Supplement

1) Background mutation rate (BMR) heterogeneity

Studies to detect cancer drivers in whole genome sequence (WGS) single nucleotide variant (SNV) data often involve an evaluation of a variant’s frequency relative to the BMR-calibrated expected frequency. A key hallmark of cancer is the breakdown of DNA repair processes, causing the entire genome to mutate randomly. Finding a significantly mutated element—which may be involved in cancer driving processes—must therefore involve a comparison to the BMR, and it is critically important to formulate an accurate BMR model (Lawrence, et al., 2013). Many studies limited their analyses to the exome, which has well-studied mutation patterns and functional characterization (Baca, et al., 2013). However, there are many important variants that drive cancer by affecting regulatory elements in the noncoding portion of the genome, the most well-known of which is TERT. Exome-only studies would not be telling the entire story, and therefore it is important to expand the scope of these analyses to the whole genome.

However, an accurate understanding of the whole genome’s BMR is much more difficult compared to the exome. The BMR varies widely from region to region, owing to a range of mutation “covariates”. These covariates include the timing of a region’s DNA replication synthesis during S phase of the cell cycle, the region’s GC percent, and the region’s sensitivity to Dnase I sites, which serves as a proxy for open chromatin. Supp Fig 1 illustrates the high variability of three such covariates—DNA replication timing, RNA-seq quantification levels, and DHS sensitivity—over a single chromosome in a liver cancer sample. A fully realized model would require a considerable amount of work to develop, taking into account not just these major influences, but many other factors such as methylation, histone modification, Hi-C and local context.

Supplementary Figure 1. Signal tracks for DNA replication timing, RNA-seq, and DHS over chromosome 11 in a single liver cancer sample. Overlaid with mutation density, it is clear that there is a strong correlation with DNA replication timing, and strong anti-correlations with RNA-seq and DHS.
With Mutations Overburdening Annotations Tool (MOAT), we sidestep the issues involved with using an explicit model, and instead focus on what background variants would be observed by creating permutations of the input data. These permutations occur in a local genome context that is small enough that the values of the covariates can be assumed to be constant. Hence, the BMR implied by the permuted data serves as a good indicator of whether the studied elements have significantly elevated mutation burdens.

2) MOAT-a Implementation

The annotation-based permutation algorithm for MOAT, a.k.a. MOAT-a, is amenable to mass parallelization on a graphics processing unit (GPU). These specialized hardware units were originally designed for rendering 3D computer graphics, and are optimized for performing parallel matrix transformations, which correspond to changes in the graphics drawn on the attached monitor. The unit’s onboard memory (known as “video RAM” or VRAM) is used to store information on 3D models composed of polygons, represented in the VRAM as the coordinates of the polygons’ vertices. The surfaces of these polygons are painted on with “textures”, which comprise the other major portion of the VRAM. Thousands of arithmetic processors—simplified versions of CPUs—carry out the transformations on this data (Nickolls, et al., 2008).

Any computer program that involves a large amount of arithmetic computation (compute intensity) relative to its amount of memory accesses (memory intensity) can be parallelized and run on a GPU, thus massively improving its running time. In recent years, the use of GPUs in this manner, known as general purpose GPU (GPGPU) computing, has skyrocketed. The Nvidia corporation, which manufactures and sells the Geforce and Quadro line of GPUs, has developed its own GPGPU language known as Compute Unified Device Architecture (CUDA). Although other frameworks have been developed since, such as Open Computing Language (OpenCL), CUDA remains the most mature GPGPU framework to this day. More recent efforts have involved the development of high level programming languages that make it easier for developers to specify what a parallel program should do without understanding too many specifics of the architecture, such as Thrust and TensorFlow (Abadi, 2016).

MOAT-a’s adaptation into a CUDA program was based on the realization that the original serial program spent a large fraction of its running time on compute intense steps that involved generating random numbers for permutation bin placement, and calculating the intersecting variants as an assessment of background mutation burden. Hence, we implemented a CUDA version that copies the variant and annotation data to the GPU’s heap, where the graphics stream processors carry out the necessary computations with a parallel speedup of up to several hundred times faster than the corresponding CPU version. We demonstrate that MOAT-a’s CUDA version scales very well with respect to the number of permutations (Supp Table 1). We believe this will greatly aid in WGS analysis given the expected deluge of sequencing data in the near future.

Supp Table 1. We ran MOAT-a’s CPU and GPU versions on a 7000 variant subset of our variant evaluation data (see Supplement section 5) on ~14,000 annotations. While the CPU version’s running time increases tremendously as the number of permutations increases, the GPU version’s running time remains stable.
| # of permutations | CPU runtime   | GPU runtime | Fold speedup |
|------------------|--------------|-------------|--------------|
| 1000             | 3min 18sec   | 14sec       | 14           |
| 10,000           | 40min 6sec   | 24sec       | 100          |
| 100,000          | 8hr 31min    | 2min 17sec  | 256          |

3) MOAT-v Implementation

The variant-based permutation algorithm for MOAT, a.k.a. MOAT-v, involves placing permuted variants within whole genome bins in a manner that reflects the observed mutation biases of the WGS sample. In addition to confining permuted variants to the same genome bin as the original variants—thus preserving the region’s mutation density—the variant placement scheme must preserve the trinucleotide mutation bias of the original variants. Differences in the biochemistry of the component bases mean that different nucleotides will mutate at different frequencies. Therefore, MOAT-v features a trinucleotide context preservation option that chooses permuted variant locations that have the same trinucleotide context as the original input variant (Supp Fig 2). In order to accommodate this feature, MOAT-v must import the reference genome into main memory, which makes MOAT-v substantially more memory intense than MOAT-a. As a result, we opted to parallelize MOAT-v for multi-core CPUs using the OpenMPI framework—a standard for coordinating multiple processes with an interprocess communication API (Gabriel, et al., 2004).

A master process will divide the work amongst the remaining worker processes, and communicate the necessary data to those processes, which serve as workers. Each worker is assigned a single chromosome, which means each process will be importing a separate reference FASTA file (the
reference genome is typically provided with a single FASTA file for each chromosome). This arrangement means that there will be no file I/O contention between the processes, enabling optimal parallel speedup. Ideally, there will be as many CPU cores as there are chromosomes (25, including chr1-22, X, Y, and M), but in the event that there are fewer cores, MOAT-v will assign as many chromosomes as it can to the workers, and when one of the workers is finished, it will signal the master that it is ready to process a new chromosome. Hence, if any processes finish early, they are assigned additional work to keep them maximally utilized. Another consequence of this arrangement is that there is no running time benefit to running MOAT-v on more than 25 cores.

4) MOAT-s Implementation

MOAT-s developed as a direct offshoot of MOAT-v, hence its parallelization scheme is very similar. MOAT-s enables the use of whole genome mutation covariate data to identify and cluster whole genome bins that mutate in a similar fashion. The variant placement step in MOAT-s permutes variants within each bin covariate cluster, instead of each bin (Supp Fig 3). Hence, the variant placement workload of a single cluster will likely involve reference genome import from several chromosomes, which will incur a small penalty to the overall running time.

The bin covariate clusters are produced by binning the genome, as with MOAT-v, and then importing one or more whole genome covariate signal tracks. These files must be provided to MOAT-s as bigWig files (Kent, et al., 2010), which makes them amenable to standard signal track analysis tools, such as bigWigAverageOverBed. MOAT-s calls upon bigWigAverageOverBed to compute the average covariate score for each bin, producing a covariate vector. These vectors are used to cluster the bins using k-means clustering, where the similarity of two bins is determined using the Euclidean distance between their covariate vectors. Since the output of this clustering

Supplementary Figure 3.
Schematic of MOAT-sim’s variant placement procedure. MOAT-sim bins the entire genome, whereupon it calculates the covariate values for each bin. The program then clusters bins with similar covariate values, represented here as bins with the same color. The input variants that fall within each cluster are then permuted to new locations chosen from the bins within the same cluster, honoring trinucleotide context preservation if requested.
algorithm depends on the selection of the number of clusters $k$ before running, we conducted an analysis of the clustering quality for a range of $k$ values. Supp Fig 4 illustrates the change in the within-cluster sum-of-squares (WSS) variance w.r.t $k$ when clustering on the major covariates that influence the BMR, including DNA replication timing, expression, and HiC. The within-cluster sum-of-squares variance demonstrates how much of the variance in the data is captured by the differences in the datapoints within each cluster. The total variance of the data is represented by the combination of the between-cluster sum-of-squares variance and the within-cluster sum-of-squares variance, and a quality clustering should capture most of the total variance between clusters, leaving intra-cluster variance very low. Hence, we sought to find the $k$ high enough to minimize the WSS while also low enough to remain computationally tractable. We observed that beyond $k=30$, the WSS begins to vary stochastically, which would indicate that for $k>30$, the clustering procedure is being influenced by noise. Therefore, we selected $k=30$ for MOAT-s.

![Supplementary Figure 4. Plot of within-cluster sum-of-squares (WSS) variation for different cluster numbers $k$. A lower WSS indicates more tightly packed clusters with more similar elements. Hence, a lower WSS means the clusters represent a cleaner separation between distinct groups. Beyond $k=30$, the WSS plateaus, and starts to exhibit minute, stochastic increases and decreases at higher $k$ values. Therefore, we chose $k=30$ for MOATsims.](image)

5) **MOAT evaluation: variant and annotation data**

The WGS variant data that we used in our evaluation of MOAT came from a range of published WGS cancer studies. The majority of this data was derived from a set of 507 whole genome cancer samples published in (Alexandrov, et al., 2013). This data includes sequence variants from patients with breast cancer, leukemia, liver cancer, lung cancer, lymphoma, medulloblastoma, pancreatic cancer, and pilocytic astrocytoma. In addition, we added 95 prostate cancer samples from (Baca, et al., 2013; Berger, et al., 2011; Cancer Genome Atlas Research, 2008; Weischenfeldt, et al., 2013), 26 unpublished glial tumor samples, 32 kidney cancer samples from the TCGA (Cancer Genome Atlas Research, 2008), and 100 stomach cancer samples from Wang et al. (Wang, et al., 2014).
The transcription start site (TSS) annotation data used in the MOAT evaluation was derived by parsing the GENCODE v16 (Harrow, et al., 2012) annotation file. We extracted the 100bp immediately upstream of each GENCODE gene annotation.

References

Abadi, M.A., Ashish; Barham, Paul; Brevdo, Eugene; Chen, Zhifeng; Citro, Craig; Corrado, Greg S.; Davis, Andy; Dean, Jeffrey; Devin, Matthieu; Ghemawat, Sanjay; Goodfellow, Ian; Harp, Andrew; Irving, Geoffrey; Isard, Michael; Jia, Yangqing; Jozefowicz, Rafal; Kaiser, Lukasz; Kudlur, Manjunath; Levenberg, Josh; Mane, Dan; Monga, Rajat; Moore, Sherry; Murray, Derek; Olah, Chris; Schuster, Mike; Shlens, Jonathon; Steiner, Benoit; Sutskever, Ilya; Talwar, Kunal; Tucker, Paul; Vanhoucke, Vincent; Vasudevan, Vijay; Viegas, Fernanda; Vinyals, Oriol; Warden, Pete; Wattenberg, Martin; Wicke, Martin; Yu, Yuan; Zheng Xiaoqiang. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv 2016.

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

Baca, S.C., et al. Punctuated evolution of prostate cancer genomes. Cell 2013;153(3):666-677.

Berger, M.F., et al. The genomic complexity of primary human prostate cancer. Nature 2011;470(7333):214-220.

Cancer Genome Atlas Research, N. Comprehensive genomic characterization defines human glioblastoma genes and core pathways. Nature 2008;455(7216):1061-1068.

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.

Kent, W.J., et al. BigWig and BigBed: enabling browsing of large distributed datasets. Bioinformatics 2010;26(17):2204-2207.

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

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

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.

Weischenfeldt, J., et al. Integrative genomic analyses reveal an androgen-driven somatic alteration landscape in early-onset prostate cancer. Cancer cell 2013;23(2):159-170.