New Variant Filtration Algorithm for the Genomic Variant Store

Announcing the release of the GATK Variant Extract-Train-Score (VETS) toolchain: a modern, scalable replacement for VQSR filtration in joint calling.

Samuel K. Lee[1], Kylee Degatano[2], George Grant[2] and Genevieve Brandt[3]
1 Broad GATK team
2 Broad Variants team
3 Broad User Education team

Overview

- We are changing the current filtering model in the Genomic Variant Store (GVS) joint calling pipeline from GATK VQSR to the new GATK Variant Extract-Train-Score (VETS) toolchain.
- VETS improves precision and recall, while also enabling analyses at a larger scale.
- A modular package using modern machine-learning toolkits, VETS replaces VQSR's Bayesian Gaussian Mixture Model with an isolation-forest algorithm.
- Starting Sept 1, GVS will run VETS with an isolation-forest model by default. GVS maintains the ability to run VQSR on callsets up to 10,000 genomes to reproduce the same results from past analyses.

Introducing VETS

As we pushed the scale capabilities of the GVS to larger joint callsets with the *All of Us* project, we ran into difficulties when attempting to train a filtration model with GATK's Variant Quality Score Recalibration (VQSR). In GVS joint calling, VQSR trained a Bayesian Gaussian Mixture Model (BGMM) on annotations from known variants and then used the model to filter out probable artifacts from the callset. Samuel Lee, a Machine Learning Scientist on the Broad Data Science Platform's Methods team, elected to revisit the implementation of VQSR to improve its scalability, robustness, and reproducibility, while also updating its design to follow modern machine-learning practices. The result is the GATK VETS toolchain, which comprises three GATK tools that can be used in combination to filter variants in joint calling: ExtractVariantAnnotations, TrainVariantAnnotationsModel, and ScoreVariantAnnotations. VETS outperforms VQSR on both precision and recall in our validation testing in less time and with fewer resources.

VQSR was originally released in 2010 at a time when genomics data was at a fraction of the scale seen today and entire callsets could easily fit into reasonable amounts of memory. As callsets grew larger over the past decade, VQSR began to struggle and our pipelines had to

resort to suboptimal strategies, like downsampling of training sets, to keep up. In contrast, VETS is better equipped to handle the future scale of genomic data because it is shardable and the steps that require more memory only load data as needed. We expect that VETS will scale to eventually support the All of Us program's target callset size of 1 million human genomes.

The landscape of machine-learning toolkits has also changed drastically in the decade since VQSR was developed. GATK VETS uses modern and reliable open-source machine-learning tools in a simple and modular way. The straightforward implementation of the open-source machine-learning algorithms in VETS makes the tool more maintainable and the resulting analysis is more transparent and reproducible. By default, GVS joint calling with VETS will use a new, more effective technique called isolation-forest modeling that has improved performance in benchmarking.

We benchmarked GVS with VETS on 3,000 human genome samples from AnVIL to assess scientific performance as well as runtime and memory requirements. GVS with VETS produces equivalent or better results for both SNPs and Indels when compared to VQSR (Figure 1). VETS used less memory and ran 13X faster than VQSR on this GVS callset (Figure 2).

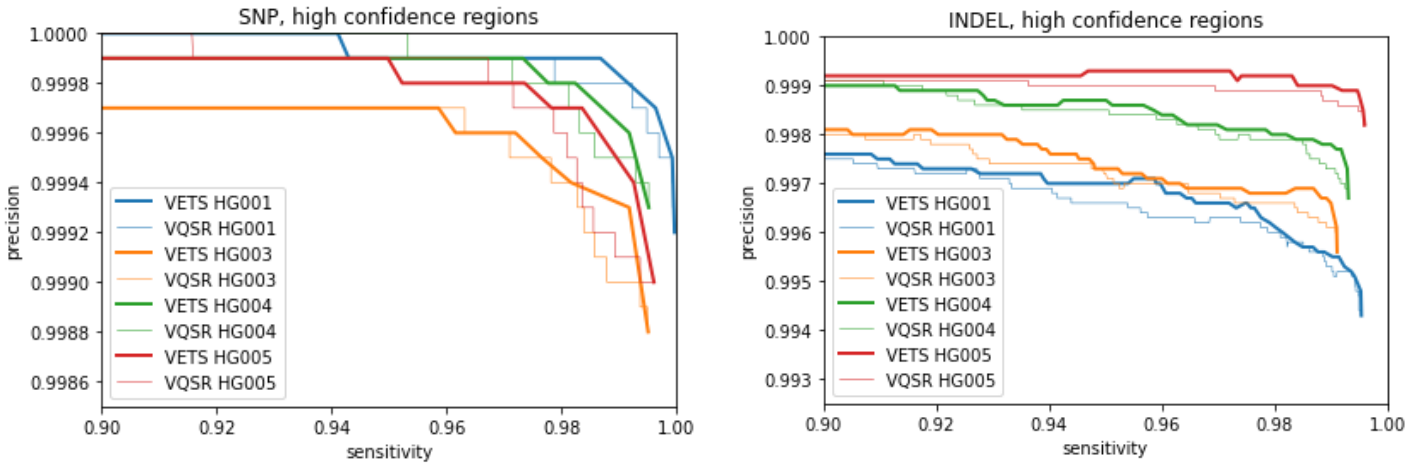

Figure 1. Precision and recall of a GVS joint callset filtered with the GATK VETS toolchain vs. with GATK VQSR. The data was 3,000 human whole genome samples from AnVIL.

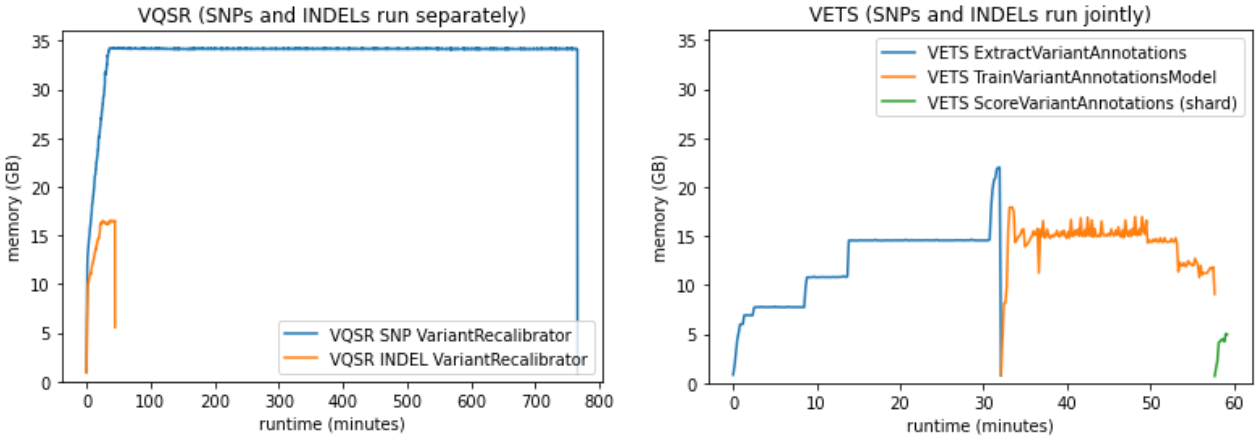

Figure 2. VQSR memory usage over time (left) vs. VETS toolchain memory usage over time (right) on a 3,000 human whole genome samples from AnVIL. We observed a decrease in memory requirements from ~35GB to ~25GB and in runtime from ~800 minutes to 60 minutes.

Starting September 1, GVS will run VETS with an isolation-forest model by default. For questions, please email variants@broadinstitute.org.

Technical details

Using VETS

To turn on VETS between now and September 1, set the input parameter `use_classic_VQSR=false`.

After September 1, if you require VQSR for callset reproducibility, you can use the input parameter `use_classic_vqsr=true` to run VQSR.

Changes in outputs

VCFs output from GVS with VETS will have some differences in fields and filters from prior GVS VCFs and GATK VCFs. Below, we explain differences unique to the release of VETS. For differences from past GATK VCFs, see this document.

Type	VETS	VQSR	Definition
Info field	`(AS_)SCORE`	`(AS_)VQSLOD`	`VQSLOD` is the log-odds ratio of being a true variant versus being an artifact under the trained model. Log-odds ratios are unbounded. In contrast, in VETS the analogous

			value is called `SCORE` and will be a value between -1 to 0. The higher the score, the more likely the variant is real.
Info field	`(AS_)CALIBRATION_SENSITIVITY`	`VQSRTrancheSNP99.90to100.00 (for example)`	`CALIBRATION_SENSITIVITY` is the exact sensitivity (in the range [0,1]) with respect to the calibration set, rather than the tranched (binned) sensitivity previously reported by VQSR.
Filter	`high_CALIBRATION_SENSITIVITY_SNP`	`low_VQSLOD_SNP`	Site failed SNP model calibration sensitivity cutoff (0.997)
Filter	`high_CALIBRATION_SENSITIVITY_INDEL`	`low_VQSLOD_INDEL`	Site failed INDEL model calibration sensitivity cutoff (0.99)

The filter fields for SNPs and Indels described in Table 1 have changed in name from "`low`" to "`high`" because they now correspond directly to the sensitivity cutoff, as opposed to the VQSLOD cutoff. The filter cutoffs remain the same in GVS. Compare two example definitions in the header:

VQSR

```
Java
##FILTER=<ID=low_VQSLOD_SNP,Description="Site failed SNP model sensitivity cutoff (99.7), corresponding with VQSLOD cutoff of -2.8042">
```

VETS

```
Java
##FILTER=<ID=high_CALIBRATION_SENSITIVITY_SNP,Description="Site failed SNP model calibration sensitivity cutoff (0.997)">
```

FAQ

Why the isolation forest model?

Variant filtration with VQSR has historically been framed as an ML classification problem with the goal of separating the variants from the artifacts. However, in typical use, we don't have good labeled "negative" training sets for artifacts---and most classification algorithms require

labels for both classes. So VQSR has to first build a "positive" model using labeled training variants, use that model to identify putative artifacts in the input data, use those putative artifacts as a labeled training set for building a "negative" artifact model, and then do ad hoc classification---phew!

When filtering with VETS in GVS, we instead frame the problem as one of outlier detection. That is: we have a bunch of labeled good variants---can we learn what those look like solely from a "positive" model? Then, if we see anything that really doesn't look like a good variant according to this model, we can infer that it is an outlier and most likely an artifact.

Isolation-forest modeling is one of the most heavily used algorithms for outlier detection. This is probably because it has the bonuses of 1) being relatively agnostic to the distribution of your data unlike VQSR's BGMM, which assumes a Gaussian distribution, 2) scaling well with the size of the dataset in terms of runtime and compute requirements, and 3) being relatively similar to random-forest methods, which are well studied.