Variant Calling Pipeline using GATK4

Published by **Mohammed Khalfan** on 2020-03-25

Last updated on 2021-02-17

This is an updated version of the variant calling pipeline post published in 2016 (link). This updated version employs GATK4 and is available as a containerized Nextflow script on GitHub.

Identifying genomic variants, including single nucleotide polymorphisms (SNPs) and DNA insertions and deletions (indels), from next generation sequencing data is an important part of scientific discovery. At the NYU Center for Genomics and Systems Biology (CGSB) this task is central to many research programs. For example, the Carlton Lab analyzes SNP data to study population genetics of the malaria parasites *Plasmodium falciparum* and *Plasmodium vivax*. The Gresham Lab uses SNP and indel data to study adaptive evolution in yeast, and the Lupoli Lab in the Department of Chemistry uses variant analysis to study antibiotic resistance in *E. coli*.

To facilitate this research, a bioinformatics pipeline has been developed to enable researchers to accurately and rapidly identify, and annotate, sequence variants. The pipeline employs the Genome Analysis Toolkit 4 (GATK4) to perform variant calling and is based on the best practices for variant discovery analysis outlined by the Broad Institute. Once SNPs have been identified, SnpEff is used to annotate, and predict, variant effects.

This pipeline is intended for calling variants in samples that are clonal – i.e. a single individual. The frequencies of variants in these samples are expected to be 1 (for haploids or homozygous diploids) or 0.5 (for heterozygous diploids). To call variants in samples that are heterogeneous, such as human tumors and mixed microbial populations, in which allele frequencies vary continuously between 0 and 1 researcher should use GATK4 Mutect2 which is designed to identify subclonal events (*workflow coming soon*).

available for BQSR. In the absence of a gold standard the pipeline performs an initial step detecting variants without performing BQSR, and then uses the identified SNPs as input for BQSR before calling variants again. This process is referred to as bootstrapping and is the procedure recommended by the Broad Institute's best practices for variant discovery analysis when a gold standard is not available.

This pipeline uses nextflow, a framework that enables reproducible and portable workflows (https://www.nextflow.io/). The full pipeline and instructions on how to use the Nextflow script are described below.

Script Location

greene

```
/scratch/work/cgsb/scripts/variant_calling/gatk4/main.nf
```

1. You don't need to copy the script, but you should copy the file `nextflow.config` from the above directory into a new project directory.
   (`/scratch/work/cgsb/scripts/variant_calling/gatk4/nextflow.config`)
2. If you want to copy and run the `main.nf` script locally, you must copy the `/bin` directory as well. This needs to be in the same directory as `main.nf` .
3. You will set parameters for your analysis in the `nextflow.config` file
4. See How to Use and Examples below to learn how to configure and run the script.
5. This Nextflow script uses pre-installed software modules on the NYU HPC, and is tailored to the unique file naming scheme used by the Genomics Core at the NYU CGSB.

github

```
https://github.com/gencorefacility/variant-calling-pipeline-gatk4
```

1. This version of the script is configured for standard Illumina filenames
2. This version of the script does not use software modules local to the NYU HPC, it is instead packaged with a Docker container which has all tools used in the pipeline. Integration with Docker allows for completely self-contained and truly reproducible analysis. This example shows how to run the workflow using the container. The container is hosted on Docker at:

Full List of Tools Used in this Pipeline

- GATK4
- BWA
- Picard Tools
- Samtools
- SnpEff
- R (dependency for some GATK steps)

How to Use This Script

Setting Up Nextflow

This pipeline is written in Nextflow, a computational pipeline framework. The easiest way to get setup with Nextflow is with conda. On the NYU HPC, conda is already installed and available as a module.

```
module load anaconda3/2019.10
```

Note: Use `module avail anaconda` to check for the latest conda version.

Update: *NYU HPC users can load Nextflow using the module load command. There is no need to install via conda. Simply load the module and skip to "Setting Up The Script" below.*

For users outside NYU, see the conda installation instructions, then come back and follow the rest of the instructions below.

Once you have conda loaded, make sure you add the bioconda channel (required for Nextflow) as well as the other channels bioconda depends on. **It is important to add them in this order** so that the priority is set correctly (if you get warnings that a channel exists and is being moved to the top, that's OK keep going):

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
```

Create a conda environment for Nextflow. I call it 'nf-env' but you can call it anything you like.

```
conda create --name nf-env
```

We'll install Nextflow inside this newly created environment. Anytime we want to use Nextflow, we'll need to load the environment first using the following command:

```
conda activate nf-env
```

When a conda environment has been loaded, you'll see its name in parenthesis at the command prompt, for example:

```
(nf-env) [user@log-1 ~]$
```

Once you've activated your conda environment, install nextflow using:

```
conda install nextflow
```

(When you're finished with nextflow, deactivate the environment using `conda deactivate`)

Setting Up The Script

Once you have Nextflow setup, we can run the pipeline. The pipeline requires six mandatory input parameters to run. The values can be hard coded into a config file, or provided as command line arguments (examples below).

1) `params.reads` The directory with your FastQ libraries to be processed, including a glob path matcher. Must include a paired end group pattern matcher (i.e. `{1,2}`).
Example:

```
params.reads = "/path/to/data/*_n0{1,2}_*.fastq.gz"
```

or

```
params.reads = "/path/to/data/*_R{1,2}_*.fastq.gz"
```

2) `params.ref` Reference genome to use, in fasta format. If you're at the NYU CGSB, there's a good chance your genome of interest is located in the Shared Genome Resource (`/scratch/work/cgsb/genomes/Public/`). Selecting a reference genome from within the Shared Genome Resource ensures that all index files and reference dictionaries required by BWA, Picard, GATK, etc. are available. If you're using your own reference data, you will need to build index files for BWA (using BWA), a fasta index (using samtools), and a reference dictionary (using Picard Tools). These files need to be located in the same directory as the reference fasta file.

3) `params.snpeff_db` The name of the SnpEff database to use. To determine the name of the SnpEff DB to use, ensure you have snpeff installed/loaded and then issue the `databases` command.

```
# $SNPEFF_JAR is the path to the snpEff.jar file.
# On the NYU HPC this is automatically set to the
# snpEff.jar path when the module is loaded
java -jar $SNPEFF_JAR databases | grep -i SPECIES_NAME
```

From the list of returned results, select the database you wish to use. The value of the **first** column is the value to input for `params.snpeff_db`.

For example if you wanted to use the Arabidopsis Thaliana SnpEff database:

```
java -jar $SNPEFF_JAR databases | grep -i thaliana
```

Output:

```
athalianaTair10   Arabidopsis_Thaliana http://downloads.sourceforge.net/...
```

Then:

```
params.snpeff_db='athalianaTair10'
```

Note: If your organism is not available in SnpEff, it is possible to create a custom SnpEff Database if a reference fasta and gff file are available.

4) `params.outdir` By default, output files will go into `params.outdir/out`, temporary GATK files will go into `params.outdir/gatk_temp`, SnpEff database files will go into `params.outdir/snpeff_db`, and Nextflow will run all analysis in `params.outdir/nextflow_work_dir`.These individual paths can be overridden by specifying the `params.out`, `params.tmpdir`, `params.snpeff_data`, and the `workDir` parameters respectively. Note: There is no `params.` prefix for `workDir`.

5) `params.pl` Platform. E.g. Illumina (required for read groups)

6) `params.pm` Machine. E.g. nextseq (required for read groups)

Examples

Note: If you are working on the NYU HPC, you should run all nextflow jobs in an interactive slurm session. This is because Nextflow uses some resources in managing your workflow.

Once you have set up your nextflow environment you can perform your analysis in multiple ways.

Copy the `nextflow.config` file into a new directory (see Script Location). Edit the first six parameters in this file, e.g.:

```
params.reads = "/path/to/reads/*_n0{1,2}_*.fastq.gz"
params.ref = "/path/to/ref.fa"
params.outdir = "/scratch/user/gatk4/"
params.snpeff_db = "athalianaTair10"
params.pl = "illumina"
params.pm = "nextseq"
```

After activating your conda environment with Nextflow installed, run:

```
nextflow run main.nf
```

By default, Nextflow will look for a config file called `nextflow.config` inside the directory from which Nextflow was launched. You can specify a different config file on the command line using `-c` :

```
nextflow run main.nf -c your.config
```

Note: Parameters hard coded in the config can be overridden by arguments passed through the command line. See following scenario.

Scenario 2: Call the script providing all required parameters through the command line (after activating your Nextflow conda environment)

```
nextflow run main.nf \
    --reads "/path/to/reads/*_n0{1,2}_*.fastq.gz" \
    --ref "/path/to/ref.fa" \
    --outdir "/scratch/user/gatk4/" \
    --snpeff_db "athalianaTair10" \
    --pl "illumina" \
    --pm "illumina"
```

Note: You are able to set/override any and all parameters in this way.

Scenario 3: Call directly from GitHub

Nextflow allows you to run scripts hosted on GitHub directly. There is no need to download the script (main.nf) to your local computer. Rather, you can call the variant calling pipeline script directly from github by entering the following command:

parameters (see Scenario 1), or pass parameters through the command line (see Scenario 2).

Scenario 4: Use with a Docker container

All software tools used in the pipeline have been packaged in a Docker container. You can instruct Nextflow to use a containerized Docker image for the analysis by passing the `-with-docker` or `-with-singularity` flags and the container. Calling the script directly from GitHub and telling it to use this docker image looks like this:

```
nextflow run gencorefacility/variant-calling-pipeline-gatk4 -with-docker
gencorefacility/variant-calling-pipeline-gatk4
```

Launching Nextflow

Once Nextflow begins running, it will actively manage and monitor your jobs. If you're working on the HPC at NYU, the corresponding nextflow.config file contains the line `process.executor = 'slurm'` which instructs Nextflow to submit jobs to SLURM. You can monitor your jobs in real time using watch and squeue:

```
watch squeue -u <netID>
```

Note: Since Nextflow actively manages your jobs, the session in which Nextflow is running must remain open. For example, if Nextflow is running and you lose power or network, Nextflow will be terminated. To avoid this, I recommend running Nextflow inside a tmux session, using nohup, or even submitting the nextflow script as a SLURM job. My preference is to use tmux (terminal multiplexer).

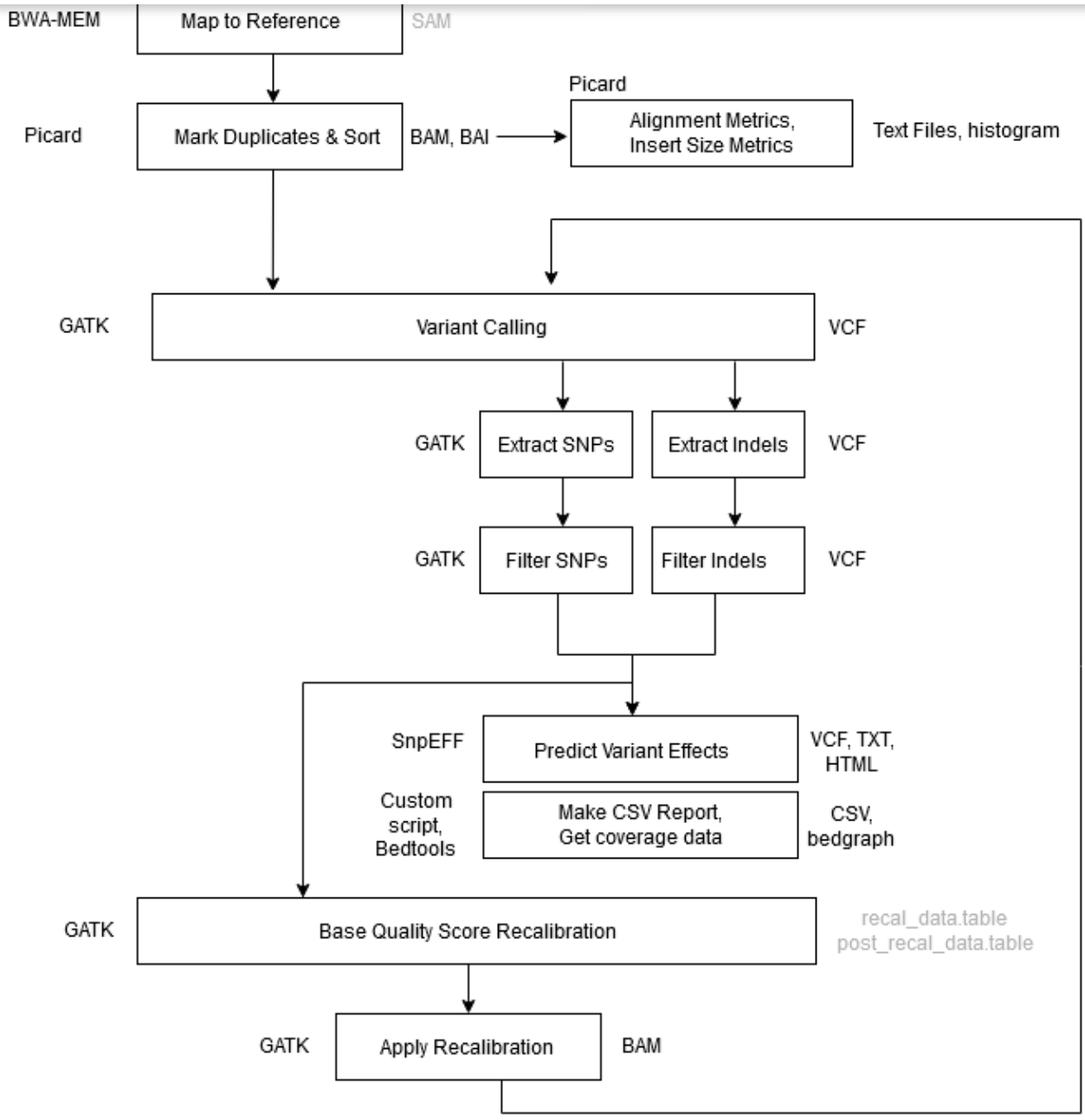
Enter the command `tmux` to enter a tmux session. Launch an interactive slurm session if you are on the NYU HPC. Activate your conda environment. Now, you can launch your Nextflow script.

"Detach" from the tmux session at any time by pressing " `ctrl` " + " `B` " together, then pressing " `D` " (for detach). If you ever lose power, connection, or detach, you can reattach your tmux session by typing `tmux a` . You will find your session running just as when you last left it.

If your Nextflow script does get interrupted for some reason, you can always resume the pipeline from the last successfully completed step using the `-resume` flag:

```
nextflow run main.nf -resume
```

Workflow Overview



Step 1	**Alignment – Map to Reference**
Tool	BWA MEM
Input	.fastq files reference genome
Output	aligned_reads.sam

	reproducibility) Readgroup info is provided with the `-R` flag. This information is key for downstream GATK functionality. GATK will not work without a read group tag.

Command	`bwa mem \`
	`-K 100000000 \`
	`-Y \`
	`-R '@RG\tID:sample_1\tLB:sample_1\tPL:ILLUMINA\tPM:HISEQ\tSM:sample_1' \`
	`ref.fa \`
	`sample_1_reads_1.fq \`
	`sample_1_reads_2.fq \`
	`> aligned_reads.sam`

Step 2	**Mark Duplicates + Sort**
Tool	GATK4 MarkDuplicatesSpark
Input	aligned_reads.sam
Output	sorted_dedup_reads.bam sorted_dedup_reads.bam.bai dedup_metrics.txt
Notes	In GATK4, the Mark Duplicates and Sort Sam steps have been combined into one step using the MarkDuplicatesSpark tool. In addition, the BAM index file (.bai) is created as well by default. The tool is optimized to run on queryname-grouped alignments (that is, all reads with the same queryname are together in the input file). The output of BWA is query-grouped, however if provided coordinate-sorted alignments, the tool will spend additional time first queryname sorting the reads internally. Due to MarkDuplicatesSpark queryname-sorting coordinate-sorted inputs internally at the start, the tool produces identical results regardless of the input sort-order. That is, it will flag duplicates sets that include secondary, and supplementary and unmapped mate records no matter the sort-order of the input. This differs from how Picard MarkDuplicates behaves given the differently sorted inputs. (i.e. coordinate sorted vs queryname sorted).

	however many cores are available on the machine — or however many you choose to allocate. (https://gatk.broadinstitute.org/hc/en-us/articles/360035890591?id=11245)

Command	`gatk \`
	`MarkDuplicatesSpark \`
	`-I aligned_reads.sam \`
	`-M dedup_metrics.txt \`
	`-O sorted_dedup_reads.bam`

Step 3	**Collect Alignment & Insert Size Metrics**
Tool	Picard Tools, R, Samtools
Input	sorted_dedup_reads.bam reference genome
Output	alignment_metrics.txt, insert_metrics.txt, insert_size_histogram.pdf, depth_out.txt

Command	`java -jar picard.jar \`
	`CollectAlignmentSummaryMetrics \`
	`R=ref.fa \`
	`I=sorted_dedup_reads.bam \`
	`O=alignment_metrics.txt`
	
	`java -jar picard.jar \`
	`CollectInsertSizeMetrics \`
	`INPUT=sorted_dedup_reads.bam \`
	`OUTPUT=insert_metrics.txt \`
	`HISTOGRAM_FILE=insert_size_histogram.pdf`
	
	`samtools depth -a sorted_dedup_reads.bam > depth_out.txt`

Tool	GATK4
Input	sorted_dedup_reads.bam reference genome
Output	raw_variants.vcf
Notes	First round of variant calling. The variants identified in this step will be filtered and provided as input for Base Quality Score Recalibration (BQSR)

Command	`gatk HaplotypeCaller \`
	`-R ref.fa \`
	`-I sorted_dedup_reads.bam \`
	`-o raw_variants.vcf`

Step 5	**Extract SNPs & Indels**
Tool	GATK4
Input	raw_variants.vcf reference genome
Output	raw_indels.vcf raw_snps.vcf
Notes	This step separates SNPs and Indels so they can be processed and used independently

Command	`gatk SelectVariants \`
	`-R ref.fa \`
	`-V raw_variants.vcf \`
	`-selectType SNP \`
	`-o raw_snps.vcf`
	`gatk SelectVariants \`
	`-R ref.fa \`
	`-V raw_variants.vcf \`
	`-selectType INDEL \`

Tool	GATK4
Input	raw_snps.vcf reference genome
Output	filtered_snps.vcf filtered_snps.vcf.idx
Notes	The filters below are a good starting point provided by the Broad. You will need to experiment a little to find the values that are appropriate for your data, and to produce the tradeoff between sensitivity and specificity that is right for your analysis. QD < 2.0: This is the variant confidence (from the QUAL field) divided by the unfiltered depth of non-hom-ref samples. This annotation is intended to normalize the variant quality in order to avoid inflation caused when there is deep coverage. For filtering purposes it is better to use QD than either QUAL or DP directly. FS > 60.0: This is the Phred-scaled probability that there is strand bias at the site. Strand Bias tells us whether the alternate allele was seen more or less often on the forward or reverse strand than the reference allele. When there is little to no strand bias at the site, the FS value will be close to 0. MQ < 40.0: This is the root mean square mapping quality over all the reads at the site. Instead of the average mapping quality of the site, this annotation gives the square root of the average of the squares of the mapping qualities at the site. It is meant to include the standard deviation of the mapping qualities. Including the standard deviation allows us to include the variation in the dataset. A low standard deviation means the values are all close to the mean, whereas a high standard deviation means the values are all far from the mean.When the mapping qualities are good at a site, the MQ will be around 60. SOR > 4.0: This is another way to estimate strand bias using a test similar to the symmetric odds ratio test. SOR was created because FS tends to penalize variants that occur at the ends of exons. Reads at the ends of exons tend to only be covered by reads in one direction and FS gives those variants a bad score. SOR will take into account the

reference allele and the alternate allele.

ReadPosRankSum < -8.0: Compares whether the positions of the reference and alternate alleles are different within the reads. Seeing an allele only near the ends of reads is indicative of error, because that is where sequencers tend to make the most errors.

Learn more about hard filtering: https://gatk.broadinstitute.org/hc/en-us/articles/360035890471-Hard-filtering-germline-short-variants

Note: SNPs which are 'filtered out' at this step will remain in the filtered_snps.vcf file, however they will be marked as '_filter', while SNPs which passed the filter will be marked as 'PASS'. We need to extract and provide only the passing SNPs to the BQSR tool, we do this in the next step (step 9).

Command	`gatk VariantFiltration \`
	`-R ref.fa \`
	`-V raw_snps.vcf \`
	`-O filtered_snps.vcf \`
	`-filter-name "QD_filter" -filter "QD < 2.0" \`
	`-filter-name "FS_filter" -filter "FS > 60.0" \`
	`-filter-name "MQ_filter" -filter "MQ < 40.0" \`
	`-filter-name "SOR_filter" -filter "SOR > 4.0" \`
	`-filter-name "MQRankSum_filter" -filter "MQRankSum < -12.5" \`
	`-filter-name "ReadPosRankSum_filter" -filter "ReadPosRankSum < -8.0"`

Step 7	**Filter Indels**
Tool	GATK4
Input	raw_indels.vcf reference genome
Output	filtered_indels.vcf filtered_indels.vcf.idx

the tradeoff between sensitivity and specificity that is right for your analysis.

QD < 2.0: This is the variant confidence (from the QUAL field) divided by the unfiltered depth of non-hom-ref samples. This annotation is intended to normalize the variant quality in order to avoid inflation caused when there is deep coverage. For filtering purposes it is better to use QD than either QUAL or DP directly.

FS > 200.0: This is the Phred-scaled probability that there is strand bias at the site. Strand Bias tells us whether the alternate allele was seen more or less often on the forward or reverse strand than the reference allele. When there is little to no strand bias at the site, the FS value will be close to 0.

SOR > 10.0: This is another way to estimate strand bias using a test similar to the symmetric odds ratio test. SOR was created because FS tends to penalize variants that occur at the ends of exons. Reads at the ends of exons tend to only be covered by reads in one direction and FS gives those variants a bad score. SOR will take into account the ratios of reads that cover both alleles.

Learn more about hard filtering: https://gatk.broadinstitute.org/hc/en-us/articles/360035890471-Hard-filtering-germline-short-variants

Note: Indels which are 'filtered out' at this step will remain in the filtered_snps.vcf file, however they will be marked as '_filter', while SNPs which passed the filter will be marked as 'PASS'. We need to extract and provide only the passing indels to the BQSR tool, we do this next.

Command

```
gatk VariantFiltration \
        -R ref.fa \
        -V raw_indels.vcf \
        -O filtered_indels.vcf \
        -filter-name "QD_filter" -filter "QD < 2.0" \
        -filter-name "FS_filter" -filter "FS > 200.0" \
        -filter-name "SOR_filter" -filter "SOR > 10.0"
```

Step 8 **Exclude Filtered Variants**

Input	filtered_snps.vcf filtered_indels.vcf
Output	bqsr_snps.vcf bqsr_indels.vcf
Notes	We need to extract only the passing variants and provide this as input to BQSR (next step).

Command	`gatk SelectVariants \`
	`--exclude-filtered \`
	`-V filtered_snps.vcf \`
	`-O bqsr_snps.vcf`
	`gatk SelectVariants \`
	`--exclude-filtered \`
	`-V filtered_indels.vcf \`
	`-O bqsr_indels.vcf`

Step 9	**Base Quality Score Recalibration (BQSR) #1**
Tool	GATK4
Input	sorted_dedup_reads.bam (from step 2) bqsr_snps.vcf bqsr_indels.vcf reference genome
Output	recal_data.table
Notes	BQSR is performed twice. The second pass is optional, only required to produce a recalibration report.

Command	`gatk BaseRecalibrator \`
	`-R ref.fa \`
	`-I sorted_dedup_reads.bam \`
	`--known-sites bqsr_snps.vcf \`

Step 10	**Apply BQSR**
Tool	GATK4
Input	recal_data.table sorted_dedup_reads.bam reference genome
Output	recal_reads.bam
Notes	This step applies the recalibration computed in the first BQSR step to the bam file. This recalibrated bam file is now analysis-ready.

Command	`gatk ApplyBQSR \`
	`-R ref.fa \`
	`-I sorted_dedup_reads.bam \`
	`-bqsr recal_data.table \`
	`-O recal_reads.bam \`

Step 11	**Base Quality Score Recalibration (BQSR) #2**
Tool	GATK4
Input	recal_reads.bam bqsr_snps.vcf bqsr_indels.vcf reference genome
Output	post_recal_data.table
Notes	This round of BQSR is optional. It is required if you want to produce a recalibration report with the Analyze Covariates step (next). For this round of BQSR, you provide the recalibrated reads obtained from the Apply BQSR step above as input.
Command	`gatk BaseRecalibrator \`

```
    --known-sites bqsr_indels.vcf \
    -O post_recal_data.table
```

Step 12	**Analyze Covariates**
Tool	GATK4
Input	recal_data.table post_recal_data.table
Output	recalibration_plots.pdf
Notes	This step produces a recalibration report based on the output from the two BQSR runs

Command	`gatk AnalyzeCovariates \`
	`-before recal_data.table \`
	`-after post_recal_data.table \`
	`-plots recalibration_plots.pdf`

Step 13	**Call Variants**
Tool	GATK4
Input	recal_reads.bam reference genome
Output	raw_variants_recal.vcf
Notes	Second round of variant calling performed using recalibrated (analysis-ready) bam

Command	`gatk HaplotypeCaller \`
	`-R ref.fa \`
	`-I recal_reads.bam \`
	`-o raw_variants_recal.vcf`

Tool	GATK4
Input	raw_variants_recal.vcf reference genome
Output	raw_indels_recal.vcfraw_snps_recal.vcf
Notes	This step separates SNPs and Indels so they can be processed and analyzed independently

Command	`gatk SelectVariants \`
	`-R ref.fa \`
	`-V raw_variants_recal.vcf \`
	`-selectType SNP \`
	`-o raw_snps_recal.vcf`
	`gatk SelectVariants \`
	`-R ref.fa \`
	`-V raw_variants.vcf \`
	`-selectType INDEL \`
	`-o raw_indels_recal.vcf`

Step 15	**Filter SNPs**
Tool	GATK4
Input	raw_snps_recal.vcf reference genome
Output	filtered_snps_final.vcf filtered_snps_final.vcf.idx
Notes	The filters below are a good starting point provided by the Broad. You will need to experiment a little to find the values that are appropriate for your data, and to produce the tradeoff between sensitivity and specificity that is right for your analysis. QD < 2.0: This is the variant confidence (from the QUAL field) divided by the unfiltered

FS > 60.0: This is the Phred-scaled probability that there is strand bias at the site. Strand Bias tells us whether the alternate allele was seen more or less often on the forward or reverse strand than the reference allele. When there is little to no strand bias at the site, the FS value will be close to 0.

MQ < 40.0: This is the root mean square mapping quality over all the reads at the site. Instead of the average mapping quality of the site, this annotation gives the square root of the average of the squares of the mapping qualities at the site. It is meant to include the standard deviation of the mapping qualities. Including the standard deviation allows us to include the variation in the dataset. A low standard deviation means the values are all close to the mean, whereas a high standard deviation means the values are all far from the mean.When the mapping qualities are good at a site, the MQ will be around 60.

SOR > 4.0: This is another way to estimate strand bias using a test similar to the symmetric odds ratio test. SOR was created because FS tends to penalize variants that occur at the ends of exons. Reads at the ends of exons tend to only be covered by reads in one direction and FS gives those variants a bad score. SOR will take into account the ratios of reads that cover both alleles.

MQRankSum < -8.0: Compares the mapping qualities of the reads supporting the reference allele and the alternate allele.

ReadPosRankSum < -8.0: Compares whether the positions of the reference and alternate alleles are different within the reads. Seeing an allele only near the ends of reads is indicative of error, because that is where sequencers tend to make the most errors.

Learn more about hard filtering: https://gatk.broadinstitute.org/hc/en-us/articles/360035890471-Hard-filtering-germline-short-variants

Note: SNPs which are 'filtered out' at this step will remain in the filtered_snps.vcf file, however they will be marked as '_filter', while SNPs which passed the filter will be marked as 'PASS'.

Command

```
gatk VariantFiltration \
```

```
            -filter-name "QD_filter" -filter "QD < 2.0" \
            -filter-name "FS_filter" -filter "FS > 60.0" \
            -filter-name "MQ_filter" -filter "MQ < 40.0" \
            -filter-name "SOR_filter" -filter "SOR > 4.0" \
            -filter-name "MQRankSum_filter" -filter "MQRankSum < -12.5" \
            -filter-name "ReadPosRankSum_filter" -filter "ReadPosRankSum <
-8.0"
```

Step 16	**Filter Indels**
Tool	GATK4
Input	raw_indels_recal.vcf reference genome
Output	filtered_indels_final.vcf filtered_indels_final.vcf.idx
Notes	The filters below are a good starting point provided by the Broad. You will need to experiment a little to find the values that are appropriate for your data, and to produce the tradeoff between sensitivity and specificity that is right for your analysis. QD < 2.0: This is the variant confidence (from the QUAL field) divided by the unfiltered depth of non-hom-ref samples. This annotation is intended to normalize the variant quality in order to avoid inflation caused when there is deep coverage. For filtering purposes it is better to use QD than either QUAL or DP directly. FS > 200.0: This is the Phred-scaled probability that there is strand bias at the site. Strand Bias tells us whether the alternate allele was seen more or less often on the forward or reverse strand than the reference allele. When there is little to no strand bias at the site, the FS value will be close to 0. SOR > 10.0: This is another way to estimate strand bias using a test similar to the symmetric odds ratio test. SOR was created because FS tends to penalize variants that occur at the ends of exons. Reads at the ends of exons tend to only be covered by reads

	Learn more about hard filtering: https://gatk.broadinstitute.org/hc/en-us/articles/360035890471-Hard-filtering-germline-short-variants Note: Indels which are 'filtered out' at this step will remain in the filtered_snps.vcf file, however they will be marked as '_filter', while SNPs which passed the filter will be marked as 'PASS'.

Command	`gatk VariantFiltration \`
	`-R ref.fa \`
	`-V raw_indels_recal.fa \`
	`-O filtered_indels_final.vcf \`
	`-filter-name "QD_filter" -filter "QD < 2.0" \`
	`-filter-name "FS_filter" -filter "FS > 200.0" \`
	`-filter-name "SOR_filter" -filter "SOR > 10.0"`

Step 17	**Annotate SNPs and Predict Effects**
Tool	SnpEff
Input	filtered_snps_final.vcf
Output	filtered_snps_final.ann.vcf snpeff_summary.html snpEff_genes.txt

Command	`java -jar snpEff.jar -v \`
	`<snpeff_db> \`
	`filtered_snps_final.vcf > $filtered_snps_final.ann.vcf`

Step 18	**Compile Statistics**
Tool	parse_metrics.sh (in `/bin`)
Input	sample_id_alignment_metrics.txt sample_id_insert_metrics.txt

	sample_id_raw_snps_recal.vcf sample_id_filtered_snps_final.vcf sample_id_depth_out.txt
Output	report.csv
Command	`parse_metrics.sh sample_id > sample_id_report.csv`
Notes	A single report file is generated with summary statistics for each library processed containing the following pieces of information: # of Reads # of Aligned Reads % Aligned # Aligned Bases Read Length % Paired % Duplicate Mean Insert Size # SNPs, # Filtered SNPs # SNPs after BQSR, # Filtered SNPs after BQSR Average Coverage Give the script a sample id and it will look for the corresponding `sample_id_*` files (listed in the input section above) and print the statistics and header to stdout.

Categories: **BIOINFORMATICS**

Tags: gatk gatk4 nextflow variant calling

26 Comments

Eric Borenstein · 2020-04-03 at 1:34 pm

This workflow can also be ran as an SBATCH rather than interactively. The SBATCH options to change would be job-name, output, and possibly time. The resources set in SBATCH are only for the job controller nextflow and not the actual compute, so no need to increase. The resources for your compute would be set in the config file given. Job process can be monitored in the slurm output file you set.

#!/bin/bash

#SBATCH –nodes=1
#SBATCH –ntasks-per-node=1
#SBATCH –cpus-per-task=2
#SBATCH –time=5:00:00
#SBATCH –mem=2GB
#SBATCH –job-name=myJob
#SBATCH –output=slurm_%j.out
module purge
module load nextflow/20.10.0

nextflow run main.nf -c your.config

REPLY

MARION · 2020-05-01 at 10:18 am

Thank you so much for this elaborate workflow Mohammed. However I am continuously getting this error nextflow.exception.AbortOperationException: Not a valid project name: main.nf when I run nextflow run main.nf. Thank you in advance!

REPLY

Mohammed Khalfan · 2020-05-04 at 6:03 pm

This error means there is no main.nf file in your working directory. You should change into the directory where the main.nf script is located and then run nextflow run main.nf.

REPLY

written.

Does NYU HPC let you use Docker?

REPLY

Mohammed Khalfan · 2020-05-04 at 6:05 pm

Thank you Darach, I'm really liking nextflow. There are a couple other workflows on our git too. Have you had the opportunity to test out the modules feature? I'm interested to try that next.

We use Singularity on the HPC which supports the use of Docker containers.

REPLY

Darach · 2020-05-04 at 7:30 pm

No, I haven't seen modules yet. It looks like a really smart idea. I look forward to what y'all write in that; I know that'll be really cleanly done.

Ah, so it makes sense to write it as a Docker recipe, because then that's more flexible for users in both platforms. Presumably folks using cloud-computing will appreciate that.

I'm going to copy the way you're writing channel operators in the actual input field. I hadn't thought to do that, and I think that style is a much cleaner way of organizing nextflow. Thanks for sharing your code with everyone, for direct use and inspiration.

REPLY

Mohammed Khalfan · 2020-05-06 at 10:22 pm

Thank you for the kind words Darach, I value your feedback.

REPLY

Marion · 2020-05-07 at 1:27 pm

Hi Mohammed, I got an error at the markDuplicatesSpark step caused by: java.lang.IllegalStateException: Duplicate key A00431:16:H5JTVDSXX:1:1101:27308:32972 (attempted merging values -1 and -1). I wonder what it is all about, I have tried to figure out how to go about this and failed. Thank you in advance.

REPLY

Marion · 2020-05-08 at 9:49 am

Yoshi · 2020-07-08 at 12:52 am

Hi, Mohammed. Thank you for the great pipeline.
I would like to use this pipeline for strain tracking of pathogenic bacteria in our society based on SNVs, and I have one question.

How can we mask (exclude) some genomic regions from the reference genome ? To go around false-positive SNVs, we usually exclude repetitive region as short reads are not correctly mapped to these regions. We also exclude dense SNVs that are due to incorrect mapping using filtering of window size 12 bp. Can we do this as well ? Many thanks for your kind support.

REPLY

Mohammed Khalfan · 2020-09-10 at 6:39 pm

Hi Yoshi,

You can use provide haplotypecaller with an intervals file with the -L option, and it will work on only the regions defined in this file. See https://gatk.broadinstitute.org/hc/en-us/articles/360035531852-Intervals-and-interval-lists.

REPLY

Diego · 2020-08-25 at 9:35 pm

Hi Mohammed, thanks so much for this comprehensive pipeline. I was wondering, do I need to cat and trim (with trimmomatic) my raw reads prior to starting the pipeline?

REPLY

Mohammed Khalfan · 2020-08-27 at 3:17 pm

Hi Diego, there is no strict rule, but it's probably better to do this.

REPLY

Niyomi · 2020-09-01 at 8:38 pm

Is this pipeline for an individual sample? I don't see any merging steps anywhere. At what point should I merge the samples if I were to use this? BAM files or the VCF's?

REPLY

Mohammed Khalfan · 2020-09-02 at 6:45 pm

to do joint genotyping for multiple samples, the pipeline is a little different. You would need to add the `-ERC GVCF` option to HaplotypeCaller to generate an intermediate GVCF, and then run gatk GenotypeGVCFs using the intermediary GVCFs as input.

REPLY

Niyomi · 2020-09-03 at 9:45 am

Thank you for your reply. Yes, I am familiar with the -ERC GVCF. My question is, do you use this pipeline to process each sample individually or do you merge samples at any point? In other words, do I follow this pipeline 250 times for 250 samples? Thank you in advance.

REPLY

Mohammed Khalfan · 2020-09-10 at 2:35 pm

This pipeline operates on each sample individually, it does not merge samples at any point, so yes you would follow the pipeline 250 times for 250 samples. The nextflow script takes care of this.

REPLY

Julia Xing · 2020-09-10 at 1:16 am

Which steps should I customize if I use single read sequencing instead of paired end run? Thanks a lot!

REPLY

Mohammed Khalfan · 2020-09-10 at 2:38 pm

Primarily the alignment step, and the way input reads are parsed by nextflow. CollectInsertSizeMetrics in the parse metrics step will fail, as will this part of the final QC step.

REPLY

XiaTian · 2020-11-09 at 9:33 pm

Thanks for your sharing. Still I have a basic question. When we use GATK HaplotypeCaller, it costs a lot of time (800M genome, 15* coverage, 5-7days, per sample). Should I set “NUMBERTHREADS” or “Xmx96g”in this step?

Hello, thanks for this pipeline, it's very helpful!
I was able to substitute bwa for minimap2 command, however with minimap2 I need to use picard AddOrReplaceReadGroups command before gatk MarkDuplicatesSpark. Can I add another process in main.nf? I've tried, but no success so far.

REPLY

Mohammed Khalfan · 2020-12-02 at 5:54 pm

Hi Nathalia, you can add another process but you will also need to setup the input channel to send data into that process, and setup the channel to send output from the new process to MarkDuplicatesSpark.

REPLY

zeeshan Fazal · 2020-12-02 at 1:32 pm

Hi Mohammed
I am trying to do variant calling and I have human cell lines. I am really confused that why your pipeline doesn't not use Goldstandard SNPs and dbSNPs like in GATK? And Can i Run this for human data?

REPLY

Mohammed Khalfan · 2020-12-02 at 5:58 pm

Hi Zeehan,

This workflow is adapted for organisms that do not have gold standard SNPs available. If you're working with human data I would check out broads official workflows such as this one: https://github.com/gatk-workflows/gatk4-germline-snps-indels

REPLY

zeeshan Fazal · 2020-12-04 at 3:02 pm

Thanks Mohammed!

REPLY

VIVEK RUHELA · 2020-12-14 at 1:56 am

Hello,
My question is that why the BQSR step is after haplotype variant calling. I think all the BAM

REPLY

Leave a Reply

Name *

Email *

Website

What's on your mind?

Related Posts

Save my name, email, and website in this browser for the next time I comment.

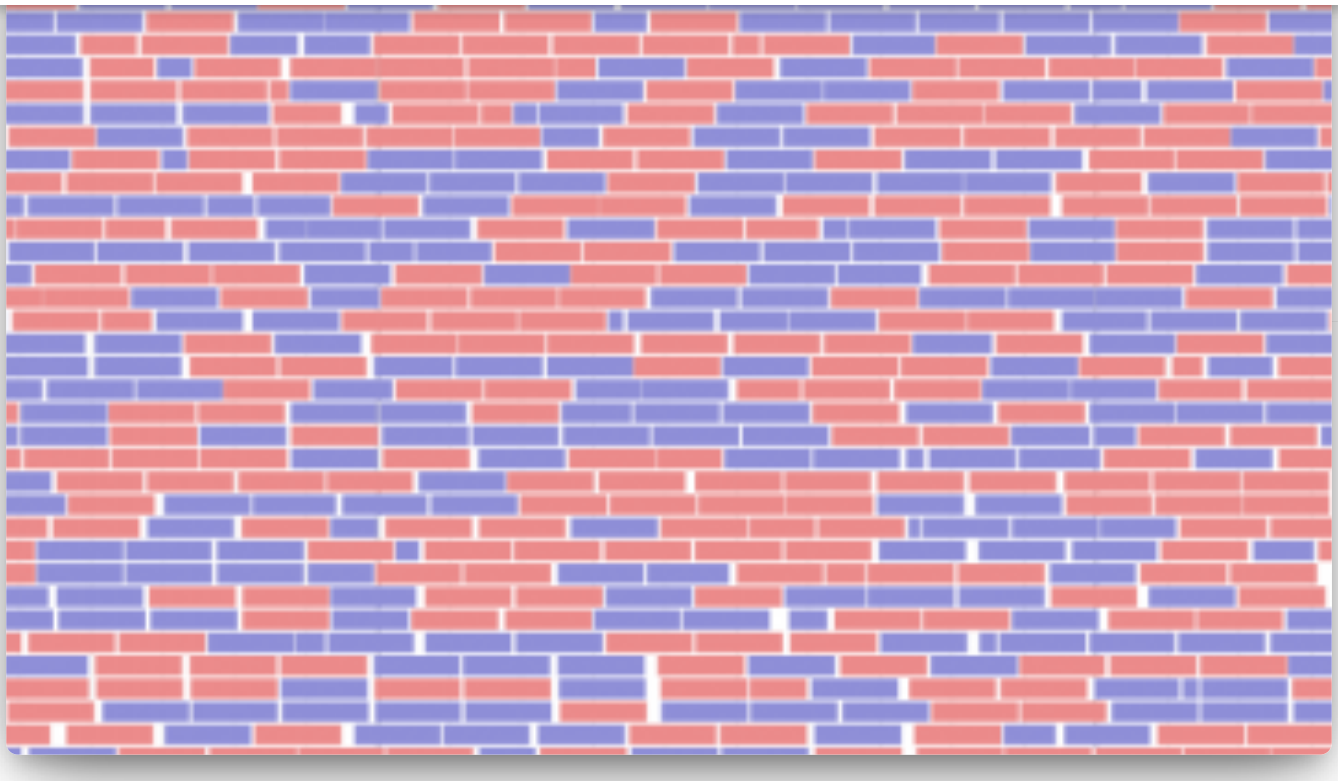
POST COMMENT

BIOINFORMATICS

Streamlined RNA-Seq Analysis Using Nextflow

nf-core is a community effort to collect a curated set of analysis pipelines built using Nextflow. This post will walk you through running the nf-core RNA-Seq workflow. The pipeline uses the STAR aligner by default,

Read more...

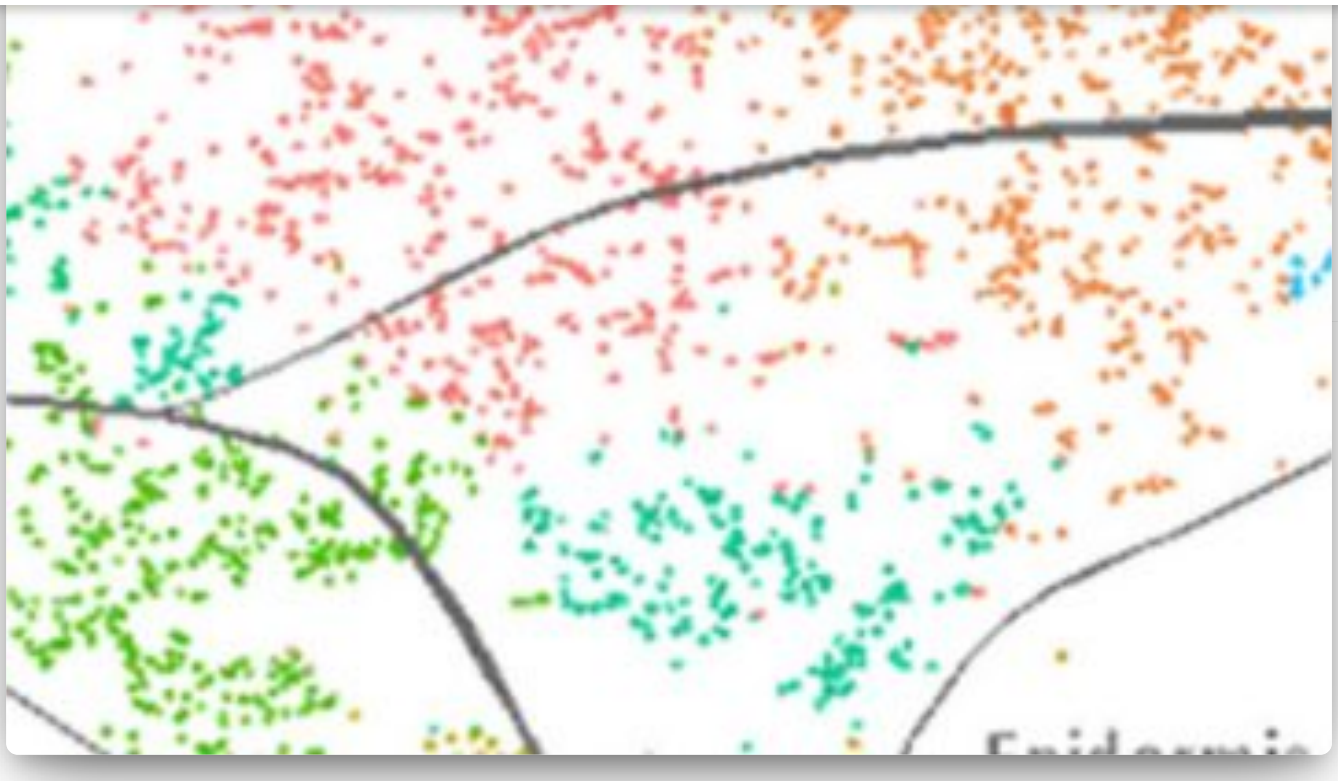


BIOINFORMATICS

JBrowse Genome Browser

During the summer of 2020, the Ghedin and Gresham labs at New York University sequenced several SARS-CoV-2 isolates from clinical samples acquired in New York City. To visualize and share the data among researchers and Read more...

SEQUENCING SHARED EQUIPMENT BIOINFORMATICS BLOG



BIOINFORMATICS

Single Cell RNA-Seq Allows For An Unprecedented Look At Plant Root Meristem Cell Identity

In the Kenneth Birnbaum Lab, we are interested in understanding how the plant root is able to grow continuously over the plant's life and maintain its specific root structure (Fig.1). More specifically, what is the

Read more…