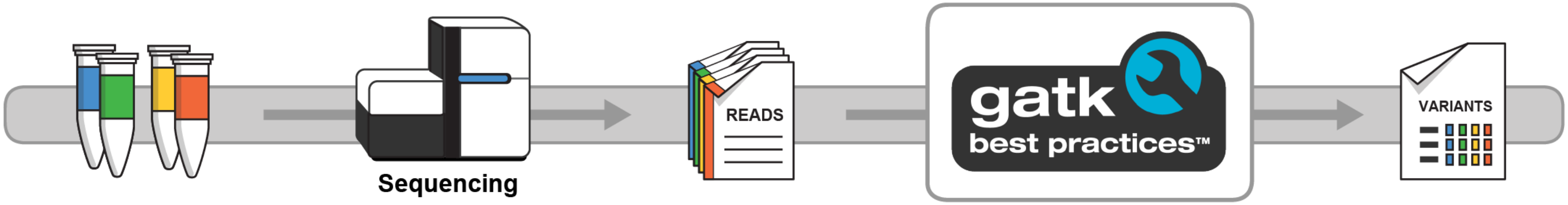
Variant Calling with GATK

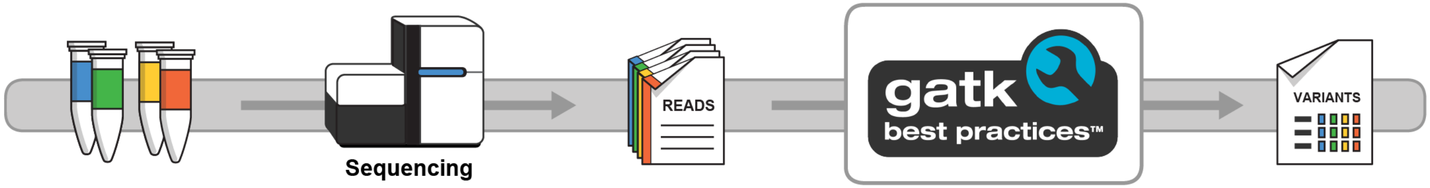
Peter Scott

pscott17@ucla.edu

UCLA Collaboratory, Winter 2019



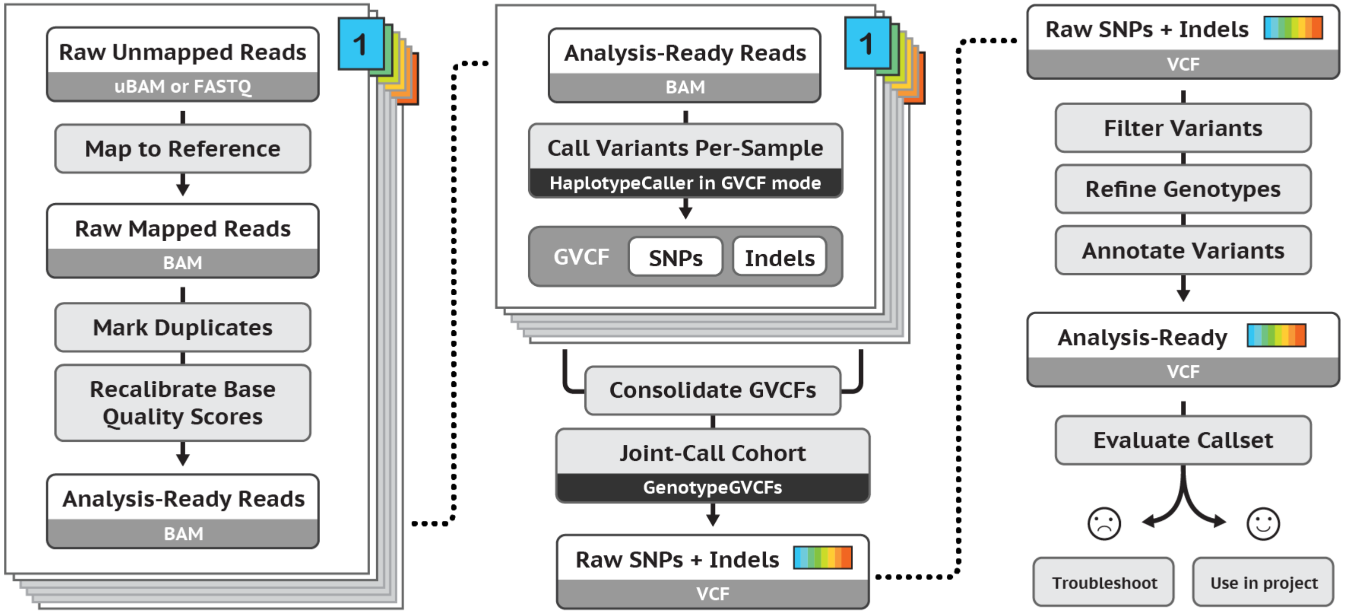
Goals:



- Go from raw data to usable variants.
  - We can’t cover every scenario, so we’ll focus on the main path + common deviations.
- Reproducible science!
  - How to use shells/loops to streamline our work and remember what we have done.
- Where are we?
  - Who I am.
  - Hoffman2? Linux? bash? fastq? VCF?

Basic Outline

- Understand raw data and ready it for GATK "best practices" for calling germline variants.
- Use materials from Broad Institute to perform "best practices".



Day 1

- Hoffman2 setup/intro
  - x2go interactive shell on Hoffman
- Read mapping, clean-up, and BQSR.
- Variant calling (if time)

Day 2

- Continue variant calling
- Hard Filtering
- VQSR

Combined work flow

- I’m going to use a combination of my slides/scripts and those available from the GATK workshops slides.
- Will cater to a basic overview and provide a ‘jumping-off’ point for your specific data
- If you want slides/handouts lets’ get them now.

Using hoffman2

- Log on to hoffman2:
  - `ssh myname@hoffman2.idre.ucla.edu`
- Request an interactive shell:
  - `qrsh -l h_rt=3:00:00,h_data=2G`
- Make a new directory for the workshop and move into it
  - `mkdir todayWS`
  - `cd todayWS`
- Move todays files to your directory:
  - **`git clone`** `https://github.com/QCB-Collaboratory/W1.UNIX.command.line.git`

What did you just get?

- `cd GATK_fall2019`
- `ls`

- data data_gatk docs gatk_profile IGV FastQC
  - data = data from Broad institude
  - data_gatk = my data for mapping, etc.
  - docs = presentations
  - gatk_profile = used to set the specific variables
  - IGV = integrative genome viewer
  - FastQC = fastq quality control

Tools we'll use

- GATK4.0
    - https://software.broadinstitute.org/gatk/gatk4
    - Promote use in GATK firecloud/Spark/Tarra/Jupyter…
- BWA
    - http://bio-bwa.sourceforge.net/
    - One of the most popular short-read mappers
- Picard
    - https://broadinstitute.github.io/picard/
    - From broad to manipulate SAM/BAM/VCF files
- Samtools
    - http://www.htslib.org/
    - Manipulate SAM/BAM/CRAM files and others with htslib.

GATK

- Basic syntax:

```
gatk --java-options "-Xmx4G" [program arguments]
```

Picard

- Basic syntax:

```
java jvm-args -jar picard.jar PicardToolName \
      OPTION1=value1 \
      OPTION2=value2
```

Slides/handouts

- If you want them on your computer now:
  - In a new terminal or log out of your current one…

  -
    ```
    scp –r yourname@hoffman2.irde.ucla.edu:/u/home/y/yourname/gatkWorkshop/docs .
    ```
  - Or use Filezilla or the similar programs.

How to call a program on Hoffman2:

- Install locally and call the program.
  - `java -Xmx4g -jar \ /u/home/m/myname/GATK.jar …..`
- Use the module pre-loaded on Hoffman.
  - `module load gatk`
  - Can also go at the top of the shell script
    - `. /u/local/Modules/default/init/modules.sh`
    - `module load modulefile`
- Make a profile to load local programs
  - `source gatk_profile`

How to run a program/script on Hoffman2

- In an interactive shell:
  - Smaller/shorter jobs. We’ll be doing this.
- Submit to the queue:

```
qsub  -cwd -V -m bea  -l h_data-4G,h_rt=24:00:00 myshell.sh
```

  - qsub = submit a job
  - -cwd = run from this current working directory(relative paths)
  - -V = keep these environmental variables
  - -m bea = email at **b**eginning, **e**nd, and **a**bort of job
  - -l h_data-4G,h_rt=24:00:00 = requested memory/time

Install X2go

- Instructions at:
  - https://www.hoffman2.idre.ucla.edu/x2go/
  - You will have to log on and start a new session
  - Use the gui to navigate to:
  - /u/home/galaxy/collaboratory/peter/IGV_2.3.98/
  - Click on IGV.sh and give it a minute

Check out IGV

- Change genome: HG38
- load one of the files in /gatkWorkshop/data/bams/

- Zoom to: chr20:10,002,280-10,002,320

Check GATK

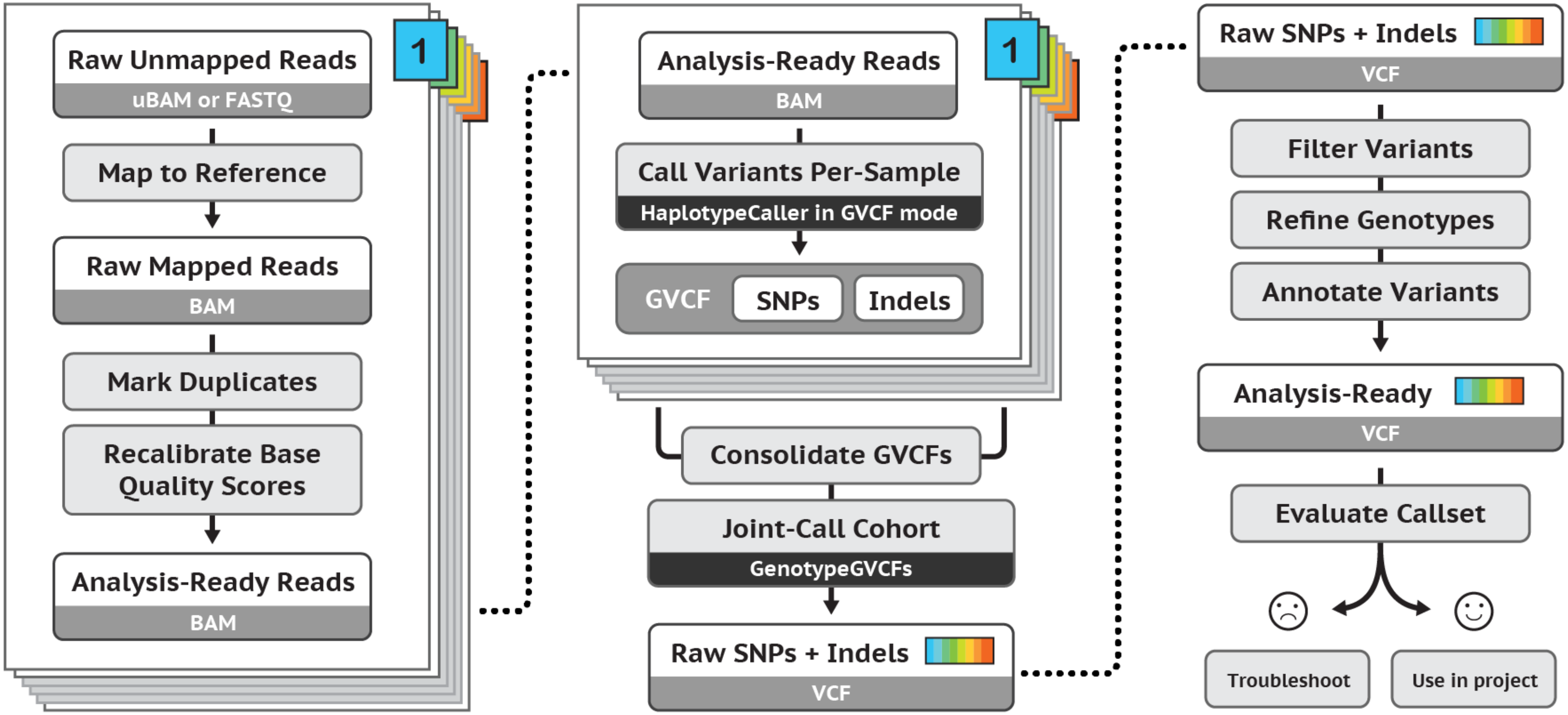
- `cd data`

-
  ```
  $GATK CountReads -R \
  /data/ref/ref.fasta \
  ```

```
-I /data/bams/father.bam
```

What did you get?

Our Path:



1

Raw Unmapped Reads

uBAM or FASTQ

Map to Reference

Raw Mapped Reads

BAM

Mark Duplicates

Recalibrate Base Quality Scores

Analysis-Ready Reads

BAM

1

Analysis-Ready Reads

BAM

Call Variants Per-Sample

HaplotypeCaller in GVCF mode

GVCF SNPs Indels

Consolidate GVCFs

Joint-Call Cohort

GenotypeGVCFs

Raw SNPs + Indels

VCF

Raw SNPs + Indels

VCF

Filter Variants

Refine Genotypes

Annotate Variants

Analysis-Ready

VCF

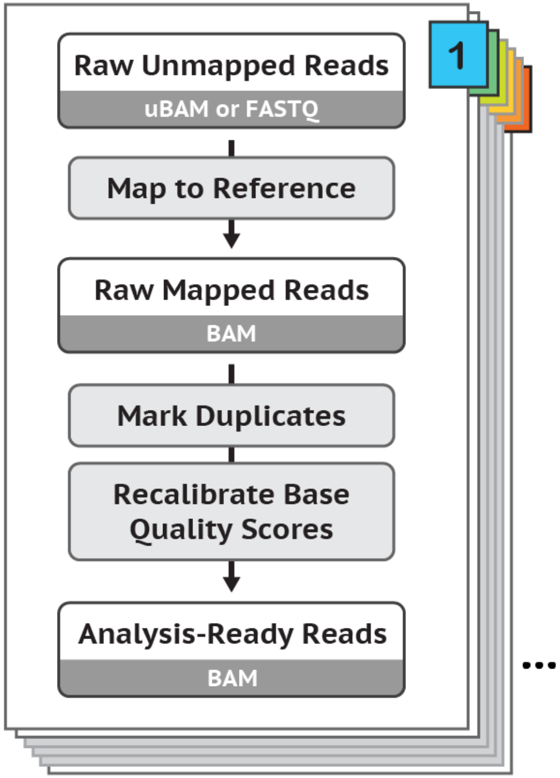
Evaluate Callset

☹ ☺

Troubleshoot

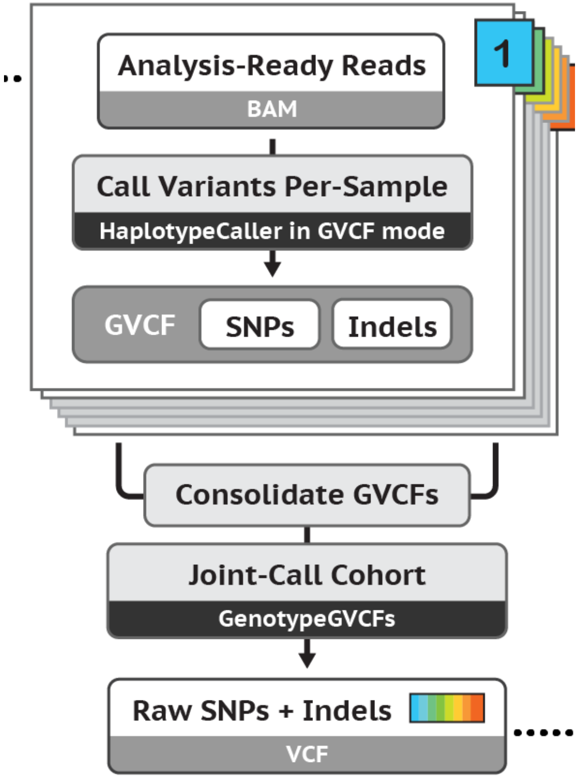
Use in project

Our Path:



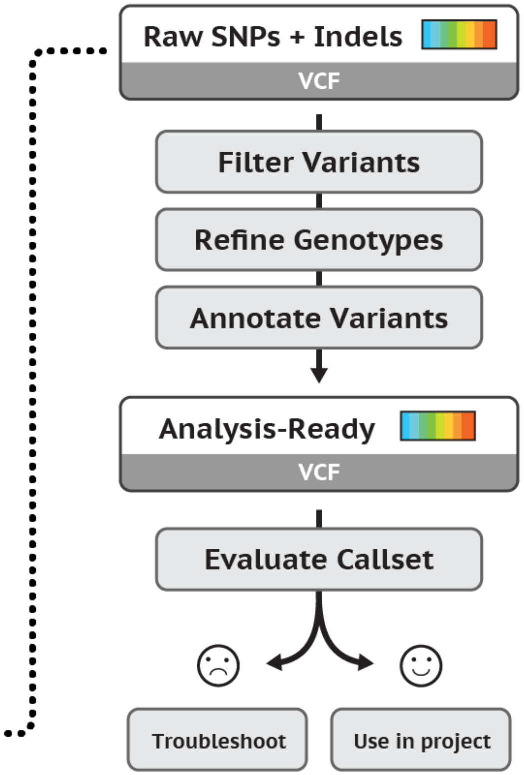
- Map reads
  - Need reference genome and data
- Process mapped reads
  - sam to bam
  - Sort
  - Clean
  - Mark duplicates
  - Add readgroups
  - BQSR

Our Path:



- Call variants
  - Make genomic VCF
- Merge GVCFs
- Joint-call samples
  - Power of ALL THE DATA!

Our Path:



- Filter our raw data
  - Haplotype caller is generous
  - Clean our data
    - VQSR
    - HARD filtering (how?)
  - Annotate our data
    - We want to know what we have!

DATA SLIDES

- GATKwr23-0A-Intro_to_Sequencing_Data.pdf (from Broad)

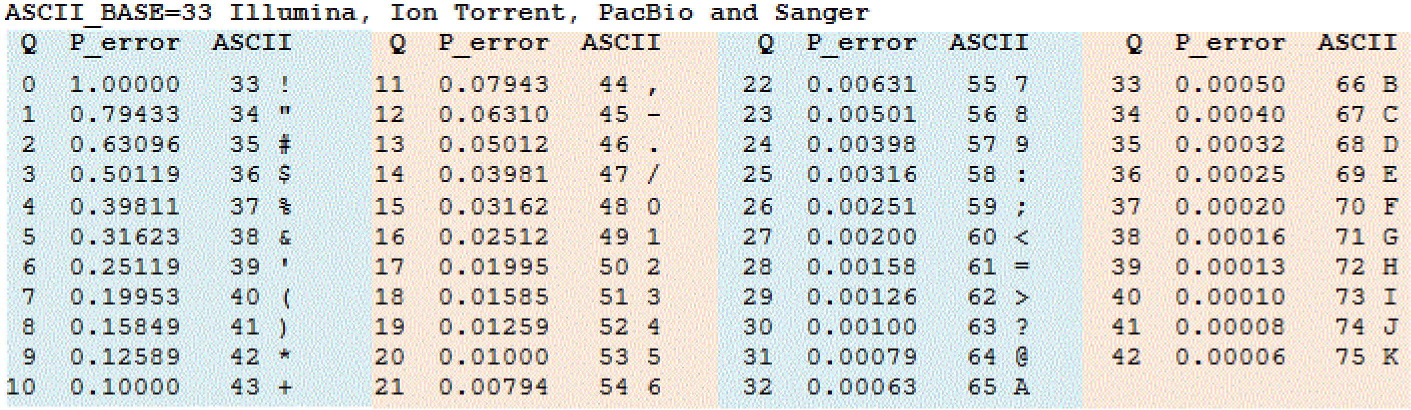
Quality Control

- Some of the best bioinformaticians I know say they spend >90% of their time making sure data quality is great before analyses.
- This really includes experimental/sequencing design.
    - Proper controls?
    - Proper depth?
    - Correct indexing?
- **AND** ensuring data is clean before mapping/analyses/etc.
    - Doesn’t seem to be a major concern for Broad
        - >>>>Human data to correct for quality later.

FASTQ data

@ERR188583.2 HS15_08626:1:1101:1121:14136#2/1
GGATCTATATCAGTCACATA
+
>@@FCG6CBC6@1D7@EE?D

- read location
- sequence
- + (well isn't' that annoying! a random "+" to link data)
- read quality (in PHRED [ASCII character coded log-based confidence in a correct read score])
  - "upper" alphabet is good; symbols are bad



ASCII_BASE=33 Illumina, Ion Torrent, PacBio and Sanger

Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	33 !	11	0.07943	44 ,	22	0.00631	55 7	33	0.00050	66 B
1	0.79433	34 "	12	0.06310	45 -	23	0.00501	56 8	34	0.00040	67 C
2	0.63096	35 #	13	0.05012	46 .	24	0.00398	57 9	35	0.00032	68 D
3	0.50119	36 $	14	0.03981	47 /	25	0.00316	58 :	36	0.00025	69 E
4	0.39811	37 %	15	0.03162	48 0	26	0.00251	59 ;	37	0.00020	70 F
5	0.31623	38 &	16	0.02512	49 1	27	0.00200	60 <	38	0.00016	71 G
6	0.25119	39 '	17	0.01995	50 2	28	0.00158	61 =	39	0.00013	72 H
7	0.19953	40 (	18	0.01585	51 3	29	0.00126	62 >	40	0.00010	73 I
8	0.15849	41)	19	0.01259	52 4	30	0.00100	63 ?	41	0.00008	74 J
9	0.12589	42 *	20	0.01000	53 5	31	0.00079	64 @	42	0.00006	75 K
10	0.10000	43 +	21	0.00794	54 6	32	0.00063	65 A			

One way to check for quality

- FastQC
- Open X2Go client
- Navigate to ~/gatkWorkshop/FastQC
    - Click on fastqc
    - Wait….
- “open” and navigate to /GATK_fall2019/quality
    - Open SRR708379_4m_R1.fastq.gz
    - Open boxTurtle_R1.fastq
    - Open HBS108600_S15_L008_R2_001.fastq.gz

If my quality is bad?

- Preprocess data before you map
- Be mindful of keeping read pairs matched
  - Trimmomatic
    - http://www.usadellab.org/cms/?page=trimmomatic
  - BBtools
    - https://jgi.doe.gov/data-and-tools/bbtools/
  - Cutadapt
    - https://cutadapt.readthedocs.io/en/stable/
  - Many, many, more…..

Making shells

- Great for reproducibility and repeatability without issue

```
#!/bin/bash
For R1 in *_R1.fastq.gz; do
        echo $R1
        R2=`echo $R1 | sed ‘s/_R1/_R2/’`
        out=`echo $R1 | sed ‘s/_R1.fastq.gz/_rawMap/’`
        echo $R2
        echo $out
        bwa mem –M –t 8 /u/home/………./ref/ref.fasta \
        ./$R1 ./$R2 > ./mapped/$out.sam
```

What do we map too?

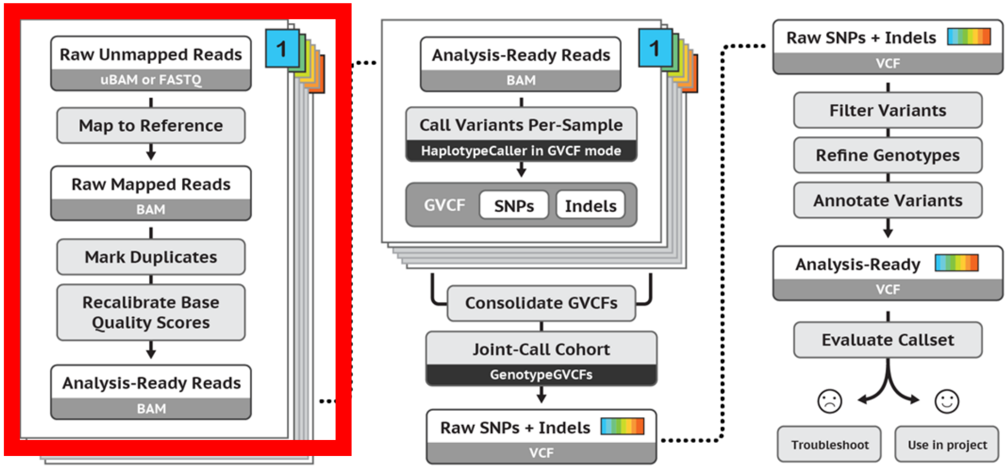
- A reference genome (.fa/.fasta)
- We need to prepare the reference, just once, for mapping
  - `cd ref`
  - `bwa index ref.fasta`
  - `samtools faidx ref.fasta`
  - `java -jar $PICARD CreateSequenceDictionary R=ref.fasta O=ref.dict`

We will:

- Map raw reads: map_loop1.sh
  - GATKwr23-DP1-Mapping.pdf (from Broad)
- Convert sam file to bam file: sam2bam_loop2.sh
  - Saves space
- If we want unmapped bam: uBAM_loop2b.sh
  - `samtools view _rawMap.bam | less`
  - `samtools view _u.bam | less`
- Sort the bam by position: sort_loop3.sh
- Mark duplicate sequences:MarkDup_loop4.sh
  - GATKwr23-DP2-Marking_duplicates.pdf (from Broad)

Next....

- Add read groups to the files: readGroup_loop5.sh
- Clean the bam: Clean_loop6.sh
- Do Base Quality Score Recalibration:
  - tableBQSR_loop7.sh
  - applyBQSR_loop8.sh
  - GATKwr23-DP3-Base_Recalibrator (from Broad)
  - Ya!!!!!



Now Variant Calling

- See (from Broad):
- GATKwr26-0B
- GATKwr26-G0
- GATKwr26-G1
- GATKwr26-G2
- Tutorial 2a.

HaplotypeCaller in VCF mode

- motherHC_1.sh
  - Generates a VCF file based on BAM file for chr20 basepairs: 10,000,000-10,200,000

  - Load input bam (bams/mother.bam) and output VCF (sandbox/motherHC.vcf) into IGV and zoom to 20:10,002,294-10,002,623

  - Hmmm… why do we call an INDEL that is so ‘poorly supported’?
    - View > preferences> alligments and “show soft-clipped bases”

HaplotypeCaller in VCF mode

- motherHCdb_2.sh
  - Generate new BAM based on behavior of haplotypecaller
  - Reduced region for area of focus
  - Does this seem better for the INDEL?

  - Load new bam into IGV
    - Right click on motherHCdebug.bam
      - Color alignments by readgroup

HaplotypeCaller in GVCF mode

- motherHC_gvcf_3.sh
    - Clear IGV
        - File> newSession
    - Load output .g.vcf file

    - What are the grey bars?
        - Invariant tracts
    - What are the colors?
        - Variable sites

Calling variants across samples

- Make Genomic Database
    - GenomicsDB_4.sh

- Find variants for checking
    - select_variants_5.sh
    - What is here? Check in IGV? Are their variants?

- Run joint genotyping
    - jointGenotype_6.sh
    - What is here? Check in IGV

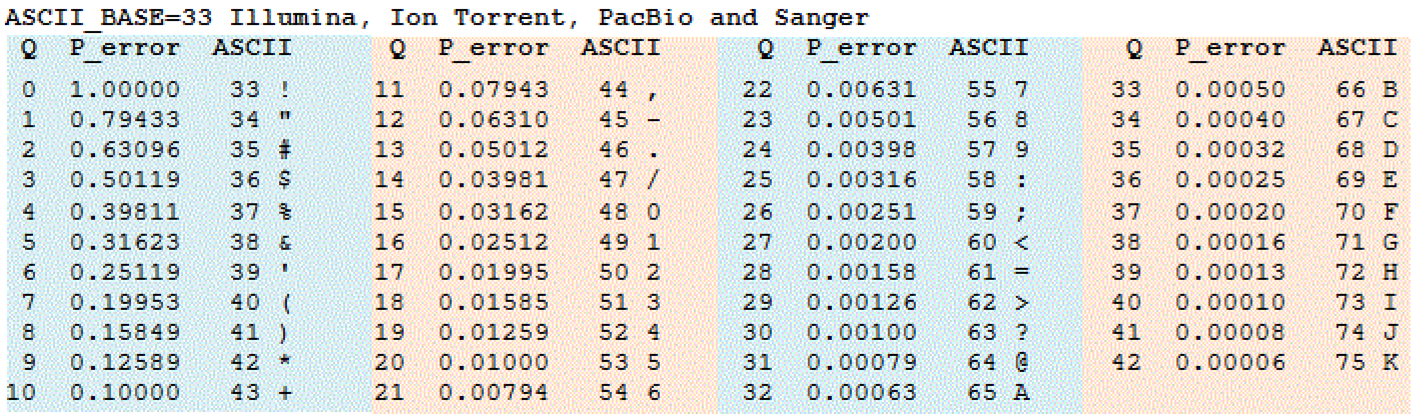
Calling variants across samples

- Can joint call with haplotypecaller
  - jointGenotype_6.sh
  - Scales poorly to add additional samples

Back to the data….

- What can we infer about our data?
  - Is the father the father (NA12877)?

- Look at PL score (PHRED-scaled Likelihood)
  - The probability of the alternative allele
  - -10*log(PL)
  - ref/ref, ref/alt, alt/alt
  - -10*log(0.460) = 3.372 (minus lowest to scale)



ASCII_BASE=33 Illumina, Ion Torrent, PacBio and Sanger

Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII	Q	P_error	ASCII
0	1.00000	33 !	11	0.07943	44 ,	22	0.00631	55 7	33	0.00050	66 B
1	0.79433	34 "	12	0.06310	45 -	23	0.00501	56 8	34	0.00040	67 C
2	0.63096	35 #	13	0.05012	46 .	24	0.00398	57 9	35	0.00032	68 D
3	0.50119	36 $	14	0.03981	47 /	25	0.00316	58 :	36	0.00025	69 E
4	0.39811	37 %	15	0.03162	48 0	26	0.00251	59 ;	37	0.00020	70 F
5	0.31623	38 &	16	0.02512	49 1	27	0.00200	60 <	38	0.00016	71 G
6	0.25119	39 '	17	0.01995	50 2	28	0.00158	61 =	39	0.00013	72 H
7	0.19953	40 (	18	0.01585	51 3	29	0.00126	62 >	40	0.00010	73 I
8	0.15849	41)	19	0.01259	52 4	30	0.00100	63 ?	41	0.00008	74 J
9	0.12589	42 *	20	0.01000	53 5	31	0.00079	64 @	42	0.00006	75 K
10	0.10000	43 +	21	0.00794	54 6	32	0.00063	65 A			

Variant Filtering

- Variant Annotations: Lots of statistics and values based on the properties of a variant relative to the sequence context.
- Importantly, these are relative. A “poor” score can be a real variant, and vice versa.
- VQSR uses machine learning and a truth set (or many) to recalculate variants based on their genomic environment.
    - Very similar to BQSR – so let’s skip it.

Flexibility?

- Scenario: I’m not working on humans, how do I perform VQSR?
  - Short answer: you can’t.

Flexibility?

- Scenario: I’m not working on humans, how do I perform VQSR?
  - Short answer: you can’t.

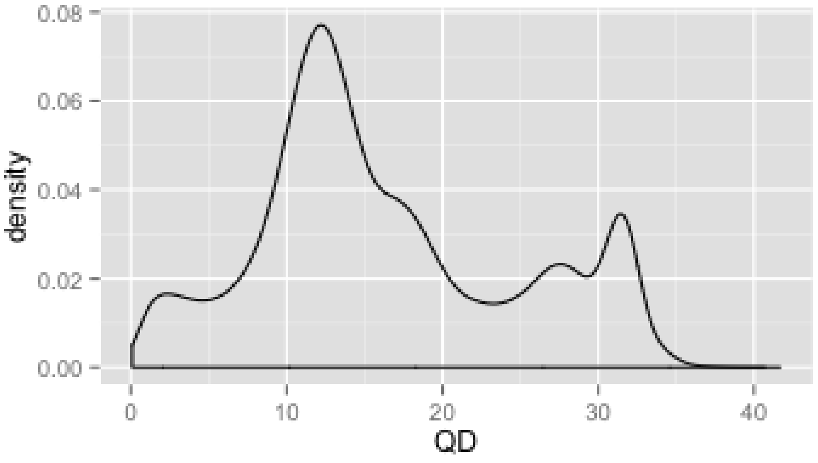
  - But, you can hard filter your call-set (but how?).
  - Answer: use filter guidelines from Broad Institute and your info found in your own data.

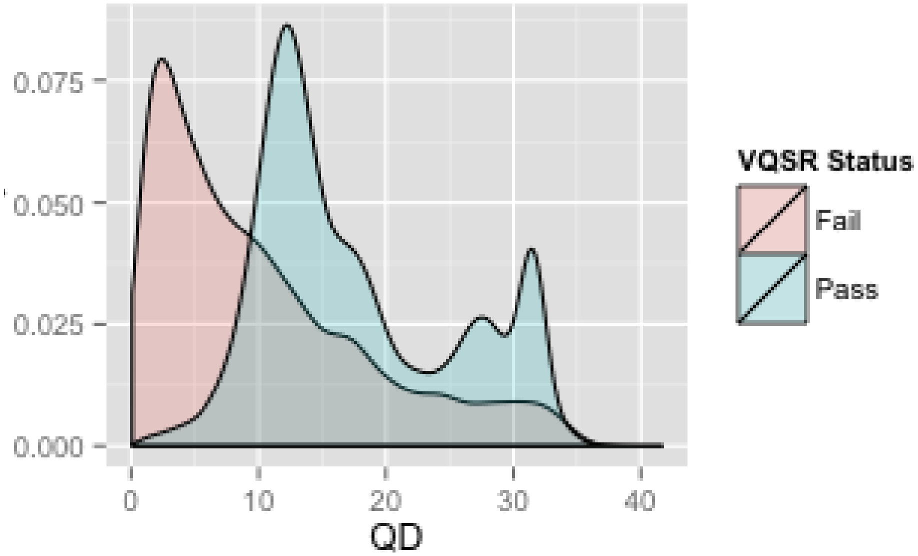
Variant Filtering

- Ideally we could VQSR; but….
  - We need lots of data
  - We need known truth sets
- Examines the context of all quality scores (similar to BQSR) and provides new quality values for them
- Based on all of the read mapped quality scores, not QS (this is what is recalculated)
- Hard filtering lets us understand the process and test filters for our data.

QualByDepth (QD) 2.0

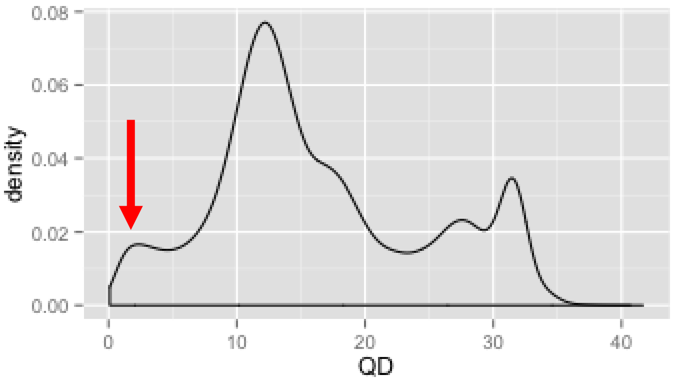
- This is the variant confidence (from the QUAL field) divided by the unfiltered depth of non-reference samples.

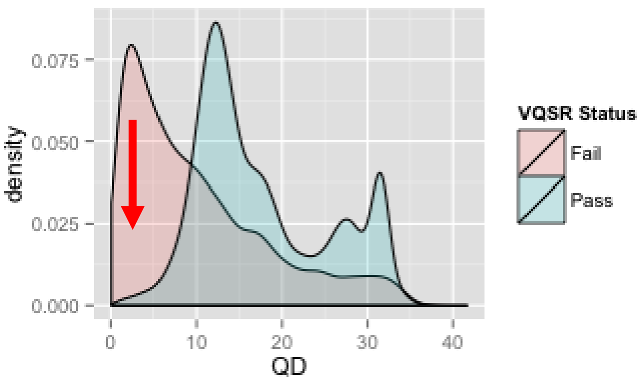




Flexibility?

- Quality by Depth (QD) pre- and post-VQSR

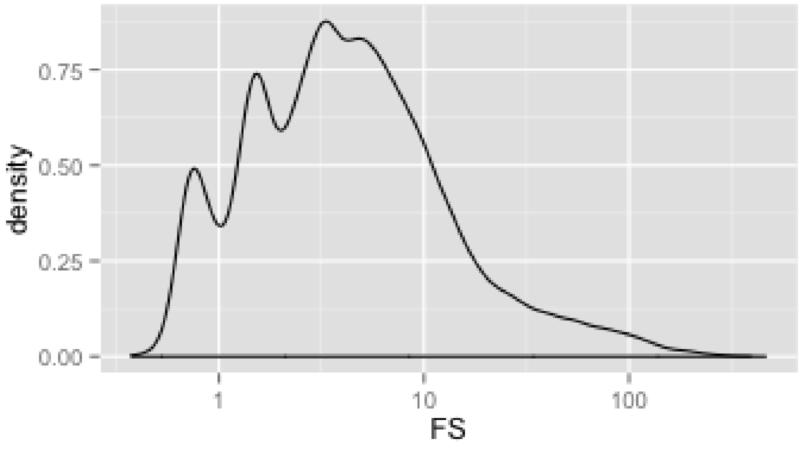


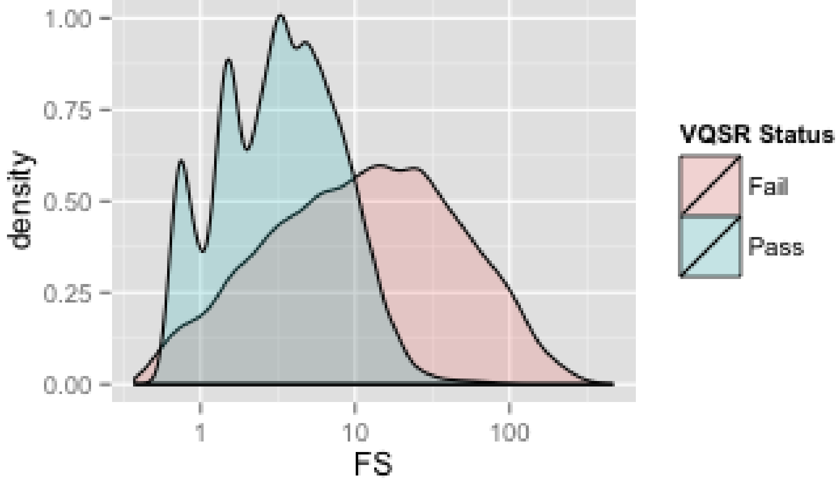


  - “safe” value from Broad is QD < 2.0

FisherStrand (FS) 60.0

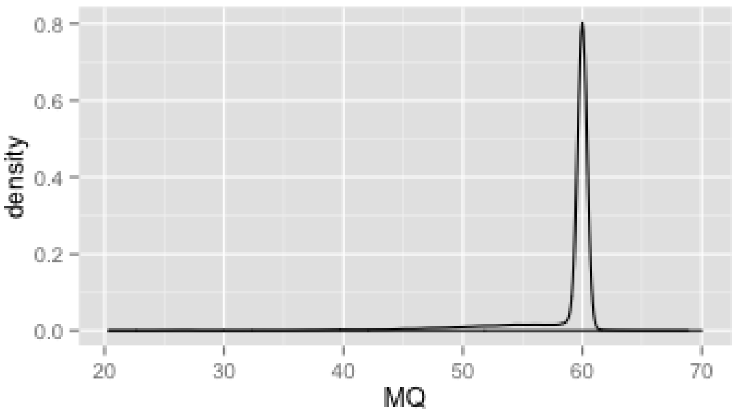
- Phred-scaled p-value using Fisher's Exact Test to detect strand bias (the variation being seen on only the forward or only the reverse strand) in the reads. More bias is indicative of false positive calls.
- Variation in only one strand is ~bad.

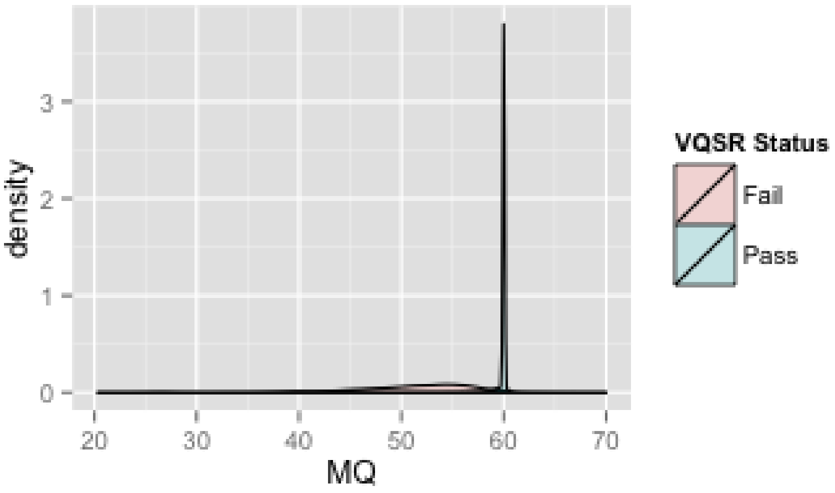




RMSMappingQuality (MQ) 40.0

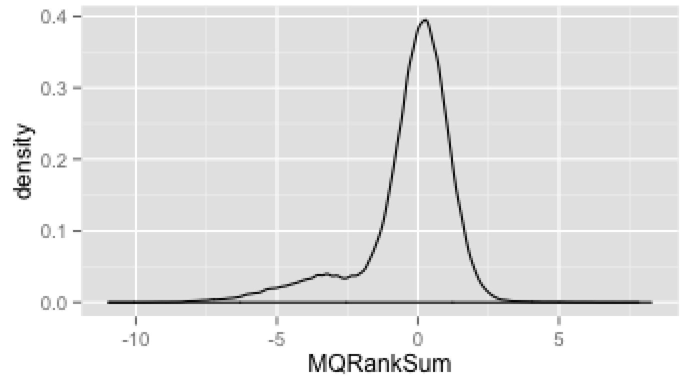
- This is the Root Mean Square of the mapping quality of the reads across all samples

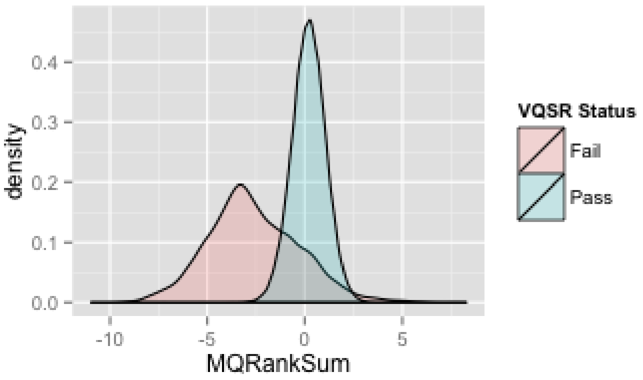




MappingQualityRankSumTest (MQRankSum) -12.5

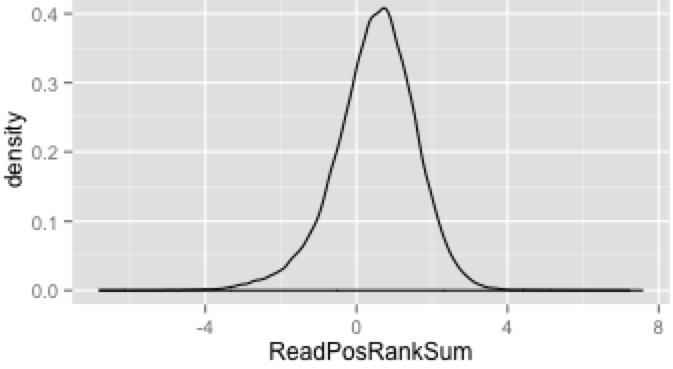
- This is the u-based z-approximation from the Mann-Whitney Rank Sum Test for mapping qualities (reads with ref bases vs. those with the alternate allele)

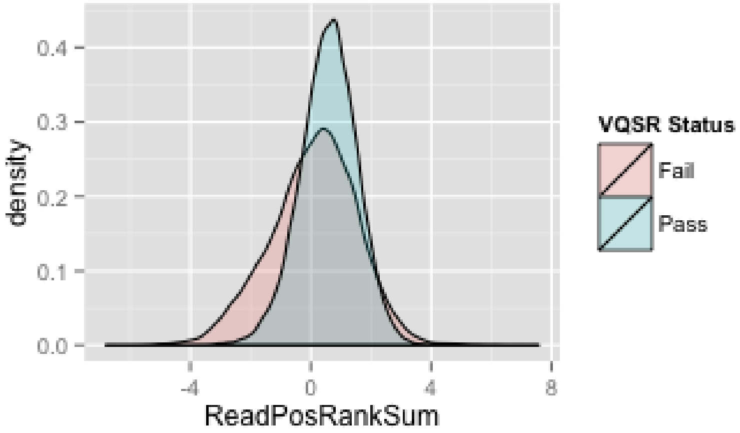




ReadPosRankSumTest (ReadPosRankSum) -8.0

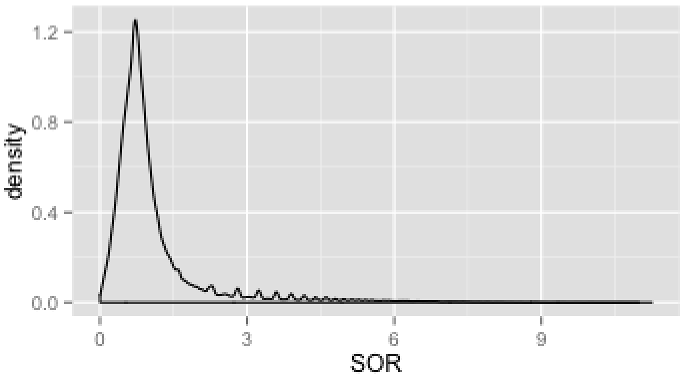
- This is the u-based z-approximation from the Mann-Whitney Rank Sum Test for the distance from the end of the read for reads with the alternate allele.
- Expect more error at ends of reads.

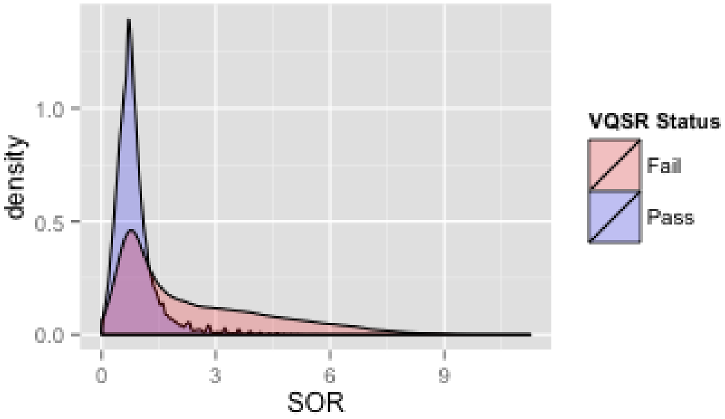




StrandOddsRatio (SOR) 3.0

- The StrandOddsRatio annotation is one of several methods that aims to evaluate whether there is strand bias in the data.
- High =~error



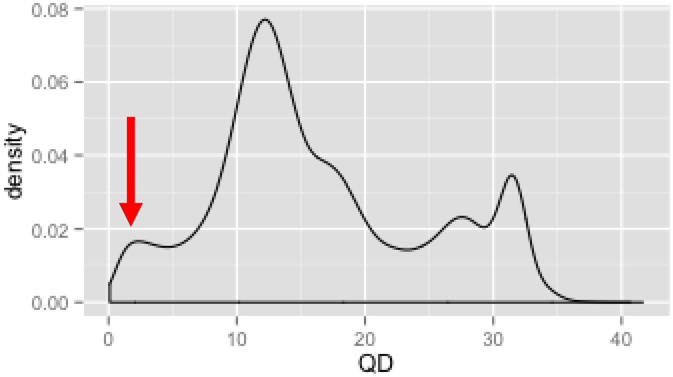


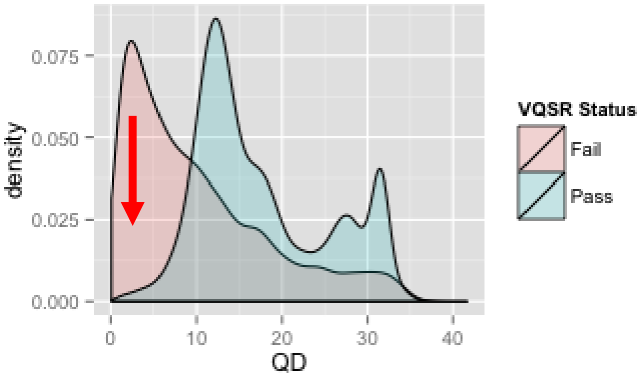
GATK starter recommendations…

```
GATK VariantFiltration \
-R reference.fa \
-V raw_snps.vcf \
--filterExpression "QD < 2.0 || FS > 60.0 \
|| MQ < 40.0 || MQRankSum < -12.5 \
|| ReadPosRankSum < -8.0" \
--filterName "my_snp_filter" \
-o filtered_snps.vcf
```

Flexibility?

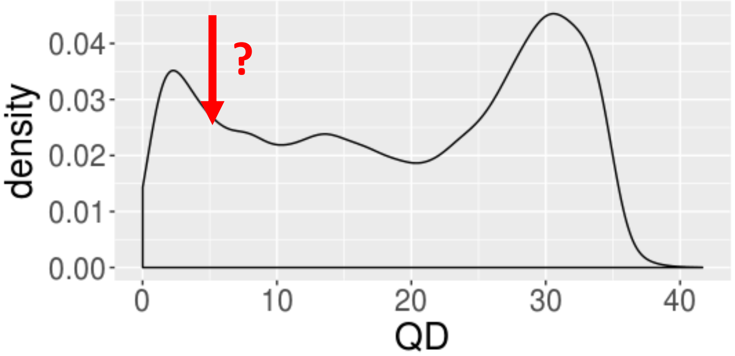
- Quality by Depth (QD) pre- and post-VQSR





  - “safe” value from Broad is QD < 2.0

- Empirical painted turtle target capture data



- Why is the distribution so different?
- Here I chose QD < 5.0

Where is all of this info?

- In the VCF file!
- With descriptions!
- Let’s use it…

How to hard filter?

- select_variants_1b.sh
  - Pull the mother only variants from our trio.vcf
- Annotate true positives in callset
  - variantAnnotator_2b.sh
- VariantsToTable to convert vcf to R-readable table
  - variants2table_3b.sh
- Open plotting_PAS in R
  - Load scripts and open data

How to hard filter?

- Hard filter your data!
  - variantsStartFilter_4b.sh

- Do some rad analyses!