Changepoint Analysis for Efficient Variant Calling

Adam Bloniarz[1,⋆], Ameet Talwalkar[2,⋆], Jonathan Terhorst[1,⋆], Michael I. Jordan[1,2], David Patterson[2], Bin Yu[1], and Yun S. Song[1,2,3,†]

[1] Department of Statistics, University of California, Berkeley
[2] Computer Science Division, University of California, Berkeley
[3] Department of Integrative Biology, University of California, Berkeley

⋆ These authors contributed equally.
† Corresponding author: yss@eecs.berkeley.edu

Abstract. We present CAGE, a statistical algorithm which exploits high sequence identity between sampled genomes and a reference assembly to streamline the variant calling process. Using a combination of changepoint detection, classification, and online variant detection, CAGE is able to call simple variants quickly and accurately on the 90-95% of a sampled genome which differs little from the reference, while correctly learning the remaining 5-10% that must be processed using more computationally expensive methods. CAGE runs on a deeply sequenced human whole genome sample in approximately 20 minutes, potentially reducing the burden of variant calling by an order of magnitude after one memory-efficient pass over the data.

Keywords: genome complexity, next-generation sequencing, variant calling, changepoint detection

1 Introduction

A central goal in computational biology is to accurately reconstruct sampled genomes from next-generation sequencing (NGS) data, a procedure termed *variant calling*. A vast number of algorithms have been developed in pursuit of this goal, and they are notoriously computationally demanding. This is due both to the difficulty of the underlying problem, as well as the sheer size of the data: a whole human genome sequenced to 30× coverage produces roughly 250 GB of sequence information and metadata; thus even one sample cannot be represented in memory on a typical workstation. As a result, variant calling algorithms spend significant time simply transferring and storing the information needed to carry out the analysis.

A potential solution to this problem is to harness inherent similarity in genetic data. Unrelated humans are estimated to share over 99% sequence identity [1], so most sequencer output will be similar to a corresponding region of the human reference sequence. So-called "reference-based" compression techniques

which exploit this feature have been proposed [2, 3, 4]; however few existing tools can operate natively on reference-compressed genomic data. The standard compressed format for aligned sequence data [BAM; 5] enjoys widespread support, but only achieves roughly 50% compression owing to its use of a generic compression algorithm (`zlib`).

In lieu of compression, a promising alternative is to use statistical methods to discover the small fraction of the sampled genome believed to harbor interesting variation, and focus further computational resources in these limited regions. We formalize this idea in terms of *complexity*. Regions which are highly mutated, have low coverage and/or were subject to sequencing errors are *complex*: they contain additional signal which cannot be retrieved from the reference genome. Conversely, regions which are similar to the reference, display expected coverage levels, and show low rates of mutation and sequencer error have low-complexity. Our goal then becomes to algorithmically classify genomic regions according to their complexity level. Concretely, we propose the following hybrid approach:

1. Perform a first-pass analysis to isolate a small fraction of the sampled genome which is "non-reference" and complex;
2. Pass these high-complexity regions to the computationally intensive variant detection algorithms described above;
3. Process the remaining low-complexity regions using a fast algorithm designed to detect simple variation.

In this work, we explore methods to isolate such regions by exploiting statistical features of the sequencer output, which can be computed quickly and without recourse to fully decoding the underlying genome. Our algorithm, called **C**hangepoint **A**nalysis of **Ge**nomic reads (CAGE), is fast and trivially parallelizable across the genome, and hence well-suited to process large amounts of NGS data quickly. Using several benchmark datasets, we demonstrate that our approach maintains state-of-the-art variant calling accuracy while subjecting less than 10% of the sampled genome to computationally intensive analysis. Additionally, we present an extension of our algorithm, called CAGE++, in which we simultaneously perform variant detection and variant calling on low-complexity genomic regions, potentially obviating the need for the third step of the hybrid approach described above. Finally, our approach is very cheap when compared to standard analysis tools [e.g. 5, 6, 7]: CAGE and CAGE++ can process a human whole genome in approximately 20 minutes on a single 32 core machine while consuming less than 16 GB of memory, thus illustrating that our proposed hybrid variant calling pipeline has the potential to significantly speedup the variant calling process.

2 Related work

High-throughput sequencing has inspired various efforts aimed at reducing the amount of data needed to be stored and analyzed, primarily in the form of compression algorithms. Lossless compression methods include reference-based

approaches [3, 4] which store variation relative to a reference sequence, as well as non reference-based methods which specialize certain existing compression techniques to genomic data [2]. Greater compression ratios may be achieved if sequencer quality scores are lossily compressed while still losslessly compressing the actual sequence data [8]. The primary disadvantage of these techniques is that few existing software tools can operate directly on the compressed data, mandating a time- and space-intensive decompression step each time the data are analyzed.

Recent versions of the Genome Analysis Toolkit [GATK, 6] employ a lossy compression tool, ReduceReads, to reduce alignment data before being processed by other variant calling tools. The tool works by discarding data in regions of the genome which contain little variation, and is thus similar in motivation to the algorithm we report here. However, the algorithms differ in several regards. CAGe is based on a statistical model of the observed data (Section 3), and is tuned using intuitive quantities such as read coverage rate, sequencer error rate, and mutation rate. ReduceReads appears to employ several heuristics when creating the compressed output, and it is not necessarily clear how to parameterize these heuristics in order to achieve a desired compression ratio or data fidelity. Additionally, though we are unaware of any formal publication or other effort to benchmark the ReduceReads algorithm, user reports from the GATK support forums indicate that it requires costly preprocessing steps in order to run, and can require a large amount of memory and processing time in order to compress a whole genome sequence.

CAGe uses a changepoint detection method to mark regions of variable complexity as it moves along the genome. A similar idea was used by Shen and Zhang [9] to detect abrupt changes in copy number. One way to view these methods is as an alternative to the hidden Markov model (HMM), which has also been previously used to detect genomic variation [10, 11]. In contrast to the latter methods, which require the number of hidden states to be known *a priori*, changepoint methods allow the number of detected segments to vary in accordance with the data. We leverage this observation, in conjunction with simple rule-based classifier, to learn the number of hidden genomic complexity states in a semiparametric manner.

Various distributional aspects of the data we consider have been previously studied. In a seminal paper, Lander and Waterman [12] showed that read depth in whole-genome shotgun sequencing experiments is well modeled by a Poisson distribution, a fact which we exploit in our model. Evans et al. [13] considered fragment site–insert length pairs embedded into the plane. This construction can be used to derive null distributions of several coverage-related statistics [13, 14]. They also describe an interesting visualization technique which can be used to detect deviations from the null coverage distribution. This approach is similar in spirit to our goal, but here we rely on automated techniques in order to detect these deviations in a high-throughput environment.

3 Methods

Following sequencing and alignment, evidence of genetic variation in NGS data is detectable in several ways. Sites which harbor isolated, single-nucleotide variants can usually be aligned unambiguously to the reference genome, resulting in a characteristic SNP signature common to half or all of the reads (depending on zygosity) in the alignment. Small insertions and deletions (≈2-10bp) are also frequently detected and compensated for by the aligner. In both of these scenarios, coverage and concordance statistics are usually unaffected by the presence of the nearby variant since the aligner is able to "explain away" the variant.

In contrast, larger structural variants produce several noticeable signals in the alignment. Novel insertions are typically flanked by reads with low mapping quality or missing mate pairs, and may also result in a coverage dropoff or decreased insert size near the insertion site. Similarly, deleted regions are evidenced by larger than expected insert sizes and a coverage dropoff. Reads that straddle the boundary of a structural variant often have a high percentage of soft-clipped bases with high Phred-scores. More complicated forms of rearrangement result in other distinctive patterns involving, for example, split mapping and orientation bias [15].

Formally, we define complexity in terms of point processes and their associated rates. At genomic position i, let

- $M_i \in \{0, 1\}$ denote the (unobserved) mutation state, assuming a biallelic mutation model;
- $R_i \in \mathbb{Z}^+$ the number of short-reads whose alignment begins at i;
- $D_i \geq R_i$ be number of sequenced bases ("coverage depth") at i; and
- $\mathbf{E}_i = (e_{i,1}, \ldots, e_{i,D_i}) \in \{0, 1\}^{D_i}$ denote a vector of indicators for whether a sequencing error occurred in each of the D_i bases aligned to i.

Note that we observe the random variables R_i and D_i but not M_i or $\mathbf{E}_i$; the M_i are what we ultimately hope to infer through later variant calling analysis, and we only observe a noisy signal of $\mathbf{E}_i$ through the sequencer quality score.

These random variables generate our data as follows. After sequencing and read mapping, we observe a collection of vectors $\mathbf{B}_1, \ldots, \mathbf{B}_L$, where L is the length of the reference genome ($\approx 3.3 \times 10^9$ in humans) and $\mathbf{B}_i = (B_{i,1}, \ldots, B_{i,D_i})$ is the vector of sequenced bases at site i, with

$$\begin{aligned} B_{i,j} &= \mathbf{1}\{\text{base } j \text{ at location } i \text{ matches the reference}\} \\ &= M_i(1 - e_{i,j}) + (1 - M_i)e_{i,j}. \end{aligned}$$

To compute the likelihood of the data, we make the following distributional assumptions:

- The R_i are independent and Poisson distributed with intensity λ_i [12].
- Conditional on the $R_1, \ldots, R_i$, the coverage depth D_i is deterministic.
- $M_i \sim \text{Bernoulli}(\mu_i)$ has a Bernoulli distribution with success parameter μ_i, the probability that a mutation occurs at i.

- The indicators $e_{i,j}$ have a common Bernoulli(ϵ_i) distribution.
- R_i, $e_{i,j}$ and M_i are mutually independent within and across sites.
- All reference bases at a mutated site are sequencer errors, as are all non-reference bases at a non-mutated site.

These assumptions are not expected to hold for real data, however they lead to a fast, easily estimated model, and moreover they do not appear to greatly affect the quality of our inference. Henceforth we abbreviate the genomic region $\{i, i+1, \ldots, j-1, j\}$ as $i : j$. In a region of uniform genomic complexity we expect that the parameters $\lambda_k, \mu_k, \epsilon_k$ are approximately constant, $(\lambda_k, \mu_k, \epsilon_k) = (\lambda, \mu, \epsilon) \triangleq \theta$ for $k \in i : j$.

The complete log-likelihood of the data, $\ell_\theta(i : j)$, is then

$$\ell_\theta(i, j) \triangleq \ell_\theta(\mathbf{B}_{i:j}, M_{i:j}, \mathbf{E}_{i:j}) = \ell_\theta(R_{i:j}) + \ell_\theta(M_{i:j}) + \ell_\theta(\mathbf{E}_{i:j} \mid M_{i:j}, \mathbf{B}_{i:j}) \quad (1)$$

where:

$$\ell_\theta(R_{i:j}) = \sum_{b=i}^{j} \log \mathcal{P}_\lambda(R_b) \quad (2)$$

$$\ell_\theta(M_{i:j}) = \sum_{b=i}^{j} \left[M_b \log \mu + (1 - M_b) \log(1 - \mu)\right] \quad (3)$$

$$\ell_\theta(\mathbf{E}_{i:j} \mid M_{i:j}, \mathbf{B}_{i:j}) = \sum_{b=i}^{j} \left[(1 - M_b)\bar{B}_b \log \epsilon + M_b(D_i - \bar{B}_b) \log(1 - \epsilon)\right], \quad (4)$$

$\mathcal{P}_\lambda$ is the Poisson likelihood with rate λ and $\bar{B}_b = \sum_k B_{b,k}$.

Let $\ell_{\hat{\theta}}(i : j) \triangleq \sup_\theta \ell_\theta(i : j)$ denote the maximized log-likelihood. It is clear from the additive form of (2)—(4) that for any $i \leq k \leq j$, we can always increase the likelihood of the data by breaking $i : j$ into two independent segments $i : k$ and $(k + 1) : j$:

$$\ell_{\hat{\theta}}(i : j) \leq \ell_{\hat{\theta}}(i : k) + \ell_{\hat{\theta}}((k + 1) : j).$$

In what follows we use this observation to quickly detect uniformly complex regions using likelihood-based methods.

3.1 Maximum likelihood estimation

The simple form of the complete likelihood (1) suggests using the EM algorithm to compute $\ell_{\hat{\theta}}(i : j)$. To do so, we must evaluate the conditional expectation

$$\mathbb{E}_{M_{i:j}, \mathbf{E}_{i:j} | \mathbf{B}_{i:j}, \theta_t}(\ell_\theta(\mathbf{B}_{i:j}, M_{i:j}, \mathbf{E}_{i:j})).$$

Unfortunately, the conditional distribution $M_{i:j}, \mathbf{E}_{i:j} \mid \mathbf{B}_{i:j}, \theta_t$ is intractable because the normalization constant requires integrating over the high-dimensional vectors $M_{i:j}$ and $\mathbf{E}_{i:j}$. Since the main goal of our algorithm is to decrease overall computation time, we instead assume that $M_{i:j}$ is known, either from a public

database of mutations [16], or from genotypes estimated on-the-fly using a fast and simple variant calling algorithm. We assume that the sampled genome(s) harbor mutations only at sites contained in this database. This enables us to quickly estimate the sample genotype in a particular region, at the expense of erroneously classifying uncalled sites as sequencer errors. Since these sites are generally segregating at low frequency in the population, or were the result of genotyping error, the overall effect of this assumption on our likelihood calculation should be small. Moreover, by training our algorithm to flag regions with an elevated sequencer error rate, we retain the ability to detect these novel variants downstream.

3.2 Augmented Likelihood

The model described above aims to capture the essential data generating mechanism of an NGS experiment. In practice, we found that augmenting the likelihood with additional terms designed to capture features of coverage, mapping quality, and related statistics improved the accuracy of our algorithm with minimal performance impact. In particular, we assume that, in a region of constant genomic complexity:

1. The mapping quality (MAPQ) distribution of short reads is Bernoulli: with probability $\tau \in [0, 1]$, a read has MAPQ $= 0$; otherwise the read MAPQ > 0. Here we bin MAPQ into two classes, zero and non-zero, since its distribution is unknown, and also because we found that the strongest signal was contained in reads which had zero mapping quality.
2. With probability $\eta \in [0, 1]$, each base pair is inserted or deleted in the sample genome; otherwise, with probability $1-\eta$ the base is subject to the standard mutational and sequencer error processes described above.

Modern aligners [7, 17] generate MAPQ scores during read mapping procedure, and also call small indels where doing so improves concordance. Hence τ and η can be be estimated with high confidence from the data. Indels which are not detected by the aligner will generate aberrations in the coverage and mismatch signals as described above.

Letting θ denote the vector of all parameters in our model, the augmented likelihood from positions i to j can be written as

$$\ell^{\text{aug}}_{\theta,\tau,\eta}(i,j) \equiv \ell_\theta(\mathbf{B}_{i:j}, M_{i:j}, \mathbf{E}_{i:j}) + \sum_{b=i}^{j} I_b \log\eta + Q_b \log\tau + (D_b - I_b)\log(1-\eta) + (D_b - Q_b)\log(1-\tau), \quad (5)$$

where I_b and Q_b count the number of inserted/deleted and MAPQ-0 bases at position b, and $\iota_{b,c}$ is the insert size of the c-th read aligned at position b. The augmented model no longer has a simple interpretation in the generative sense; and in fact we unrealistically assume that the random variables with which we

have augmented the likelihood function are mutually independent and identically distributed. On the other hand, this enables us to quickly estimate these parameters from data, and these parameter estimates in turn enable us to easily detect the signatures of mutational events which mark complex regions of the genome.

3.3 Change Point Detection

CAGE classifies genomic regions using parameter estimates obtained by maximizing (5). We assume that these parameters are piecewise constant within (unobserved) segments of the sample genome, and estimate the segments using a changepoint detection method. Let

$$(\hat{\theta}_{i:j}, \hat{\tau}, \hat{\eta}) = \arg\max_{\theta,\tau,\eta} \ell^{\text{aug}}_{\theta,\tau,\eta}(i,j)$$
$$C(i,j) = -\ell^{\text{aug}}_{\hat{\theta}_{i:j},\hat{\tau},\hat{\eta}}(i,j)$$

be defined using the likelihood function given above, and let

$$\boldsymbol{\tau} \triangleq (\tau_0 = 0, \tau_1, \tau_2, \ldots, \tau_{m+1} = s)$$

be a sequence of changepoints dividing the region $0, \ldots, s$. $C(i,j)$ is the negative log-likelihood of segment $i:j$ evaluated at the MLE, and hence

$$\sum_{i=1}^{m+1} [C(\tau_{i-1}+1, \tau_i) + \beta] \tag{6}$$

is a natural measure of fit for the segmentation $\boldsymbol{\tau}$. Here β is a regularization parameter which penalizes each additional changepoint. In practice, rather than considering all $O(10^9)$ loci in the human genome as potential changepoints, we restrict the τ_i in (6) to integer multiples of some window size w. We typically set $25 \leq w \leq 200$ when evaluating our algorithm. This speeds up the optimization, and also decreases the variance of the maximum likelihood parameter estimates for each segment.

Exact minimization of (6) over m and $\boldsymbol{\tau}$ can be achieved in quadratic time via a dynamic programming algorithm [18]. For likelihood-based changepoint detection, properties of the likelihood function certify that certain changepoint positions can never be optimal. Killick et al. [19] exploit this property to formulate a pruning algorithm which minimizes (6) in expected linear time. The pruning process enables both computational savings, as well as significant memory savings since the in-memory data structures can be repeatedly purged of all data prior to the earliest possible changepoint. The resulting algorithm consumes only a few gigabytes of memory, even when processing data sets which are tens of gigabytes in size. Thus, multiple chromosomes can be processed simultaneously on a typical workstation.

Our cost calculations take further advantage of a property of maximum likelihood-based cost functions for parametric families that enable us to avoid

calculating the full likelihood. From (2)–(4) and (5) it is seen that our cost function factors as $C(i,j) = g_{\hat{\theta}_{i:j},\hat{\tau},\hat{\eta}}\left(T\left(i,j\right)\right) + h\left(i,j\right)$ where $T(i,j)$ and $h(i,j)$ depend only on the data in region $i:j$ and h does not depend on the parameters. Since our model assumes the data are independent and identically distributed within a segment, we have

$$h\left(i,j\right) = \sum_{b=i}^{j} h_1(b)$$

where h_1 is a univariate function. Thus the optimization (6) can be decomposed as

$$\begin{aligned}
&\min_{\substack{m,\boldsymbol{\tau} \\ s=\tau_{m+1}}} \sum_{i=1}^{m+1} \left[g_{\hat{\theta},\hat{\tau},\hat{\eta}}\left(\tau_{i-1}+1,\tau_i\right) + h\left(\tau_{i-1}+1,\tau_i\right) + \beta \right] \\
&\qquad = \min_{\substack{m,\boldsymbol{\tau} \\ s=\tau_{m+1}}} \sum_{i=1}^{m+1} \left[g_{\hat{\theta},\hat{\tau},\hat{\eta}}\left(\tau_{i-1}+1,\tau_i\right) + \sum_{b=\tau_{i-1}+1}^{\tau_i} h_1\left(b\right) + \beta \right] \\
&\qquad = \sum_{b=1}^{n} h_1\left(b\right) + \min_{\substack{m,\boldsymbol{\tau} \\ s=\tau_{m+1}}} \sum_{i=1}^{m+1} \left[g_{\hat{\theta},\hat{\tau},\hat{\eta}}\left(\tau_{i-1}+1,\tau_i\right) + \beta \right].
\end{aligned}$$

We see that it is not necessary to evaluate h at all in order to carry out the optimization. In our setting this function involves a number of log-factorial terms which are relatively expensive to evaluate.

3.4 Identification of High-Complexity Regions

The changepoint detection algorithm described above determines which genomic regions have uniform genomic complexity. Next, we use this information to allocate additional computational resources to complex regions. In this paper we use a binary classification scheme in which a region is labeled as either high or low. For each changepoint region, we compute the features considered in CAGE's augmented likelihood defined in (5), and classify each region as high-complexity if any of these features are outliers, using hand-tuned thresholds for each feature.

3.5 Integrated Variant Calling Algorithm

The hybrid variant calling approach described in Section 1 relies on a fast algorithm to detect variation in low-complexity regions. Additionally, CAGE requires estimates of ground-truth locations of mutations and short indels in its likelihood calculation, as discussed in Section 3.1. We hypothesized that variation in low-complexity regions of the genome should be particularly easy to call, and implemented a simple, rule-based variant calling heuristic to be run alongside the core CAGE algorithm during the initial pass over the data.

We refer to this modified algorithm as CAGE++. It uses a count-based method to identify variants from a pileup, and relies on read depth, strand bias

and read quality scores to filter calls. This method has five tuning parameters: CAGe++ ignores all bases with quality scores less than α_1, and calls a variant at a particular pileup location if:

1. The pileup depth is at least α_2;
2. An alternative allele appears in at least α_3 percent of the reads, and no fewer than α_4 actual reads; and
3. The strand bias is less than α_5 percent.

As we show in Section 4, this heuristic is extremely fast while remaining competitive with more sophisticated algorithms in terms of accuracy.

4 Results

We compared our proposed algorithms to two baseline variant calling algorithms from GATK version 2.8. The first is a computationally cheap caller known as UnifiedGenotyper (`GATK-ug`). The second algorithm, HaplotypeCaller (`GATK-ht`), is more accurate but relies on computationally demanding local de novo assembly. We compare these two variant callers with two hybrid approaches. In one approach, we first segment the genome by complexity using CAGe, and then use `GATK-ug` and `GATK-ht` to process the low- and high-complexity regions, respectively. In the second approach, we use CAGe++ to both segment the genome by complexity and perform variant calling on the low-complexity regions, and then rely on `GATK-ht` to process the high-complexity regions. To measure the effectiveness of the changepoint detection component of CAGe and CAGe++, we also evaluate a simple alternative hybrid approach, called AllChange, in which we treat each window as a distinct region and rely solely on our rule-based classifier to determine genomic complexity.[4]

To perform the CAGe and AllChange hybrid approaches, we first ran `GATK-ug` on the full genome, and used these predictions as estimates of $M_{i:j}$, as discussed in Section 3.1. Since CAGe++ calls variants directly, the CAGe++ approach did not rely on `GATK-ug`. For all three hybrid approaches, after identifying regions of high-complexity via binary classification, we then executed `GATK-ht` on each region, expanding each region by 10% of its length to minimize errors at the boundaries. We then combined the resulting predictions from the low-complexity regions (either from `GATK-ug` or directly from CAGe++) with the predictions from `GATK-ht` on the high-complexity regions.

We performed all our experiments on an x86-64 architecture multicore machine with 12 2.4Ghz hyperthreaded cores and 284 GB of main memory. We tuned the changepoint parameters for CAGe and CAGe on a small hold-out set, setting the window size to $w = 100$, the regularization parameter to $\beta = 3.0$. We used the same window size for AllChange. For CAGe++, we set the variant calling parameters to $(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5) = (12, 10, 20, 3, 20)$.

[4] AllChange is similar to GATK's ReduceReads algorithm and is also what we obtain from CAGe with $\beta \equiv 0$.

4.1 Datasets and Evaluation

To evaluate the performance of these algorithms we used SMaSH [20], a recently developed suite of tools for benchmarking variant calling algorithms. Briefly, SMaSH is motivated by the lack of a gold-standard NGS benchmarking dataset which both a) mimics a realistic use-case (i.e. is not generated by simulating sequencer output) and b) includes comprehensive, orthogonal validation data (i.e. is not generating using sequencer output). The datasets comprised by SMaSH present different trade-offs between practical relevance and the quality and breadth of the validation data. We worked with the following SMaSH datasets:

- Venter Chromosome 20 and full genome: generated from Craig Venter's genome [21] and including noise-free validation data and synthetically generated short-reads (30× coverage, mean insert size of 400);
- Mouse Chromosome 19: derived from the mouse reference genome and including noisy validation data and overlapping short-reads generating from a GAIIx sequencer (60× coverage, mean insert size of −34);
- NA12878 Chromosome 20: based on a well studied human subject, including short-reads from a HiSeq2000 sequencer (50× coverage, mean insert size of 300). SMaSH's validation data for this dataset consists primarily of well-studied SNP locations, so we instead rely on a richer set of validated variants (SNPs and indels only) provided by Illumina's Platinum Genomes project [22]. We nonetheless leverage the SMaSH evaluation framework to compute accuracy. Since this validation set is only a sample of the full set of variants, we do not report precision results.

4.2 Accuracy

Table 1 summarizes the accuracy of `GATK-ug`, AllChange, CAGe/CAGe++ and `GATK-ht`. Precision and recall are calculated with respect to the validated variants in the SMaSH datasets, except in the case of NA12878 as described above. As expected, `GATK-ht` generally outperforms `GATK-ug`; the difference is particularly pronounced for indels. Second, the CAGe approach in which `GATK-ht` is applied to the high-complexity regions and `GATK-ug` is applied to the remainder yields comparable accuracy to `GATK-ht`, with the CAGe approach being slightly better on Mouse, comparable on Venter, and slightly worse on NA12878. Third, AllChange accuracy is comparable to that of CAGe, indicating that the features we consider in our likelihood model are indeed predictive of genome complexity.

As shown in Table 2, the AllChange high-complexity regions are larger than the corresponding CAGe and CAGe++ regions, and by a large margin for the two human datasets, thus highlighting the effectiveness of the changepoint detection algorithm. The table also shows that a large fraction of the structural variants are concentrated in these high-complexity regions. Moreover, when we investigated the handful of remaining structural variants which fell into low-complexity regions, we found that they were difficult to discern even by visually

Table 1: Precision/Recall of various variant calling algorithms.

Variant	Dataset	GATK-ug	AllChange	CAGe	CAGe++	GATK-ht
SNPs	Venter	93.6 / 98.7	98.8 / 98.2	98.6 / 98.2	98.9 / 96.5	98.9 / 98.1
	Mouse	93.3 / 96.4	98.2 / 95.5	98.1 / 95.6	98.6 / 95.4	98.5 / 95.4
	NA12878	- / 99.7	- / 99.6	- / 99.6	- / 99.6	- / 99.6
Indel-Ins	Venter	94.3 / 81.3	93.1 / 92.1	93.1 / 91.9	93.1 / 91.8	93.1 / 92.0
	Mouse	94.2 / 75.1	89.7 / 89.5	89.8 / 89.4	90.0 / 88.3	89.2 / 89.7
	NA12878	- / 62.2	- / 93.0	- / 92.7	- / 92.8	- / 93.4
Indel-Del	Venter	95.0 / 87.2	95.0 / 93.4	95.1 / 93.3	95.0 / 93.6	95.1 / 93.9
	Mouse	92.1 / 90.7	80.1 / 94.4	81.0 / 94.4	85.6 / 94.2	77.7 / 94.6
	NA12878	- / 64.3	- / 94.3	- / 94.3	- / 94.3	- / 94.6

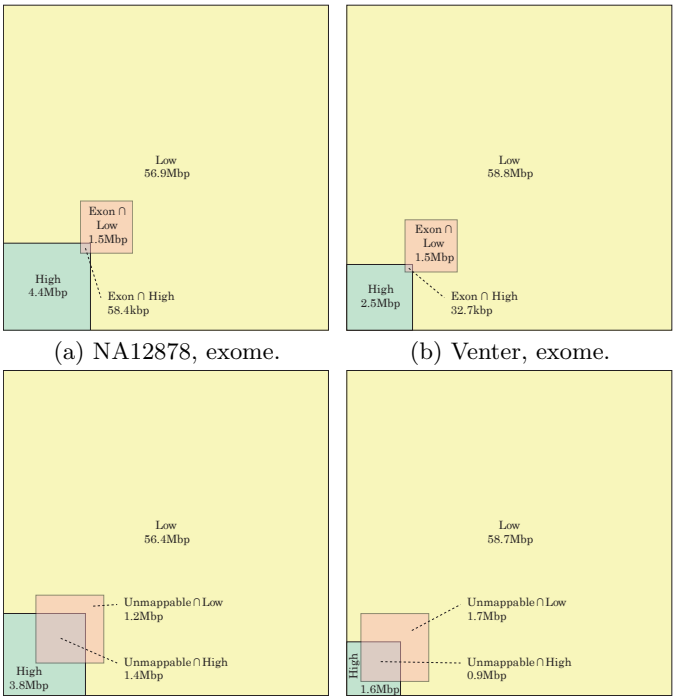
Table 2: Segregation of variants in high-complexity regions for AllChange and CAGe.

Algorithm	Dataset	Size of high	% SNPs	% Indels	% SVs
AllChange	Venter	13.1%	28.2%	58.3%	98.4%
	Mouse	13.2%	72.5%	79.9%	98.7%
	NA12878	13.5%	-	-	-
CAGe	Venter	4.0%	14.3%	53.4%	98.4%
	Mouse	11.3%	61.3%	77.1%	98.8%
	NA12878	7.1%	-	-	-
CAGe++	Venter	6.3%	32.1%	99.6%	97.2%
	Mouse	9.7%	55.8%	97.3%	98.2%
	NA12878	8.3%	-	-	-

inspecting the raw data. We further note that the basic variant caller in CAGe++ leads to a much higher fraction of indels being placed in high-complexity regions.

4.3 Computational Performance

We evaluated the runtime of CAGe on the full Venter genome, executing it on a single Amazon EC2 cc2.8xlarge instance with 59 GB of main memory and 32 cores. We divided the genome into roughly equal sized subproblems, and CAGe completed in 13 minutes when executing all subproblems in parallel, with peak memory usage of less than 16 GB. Next, we evaluated the performance of CAGe++. Since CAGe is heavily I/O bound, the additional computation required by the variant caller component of CAGe++ has a small impact on overall execution time, increasing runtime relative to CAGe by approximately 50%. Finally, we evaluated the speedup obtained by executing GATK-ht only on CAGe's high-complexity regions, but we observed modest speedups. Indeed, on Venter, where CAGe's high-complexity regions comprise a mere 4% of chromosome 20, we observed a 1.4× speedup. As a baseline comparison, we also executed GATK-ht on randomly selected contiguous regions each consisting of 4% of chromosome 20, and observed an average speedup of 2.8×. The sublinear scaling of GATK-ht suggests that it may not be well suited for a hybrid variant calling approach.



(a) NA12878, exome. (b) Venter, exome.

(c) NA12878, GEM non-unique. (d) Venter, GEM non-unique.

Fig. 1: Overlap between CAGE regions and known genome annotations. (a,b) Overlap with the exome. (c,d) Overlap with GEM non-unique regions.

4.4 Properties of CAGe Regions

To further validate our approach, we examined how low- and high-complexity regions produced by CAGE interact with various genome annotation tracks, as well as with each other when processing multiple samples at once. Figure 1 (a,b) depicts the overlap between these regions and portions of the sampled genome that are annotated as exons according to the Illumina TruSeq exon capture kit. We found that low-complexity regions are comparatively enriched for exons: 96.3% (97.9%) of the exome falls in low-complexity for NA12878 (Venter). This is expected since exons are under stronger purifying selection than noncoding regions of the genome and hence harbor less variation [23].

We also explored the relationship between genome mappability [24] and CAGE classification. Mappability estimates the uniqueness of each k-mer of the reference genome. Repetitive and duplicated regions have lower mappability scores, while k-mers that are unique among all reference k-mers have a mappability score of 1. In our experiments we set $k = 100$ to match the standard read length of NGS data. Figure 1 (c,d) compares the overlap between high-complexity regions, low-complexity regions, and segments of the genome that

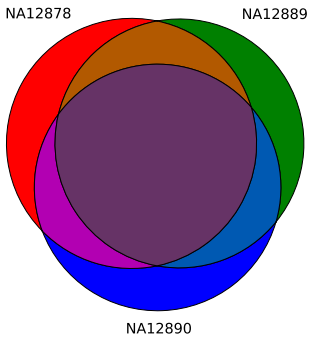


Fig. 2: Overlap of chromosome 20 high regions for three unrelated individuals.

have non-unique k-mers (mappability < 1). High-complexity regions are comparatively enriched for segments that are more difficult to map, with 53.3% (7.5× enrichment) of the non-unique locations residing in the NA12878 high-complexity regions and 34.1% (8.5× enrichment) in the Venter high-complexity regions. This enrichment is consistent with the fact that non-unique locations of the genome are prone to alignment errors that can result in high-complexity pileups around these locations.

Since variant calling is often performed with many samples in parallel, we next studied the concordance of high-complexity regions between individual samples. We computed high-complexity regions for samples NA12889, NA12890 and NA12878 using data released by the Platinum Genomes project. The individuals are members of the CEPH/UTAH 1463 pedigree but are unrelated (mother and paternal grandparents). We ran CAGe on chromosome 20 of each of these individuals, setting the thresholds of our rule-based classifiers to generate a high-complexity region of consistent size on each of the chromosomes (respectively, 8.1%, 8.2%, 8.3%). The Venn diagram in Figure 2 characterizes the overlap among these regions. The high-complexity regions are fairly consistent among the three individuals; the union of their high-complexity regions consists of 11.6% of the chromosome.

5 Discussion

These experiments illustrate that a hybrid approach has the potential to accurately detect regions of a sampled genome that harbor the majority of complex variation, as well as improve the computational performance of variant calling algorithms. It is possible to partition a genome into high- and low-complexity regions such that:

1. Low-complexity regions comprise a large majority of the genome; and
2. Fast, simple variant calling algorithms work as well as slower, more complex ones on these regions.

This strategy leads to a large increase in throughput compared with traditional variant calling pipelines.

There are several avenues for improving upon this work. Our experiments demonstrate the promise of, at least in the case of deeply sequenced samples, employing a trivial, rule-based variant calling algorithm to process a large fraction of the data with minimal impact on accuracy. More experiments are needed to confirm that this finding translates to larger samples and/or other types of sequencing experiments.

The experimental results presented above used a hand-trained classifier to segment the sampled genomes into high- and low-complexity regions. In order to employ our algorithm on a larger scale it is necessary to automatically train this classifier. Since it is usually straightforward via visual inspection to determine whether a region harbors a complex variant, one potential solution is to build a streamlined program to facilitate the rapid generation of training examples by human supervision. We have implemented a prototype of this software and found that a knowledgeable human subject is capable of generating on the order of 1,000 training examples per hour. More work is needed to integrate this supervised classifier into CAGe.

Another extension would be to the multi-class regime where regions are placed into one of several categories based on whether they are believed to harbor SNPs, indels, various types of structural variants or some combination thereof. The summary statistics generated in the maximum likelihood step of CAGe could be used to send segments to specialized variant calling algorithms designed to handle these respective categories.

Acknowledgments. This research is supported in part by a National Defense Science and Engineering Graduate Fellowship (AB), an NSF award No. 1122732 (AT), NSF award No. 0130526 (JT, AB, BY), an NIH National Research Service Award Trainee appointment on T32-HG00047 (JT), and an NSF CAREER Grant DBI-0846015 (JT, YSS).

References

[1] S.A. Tishkoff and K. K. Kidd. Implications of biogeography of human populations for 'race' and medicine. *Nature Genetics*, 36:S21–S27, 2004.

[2] A. J. Cox, M. J. Bauer, T. Jakobi, and G. Rosone. Large-scale compression of genomic sequence databases with the burrows-wheeler transform. *Bioinformatics*, 28(11):1415–1419, 2012.

[3] Hsi-Yang F. M., R. Leinonen, G. Cochrane, and E. Birney. Efficient storage of high throughput dna sequencing data using reference-based compression. *Genome Research*, 21(5):734–740, 2011.

[4] D. C. Jones, W. L. Ruzzo, X. Peng, and M. G. Katze. Compression of next-generation sequencing reads aided by highly efficient de novo assembly. *Nucleic Acids Research*, 40(22):e171, 2012.

[5] H. Li et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[6] M. A. DePristo et al. A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature Genetics*, 43(5):491–498, 2011.

[7] Zaharia M., Bolosky W., Curtis K., Fox A., Patterson P., Shenker S., Stoica I., Karp R., and Sittler T. Faster and more accurate sequence alignment with SNAP, 2011. URL `arXiv:1111.5572[cs.DS]`.

[8] N. Popitsch and A. von Haeseler. Ngc: lossless and lossy compression of aligned high-throughput sequencing data. *Nucleic Acids Research*, 41(1):e27, 2013.

[9] J. J. Shen and N. R. Zhang. Change-point model on nonhomogeneous Poisson processes with application in copy number profiling by next-generation DNA sequencing. *The Annals of Applied Statistics*, 40(6):476–496, 2012.

[10] Y. Shen, Y. Gu, and I. Pe'er. A Hidden Markov Model for Copy Number Variant prediction from whole genome resequencing data. *BMC bioinformatics*, 12(Suppl 6):S4, 2011.

[11] K. Wang et al. PennCNV: An integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data. *Genome research*, 17(11):1665–1674, 2007.

[12] E. S. Lander and M. S. Waterman. Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*, 2(3):231–239, 1988.

[13] S. N. Evans, V. Hower, and L. Pachter. Coverage statistics for sequence census methods. *BMC Bioinformatics*, 11:430, 2010.

[14] V Hower, R Starfield, A Roberts, and L Pachter. Quantifying uniformity of mapped reads. *Bioinformatics*, 28(20):2680–2682, 2012.

[15] P. Medvedev, M. Stanciu, and M. Brudno. Computational methods for discovering structural variation with next-generation sequencing. *Nature Methods*, 6:S13–S20, 2009.

[16] S. T. Sherry et al. dbsnp: the ncbi database of genetic variation. *Nucleic Acids Research*, 29(1):308–311, 2001.

[17] Li H. and Durbin R. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, 25:1754–1760, 2009.

[18] B. Jackson et al. An algorithm for optimal partitioning of data on an interval. *Signal Processing Letters, IEEE*, 12:105–108, 2005.

[19] R. Killick, P. Fearnhead, and I. A. Eckley. Optimal Detection of Changepoints With a Linear Computational Cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[20] A Talwalkar et al. Smash: A benchmarking toolkit for variant calling, 2013. URL `arXiv:1310.8420[q-bio.GN]`.

[21] S. Levy et al. The diploid genome sequence of an individual human. *PLoS Biology*, 5(10):e254, 2007.

[22] Illumina Corporation. Platinum genomes project, 2013. URL `http://www.platinumgenomes.org`.

[23] Z. Zhao, Y. Fu, D. Hewett-Emmett, and E. Boerwinkle. Investigating single nucleotide polymorphism (snp) density in the human genome and its implications for molecular evolution. *Gene*, 312(0):207–213, 2003.

[24] T. Derrien et al. Fast computation and applications of genome mappability. *PLoS ONE*, 7(1):e30377, 2012.