novoBreak: local assembly for breakpoint detection in cancer genomes

Zechen Chong1,8, Jue Ruan2,8, Min Gao3,8, Wanding Zhou1, Tenghui Chen1, Xian Fan1, Li Ding4, Anna Y Lee5, Paul Boutros5–7, Junjie Chen3 & Ken Chen1

We present novoBreak, a genome-wide local assembly algorithm that discovers somatic and germline structural variation breakpoints in whole-genome sequencing data. novoBreak consistently outperformed existing algorithms on real cancer genome data and on synthetic tumors in the ICGC-TCGA DREAM 8.5 Somatic Mutation Calling Challenge primarily because it more effectively utilized reads spanning breakpoints. novoBreak also demonstrated great sensitivity in identifying short insertions and deletions.

Somatic structural variation is a major driving force of tumor initiation and progression. Sporadic and recurrent chromosomal aberrations have been observed in most cancer types and many of these aberrations are desirable therapeutic targets. The advent of high-throughput sequencing has made it possible to detect structural variants (SVs) genome wide at base-pair resolution. However, computational approaches exhibit limited sensitivity and only apply to a restricted set of SV types.

One approach for SV detection is to align paired-end short reads to a reference genome and identify signals in discordant read pairs, read depth, split reads, or combinations of these signals. Another approach relies on targetted local assembly of aligned and partially aligned reads in candidate SV regions that were discovered a priori. These approaches depend heavily on read alignment accuracy, which is often limited for reads that span breakpoints or differ substantially from the reference genome. In comparison with these approaches, whole-genome assembly approaches are less biased. However, assembling a whole genome is computationally intensive, and results are often affected by repeats, polyploidy, read length, and sequencing coverage.

We developed novoBreak, an algorithm that generates local assemblies of breakpoints genome wide. Assemblies are based on clusters of reads which share a set of short nucleotide stretches of length $k$ ($k$-mers) that are present in a subject genome (e.g., a tumor genome) but not in the reference genome or control data (e.g., a matched normal genome; see Online Methods and Fig. 1). When applied to somatic breakpoint detection from matched tumor and normal tissue data, novoBreak constructs a hash table that contains all the $k$-mers, their host reads and frequencies of the $k$-mers (see Online Methods and Supplementary Note 1). Next, it filters out $k$-mers representing reference alleles or sequencing errors and retains those representing variants or novel sequences not present in the reference genome. It then queries the normal reads and further classifies the $k$-mers into (1) germline $k$-mers, those present in both the tumor and the normal genome; and (2) somatic $k$-mers, those present in the tumor but not the normal genome. Then, novoBreak identifies clusters of read pairs spanning each somatic breakpoint and assembles each cluster of reads into contigs (see Online Methods and Supplementary Note 2). By comparing the resulting high-quality contigs with the reference, novoBreak identifies breakpoints and associated SVs. Finally, novoBreak outputs a quantitative report of supporting evidence at each breakpoint.

We examined the performance of novoBreak in the ICGC-TCGA DREAM 8.5 Somatic Mutation Calling Challenge (https://www.synapse.org/#!Synapse:syn312572), which is designed to identify the best algorithms for detecting somatic mutation in next-generation sequencing data. In each of the synthetic subchallenges, a high coverage (60–80×) whole-genome sequencing (WGS) bam file produced from a cell line or patient tissue was divided into two parts (30–40× each). One part was treated as the normal data and the other as the tumor data containing mutations spiked in by BAMSurgeon. Four in silico subchallenges (IS) were implemented with increasing numbers and types of variants and cellular complexity. In total, 204 submissions were made by 27 teams that developed widely used SV detection tools including Breakdancer, Delly, PinDel, and Manta (https://github.com/StructuralVariants/manta). novoBreak consistently achieved the best balanced accuracy (sensitivity and precision) in IS2, IS3 and IS4 (Supplementary Table 1). Almost all the top-performing tools achieved high precision (>0.98) after stringent filtering, but novoBreak achieved high sensitivity, which was particularly evident when insertions were introduced in IS2 and IS3.

The most challenging synthetic tumor was presented in the IS3 subchallenge; it contained single-nucleotide variants, deletions, duplications, inversions and insertions (mobile elements), as well as insertions and deletions shorter than 100 bp (indels); it also included subclones at 50%, 33%, and 20% cellular fraction. novoBreak achieved the highest balanced accuracy of 0.892 (sensitivity, 0.801; precision, 0.984), primarily on account of its higher sensitivity...
in detecting insertions (Fig. 2a). It discovered 100 (4.3%) and 120 (5.1%) more insertions in the ground truth than did DELLY and Manta, respectively. Compared with alignment-based approaches, novoBreak more effectively used reads spanning insertion breakpoints (Supplementary Fig. 1). Further analysis of the SVs missed by DELLY and Manta indicated that novoBreak performs better in low-coverage regions with few discordantly paired or split reads (Supplementary Note 3). Breakpoints identified by novoBreak also had the highest precision; 98.9% were within −2 bp to 2 bp relative to the ground truth (Fig. 2b).

Detection of indels, particularly insertions, is challenging because of difficulties in achieving accurate short-read alignment. NovoBreak ranked second and first for indel detection in IS3 and IS4, respectively (Supplementary Table 2). The IS4 genome contained three times more simulated indel and SNV events than did the other subchallenges, and it included subclonal events at relatively low allelic frequencies (15%). Encouragingly, novoBreak achieved a balanced indel detection accuracy of 0.857 (sensitivity, 0.788; precision, 0.926), close to the best SNV detection accuracy (Fig. 2c). GATK-HaplotypeCaller had significantly lower sensitivity in detecting 1-, 2-, and 3-bp indels, likely on account of limitations of aligning short reads and stringent filtering. In contrast, Strelka demonstrated reduced sensitivity as indel size increased. It did not report any insertion longer than 25 bp.

We compared novoBreak with BreakDancer (v1.1.2), DELLY (v0.6.3) and Fermi (v1.1.751-beta) using WGS data from the melanoma tumor cell line COLO-829 (ref. 17). Fermi is a string-graph-based whole-genome assembler that retains contigs containing SNPs, indels, and SVs. Because Fermi does not come with a ready-to-use tool to call SV breakpoints, we used the SV-calling steps of novoBreak to evaluate Fermi’s assembly results. These data were previously analyzed by a read-pair approach, and CREST and Sanger sequencing (Supplementary Table 3). We used these 48 breakpoints as ground truth to benchmark these tools. Under default parameters, BreakDancer identified 37 true positives (TPs), with a total of 14,340 predicted; DELLY, 34 TPs, with 1,113 predicted; and Fermi, 40 TPs, with 16,849 predicted. A large fraction of SVs reported by these tools were likely germline instead of false SVs. In contrast, novoBreak identified 44 TPs with 78 breakpoints predicted (Fig. 2d and Supplementary Table 4). Of the four missing TPs, two were missed by all the tools, and two could be recovered by novoBreak under less stringent settings. We designed PCR primers around the 34 novel breakpoints and validated 9 by PCR and Sanger sequencing (Supplementary Table 5). The remaining novel breakpoints were not necessarily false calls and could be attributed to deficiency in validation experiments or evolution of the cultured cell line. Indeed, 19 (57.6%) of the 34 calls were also predicted by at least one other tool. These results demonstrate novoBreak’s high sensitivity and specificity in analyzing real tumor data under default settings. Users can adjust the filtering parameters to obtain different sensitivity and specificity tradeoff in different applications.

To further evaluate the sensitivity of novoBreak on cancer patient data, we analyzed the WGS data of a patient with low-grade glioma (Supplementary Note 4) and those of 22 invasive breast carcinoma samples in The Cancer Genome Atlas (TCGA). This set of TCGA samples was analyzed previously by INTEGRATE, which integrates matched whole-genome and whole-transcriptome sequencing (WTS) data to discover gene fusions. Overall, novoBreak identified 1,628 deletions; 1,724 duplications; 2,335 inversions; and 1,982 translocations, equivalent to 349 SVs per sample (Supplementary Tables 6 and 7). It identified 104 (86.7%) of the 120 known high-confidence gene fusions (Supplementary Table 8). The true sensitivity was probably higher because 19% of the known SVs were likely false positives. In addition, they were identified using both WGS and WTS data; whereas novoBreak only examined the WGS data.

The most significant improvement of novoBreak over other approaches is the k-mer identification, filtering, and classification strategy, which substantially narrows down the number of putative SV breakpoints and focuses computational power on the most informative portion of the data. By clustering and performing local assembly around breakpoints, novoBreak takes full advantage of
unmapped reads and/or partially mapped reads. The scoring and filtering strategy of novoBreak provides high precision. A caveat of novoBreak is that it misses SV breakpoints in repetitive sequences longer than 2k − 1 bp. Future versions with increased k should alleviate this limitation. We expect that the k-mer targeted assembly framework of novoBreak will facilitate comprehensive, sensitive, efficient, and accurate identification of novel sequence alterations in genomic, exomic, and transcriptomic sequencing data. The source code of novoBreak (Supplementary Software) is freely available for academic use at http://sourceforge.net/projects/novobreak/.

METHODS

Methods, including statements of data availability and any associated accession codes and references, are available in the online version of the paper.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

ACKNOWLEDGMENTS

We thank the ICGC-TCGA DREAM SMC Challenge organizers and participants for providing data and evaluations; and we thank A.K. Eterovic and G.B. Mills for assistance with the experiment and manuscript. This study was supported in part by the National Institutes of Health (grant numbers R01 CA172652 to K.C. and U41 HG007497 to C. Lee, Jackson Lab), the National Cancer Institute Cancer Center Support Grant (P30 CA016672 to R. Depinho, MD Anderson Cancer Center), Andrew Sabini Family Foundation to K.C., and a training fellowship from the Computational Cancer Biology Training Program of the Gulf Coast Consortia (CPRIT grant number RP140113) to Z.C. The results published here are in part based upon data generated by TCGA established by the NCI and NHGRI. Information about TCGA and the investigators and institutions who constitute the TCGA research network can be found at http://cancergenome.nih.gov/.

Figure 2 | novoBreak performance on DREAM 8.5 Mutation Calling Challenge data. (a) Precision and recall comparison among three top-performing tools on IS3 synthetic tumor data. Star indicates the best results for each tool. (b) Breakpoint precision for the three tools on IS3 data. X-axis is the offset (in base pairs) between the true and predicted breakpoint coordinates. Y-axis is the fraction of predicted breakpoints at each of the offset values. (c) Indel detection sensitivity of three tools in the IS4 data. (d) Summary of SV breakpoints detected by four tools in COLO-829 cell line data.

AUTHOR CONTRIBUTIONS

Z.C., J.R., and K.C. conceived the algorithm. Z.C. developed the software. Z.C. and K.C. designed and analyzed the experiments. M.G. and J.C. designed and performed the validation experiments. W.Z. designed the scoring statistics. M.G., T.C., X.F., L.D., A.Y.L., and P.B. tested the algorithm and performed additional analyses. K.C. supervised the projects. Z.C. and K.C. wrote the manuscript with input from all authors. All authors have read and approved the final manuscript.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

Reprints and permissions information is available online at http://www.nature.com/reprints/index.html.

1. Mitelman, F., Johansson, B. & Mertens, F. Nat. Rev. Cancer 7, 233–245 (2007).
2. Stephens, P.J. et al. Nature 462, 1005–1010 (2009).
3. Alkan, C., Coe, B.P. & Eichler, E.E. Nat. Rev. Genet. 12, 363–376 (2011).
4. Alkan, C., Sajjadian, S. & Eichler, E.E. Nat. Methods 8, 61–65 (2011).
5. Chen, K. et al. Nat. Methods 6, 677–681 (2009).
6. Abyzov, A., Urban, A.E., Snyder, M. & Gerstein, M. Genome Res. 21, 974–984 (2011).
7. Ye, K., Schulz, M.H., Long, Q., Apweiler, R. & Ning, Z. Bioinformatics 25, 2865–2871 (2009).
8. Rausch, T. et al. Bioinformatics 28, i333–i339 (2012).
9. Chen, K. et al. Genome Res. 24, 310–317 (2014).
10. Li, Y. et al. Nat. Biotechnol. 29, 723–730 (2011).
11. Earl, D. et al. Genome Res. 21, 2224–2241 (2011).
12. Boutros, P.C. et al. Nat. Genet. 46, 318–319 (2014).
13. Ewing, A.D. et al. Nat. Methods 12, 623–630 (2015).
14. McKenna, A. et al. Genome Res. 20, 1297–1303 (2010).
15. Saunders, C.T. et al. Bioinformatics 28, 1811–1817 (2012).
16. Li, H. Bioinformatics 28, 1838–1844 (2012).
17. Pleasance, E.D. et al. Nature 463, 191–196 (2010).
18. Wang, J. et al. Nat. Methods 8, 652–654 (2011).
19. Zhang, J. et al. Genome Res. 26, 108–118 (2016).
ONLINE METHODS
The *novoBreak* pipeline. *novoBreak* is developed to comprehensively discover exact chromosomal breakpoints introduced by structural variations in genomes or transcriptomes. It is based on (1) a genome-wide classification and filtering strategy, which identifies specific nucleotide signatures (*novo k*-mers) and (2) a local assembly approach, which constructs breakpoint sequences from reads containing the *novo k*-mers. The workflow of *novoBreak* consists of the following steps (Fig. 1): (1) *novoBreak* begins with an indexing and filtering procedure to obtain *novo k*-mers; and associated short reads, described in the section “Indexing and filtering *k*-mers”. (2) Paired-end reads containing the same set of *novo k*-mers are clustered together. Each cluster contains read pairs covering the same breakpoint. An assembly algorithm is then applied to each cluster to construct a breakpoint-spanning sequence. The clustering and local assembly step is described later in the section “Clustering and local assembly algorithm.” (3) After short reads are assembled in each cluster, the resulting contigs are aligned to the reference using BWA-MEM\(^{20}\) ([Supplementary Note 1](#supplementary)) with ‘-M’ option to obtain secondary alignments. The alignment results are parsed to infer breakpoints and the associated SVs. For short SVs, such as INDELs, *novoBreak* directly parses the Compact Idiosyncratic Gapped Alignment Report (CIGAR) strings of the aligned contigs. For large SVs, *novoBreak* considers both the primary and the secondary alignments of each contig. In current implementation, *novoBreak* predicts deletions, insertions, inversions, duplications, and translocations at base-pair resolution. (4) To achieve a high precision, *novoBreak* employs a scoring and filtering module, as described in the section “Scoring method.”

Indexing and filtering *k*-mers. Given a sequence \(S\) of length \(L\), a *k*-mer is a length \(k\) \((k < L)\) substring of sequence \(S\). We notice that if a read \(R\) contains a breakpoint of a structural change with respect to the reference or the normal genome of a cancer patient, there are at most \(k - 1\) *k*-mers \((k < |R|)\) covering the breakpoint. The default \(k\) is 31 ([Supplementary Note 1](#supplementary)) in *novoBreak*. We define these *k*-mers as ‘*novo k*-mers’ because they contain novel sequence information specific to the subject. In a tumor-normal paired cancer genome sequencing study, the *novo k*-mers contain the somatic breakpoints that specifically exist in the tumor but not in the paired normal sample. The first critical step of *novoBreak* is to obtain the *novo k*-mers. An effective approach for this is to implement a hash table that first indexes and loads all the *k*-mers in all the reads in the tumor sample into the memory and then eliminate *k*-mers that are present in the reference or the normal genome. The remaining high-frequency *k*-mers should contain genuine somatic breakpoints, including SNVs, small indels, and large SVs. This approach is computationally feasible for whole-exome or whole-transcriptome analysis. But for high-coverage whole-genome analysis, the memory cost is extremely high (usually a few hundred gigabytes) mainly because of the presence of sequencing errors. A critical component of *novoBreak* is the reduction of memory consumption. For whole-genome sequencing data, instead of indexing the sequenced reads, *novoBreak* starts from hashing all the *novo k*-mers in the reference genome. Then, it adopts a two-pass approach to calculate *novo k*-mers in the sequenced genomes. The first pass scans every reads and marks the status (presence or absence) of each constituent *k*-mer in the reference genome using the preconstructed hash table. In the process, *novoBreak* automatically trims off error-prone ends in low-quality reads ([Supplementary Note 1](#supplementary)). *novoBreak* uses a bit array data structure to mark a read. If a *k*-mer in a read is in the hash table, it will be marked as 1 (otherwise 0) in the corresponding bit in the bit array. When all the reads are processed, the hash table for the reference *k*-mers is released. Next, *novoBreak* goes through the reads containing at least one 0 bit to obtain the minimal occurrence of the nonreference *k*-mers. *novoBreak* adopts Bloom filter\(^{21}\), a probabilistic data structure that tests whether a given element is in a set. A Bloom filter is a bit array of \(m\) bits, initialized to be 0. \(k\) different hash functions are applied to an element and map the element to \(k\) different positions in the array. To add an element, these \(k\) positions will be set to 1. To test whether an element is in the set, each of the \(k\) positions will be examined. If there is a 0 at any of the \(k\) positions, the element is definitely not in the set. If all the \(k\) positions are 1, then either the element is in the set or the positions were coincidently set to 1 by other elements. Such false positive (FP) errors could happen because different elements could be coincidently hashed to same positions in the bit array. Fortunately, the chance of having an error is very small, less than

\[
\frac{1 - e^{-\frac{k(n + 0.5)}{m - 1}}}{k^n},
\]

where \(n\) is the total number of elements, \(m\) is the size of the bit array of the Bloom filter, and \(k\) is the number of hash functions. Note these rare FP errors do not hurt sensitivity and have negligible possibility of introducing FP breakpoints on account of the subsequent read clustering, assembly, alignment, and variant calling steps. We expand the above standard Bloom filter from one bit to two or more (default to three bits in *novoBreak*) to count if a *k*-mer has occurred more than a minimal number of times (default three in *novoBreak*; [Supplementary Note 1](#supplementary)) in the data set. Thus, *k*-mers introduced by sequencing errors will be automatically disregarded, and the remaining are *novo k*-mers from the variant alleles. For somatic analysis, *novoBreak* will further scan the normal control reads using a hash table and count the occurrence of these *k*-mers in the normal reads. Based on these counts, candidate somatic *k*-mers (i.e., *k*-mers only present in the tumor but not the normal sample) can be identified, with the effect of cross-contamination between the samples being accounted for. Finally, *novoBreak* loads read pairs containing the candidate somatic *k*-mers and automatically removes duplicated read pairs that have identical sequences in both reads.

Clustering and local assembly. With *novo k*-mers and the associated read pairs identified, a straightforward method is to assemble all the read pairs directly. However, the cost of assembly is still very high because of a large number of reads. In addition, presence of alternative alleles, repeats, and sequencing errors can easily cause misassemblies. Note that, as shown in [Supplementary Figure 2](#supplementary), at each breakpoint there are \(k - 1\) *novo k*-mers with many reads covering them. Reads covering the same breakpoint share a subset of the \(k - 1\) *novo k*-mers. Based on this pairwise relationship between *k*-mers and reads, we can find the set of read pairs covering a breakpoint using a union-find algorithm\(^{32}\), which identifies all the connected components in an undirected graph consisting of reads and *k*-mers (as nodes) and their connections (as edges). To avoid having large clusters with many reads due
to repeats or sequencing errors, novoBreak trims the connected components based on read and k-mer statistics. For the purpose of detecting SVs, the computational cost is further reduced by directly reading from bam files and correcting base errors based on high-quality aligned reads.

After clustering, it is relatively easy to locally assemble the read pairs in each cluster, since the number of read pairs is small, and read pairs originate from the same locus of an allele. Almost every modern assembler can be applied for such a task. novoBreak pipeline uses SSAKE\textsuperscript{23} (Supplementary Note 2) to assemble read pairs into contigs. The setting of SSAKE in novoBreak is \( \text{''p 1 -k 2 -n 1 -m 16 -x 3 -w 1 -z 30 -o 1''} \). SSAKE can generate multiple contigs from each cluster. Each contig is aligned by BWA-MEM and analyzed independently. After all the candidate breakpoints are generated, novoBreak merges them and creates a unique set of SVs.

**Somatic structural variant scoring methods.** novoBreak scores and ranks each predicted breakpoint based on assembly and mapping results. At a given locus, novoBreak calculates a statistical quality score

\[
Q = -10 \log_{10} \left( \frac{\Pr[D \mid G = 0,2]}{\Pr[D \mid G = 1]} \right),
\]

where \( D = \{ D_{I,R} \} \) comprises of the counts of read pairs supporting the reference allele \( R = r \) and those supporting the variant allele \( R = v \) from the tumor \( \{ I = t \} \) and the normal \( \{ I = n \} \) data, respectively; \( G = 0,1,2 \) indicates whether the locus has a reference (no SV in either tumor or normal), somatic (SV only in tumor) or germline (SV in both tumor and normal) status.

We can compute the likelihood of the data, given the status of a locus. For example, likelihood of the somatic status \( G = 1 \) can be estimated as:

\[
\Pr[D \mid G = 1] = \prod_{I = t_1}^{n} \Pr[D_{I,R} \mid G = 1]
\]

Because the variant allele fraction in the tumor is unknown, novoBreak uses a beta-binomial distribution to estimate the likelihood of the observed read counts. For example, the number of read pairs supporting a breakpoint in the tumor sample is

\[
\Pr[D_{I,R} \mid G = 1] = \Pr(D_{I,R} \mid D', \hat{a}_{i,R}, \hat{a}_{i,G})
\]

where \( \hat{a}_{i,R} \) and \( \hat{a}_{i,G} \) indicate the parameters used for the combinations among \( I \in \{ t, n \} \) and \( G \in \{ 0, 1, 2 \} \). For the somatic status \( G = 1 \), novoBreak initializes \( \hat{a}_{n,1} = 10 \) and \( \hat{a}_{t,1} = 1 \) to reflect the concept that SV signal in the normal sample is largely due to noise. For the germline status \( G = 2 \), novoBreak uses a uniform distribution \( \hat{a}_{n,2} = \hat{a}_{n,2} = \hat{a}_{t,2} = 10 \). For the reference status \( G = 0 \), \( \hat{a}_{n,0} = \hat{a}_{t,0} = 10 \). The reference and somatic indels with the FILTER field "PASS" was too few, we also selected “QSI_ref” field for evaluation.

**Indel analysis.** Indels detection on the IS4 data was performed by novoBreak, GATK-HaplotypeCaller and Strelka as follows.

**novoBreak.** novoBreak (v1.03) was run under the parameters \( \text{'-k31 -m2'} \). All the assembled contigs and unassembled short read pairs containing the novo \( k \)-mers were mapped to the reference using BWA-MEM\textsuperscript{20}. The alignment results were sorted, and the coordinates of indels were adjusted using SortSam of Picard (v1.107) (http://broadinstitute.github.io/picard/) and LeftAlignIndels of GATK\textsuperscript{14,24,25} (v2.8-1), respectively. The CIGAR strings of the alignment results were parsed to generate an indel list (in VCFv4.1 format). Indels were further filtered using Database of Single Nucleotide Polymorphisms dbSNP (build ID, 138; available from http://www.ncbi.nlm.nih.gov/SNP/) and low-complexity regions identified with the mdust program (http://compbio.dfci.harvard.edu/tgi/). Finally, only indels with allele fraction greater than 1% were selected.

**GATK-HaplotypeCaller.** GATK v2.8-1 was run on the same data. First, tumor and normal bam files were realigned using IndelRealigner and left aligned using LeftAlignIndels. HaplotypeCaller was run on tumor and normal bam files with the parameters \( \text{'--genotyping_mode DISCOVERY -stand_emit_conf 10 -stand_call_conf 30'} \), respectively. Then, SelectVariants of GATK was run with parameter \( \text{'--selectType INDEL'} \) to generate an indel VCF file for the tumor and the normal. Indels from the tumor and the normal samples were further filtered using VariantFilter with parameters \( \text{'--filterExpression "QD < 2.0 || FS > 200.0 || ReadPosRankSum < -20.0"'} \). Finally, only indels in the tumor VCF file, but not in the normal VCF file, with “PASS” labels were evaluated as somatic indels.

**Strelka.** Strelka\textsuperscript{13} (v1.0.14) was also tested on the data set. Files and directories were generated and configured according to the documentation (https://sites.google.com/site/strelkasomaticvariantcaller/). Strelka was run with the default parameters. Since the number of somatic indels with the FILTER field “PASS” was too few, we also selected “QSI_ref” field for evaluation.

**Experimental validation.** The COLO-829 and COLO-829BL cell lines were purchased from the American Type Culture Collection (ATCC). Primers for genomic PCR were designed using Primer3 (http://biotools.umassmed.edu/bioapps/primer3_www.cgi). The COLO-829 and COLO-829BL cells were cultured in RPMI 1640 (Sigma) supplemented with 10% fetal bovine serum (FBS) and 1% penicillin and streptomycin. Genomic DNA was extracted from COLO-829 and COLO-829BL cells using genome DNA kit (Invitrogen), and the PCR was performed using GoTaq DNA Polymerase (Promega). Thermal cycling conditions were one cycle of 95 °C for 2 min; followed by 30 cycles of 95 °C for 30 s, 65 °C for 30 s, and 72 °C 1 min; followed by a final extension step of 72 °C for 10 min. PCR products were electrophoresed on 1% agarose gels with ethidium bromide, visualized using UV light illumination.

**Code availability and system requirements.** novoBreak is written in C and Perl. The source code is freely available at https://sourceforge.net/projects/novobreak/?source=updater. For a 40x 2 \times 101 bp whole-genome tumor and normal pairs, novoBreak needs a main memory less than 40 GB and a running time less than ~6 h with 10 CPU cores.

doi:10.1038/nmeth.4084

NATURE METHODS
Data availability. ICGC-TCGA DREAM Challenge data\textsuperscript{13} (SRA, SRP042948) were downloaded from https://www.synapse.org/#!Synapse:syn2280639 with public token (\textit{in silico} 1, 2, and 3) or approval access with private token from ICGC (\textit{in silico} 4).

The whole-genome sequencing data\textsuperscript{17} (EGAS00000000055) of the immortal melanoma cancer cell line COLO-829 and lymphoblastoid cell line derived from the same patient COLO-829BL was requested from the European Genome-phenome Archive.

The TCGA breast cancer WGS data were obtained through dbGAP (accession number phs000178.v7.p6).

The low-grade glioma sample (SJLGG039) is available at European Genome-phenome Archive under accession EGAS00001000255.

20. Li, H. Preprint at https://arxiv.org/abs/1303.3997 (2013).
21. Bloom, B.H. Communications of the ACM 13, 422–426 (1970).
22. Sedgewick, R. & Wayne, K. Algorithms 4th edn. (Addison-Wesley, 2011).
23. Warren, R.L., Sutton, G.G., Jones, S.J.M. & Holt, R.A. Bioinformatics 23, 500–501 (2007).
24. DePristo, M.A. et al. Nat. Genet. 43, 491–498 (2011).
25. Van der Auwera, G.A. et al. Curr. Protoc. Bioinformatics 11, 43.11.10.1–43.11.10.33 (2013).