{-6}	-1.02	0.11	5.60×10^{-21}	-0.96	0.10	1.34×10^{-20}	-0.67	0.10	2.34×10^{-11}	-0.69	0.10	9.35×10^{-13}
Mutationseq	-0.13	0.13	2.86×10^{-1}	-0.65	0.12	3.00×10^{-8}	-0.30	0.07	1.08×10^{-5}	-1.37	0.11	4.20×10^{-35}	-1.30	0.11	3.38×10^{-35}	-0.88	0.10	5.68×10^{-17}	-0.85	0.10	7.52×10^{-19}
MuTect:T3							-0.13	0.10	1.82×10^{-1}	-0.05	0.15	7.62×10^{-1}	-0.08	0.14	5.62×10^{-1}	-0.13	0.14	3.58×10^{-1}	-0.12	0.14	4.03×10^{-1}
MuTect:T4							-0.18	0.10	6.09×10^{-2}	-0.12	0.15	4.16×10^{-1}	-0.09	0.14	5.20×10^{-1}	-0.13	0.15	3.78×10^{-1}	-0.13	0.14	3.56×10^{-1}
MuTect:T5							-0.21	0.10	2.63×10^{-2}	-0.23	0.15	1.37×10^{-1}	-0.12	0.14	4.25×10^{-1}	-0.20	0.16	2.06×10^{-1}	-0.15	0.15	3.16×10^{-1}
MuTect:T6							0.02	0.10	8.21×10^{-1}	0.17	0.15	2.63×10^{-1}	0.09	0.15	5.42×10^{-1}	-0.33	0.16	3.61×10^{-2}	-0.24	0.15	1.18×10^{-1}
SomaticSniper:T3							-0.39	0.10	4.41×10^{-5}	-0.18	0.15	2.45×10^{-1}	-0.17	0.15	2.56×10^{-1}	-0.15	0.14	2.88×10^{-1}	-0.13	0.14	3.51×10^{-1}
SomaticSniper:T4							-0.04	0.10	6.61×10^{-1}	-0.06	0.15	6.84×10^{-1}	0.03	0.15	8.62×10^{-1}	-0.02	0.15	8.66×10^{-1}	-0.03	0.14	8.56×10^{-1}
SomaticSniper:T5							-0.55	0.10	8.03×10^{-9}	-0.26	0.15	8.70×10^{-2}	-0.32	0.15	2.82×10^{-2}	-0.30	0.15	5.04×10^{-2}	-0.28	0.15	5.47×10^{-2}
SomaticSniper:T6							0.02	0.10	8.57×10^{-1}	0.20	0.15	1.99×10^{-1}	0.09	0.15	5.49×10^{-1}	-0.35	0.15	2.13×10^{-2}	-0.27	0.15	6.33×10^{-2}
Strelka:T3							-0.10	0.10	2.85×10^{-1}	-0.05	0.15	7.69×10^{-1}	-0.07	0.15	6.10×10^{-1}	-0.08	0.14	5.89×10^{-1}	-0.09	0.14	5.27×10^{-1}
Strelka:T4							-0.08	0.10	3.95×10^{-1}	0.20	0.15	1.83×10^{-1}	0.16	0.14	2.55×10^{-1}	0.02	0.15	9.12×10^{-1}	0.04	0.14	7.75×10^{-1}
Strelka:T5							-0.37	0.10	8.92×10^{-5}	-0.22	0.15	1.49×10^{-1}	-0.30	0.15	3.72×10^{-2}	-0.32	0.15	3.58×10^{-2}	-0.31	0.15	3.21×10^{-2}
Strelka:T6							-0.15	0.09	1.14×10^{-1}	0.36	0.15	2.13×10^{-2}	0.33	0.15	2.44×10^{-2}	-0.28	0.15	6.91×10^{-2}	-0.22	0.15	1.32×10^{-1}
Mutationseq:T3							-0.18	0.10	5.60×10^{-2}	-0.09	0.16	5.87×10^{-1}	-0.07	0.15	6.17×10^{-1}	-0.05	0.14	7.42×10^{-1}	-0.09	0.14	5.15×10^{-1}
Mutationseq:T4							-0.17	0.10	8.14×10^{-2}	-0.09	0.16	5.53×10^{-1}	-0.06	0.15	6.75×10^{-1}	-0.01	0.15	9.32×10^{-1}	-0.04	0.14	7.53×10^{-1}
Mutationseq:T5							-0.44	0.10	4.01×10^{-6}	-0.45	0.16	4.16×10^{-3}	-0.47	0.15	1.58×10^{-3}	-0.23	0.15	1.35×10^{-1}	-0.28	0.14	5.10×10^{-2}
Mutationseq:T6							0.03	0.09	7.72×10^{-1}	-0.22	0.16	1.57×10^{-1}	-0.50	0.15	8.18×10^{-4}	-0.35	0.16	2.27×10^{-2}	-0.35	0.15	1.52×10^{-2}
phi	22.37	1.76	6.54×10^{-37}				33.03	2.18	5.48×10^{-52}	14.16	0.89	3.10×10^{-57}	16.14	1.02	6.81×10^{-57}	42.08	3.79	1.40×10^{-28}	46.35	4.14	4.36×10^{-29}
Log-likelihood	1506.53						941.02			500.33			542.26			386.54			398.21		
Pseudo R-squared	0.35						0.68			0.83			0.85			0.81			0.83		

β regressions were used for all subchallenges except 1B where a binomial regression was used. N=500 for subchallenges 1 and 2 and N=250 for subchallenge 3.

Supplementary Table 3: β regressions for subchallenges 1C and 2A for sensitivity (N=500).

β	Estimate 1C	Std. Error 1C	P-value 1C	Estimate 2A	Std. Error 2A	P-value 2A
Intercept	0.95	0.06	4.96×10^{-61}	0.12	0.09	1.92×10^{-1}
Effective Depth	0.09	0.02	2.73×10^{-6}	0.37	0.03	2.69×10^{-32}
T3	0.00	0.07	9.93×10^{-1}	-0.09	0.10	3.80×10^{-1}
T4	-0.12	0.07	7.12×10^{-2}	0.09	0.10	3.96×10^{-1}
T5	0.06	0.07	3.84×10^{-1}	0.45	0.10	1.64×10^{-5}
T6	0.04	0.07	5.76×10^{-1}	0.48	0.10	4.35×10^{-6}
PhyloWGS	0.07	0.02	4.74×10^{-5}	0.05	0.03	1.13×10^{-1}
Downsampling	-0.02	0.02	3.09×10^{-1}	-0.00	0.03	9.84×10^{-1}
MuTect	0.02	0.07	8.29×10^{-1}	-0.33	0.11	3.14×10^{-3}
SomaticSniper	0.16	0.08	4.14×10^{-2}	-0.54	0.12	1.25×10^{-5}
Strelka	-0.02	0.07	7.50×10^{-1}	-0.70	0.12	1.55×10^{-9}
Mutationseq	0.08	0.08	3.07×10^{-1}	-0.92	0.12	1.17×10^{-13}
Sensitivity	0.23	0.02	4.26×10^{-21}	0.30	0.04	8.92×10^{-13}
MuTect:T3	-0.12	0.09	1.90×10^{-1}	-0.04	0.14	7.58×10^{-1}
MuTect:T4	-0.24	0.09	8.05×10^{-3}	-0.19	0.14	1.88×10^{-1}
MuTect:T5	-0.19	0.09	4.14×10^{-2}	-0.18	0.15	2.07×10^{-1}
MuTect:T6	-0.07	0.09	4.73×10^{-1}	0.03	0.15	8.21×10^{-1}
SomaticSniper:T3	-0.35	0.09	1.15×10^{-4}	-0.12	0.15	4.02×10^{-1}
SomaticSniper:T4	-0.17	0.09	7.12×10^{-2}	-0.17	0.15	2.52×10^{-1}
SomaticSniper:T5	-0.51	0.09	2.56×10^{-8}	-0.19	0.15	1.88×10^{-1}
SomaticSniper:T6	-0.10	0.09	3.11×10^{-1}	0.03	0.15	8.18×10^{-1}
Strelka:T3	-0.08	0.09	3.67×10^{-1}	-0.02	0.15	8.97×10^{-1}
Strelka:T4	-0.20	0.09	3.24×10^{-2}	0.13	0.15	3.86×10^{-1}
Strelka:T5	-0.28	0.09	2.51×10^{-3}	-0.07	0.15	6.46×10^{-1}
Strelka:T6	-0.24	0.09	8.63×10^{-3}	0.23	0.15	1.19×10^{-1}
Mutationseq:T3	-0.15	0.09	9.84×10^{-2}	-0.09	0.15	5.56×10^{-1}
Mutationseq:T4	-0.19	0.09	3.95×10^{-2}	-0.09	0.15	5.51×10^{-1}
Mutationseq:T5	-0.38	0.09	3.85×10^{-5}	-0.37	0.15	1.38×10^{-2}
Mutationseq:T6	0.03	0.09	7.75×10^{-1}	-0.25	0.15	9.97×10^{-2}
phi	37.83	2.48	1.01×10^{-52}	15.97	1.01	8.62×10^{-57}
Log-likelihood	980.87			526.52		
Pseudo R-squared	0.72			0.85		

Supplementary Table 4: Genaralized linear models for each subchallenge with the real tumour BAM for PD4120.

β	Estimate 1A	Std. Error 1A	P-value 1A	Estimate 1B	Std. Error 1B	P-value 1B	Estimate 1C	Std. Error 1C	P-value 1C	Estimate 2A	Std. Error 2A	P-value 2A	Estimate 2B	Std. Error 2B	P-value 2B	Estimate 3A	Std. Error 3A	P-value 3A	Estimate 3B	Std. Error 3B	P-value 3B
Intercept	1.97	0.09	7.31×10^{-118}	0.79	0.16	7.04×10^{-7}	0.87	0.05	1.60×10^{-60}	0.12	0.09	1.74×10^{-1}	0.44	0.10	1.65×10^{-5}	1.04	0.02	0.00×10^{0}	1.03	0.02	0.00×10^{0}
Effective Depth	0.09	0.03	4.20×10^{-3}	0.73	0.07	1.21×10^{-27}	0.15	0.02	1.91×10^{-12}	0.54	0.04	2.25×10^{-42}	0.66	0.04	9.07×10^{-51}	0.13	0.01	4.10×10^{-46}	0.12	0.01	1.08×10^{-40}
PhyloWGS	-0.65	0.07	3.02×10^{-23}	-0.57	0.13	9.87×10^{-6}	-0.10	0.04	1.63×10^{-2}	0.29	0.08	1.16×10^{-4}	0.13	0.08	1.28×10^{-1}						
Downsampling	-0.00	0.06	9.85×10^{-1}	0.02	0.13	8.98×10^{-1}	-0.01	0.04	8.77×10^{-1}	-0.00	0.08	9.70×10^{-1}	0.02	0.08	8.36×10^{-1}	0.06	0.02	7.04×10^{-4}	0.05	0.02	4.20×10^{-3}
SomaticSniper	-0.20	0.09	2.44×10^{-2}	-0.39	0.18	3.11×10^{-2}	-0.00	0.06	9.58×10^{-1}	-0.33	0.11	1.99×10^{-3}	-0.53	0.12	6.05×10^{-6}	-0.12	0.02	2.89×10^{-6}	-0.12	0.03	6.45×10^{-6}
Strelka	-0.04	0.09	6.74×10^{-1}	-0.07	0.18	7.16×10^{-1}	-0.01	0.06	8.12×10^{-1}	0.10	0.11	3.36×10^{-1}	0.08	0.12	4.81×10^{-1}	0.02	0.03	3.64×10^{-1}	0.02	0.03	5.20×10^{-1}
Mutationseq	-0.03	0.09	7.32×10^{-1}	-0.10	0.18	5.85×10^{-1}	0.03	0.06	6.66×10^{-1}	-0.12	0.11	2.75×10^{-1}	-0.13	0.12	2.63×10^{-1}	-0.07	0.03	3.51×10^{-3}	-0.08	0.03	2.30×10^{-3}
phi	32.14	4.92	6.55×10^{-11}				33.85	4.93	6.83×10^{-12}	11.47	1.61	9.88×10^{-13}	9.55	1.36	1.93×10^{-12}	523.47	107.02	1.00×10^{-6}	481.08	98.35	1.00×10^{-6}
Log-likelihood	197.56						157.65			66.03			67.80			152.26			149.27		
Pseudo R-squared	0.63						0.33			0.69			0.72			0.84			0.81		

β regressions were used for all subchallenges except 1B where a binomial regression was used. N=100 for subchallenges 1 and 2 and N=50 for subchallenge 3.

Supplementary Table 5: Genaralized linear models for each subchallenge with errors in the CNA-profiles.

β	Estimate 1A	Std. Error 1A	P-value 1A	Estimate 1B	Std. Error 1B	P-value 1B	Estimate 1C	Std. Error 1C	P-value 1C	Estimate 2A	Std. Error 2A	P-value 2A	Estimate 2B	Std. Error 2B	P-value 2B	Estimate 3A	Std. Error 3A	P-value 3A	Estimate 3B	Std. Error 3B	P-value 3B
Intercept	1.07	0.08	3.15×10^{-42}	0.70	0.19	1.73×10^{-4}	1.22	0.02	0.00×10^{0}	0.45	0.05	1.18×10^{-18}	0.34	0.05	7.35×10^{-12}	1.02	0.04	1.91×10^{-164}	1.01	0.03	2.47×10^{-282}
Effective Depth	0.03	0.01	2.63×10^{-4}	0.27	0.01	1.54×10^{-94}	0.25	0.01	5.55×10^{-69}	0.47	0.01	0.00×10^{0}	0.49	0.01	0.00×10^{0}	0.28	0.00	0.00×10^{0}	0.27	0.00	0.00×10^{0}
T3	-0.06	0.10	5.43×10^{-1}	-0.04	0.06	5.19×10^{-1}	-0.09	0.01	7.30×10^{-16}	-0.05	0.02	1.94×10^{-3}	0.03	0.02	5.49×10^{-2}	-0.04	0.05	3.84×10^{-1}	-0.06	0.03	2.99×10^{-2}
T4	0.01	0.10	8.92×10^{-1}	-0.27	0.06	1.10×10^{-5}	-0.27	0.01	1.24×10^{-131}	0.16	0.02	1.40×10^{-21}	0.11	0.02	1.26×10^{-11}	-0.03	0.05	5.11×10^{-1}	-0.05	0.03	1.13×10^{-1}
T5	-0.12	0.10	2.43×10^{-1}	-0.45	0.06	6.79×10^{-16}	-0.37	0.01	3.56×10^{-261}	-0.12	0.02	3.85×10^{-12}	-0.14	0.02	7.67×10^{-18}	-0.11	0.05	3.27×10^{-2}	-0.15	0.03	4.04×10^{-7}
T6	-0.24	0.11	2.73×10^{-2}	-0.38	0.06	1.03×10^{-10}	0.08	0.01	5.11×10^{-14}	0.57	0.02	1.55×10^{-262}	0.64	0.02	0.00×10^{0}	0.15	0.05	2.83×10^{-3}	0.21	0.03	1.35×10^{-12}
PhyloWGS	0.19	0.02	7.80×10^{-28}	0.05	0.04	2.25×10^{-1}	0.04	0.01	6.06×10^{-10}	0.07	0.01	1.85×10^{-10}	0.13	0.01	2.00×10^{-35}						
MuTect	-0.01	0.03	8.21×10^{-1}	-0.60	0.23	9.92×10^{-3}	-0.32	0.01	1.82×10^{-188}	-0.71	0.07	6.31×10^{-26}	-0.70	0.07	1.56×10^{-26}	-0.43	0.01	1.30×10^{-275}	-0.43	0.03	1.19×10^{-48}
SomaticSniper	-0.01	0.03	7.09×10^{-1}	-0.15	0.30	6.10×10^{-1}	-0.32	0.01	2.60×10^{-190}	-1.09	0.07	3.34×10^{-57}	-1.09	0.07	8.98×10^{-61}	-0.68	0.01	0.00×10^{0}	-0.70	0.03	4.16×10^{-122}
Strelka	-0.04	0.03	1.78×10^{-1}	-0.33	0.24	1.62×10^{-1}	-0.39	0.01	1.69×10^{-276}	-1.02	0.07	4.48×10^{-51}	-0.92	0.07	4.62×10^{-44}	-0.61	0.01	0.00×10^{0}	-0.64	0.03	2.47×10^{-104}
Mutationseq	0.01	0.03	7.82×10^{-1}	-0.67	0.23	3.90×10^{-3}	-0.40	0.01	1.32×10^{-296}	-1.59	0.07	5.03×10^{-114}	-1.54	0.07	1.16×10^{-112}	-0.83	0.01	0.00×10^{0}	-0.78	0.03	5.90×10^{-149}
ploidy x2	-0.50	0.11	1.26×10^{-5}	-0.49	0.24	4.09×10^{-2}	-0.20	0.02	1.04×10^{-22}	-0.30	0.07	1.12×10^{-5}	-0.42	0.07	2.63×10^{-10}	0.29	0.05	5.87×10^{-9}	0.08	0.02	1.02×10^{-3}
scramble	-0.30	0.08	3.79×10^{-4}	-0.23	0.20	2.42×10^{-1}	0.01	0.02	4.17×10^{-1}	-0.28	0.05	2.82×10^{-7}	-0.24	0.05	5.68×10^{-6}	-0.02	0.04	5.84×10^{-1}	-0.07	0.02	4.51×10^{-4}
scramble gains	-0.26	0.08	1.89×10^{-3}	-0.19	0.20	3.24×10^{-1}	-0.02	0.02	2.72×10^{-1}	-0.30	0.05	2.37×10^{-8}	-0.29	0.05	5.28×10^{-8}	-0.03	0.04	4.63×10^{-1}	-0.04	0.02	4.14×10^{-2}
scramble loss	0.29	0.08	4.80×10^{-4}	0.02	0.20	9.17×10^{-1}	0.04	0.02	2.11×10^{-2}	-0.11	0.05	3.53×10^{-2}	-0.07	0.05	2.00×10^{-1}	-0.09	0.04	2.25×10^{-2}	-0.10	0.02	5.76×10^{-7}
proportion CNAs scrambled	-0.15	0.01	1.75×10^{-56}	-0.00	0.00	5.19×10^{-1}	-0.03	0.00	1.36×10^{-12}	-0.03	0.01	5.16×10^{-6}	-0.03	0.01	3.60×10^{-6}	0.02	0.00	4.49×10^{-6}	0.03	0.00	1.34×10^{-11}
T3:ploidy x2	0.04	0.16	8.10×10^{-1}													-0.31	0.07	9.13×10^{-6}			
T3:scramble	0.16	0.12	1.81×10^{-1}													-0.13	0.05	1.43×10^{-2}			
T3:scramble gains	0.14	0.12	2.45×10^{-1}													0.05	0.05	3.71×10^{-1}			
T3:scramble loss	-0.00	0.11	9.95×10^{-1}													-0.04	0.05	4.30×10^{-1}			
T4:ploidy x2	0.24	0.16	1.34×10^{-1}													-0.25	0.07	2.75×10^{-4}			
T4:scramble	0.04	0.12	7.33×10^{-1}													0.04	0.05	5.12×10^{-1}			
T4:scramble gains	0.08	0.11	4.77×10^{-1}													0.04	0.05	5.16×10^{-1}			
T4:scramble loss	0.03	0.11	7.89×10^{-1}													0.07	0.05	2.24×10^{-1}			
T5:ploidy x2	0.18	0.16	2.62×10^{-1}													-0.18	0.07	1.06×10^{-2}			
T5:scramble	0.52	0.12	8.27×10^{-6}													0.03	0.05	5.61×10^{-1}			
T5:scramble gains	0.42	0.12	2.69×10^{-4}													0.07	0.05	1.83×10^{-1}			
T5:scramble loss	0.04	0.11	7.11×10^{-1}													0.10	0.05	6.93×10^{-2}			
T6:ploidy x2	0.30	0.16	6.40×10^{-2}													-0.35	0.07	4.28×10^{-7}			
T6:scramble	0.52	0.12	1.01×10^{-5}													-0.02	0.05	7.24×10^{-1}			
T6:scramble gains	0.33	0.12	5.19×10^{-3}													-0.03	0.05	5.99×10^{-1}			
T6:scramble loss	0.05	0.12	6.41×10^{-1}													0.07	0.05	1.94×10^{-1}			
MuTect:ploidy x2				0.10	0.31	7.62×10^{-1}				0.15	0.09	1.23×10^{-1}	0.23	0.09	1.44×10^{-2}						
MuTect:scramble				0.24	0.25	3.38×10^{-1}				0.18	0.07	1.23×10^{-2}	0.15	0.07	3.81×10^{-2}						
MuTect:scramble gains				0.18	0.25	4.87×10^{-1}				0.21	0.07	4.35×10^{-3}	0.19	0.07	8.26×10^{-3}						
MuTect:scramble loss				0.02	0.26	9.53×10^{-1}				0.12	0.07	1.13×10^{-1}	0.08	0.07	2.41×10^{-1}						
SomaticSniper:ploidy x2				-0.41	0.39	2.87×10^{-1}				0.25	0.10	9.48×10^{-3}	0.35	0.09	1.61×10^{-4}						
SomaticSniper:scramble				0.31	0.32	3.33×10^{-1}				0.26	0.07	4.91×10^{-4}	0.25	0.07	4.97×10^{-4}						
SomaticSniper:scramble gains				0.14	0.32	6.63×10^{-1}				0.30	0.07	6.12×10^{-5}	0.30	0.07	2.90×10^{-5}						
SomaticSniper:scramble loss				0.14	0.32	6.56×10^{-1}				0.13	0.07	9.06×10^{-2}	0.10	0.07	1.52×10^{-1}						
Strelka:ploidy x2				0.36	0.32	2.69×10^{-1}				0.44	0.09	3.28×10^{-6}	0.42	0.09	5.83×10^{-6}						
Strelka:scramble				0.22	0.26	4.07×10^{-1}				0.34	0.07	3.80×10^{-6}	0.27	0.07	1.98×10^{-4}						
Strelka:scramble gains				0.16	0.26	5.49×10^{-1}				0.41	0.07	3.90×10^{-8}	0.32	0.07	1.00×10^{-5}						
Strelka:scramble loss				-0.08	0.26	7.66×10^{-1}				0.19	0.07	1.04×10^{-2}	0.11	0.07	1.13×10^{-1}						
Mutationseq:ploidy x2				-0.02	0.31	9.51×10^{-1}				0.45	0.10	4.25×10^{-6}	0.51	0.10	6.77×10^{-8}						
Mutationseq:scramble				0.35	0.25	1.64×10^{-1}				0.32	0.08	3.00×10^{-5}	0.30	0.07	6.37×10^{-5}						
Mutationseq:scramble gains				0.27	0.25	2.81×10^{-1}				0.37	0.08	1.39×10^{-6}	0.36	0.07	1.96×10^{-6}						
Mutationseq:scramble loss				0.07	0.25	7.84×10^{-1}				0.15	0.08	5.71×10^{-2}	0.12	0.07	1.18×10^{-1}						
MuTect:T3																			0.00	0.04	9.87×10^{-1}
MuTect:T4																			-0.08	0.04	5.16×10^{-2}
MuTect:T5																			-0.09	0.04	2.61×10^{-2}
MuTect:T6																			-0.04	0.04	3.46×10^{-1}
SomaticSniper:T3																			-0.09	0.04	3.83×10^{-2}
SomaticSniper:T4																			0.04	0.04	2.78×10^{-1}
SomaticSniper:T5																			0.02	0.04	7.15×10^{-1}
SomaticSniper:T6																			-0.10	0.04	2.08×10^{-2}
Strelka:T3																			-0.04	0.04	3.92×10^{-1}
Strelka:T4																			0.08	0.04	6.60×10^{-2}
Strelka:T5																			-0.04	0.04	3.54×10^{-1}
Strelka:T6																			-0.01	0.04	7.71×10^{-1}
Mutationseq:T3																			-0.07	0.04	1.17×10^{-1}
Mutationseq:T4																			0.01	0.04	8.79×10^{-1}
Mutationseq:T5																			-0.00	0.04	9.95×10^{-1}
Mutationseq:T6																			-0.17	0.04	3.51×10^{-5}
Effective Depth:ploidy x2							-0.17	0.02	4.06×10^{-17}												
Effective Depth:scramble							-0.01	0.02	3.69×10^{-1}												
Effective Depth:scramble gains							-0.02	0.02	2.16×10^{-1}												
Effective Depth:scramble loss							0.00	0.02	7.64×10^{-1}												
phi	2.80	0.07	0.00×10^{0}				29.78	0.67	0.00×10^{0}	14.59	0.31	0.00×10^{0}	15.50	0.34	0.00×10^{0}	54.84	1.74	2.76×10^{-219}	46.82	1.46	1.05×10^{-226}
Log-likelihood	7042.39						7081.81			3939.12			4132.92			3380.31			3224.20		
Pseudo R-squared	0.31						0.61			0.79			0.80			0.84			0.83		

β regressions were used for all subchallenges except 1B where a binomial regression was used. N=4250 for subchallenges 1 and 2 and N=2125 for subchallenge 3.

Supplementary Note 1 – Metrics

A community effort to create standards for evaluating tumor subclonal reconstruction

Desirable properties for clustering metrics

In the literature on metrics for evaluating clustering predictions there have been multiple desirable properties identified for any measure used to evaluate clustering algorithms[1]. Since the tasks for SC2 and SC3 essentially amount to clustering SNVs based on their associated subclonal lineages and their phylogenetic relationships, we were interested in using scoring metrics that possessed these properties.

Of the seven desirable properties identified in Rosenberg, 2007[2] we selected four that were applicable to the scoring metrics for SC2 and SC3. We then replicated the simulation used in (Rosenberg 2007) to test if a given metric possessed the identified desirable properties and ran this simulation on any potential scoring metrics for SC2 or SC3. Finally, we used these results to eliminate any scoring metric that did not satisfy these desirable properties.

Definitions. Before listing the metric properties identified in (Rosenberg 2007) we must first define the clustering problem used in the simulation, the input any scoring metric for the clustering problem will receive, and all parameters associated with the clustering problem. The clustering problem is defined as follows: given N items that are each assigned to an unknown class from an unknown set of classes, $C = \{c_i | i = 1,...,n\}$, cluster (or group) the items into a set of clusters, $K = \{k_i | i = 1,...,m\}$ such that $n = m$

if $A = \{a_{ij}\} = \{\#$ of data points that are members of c_i and are assigned to $k_j\}$ is the contingency table representing the clustering solution then:

$$\forall\ i \in \{1,...,n\}\ \exists\ i* \in \{1,...,n\} \text{ s.t.}$$

$$a_{i,i^*} \neq 0$$

$$a_{ij} = 0;\ \forall\ j \in \{1,\ldots,n\},\ j \neq i^*$$

$$i^*$$

$$= j^*;\ \forall\ j \in \{1,\ldots,n\},\ j \neq i$$

i.e. the cluster assignments are as close as possible to the class assignments.

Any scoring metric for the clustering problem will receive the following inputs:

- the number of items, N
- the set of classes, C
- the set of clusters, K
- the contingency matrix $A = \{a_{ij}\}$

To define the clustering solution parameters we first distinguish between two types of clusters: *useful clusters* and *noise clusters*. Useful clusters are clusters that are assigned mostly items from one class, and thus can be 'associated' with that class. Noise clusters are clusters that contain an equal amount of items from all classes and thus cannot be associated with any class.

The parameters for any solution to the scoring problem are then as follows:

- M the score for the given solution
- n_C the number of elements in each class
- |C| the number of classes
- |K| the number of clusters
- $|K_{noise}|$ the number of 'noise' clusters; $|K_{noise}| < |K|$
- $|K_u|$ the number of 'useful' clusters; $|K_u| = |K| - |K_{noise}|$
- ϵ the error probability *i.e.* the proportion of items that are assigned to an incorrect cluster; $\epsilon = \epsilon_1 + \epsilon_2$
- ϵ_1 the proportion of items that are assigned to in incorrect useful cluster *i.e.* a useful cluster that is not associated with the class that the item belongs to
- ϵ_2 the proportion of items that are assigned to noise clusters

Desirable Clustering Metric Properties

The four desirable metric properties for and scoring metric for SC2 or SC3 are:

1. For $|K_u| < |C|$, $\Delta|K_u| \leq (|C| - |K_u|)$, $\frac{\Delta M}{\Delta|K_u|} > 0$

2. For $|K_u| > |C|$, $\frac{\Delta M}{\Delta |K_u|} < 0$
3. $\frac{\delta M}{\delta \varepsilon_1} \leq 0$, with equality only if $|K_u| = 1$
4. $\frac{\delta M}{\delta \varepsilon_2} \leq 0$, with equality only if $|K_{noise}| = 0$

Other potential metrics for SC2 and SC3

In deciding on a scoring metric for SC2 and SC3 we also considered a number of other measures of matrix differences as potential candidates for our scoring metric, besides PCC and AJSD, which were all considered as both SC2 and SC3 scoring metrics. These metrics were considered in place of ASJD and the combination of PCC, MCC and were applied to the same input matrices as defined above.

Matthews correlation coefficient. The MCC is only defined between two binary matrices, thus we define CCM^{P*}:= round(CCM^P) and CCM^{T*} := round(CCM^T) and calculate the MCC using these matrices. The MCC between a matrix of predictions, C, and a truth matrix, K, is defined as:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where TP $:= \Sigma C_{ij}K_{ij}$ is the number of true positives, TN := $\Sigma (1 - C_{ij})(1 - K_{ij})$ is the number of true negatives, FP := $\Sigma C_{ij}(1 - K_{ij})$ is the number of false positives, and FN := $\Sigma (1 - C_{ij})K_{ij}$ is the number of false negatives.

Sum of Squared Error (SSE). The sum of the squared difference between each entry in the predicted matrix, C, and true matrix, K:

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(C_{ij} - K_{ij}\right)^2$$

Spearman's rank correlation coefficient (SCC). The normalized difference in rank between the entries of the predicted and true matrices:

$$SCC = 1 - \frac{6 \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij}^2}{n^2(n^4 - 1)}$$

where $d_{ij} = x_{ij} - y_{ij}$ is the difference between the rank of the ijth entry of the predicted matrix, x_{ij}, and the rank of the ijth entry of the true matrix, y_{ij}.

Clonal Fraction (CF). The CF of a reconstruction captures the accuracy of the predicted fraction of clonal mutations, i.e. mutations assigned to the clonal peak, against the true fraction of clonal mutations. This is defined as:

$$CF = max(0{,}1 - \frac{|c_p - c_t|}{c_t})$$

where c_t is the true CF and c_p is the predicted CF ($\frac{\#\ SNVs\ assigned\ to\ clonal\ peak}{Total\#SNVs}$).

Testing scoring metrics: desirable properties

To evaluate the degree to which different scoring metrics satisfied these desirable properties we systematically varied $|K_u|$, $|K_{noise}|$, while keeping n_C = 200 fixed. The values we iterate over for each parameter were:

$|C| \in \{2, ..., 9\}$

$|K\ u\ | \in \{2, ..., 10\}$

$|K\ noise\ | \in \{1, 3\}$

$\epsilon_1 \in \{0, 0.05, \cdots, 0.4\}$

$\epsilon_2 \in \{0, 0.05, \cdots, 0.4\}$

We then used the scores from each of these parameter settings to calculate the proportion of settings under which each of the scoring metrics satisfied each of the four desirable properties **(Supplementary Figure 1)**.

Scoring metric behaviour tests. In addition to testing our SC2 and SC3 scoring metrics for the desirable clustering metric properties defined above we also tested the behaviour of our scoring metrics for all sub-challenges under various mistake scenarios.

Mistake scenario definition. A mistake scenario for a sub-challenge is a hypothetical error in solving the sub-challenge problem and this error cannot be divided into multiple types of errors (*i.e.* merging two subclonal lineages where one subclonal lineage is directly ancestral to the other for SC3). The idea with these mistake scenarios is that each scenario is correct except for one mistake so that each type of mistake can be examined individually. It is possible, and likely, that some submissions to a sub-challenge will be combinations of multiple mistake scenarios.

SC1C behaviour tests. To analyse the performance of our scoring metric for SC1C we tested the behaviour of our metric under several mistake scenarios as well as when random error was added to either the predicted cellular prevalence of each subclonal lineage, φ, or the predicted number of SNVs assigned to a subclonal lineage, α.

We decided not to use squared errors in SC1 metrics but rather absolute errors in order to penalize small mistakes more severely. For SC1C in particular the mistakes are all small in terms of the effect on our metric input so the difference between using squared error and absolute error was amplified.

Our goal in testing the SC1C scoring metric was to verify that our metric penalized mistakes in φ more than mistakes in α and also penalized certain mistake scenarios in a desired way. We chose to emphasize mistakes in φ in SC1C since the predicted number of SNVs assigned to each subclonal lineage also affects SC2 and SC3 scores but the predicted cellular prevalence of each subclonal lineage only affects the SC1C score.

In order to test our metric we first looked at its behaviour in several mistake scenarios. We simulated a tumor with 0.85, 0.5, and 0.3 CP clusters with 200 mutations in each and scored each of six possible cluster merging and splitting solutions (only splitting a single cluster at a time). Unlike for SC2, for SC1C we wanted splitting a subclonal lineage to be penalized less than merging two subclonal lineages, since there is more error in the cellular prevalence when two subclonal lineages are merged than when one subclonal lineage is split. Otherwise we wanted SC1C to show the same behaviour with these mistake scenarios as SC2. Our scoring metric for SC1C using absolute error demonstrated the desired behaviour with respect to the mistake scenarios (Wilcox-test P-value for merging *vs.* splitting: 0.063; mean score when merging clusters: 0.877 ± 0.032 s.e., mean score when splitting a cluster 0.98 ± 0 s.e).

We also looked at our metric's behaviour when a random error was added to either φ or α (or both). We simulated a tumor with 0.85, 0.5, and 0.3 CP clusters (these are the true φ values) with 200 mutations each (true α). To add random noise to φ, for each concentration (c) in (2,5,10,1000), each φ was drawn from a random β distribution with a= true φ * c, and b= (1- true φ)*c. Similarly, we adjusted the true α vector by multiplying it with a single draw from a Dirichlet distribution parameterized by [c c c]. Increasing the concentration of correct φ and α reduces the random noise in each of these metrics. Our goal in performing this test was to verify that our metric a) penalizes error in either α or φ, b) penalizes a submission more when there is more error in either φ or α, and c) penalizes error in φ more than error in α. Our scoring metric for SC1C satisfied all of these criteria (φ: $\beta_{\text{correct concentration}}$: 0.00025, P-value: 0.037; α: $\beta_{\text{correct concentration}}$: 0.00023, P-value: 0.040).

Based on these two tests we were satisfied with the behaviour of our metric for SC1C using the absolute error and so chose it as our final scoring metric for SC1C.

SC2 and SC3 behaviour tests. After screening our scoring metrics using the results of the Rosenberg simulation we then examined the behaviour of each scoring metric under the different SC2 mistake scenarios. To systematically assess errors on a variety of tree architectures, for each possible tree with 1-6 nodes of equal size we generated every possible unique architecture while keeping node ordering consistent (*e.g.* in a three-node tree 1 would always have to be the root and the parent of 2, but it could be the parent or the grandparent of 3) and only allowing one node to have at most two children. Additionally, we generated three and four node linear trees with three different SNV distributions ranging from the majority of SNVs occurring in the clonal node, to the majority of SNVs occurring in the most subclonal node for a total of 30 tree architectures. For each of these architectures, we then introduced eight mistake scenarios that reflect common algorithm errors that cover over-clustering, under-clustering and incorrect inference of evolutionary relationships:

- Two clonal nodes are merged
- Two subclonal nodes are merged
- A spurious subclonal node is present composed of SNVs from adjacent nodes
- Two clonal nodes are merged and a spurious subclonal node is present
- A subclonal node is split into two nodes
- The grandparent is incorrectly inferred as the parent. If there was no grandparent or the grandparent had more than one child, then the closest sibling or uncle was inferred as the parent.
- A linear architecture
- A monoclonal solution

We then scored the CCM and ADM reflecting these errors against the truth for that tree with the candidate SC2 and SC3 metrics. For SC2B simulations, we subtracted/added a β-distributed error centred around a=1, b=10 for true positives/true negatives and a=2, b=10 for false positives/false negatives. Nine experts then ranked the overall severity of the mistake scenarios for each of the 30 architectures. We then assessed correspondence between expert rankings and rankings produced by each candidate metric using pairwise rank correlations. For SC3, we also considered clonal fraction due to its biological relevance.

We then assessed how well the average of the two and three highest scoring, but weakly correlated, metrics for SC2 and SC3 corresponded with the expert rankings as well as the average of all metrics. For SC2 we chose the average of AUPR and JS, which yielded the highest pairwise rank correlation with the expert rankings. For SC3, we chose Pearson, as JS, Pearson, MCC, and AUPR all produced non-significantly different rankings but significantly better than the clonal fraction.

Testing scoring metrics: real data

Unlike simulated data, where the truth is available by design, in real data, there is no available truth and no easy way to derive a perfect truth. Still, we identified three ways in which one could derive a gold standard to use as imperfect “truth” that should be closer to the truth than other typical setting.

Higher depth (233.64x) compared to typical lower/medium depth (60-70X). We took a real tumor sequenced at high depth (mean depth 233.64x) and down-sampled the reads to get lower depth BAM files. We called SNVs with five different somatic SNV detection approaches (**Online Methods**), but only retained to 10,000 SNVs that also appeared in the consensus variant calls derived on the full depth tumor. We then performed reconstructions on the down-sampled BAMs and scored them against consensus mutation assignments and subclonal reconstructions obtained from the full depth BAM. As expected, the scores decreased with depth **(Figure 5)**.

Truth derived from better set of SNV calls (consensus calls) compared to subsets (e.g. restricting to clocklike mutations in a (C>T)pG context) or suboptimal sets from individual somatic variant SNV detection pipelines. We took data from 538 donors from the PCAWG study[3] and executed DPClust on different sets of SNVs: a consensus set was used as “truth”, and three individual somatic SNV detection pipelines (MuTect, DKFZ pipeline and Sanger pipeline run on mutations in a (C>T)pG context). We also executed PhyloWGS on the consensus sets **(Supplementary Figure 6c)**. We then scored SC1C for each run against the “truth”. This showed that the reconstructions from two methods run on the same set of mutations are closer than reconstructions with the same somatic SNV detection pipeline on different sets of mutations and highlighted the importance of a good set of starting mutations. This corroborated the results on simulated data (**Figure 5**). Also, the SC1C scores of two methods should only be compared when run on the same set of mutations.

Truth derived from multi-region sequencing (5 regions) compared to single region. We took data from 10 donors with at least 5 tumor regions or metastases sequenced and executed DPClust in 5 dimensions to derive a “truth”. We scored each one-dimensional DPClust run on individual region against the “truth” derived from the multi-dimensional run for SC1C and SC2A and compared them against scores obtained from randomized 1C and 2A inputs. Interestingly, this showed that most but not all reconstructions based on individual tumors had better scores than randomized inputs, highlighting the important difference between reconstructions obtained with increased number of regions **(Supplementary Figure 6d)**. We note that we did not derive an “obtainable” truth and thus, a large part of the information retrieved from multi-dimensional reconstructions (up to 30 clusters/clones) cannot possibly be retrieved from single-region runs (up to 5 clusters/clones).

References

1. Dom, B. E. *An Information-Theoretic External Cluster-Validity Measure*. (2001).
2. Rosenberg, A. & Hirschberg, J. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. in *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic* 410–420 (2007).
3. Campbell, P. J. *et al.* Pan-cancer analysis of whole genomes. *bioRxiv* 162784 (2017) doi:10.1101/162784.

Supplementary Note 2 – BAMSurgeon simulator

A community effort to create standards for evaluating tumor subclonal reconstruction

Phasing pipeline

In this note, we give the details of each step of the phasing after a brief overview of how the steps fit together.

Phasing Terminology. Before proceeding, we clarify some of the phasing terminology we will be using. For each of the samples in the trio, GATK produced a set of unphased variants. Each of these variants is diploid, and is either homozygous (same allele twice) or heterozygous (2 different alleles).

- A *phase set* is simply a set of heterozygous variants (”hets”) that are phased together. Thus, if 2 hets A/C and G/T are part of the same phase set, and their individual phasings are, say, A|C and T|G, then the A and T alleles are part of one haplotype, and the C and G alleles are part of the other haplotype. Phase sets are chromosome-specific, but they are not required to be contiguous.
- That is, if 3 hets are consecutive on the reference (with no other variants in between them), it is possible for the first and last het to be part of the same phase set, and for the middle one to be unphased or part of a different phase set.
- A *phasing* (of a sample) consists of (1) a set of phase sets and (2) for each het that is part of a phase set, an assignment of the 2 alleles at that het to the 2 possible phases that are part of that phase set. Every single het can either be part of one unique phase set, or it can be part

of no phase sets at all. Note that there is no difference between an unphased het and a het that is part of a singleton phase set: neither case implies anything about the relative phasing of that het with respect to any other hets.

- A *parent-based phasing* (of a sample) is a special case of a phasing, in which one phase set per chromosome is singled out from other phase sets, and it is used to represent parent-of-origin information for the individual hets that are part of it. We refer to this phase set the parent phase set. Thus, if A|C and T|G are 2 hets part of the parent phase set in a parent-based phasing, then the A and T alleles are inherited from parent 1, and the C and G alleles are inherited from parent 2.

When we say that a phasing *A is extended by* a phasing B, then the phase sets of B are used to merge phase sets of A. In particular, every phase set in A appears as a subset of the resulting extended phasing.

Phasing Overview First, we used the ***ngs-phase*** program individually on the PE and MP data of each sample to build an initial phasing directly supported by NGS reads, referred to as ngs phasing. Second, we used the ***add-parent-phasing*** program to build an initial parent-based phasing for the child sample, referred to as ***parent_base*** phasing. Third, we used the ***extend-phase-set*** program to extend the ***parent_base*** phasing in the child with the ngs phasing in the two parents, obtaining the parent-based ***parent_ngs*** phasing. Fourth, we used the ***extend-phase-set*** program to extend the ***ngs*** phasing in the child with the ***parent_ngs*** phasing in the child, obtaining the ***ngs+parent_ngs*** phasing. Fifth, we used the ***extend-phase-set*** program to extend the ***ngs+parent_ngs*** phasing in the child with phasing produced by Beagle, obtaining the ***ngs+parent_ngs+beagle*** phasing. Finally, we used the programs ***random-flip-phase-set*** and ***collapse-phase-sets*** to randomly rotate and collapse each of the remaining phase sets into a single phase set, obtaining the final ***ngs+parent_ngs+beagle+rflip*** phasing. This phasing was used to separate the mappings of PE dataset from the child into individual haplotypes (**Supplementary Figure 2 e-i**).

The ngs phasing. The ngs phasing of each sample and chromosome gathers (sometimes conflicting) phasing information available in PE and MP NGS data. To build this phasing, we used the ***ngs-phase*** program, which works as follows.

First, each NGS fragment was inspected to determine if it contains evidence of any het alleles. More specifically, the mapping of each read in the fragment was tested for overlapping het sites. When a mapping overlapped a single nucleotide variant (SNV), if a unique read base was mapped to the het site and that base was one of the 2 possible alleles, that allele was considered supported by that read. When a mapping overlapped a short insertion or deletion (indel), a subsequence of the read was aligned using the Smith-

Waterman algorithm to the 2 possible het alleles augmented with a small common flank. If one of the alignment scores was better than the other, the allele corresponding to the better-scoring alignment was considered supported by that read. If the two reads part of the same fragment were found to support different alleles of the same het, that fragment was not used for phasing that het.

Next, all hets were initially placed in singleton phase sets. For every pair of phase hets, we computed the number of NGS fragments supporting every pair of allele connections between those sets. This produced 4 counts c_{00}, c_{01}, c_{10}, and c_{11}, corresponding to connections between alleles 0/0, 0/1, 1/0, and 1/1, respectively. We defined a measure of discordance for that pair of hets as:

$$(1 + \min(c_{00}+ c_{11}, c_{01}+ c_{10}))/(1 + c_{00} + c_{01} + c_{01} + c_{11})$$

As an exception, when:

$$E\min(c_{00} + c_{11}, c_{01} + c_{10}) = 0$$

The discordance was set to

$$\min(\boldsymbol{max\_discordance} - \epsilon * s, 1/(1 + s))$$

where $s = c_{00} + c_{01} + c_{10} + c_{11}$.

This allowed the Greedy algorithm used (described below) to always connect phase sets with low support when there were no discordant reads. Our approach to merging phase sets was Greedy, where we repeatedly: picked the pair of phase sets with minimum discordance, merging them, and updating the counts and discordance values. When merging the phase set B into the phase set A, with relative phasing $rel_{AB} \in \{0, 1\}$, we considered all other phase sets C connected to B. The count of fragments connecting phase $s_B \in \{0, 1\}$ of B with phase $s_C \in \{0, 1\}$ of C was added to the count of fragments connecting phase $(s_B + rel_{AB}) \bmod 2$ of A to phase s_C of C. The Greedy merging process was stopped when the minimum possible discordance between any 2 phase sets went above ***max_discordance*** = .2. At that point, the current phase sets and resulting phasing was reported.

The ***ngs-phase*** program had another, somewhat independent functionality, which was to categorize each het as being either ”reliable” or ”unreliable” from NGS data. A het was labelled as ”unreliable” if the NGS data covering it showed evidence that the site might not be diploid. Specifically, this happened if the total number of fragments covering a site was less than (1/2)×, or more than 2×, the overall coverage; or if either of the 2 alleles was supported by less than (1/4)× the number of fragments spanning the site.

The parent_base phasing. The ***parent_base*** phasing of each chromosome in the child sample is a *parent-based* phasing, which collects parent-of-origin information in a special parent phase set. This phasing was constructed using the ***add-parent-phasing*** program, which considers independently every het in the child along with the corresponding parent alleles at that locus. If a child allele $s \in \{0, 1\}$ appears only in parent $p \in \{0, 1\}$, and the child allele $1 - s$ is present in parent $1-p$, then the het is added to the parent phase set as $((s + p) \bmod 2) \mid ((s + p + 1) \bmod 2)$. In all other cases, the child het is left unphased. These include, for example, *de novo* mutations (child alleles not appearing in either parent), and triple hets (child hets that are also hets with the same alleles in both parents).

The parent_ngs phasing. The ***parent_ngs*** phasing of the child sample is an extension of the corresponding ***parent_base*** phasing of the child with the ***ngs*** phasing of each parent. This is also a parent-based phasing, and its goal is to compute parent-of-origin information for those sites left unphased in the ***parent_base*** phasing, using NGS data in the parents. For example, consider 2 nearby child hets with the following alleles in the child and 2 parents: (A_1/C_1, A_1/T_1, C_1/C_1) and (C_2/G_2, C_2/G_2, C_2/G_2). The ***parent_base*** phasing will phase the first het as $A_1|C_1$, but the second het will be left unphased. Now assume NGS data in the first parent contains evidence that the A_1 and G_2 alleles appear together in a parent haplotype. In this case, the ***parent_ngs*** phasing will phase the second het as $G_2|C_2$.

To perform the phase set extension, we used the program ***extend-phase-set***, which works as follows. The program is given as input a “base” phasing and an “extension” phasing. The program considers in turn each phase set B in the extension phasing, and looks for pairs of hets that are consecutive in B, and are part of different phase sets A_1 and A_2 in the base phasing. The program greedily picks the pair of such hets that are closest together in genomic distance, and merges the base phase sets A_1 and A_2 using the orientation in the extension phase set B.

When the base phasing is a *parent-based* phasing, as is the case when extending ***parent_base***, the program only extends the special parent phase set, and only ever rotates (if necessary) the other phase sets that are merged in the parent phase set.

The ngs+parent_ngs phasing. The ***ngs+parent_ngs*** phasing of the child sample is an extension of the ngs phasing in the child with the corresponding ***parent_ngs*** phasing. Thus, this phasing prioritises information from NGS data, and in its absence, it uses parent-of-origin information to further extend phase sets. The phasing is also obtained with the ***extend-phase-set*** program detailed above. Note that neither ***ngs*** nor ***ngs+parent_ngs*** are *parent-based* phasings, and in this case ***extend-phase-set*** can freely rotate and merge all phase sets as needed (because there is no parent-of-origin information to be lost by rotating the special parent phase set).

Beagle. In order to gather phasing information not available through NGS reads, we used Beagle version 4.0 on the set of variants produced by GATK. Prior to running Beagle, we merged sites with differing alleles with bcftools merge -m alle was run on the resulting trio variants, with appropriate pedigree information.

The ngs+parent_ngs+beagle phasing. The ***ngs+parent_ngs+beagle*** phasing is an extension of the ***ngs+parent_ngs*** phasing with the phasing produced by Beagle. This was again obtained by using the ***extend-phase-set*** program.

The ngs+parent_ngs+beagle+rflip phasing. Finally, the ***ngs+parent_ngs+beagle+rflip*** phasing was obtained by randomly rotating all remaining independent phase sets, and merging them into a final phase set. The programs used for this step were ***random-flip-phase-set*** (for random rotations) and ***collapse-phase-sets*** for merging.

Splitting. The ***ngs+parent_ngs+beagle+rflip*** is a complete phasing of all hets in the child sample. We then used the program ***bam-phase-split***, also part of PhaseTools, to transform a phasing of all hets into a phasing of all fragments and mappings of the high-coverage PE dataset. The program works as follows.

First, the program loads all variations with their phasing, it then iterates through the mappings, phases each fragment, and outputs 2 mapping files for each autosome, corresponding to the 2 phases. Each NGS fragment is processed independently of all others, and a decision is taken to place that fragment in 1 of the 2 possible phases. All reads in a fragment are considered together when phasing that fragment, with one exception: if 2 reads part of the same fragment are mapped to different chromosomes, they will be processed independently of each other.

If none of the reads in a fragment span any hets, the fragment is placed in one of the 2 phases at random. If a read spans one or more hets, for each such het, we inspect the read for evidence of 1 of the 2 possible alleles at that site. This is the exact same process performed by the program ***ngs-phase***, described earlier. As is the case with that program, if 2 reads in a fragment are found to contain conflicting evidence with respect to one het, that het is not used for determining the phase of that fragment. Finally, all hets spanned by any of the reads in the fragment for which the fragment contains evidence of one allele (thus, of one phase) are integrated into a phasing decision as follows. As explained before, the program ***ngs-phase*** assigns a label of "reliable" or "unreliable" to all hets. To phase a fragment, ***bam-phase-split*** first computes the phasing majority vote using all reliable hets only. If that comes out equal, it also computes the phasing majority vote using all unreliable hets. If that one is equal as well, the fragment is placed in one of the 2 phases at random. Otherwise, the vote is used to decide which phase to put the fragment in. All reads in the fragment are placed in that same phase.

Phasing sex chromosomes. In males, the sex chromosomes share pseudoautosomal regions (PARs), which are homologous sequences that play a role in genetic recombination during sexual reproduction. There are two well-known homologous sequences - PAR1 (chrX:60,001-2,699,520 = chrY:10,001-2,649,520) and PAR2 (chrX:154,931,044-155,260,560 = chrY:59,034,050-59,363,566) - any genes within them have been discovered to be inherited just like any autosomal genes. Thus, we can phase the reads in these regions just like we would other autosomes. However, since the sequences are homologous between chrX and chrY, it is very common (as it is in our case) to store the coordinates only in one of the sex chromosomes, in this case chrX. To reduce runtime and increase parallelization, for each PAR, we extracted the beagle allele frequencies from the trio, germline trio variants called by our pipeline, and PAR reads from chrX. The entire PhaseTools package workflow was executed on each PAR, and the reads split into phase “0” and phase “1” chrX BAMs. The non-PAR reads from the original chrX were merged with chrX phase “0” from both PARs to generate the phased chrX. To obtain the phased chrY, phase “1” of chrX PARs were reconfigured into their homologous locations on chrY using a direct PAR-to-PAR mapping, and then merged with the original chrY.

Mutation distribution and read splitting

A single configuration file is initially prepared, outlining the clonal and subclonal structure for the designed tumor tree at the chromosome level. This includes the number of single nucleotide variants (SNVs), insertions/deletions (Indels), and structural variants (SVs) to be applied to each paternal and maternal phase (phase 0/1 or “a”/”b”) of every chromosome. If there is more than one copy of a phase, indices are used to specify which copy the mutations are spiked into or which copy has the chromosomal event. Mutations are split according to chromosome size to get to obtain the desired total number of mutations. Furthermore, they are divided evenly between each chromosomal copy where there exists a phase gain, or combined where there exists a phase deletion. At each node, all mutations are spiked into the tumor portion prior to any copy number aberrations. Cellular prevalences are initially specified for each node split at the chromosome level.

Each phase of every chromosome is considered separately. To begin with, the read pools assigned to each leaf node is determined by the cellular prevalences of each node. To take into account chromosomal amplifications, we first adjust these read pools at each node, and then down-sample the leaf nodes in proportion to the max copy number at the chromosome level.

Pseudoreads are assigned to each leaf node, initialized by the cellular prevalence, and applying any chromosome gains or deletions. The number of copies at each node is tracked as well as the total number of pseudoreads (the desired number of virtual reads), where each copy is assigned the same number of pseudoreads := the number of pseudoreads assigned to the node divided by the number of copies at that node. A gain increments the pseudoreads equivalent to one copy, while a double gain increments it twice. A whole genome duplication doubles the pseudoreads as well as the copies for that node. Similarly, a deletion decrements the pseudoreads equivalent to one copy. The actual total number of reads assigned to the node is equal to pseudoreads assigned to the node divided by the total pseudoreads across all the leaf nodes.

Simulating replication timing

Replication timing data for the hg19 human reference genome build was obtained. Each genomic region was represented by an interval of 100Kbp, with DNA replication timing according to the HeLa cell line[2], and a noncoding mutation frequency per base pair, as averaged across 126 cancer samples subjected to WGS[1]. Gap region positions of GRCh37/hg19 reference genome were obtained use the Table Browser Tool at UCSC Genome Bioinformatics. Linear regression was used to model mutation rate from replication timing. Logs of regional mutation rates were used as they exhibited normal distribution across the genome and correlated strongly with replication timing. Missing replication timing values were imputed using the MICE (v2.25) package in R[3]. Bedtools (v2.24.0) was first used to obtain non-gap regions of reference genome, which were later excluded from the MICE algorithm[4]. Imputed data is taken after 40 iterations. Using the linear regression model generated from the original replication time and log mutation rate pairs, new log regional mutation rates were predicted for each region according to its replication time, imputed or original.

We then used the regional mutation rates to adjust mutation frequency for replication time. We first converted log-regional mutation rates to per base mutation rates and scaled them relative to the region with the highest mutation rate so each position had a timing factor between zero and one. We used these timing factors to moderate how likely a mutation was to occur at a given site. A mutation was only retained if a random number drawn from a uniform distribution between zero and one was less than or equal to the timing factor resulting in mutation enrichment at late replicating sites with timing factors close to one.

Simulating trinucleotide signatures

Each mutational signature s_i[5] is defined by the vector q_i giving the 96 probabilities of a mutation originating from that signature to mutate each of the 96 trinucleotide contexts. In each tumor simulation, SNVs are generated by a spectrum of multiple signatures. We spike N SNVs and associate weights w_i to each active signature, such that $\sum_i w_i = 1$. We then derive the mixture of signatures $q = \sum_i w_i q_i$. We could draw the N SNVs to spike from this multinomial distribution q. However, because we also model replication-timing biases, which impacts the spectrum of mutable trinucleotide contexts, prior to spiking in mutations, we derive the expected counts of mutations in each trinucleotide context, given by the vector Nq, used as a vector of quota. Thus, when spiking in SNVs we only spike a mutation if we have not yet reached the quota for that trinucleotide and if the mutation is not reverting a germline variant.

Simulating BAM files with variant allele frequencies derived from 3D tumor growth models

As training set, we simulated 1,366 3D tumors and derived allele frequency spectra of the variants in 8 virtual resections using a published tool[6]. These were simulated with increasing underlying selective pressure {0%, 1%, 2.5%, 5%, 10%}, mutation rates {0.1, 0.3, 0.6, 1, 2, 5, 10}, deme sizes {2500, 5000, 10000}, and 15 replicates per combination leading to a total of 1,575 runs, among which 1,366 completed within reasonable timeframe and gave properly formatted output. As test set, we simulated an additional 5 tumors with {0%, 1%, 2.5%, 5%, 10%} selective pressure. The code was edited to return exact variant allele frequencies of all mutations (VAFs) instead of frequencies with added (beta-)binomial noise. For 3 regions in each of the 5 test tumors, the perfect VAFs were then given as input to BAMsurgeon to generate BAM files with each of the mutations spiked at the given VAF, adding binomial noise within the BAMSurgeon pipeline. Mutations were then genotyped in each BAM file and the VAF spectra of the 3 regions used to predict whether the tumor was compatible with neutrality or selective pressure.

To this end, we trained an SVM model to predict neutrality on the 1,366 training tumors as described in Sun et al.[6] Although we trained on only three regions per tumor, we could correctly predict the (non-)neutrality of the tumors for which we simulated BAM files using BAMSurgeon (visualization akin to Sun et al.[6] in **Supplementary Figure 2d**). Interestingly, when running only on the mutations called by MuTect, the predictions were not as accurate (all tumors, including the neutrally growing tumor were predicted as having underlying selection), suggesting that the predictor should ideally be trained on realistic calls rather than “perfect” calls. This highlights the benefits of our new pipeline from which realistic SNV call sets can be derived, as compared to working with perfect simulated SNV sets.

Large scale SV simulations

Translocations are a critical type of oncogenic mutation, which was not included in the SMC-DNA simulated data challenges[7]. To do this, address this gap, we developed a new approach. For two regions (named A and B), an unbalanced translocation is simulated by selecting reads aligned to region A and reads aligned to region B and assembling contigs for each set of reads. To control for contig mis-assembly, each contig is aligned to the reference genome using exonerate[8], any unaligned portion at the ends is trimmed, and reads corresponding to the trimmed portion(s) of the contigs are de-selected. The contig break-ends are then fused either head-to-tail or head-to-head depending on user specification. Read coverage is generated over the fused contigs using *wgsim*[9]. Finally, altered reads are re-aligned to the reference genome and used to replace reads in the original BAM file based on read name, creating a simulated translocation that accurately reflects the expected pattern of discordant read pair mappings and split reads.

The ability to simulate translocations combined with adjustments to read coverage makes the simulation of arbitrarily large deletions, duplications, and inversions possible. Large-scale duplications are simulated by adding reads from a donor BAM at an appropriate depth, and deletions by removing reads. As these SVs can be arbitrarily large (size is only constrained by breakpoint insertion feasibility in a given region, it was important to simulate mutations accumulating within large SVs. To achieve this, we simulated mutations for each leaf BAM in reverse temporal order. So for a given leaf in a two-node tree, the large SVs which occur in that node are inserted first, then simple somatic mutations (SSMs) and small SVs for that leaf, followed by large SVs that occurred in the parent node, and finally SSMs and small SVs from the parent node. Using this strategy, SSMs that occur earlier than large duplications will be present in all reads covering that region (*e.g.* in all duplicated copies), while those that follow the duplication will only occur on a subset of the reads (*e.g.* on a single copy of the region). All SSMs and small SVs will appear as though they occur before the large SVs at that node, but simulating arbitrarily complex mutational sequences across a node’s evolutionary history is possible. Large deletions are accomplished by creating a translocation to represent the breakpoint joining the break ends on either side of the specified deletion. Reads corresponding to locations within the deleted region are down-sampled according to a user-specified VAF. Large duplications, specifically tandem segmental duplications, are created by simulating the breakpoint between the 3' end of the duplicated region and the 5' end of the duplicated region using the translocation functionality and increasing read coverage over the duplicated segment. Additional read coverage over the duplication is supplied via a user-specified source BAM file. Our pipeline generates donor BAM files by down-sampling a distinct pool of reads from the original BAM file at the appropriate depth to reflect copy number and cellular prevalence of the node. Large inversions consist of two separate inverted translocations, one at each breakpoint, and no adjustment of coverage is necessary.

References

1. Lawrence, M. S. *et al.* Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature* **499**, 214–218 (2013).
2. Chen, C.-L. *et al.* Impact of replication timing on non-CpG and CpG substitution rates in mammalian genomes. *Genome Res.* **20**, 447–457 (2010).
3. mice: Multivariate Imputation by Chained Equations in R | van Buuren | Journal of Statistical Software. doi:10.18637/jss.v045.i03.
4. Quinlan, A. R. & Hall, I. M. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26**, 841–842 (2010).
5. Alexandrov, L. B. *et al.* Signatures of mutational processes in human cancer. *Nature* **500**, 415–421 (2013).
6. Sun, R. *et al.* Between-Region Genetic Divergence Reflects the Mode and Tempo of Tumor Evolution. *Nat Genet* **49**, 1015–1024 (2017).
7. Lee, A. Y.-W. *et al.* Combining accurate tumour genome simulation with crowd sourcing to benchmark somatic structural variant detection. *bioRxiv* 224733 (2017) doi:10.1101/224733.

8. Slater, G. S. C. & Birney, E. Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics* **6**, 31 (2005).
9. Li, H. *wgsim: Reads simulator*. (2018).

Supplementary Note 3 – DPClust and PhyloWGS

A community effort to create standards for evaluating tumor subclonal reconstruction

Running PhyloWGS and DPClust

We used PhyloWGS and DPClust for subclonal reconstruction in our internal benchmark. The main differences between those two algorithms are outlined in the following table:

PhyloWGS	**DPClust**
Assigns CNAs to clusters	Does not assign CNAs to clusters
Estimates multiplicity of mutations during MCMC	Uses most likely multiplicity of each mutation during MCMC
Reconstructs phylogenies during MCMC	Reconstructs phylogenies after MCMC
Estimates purity from SNVs and CNAs	Estimates purity from CNAs

PhyloWGS and DPClust differ in several ways. The copy number state of a region of the genome affects the allele frequency of mutations located within that region, as does the number of chromosome copies bearing a mutation, or ‘multiplicity’. PhyloWGS models the interaction between copy number aberrations and SNV allele frequencies within Markov chain Monte-Carlo (MCMC), whereas DPClust sets the multiplicity of each mutation to its most likely value prior to MCMC. PhyloWGS includes CNAs within its clustering model, but DPClust does not. DPClust sets purity to a fixed value based on CNA analysis, whereas PhyloWGS estimates purity during clustering of CNAs and SNVs. Whereas PhyloWGS estimates phylogenies through MCMC by sampling over a distribution of phylogenies, DPClust reconstructs phylogenies after MCMC, using a set of rules.

Overall, PhyloWGS is more flexible in how it treats copy number aberrations and their effects on SNV allele frequencies. This gives it superior ability in reconstructing phylogenies, but means that it is more likely to be misled by miscalled CNAs and SNVs, resulting in inferior estimates of tumor purity.