1 CATEGORY

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

Lucas Lochovsky[1,*], Jing Zhang[12] and Mark Gerstein[1,2,3]

[1]Department of XXXXXXX, Address XXXX etc.

[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

High throughput sequencing of genetic disease patient genomes 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, efficient analysis is necessary to meaningfully interpret this data and derive actionable results. Here, we introduce the Mutations Overburdening Annotations Tool (MOAT), a new computational tool designed to identify functional annotations with a high mutation burden relative to the surrounding genome. Such annotations may be potential 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 **[...]**

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.

One means of detecting deviation from the expected background mutation rate is to look for elements that have a high variant density compared to the immediately surrounding genome. It is well known that the background mutation rate varies widely across the whole genome, and this rate is essentially confounded by numerous genomic features. Our Mutations Overburdening Annotations Tool (MOAT) is designed to automatically overcome such confounding effect in a non-parametric way and compute the significance of the mutation burden of any element.

MOAT simulates the local background distribution of somatic mutations in the human genome by creating permutations of the input variant set. In other words, given the number of samples and variants in the input file, how would those variants be distributed under the assumption that they arose solely due to background mutation processes? To answer this question, MOAT calculates new positions for each variant in the input data, accounting for mutability factors in the local genome context. These permuted variant sets enable comparison between the observed mutation burden and the expected mutation burden.

MOAT offers users two types of permutation algorithm to empirically assess the background mutation rate: MOAT-a (annotation centric) and MOAT-v (variant centric). 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.

3 METHODS

3.1 MOAT-a: Annotation-based Permutation

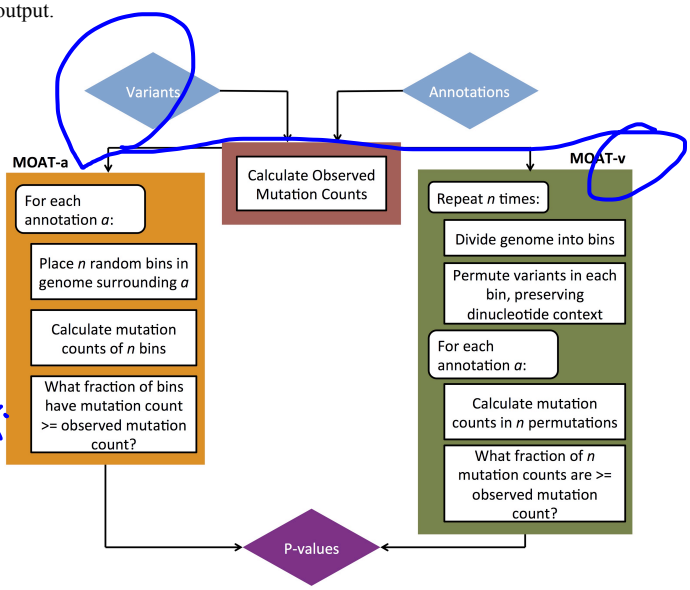
The parallel version of MOAT's annotation-based permutation algorithm, MOAT-a, is a C++ program that uses NVIDIA's CUDA language to instantiate parallel GPU threads, and divides the computational workload across these threads. MOAT-a's steps are illustrated in Fig. 1. MOAT-a takes a list of variant calls and a list of annotations as input, and iterates through the annotations, computing the intersecting variant count per annotation. MOAT-a then looks at the genome within a user-defined distance from the current input annotation, and randomly places windows in this region of length equal to the input annotation's length. MOAT-a will find the number of intersecting variants from the *vfile* that intersect each of the random bins. These variant counts are compared to the input annotation's intersecting variant count. The input annota-

*To whom correspondence should be addressed.

tion's p-value is defined as the percent of bins with a variant count equal to or greater than the input annotation's variant count. Hence, the p-value represents the degree to which the input annotation's mutation burden is elevated relative to the local background.

MOAT-a's operations are well suited for massively parallel computing. Therefore, we adapted MOAT-a into a CUDA program, which enables the acceleration of computation by utilizing

Fig. 1. Schematic overview of MOAT. The input data consists of annotations where the user wants to find significant mutation burdens as determined by the variant set. Intersecting the variants with the annotations produces the observed mutation counts. MOAT then follows one of two possible command paths to generate a background mutation distribution to evaluate the significance of each mutation burden. Each annotation is given a p-value in the output.



the thousands of stream processors found on graphics cards that are ordinarily purposed for the rapid rendering of 3D graphics. For our purposes, CUDA copies the variant and annotation data to an NVIDIA Geforce GPU's video memory, and performs thousands of intersection calculations in parallel.

3.2 MOAT-v: Variant-based Permutation

MOAT-v's variant-based permutation algorithm creates permuted datasets by assigning new coordinates to each input variant within the local genome region. These regions are fixed-width bins of a user-defined length, with the exception of mappability blacklist regions that include ENCODE consensus excludable regions, as well as centromeres and telomeres.

As with MOAT-a, MOAT-v takes variants and annotations as inputs (Fig. 1). MOAT-v will generate a permuted dataset by subdividing the genome into bins of a user-defined size, and assigning each bin's variants new positions within the same bin. These new positions are chosen such that the dinucleotide context of the original variant is preserved. For example, if MOAT-v is given an input variant that has a reference base G, and is adjacent to a C, then MOAT-v gathers up every position in the same bin where GC occurs in the reference, and selects one of these with uniform probability. The selected position is the input variant's coordinates in the permuted dataset.

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 dinucleotide 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. Under this arrangement, the work of generating a single permutation is split by chromosome, and each chromosome is assigned one of the available CPU cores. Since each chromosome's reference sequence is held in a separate FASTA file, each core will load a separate file, ensuring no resource contention. When one core finishes a chromosome, it is assigned the next chromosome that has not yet been assigned. This continues until each chromosome has been processed, after which the permuted variants are gathered and work begins on the next permutation, or, if all the permutations are complete, p-values are calculated.

4 RESULTS

4.1 MOAT-a

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 vastly 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	7min	4min	1.75x
TSS	~200,000	49min	5min	9.80x
DHS	~3,000,000	11hr6min	7min	95.14x

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 ~7 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 ~200,000 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. The performance of the CPU and GPU versions are comparable, but it is clear that for 3 million annotations, the CPU version's runtime increases considerably, while the GPU version never exceeds 10 minutes. 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 (p-values: 0.01 and 0.03). After applying Bcnjamini-Hochberg false discovery rate correction (Benjamini and Hochberg, 1995) to the p-values, there were 201 promoters, 258 TSSes, and 35 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 ~7 million input variants, and 1,000,000-bp bins.

# of worker CPU cores	Running time	Fold speedup
1	14hr54min	1.00x
2	7hr56min	1.88x
4	4hr31min	3.30x
8	2hr39min	5.62x

5 DISCUSSION

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. 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 a GPGPU-accelerated version that dramatically increases the speed of its mutation burden analysis. Given the demand for efficient, meaningful analysis of genome sequence data that is now being produced at very high rate, we consider MOAT's provision of such analysis for genetic disease drivers quite timely.

ACKNOWLEDGEMENTS

The quick brown fox jumps over the lazy dog. The quick brown

Funding: The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

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.

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

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.