1 GENOME ANALYSIS

Efficient Detection of Highly Mutated Regions with Mutations Overburdening Annotations Tool (MOAT)

Lucas Lochovsky[1,2], Jing Zhang[1,2] and Mark Gerstein[1,2,3*]

[1]Program in Computational Biology and Bioinformatics, Yale University, New Haven, Connecticut 06520, USA

[2]Department of Molecular Biophysics and Biochemistry, Yale University, New Haven, Connecticut 06520, USA

[3]Department of Computer Science, Yale University, New Haven, Connecticut 06520, USA

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

A major challenge of working with somatic cancer variants is the need to identify those variants responsible for disruptions that drive cancer progression out of the thousands that arise due to background mutation processes. One approach is to scan the genome for elements with a high frequency of intersecting somatic variants, or elevated mutation burden. The detection of significant mutation burden is also useful in germline variant analysis, as rare variant burdens may indicate increased risk of genetic disease. Here, we introduce the Mutations Overburdening Annotations Tool (MOAT), a new computational tool designed to identify regions with a high mutation burden relative to the surrounding genome in a non-parametric way. MOAT is useful for prioritizing annotations to study in downstream analyses, as high mutation burden annotations are most likely to be driver elements in genetic disease. We release an implementation that offers users two forms of mutation burden analysis through empirical permutations, as well as serial and parallel versions of each form. We also demonstrate MOAT's capability for finding known noncoding drivers in cancer variant data.

Availability: MOAT is available at moat.gersteinlab.org

2 INTRODUCTION

High throughput sequencing of genetic disease cohorts has enabled the identification of the molecular causes of these illnesses. This data can be utilized to find the somatic single nucleotide variants (SNVs) in each patient. However, due to the relatively high number of neutral variants in such patients' genomes, it is not immediately apparent which variants are directly connected to the disease phenotype. A common strategy for addressing this issue is to look for genomic elements with a high accumulation of variants. By modeling the factors that influence the stochastic mutation rate, the elements that are more mutated than expected under the background model can be determined.

It is well known that the background mutation rate is highly heterogeneous across the whole genome due to numerous external features, such as replication timing and chromatin structure (Lochovsky, et al., 2015). This effect may change in a dynamic way across the genome and is usually difficult to model. Hence, our Mutations Overburdening Annotations Tool (MOAT) relies on no assumption except that background mutation rate changes slowly across the genome and approximately remains the same within a local context. Therefore, such non-parametric scheme provides robust mutation burden significance results of any element through permutations.

MOAT offers two methods for determining elevated mutation burdens, the annotation-centric algorithm (MOAT-a) and the variant-centric algorithm (MOAT-v), both of which involve a comparison of each annotation's mutation accumulation to that of the surrounding genome. In the following sections, we will describe the implementation of MOAT for parallel computer systems, which enables highly efficient data size scalability. This scalability is important for guaranteeing a reasonable running time given the high computational intensity of the permutation step. We also evaluate MOAT's ability to recall known noncoding cancer drivers on a collection of several hundred cancer whole genomes' variant data.

3 METHODS

MOAT takes two input files: the annotation file (*afile*) and the variant file (*vfile*).

3.1 MOAT-a: Annotation-centric Permutation

The parallel version of MOAT's annotation-centric permutation algorithm, MOAT-a, is a C++ program that uses NVIDIA's CUDA language (Nickolls, et al., 2008) to instantiate parallel graphics processing unit (GPU) threads, and divides the computational workload across these threads. MOAT-a's steps are illustrated in Fig. 1. MOAT-a iterates through the annotations, computing the intersecting variant count per annotation. It then defines an extended region with a user-defined distance centered at the current input annotation, and randomly moves the annotation within this extended region. MOAT-a will find the variant counts from the *vfile* that intersect each of the random bins, which are compared to the input annotation's variant count. The input annotation's p-value is defined as the fraction of bins with a variant count equal to or greater than the input annotation's variant count.

*To whom correspondence should be addressed.

Jing Zhang 5/4/2016 11:14 AM
Deleted: Our

Jing Zhang 5/4/2016 11:20 AM
Deleted: is designed to automatically overcome such confounding effect in a non-parametric way and

Jing Zhang 5/4/2016 11:21 AM
Deleted: compute the

Jing Zhang 5/4/2016 11:22 AM
Deleted: of the

Jing Zhang 5/4/2016 11:22 AM
Deleted: mutation burden

Lucas Lochovsky 3/22/2016 12:05 PM
Deleted: High throughput sequencing of genomes for patients with genetic diseases has opened up the possibility of finding the precise causes of these diseases, paving the way for more effective drug development for these illnesses in the future. However, the analysis of this data has not kept pace with the data's production rate. Fast and efficient analysis is necess ... [1]

Jing Zhang 5/4/2016 10:32 AM
Deleted: functional annotations

Jing Zhang 5/4/2016 11:25 AM
Deleted: In essence, MOAT flags an ... [3]

Lucas Lochovsky 3/22/2016 2:19 PM
Deleted: MOAT offers users two typ ... [4]

Lucas Lochovsky 3/22/2016 12:11 PM
Deleted: Such

Lucas Lochovsky 3/22/2016 12:14 PM
Deleted: may be potential

Lucas Lochovsky 3/22/2016 12:12 PM
Deleted:

Lucas Lochovsky 3/23/2016 10:02 AM
Formatted: Tabs:Not at 2.44"

Lucas Lochovsky 5/4/2016 2:34 PM
Deleted: defined

Jing Zhang 5/4/2016 10:34 AM
Deleted: One means of detecting devi ... [2]

Jing Zhang 5/4/2016 11:15 AM
Deleted: the confounding effect from

Jing Zhang 5/4/2016 11:17 AM
Deleted: genomic

Jing Zhang 5/4/2016 11:15 AM
Deleted: .

Lucas Lochovsky 5/4/2016 3:07 PM
Deleted: [add larva as ref]

Jing Zhang 5/4/2016 11:18 AM
Formatted: Highlight

A typical MOAT-a run involves an annotation count on the order of ~10^5 at a minimum, each of which are permuted 1000 times. Hence, the overall computation involves millions—or even billions—of permutation and intersection calculations, which take an inordinate amount of time to complete. However, this computation is very easily adaptable to parallel execution, so MOAT-a breaks up the overall computation into many separate, independent units of computation.

MOAT-a's speed can be further improved by taking advantage of the thousands of parallel stream processors available on modern graphics processing units (GPUs). GPUs are designed for efficiently processing 3D graphics calculations, which largely consist of numerous matrix operations with relatively low memory usage. Due to the limited amount of video RAM (VRAM) available on GPUs, MOAT-a's GPU version was planned to use GPU acceleration only for the most compute intense step. This is the permutation step, where new annotation locations are determined in the local genome surrounding each annotation. The annotation coordinates are copied to VRAM, and one permutation per annotation is calculated in parallel. The coordinates for the permuted annotations are copied back to main memory for the fast intersection step. This permutation/intersection loop is performed *n* times (the user-defined total number of permutations), after which p-values are calculated using the observed variant counts and the permuted variant counts (Fig. 1).

3.2 MOAT-v: Variant-centric Permutation

MOAT-v's variant-centric permutation algorithm creates permuted datasets by assigning new coordinates to each variant within a local genome region to account for the covariate effects from known genomic features (Fig. 2a). These covariates influence the whole genome background mutation rate, hence they must be taken into consideration when assessing an annotation's mutation burden relative to background mutation. For small enough regions, we assume these covariates are essentially constant, and we can perform variant permutations under the assumption of uniformity, with one key constraint. MOAT-v must preserve the trinucleotide context of the original variant when choosing a new variant location. In other words, the new variant must have the same nucleotide identity as the original variant, and the two neighbors of the new variant must also have the same nucleotide identity as the original variant's neighbors. For example, if MOAT-v is given an input variant that has a reference base G, and is surrounded by a T and a C (i.e. the variant's trinucleotide context is TGC), then MOAT-v gathers up every position in the same bin where TGC occurs in the reference, and selects one of these with uniform probability (Fig. 2b). This constraint reflects the differential mutation probabilities of different trinucleotides (i.e. due to biochemical differences, some trinucleotides are more likely to be mutated than others), and ensures that the permuted variant set does not change the expected distribution of mutated trinucleotides.

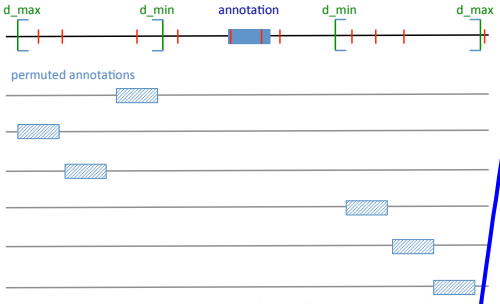

Figure 1 For each input annotation, MOAT-a finds the number of intersecting *vfile* variants (red). The annotation's coordinates are then shuffled to a new location within the local genome context bounded by user-defined parameters *d_min* and *d_max*, producing *n* permutations (blue). Each permutation's intersecting variant count is computed

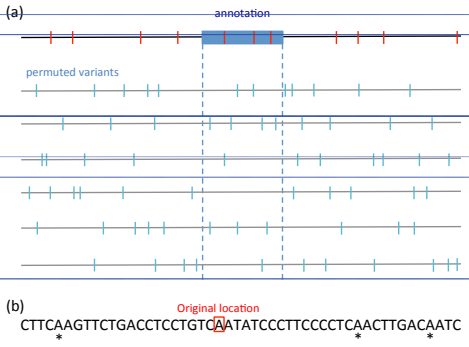

Figure 2 (a) In MOAT-v, the variant locations are permuted within the local genome context. The whole genome is divided into bins of a user-defined size, and variants are moved to new coordinates within the same bin, preserving the local mutation context. As with MOAT-a, *n* permutations are produced. (b) To reflect the influence of nucleotide identity on mutation likelihood, MOAT-v ensures that variants are moved to locations with the same trinucleotide context.

MOAT-v takes a *vfile* and an *afile* as inputs, and generates a permuted dataset by subdividing the genome into bins of a user-defined size (excluding mappability blacklist regions), and assigning each bin's variants new positions within the same bin, preserving trinucleotide context in the process. This process continues until *n* permutations have been generated. At this point, MOAT-v will calculate *n* intersecting permuted variant counts for each of the input annotations. A p-value for each annotation is determined based on the fraction of the *n* intersecting permuted variant counts that are equal to or greater than the intersecting variant count derived from the original *vfile* variants.

Initial prototypes of the parallel version of MOAT-v used the Nvidia CUDA framework, but the necessity of loading the reference genome sequence to preserve trinucleotide context in the permutation step resulted in prohibitive memory requirements with respect to the available GPU video RAM. As a result, MOAT-v was instead written to parallelize its workflow across multi-core CPUs using the OpenMPI framework (Gabriel, et al., 2004). Under this arrangement, a single CPU core is designated to run the "master" process, and is responsible for dividing up the overall work and distributing it to the "worker" processes, which run on the remaining cores.

4 RESULTS

4.1 MOAT-a

Jing Zhang 5/4/2016 11:26 AM
Deleted:

Jing Zhang 5/4/2016 11:27 AM
Formatted ... [6]

Jing Zhang 5/4/2016 11:26 AM
Formatted ... [5]

Jing Zhang 5/4/2016 11:28 AM
Deleted: —in other words, it is "emb ... [7]

Jing Zhang 5/4/2016 11:27 AM
Formatted ... [8]

Jing Zhang 5/4/2016 11:29 AM
Deleted: It is easy to

Jing Zhang 5/4/2016 11:27 AM
Formatted ... [9]

Jing Zhang 5/4/2016 11:28 AM
Deleted: : each annotation's calculat ... [10]

Jing Zhang 5/4/2016 11:26 AM
Deleted:

Jing Zhang 5/4/2016 11:27 AM
Formatted ... [11]

Jing Zhang 5/4/2016 11:30 AM
Formatted ... [12]

Jing Zhang 5/4/2016 11:30 AM
Deleted: . ... any computational ... [13]

Jing Zhang 5/4/2016 11:27 AM
Formatted ... [14]

Lucas Lochovsky 3/23/2016 12:06 PM
Deleted: ... [15]

Lucas Lochovsky 3/23/2016 1:33 PM
Deleted: .

Lucas Lochovsky 3/23/2016 1:35 PM
Deleted: These regions are fixed-wi ... [16]

Lucas Lochovsky 3/23/2016 1:37 PM
Deleted: As with MOAT-a, …OAT- ... [17]

Lucas Lochovsky 3/23/2016 1:37 PM
Formatted ... [18]

Lucas Lochovsky 3/23/2016 1:37 PM
Deleted: *annotation*

Lucas Lochovsky 3/23/2016 1:38 PM
Formatted ... [19]

Lucas Lochovsky 3/23/2016 1:36 PM
Deleted: *s*…as inputs (Fig. 2a)…. M ... [20]

Lucas Lochovsky 3/23/2016 10:00 AM
Deleted: <sp>

Jing Zhang 5/4/2016 11:34 AM
Deleted: There are several ways to c ... [21]

Lucas Lochovsky 3/30/2016 1:59 PM
Formatted ... [22]

Table 1. Speed benchmark of MOAT-a (CPU and GPU versions) with respect to the number of input annotations. Each time trial involved using MOAT-a to generate 1000 permuted variant datasets. For large datasets, the GPU version substantially outperforms the CPU version.

Annotation set	Number of annotations	CPU version running time	GPU version running time	Fold speedup of GPU version
DRM	~14,000	1hr23min	1hr22min	1.01x
TSS	~130,000	1hr55min	1hr26min	1.34x
DHS	~3,000,000	13hr46min	2hr12min	6.26x

We demonstrate the magnitude of the CUDA speedup by evaluating the running time of MOAT-a on datasets of various sizes, using both the CPU and GPU versions to calculate the output. We took a dataset of pan-cancer whole genome variant calls that includes 507 cancer genomes of various types from (Alexandrov, et al., 2013), and 100 stomach cancer genomes from (Wang, et al., 2014), totaling ~8 million variants. We used 3 different annotation sets for our evaluation, representing 3 different input sizes to demonstrate MOAT-a's scalability. These include the Distal Regulatory Module (DRM) annotations from (Yip, et al., 2012), transcription start site (TSS) annotations derived by taking the 100bp regions upstream of each GENCODE gene start (Harrow, et al., 2012), and the Dnase I hypersensitive (DHS) sites from the ENCODE project (Thurman, et al., 2012). These annotation sets represent 3 different orders of magnitude in size: the DRM set spans ~14,000 annotations, the TSS set spans ~130,000 annotations, and the DHS set spans ~3 million annotations. We tested MOAT-a's running time on these 3 annotation sets with the number of random bins n = 1000, the results of which are shown in Table 1. It is clear that when scaling up to very large datasets, the CPU version's runtime increases considerably, while the GPU version's runtime rises very gradually. MOAT-a's running time is not affected by the number of variants (data not shown).

Due to the relative lack of verified noncoding regulatory elements associated with cancer, it is difficult to assess the accuracy of MOAT's predictions. Nevertheless, we demonstrate MOAT's usefulness for finding elevated mutation burdens in genomic elements by identifying highly mutated GENCODE transcription start sites, promoters, and distal regulatory modules, using the aforementioned pancancer variant dataset. TERT, which has well-documented cancer-associated promoter mutations (Vinagre, et al., 2013), was found to have two TSSes with significant mutation burden (both had BH-corrected p-values of zero). Other well-known cancer-associated TSS sites, including TP53 , LMO3, and AGAP5, also had significant mutation burdens (all had BH-corrected p-values of zero). After applying Bcnjamini-Hochberg (BH) false discovery rate correction (Benjamini and Hochberg, 1995) to all p-values, there were 5037 promoters, 1148 TSSes, and 305 DRMs with significant mutation burdens. These may be used as a shortlist for investigating and validating individual variants' associations with cancer.

4.2 MOAT-v

Using the same set of cancer variants used in the MOAT-a tests, parallel MOAT-v's running time was evaluated across multiple CPU configurations to demonstrate the performance gains of the OpenMPI implementation. MOAT-v in OpenMPI is set up to run one master process on one of the available CPU cores, and use the rest for worker processes. Hence, the program must be run with 3 cores to get two cores to process the work simultaneously, 4 cores to get three cores to process the work simultaneously, etc. Table 2 represents the running time improvement relative to the number of workers added. This improvement scales close to linear with the number of workers, indicating that the load balancing between each CPU core is very evenly divided, enabling significant time savings when MOAT-v is run in parallel.

Table 2. Speed benchmark of MOAT-v with respect to the number of CPU cores assigned worker processes. Each time trial involved using MOAT-v to generate one permuted variant dataset using ~8 million input variants, and 1,000,000-bp bins.

# of worker CPU cores	Running time	Fold speedup
1	3hr44min	1.00x
2	1hr54min	1.97x
4	1hr4min	3.50x
8	40min	5.60x

MOAT-v was used on the same variant and annotation sets used to demonstrate MOAT-a's usefulness for finding elevated cancer mutation burdens. MOAT-v produced comparable results—the same known cancer-associated TSSes flagged as significant in MOAT-a were also flagged in MOAT-v. After applying BH correction to all p-values, there were 1394 promoters, 451 TSSes, and 109 DRMs with significant mutation burdens. Hence, MOAT-v appears to be the more conservative algorithm.

5 DISCUSSION

Identification of genomic elements with a high mutation burden is useful for narrowing down the exact site of functional disruption. We introduce Mutations Overburdening Annotations Tool (MOAT), a new software tool to facilitate such analyses. We demonstrate the usefulness of this tool for flagging putative noncoding cancer drivers, and provide CUDA- and OpenMPI-accelerated versions that dramatically increase the speed of mutation burden analysis. Given the demand for efficient, meaningful analysis of genome sequence data that is now being produced at a very high rate, we consider MOAT's provision of such analysis for genetic disease drivers quite timely.

Funding: This work was supported by the National Institutes of Health [5U41HG007000-04].

REFERENCES

Alexandrov, L.B., *et al.* Signatures of mutational processes in human cancer. *Nature* 2013;500(7463):415-421.

Benjamini, Y. and Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*. 1995;57(1):289-300.

Gabriel, E., *et al.* Open MPI: Goals, concept, and design of a next generation MPI implementation. *Springer* 2004:97-104.

Lucas Lochovsky 5/4/2016 2:42 PM
Deleted: .

Jing Zhang 5/4/2016 11:36 AM
Deleted: Finding the genetic basis of disease enables the development of highly targeted therapies that promise to be far more effective than previous therapies. The current wave of next generation sequencing of thousands of genomes has provided the data necessary to find the precise phenomena responsible for the functional disruption that gives rise to disease phenotypes.

Harrow, J., *et al.* GENCODE: the reference human genome annotation for The ENCODE Project. *Genome research* 2012;22(9):1760-1774.

Lochovsky, L., *et al.* LARVA: an integrative framework for large-scale analysis of recurrent variants in noncoding annotations. *Nucleic acids research* 2015;43(17):8123-8134.

Nickolls, J., *et al.* Scalable parallel programming with CUDA. *Queue* 2008;6(2):40-53.

Thurman, R.E., *et al.* The accessible chromatin landscape of the human genome. *Nature* 2012;489(7414):75-82.

Vinagre, J., *et al.* Frequency of TERT promoter mutations in human cancers. *Nature communications* 2013;4:2185.

Wang, K., *et al.* Whole-genome sequencing and comprehensive molecular profiling identify new driver mutations in gastric cancer. *Nature genetics* 2014;46(6):573-582.

Yip, K.Y., *et al.* Classification of human genomic regions based on experimentally determined binding sites of more than 100 transcription-related factors. *Genome biology* 2012;13(9):R48.