Day 2, AM Session

From raw reads to filtered variants

Richard Flamio Jr., Ph.D.

Madison Zimmerman

Kristina M. Ramstad, Ph.D.

University of South Carolina Aiken

Addition on yesterday’s discussion and downloading script for today’s lesson

GitHub

- Online service for software development and code storage

- Navigate to https://github.com/
- Find rflamio
- Repositories > sockeye_GWAS
  - README.md files provide instructions to the reader about repository contents
  - Can include R Markdowns, scripts, datasets (including intermediate files), etc.

GitHub and R Markdown

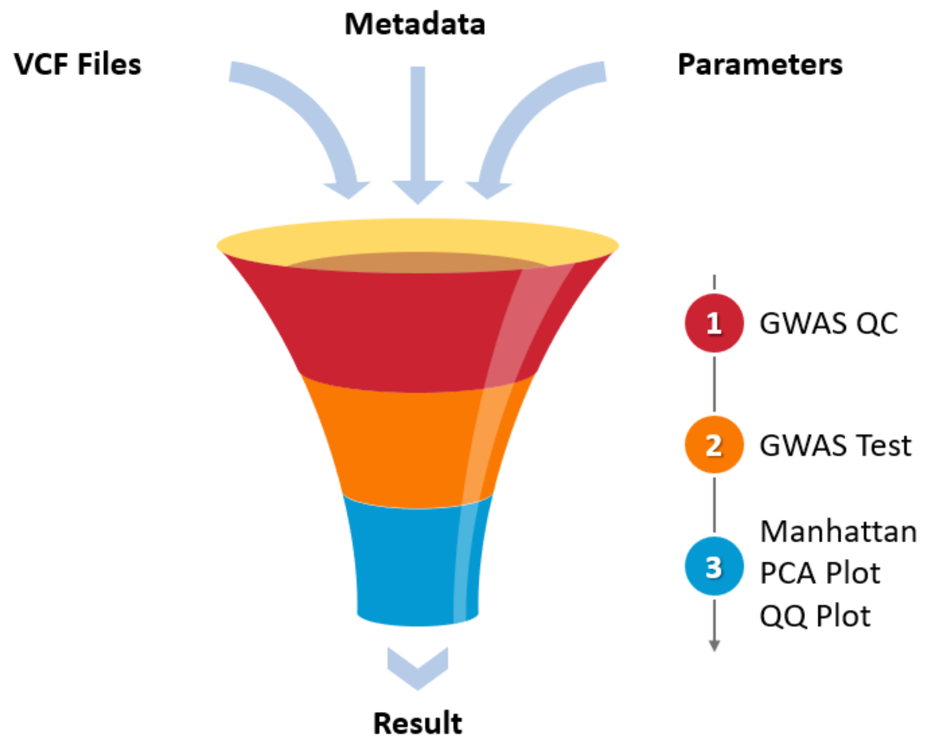
- Download ‘SockeyeVariantCalling.Rmd’ and open in R Studio
- Observe prelude, non-chunks, and chunks
- Observe Table of Contents on right-hand side

If you have an R server, you can run R and Bash scripts at the same time without performing the functions in separate windows.

Variant calling and filtration

First, some background

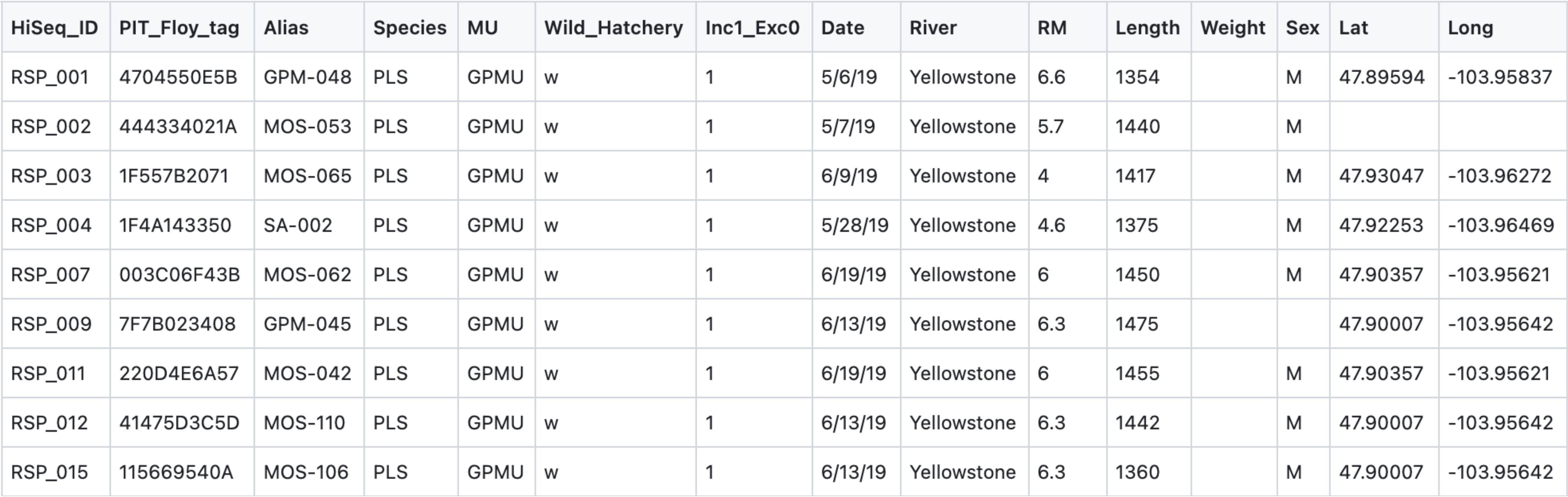
- This course sets the student up for conducting a genome-wide association study (GWAS)
- Retrieving genomic data from many individuals and correlating variation between individuals with a changing phenotypic trait



https://shivom.gitbook.io/documentation/pipelines/gwas-pipeline

Metadata

- Any data associated with an individual
- Sex, age, population, phenotypic trait of interest, etc.
- Usually data tables stored in .csv or .txt file
    - I create mine in Microsoft Excel



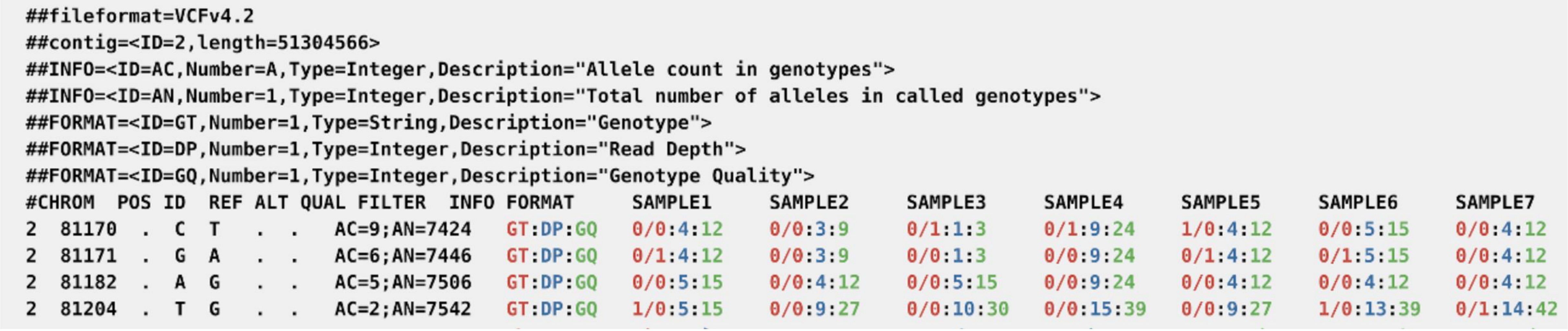
HiSeq_ID	PIT_Floy_tag	Alias	Species	MU	Wild_Hatchery	Inc1_Exc0	Date	River	RM	Length	Weight	Sex	Lat	Long
RSP_001	4704550E5B	GPM-048	PLS	GPMU	w	1	5/6/19	Yellowstone	6.6	1354		M	47.89594	-103.95837
RSP_002	444334021A	MOS-053	PLS	GPMU	w	1	5/7/19	Yellowstone	5.7	1440		M		
RSP_003	1F557B2071	MOS-065	PLS	GPMU	w	1	6/9/19	Yellowstone	4	1417		M	47.93047	-103.96272
RSP_004	1F4A143350	SA-002	PLS	GPMU	w	1	5/28/19	Yellowstone	4.6	1375		M	47.92253	-103.96469
RSP_007	003C06F43B	MOS-062	PLS	GPMU	w	1	6/19/19	Yellowstone	6	1450		M	47.90357	-103.95621
RSP_009	7F7B023408	GPM-045	PLS	GPMU	w	1	6/13/19	Yellowstone	6.3	1475			47.90007	-103.95642
RSP_011	220D4E6A57	MOS-042	PLS	GPMU	w	1	6/19/19	Yellowstone	6	1455		M	47.90357	-103.95621
RSP_012	41475D3C5D	MOS-110	PLS	GPMU	w	1	6/13/19	Yellowstone	6.3	1442		M	47.90007	-103.95642
RSP_015	115669540A	MOS-106	PLS	GPMU	w	1	6/13/19	Yellowstone	6.3	1360		M	47.90007	-103.95642

Flamio et al. 2022

In this lesson we will focus on producing and filtering VCF files

VCF = Variant Call Format

- Stores variant calls for individuals
- Ideal to have a joint file where many individuals are in the same file with their corresponding variants

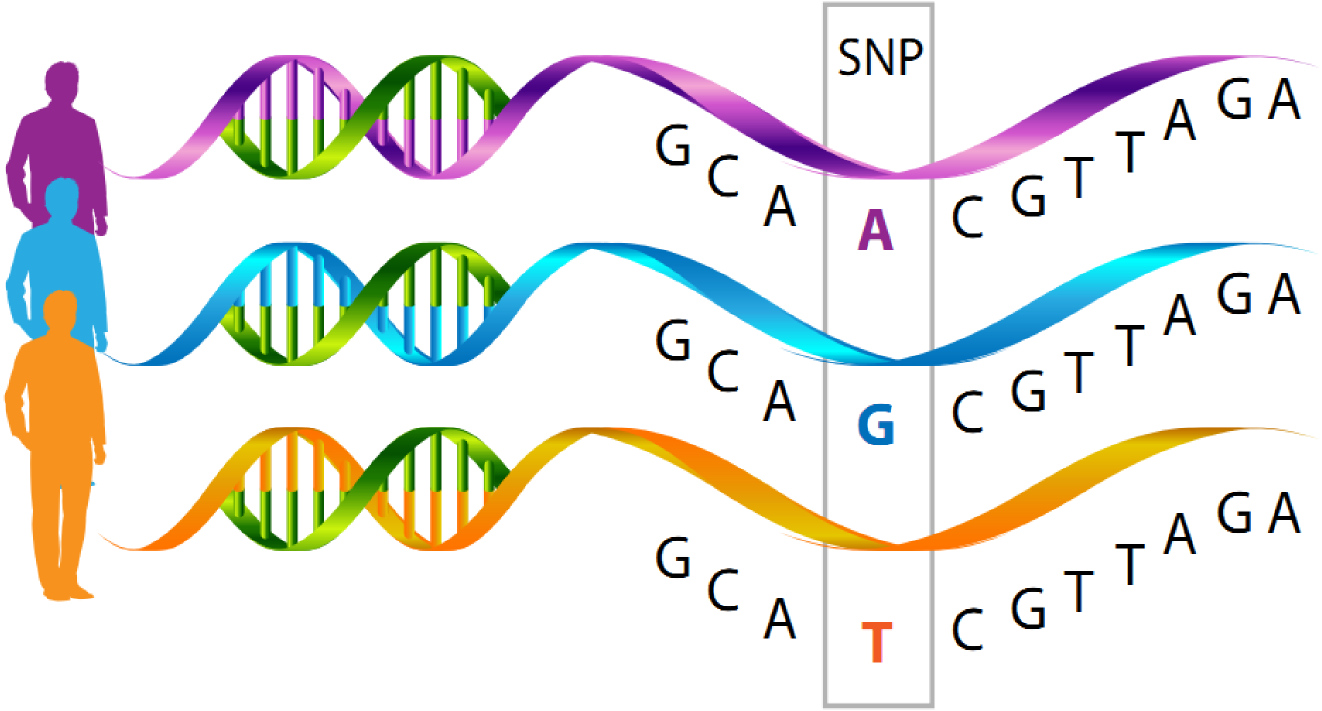


```
##fileformat=VCFv4.2
##contig=<ID=2,length=51304566>
##INFO=<ID=AC,Number=A,Type=Integer,Description="Allele count in genotypes">
##INFO=<ID=AN,Number=1,Type=Integer,Description="Total number of alleles in called genotypes">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
#CHROM  POS ID  REF ALT QUAL FILTER  INFO FORMAT     SAMPLE1     SAMPLE2     SAMPLE3     SAMPLE4     SAMPLE5     SAMPLE6     SAMPLE7
2  81170  .  C  T   .  .   AC=9;AN=7424  GT:DP:GQ  0/0:4:12    0/0:3:9     0/1:1:3     0/1:9:24    1/0:4:12    0/0:5:15    0/0:4:12
2  81171  .  G  A   .  .   AC=6;AN=7446  GT:DP:GQ  0/1:4:12    0/0:3:9     0/0:1:3     0/0:9:24    0/1:4:12    0/1:5:15    0/0:4:12
2  81182  .  A  G   .  .   AC=5;AN=7506  GT:DP:GQ  0/0:5:15    0/0:4:12    0/0:5:15    0/0:9:24    0/0:4:12    0/0:4:12    0/0:4:12
2  81204  .  T  G   .  .   AC=2;AN=7542  GT:DP:GQ  1/0:5:15    0/0:9:27    0/0:10:30   0/0:15:39   0/0:9:27    1/0:13:39   0/1:14:42
```

https://en.wikipedia.org/wiki/Variant_Call_Format#/media/File:Binary_BCF_versus_VCF_format.png

Single nucleotide polymorphism (SNP)

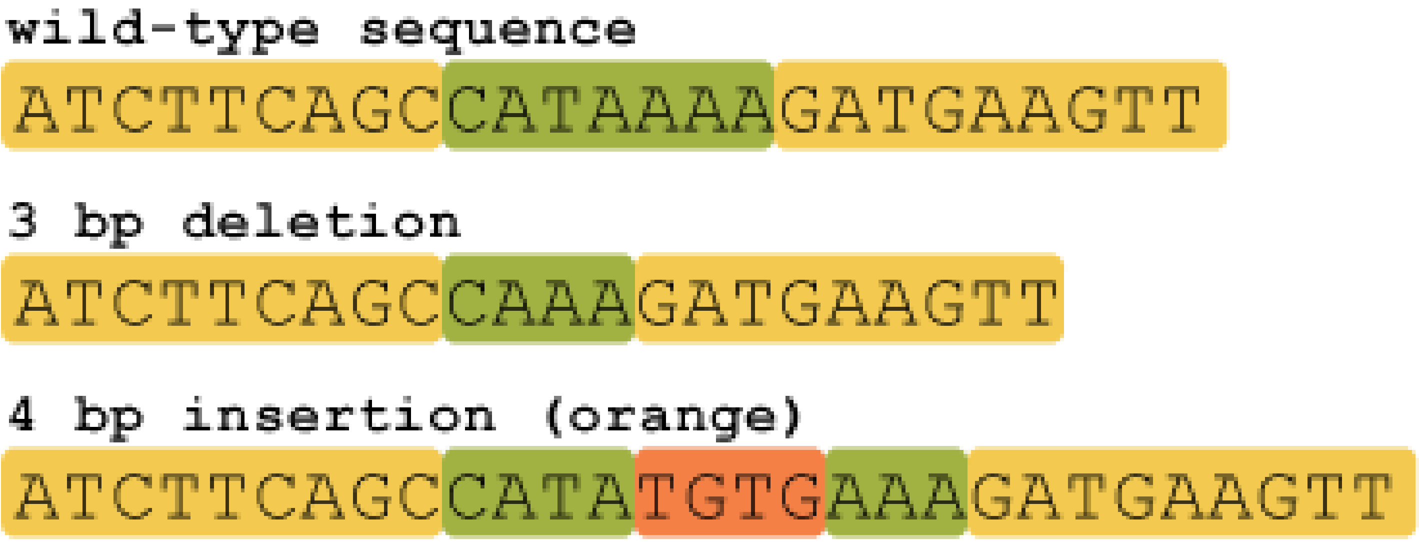
Changes at a specific letter (A,T,C,G) at a position (locus) in the genome



https://www.nutrigeneticsspecialists.com/single-post/2017/03/27/what-is-a-snp

Insertion-deletion mutations (indels)

Insertions or deletion at a locus



```
wild-type sequence
ATCTTCAGCCATAAAAGATGAAGTT

3 bp deletion
ATCTTCAGCCAAAGATGAAGTT

4 bp insertion (orange)
ATCTTCAGCCATATGTGAAAGATGAAGTT
```

https://hackbrightacademy.com/blog/indel-finder-how-the-python-version-of-this-program-works/

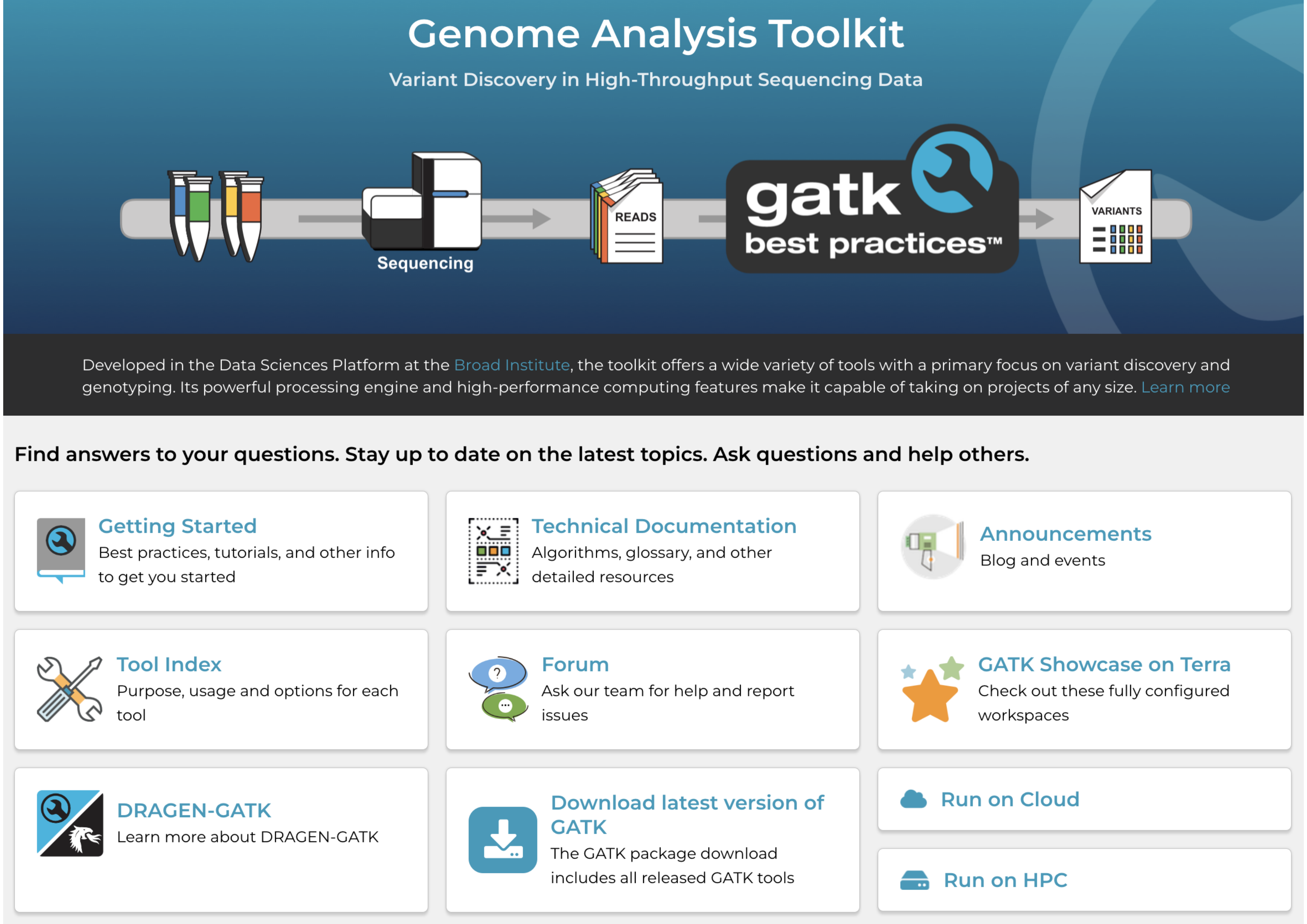
What type of sequencing is useful for GWAS?

- Any sequencing platform that produces data encompassing hundreds of thousands of variants
- Whole-genome coverage versus directed chromosome approach
- Dense genomic sampling and sufficient sample size are key
- Common sequencing platforms are whole genome sequencing (WGS) and SNP microarrays
  - Microarrays are specifically useful for humans where much genomic information is known

So you have sequenced the whole genome of many individuals with whole genome resequencing? How do you produce a VCF file from raw reads?

Genome Analysis Toolkit (GATK)

- A set of tools that allows variant discovery of high-throughput sequencing data
- https://gatk.broadinstitute.org/hc/en-us
- Please navigate through the above site as I explain the next couple of slides



Genome Analysis Toolkit

Variant Discovery in High-Throughput Sequencing Data

Developed in the Data Sciences Platform at the Broad Institute, the toolkit offers a wide variety of tools with a primary focus on variant discovery and genotyping. Its powerful processing engine and high-performance computing features make it capable of taking on projects of any size. Learn more

Find answers to your questions. Stay up to date on the latest topics. Ask questions and help others.

Getting Started
Best practices, tutorials, and other info to get you started

Technical Documentation
Algorithms, glossary, and other detailed resources

Announcements
Blog and events

Tool Index
Purpose, usage and options for each tool

Forum
Ask our team for help and report issues

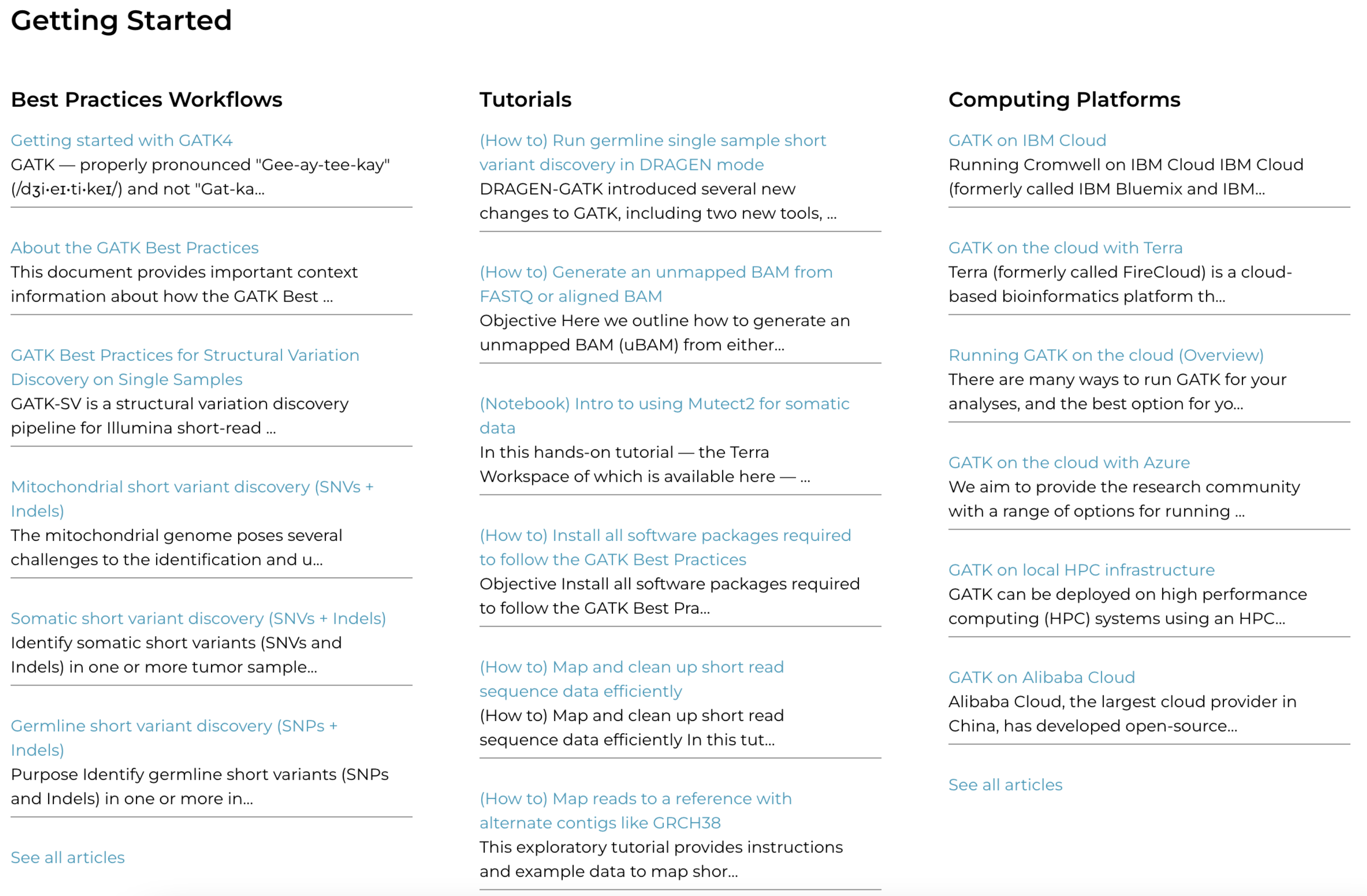
GATK Showcase on Terra
Check out these fully configured workspaces

DRAGEN-GATK
Learn more about DRAGEN-GATK

Download latest version of GATK
The GATK package download includes all released GATK tools

Run on Cloud

Run on HPC



Getting Started

Best Practices Workflows

Getting started with GATK4
GATK — properly pronounced "Gee-ay-tee-kay" (/dʒi·eɪ·ti·keɪ/) and not "Gat-ka...

About the GATK Best Practices
This document provides important context information about how the GATK Best ...

GATK Best Practices for Structural Variation Discovery on Single Samples
GATK-SV is a structural variation discovery pipeline for Illumina short-read ...

Mitochondrial short variant discovery (SNVs + Indels)
The mitochondrial genome poses several challenges to the identification and u...

Somatic short variant discovery (SNVs + Indels)
Identify somatic short variants (SNVs and Indels) in one or more tumor sample...

Germline short variant discovery (SNPs + Indels)
Purpose Identify germline short variants (SNPs and Indels) in one or more in...

See all articles

Tutorials

(How to) Run germline single sample short variant discovery in DRAGEN mode
DRAGEN-GATK introduced several new changes to GATK, including two new tools, ...

(How to) Generate an unmapped BAM from FASTQ or aligned BAM
Objective Here we outline how to generate an unmapped BAM (uBAM) from either...

(Notebook) Intro to using Mutect2 for somatic data
In this hands-on tutorial — the Terra Workspace of which is available here — ...

(How to) Install all software packages required to follow the GATK Best Practices
Objective Install all software packages required to follow the GATK Best Pra...

(How to) Map and clean up short read sequence data efficiently
(How to) Map and clean up short read sequence data efficiently In this tut...

(How to) Map reads to a reference with alternate contigs like GRCH38
This exploratory tutorial provides instructions and example data to map shor...

Computing Platforms

GATK on IBM Cloud
Running Cromwell on IBM Cloud IBM Cloud (formerly called IBM Bluemix and IBM...

GATK on the cloud with Terra
Terra (formerly called FireCloud) is a cloud-based bioinformatics platform th...

Running GATK on the cloud (Overview)
There are many ways to run GATK for your analyses, and the best option for yo...

GATK on the cloud with Azure
We aim to provide the research community with a range of options for running ...

GATK on local HPC infrastructure
GATK can be deployed on high performance computing (HPC) systems using an HPC...

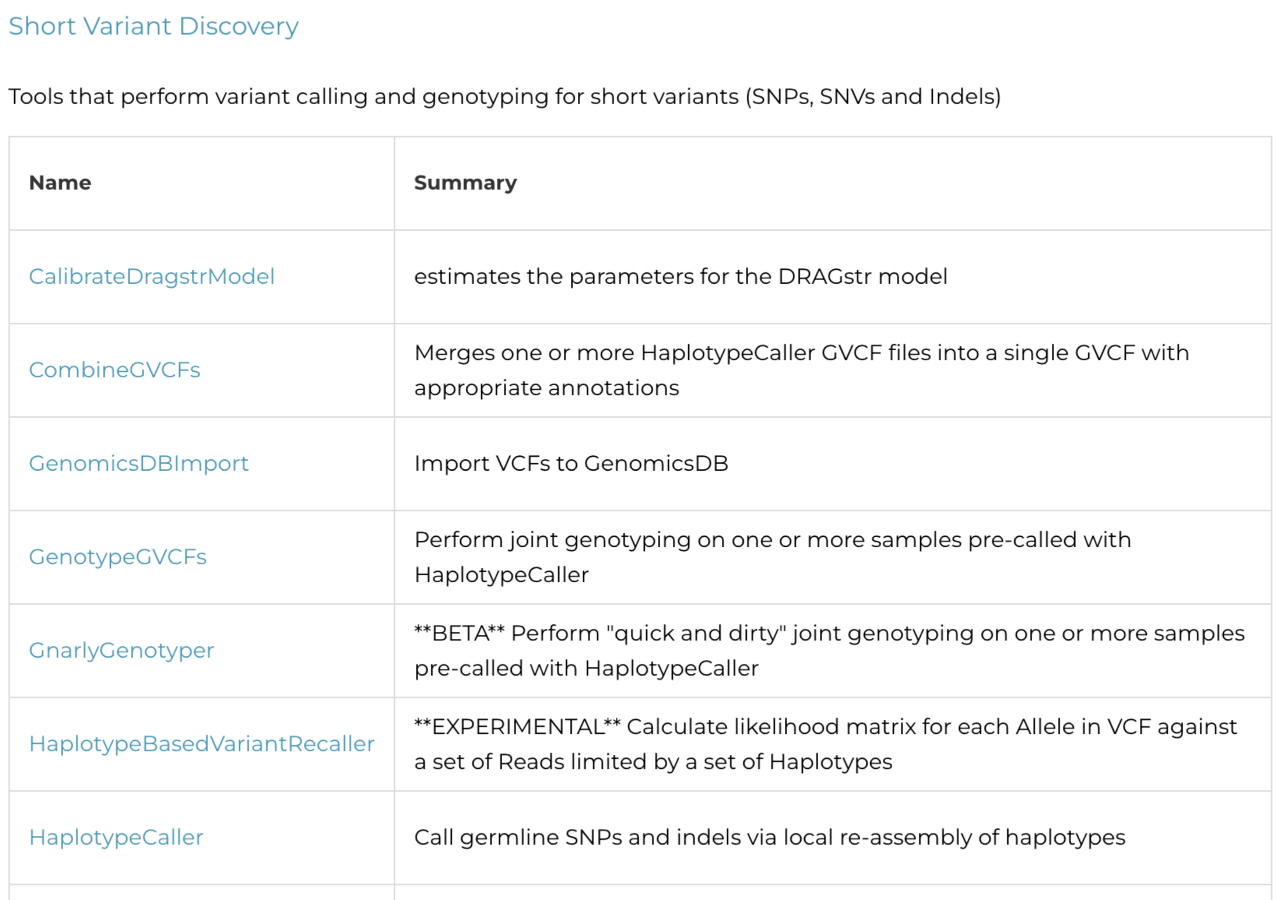
GATK on Alibaba Cloud
Alibaba Cloud, the largest cloud provider in China, has developed open-source...

See all articles

Click on the workflow ‘Germline short variant discovery’ and glance at the sections of the documentation

Tool Index

- First choose tool documentation index for specific version of GATK
- Tool index is sorted by analysis type



Short Variant Discovery

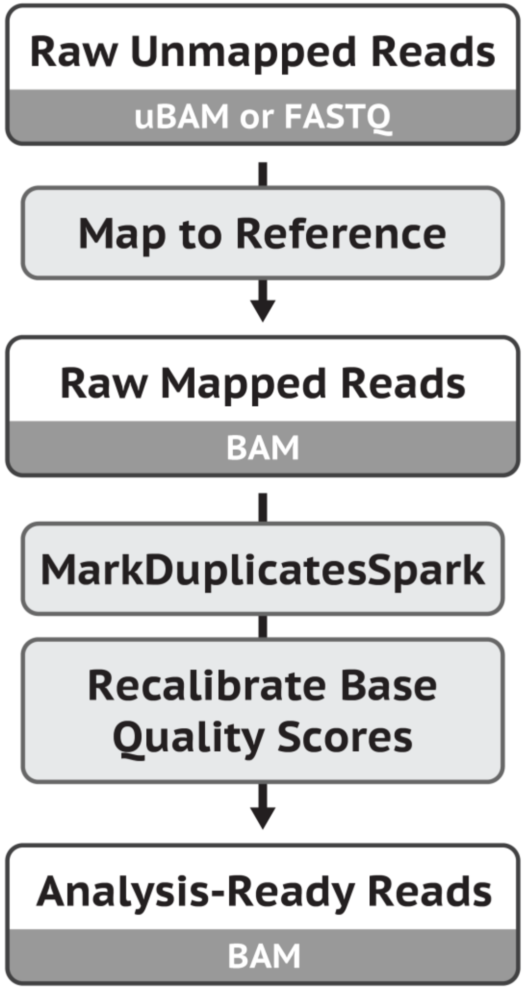
Tools that perform variant calling and genotyping for short variants (SNPs, SNVs and Indels)

Name	Summary
CalibrateDragstrModel	estimates the parameters for the DRAGstr model
CombineGVCFs	Merges one or more HaplotypeCaller GVCF files into a single GVCF with appropriate annotations
GenomicsDBImport	Import VCFs to GenomicsDB
GenotypeGVCFs	Perform joint genotyping on one or more samples pre-called with HaplotypeCaller
GnarlyGenotyper	**BETA** Perform "quick and dirty" joint genotyping on one or more samples pre-called with HaplotypeCaller
HaplotypeBasedVariantRecaller	**EXPERIMENTAL** Calculate likelihood matrix for each Allele in VCF against a set of Reads limited by a set of Haplotypes
HaplotypeCaller	Call germline SNPs and indels via local re-assembly of haplotypes

Click on the HaplotypeCaller tool under tool index 4.4.0.0.

Notice the different sections and arguments.

Data pre-processing for variant discovery



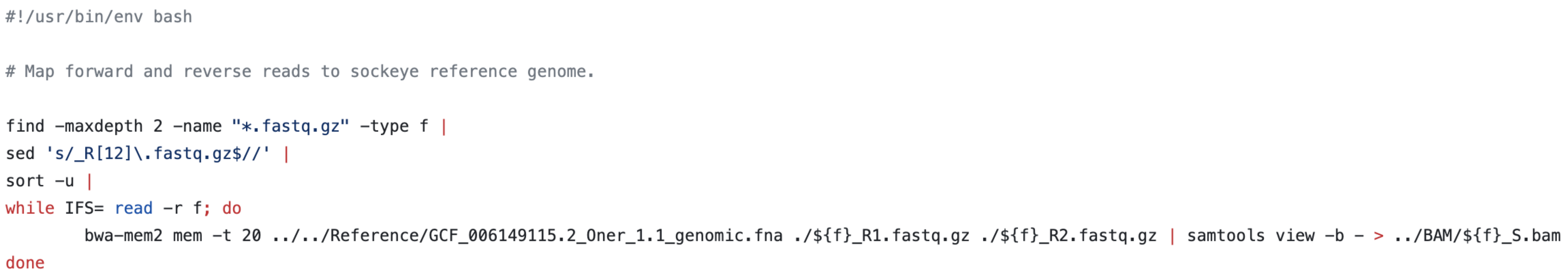
Step 1: Map raw reads to the reference

- Can use the tool bwa-mem2 (https://github.com/bwa-mem2/bwa-mem2)

- 1st step is indexing the reference
  - This allows for rapid search and alignment
  - bwa-mem2 index *referencefile*

- 2nd step is mapping forward and reverse reads to the reference

Mapping script



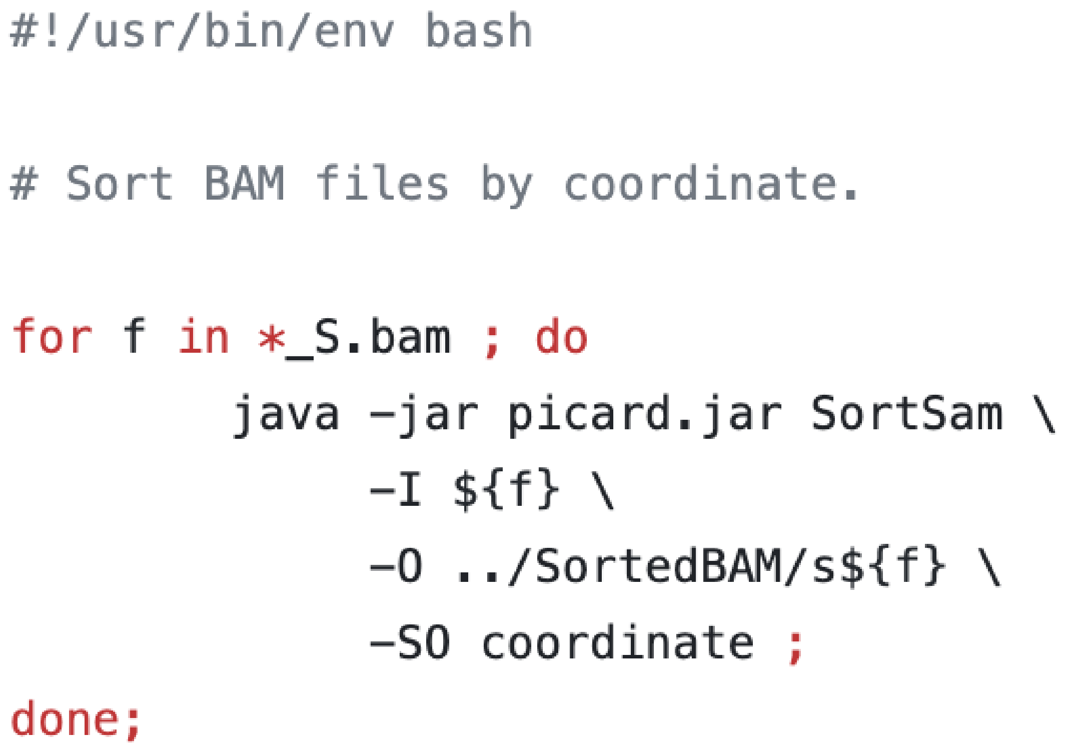
```
#!/usr/bin/env bash

# Map forward and reverse reads to sockeye reference genome.

find -maxdepth 2 -name "*.fastq.gz" -type f |
sed 's/_R[12]\.fastq.gz$//' |
sort -u |
while IFS= read -r f; do
        bwa-mem2 mem -t 20 ../../Reference/GCF_006149115.2_Oner_1.1_genomic.fna ./${f}_R1.fastq.gz ./${f}_R2.fastq.gz | samtools view -b - > ../BAM/${f}_S.bam
done
```

This produces BAM (Binary Alignment/Map) files (*.bam)

- Compressed binary file format of a SAM (Sequence Alignment/Map) file
- Sequence alignment information of reads

Next, we sort each BAM file by coordinate

- This is necessary to perform before the MarkDuplicate step
- This uses the SamSort tool
- I created a custom script using a for loop

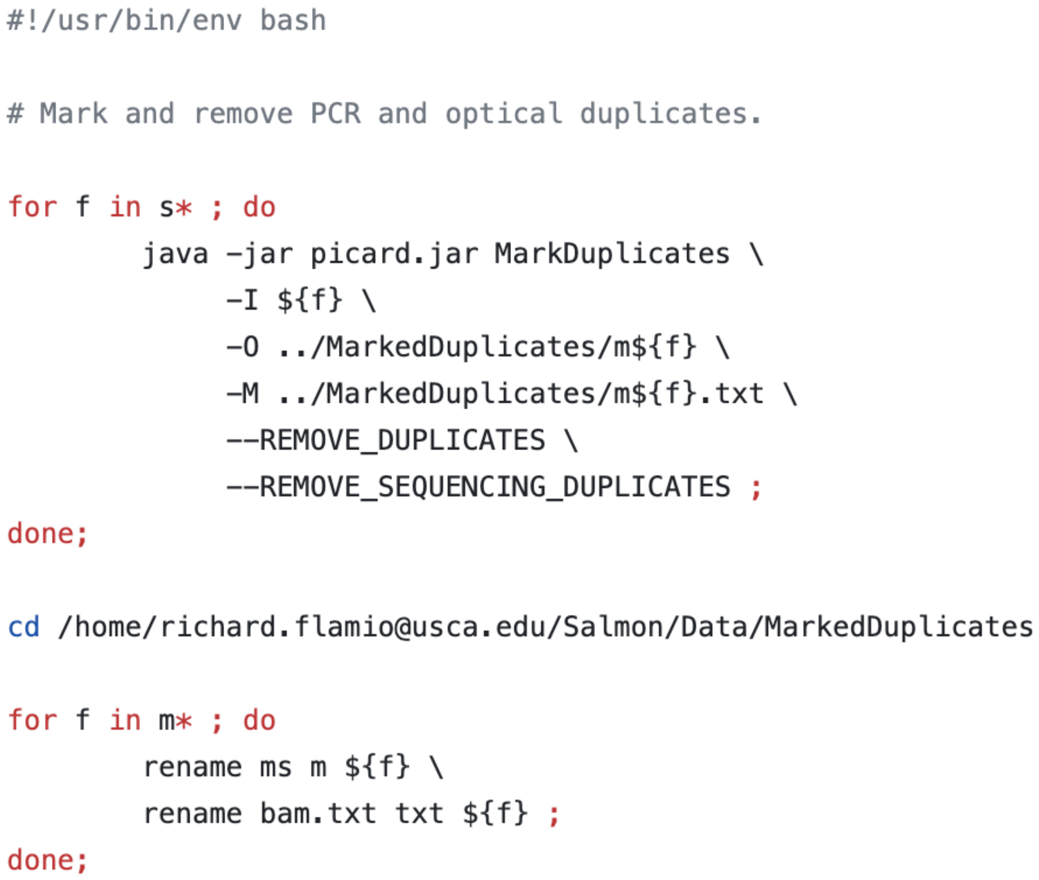


```
#!/usr/bin/env bash

# Sort BAM files by coordinate.

for f in *_S.bam ; do
        java -jar picard.jar SortSam \
             -I ${f} \
             -O ../SortedBAM/s${f} \
             -SO coordinate ;
done;
```

Mark and remove duplicates

- The program MarkDuplicates can be used to identify PCR artifacts and optical duplicates
  - Optical duplicates arise from an error in which a sequencing instrument’s optical sensor detects multiple reads where there should be just one.
- We can use special options to not only identify the duplicates but also remove them.

Mark and remove duplicates



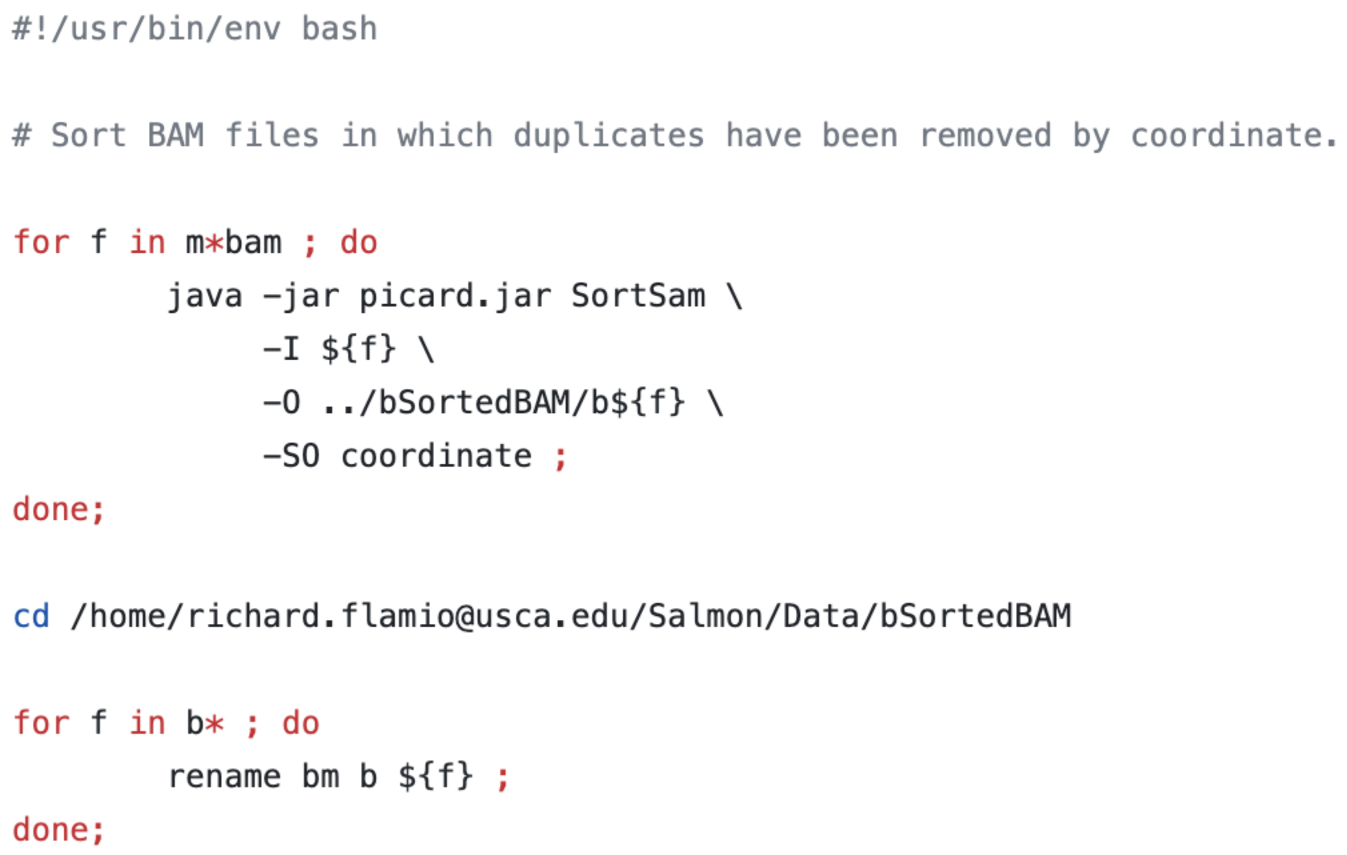
```
#!/usr/bin/env bash

# Mark and remove PCR and optical duplicates.

for f in s* ; do
        java -jar picard.jar MarkDuplicates \
             -I ${f} \
             -O ../MarkedDuplicates/m${f} \
             -M ../MarkedDuplicates/m${f}.txt \
             --REMOVE_DUPLICATES \
             --REMOVE_SEQUENCING_DUPLICATES ;
done;

cd /home/richard.flamio@usca.edu/Salmon/Data/MarkedDuplicates

for f in m* ; do
        rename ms m ${f} \
        rename bam.txt txt ${f} ;
done;
```

Intermediate steps

1. Sort by coordinate again after mark and remove duplicates
2. Fix tags using the Picard tool ‘SetNmMdandUqTags’
3. Add read groups using the tool ‘AddOrReplaceReadGroups’
   - Read groups allow the researcher to differentiate samples and also includes technical information.
   1. RGLB = DNA library preparation identifier (e.g., lib1)
   2. RGPL = Platform technology (e.g., ILLUMINA)
   3. RGPU = Read group platform unit (e.g., unit1)
   4. RGSM = Sample
   5. RGID = Read group identifier

I set the sample and the read group identifier as the same thing.

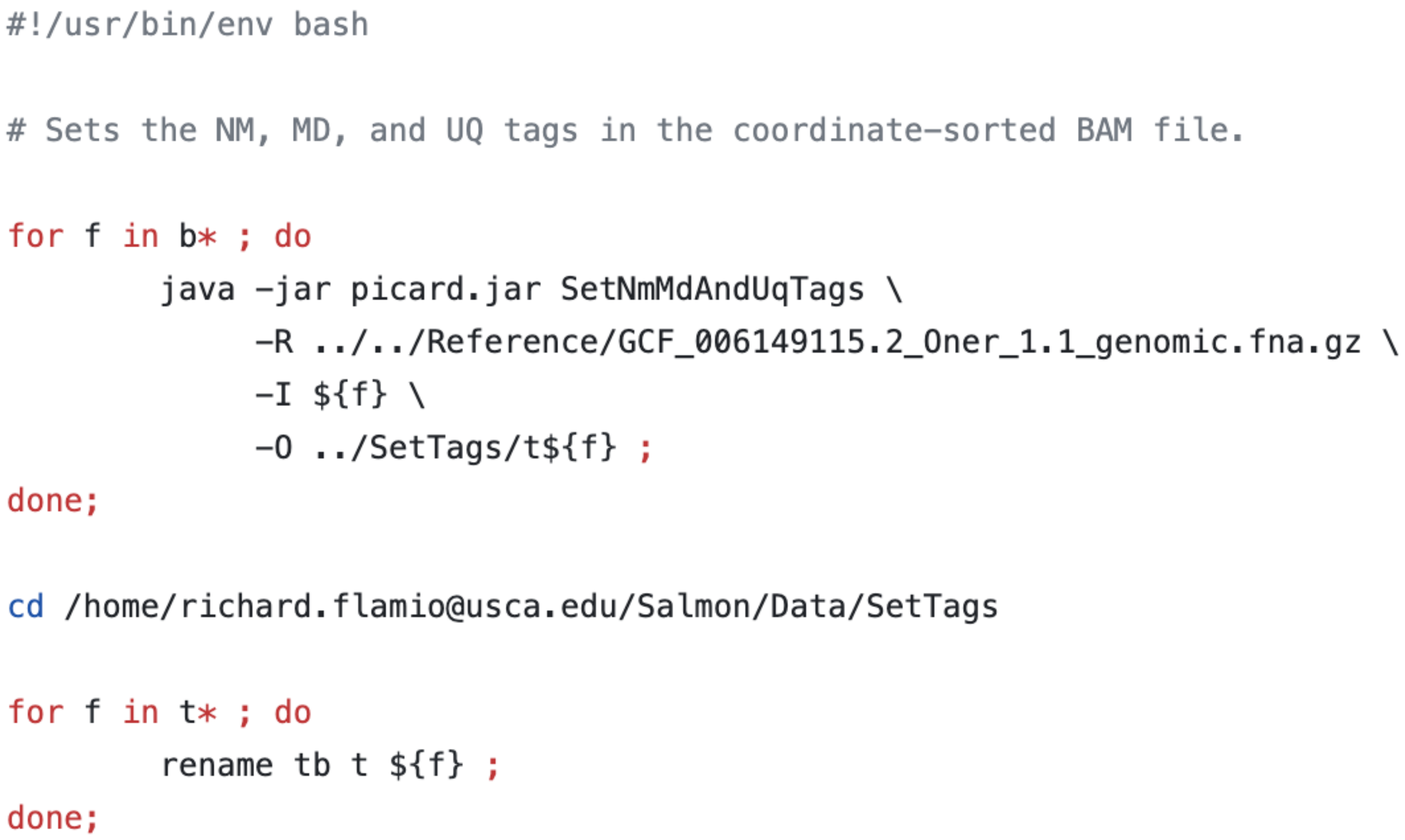
Second round of coordinate sorting



```
#!/usr/bin/env bash

# Sort BAM files in which duplicates have been removed by coordinate.

for f in m*bam ; do
        java -jar picard.jar SortSam \
             -I ${f} \
             -O ../bSortedBAM/b${f} \
             -SO coordinate ;
done;

cd /home/richard.flamio@usca.edu/Salmon/Data/bSortedBAM

for f in b* ; do
        rename bm b ${f} ;
done;
```

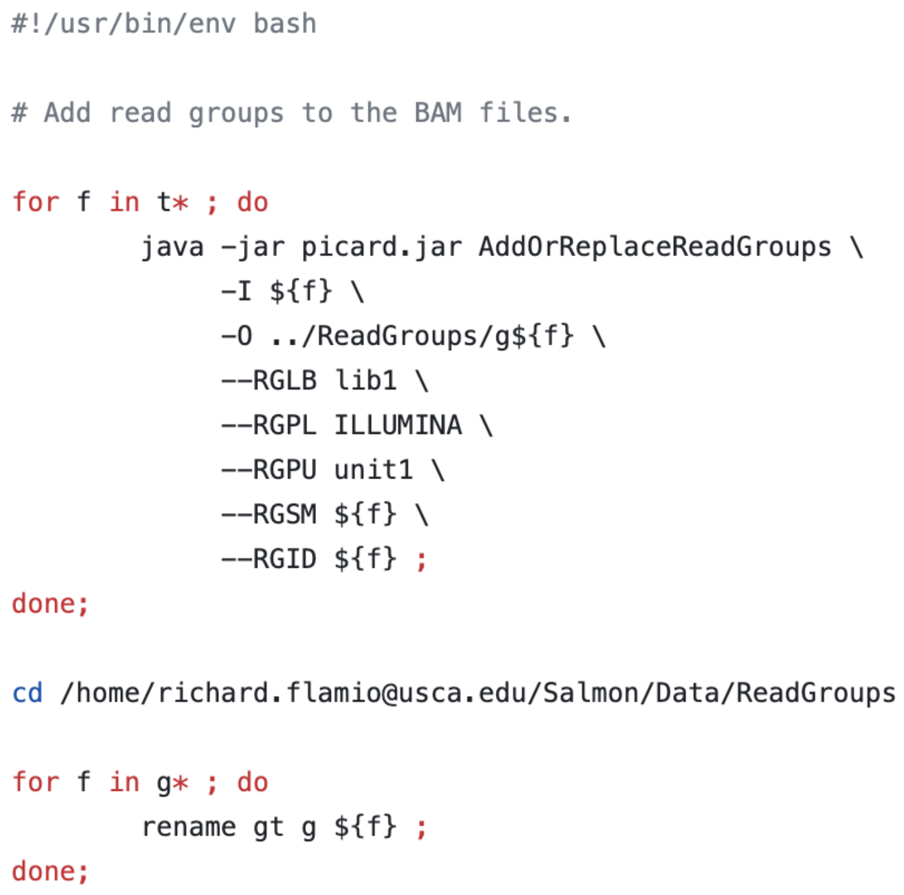
Fixing tags

- Tags are fields in BAM/SAM files
- NM = edit distance to the reference
- MD = mismatched and deleted bases compared to reference
- UQ = Phred likelihood



```bash
#!/usr/bin/env bash

# Sets the NM, MD, and UQ tags in the coordinate-sorted BAM file.

for f in b* ; do
        java -jar picard.jar SetNmMdAndUqTags \
             -R ../../Reference/GCF_006149115.2_Oner_1.1_genomic.fna.gz \
             -I ${f} \
             -O ../SetTags/t${f} ;
done;

cd /home/richard.flamio@usca.edu/Salmon/Data/SetTags

for f in t* ; do
        rename tb t ${f} ;
done;
```

Read Groups



```
#!/usr/bin/env bash

# Add read groups to the BAM files.

for f in t* ; do
        java -jar picard.jar AddOrReplaceReadGroups \
             -I ${f} \
             -O ../ReadGroups/g${f} \
             --RGLB lib1 \
             --RGPL ILLUMINA \
             --RGPU unit1 \
             --RGSM ${f} \
             --RGID ${f} ;
done;

cd /home/richard.flamio@usca.edu/Salmon/Data/ReadGroups

for f in g* ; do
        rename gt g ${f} ;
done;
```

Base Quality Score Recalibration

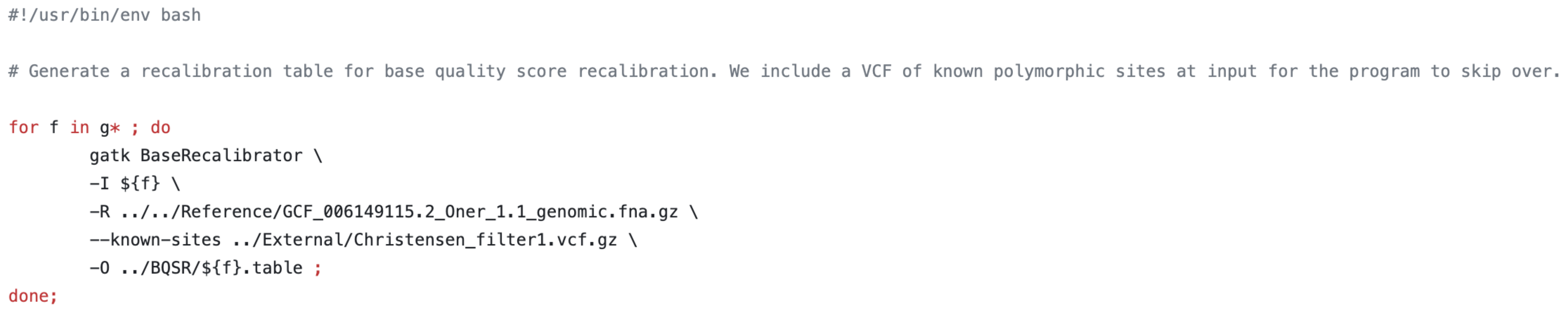
- Detects systematic errors made by the sequencing machine when it estimates the accuracy of each base call
- Can skip this step if you have a non-model organism with limited resources (no datasets of observable variation)

- First, use BaseRecalibrator (GATK tool) to generate a recalibration table.
- Then, use ApplyBQSR (GATK tool) to apply the recalibration.

BaseRecalibrator

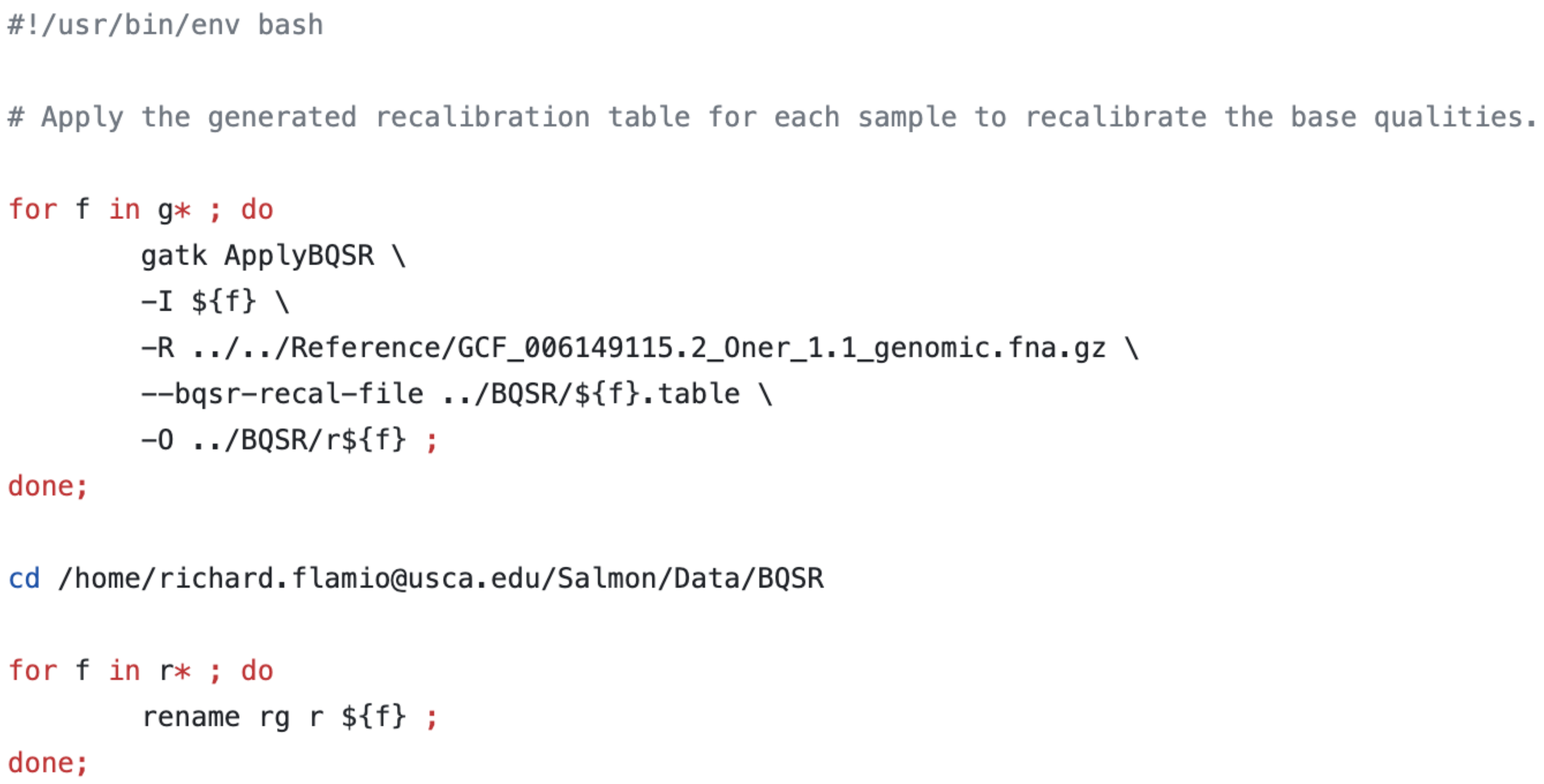
- Requires the reference to have .fai and .dict files
  - These can be produced by using samtools (http://www.htslib.org/)
- Requires a file of known variable sites to avoid during calibration.
  - This file needs to be indexed.
  - If non-model organism, absence of this resource is why this step might need to be skipped.
- Ideally base quality score recalibration does a better job at filtering and retaining variants than hard filtering would do.

Base Recalibrator



```
#!/usr/bin/env bash

# Generate a recalibration table for base quality score recalibration. We include a VCF of known polymorphic sites at input for the program to skip over.

for f in g* ; do
        gatk BaseRecalibrator \
        -I ${f} \
        -R ../../Reference/GCF_006149115.2_Oner_1.1_genomic.fna.gz \
        --known-sites ../External/Christensen_filter1.vcf.gz \
        -O ../BQSR/${f}.table ;
done;
```

ApplyBQSR

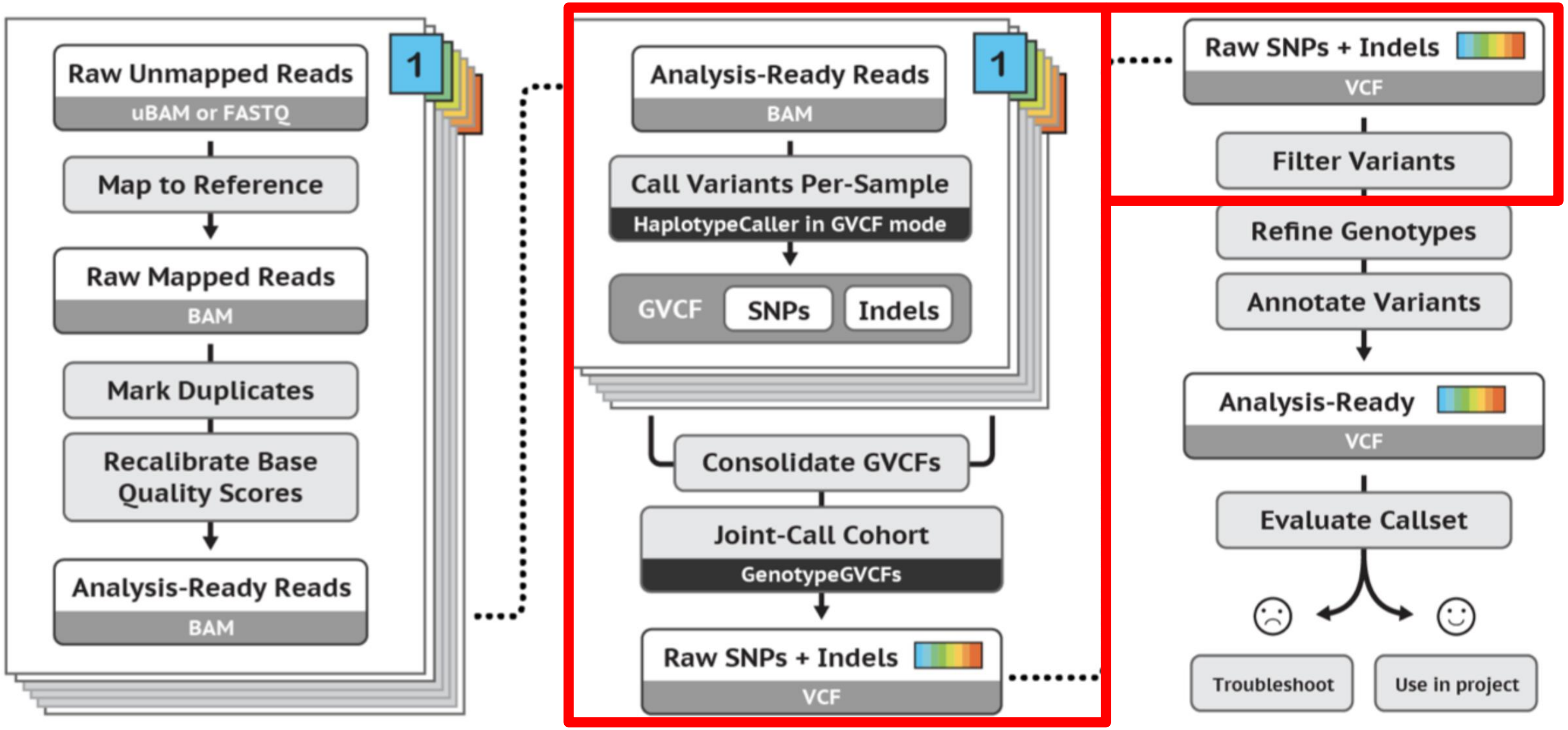


```
#!/usr/bin/env bash

# Apply the generated recalibration table for each sample to recalibrate the base qualities.

for f in g* ; do
        gatk ApplyBQSR \
        -I ${f} \
        -R ../../Reference/GCF_006149115.2_Oner_1.1_genomic.fna.gz \
        --bqsr-recal-file ../BQSR/${f}.table \
        -O ../BQSR/r${f} ;
done;

cd /home/richard.flamio@usca.edu/Salmon/Data/BQSR

for f in r* ; do
        rename rg r ${f} ;
done;
```

Now we have reads ready for variant calling

10 minute break

Germline short variant discovery (SNPs and Indels)



Variant Discovery

- Use HaplotypeCaller in GVCF mode to call SNPs and indels per sample
- Why GVCF mode?
    - VCF (variant call format) produces files with only sites in which that sample was variable
    - GVCF (genomic variant call format) produces files with non-variant positions as well within a sample
- You can control what annotations are appended to the GVCF file
    - i.e. MappingQuality score

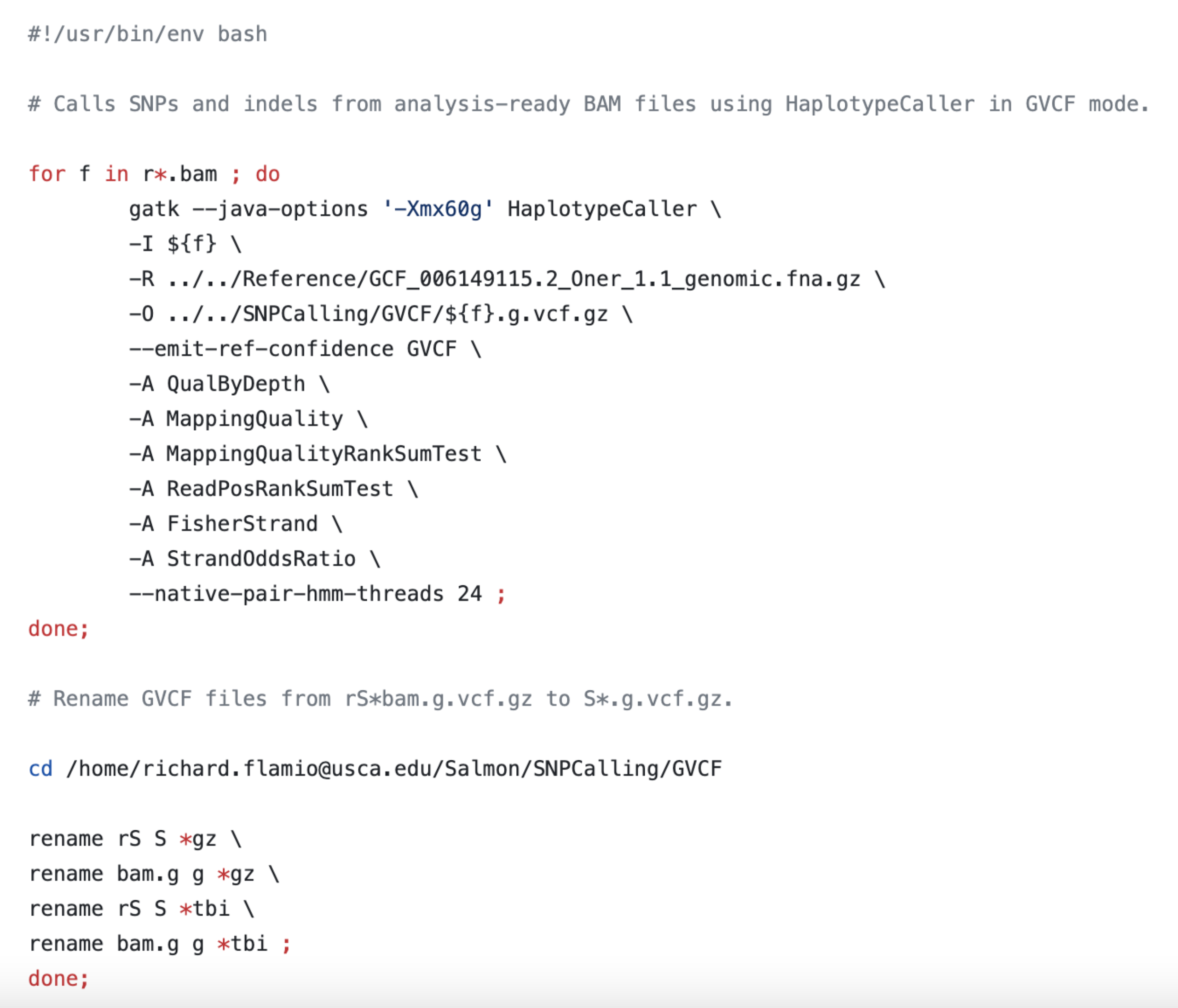
HaplotypeCaller

- This process takes a lot of time and is the rate-limiting step in the pipeline.

For example, it took over two weeks at 6hr/sample to perform this step with 24 threads on 49 samples.

- This can change based on the size of your reference genome and how much sequence data you produced.

HaplotypeCaller



```bash
#!/usr/bin/env bash

# Calls SNPs and indels from analysis-ready BAM files using HaplotypeCaller in GVCF mode.

for f in r*.bam ; do
        gatk --java-options '-Xmx60g' HaplotypeCaller \
        -I ${f} \
        -R ../../Reference/GCF_006149115.2_Oner_1.1_genomic.fna.gz \
        -O ../../SNPCalling/GVCF/${f}.g.vcf.gz \
        --emit-ref-confidence GVCF \
        -A QualByDepth \
        -A MappingQuality \
        -A MappingQualityRankSumTest \
        -A ReadPosRankSumTest \
        -A FisherStrand \
        -A StrandOddsRatio \
        --native-pair-hmm-threads 24 ;
done;

# Rename GVCF files from rS*bam.g.vcf.gz to S*.g.vcf.gz.

cd /home/richard.flamio@usca.edu/Salmon/SNPCalling/GVCF

rename rS S *gz \
rename bam.g g *gz \
rename rS S *tbi \
rename bam.g g *tbi ;
done;
```

Produce a GenomicsDB workspace

- Before joint-calling of variants, we need to produce a GenomicsDB workspace.
- Uses the tool GenomicsDBImport.
- Requires:
  - GVCF files
  - Path to a new database
  - The intervals in which to process the data
    - I did this by creating a bed file in which intervals were the names of the chromosomes in the genome
- This step can be optimized for efficiency
  - Include a path to a location where a large amount of temporary disk storage is available
  - Merging input intervals
  - Bypassing feature reader

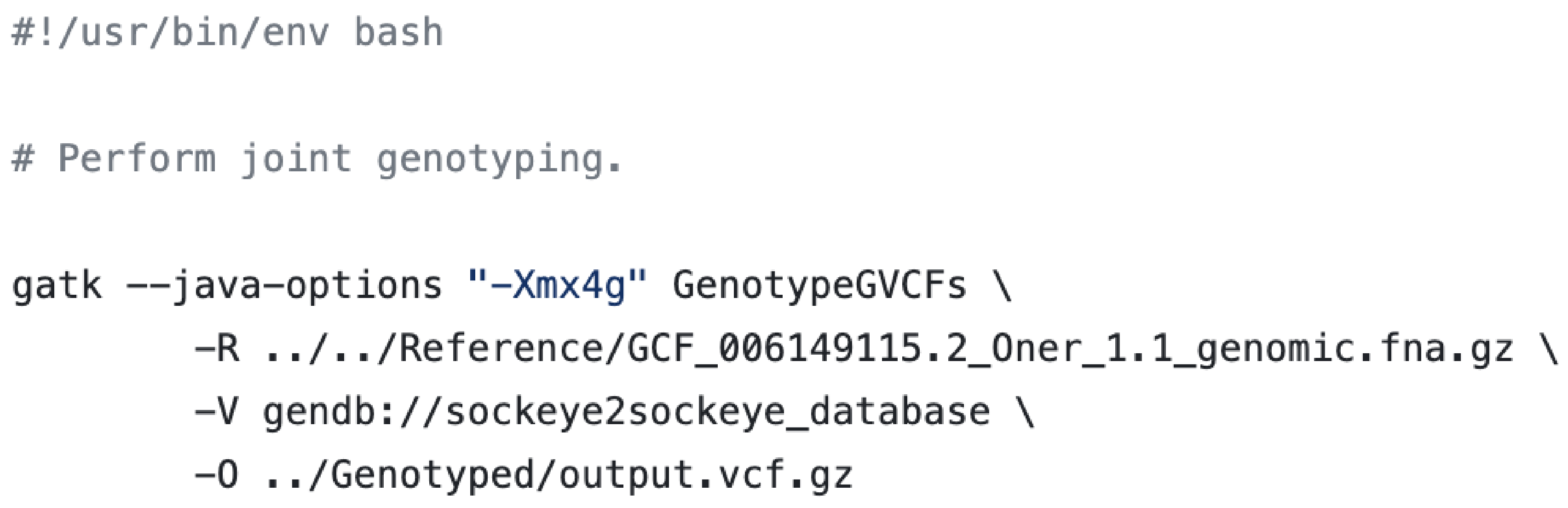
GenomicsDB workspace

Navigate to the consolidate_S.sh script on my github page and we will look at the script together.

https://github.com/rflamio/sockeye_GWAS/blob/main/VariantCalling/consolidate_S.sh

Joint-calling

- We can now joint-call the data to produce one VCF file!
- Uses GenotypeGVCFs tool

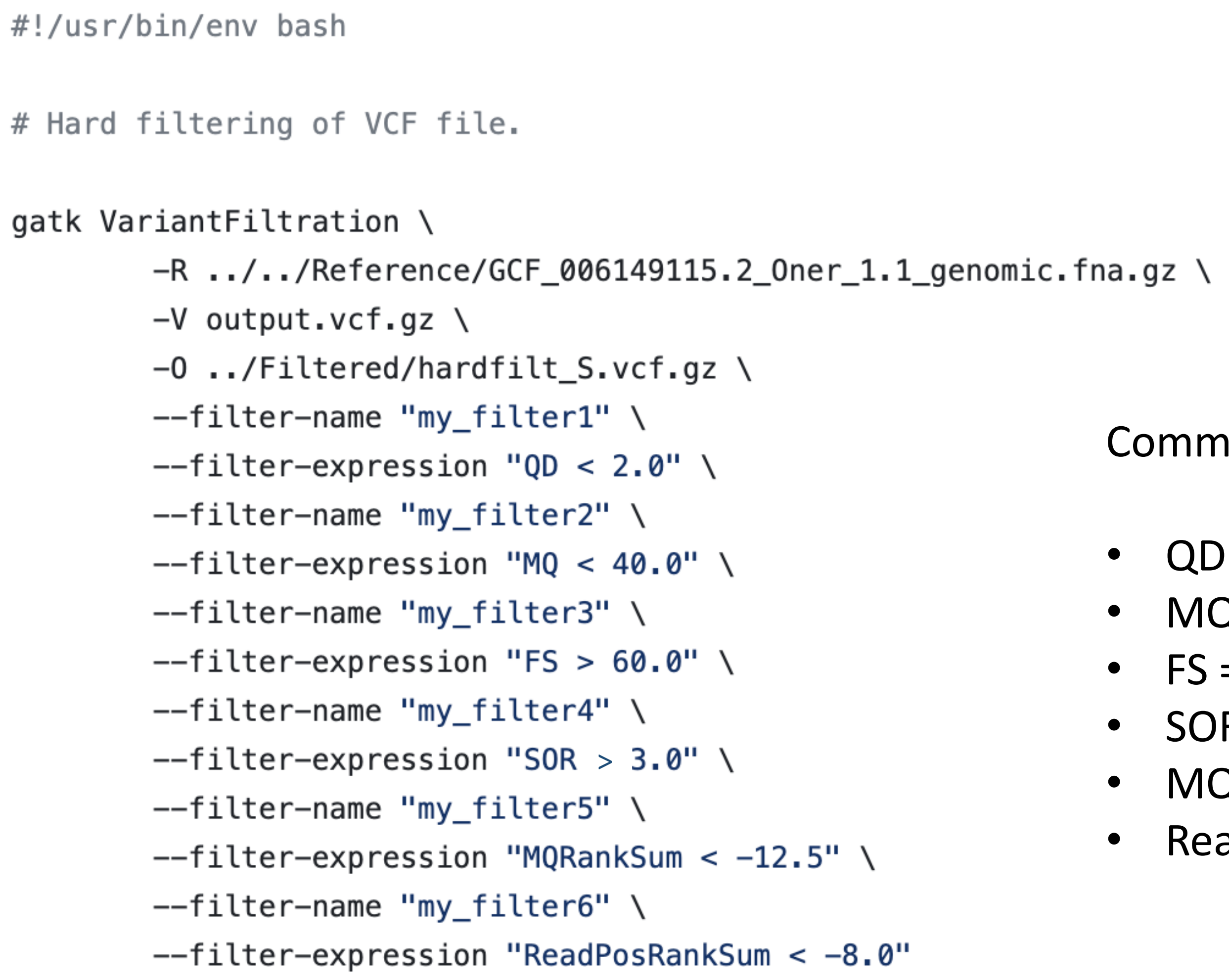


```
#!/usr/bin/env bash

# Perform joint genotyping.

gatk --java-options "-Xmx4g" GenotypeGVCFs \
        -R ../../Reference/GCF_006149115.2_Oner_1.1_genomic.fna.gz \
        -V gendb://sockeye2sockeye_database \
        -O ../Genotyped/output.vcf.gz
```

Variant Filtration in GATK

- For model organisms, can use VariantQualityScoreRecalibration (VQSR) similar to BQSR at this step.
- For non-model organisms, you perform hard-filtering.
  - Relies on hard cutoffs.
  - VariantFiltration tool

Hard Filtering



```
#!/usr/bin/env bash

# Hard filtering of VCF file.

gatk VariantFiltration \
        -R ../../Reference/GCF_006149115.2_Oner_1.1_genomic.fna.gz \
        -V output.vcf.gz \
        -O ../Filtered/hardfilt_S.vcf.gz \
        --filter-name "my_filter1" \
        --filter-expression "QD < 2.0" \
        --filter-name "my_filter2" \
        --filter-expression "MQ < 40.0" \
        --filter-name "my_filter3" \
        --filter-expression "FS > 60.0" \
        --filter-name "my_filter4" \
        --filter-expression "SOR > 3.0" \
        --filter-name "my_filter5" \
        --filter-expression "MQRankSum < -12.5" \
        --filter-name "my_filter6" \
        --filter-expression "ReadPosRankSum < -8.0"
```

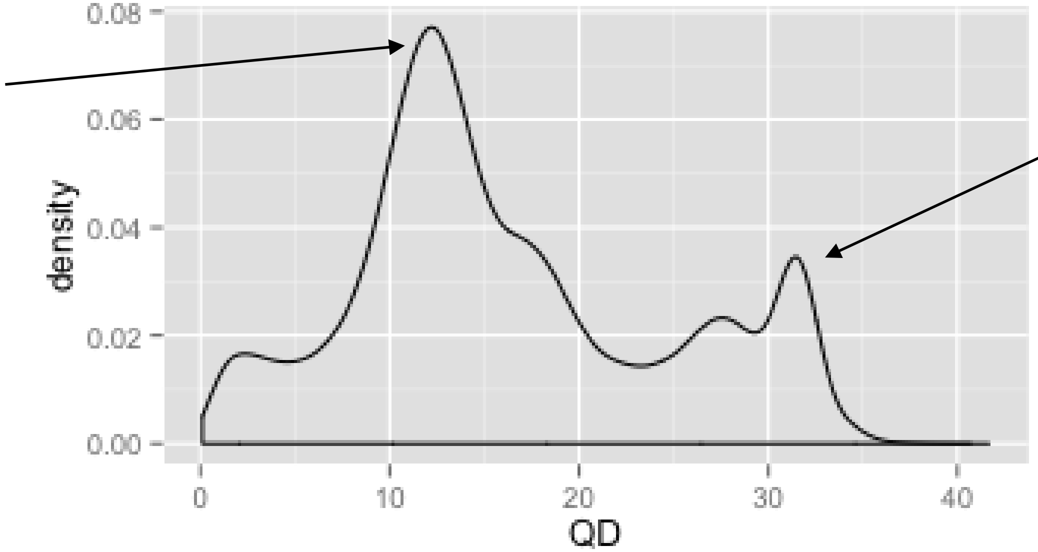
Common filters:

- QD = Quality by depth
- MQ = Root Mean Square Mapping Quality
- FS = Fisher Strand
- SOR = Strand Odds Ratio
- MQRankSum = Mapping Quality Rank Sum Test
- ReadPosRankSum = Read Position Rank Sum Test

QualByDepth (QD)

- Variant confidence ÷ depth for individuals not homozygous for reference allele
- Recommend filter of QD < 2

Heterozygous variants

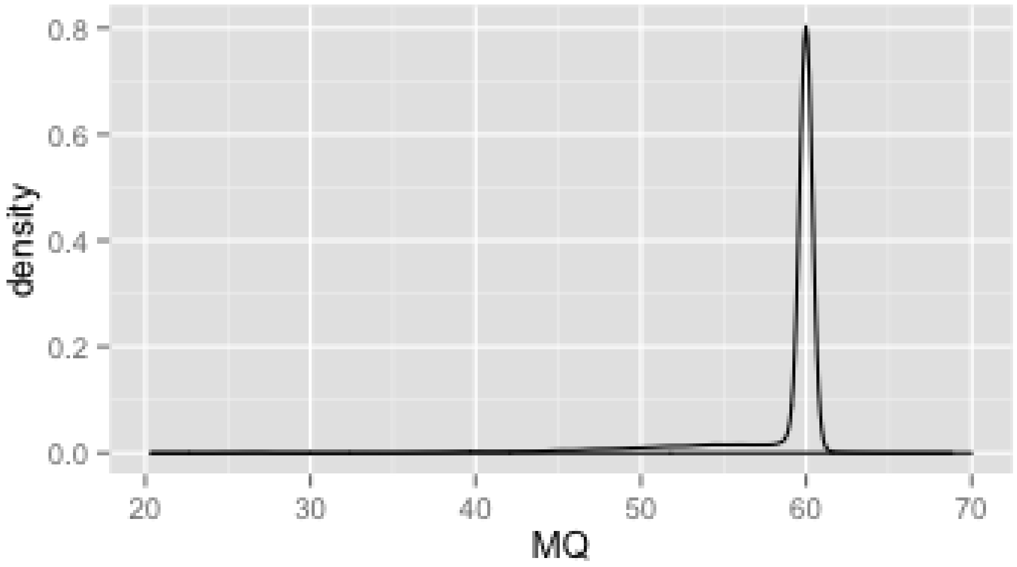


Homozygous for alternate

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

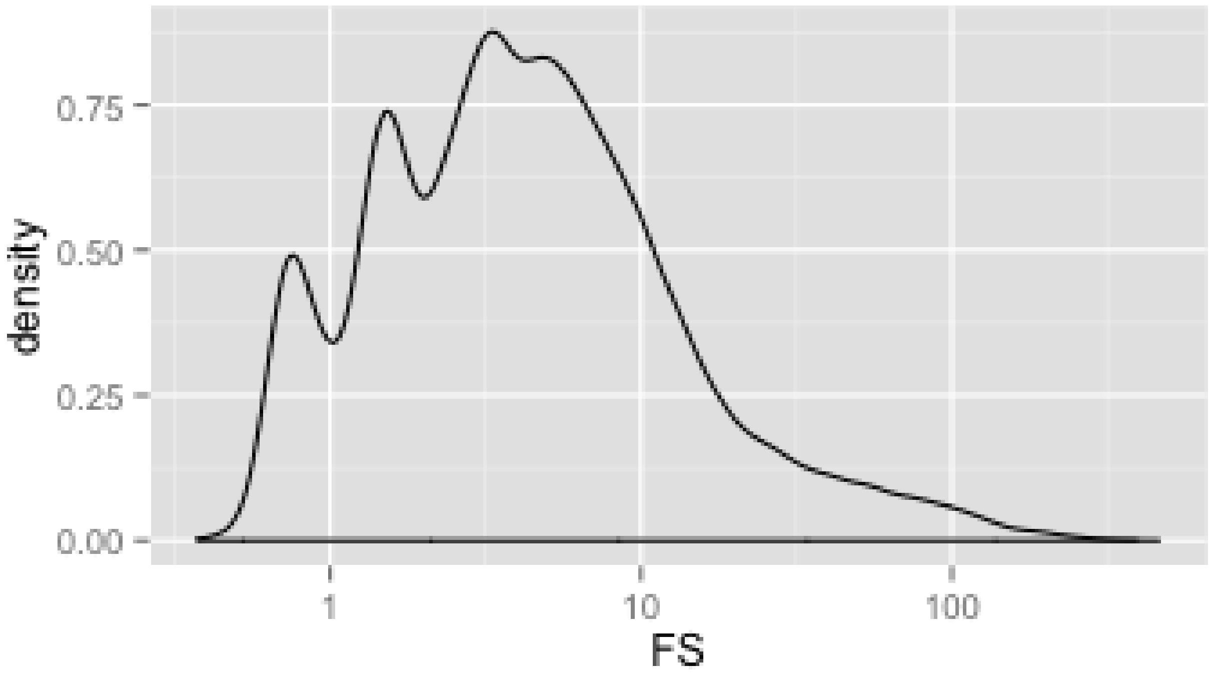
RMS Mapping Quality (MQ)

- Root mean square of mapping quality of all reads at a locus
- Includes standard deviation thus capturing variation in the dataset
- Good MQ = 60
- Recommendation to remove MQ < 40



Fisher Strand

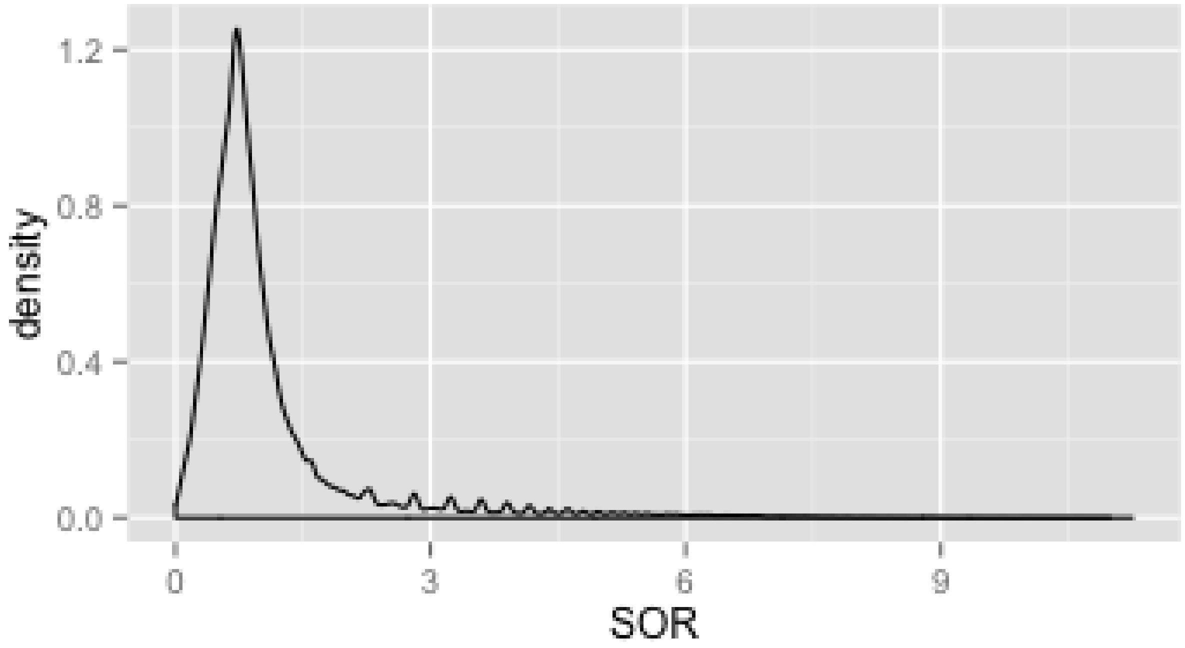
- Phred-scaled probability of strand bias
- Strand bias = when alternate allele is observed at a different frequency on either forward or reverse read compared to reference allele
- FS = 0 indicated little strand bias
- Remove many false positive variants with FS > 60



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

Strands Odd Ratio (SOR)

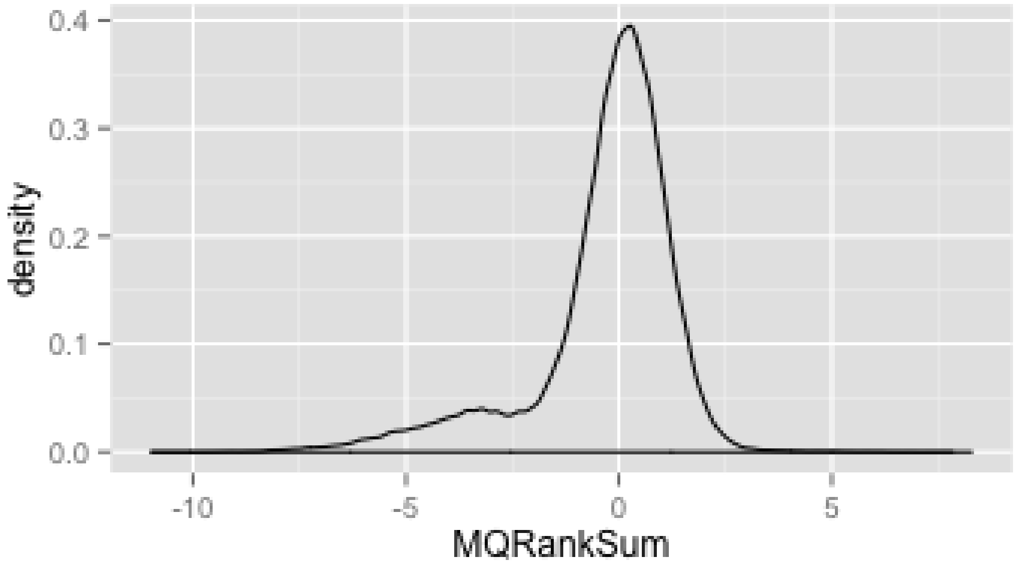
- Another measure of strand bias but does not penalize variants at ends of exons
  - Variants at ends of exons usually represented by only forward or reverse read
- This measure looks at the ratio of reads with each allele
- Recommend to filter out SOR > 3
  - Most data is below an SOR of 3



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

Mapping Quality Rank Sum Test

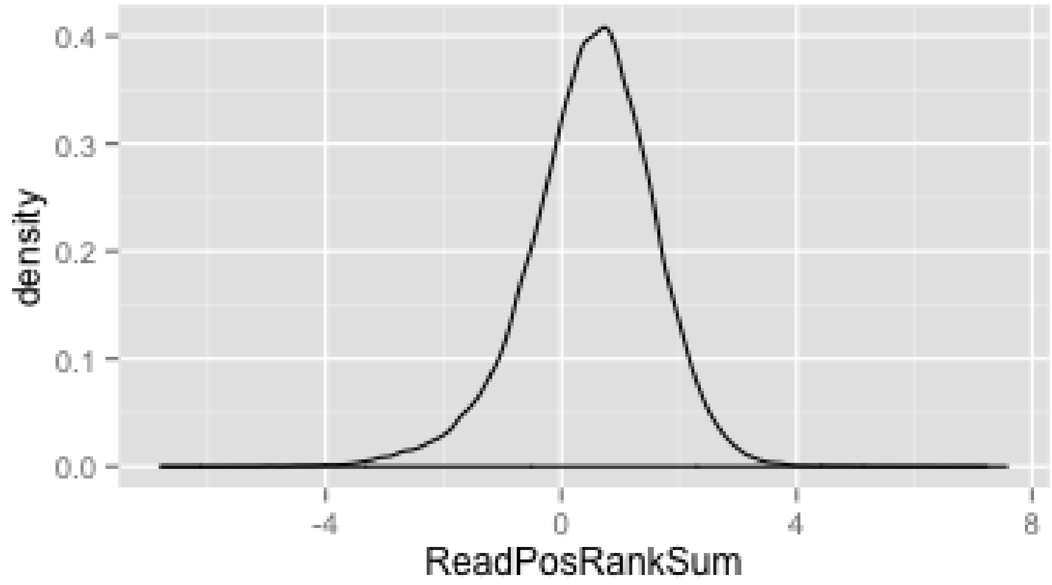
- Compares mapping qualities of reads for reference and alternate alleles
- 0 = ideal, no difference
- < 0 = reference allele mapping qualities are better than alternate allele mapping qualities
- > 0 = alternate allele mapping qualities are better than reference allele mapping qualities
- Recommended filter of MQRankSum < -12.5
  - Why no filter in opposite direction?



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

Read Position Rank Sum Test

- Compares position of reference and alternate alleles in the read
- 0 = ideal, no difference
- < 0 = alternate read found more often on end of read
- > 0 = reference allele found more often on end of read
- Recommended filter of ReadPosRankSumTest < -8



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

Additional VCF filtering programs

- VCFtools
  - https://vcftools.github.io/man_latest.html#SITE%20FILTERING%20OPTIONS
- BCFtools
  - https://samtools.github.io/bcftools/bcftools.html

Some BCF capabilities

- bcftools *command options*
- bcftools view
  - Filter VCF files
- Count number of variants in dataset

```
bcftools view –H vcf_filename | wc –l
```

- Retain only biallelic SNPs

```
bcftools view –m2 –M2 –v snps vcf_filename > filtered_filename
```

where H = no header, m and M = min and max number of alleles allowed, v = variant type

Some VCF capabilities

- vcftools --vcf *filename options* --out *filename*

- Navigate to https://vcftools.github.io/man_latest.html#SITE%20FILTERING%20OPTIONS

- Example: vcftools --vcf all_sim.vcf --remove males.txt --recode –recode-INFO-all --out females_sim

Exercise:

Given the VCF file hardfilter_S.vcf.gz

1. Count the number of variants in the dataset.
2. Retain only biallelic SNPs.
3. Remove sites with a MAF < 0.2. Recode.
    1. How many sites were removed?
    2. How many sites remain?
4. Remove individual tS01. Recode.
5. Remove variants with > 5% missing data. Recode.
6. How many SNPs and individuals are in the final dataset?