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

Identifying genomic regions with a higher-than-expected mutation burden (i.e. overburdened regions) has a number of useful applications. In the context of somatic cancer variants, overburdened regions may be highly associated with cancer progression. In the germline, accumulation of rare variants could be an indicator of positive selection. Mutation burdens could also be applied in the context of *de novo* mutations to identify precursor variants for genetic disease. Here, we release a new GPU-based computational tool, called MOAT (the Mutations Overburdening Annotations Tool), to perform mutation burden analysis with great speed. MOAT makes no assumptions about the mutation process, except that the background mutation rate (BMR) changes smoothly with other genomic features. This nonparametric scheme randomly permutes the variants (or target regions) on a relatively large scale where the BMR is assumed to be constant to provide robust burden analysis in various scenarios. Furthermore, it also incorporates a somatic variant simulator called MOATsim, which randomly permutes the input variants with effective covariate control. MOAT also offers the option to evaluate the significance of annotations' whole genome signal scores, as implied by their intersecting variants, with the same permutation algorithms used for burden analysis. In conclusion, MOAT will be useful for a broad range of analyses that would benefit from a methods evaluation on variant permutation data.

Availability: MOAT is available at moat.gersteinlab.org

2 INTRODUCTION

A common analysis strategy in high throughput sequencing is to look for genomic elements with a high accumulation of variants across a large cohort of patients. However, it is well known that the background mutation rate (BMR) is highly heterogeneous across the whole genome due to numerous external features. For example, replication timing and chromatin structure usually affects the BMR at a scale of up to one megabase (Lawrence, et al., 2013). Such effects may change in a dynamic way across the genome, and are usually difficult to model. Hence, our Mutations Overburdening Annotations Tool (MOAT) relies on no assumption except that the BMR changes slowly across the genome and approximately remains the same within a local context. Therefore, this nonparametric scheme provides robust mutation burden significance results for 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. We also offer MOATsim, a variant distribution simulator based on MOAT-v's source code, which reflects the distribution of whole genome covariates that influence the BMR. MOAT can also be given a whole genome signal track, which will be used to compute the variants' signal scores, and produce aggregate annotation scores. The scores produced on the observed data will be compared to the scores produced on the permutation data to gauge the signal significance of the observed data. In the following sections, we will describe the implementation of MOAT for parallel computer systems, which enables highly efficient data size scalability. We also evaluate MOAT's ability to recall known noncoding cancer drivers on a collection of several hundred cancer whole genomes' variant data, on the basis of both mutation burden and functional impact.

3 METHODS

A number of covariates jointly affect the BMR in a complicated and dynamic manner, making variant burden analysis very challenging (Lawrence, et al., 2013). However, the length of the target region usually varies from hundreds of bases (such as transcription start sites) to thousands of bases (such as enhancers), while external features, such as replication timing, work at up to megabase resolution. Therefore, MOAT circumvents the necessity for such parametric models by explicitly permuting the variants or annotations within a region that has essentially constant levels of all the covariates. It assumes constant covariate values over a fairly large interval—with a typical size of ~100kb—that is appreciably larger that the annotations we wish to assess for elevated mutation burdens. Hence, such analyses would be useful for annotations in the

*To whom correspondence should be addressed.

†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.

~1kb size range, such as transcription factor binding sites (TFBSes), but not suitable for significantly smaller annotations, like transcription factor (TF) motifs, or much larger ones like topologically associating domains (TADs).

One important issue with these permutation algorithms is that their running times do not scale well to noncoding annotation sets. A typical burden analysis usually consists of more than 10^5 annotations (such as all GENCODE transcription start sites), with 1000 permutations per region. It can easily expand to billions of permutation and intersection calculations, making the overall operation very compute intense. Our analyses indicate that an unoptimized run (i.e. running the software on a single Intel Xeon CPU core with no parallelism) that involves ~3 million annotations with 1000 permutations each would take about 14 hours to complete.

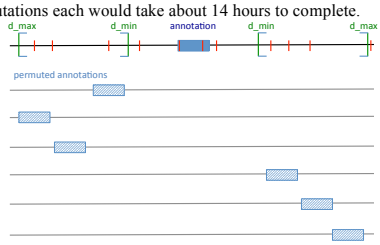

Figure 1 For each input annotation, MOAT-a finds the number of intersecting *vfile* variants (red). The annotation's coordinates are then shuffled to a new location within the local genome context bounded by user-defined parameters *d_min* and *d_max*, producing *n* permutations (blue). Each permutation's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

Here we have addressed this compute intensity by taking advantage of very large-scale parallelization available on modern GPU technology. The key realization behind this design choice is the fact that MOAT's most intense operations consist of a huge number of mathematical calculations with no data dependencies between each calculation, which plays perfectly into the strengths of general purpose GPU pipelines. In the following sections, we demonstrate MOAT's parallelization schemes for both the variant-based and annotation-based permutation algorithms.

3.1 MOAT-a: Annotation-centric Permutation

MOAT takes two input files: the annotation file (*afile*) and the variant file (*vfile*). 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) for general purpose graphics processing unit (GPGPU) programming. Over the past decade, computing hardware has become increasingly parallel in nature, meaning that multiple sets of instructions, or threads of execution, can be carried out concurrently on a series of interlinked CPUs. One of the most common types of problems that can be easily adapted to parallel computing is the rendering of computer graphics. This spurred the development of specialized GPU hardware with hundreds of processing units that are simplified to handle only instructions relevant to graphics processing. GPUs are therefore well suited for problems that can be divided into many data-independent units of parallel computation with a large proportion of mathematical compute operations. Here, we have successfully adapted MOAT-a to this computing paradigm.

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 a genomic block with user-defined boundaries centered on the current input annotation, and randomly moves the annotation within this block. 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. If MOAT is run with the whole genome signal score option, the variants' scores are computed and used to calculate the observed and permuted annotation scores by summing the scores of the intersecting variants.

In the GPU version, the input variants, annotations, and variant signal scores are copied to the video RAM (VRAM), making the data accessible to the GPU's threads. The annotations are divided among the threads, and each thread iterates through its subset of annotations, finding the observed variant count for each annotation, computing the permuted locations, and calculating the permuted variant counts and annotation signal scores. The permuted annotation locations are temporarily stored to calculate the variant counts and derive the annotation's p-value, after which the memory is freed for the next annotation. Only the p-values for mutation burden and whole genome signal are persistently stored, which controls MOAT-a's usage of VRAM—a far more limited resource compared to main memory. The p-values are then copied back to main memory for generating the program's output.

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). MOAT-v offers the option to 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. Hence, MOAT-v preserves the mutational signature of the input variants.

MOAT-v takes a *vfile* and an *afile* as inputs, and generates a permuted dataset by subdividing the genome into blocks of a user-defined size (excluding mappability blacklist regions), and assigning each block's variants new positions within the same block, 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 count derived from the original *vfile* variants. As with MOAT-a, MOAT-v offers

Deleted: of GENCODE annotation

Deleted: and

users the option to conduct a significance analysis of the annotations' whole genome signal scores, using the scores of the intersecting permuted variants as the background.

3.3 MOATsim: Simulated Somatic Variant Datasets

In addition to the main MOAT programs, we have developed a variant simulator, MOATsim, that reflects the levels of whole genome covariates that directly influence the background mutation rate. It is based off the variant permutation step in MOAT-v, but also incorporates bigWig signal files in its determination of permuted variant locations. Before working with the actual variant data, MOATsim imports the covariate data, and evaluates covariate signals over a set of whole genome bins (user-defined size, mappability blacklist regions excluded). These bins are then clustered based on their covariate signal profile using k-means clustering (Lloyd, 1982). With this information, variants can be permuted not just within their local genome context, but across all bins with the same covariate signal profile (i.e. within the same cluster). This additional functionality enhances MOATsim's accurate reflection of the expected somatic variant distribution of a human genome, and builds on the block bootstrap idea previously used for genome structure correction in (Bickel, et al., 2010).

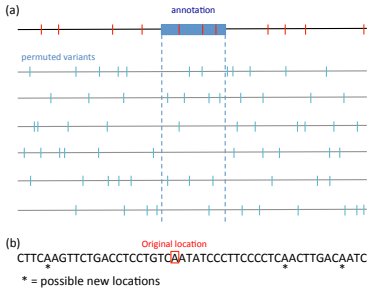

Figure 2 (a) In MOAT-v, the variant locations are permuted within the local genome context. The whole genome is divided into bins of a user-defined size, and variants are moved to new coordinates within the same bin, preserving the local mutation context. As with MOAT-a, *n* permutations are produced.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

4 RESULTS

4.1 MOAT-a

Table 1. Speed benchmark of MOAT-a with respect to the number of input annotations, and MOAT-v with respect to the number of parallel CPUs. MOAT-a's time trials involved generating 1000 permuted variant datasets, while MOAT-v's time trials involved generating one permuted variant dataset using ~8 million input variants, and 1,000,000-bp bins. For large datasets, the GPU algorithm substantially outperforms the CPU version. The timing of both versions included a whole genome signal analysis with Funseq functional impact scores.

MOAT-a		MOAT-v	
Annotations	GPU speedup	Parallel CPUs	Speedup
~14,000	1.01x	2	1.90x
~130,000	1.34x	4	3.55x
~3,000,000	6.26x	8	5.72x

We demonstrate the magnitude of the CUDA speedup by evaluating the running time of MOAT-a on datasets of various sizes. We took a dataset of pancancer 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 as input to demonstrate MOAT-a's scalability (Harrow, et al., 2012; Thurman, et al., 2012; Yip, et al., 2012). We tested MOAT-a's running time on these 3 annotation sets with the number of random bins $n = 1000$ (Table 1, left half). 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.

Due to the relative lack of verified noncoding regulatory elements associated with cancer, it is difficult to assess the accuracy of MOAT-a's predictions. Nevertheless, we demonstrate MOAT-a's usefulness for finding elevated mutation burdens in genomic elements by identifying highly mutated GENCODE transcription start sites, promoters, and enhancers, 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)(Benjamini and Hochberg, 1995). 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). Part of these results are also backed up by the concurrent Funseq (Fu, et al., 2014) functional impact analysis, such as TP53.

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 (Gabriel, et al., 2004) 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. This master process has the sole responsibility of dividing the workload across the workers processes, and gathering their results. 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. If a whole genome signal analysis is requested, the master process carries out the signal analysis in parallel with the workers' permutation computations. Table 1 (right half) 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. MOATsim's running time exhibited similar characteristics (data not shown).

MOAT-v was used on the same variant and annotation sets used to demonstrate MOAT-a's usefulness for finding elevated cancer

Deleted: 97x

Deleted: 50x

Deleted: 60x

Deleted: These

Deleted:

Deleted: that was conducted, although LMO3 was rendered non-significant after BH correction

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.

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.

Bickel, P.J., *et al.* Subsampling methods for genomic inference. *The Annals of Applied Statistics* 2010;4(4):1660-1697.

Fu, Y., *et al*. FunSeq2: A framework for prioritizing noncoding regulatory variants in cancer. *Genome biology* 2014;15(10):480.

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.

Lawrence, M.S., *et al.* Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature* 2013;499(7457):214-218.

Lloyd, S.P. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory* 1982;28:128-137.

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.