(Notebook) Intro to using Mutect2 for somatic data

GATK Team
June 25, 2024 at 3:15 PM · Updated

In this hands-on tutorial — the Terra Workspace of which is available here — we will call somatic short mutations, both single nucleotide and indels, using GATK4 Mutect2 and FilterMutectCalls.

Example data are based on a breast cancer cell line and its matched normal cell line, derived from blood. Both cell lines are consented and known as HCC1143 and HCC1143_BL (blood normal), respectively and are 2x76 paired end whole exome sequences aligned to GRCh38.

This tutorial was last tested with the GATK v4.1.4.1 and IGV v2.8.0.

See GATK Tool Documentation for further information on the tools we use below. For a primer on somatic calling, read here.

Set up your Notebook

Set runtime values

If you are opening a notebook for the first time, or you didn't adjust any runtime values, now is the time to edit them. Click on the gear icon in the upper right to edit your Notebook Runtime. Set the values as specified below:

Option	Value
Environment	Default
Profile	Custom
CPU	4
Disk size	100 GB
Memory	15 GB
Startup Script	`gs://gatk-tutorials/scripts/gatk_4141.sh`

Then click the "Update" button. Terra will begin to create a new runtime with your settings. When it is finished, you will see a pop-up asking you to apply the new settings.

Check kernel type

A kernel is a *computational engine* that executes the code in the notebook. For this particular notebook, we will be using a Python 3 kernel so we can execute GATK commands using *Python Magic* (`!`). In the upper right corner of the notebook, just under the Notw click on the Jupyter logo in the upper left corner.

For this tutorial, we need to copy some files from this temporary folder to and from our workspace bucket. Run the two commands below to set up the workspace bucket variable and the file paths inside your notebook.

Tool Tip: To run a cell in a notebook, press `SHIFT + ENTER`

```

# Set your workspace bucket variable for this notebook.

import os BUCKET = os.environ['WORKSPACE_BUCKET']

# Set workshop variable to access the most recent materials

WORKSHOP = "workshop_2002"

# Create directories for your files to live inside this notebook

!mkdir -p /home/jupyter-user/notebooks/sandbox ```
```

Check data permissions

For this tutorial, we have hosted the starting files in a public Google bucket. First, check that the data is available to your user account using the command below. If it is not, then you simply need to install Google Cloud Storage.

```
# Check if data is accessible. The command should list several gs:// URLs.

! gsutil ls gs://gatk-tutorials/$WORKSHOP/3-somatic/

# If you do not see gs:// URLs listed above, run this cell to install Google Cloud Storage.

# Afterwards, restart the kernel with Kernel > Restart.

# ! pip install google-cloud-storage
```

Download Data to the Notebook

Some tools are not able to read directly from a Google bucket, so you must download their files to the local notebook folder.

```
!gsutil -m cp -r gs://gatk-tutorials/$WORKSHOP/3-somatic/bams /home/jupyter-user/notebooks/
!gsutil -m cp -r gs://gatk-tutorials/$WORKSHOP/3-somatic/ref /home/jupyter-user/notebooks/
!gsutil -m cp -r gs://gatk-tutorials/$WORKSHOP/3-somatic/resources /home/jupyter-user/notebooks/
!gsutil -m cp -r gs://gatk-tutorials/$WORKSHOP/3-somatic/mutect2_precomputed /home/jupyter-user/notebooks/
```

Set up Integrative Genomics Viewer (IGV)

We will be using IGV in this tutorial to view BAM and VCF files. In order to do so without downloading each individual file, we will connect IGV with our Google account. - Download IGV to your local machine if you haven't already done so. - Follow these instructions to connect your Google account to IGV.

CALL SOMATIC SNV & INDELS WITH MUTECT2

Call somatic SNVs and indels and generate a BAMOUT

We start by calling somatic short mutations on our HCC1143 tumor sample and matched normal using Mutect2. This command produces:

1. A raw unfiltered somatic callset restricted to the specified intervals list
2. A BAM containing reassembled alignments
3. Mutect stats file named `/home/jupyter-user/notebooks/sandbox/1_somatic_m2.vcf.gz.stats`

```
!gatk Mutect2 \
-R /home/jupyter-user/notebooks/ref/Homo_sapiens_assembly38.fasta \
-I /home/jupyter-user/notebooks/bams/tumor.bam \
-I /home/jupyter-user/notebooks/bams/normal.bam \
-normal HCC1143_normal \
-pon /home/jupyter-user/notebooks/resources/chr17_m2pon.vcf.gz \
--germline-resource /home/jupyter-user/notebooks/resources/chr17_af-only-gn
-L /home/jupyter-user/notebooks/resources/chr17plus.interval_list \
-O /home/jupyter-user/notebooks/sandbox/1_somatic_m2.vcf.gz \
-bamout /home/jupyter-user/notebooks/sandbox/2_tumor_normal_m2.bam
```

➤ **What is the value of using a matched normal control?**

Mutect2 uses the matched normal to additionally **exclude rare germline variation** not captured by the germline resource and individual-specific artifacts.

Mutect2 uses a germline population resource towards evidence of alleles being germline. The simplified sites-only gnomAD resource retaining allele-specific frequencies is available at ftp://gsapubftp-anonymous@ftp.broadinstitute.org/bundle/Mutect2.

A panel of normals (PoN) has a vital role that fills a gap between the matched normal and the population resource. Mutect2 uses the PoN to catch additional **sites of noise in sequencing data**, like mapping artifacts or other somewhat random but systematic artifacts of sequencing and data processing.

Make a panel of normals (PoN)

The PoN used here was made using GATK4.beta.6 with **40 exome samples** aligned to GRCh38 from the 1000 Genomes Project. Ideally, the PoN should include **technically similar samples that were sequenced on the same platform**, e.g. HiSeqX, using the same chemistry and analyzed using the same reference genome and tool-chain. However, **even an unmatched PoN is better than no PoN at all**. This is because mapping artifacts and polymerase slippage errors occur for pretty much the same genomic loci for short read sequencing approaches.

To make your own PoN:

The tumor-only mode command below takes an extremely long time to run, so we are not doing it today. To run it at a later date/time you will need 40 normals to pass into the initial step, but the command structure for all three steps is given below.

1) Run Mutect2 in tumor-only mode on each normal BAM individually,

```
!gatk Mutect2 -R reference.fasta -I normal1.bam --max-mnp-distance 0 -O nor
!gatk Mutect2 -R reference.fasta -I normal2.bam --max-mnp-distance 0 -O nor
...
!gatk Mutect2 -R reference.fasta -I normal40.bam --max-mnp-distance 0 -O no
```

2) Create a GenomicsDB from the normal Mutect2 calls,

```
!gatk GenomicsDBImport -R reference.fasta -L intervals.interval_list \
  --genomicsdb-workspace-path pon_db \
  -V normal1.vcf.gz \
  -V normal2.vcf.gz \
  ...
  -V normal40.vcf.gz
```

3) and then Combine the normal calls using CreateSomaticPanelOfNormals.

```
!gatk CreateSomaticPanelOfNormals -R reference.fasta \
  --germline-resource af-only-gnomad.vcf.gz \
  -V gendb://pon_db \
  -O pon.vcf.gz
```

FILTER FOR CONFIDENT SOMATIC CALLS

In section 2.1, we generated an unfiltered Mutect2 callset. Now, we will use filtering tools to identify which mutation candidates are likely to be real somatic mutations.

Estimate cross-sample contamination

We estimate cross-sample contamination with two tools `GetPileupSummaries` and `CalculateContamination`.

Run `GetPileupSummaries` to summarize read support for a set number of known variant sites.

The tool tabulates read counts that support REF, ALT and OTHER alleles for the sites in the resource. This involves a known germline variant resource to **limit analysis to sites that are commonly variant**. Use a population germline resource containing only common biallelic variants

Here we use a human population germline resource, gnomAD. Let's run the tool on the tumor and the normal.

```

# Run on Tumor samples
```

!gatk GetPileupSummaries \ -I /home/jupyter-user/notebooks/bams/tumor.bam \ -V /home/jupyter-user/notebooks/resources/chr17*small*exac*common*3*grch38.vcf.gz \ *-L /home/jupyter-user/notebooks/resources/chr17*small*exac*common*3grch38.vcf.gz \ -O /home/jupyter-user/notebooks/sandbox/7*tumor*getpileupsummaries.table

Run on Normal samples

!gatk GetPileupSummaries \ -I /home/jupyter-user/notebooks/bams/normal.bam \ -V /home/jupyter-user/notebooks/resources/chr17*small*exac*common*3*grch38.vcf.gz \ *-L /home/jupyter-user/notebooks/resources/chr17*small*exac*common*3grch38.vcf.gz \ -O /home/jupyter-user/notebooks/sandbox/7*normal*getpileupsummaries.table ```

Each command produces a six-column table as shown. The `alt_count` is the count of reads that support the ALT allele in the germline resource. The `allele_frequency` corresponds to that given in the germline resource. Counts for `other_alt_count` refer to reads that support all other alleles.

``` !head /home/jupyter-user/notebooks/sandbox/7*tumor*getpileupsummaries.table

!head /home/jupyter-user/notebooks/sandbox/7*normal*getpileupsummaries.table ```

The tool considers *homozygous-variant* sites in the sample where the alternate allele frequency (AF) in the population resource ranges between 0.01 and 0.2. This range is adjustable. We can expect a lot of contamination by alternate alleles at sites where the alternate AF is large, so those sites wouldn't tell us much. Conversely, at homozygous-alternate sites where the variant allele is rare in the population, we are more likely to observe the presence of REF or other alleles if there was cross-sample contamination, and therefore we will be able to measure contamination more accurately.

### Estimate contamination with `CalculateContamination`.

The tool gives the fraction contamination. This estimation informs downstream filtering by FilterMutectCalls.

```
!gatk CalculateContamination \
-I /home/jupyter-user/notebooks/sandbox/7_tumor_getpileupsummaries.table \
-tumor-segmentation /home/jupyter-user/notebooks/sandbox/segments.table \
-O /home/jupyter-user/notebooks/sandbox/8_tumor_calculatecontamination.tabl
```

Let's also try out an additional feature of the tool. We can provide both the tumor and the matched normal pileup table. The pairing can allow for a slightly more accurate estimate.

```
!gatk CalculateContamination \
-I /home/jupyter-user/notebooks/sandbox/7_tumor_getpileupsummaries.table \
-matched /home/jupyter-user/notebooks/sandbox/7_normal_getpileupsummaries.t
-tumor-segmentation /home/jupyter-user/notebooks/sandbox/segments.table \
-O /home/jupyter-user/notebooks/sandbox/8_pair_calculatecontamination.table
```
```

The resulting files from the two variations each give the fraction contamination. Run the two cells below to view results:

``` !cat /home/jupyter-user/notebooks/sandbox/8*tumor*calculatecontamination.table

!cat /home/jupyter-user/notebooks/sandbox/8*pair*calculatecontamination.table ```

For our small tumor BAM file, you can see the contamination is ~0.0191 with an error of ~0.0022. We get a slightly lower number, ~0.0120 +/– 0.00454 for the matched estimate. For the full BAM file, we see a slightly larger contamination number. This threshold informs you to be wary of calls with less than that number for the alternate allele fraction.

---

## Apply filters with `FilterMutectCalls`

In this step, we filter the small data, 1*somatic*m2.vcf, with `FilterMutectCalls`. The tool **uses the annotations within the callset, and if provided, uses the contamination table** in filtering. Default settings are tuned for human somatic analyses.

```
!gatk FilterMutectCalls \
-R /home/jupyter-user/notebooks/ref/Homo_sapiens_assembly38.fasta \
-V /home/jupyter-user/notebooks/sandbox/1_somatic_m2.vcf.gz \
--contamination-table /home/jupyter-user/notebooks/sandbox/8_pair_calculate
--stats /home/jupyter-user/notebooks/sandbox/1_somatic_m2.vcf.gz.stats \
--tumor-segmentation /home/jupyter-user/notebooks/sandbox/segments.table \
-O /home/jupyter-user/notebooks/sandbox/9_somatic_oncefiltered.vcf.gz
```

This produces a VCF callset `9_somatic_oncefiltered.vcf.gz` and index. Calls that are **likely true positives get the PASS label** in the FILTER field, and calls that are likely false positives are labeled with the reason(s) for filtering in the FILTER field of the VCF. We can view the available filters in the VCF header using

```
!zgrep '##FILTER' /home/jupyter-user/notebooks/sandbox/9_somatic_oncefilter
```

This step seemingly applies **20 filters, including contamination**. However, if an annotation a filter relies on is absent, the tool skips the particular filtering. The filter will still appear in the header. For example, the `duplicate_evidence` filter requires a nonstandard annotation that our callset omits.

REVIEW CALLS WITH IGV

Deriving a good somatic callset involves comparing callsets from different callers, manually reviewing passing and filtered calls and, if necessary, additional filtering. Manual review extends from deciphering call record annotations to the nitty-gritty of reviewing read alignments using a visualizer.

How do we account for variant calls based on the read data? Remember Mutect2 reassembles reads just like HaplotypeCaller, so the clean alignments will not necessarily
```

reflect the calls. We must examine the BAMOUT that Mutect2's graph-assembly produces. We already generated this BAMOUT in section 1.1 (`sandbox/2_tumor_normal_m2.bam`). We are going to copy it into our bucket for loading into the IGV.

Copy the result of our analysis into the workspace bucket so we can load it into IGV.

We use the google cloud utilities (`gsutil`) command for copy (`cp`) to put our sandbox files into the bucket wher we can load them into the IGV. The other files (`bams` , `resources` , `mutect2_precomputed`) were already made available through our gatk-tutorials bucket so we don't have to copy those again.

```
! gsutil cp /home/jupyter-user/notebooks/sandbox/2_tumor_normal_m2.bam $BUC
! gsutil cp /home/jupyter-user/notebooks/sandbox/2_tumor_normal_m2.bai $BUC
```

Set up your IGV window

Before continuing, load Human (hg38) as your reference genome.

It is important to do this first, as changing the reference genome will require you to reload all files you may have previously loaded.

Next, go to `File` --> `Load from URL` and load these files in order:

```

# Run this cell to print out the URLs of files to load. Copy and paste the result into IGV.

! echo gs://gatk-tutorials/$WORKSHOP/3-somatic/mutect2*precomputed/9*somatic*oncefiltered.vcf.gz ! echo $BUCKET/sandbox/2*tumor*normal*m2.bam ! echo gs://gatk-tutorials/$WORKSHOP/3-somatic/bams/tumor.bam ! echo gs://gatk-tutorials/$WORKSHOP/3-somatic/bams/normal.bam ```

## Navigate to the location of the genome where variants were called

Navigate IGV to the **TP53** locus at **chr17:7,666,402-7,689,550.**


Fig02

- Zoom into **chr17:7,673,333-7,675,077** to better see the somatic call
- Scroll through the data and notice the coverage for the samples.

**➤ We see a C→T variant light up in red for the tumor but not the normal. What do you think is happening in 2*tumor*normal_m2.bam?**
```

➤ What does the coverage tell you?

If these alignments seem hard to decipher, it is because we need to tweak some settings.

To make room to focus on track [2*tumor*normal_m2.bam], we need to remove the tumor and normal bam tracks. Shift+select on the left panels for both of those tracks & their corresponding coverage. Then right-click and `Remove Tracks` to remove the tumor and normal BAMs.

Next, go to `View>Preferences>Alignments`. Uncheck `Downsample reads`.

Now, right-click on the alignments track and

- Group by sample
- Color alignments by tag: HC
- Sort by base

You can now scroll and click on a read in each group to determine which group belongs to which sample. The depth at this site is very high, but you can see a squished view of all the reads in the image to the right.

➤ What are the three grouped tracks for the bamout? What do the colors indicate? What differentiates the pastel versus gray reads?

➤ How do you feel about this somatic call?

ANNOTATE MUTATIONS WITH FUNCOTATOR

Another approach to filtering mutation calls is by the significance of their functional impact. For example, a stop codon in the middle of a protein coding region or a missense mutation that changes how a protein functions is more significant than a silent mutation or a mutation in the middle of an intron.

To gauge functional impact, we must know which regions of the genome code for protein sequence and which correspond to elements important to gene expression. Transcript annotation resources such as GENCODE capture such information in a standardized General Transfer Format (GTF).

GATK4 Funcotator annotates variant alleles with information from any number of annotation resources. The annotation resources must be organized in a particular way. You can download prepared Funcotator resource bundles from `gs://broad-public-datasets/funcotator/` or use GATK4 `FuncotatorDataSourceDownloader` to download the latest data sources directly from your GATK4 install. For this tutorial, we have specially prepared a small annotation resource.

Annotate the `9_somatic_oncefiltered.vcf.gz` mutation callset with the resource.

```
!gatk Funcotator \
--data-sources-path /home/jupyter-user/notebooks/resources/funcotator_dataS
--ref-version hg38 \
-R /home/jupyter-user/notebooks/ref/Homo_sapiens_assembly38.fasta \
-V /home/jupyter-user/notebooks/mutect2_precomputed/9_somatic_oncefiltered.
-O /home/jupyter-user/notebooks/sandbox/12_somatic_oncefiltered_funcotate.v
--output-file-format VCF
```

This produces a VCF callset with annotations. If needed, Funcotator can instead write results in historic Mutation Annotation Format (MAF) given `--output-file-format MAF`.

➤ Examine the annotations for the TP53 mutation that we viewed earlier in IGV, at chr17:7674220.

```

!zgrep chr17 /home/jupyter-user/notebooks/sandbox/12*somatic*oncefiltered_funcotate.vcf.gz | grep 7674220

```

We see an arginine (R) to glutamine (Q) missense mutation at position 248. In our 124 mutation records, 21 are annotated with MISSENSE, and of these, ten PASS filters.

Count the total mutation records

```

!zgrep -v "^#" /home/jupyter-user/notebooks/sandbox/12*somatic*oncefiltered_funcotate.vcf.gz | wc
```

Count the number of MISSENSE records

```
!zgrep -v "^#" /home/jupyter-user/notebooks/sandbox/12*somatic*oncefiltered_funcotate.vcf.gz | grep "|MISSENSE|" | wc
```

Count the number of MISSENSE records that PASS filters

```
!zgrep -v "^#" /home/jupyter-user/notebooks/sandbox/12*somatic*oncefiltered_funcotate.vcf.gz | grep "|MISSENSE|" | grep PASS | wc

```

Review filtered indels to study the logic behind different filters

Explore a few insertion and deletion sites in IGV and consider the evidence that supports the filtering decisions.

CHROM	POS	REF	ALT	
chr17	7,221,420	CACTGCCCTAGGTCAGGA	C	artifact*in*norm
chr17	19,748,387	G	GA	contaminatio
chr17	50,124,771	GCACACACACACACACA	G,GCACA,GCACACA	artifact*in*norm

Here are a few more filtered indel calls to explore.

CHROM	POS	REF	ALT	FILTER
chr17	26,982,033	G	GC	artifact*in*normal;bad*haplotype;clustered*eve
chr17	35,671,734	CTT	C,CT,CTTT	artifact*in*normal;multiallelic;panel*of*normals
chr17	47,157,394	CAA	C,CAAA	artifact*in*normal;germline*risk;panel*of_norm
chr17	68,907,890	GA	G,GAA	artifact*in*normal;base*quality;germline*risk;pa
chr17	69,182,632	C	CA	artifact*in*normal;contamination;str*contracti*

Was this article helpful?

✓ Yes × No

1 out of 1 found this helpful

Return to top

Recently viewed articles

Mutect2

Funcotator Information and Tutorial

PrintReads

FuncotatorDataSourceDownloader

Panel of Normals (PON)

Related articles

Mutect2

(How to) Call somatic mutations using GATK4 Mutect2

FAQ for Mutect2

Mutect2

Mutect2

---

3 comments

Sort by

Ashi
November 18, 2020 at 3:50 AM

Hi,

My sample is WGS FFPE (tumor matched with normal tissue). I would like to add "Read Orientation Artifacts Workflow" to this workflow.

How do I do it? Could you please add some hands-on script in this document?

I checked this document (https://gatk.broadinstitute.org/hc/en-us/articles/360035531132), but not sure.

Thanks!

0

Ashi
November 18, 2020 at 4:11 AM · Edited

Hi,

I have an another question.

For `GetPileupSummaries step, can I specify normal.bam files separately like this?:`

```
gatk GetPileupSummaries \
   -I normal1.bam \
   -I normal2.bam \
   -I normal3.bam \
   -V gnomad.vcf.gz \
   -L common_snps.interval_list \
```

```
    -O normal.pileups.table
```

0

Rafael Viana
July 8, 2021 at 3:31 PM

Very nice article.
I just have 2 questions.

1- Once you filter your variants and have the PASS flag.
Is there a tool to remove the ones without the PASS flag?

2- Here you get the differences between 2 samples.
Lets say you have a suspect variant in your healthy sample, and you expect to find it in the sick sample
Is it possible to check the samples similarities?

Kind regards

0

Please sign in to leave a comment.