GATK TUTORIAL :: Hard Filtering :: Worksheet

April 2018

This GATK workshop tutorial describes how to analyze and hard-filter raw variant calls based on site-level annotations. Our goal here is to get you on the road to understanding the **meaning and role of variant annotations in variant filtering**. To this end, we show how to use GATK to plot the distribution of annotation values, using a truth set to gauge the relative informativeness of specific variant annotations.

About the tools & software: RStudio and packages mentioned in the installation instructions are required for this tutorial. We will use RStudio to visualize annotation values.

TABLE OF CONTENTS

---

1 DATASETS

1.1 WGS trio callset

We start with a callset generated using HaplotypeCaller and GenotypeGVCF's GVCF workflow on public Illumina Platinum Genomes CEU Trio data. The trio consists of dad (NA12877), mom (NA12878) and son (NA12882). Calls are unfiltered (i.e. they did not undergo QUAL thresholding, and all the sites have "PASS" or "." in their "FILTER" field), are subset to chromosome 20, and include all variant types (i.e. indel, SNP and mixed sites).

1.2 Truthset

The GIAB (Genomes in a Bottle) v3.3 callset is a set of highly curated calls made for NA12878 as part of a *NIST* (the US National Institute for Standards and Technology) benchmarking project. It integrates several datasets that represent different sequencing platforms.

- The GIAB version we are using is v3.3, released 9/12/2016. A newer minor release that increases the number of indels by ~33%, v3.3.1, is available as of 10/14/2016.
- Briefly, the platforms are 10X Genomics Chromium, 300x Illumina paired end WGS, 100x Complete Genomics WGS, 1000x Ion Torrent exome sequencing, and SOLiD WGS. SOLiD, due to its shorter reads and low coverage, was only used to confirm variants and potential errors around SVs were excluded using SV calls derived from multiple PacBio callers and multiple integrated Illumina callers using MetaSV.
- The callset contains high-confidence variants outside of the high-confidence bed file regions so that you can compare complex variants or variants near boundaries. Details can be found here[1].

Comparing to this callset within its *high-confidence intervals* allows us to evaluate calls more thoroughly, because it gives us reasonable assumptions about exactly which calls are probably true positives (TPs), false positives (FPs), *as well as which sites are false negatives (FNs)*. To reiterate, knowing FN sites is dependent on restricting the evaluation to the *high-confidence intervals*. These regions exclude ~13% of the non-N hg19 genome where sequence quality and sequence context issues make reliable calling difficult for typical methods.

Note: the fact that so much of the reference has been removed from the high-confidence interval is highly problematic. As you can imagine this part of the reference may contain important genes and regulatory information, and variation there can be causal to disease. We have been developing a more inclusive data set called SynDip which doesn't suffer from the same issues.

---

[1] https://github.com/genome-in-a-bottle/giab_announcement_blog/blob/master/2016/20160909-New_GIAB_Highconfidence_calls_for_NA12878_AJTrio_and_Chinese_son.md

2 PLOTTING ANNOTATION PROFILES

One approach to understanding how informative variant annotations can be for filtering is to plot the distribution of annotation values observed for the variants in our callset. This is especially useful when we have a truth set available, because we can differentiate the profiles of TP versus FP variants, and from that derive some expectations that can be applied to novel callsets.

First, we apply some transformations to the data to get it in the right shape, then we use plotting formulas in RStudio to visualize the annotation profiles. Note that we focus on SNPs here, but you can apply similar principles for indels.

2.1 Use SelectVariants to subset the callset to just the SNPs of a single sample

We subset the trio callset to just the SNPs of the mother (sample NA12878), making sure to remove sites for which the sample genotype is *homozygous-reference.*

```
gatk SelectVariants \
      -R ref/ref.fasta \
      -V input_vcfs/trio.vcf.gz \
      -sn NA12878 \
      -select-type SNP \
      --exclude-non-variants \
      -O sandbox/motherSNP.vcf.gz
```

This produces a VCF with just the mother's SNPs. Note that the tool recalculates DP per site as well as AC, AF, and AN, to reflect only the subset sample(s).

2.2 Use VariantAnnotator to annotate true positives in the callset

We use VariantAnnotator to annotate which variants in our callset are also present in the truthset (GIAB), which are considered true positives. Variants not present in the truthset are considered false positives.

```
gatk VariantAnnotator \
      -R ref/ref.fasta \
      -V sandbox/motherSNP.vcf.gz \
      --resource giab:resources/motherGIABsnps.vcf.gz \
```

```
          -E giab.callsets \
          -O sandbox/motherSNP.giab.vcf.gz
```

This produces a single file where variants that are present in the truthset are annotated with the `giab.callsets` annotation, with the value indicating how many of the callsets used to develop the truthset agreed with that call.

2.3 Use VariantsToTable to tabulate annotation values to use with RStudio

We use R to visualize the distributions of annotation values. To produce an R-readable file, we convert the information from the callset into a table using VariantsToTable.

```
gatk VariantsToTable \
      -R ref/ref.fasta \
      -V sandbox/motherSNP.giab.vcf.gz \
      -F CHROM -F POS -F QUAL \
       -F BaseQRankSum -F MQRankSum -F ReadPosRankSum \
      -F DP -F DP_Orig -F FS -F MQ -F QD -F SOR \
      -F giab.callsets \
      -GF GQ \
      -O sandbox/motherSNP.giab.txt
```

This produces a table where each line represents a variant record from the VCF, and each column represents an annotation we have specified. Wherever the requested annotations are not present (some, like the RankSum annotations, cannot be calculated for some case figures), the value will be replaced by NA.

Notice how the tool differentiates cohort-level fields (`-F`) and sample-level genotype fields (`-GF`).

2.4 Use RStudio and a provided script to make density and scatter plots

Plotting the density of values for an annotation shows us the overall range and distribution of values observed in our callset. In combination with some basic knowledge of what the annotation represents and how it is calculated, this allows us to make a first estimation of value thresholds that might separate FPs from TPs.

Plotting the scatter of values for two annotations, one against the other, additionally shows us what tradeoffs we make when setting a threshold on annotation values individually. We provide pre-written R functions to easily make these types of plots in RStudio.

A. Load plotting functions

Open `R/plotting.R` script in a text editor and copy-paste its contents into RStudio's *Console*. Press *Enter*. This loads the plotting functions that you will use in a few minutes.

B. Import the annotation data

Import the `sandbox/motherSNP.giab.txt` you made earlier. Here you may need to adapt what you do depending on the version of RStudio you have.

In the most recent versions that we have used, you can do this with the button *Import Dataset* > *Import CSV.* You will need to change the *Delimiter* from *Comma* to *Tab,* and switch on the header option if it's not already set by checking the box *First row as names*. You will also need to change the value type of the `giab.callsets` column to `Character` instead of `Integer`.

In older versions the interface is a bit different. We don't have detailed instructions but if you have trouble finding the right way to do it, just run the commands below in the console. Be sure to adapt the file path to `motherSNP.giab.txt` based on where it lives on your machine!

```
library(readr)

motherSNP.giab <-
read_delim("/path/gatk_bundle/2-germline/sandbox/motherSNP.giab.txt","\t",
escape_double = FALSE, col_types = cols(giab.callsets = col_character()),
trim_ws = TRUE)
```

The second command is what RStudio composes and runs for you when you use the interface. You need to type it all on one line.

Finally, rename the `giab.callsets` column to `set` using this command:

```
names(motherSNP.giab)[names(motherSNP.giab) == 'giab.callsets'] <- 'set'
```

If you weren't able to change the value type of `'set'` in the beginning, you can do so now:

```
motherSNP.giab$set <- as.character(motherSNP.giab$set)
```

C. Make a density plot for the QUAL annotation

```
qual = makeDensityPlot(motherSNP.giab, "QUAL")
```

You can visualize the plot in RStudio by invoking the defined variable `qual`

D. Limit the x-axis by adding `xmax`

```
qual = makeDensityPlot(motherSNP.giab, "QUAL", xmax=10000)
```

→ *How does the density distribution relate to what the annotation represents? Can we find some clues of what might distinguish good vs. bad variants?*

E. Split by set

To make a density plot split by set, we add `split="set"`. Note that each split set represents fractions of its whole.

```
qual = makeDensityPlot(motherSNP.giab, "QUAL", xmax=10000, split="set")
```

→ *When we plot the split version, can we see a clear difference between the set distributions? What does that tell us?*

F. Make a QD density plot

```
qd = makeDensityPlot(motherSNP.giab, "QD")
```

→ *Why are there two peaks?*

G. Make a scatter plot of QD vs. DP using the `makeScatterPlot` function

Play with the axis limits to zoom in on subsets of the data.

```
QD_DP = makeScatterPlot(motherSNP.giab, "QD", "DP", split="set")
```

H. Make a scatterplot with marginal density plots

```
QD_DP_md = makeScatterPlotWithMarginalDensity(motherSNP.giab, "QD", "DP",
split="set", ymax=250, ptSize=0.5, ptAlpha=0.2)
```

→ *When plotting two annotations, does the combination of the two tell us anything more than either did separately?*

Try playing with these functions to generate plots for the various annotations available in the dataset.

---

3 APPLY HARD FILTERS

Based on the plots we generated, we're going to apply some filters to weed out false positives.

3.1 Use VariantFiltration to filter on QUAL < 30

By default, GATK GenotypeGVCFs filters out variants with QUAL < 10. Here we filter sites with QUAL < 30, using *VariantFiltration* as follows.

```
gatk VariantFiltration \
      -R ref/ref.fasta \
      -V sandbox/motherSNP.vcf.gz \
      --filter-expression "QUAL < 30" \
      --filter-name "qual30" \
      -O sandbox/motherSNPqual30.vcf.gz
```

This produces a VCF with all the original variants; those that failed the filter are annotated with the filter name in the FILTER column.

3.2 Use VariantFiltration to filter on multiple annotations simultaneously

We can filter on several annotations at the same time.

```
gatk VariantFiltration \
      -R ref/ref.fasta \
      -V sandbox/motherSNP.vcf.gz \
      --filter-expression "QD < 2.0 || FS > 60.0 || MQ < 40.0 || \
      MQRankSum < -12.5 || ReadPosRankSum < -8.0 || \
      SOR > 3.0 || QUAL < 30" \
      --filter-name "basic_filters" \
      -O sandbox/motherSNPbasicFilters.vcf.gz
```

The command filters on seven annotations. The double pipe (||) is equivalent to an *or*.

- If an annotation is missing, VariantFiltration conservatively forgoes judgement on the annotation. To fail such sites, use the `--missingValuesInExpressionsShouldEvaluateAsFailing` option.