Variant Calling with GATK- Day 3

- Introduction to Variant Filtering
  - GATKwr17-06-Variant_filtering.pdf
- Hard Filtering Variants

Variant Calling with GATK- Day 3

- Introduction to Variant Filtering
  - GATKwr17-06-Variant_filtering.pdf
    - **Just the first 6 slides**
    - open it on your local computer from gatkWorkshop/1702/presentations/
- Hard Filtering Variants

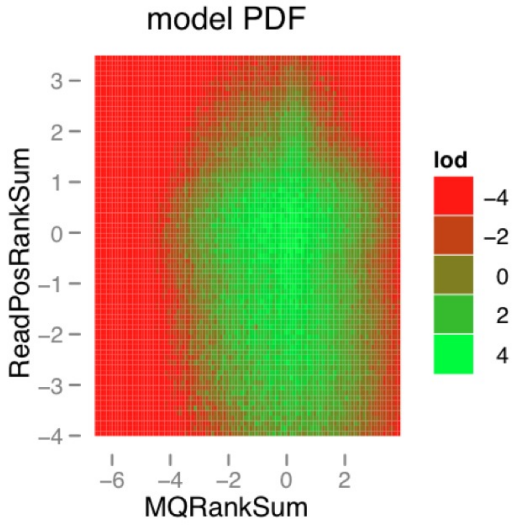
Hard Filtering vs VQSR

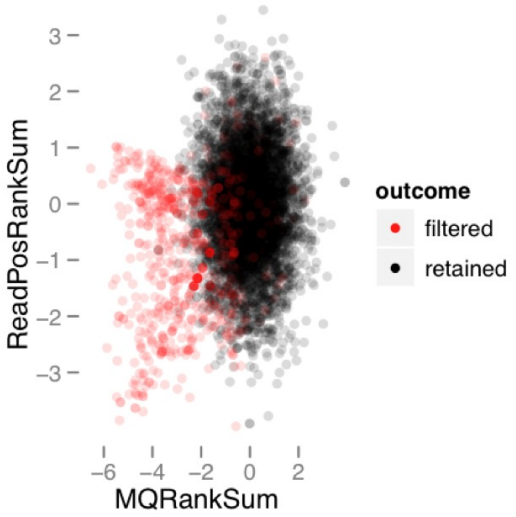
- “hard” because it sets cutoffs for each annotation below which the variant will be filtered.
- VQSR will look at all annotations simultaneously and adjust cutoffs depending on the context.

This workshop covers Hard Filtering because

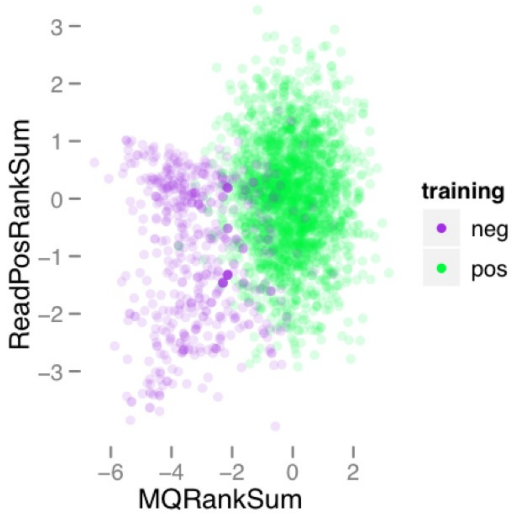
- VQSR is only currently possible with humans and, with difficulty, perhaps a few other model organisms.
- Learning to hard filter teaches you how to interpret your data

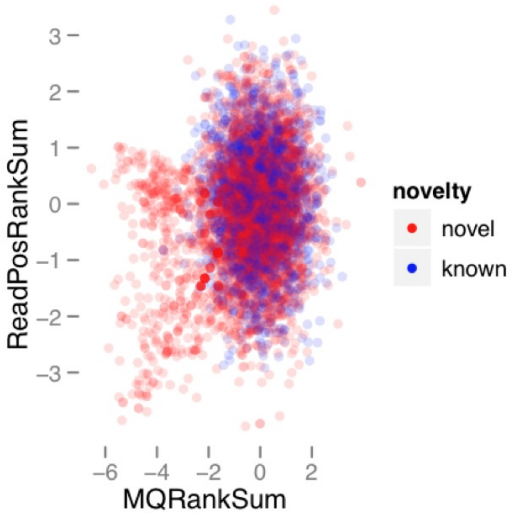
Mixture model plots generated by VQSR





Contrastive evaluation is an important feature of the algorithm





Steps for hard filtering

- Separate SNVs and Indels
- For each, visualize distributions, check alignments & choose cutoffs
- Apply hard filtering cutoffs
- Recombine SNVs and Indels

You can do the following commands on your local computer if you have gatk installed, otherwise do them on hoffman and transfer the results to your local computer for visualizing in IGV

Inspect trio.vcf

- the variants we will filter

Count the variants in the trio vcf file you created yesterday

```
gunzip -c inputVcfs/trio.vcf.gz |grep -v '#' |wc
```

wc is a unix command for word count. It gives three numbers

1. The number of lines in the file
2. The number of words in the file
3. The number of characters in the file

grep –v '^#' gives all lines without a match to # at the beginning of the line

How many variants are there in trio.vcf?

Take a look at the variant lines (skipping the header lines)…

```
gunzip -c inputVcfs/trio.vcf.gz |grep -v '#' |more
```

Take a look at the whole file

```
gunzip -c inputVcfs/trio.vcf.gz |more
```

Separate SNPs and INDELs

- since we will want to apply different filters to each

```
java -Xmx1g -jar $GATK -T SelectVariants -R ref/ref.fasta \
-V inputVcfs/trio.vcf.gz -selectType SNP -o sandbox/trio.snps.vcf
```

```
java -Xmx1g -jar $GATK -T SelectVariants -R ref/ref.fasta \
-V inputVcfs/trio.vcf.gz -selectType INDEL -o sandbox/trio.indels.vcf
```

Count the variants in the new files

```
grep –v ‘#’ sandbox/trio.snps.vcf |wc
```

```
grep –v ‘#’ sandbox/trio.indels.vcf |wc
```

Count the variants in the original file again

```
gunzip -c inputVcfs/trio.vcf.gz |grep -v '#' |wc
```

Are there other kinds of variants besides SNPs and INDELs?

Visualize the distribution of values for annotations

Extract a list of all annotations present in your vcf file

```
head -5000 sandbox/trio.snps.vcf | perl -ne '/^.*INFO.*ID=([a-zA-Z0-9_]+),/ &&
print "-F $1 "' | uniq > sandbox/trio.snps.info_fields
```

This gives a list, handily formatted for use with the VariantsToTable command to extract the values.

```
more sandbox/trio.snps.info_fields
```

Choose which ones you want to visualize, and add them to the VariantsToTable command…

```
java -Xmx1g -jar $GATK -R ref/ref.fasta -T VariantsToTable \
-V sandbox/trio.snps.vcf --allowMissingData -F AN -F BaseQRankSum -F DP \
-F FS -F MQ -F MQRankSum -F QD -F ReadPosRankSum -F SOR \
--out sandbox/trio.snps.tab
```

Look at the table you created

```
more sandbox/trio.snps.tab
```

Visualize the distribution of values for an annotation

Bring the annotation values table to you local computer in order to plot the distributions. Use cyberduck, filezilla, scp or another method.

```
scp –pr \
joebruin@dtn2.hoffman2.idre.ucla.edu:gatkWorkshop/1702/data_bundle/data/
sandbox/trio.snps.tab ./
```

Open R (in a terminal window or Rstudio)

```
R
```

```
snps <- read.table("trio.snps.tab",header=T)
```

```
str(snps)
```

Work through the following slides, and if time read through the presentation at:

https://software.broadinstitute.org/gatk/documentation/article.php?id=6925

(note, to really compare to VQSR they should show pass/fail curves for fully hard filtered sets, not just one hard filter at a time)

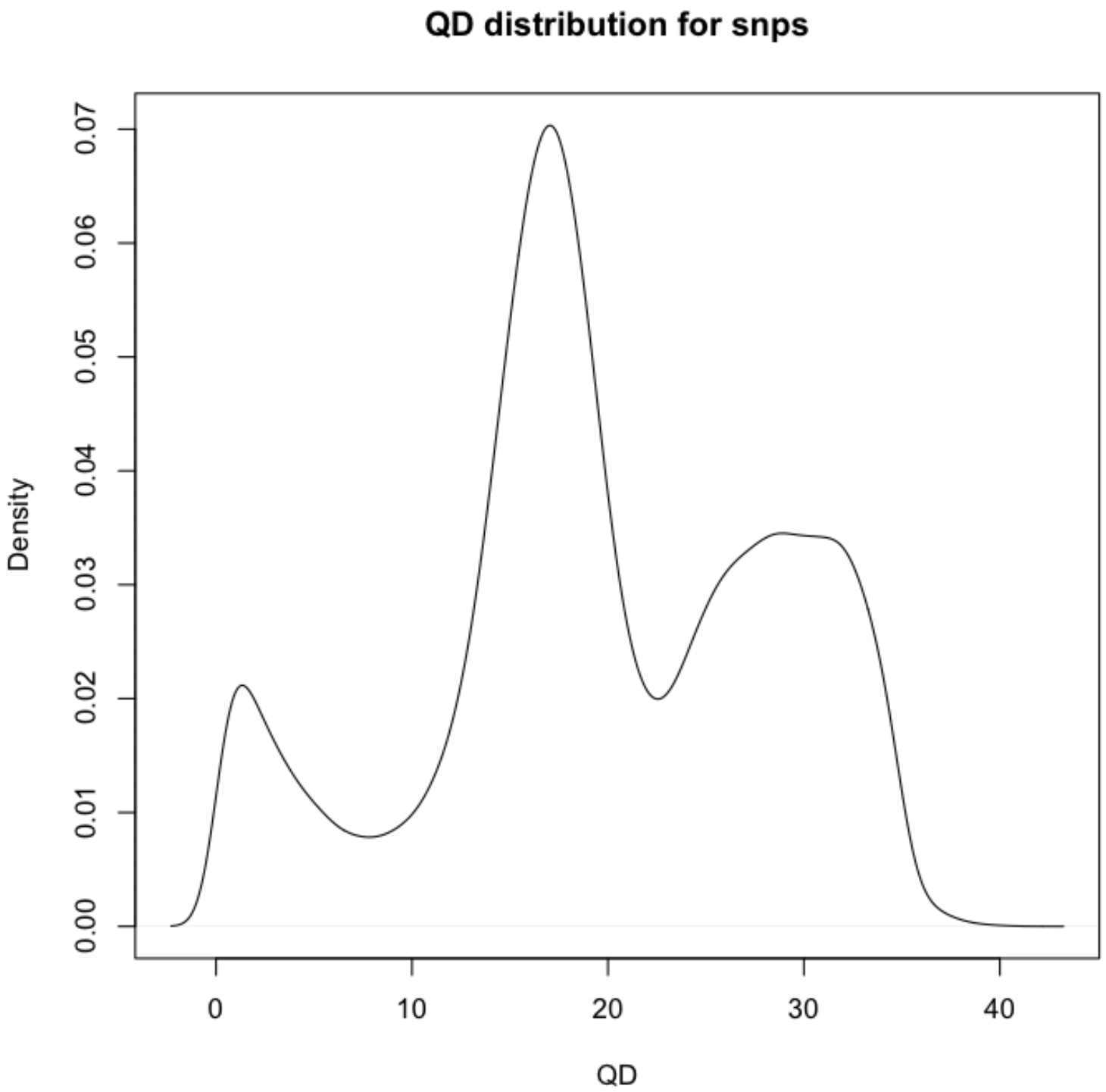
Visualize the distribution of values for an annotation

```
snps <- read.table(“trio.snps.tab”,header=T)
```

```
str(snps)
```

```
dQD <- density(snps$QD,na.rm=T)
```

```
plot(dQD,main="QD distribution for snps", xlab="QD")
```


QD distribution for snps
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0
10
20
30
40
QD

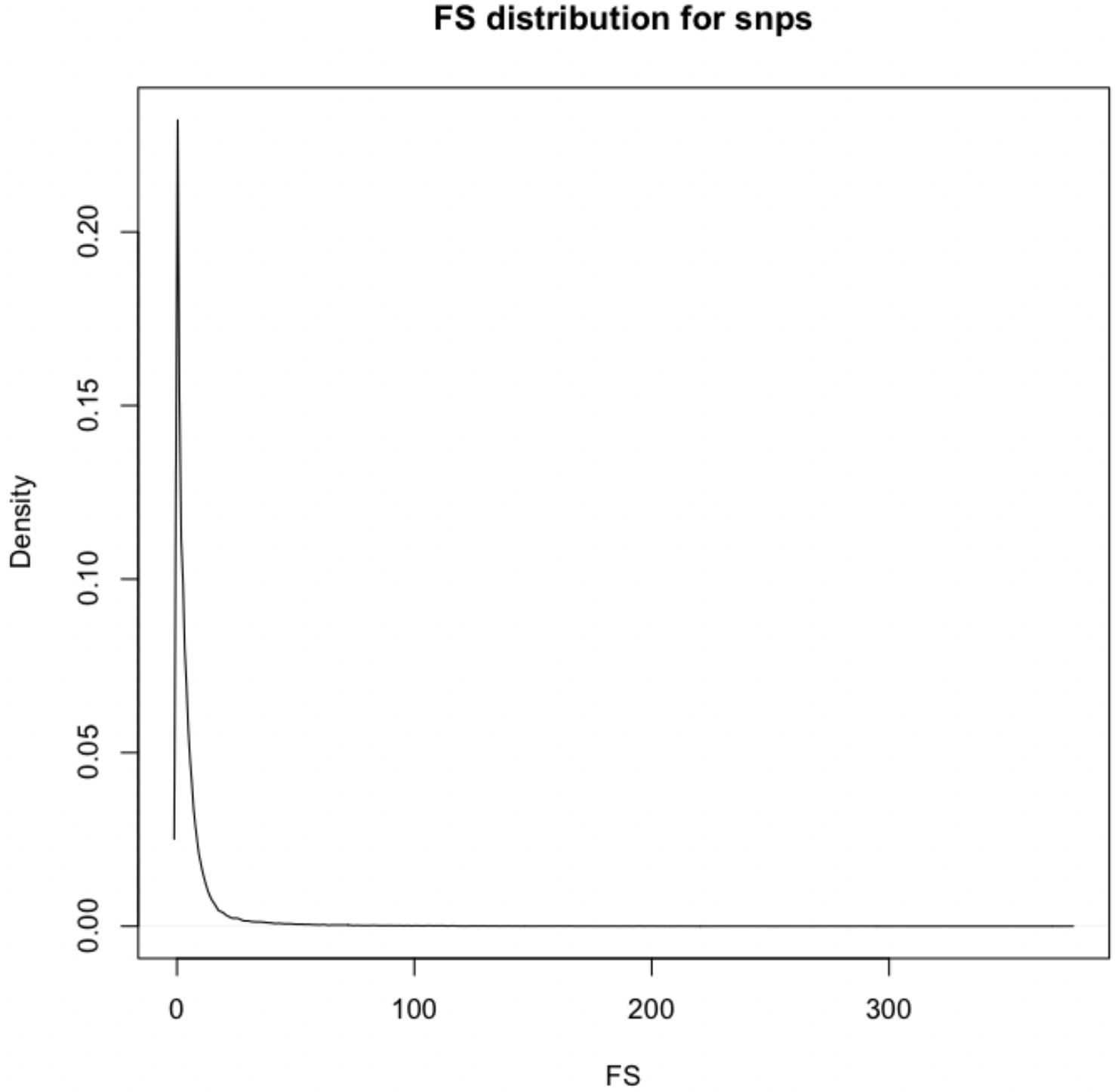
Visualize the distribution of values for an annotation

Repeat for other annotations, e.g. Fisher Strand Bias (FS)

```
dFS <- density(snps$FS,na.rm=T)
```

```
plot(dFS,main="FS distribution for snps", xlab="FS")
```

SB, SOR and FS are all related measures of strand bias. SB simple counts. FS and SOR are different statistical tests on SB. SOR is better for high coverage data

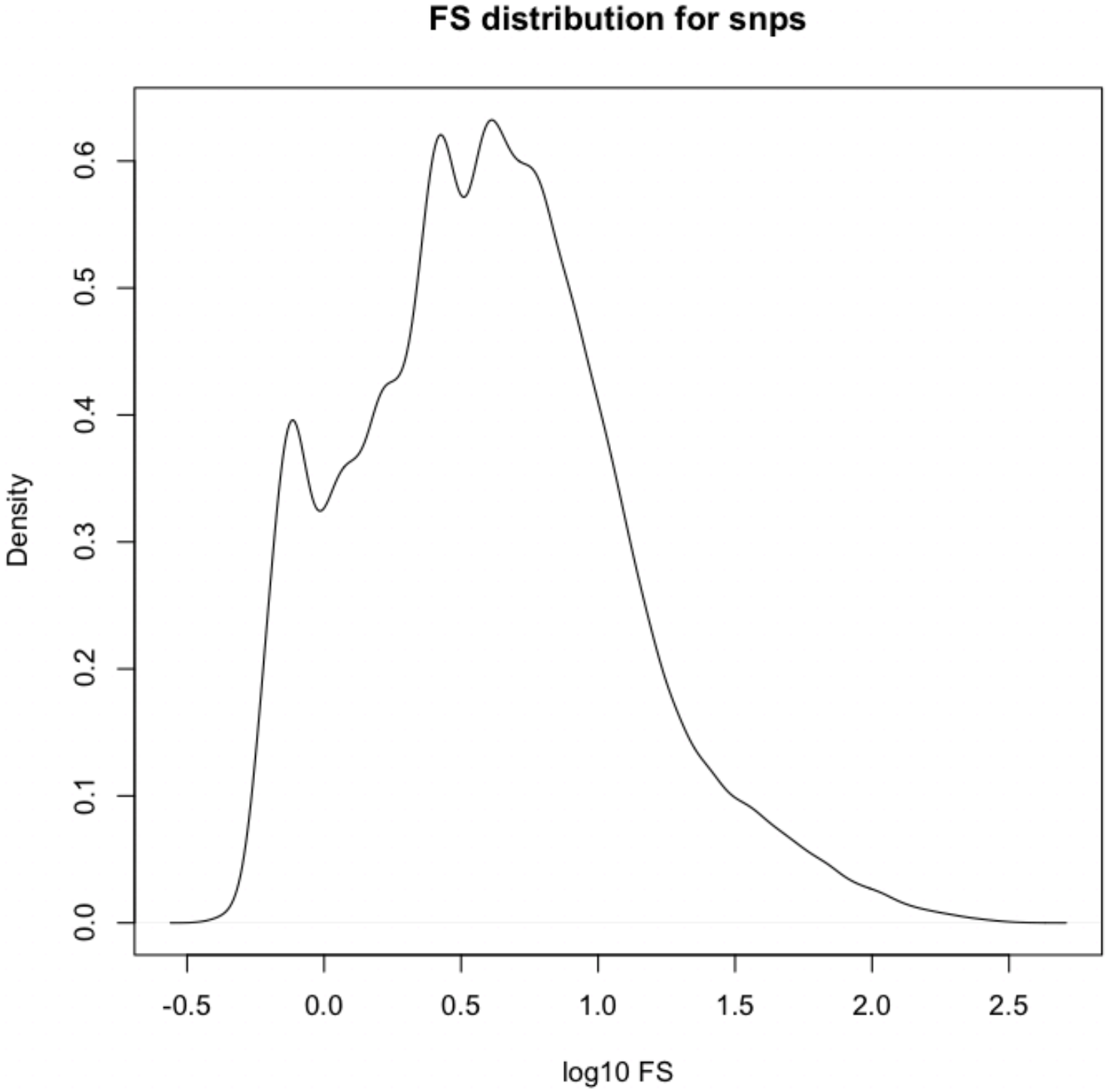

FS distribution for snps
Density
0.00
0.05
0.10
0.15
0.20
0
100
200
300
FS

Visualize the distribution of values for an annotation

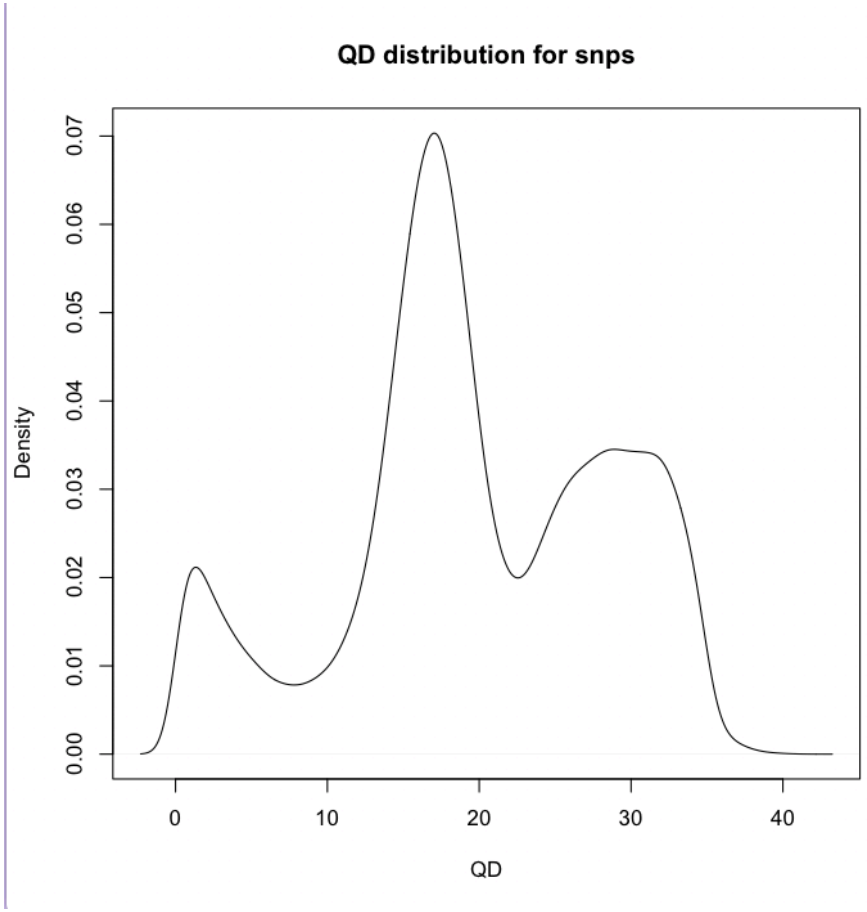
Redo FS with a log base 10 scale

```
dFS <- density(log10(snps$FS),na.rm=T)
```

```
plot(dFS,main="FS distribution for snps", xlab="log10 FS")
```


FS distribution for snps
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
-0.5
0.0
0.5
1.0
1.5
2.0
2.5
log10 FS

Select slices of variants for inspection



Lets check the variants that have QD<10, to help decide what cutoff to use.

```
java -Xmx1g -jar $GATK -R ref/ref.fasta -T SelectVariants -V sandbox/trio.snps.vcf \
-select "QD > 2.0" -select "QD < 10.0" -o sandbox/trio.snps.QD2-10.vcf
```

Select slices of variants for inspection

We now have a file with just the variants that have QD values between 2 and 10. We want to transfer this file to our local computer and use it to guide us to the variant positions in IGV, so we can inspect the read alignments for these regions. But how many variants are there?

```
grep –v '^#' sandbox/trio.snps.QD2–10.vcf |wc
```

10,481

Too many to inspect them all. We could load the whole file and ”randomly” jump around to different variants. But we tend not to be very good at that (e.g. we tend to go to similar places on each chromosome). So to get a truly random sample use the SelectVariants –fraction option

```
java -Xmx1g -jar $GATK -R ref/ref.fasta -T SelectVariants -V sandbox/trio.snps.QD2-10.vcf \
-fraction 0.002 -o sandbox/trio.snps.QD2-10.frac002.vcf
```

```
grep –v '^#' sandbox/trio.snps.QD2–10.frac002.vcf |wc
```

Now 23 or so variants.

Select slices of variants for inspection

We now have a file with 23 random variants that have QD values between 2 and 10. The usual method would be to transfer this file and proceed with the inspection. **But**, the workshop bam files only have data for one region on chromosome 20 (positions 10,000,000-10,200,000). So instead of the preferred method of taking a random set of variants, instead we redo the selection and only take the variants that fall in that region of chromosome 20.

```
java -Xmx1g -jar $GATK -R ref/ref.fasta -T SelectVariants -V sandbox/trio.snps.vcf -select "QD > 2.0" \
-select "QD < 10.0" -L 20:10,000,000-10,200,000 -o sandbox/trio.snps.QD2-10.chr20part.vcf
```

```
grep –v '^#' sandbox/trio.snps.QD2–10.chr20part.vcf |wc
```

12 variants

Select slices of variants for inspection

We want to transfer this file to our local computer and use it to guide us to the variant positions in IGV, so we can inspect the read alignments for these regions.

Use cyberduck, filezilla, scp or another method.

```
scp –pr \
joebruin@dtn2.hoffman2.idre.ucla.edu:gatkWorkshop/1702/data_bundle/data/sandbox/trio.snps.QD2-10.chr20part.vcf* ./
```

Notice the * at the end, so we transfer the vcf index file as well.

Select slices of variants for inspection

On your local computer,
Open IGV, choose the same reference genome as yesterday (**Human (1kg, b37+decoy).** Load the following files

- trio.snps.QD2-10.chr10part.vcf
- bams/NA12877_wgs_20.bam
- bams/NA12878_wgs_20.bam
- bams/NA12882_wgs_20.bam

Right click on each bam track and make sure color alignments is set to "insert size and pair orientation"
Zoom in to the alignments around each variant.

Go to chromosome 20 and zoom in on the leftmost variant at position 10,046,537.
Inspect the evidence for this and the other variants (select the vcf track, on the left, and use control-f and control-b to jump forward and backward between variants.

Select slices of variants for inspection

Hover over the variant to see annotation data – look at the QD for each one. Consider the evidence from the read alignments versus the QD value. It will be messy. There are a lot of factors that go into these calculations, and there is never a perfect cutoff. You choose the balance of false positives vs false negatives that works best for your project.

At position 10,046,537: QD=8.59, the evidence for heterozygosity in NA12878 is subtle, but hard to interpret as random.

At position 10,122,380: QD=8.41, but the evidence for heterozygosity in NA12877 look stronger.

Cluster of variants around 10,132,713. All with low QD. Seems to be caused by difficult alignments in the region. See the excess of soft-clipping in the region and the repetitive reference sequence.

- Using combinations of all of these tools, choose annotation cutoffs to apply for each set (snps and indels).
- You can begin with GATK's suggested **starting** values, given below. **BUT,** as they emphasize these numbers should be adjusted based on your data and the needs of your project.

SNPs

- QD < 2.0
- FS > 60.0
- MQ < 40.0
- MQRankSum < -12.5
- ReadPosRankSum < -8.0

Indels

- QD < 2.0
- FS > 200.0
- ReadPosRankSum < -20.0

*note, mapping quality filters are not recommended for indels

For our data set we'll use somewhat stringent filtering, SNPs: QD < 10.0, ReadPosRankSum < -4.0, MQRankSum < -5.0.

INDELS: QD < 10.0, ReadPosRankSum -10.0

Apply the Filtering

First SNPs…

```
java -Xmx1g -jar $GATK -T VariantFiltration  -R ref/ref.fasta -V
sandbox/trio.snps.vcf \
--filterExpression "QD < 10.0" --filterName QD10 \
--filterExpression "FS > 60.0" --filterName FS60 \
--filterExpression "MQ < 40.0"  --filterName MQ40 \
--filterExpression "MQRankSum < -5.0" --filterName MQRS-5 \
--filterExpression "ReadPosRankSum < -4.0" --filterName RPRS-4 \
-o sandbox/trio.snps.filtered.vcf 2> checkThisErrorOutput
```

Note, the cutoff values in the expression above must be given as doubles (0.0 rather than 0)

Standard Error is being redirected to a file because the output tends to be large. Every variant with a missing annotation throws an warning (even though they are expected). Check the file for other errors (maybeuse search and grep -v)

```
less  sandbox/checkThisErrorOutput
```

Take a look at the filtered variant output…

```
less  sandbox/trio.snps.filtered.vcf
```

Look for the header line, starting with #CHROM, and see that the FILTER field is the 7th column.

Note that now the FILTER field no longer has "." for any variants. It either has "PASS" or it has some other tag that indicates the variant has not passed that filter.

You can look at just the filter field with this command…

```
grep –v '##'  sandbox/trio.snps.filtered.vcf |awk '{print $7}' |less
```

“Filter” != “remove”

```
grep -v ’^#' sandbox/trio.snps.vcf |wc
```

```
grep -v ’^#' sandbox/trio.snps.filtered.vcf |wc
```

```
grep -v ’^#' sandbox/trio.snps.filtered.vcf |grep PASS |wc
```

A better way of counting PASSing variants…

```
grep -v '#' sandbox/trio.snps.filtered.vcf |awk '$7=="PASS"' |wc
```

If you want to remove the filtered variants, use –ef (exclude filtered). See below.

To look at just the set of filtered variants

You may want to browse through them in IGV to make sure you're happy with the cutoffs.
Many ways to get there. Here, first we're selecting the set of non-filtered (i.e. PASSing) variants using the tag –ef (--excludeFiltered)

```
java –Xmx1g –jar $GATK –R ref/ref.fasta  –T SelectVariants \
–V sandbox/trio.snps.filtered.vcf –ef \
–o sandbox/trio.snps.filtered.ef.vcf
```

Then we use the parameter –discordance to select all the variants that are not in the PASSing set.

```
java –Xmx1g –jar $GATK –R ref/ref.fasta  –T SelectVariants \
–V sandbox/trio.snps.filtered.vcf \
––discordance sandbox/trio.snps.filtered.ef.vcf \
–o sandbox/trio.snps.filteredOnly.vcf
```

vcftools FILTER summary

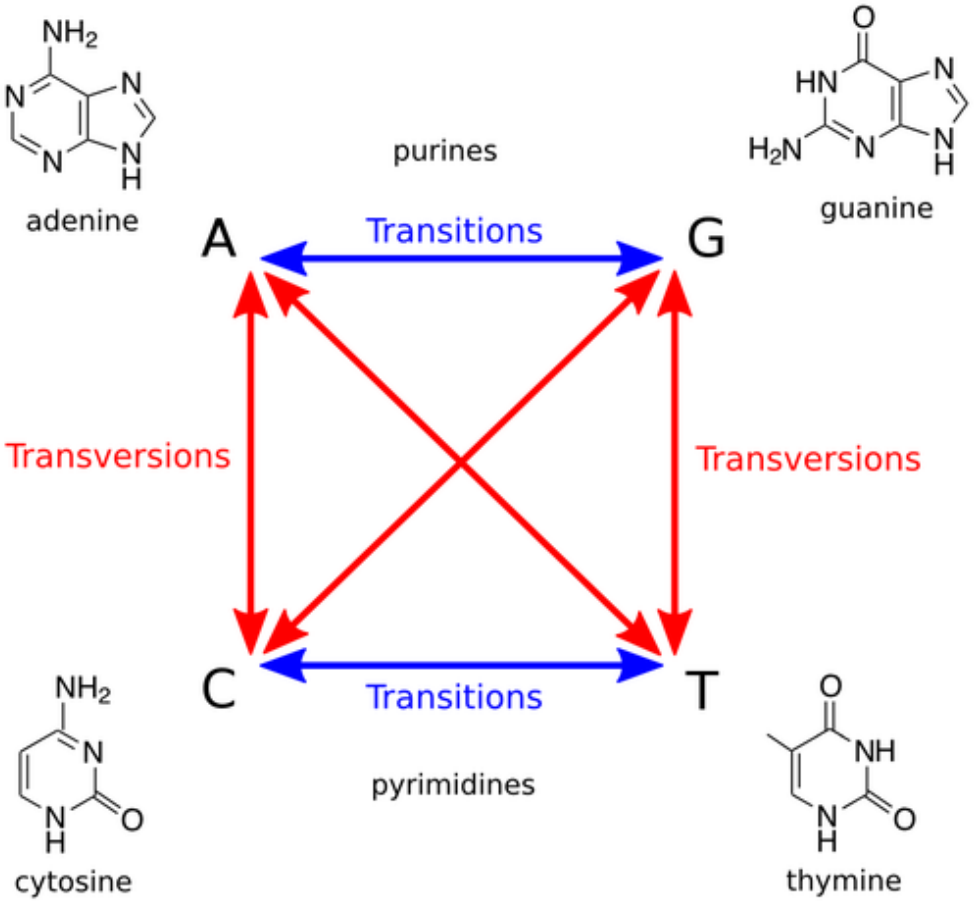
vcftools is a package of tools for working with vcf files. A lot of good stuff. See the documentation at:

https://vcftools.github.io/man_latest.html

Here we'll use the FILTER summary option to see how many variants were filtered with each filter and the Transition/Transversion ratio for each.

```
vcftools --vcf sandbox/trio.snps.filtered.vcf --FILTER-summary --out sandbox/trio.snps.filtered
```

Transition/Transversion Ratios



Random: Ts/Tv = 0.5
Biology: Ts/Tv >= 2 (usually)

Since there are twice as many possible transversions (Tv) as transitions (Ts), random changes would give Ts/Tv = 0.5. However biology tends to favor transitions over transversions, so when we see high Ts/Tv it gives us confidence the variants are real (biological)

vcftools FILTER summary

```
less sandbox/trio.snps.filtered.FILTER.summary
```

FILTER	N_VARIANTS	N_Ts	N_Tv	Ts/Tv
PASS	99946	69084	30862	2.23848
QD10	9988	5905	4083	1.44624
MQRS-5;QD10	3413	2038	1375	1.48218
MQRS-5	2755	1710	1045	1.63636
MQ40	2533	1628	905	1.7989
MQ40;QD10	1496	900	596	1.51007
FS60;QD10	629	233	396	0.588384

…

The lower values for the non-PASSing sets are encouraging. FS60 especially seems to capture more or less random (with respect to the bases involved) changes.

Apply INDEL filters

```
java -Xmx1g -jar $GATK -T VariantFiltration  -R ref/ref.fasta -V sandbox/trio.indels.vcf \
--filterExpression "QD < 10.0" --filterName QD10 \
--filterExpression "FS > 200.0" --filterName FS200 \
--filterExpression "ReadPosRankSum < -10.0" --filterName RPRS-10 \
-o sandbox/trio.indels.filtered.vcf 2> sandbox/checkThisErrorOutput
```

- Readjust filter values based on your visual inspection, and repeat.
- You won’t be able to achieve perfection!
- Note again, you are keeping all of the called variants and simply determining which ones to tag as filtered. Future users can reset the cutoffs as needed.

When you've settled on a final call set, recombine the SNPs and INDELs

```
java -Xmx1g -jar $GATK -T CombineVariants -R ref/ref.fasta \
-V sandbox/trio.snps.filtered.vcf \
-V sandbox/trio.indels.filtered.vcf \
-o sandbox/trio.filtered.vcf --assumeIdenticalSamples
```

```
grep -v '^#' sandbox/trio.filtered.vcf |wc
```

```
grep -v '^#' sandbox/trio.filtered.vcf |grep –c PASS
```

Note, if you wish to do BQSR on non-human samples, you can use the above filtered file (but generated from the whole genome) as the "known" variant input. This set does not need to be precise since the amount of error in the reads usually far exceeds the number of variants that are called, and true positives should not generally exhibit typical BQSR captured patterns.

You expect to see a variant at a specific site, but it's not getting called

https://www.broadinstitute.org/gatk/guide/article?id=1235

1. **Generate the bamout and compare it to the input bam**
2. **Check the base qualities of the non-reference bases**
   - **Min base qual applied. Note, DP is unfiltered depth**
3. **Check the mapping qualities of the reads that support the non-reference allele(s)**
4. **Check how many alternate alleles are present**