1. GATK4 basics

1.0 Presentation: Introducing GATK4

GATK4 Overview: [https://software.broadinstitute.org/gatk/gatk4](https://software.broadinstitute.org/gatk/gatk4)
Quick Start: [https://software.broadinstitute.org/gatk/documentation/quickstart](https://software.broadinstitute.org/gatk/documentation/quickstart)

Presentation materials and recordings:
[https://software.broadinstitute.org/gatk/documentation/presentations](https://software.broadinstitute.org/gatk/documentation/presentations)

1.1 Exercise: Run GATK and Picard commands in the GATK4 Docker

This exercise covers using Docker, the gatk wrapper, basic GATK4 syntax, JVM options and basic Spark functionality.

**1.1.0 Prerequisite:** as instructed in the pre-workshop instructions, you should have downloaded the data bundle, installed Docker and pulled the official GATK docker using this command:

docker pull broadinstitute/gatk:4.0.5.1

Optionally, increase the memory allocated to Docker.

**1.1.1 Action:** Spin up the container, mounting the location of the data bundle inside the docker

docker run -v /path/gatk_data:/gatk/gatk_data -it broadinstitute/gatk:4.0.5.1

**1.1.2 Action:** List contents of the working directory, then navigate to the gatk_data directory and list its contents. Navigate to the 2-germline directory.

**1.1.3 Action:** Invoke GATK4 by calling the wrapper script: gatk

**1.1.4 Action:** Get usage for a GATK4 tool: gatk HaplotypeCaller --help

**1.1.5 Action:** Run a full GATK4 command

gatk HaplotypeCaller -R ref/ref.fasta -I bams/mother.bam -O sandbox/variants.vcf

**1.1.5 Action:** Add JVM options to the command

gatk --java-options "-Xmx4G" HaplotypeCaller \ 
    -R ref/ref.fasta -I bams/mother.bam -O sandbox/variants.vcf
1.1.6 Action: Run a Picard command

```bash
gatk ValidateSamFile -I bams/mother.bam -MODE SUMMARY
```

1.1.7 Action: Run a command with local Spark multithreading

```bash
gatk --java-options "-Xmx6G" MarkDuplicatesSpark \
    -R ref/ref.fasta \
    -I bams/mother.bam \
    -O sandbox/mother_dedup.bam \
    -M sandbox/metrics.txt \n    -- \
    --spark-master local[*]
```

Optionally, assign a specific number of nodes instead of the asterisk (which requests all available nodes).

1.2 Demo: Run Spark-enabled tools on Google Dataproc

This demo shows how to use a Spark cluster to run Spark-enabled GATK4 tools.

- Google console UI for Dataproc service: https://console.cloud.google.com/
- Select Dataproc in the menu & click "Create a cluster" (pre-staged for demo)
- Set up jar cache to avoid re-uploading the jar file every time:
  ```bash
  export GATK_GCS_STAGING=gs://gatk-jar-cache/
  ```
- Command line:
  ```bash
  gatk MarkDuplicatesSpark \
      -R gs://gatk-workshops/GCCBOSC2018/ref/ref.fasta \
      -I gs://gatk-workshops/GCCBOSC2018/bams/mother.bam \
      -O gs://my-output-data/mother_dedup.bam \
      -M gs://my-output-data/metrics.txt \n      -- \
      --spark-runner GCS \n      --cluster aardvark-01
  ```
2. Best Practices workflows

2.0 Presentation: Overview of the main workflows

Workflow docs: [https://software.broadinstitute.org/gatk/best-practices/](https://software.broadinstitute.org/gatk/best-practices/)

2.1 Exercise: Run GVCF workflow tools

This exercise covers using HaplotypeCaller, GenomicsDBImport and GenotypeGVCFs to perform joint calling on multiple input samples.

**2.1.0 Action:** Run HaplotypeCaller on three input bams (mother, father, son)

```
gatk HaplotypeCaller -R ref/ref.fasta -I bams/mother.bam -O sandbox/mother.g.vcf -ERC GVCF
```

```
gatk HaplotypeCaller -R ref/ref.fasta -I bams/father.bam -O sandbox/father.g.vcf -ERC GVCF
```

```
gatk HaplotypeCaller -R ref/ref.fasta -I bams/son.bam -O sandbox/son.g.vcf -ERC GVCF
```

**2.1.1 Action:** Run GenomicsDBImport on three GVCFs to consolidate

```
gatk GenomicsDBImport -V sandbox/mother.g.vcf -V sandbox/father.g.vcf -V sandbox/son.g.vcf --genomicsdb-workspace-path sandbox/trio.gdb_workspace --intervals 20
```

Note: The tool currently requires a single interval to run on, but multi-interval support will be released in the near future.

**2.1.2 Action:** Run GenotypeGVCFs on the GDB workspace to produce final multisample VCF

```
gatk GenotypeGVCFs -R ref/ref.fasta -V gendb://sandbox/trio.gdb_workspace -G StandardAnnotation -O sandbox/trio_variants.vcf
```

**2.1.3 Action:** Run SelectVariants to extract data from the GDB workspace in GVCF format

```
gatk SelectVariants -R ref/ref.fasta -V gendb://sandbox/trio.gdb_workspace -O sandbox/trio_combined.g.vcf
```

Note: This is an optional step that can be used for troubleshooting.
3. Reference implementations in WDL

3.0 Presentation: Overview of WDL and related resources

Workflows repository in Github: https://github.com/gatk-workflows/
Workflows in Dockstore: https://dockstore.org/search?descriptorType=wdl&search=gatk

OpenWDL: http://openwdl.org/ and https://github.com/openwdl/wdl
WDL support: https://gatkforums.broadinstitute.org/wdl/
Cromwell: https://github.com/broadinstitute/cromwell
FireCloud: https://www.firecloud.org/

3.1 Demo: Run a GATK pipeline written in WDL locally

This demo shows how to use Cromwell to run a GATK pipeline written in WDL.

- Navigate to 4-pipelining directory
- Open and examine example WDL scripts in hello_world and hello_gatk
- Run Cromwell command to run hello_gatk.wdl:
  
  ```
  java -jar jars/cromwell-30.2.jar run hello_gatk/hello_gatk.wdl \
  -i hello_gatk/hello_gatk.inputs.json
  ```

  See the other materials in this directory for additional exercises.

3.2 Demo: Run a full GATK pipeline in a preloaded FireCloud workspace

This demo shows how to run the complete GATK reference implementations in FireCloud.

- Go to https://portal.firecloud.org/#workspaces/help-gatk/five-dollar-genome-analysis-pipeline
- Clone the workspace
- Open Method Configurations > Launch …
- Select example data and launch