### Referee 1:

# Referee general comments:

*In this Application note, the authors described a program, MOAT, usable to detect regions of a genome with significantly more mutations than expected. The null expectation is based on random permutation of the mutations within a local region. To offer significant speed, the authors provide a GPU implementation of their program.*

***Author’s Response:***

We appreciate the comments of the reviewers.

### -- Conceptual questions/suggestions --

# Referee conceptual comment 1:

*With the best of my effort to understand, the problem that MOAT is aiming to solve is unclear. Due to this very broad definition of the problem, the solution chosen by the authors is difficult to understand and support. If the problem is to detect local density of variants, why algorithms typically used in peak-finding are not appropriate (knowing that they are very much quicker than permutations)?*

***Author’s Response:***

We thank the reviewers for their suggestions. We have updated our text (with accompanying supplemental figures) to for a tighter problem focus, and a discussion on the suitability of peak calling algorithms as follows:

“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 cancer driver detection.”

“A common analysis strategy in cancer driver detection is to look for genomic elements with a high variant accumulation across many patients.”

“On the surface, an analysis of whole genome variant density would appear to be a facile adaptation of standard peak calling algorithms that are more commonly used for finding regions of high read density. However, due to the aforementioned BMR heterogeneity, a mere scan for variant aggregations would not be a sufficiently specific evaluation of significantly elevated mutation burdens. This heterogeneity takes many forms, as indicated in Supp Fig XX, resulting in patient-specific rates, region-specific rates, and local nucleotide context-specific rates. Hence, a proper mutation burden evaluation must accommodate a comparison of observed variant density to the expected density under these context-specific background rates, which is not possible with standard peak calling algorithms.”





# Referee conceptual comment 2:

*Some comparisons with existing tools and a relevant example showing the unique approach of the authors is needed.*

***Author’s Response:***

How about compare out results with LARVA and oncodriver?

### -- Technical questions/suggestions --

# Referee technical comment 1:

*The authors mentioned they can preserve the trinucleotide context when choosing the new variant location. To my understanding the statistics is based on counts of variant over a region: how the precise location within a trinucleotide can be taken into account by only restricting the permutation location?*

***Author’s Response:***

We provide a new supplemental schematic to demonstrate the trinucleotide context preservation in the variant permutation procedure, as follows:



# Referee technical comment 2:

*For MOAT-a, the authors try to demonstrate the accuracy of the tool by stating that it can detect mutation burdens on known cancer-associated TSS. But no complete evaluation is reported. What is the specificity? What are the scores/ranks of these sites compared to all sites?*

***Author’s Response:***

We provide the following Q-Q plot of MOAT-a’s raw p-values.

Other aspects of this analysis to follow.



### Referee 2:

# Referee general comments:

*The authors developed a tool MOAT to perform mutation burden analysis with great speed. Generally, this tool is promising and useful. The paper is well organized and prepared.*

***Author’s Response:***

We thank the reviewer for these comments.

### Referee 3:

# Referee general comments:

*MOAT is a computational system for identifying significant higher-than-expected mutation burdens in genomic regions with an empirical, nonparametric method, which is highly useful for whole genome/exome studies. The expected mutation count is derived by simulating the expected distribution of background mutations. To produce this expected distribution, MOAT offers two types of permutation algorithm: one permutes the locations of annotations (MOAT-a), and one permutes the locations of variants (MOAT-v). MOAT utilizes GPU/OpenMPI parallelization to address running time issues in the scale of whole genome-wide analysis. The manuscript is well-written and concise.*

***Author’s Response:***

We thank the reviewer for these comments.

# Referee comment 1:

*MOAT relies on the assumption that the BMR (background mutation rate) remains approximately the same within a local context. Both MOAT-v and MOAT-a require the user to define the “local context” ([dmax], [dmin] in MOAT-a; [width] in MOAT-v). It may be very difficult for some users to set these parameters, can there be recommended values as default? How do these key parameters influence MOAT’s performance?*

***Author’s Response:***

We have added discussions to the text concerning the best practices for choosing runtime parameters for each of MOAT-a, MOAT-v, and MOAT-sim, as follows:

MOAT-a:

“The boundaries of the intervals for choosing permuted annotations—*d\_min* and *d\_max*—are adjustable to allow users to scale the surrounding genome context with respect to the size of the original annotation. The permutation intervals ideally will provide enough range to enable non-overlapping sampling. For example, in our analysis of transcription start site (TSS) mutation burdens, where TSSes are roughly 100 bp in length, we used a *d\_min* of 2kb and a *d\_max* of 10kb. As a rule of thumb, the choice of *d\_min* must be large enough to avoid the possibility that a potential mutation burden signal may "bleed" into the permutation intervals, while the choice of *d\_max* must be small enough that the BMR covariates are approximately constant within the permutation intervals.”

MOAT-v:

“The ability to adjust the width of the whole genome bins in MOAT-v enables users to select a width that represents regions in which the BMR covariates are expected to be approximately constant. Hence, the permutations created by MOAT-v will honor the regional mutation density expected due to these covariates. Our analysis of a few of the most significant covariates, including DNA replication timing, histone marks and GC content, indicate that they are roughly constant at a 100kb resolution.”

MOAT-sim:

“As with MOAT-v, bin width should be chosen based on the resolution at which the input covariates are approximately constant.”

# Referee comment 2:

*The MOATsim algorithm uses k-means clustering, how was the value k decided? Also, this program is called Simulated Somatic Variant Datasets, does this tool only works with somatic variants?*

***Author’s Response:***

We have added text to the manuscript explaining the choice of *k*, along with a supplemental figure:

“The whole genome bins are clustered using *k*-means clustering, which uses the distances between the bins' covariate signal profiles to group the bins into a predefined number of clusters. We selected *k* based on a within cluster sum of squares analysis (Fig XX)—the ratio of the within cluster sum of squares to the between cluster sum of squares did not change appreciably after *k*=30, and was subject to stochastic variation at higher values.”



With regard to germline analysis, MOAT's various programs are optimized for use with somatic variants. Adapting this framework for use with germline variant data is not straightforward. Germline variant distributions are influenced by linkage disequilibrium, and MOAT would require substantial additional development and optimization to accommodate this background mutation model. We plan to investigate such developments in future research.

# Referee comment 3:

*In Table 1, why the GPU is slower (0.86x) in ~14,000 MOAT-a? Is there a conceptual threshold number over which GPU will work better than CPU?*

***Author’s Response:***

At small input sizes, the overhead of setting up the computation on the GPU outweighs the speedup of the actual GPU computation. This is a common observation of many algorithms: given a naïve algorithm A and an optimized algorithm B, B will be faster than A in most cases, but there is a certain “crossing point” where very small inputs below a certain size will run faster in A than in B.

# Referee comment 4:

*In the cancer variant case study, the authors mentioned that several known cancer-associated elements were found to have significant mutation burdens, which is great. Can the authors provide a QQ plot of this study to show the power and there is no bias (in/deflation)?*

***Author’s Response:***

We provide the following Q-Q plot of MOAT-a’s raw p-values.



# Referee comment 5:

*Users may want some sample runs to get started. Sample input files for a complete run and recommended parameter values would be helpful. For example, can the authors provide a pipeline with all parameters they have used in their cancer variant case study? Also, can the parameter [blacklist file] set to be optional or disabled if there is no region list we want to remove? It would be even better if the authors could provide such a file that contains regions of “poor mappability, such as centromeres and telomeres, among others”.*

***Author’s Response:***

Generate simulated dataset as a demonstration.

# Referee comment 6:

*Minor typo in the README.txt file: “We provide precomputed Funseq scores on the MOAT website at: []”, the URL is missing*

***Author’s Response:***

We apologize for this oversight. The correct URL has been added to the README.