**Response to Reviewers**

We would like to thank the editor and reviewers for their comments.

**Genome-wide analysis of chemically induced mutations in mouse in phenotype-driven screens**

Denis C Bauer, BSc(Hons), PhD; Brendan J McMorran; Simon J Foote, FFSc (RCPA), FAHMS, FTSE; Gaetan Burgio, MSc, MBBS, MD, PhD

The reviewers’ comments and suggestions are boxed in gray and itemized below, followed by our responses and a description of the consequent revisions to the paper. Text from the manuscript or supplemental materials is italic and changes are “quoted in red”. Please note that in our responses we have used the abbreviations ‘pg’ and ‘para’ for page and paragraph, respectively.

**Reviewer: 1**

Figure 2 and 3 do not add much to the paper as they are poorly labeled and not easily interpretable.

Updated the figures … LIKE THIS

We also updated Figure 1 to 1) added binary-conversion for ADAM 2) corrected labelling

A better description of new features (in addition to a reasonable speed up) need to be discussed

**Reviewer 2:**

I wasn't able to find the reference implementations in R and Python.

We update the method section as follows

We also include the R and Python source file as supplementary material.

JADA

As a baseline for performance comparisons it would appear more natural (to me), to use a C++ implementation, which would actually be very easy to do using the bcftools API to read VCF/BCF and mlpack for clustering.

We investigated the suggested option and do not think it is appropriate as the kmeans clustering in mlpack is not multithreaded and hence performing worse than R and python whose kmeans approach is multithreaded. However we implemented a solution in Matlab which uses c++ libraries.

Scalability is an issue and will certainly become ever more important, especially in the light of projects like Genomics England and the Precision Medicine Initiative. So exploring the use concepts like Spark is certainly welcome. However, VariantSpark caters to only one very specific use case, rather than providing a platform to address many problems one faces. In order for it to become widely adopted, it would need to have a wider scope (in my opinion).

We agree that the reviewed version of the paper focuses on only one application we hence elaborated on the options of using the other machine learning algorithms

JADA*.*

R and Python do have specific packages to parse VCFs (Bioconductor/VariantAnnotation and pysam). I wonder whether using these would result in better performance.

We tried the VariantAnnotation package and found it to actually be faster than R’s built in library for reading delimited files. Therefore, we will use this package in our comparison. We also optimized our custom function that converts VCF variant strings to integers

(I.e. “1|0:99” to 1).

To elaborate, we now parse VCF files to matrix using readGT from the VariantAnnotation package.

vcf <- readGT("input.vcf")

Optimizations to our function include using fixed character matching instead of regex.

strsplit(a, ":"**, fixed = TRUE**)

As before, we apply our function to the matrix using vapply, with a pre-defined return type. We have compared it to other available apply methods, and find it to be the fastest in this case.

We include the R source file as a supplementary document.

To demonstrate scalability to today's data sets, using the present 1000 Genomes release (phase 3) would be appropriate.

We agree with the reviewer and have included the phase3 data in our paper.

JADA