Genomic Sequence Data Analysis

Alignment of Illumina sequencing reads and variant calling

Robert Bukowski, Cheng Zhou, Qi Sun

Bioinformatics Facility, Institute of Biotechnology

Slides: http://biohpc.cornell.edu/lab/doc/Variant_workshop.pdf

Exercise instructions: http://biohpc.cornell.edu/lab/doc/Variant_exercise.pdf

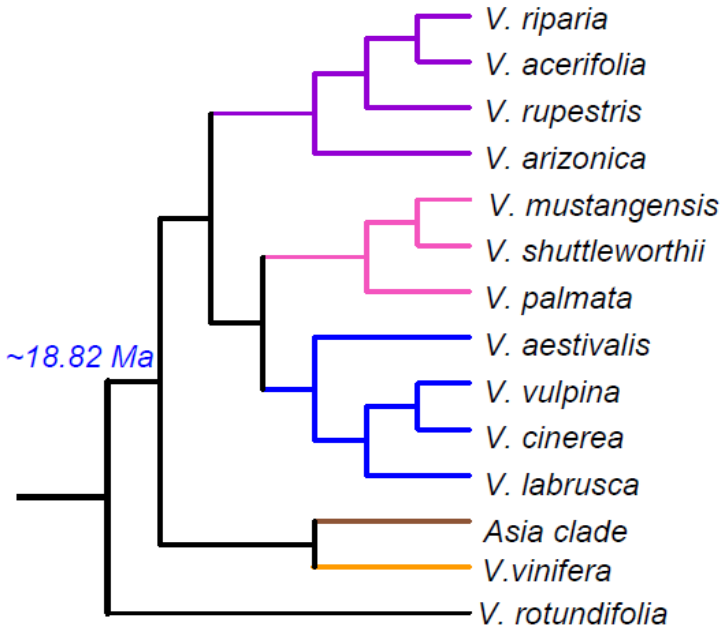
Workshop contact: brc_bioinformatics@cornell.edu

How to sequence genomes of every grape vine (or every cat) in the world?

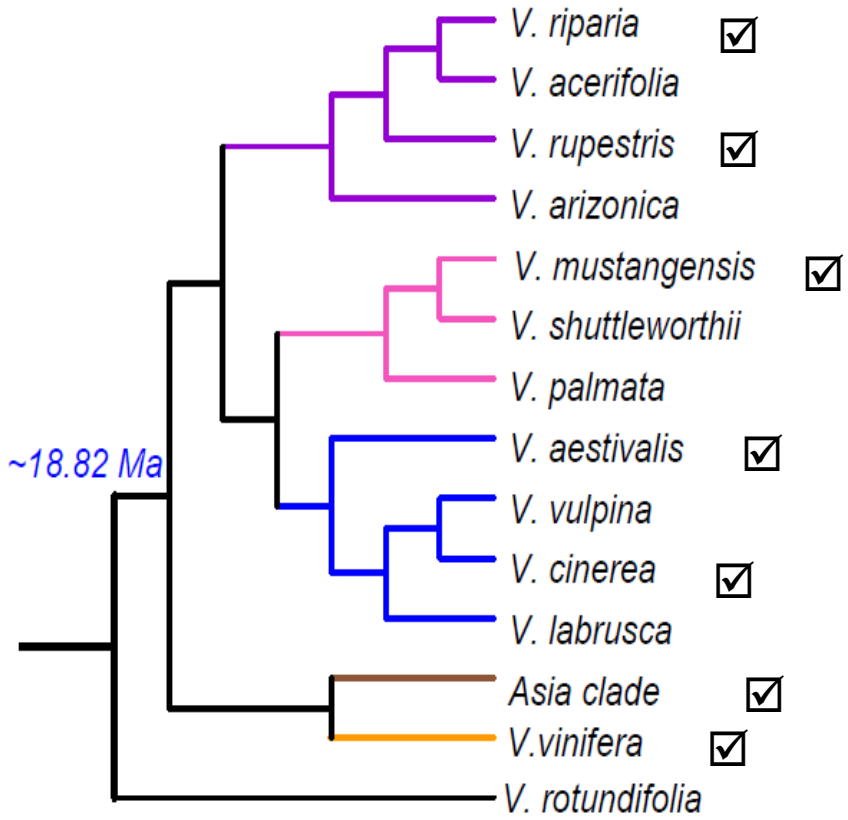
- Cost effectively

How to represent genomes of every grape vine (or every cat) in the world?

- 2D Matrix (vcf) vs Graph (vg, fastg)



Tier 1. *De novo* genome assembly of the reference panel (Covered in 3rd week)



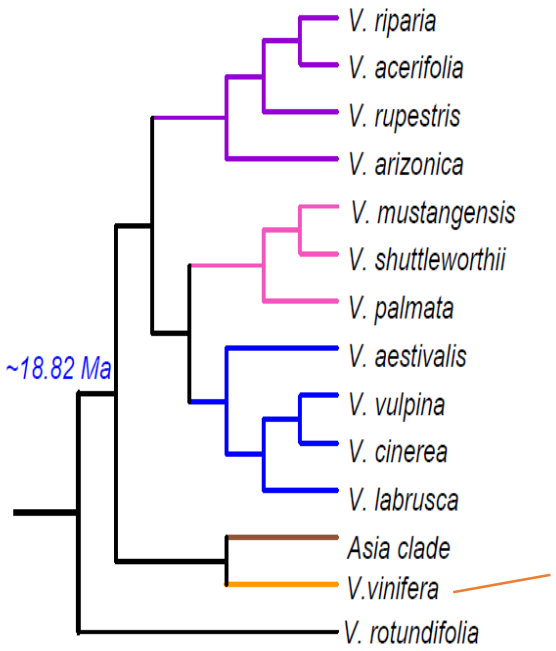
Select a panel of individuals to represent all genetic diversity in grapes;

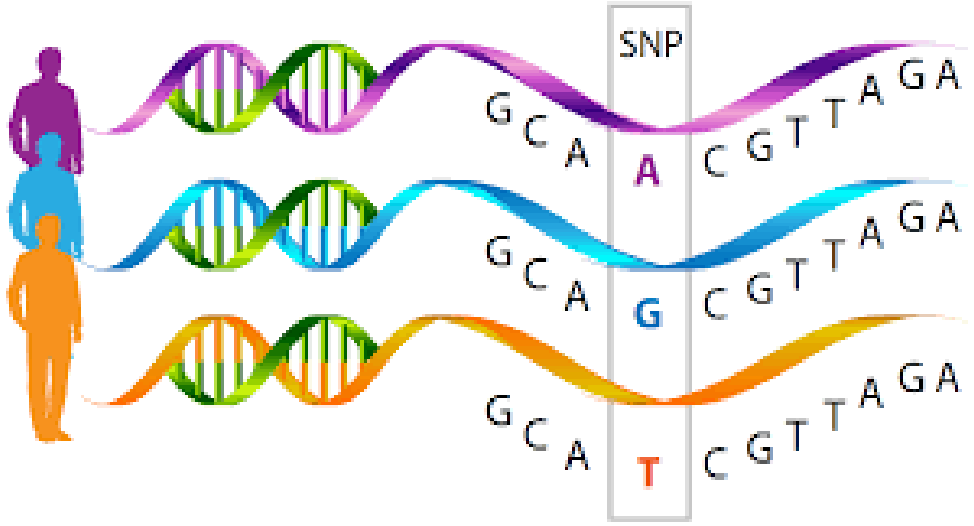
- All gene space;
- All chromosomal structural variations, e.g. large insertions/deletions and translocations

Sequence and assemble using the best technologies we have today, e.g. PacBio, Nanopore, BioNano, Hi-C, et al.

Tier 2. Whole Genome Shotgun (WGS), a.k.a. re-sequencing (Covered in 1st week)

Reference . . . G C A A C G T T A GA . . .





WGS on population in V. vinifera

Representation of Genomes

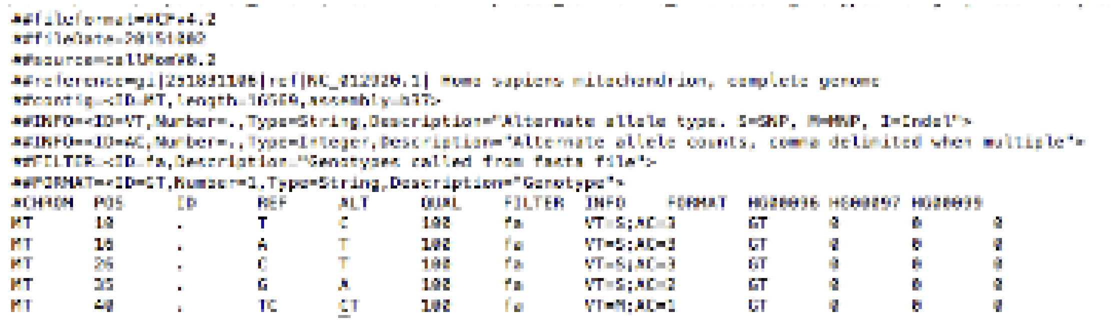
De novo assembly — WGS or Targeted

FASTA File — **VCF File**

Current state

Flat view:

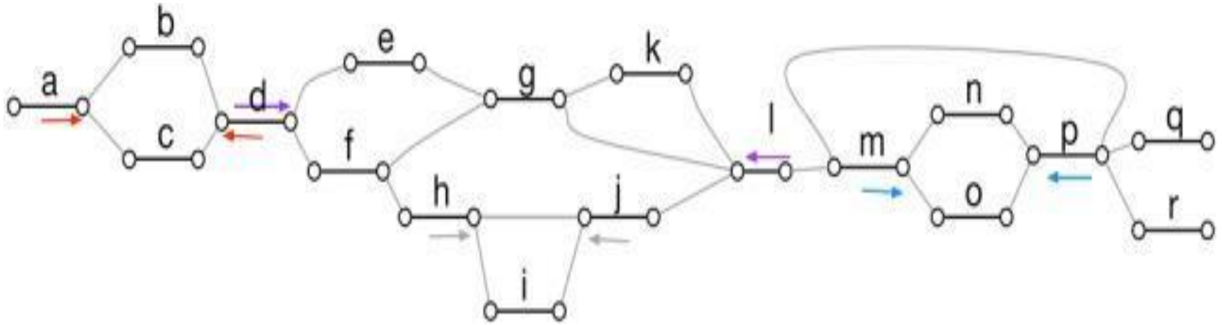
```
>chromosome51
GATGGGATTGGGGTTTTCCCCTCCCATGTGCTCAAGACTGGCGCTAAAAGTTTTGAGCTTCTCAAAAGTC
TAGAGCCACCGTCCAGGGAGCAGGTAGCTGCTGGGCTCCGGGGACACTTTGCGTTCGGGCTGGGAGCGTG
CTTTCCACGACGGTGACACGCTTCCCTGGATTGGCAGCCAGACTGCCTTCCGGGTCACTGCCATGGAGGA
GCCGCAGTCAGATCCTAGCGTCGAGCCCCCTCTGAGTCAGGAAACATTTTCAGACCTATGGAAACTACTT
CCTGAAAACAACGTTCTGTCCCCCTTGCCGTCCCAAGCAATGGATGATTTGATGCTGTCCCCGGACGATA
TTGAACAATGGTTCACTGAAGACCCAGGTCCAGATGAAGCTCCCAGAATGCCAGAGGCTGCTCCCCCCGT
GGCCCCTGCACCAGCAGCTCCTACACCGGCGGCCCCTGCACCAGCCCCCTCCTGGCCCCTGTCATCTTCT
```



```
##fileformat=VCFv4.2
##INFO=<ID=VT,Number=.,Type=String,Description="Alternate allele type. S=SNP, M=MNP, I=Indel">
##FILTER=<ID=fa,Description="Genotypes called from fasta file">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
#CHROM  POS  ID  REF  ALT  QUAL  FILTER  INFO      FORMAT
MT           .   T    C    100   fa      VT=S;AC=  GT      0  0  0
MT           .   A    T    100   fa      VT=S;AC=  GT      0  0  0
MT           .   C    T    100   fa      VT=S;AC=  GT      0  0  0
MT           .   G    A    100   fa      VT=S;AC=  GT      0  0  0
MT           .   TC   CT   100   fa      VT=M;AC=1 GT      0  0  0
```

Future

Graph view:



Paten et al. *Genome Res*. 2017, May; **27(5)**: 665-676.

Tier 3. Targeted or skim sequencing

(Not covered in this workshop)

Technologies

- GBS / RAD
- Amplicon
- Exome
- SKIM

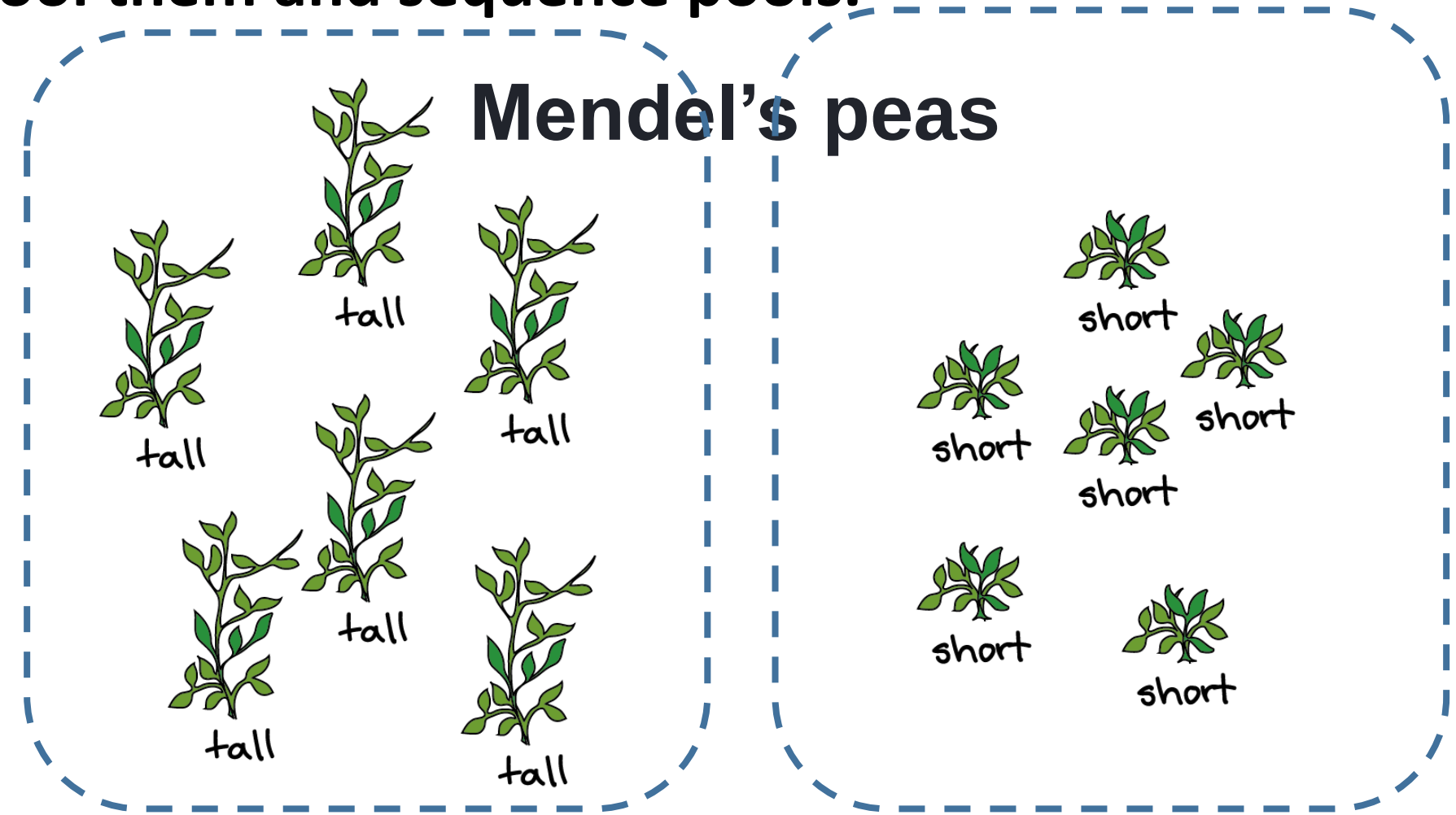
Purposes

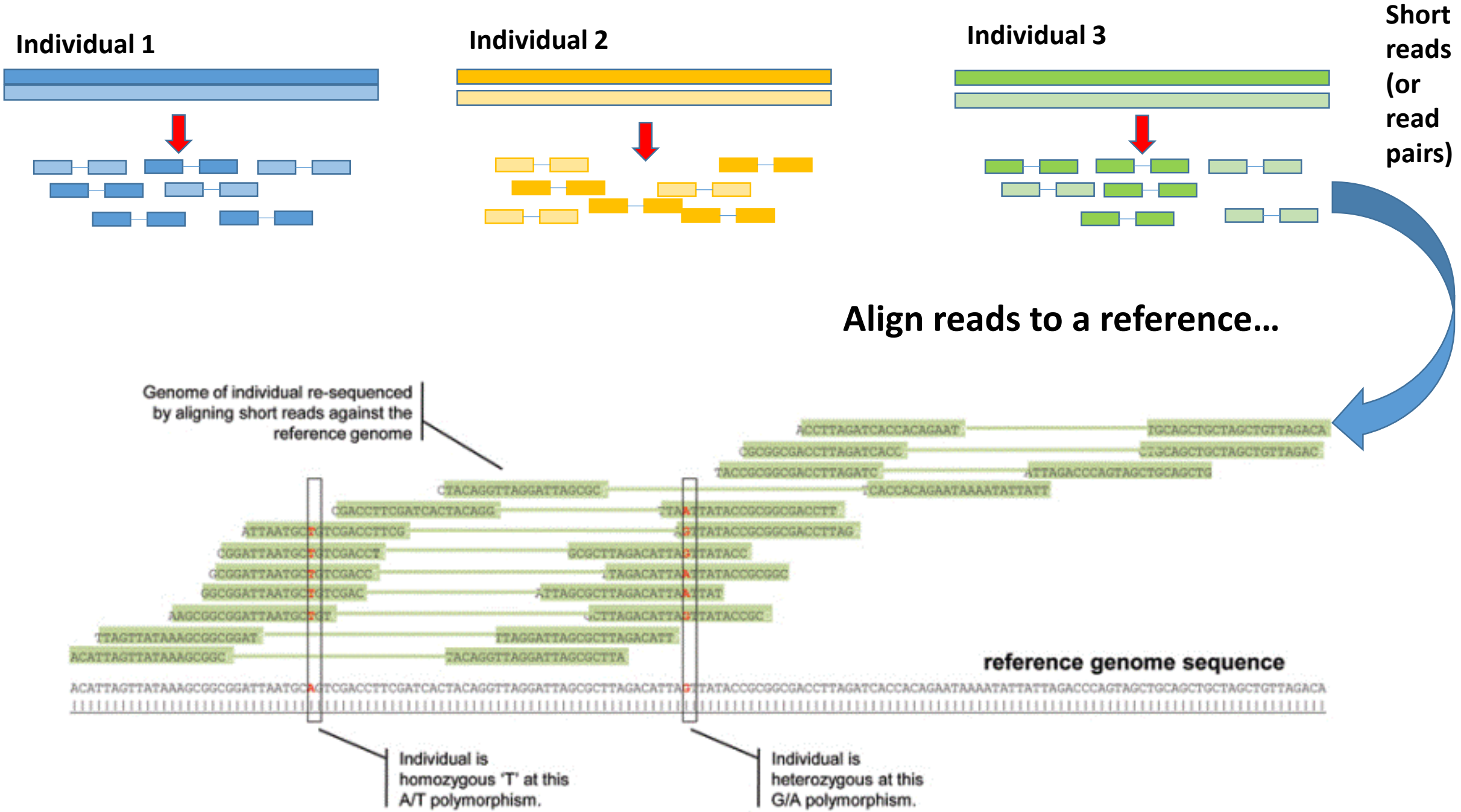
- Population structure
- Genome construction by imputing haplotypes assembled in tier 1 & 2

Alternatively, if you do not have the budget to sequence each individual genome?

(Bulked segregant analysis covered in 2nd week)

Pool them and sequence pools.




Individual 1
Individual 2
Individual 3
Short reads (or read pairs)
Align reads to a reference…
Genome of individual re-sequenced by aligning short reads against the reference genome
reference genome sequence
Individual is homozygous 'T' at this A/T polymorphism.
Individual is heterozygous at this G/A polymorphism.

Expected output: table of genotypes at variant sites

Variant site chr and position	Indiv1	Indiv2	Indiv3	….
site1	AA	AA	AC	…
site2	GT	missing	TT	…
…	…	…	…	…
siteN	CC	CC	AA	…

Table above is very schematic. In reality, genotypes are recorded in **VCF** format (**Variant Call Format**)

Additional information about variants is also produced and recorded in VCF (such as call quality info)

More about VCF – coming soon

State of the art: GATK from Broad Institute

software.broadinstitute.org/gatk/

gatk | Best Practices | User Guide | Blog | Forum | Events | Download | Search

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

Best Practices
Pipelines optimized for accuracy and performance

Blog
Announcements and progress updates

Events calendar
Webinars, workshops, conferences

User Guide
Detailed documentation, tutorials and resources

Forum
Ask our team for help and report issues

Run on Cloud | Run on HPC

Download GATK 4.1.4.0

GATK 4

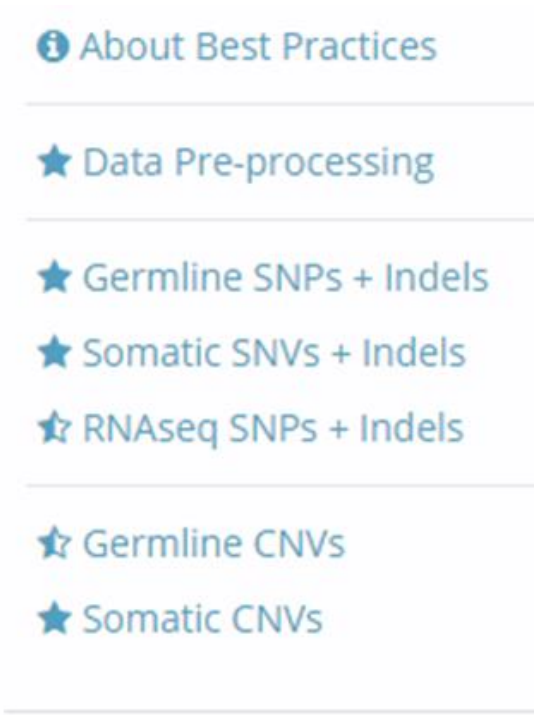
- Developed in conjunction with 1000 (human) genomes project
- Package of command-line tools (written mostly in **Java**):

Copy Number Variant Discovery
Coverage Analysis
Diagnostics and Quality Control
Intervals Manipulation
Metagenomics
Other
Read Data Manipulation
Reference
Short Variant Discovery
Structural Variant Discovery
Variant Evaluation and Refinement
Variant Filtering
Variant Manipulation

Base Calling

Read Filters
Variant Annotations

- "Best practices" protocols



GATK 4 methodologically very similar to v 3.x

Code re-organized and optimized for large-scale projects in the cloud (but works on smaller scale as well)

Multi-threading parallelization using **Spark** (bundled with GATK) – most tools still in Beta stage

Developed in collaboration with Google and Intel

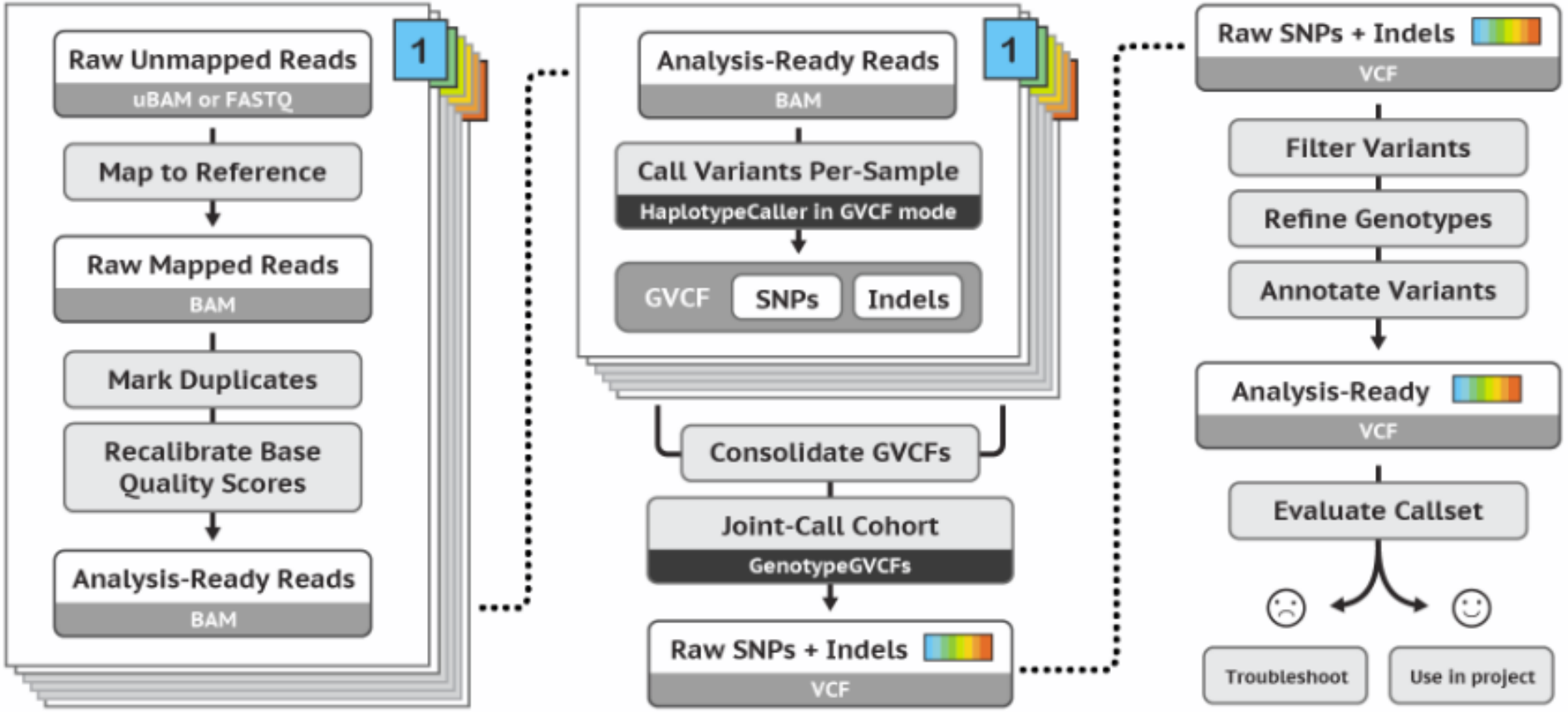
Detailed information at https://software.broadinstitute.org/gatk/ (visit **User's Guide → Tool documentation** and **Forums**)

In this workshop: use of GATK 4 for germline short variant calling in BioHPC environment

“Best Practices” for DNA-Seq variant calling

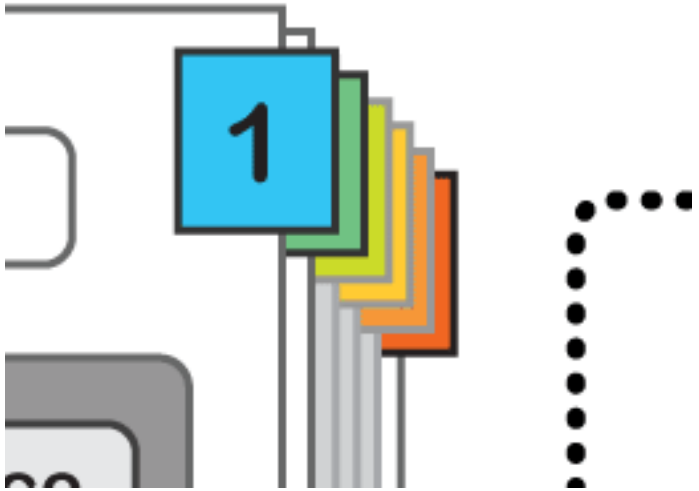
Purpose

Identify germline short variants (SNPs and Indels) in one or more individuals to produce a joint callset in VCF format.



?

Best Practices for DNA-Seq variant calling



What are the colored tabs?

Each tab stands for **a FASTQ** file (SE case) or a **pair of FASTQ files** (PE case) with reads from **one sample, one Illumina lane, one library (i.e. one read group)**

A lane may contain reads from
- a single sample/library, OR…
- multiple samples/libraries (multiplexing)

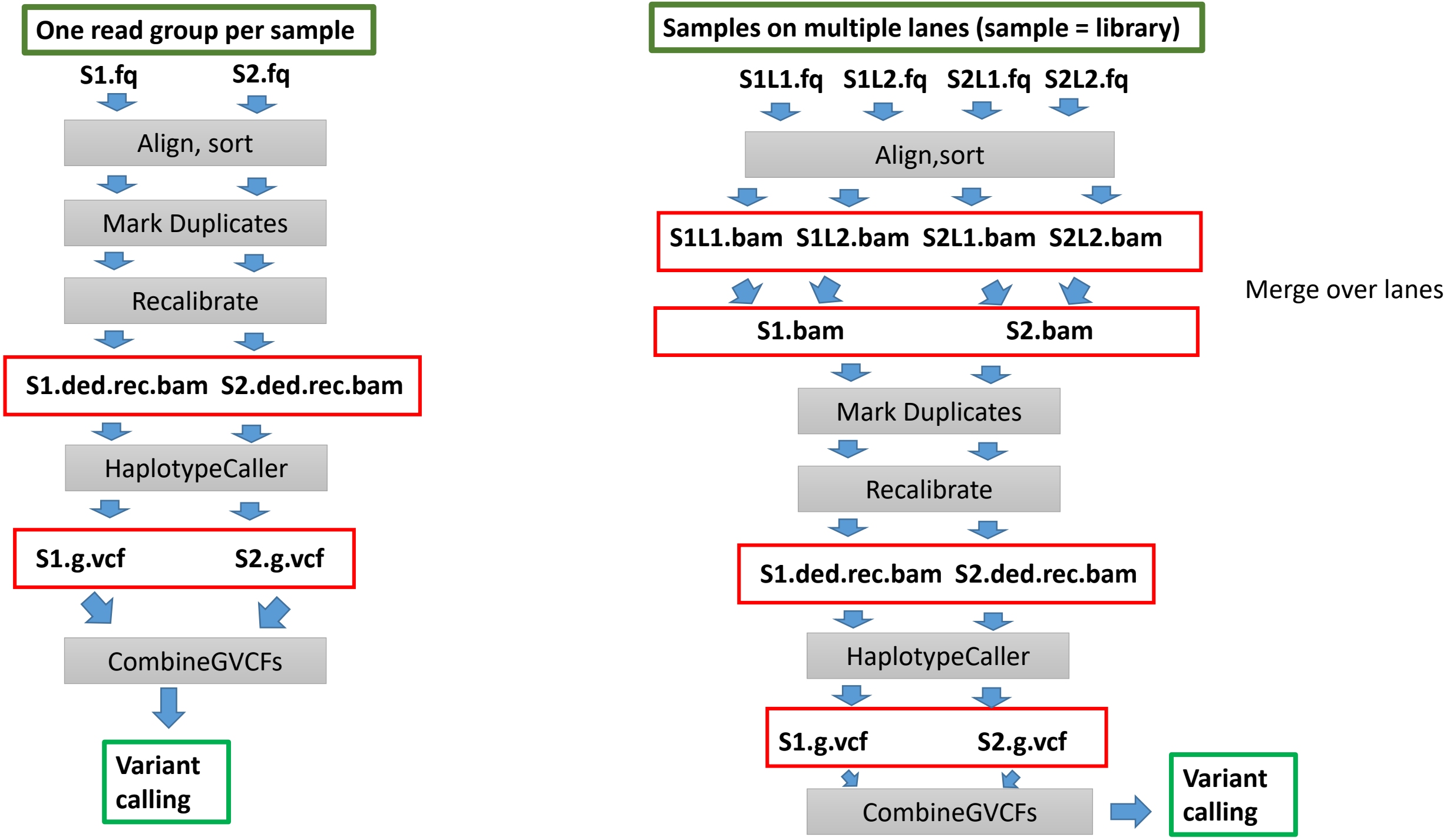
Reads from one sample/library may initially be in
- One FASTQ file, OR…..
- Multiple FASTQ files

What's good for computational efficiency: process all datasets independently – **in parallel**

… is bad for accuracy: GATK tools prefer large datasets as possible - long compute times and loss of parallelism

- **Mark Duplicates** works best if given **all reads from a library** (sometimes scattered among files)
- **Haplotype calling** (discussed later) works best with all reads from a sample, and would be delighted to use all reads available (whole cohort)

Compromises have to be made


One read group per sample
S1.fq
S2.fq
Align, sort
Mark Duplicates
Recalibrate
S1.ded.rec.bam
S2.ded.rec.bam
HaplotypeCaller
S1.g.vcf
S2.g.vcf
CombineGVCFs
Variant calling
Samples on multiple lanes (sample = library)
S1L1.fq
S1L2.fq
S2L1.fq
S2L2.fq
Align,sort
S1L1.bam
S1L2.bam
S2L1.bam
S2L2.bam
Merge over lanes
S1.bam
S2.bam
Mark Duplicates
Recalibrate
S1.ded.rec.bam
S2.ded.rec.bam
HaplotypeCaller
S1.g.vcf
S2.g.vcf
CombineGVCFs
Variant calling

Input: paired-end (PE) reads

Paired-end case: we have two "**parallel**" FASTQ files – one for "**left**" and another for "**right**" end of the fragment:

First sequence in "left" file

```
@HWI-ST896:156:D0JFYACXX:5:1101:1652:2132 1:N:0:GATCAG
ACTGCATCCTGGAAAGAATCAATGGTGGCCGGAAAGTGTTTTTCAAATACAAGAGTGACAATGTGCCCTGTTGTTT
+
ACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBC?CCCCCCCCC@@CACCCCCACCCCCCCCCCCCCCCCCCCCCC
```

First sequence in "right" file

```
@HWI-ST896:156:D0JFYACXX:5:1101:1652:2132 2:N:0:GATCAG
CTCAAATGGTTAATTCTCAGGCTGCAAATATTCGTTCAGGATGGAAGAACATTTTCTCAGTATTCCATCTAGCTGC
+
C<CCCCCCCACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBCCCCCCCCCCCCCCCCACCCCCACCC=
```

Phred base quality score

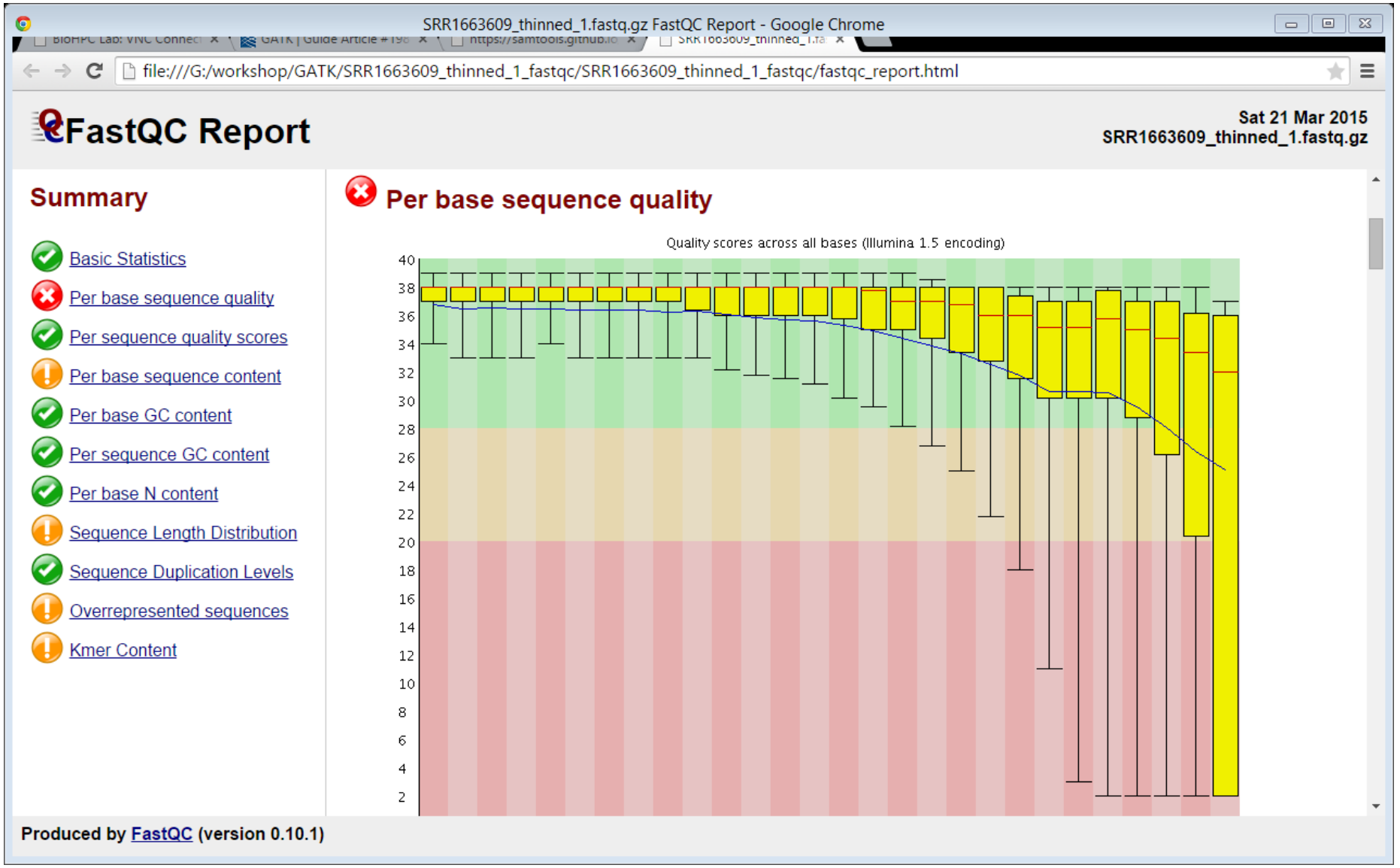
The two ends come from **opposite strands** of the fragment being sequenced

End 1 End 2

For example, "C" stands for: 67 – 33 = 34, i.e., probability of the base (here: C) being miscalled is $\mathbf{10^{-3.4}}$.

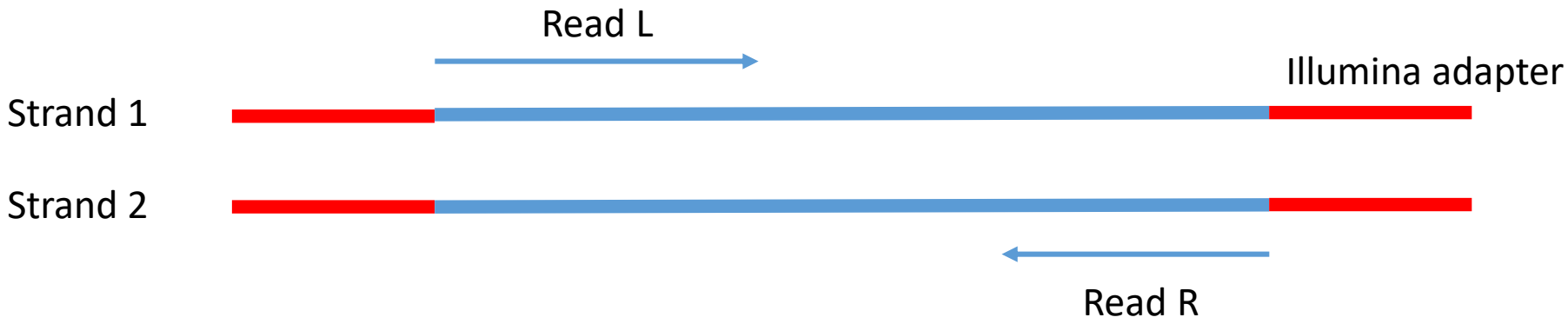
Base qualities are typically used in genotype likelihood models – they better be accurate!

Read quality assessment with fastqc

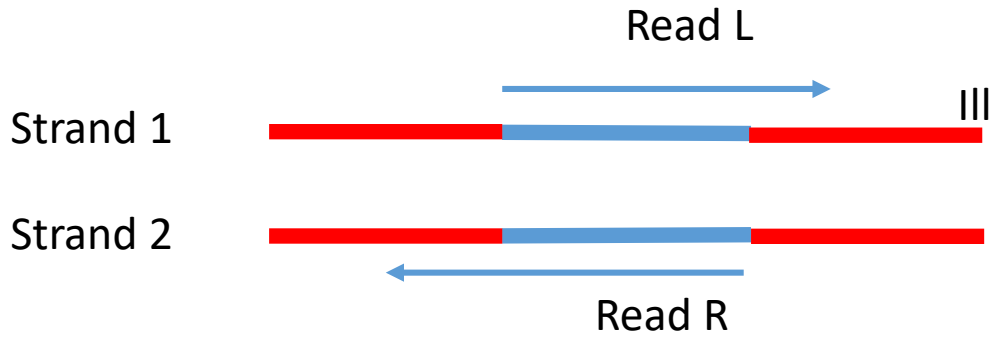


Run the command: **`fastqc my_file.fastq.gz`** to generate html report

Sequencing a long fragment



Sequencing a short fragment



Read-through: sequenced reads cut into adapters

Tool for removal of low-quality portions of reads and/or adapter sequences:

- **Trimmomatic** (http://www.usadellab.org/cms/?page=trimmomatic, https://biohpc.cornell.edu/lab/userguide.aspx?a=software&i=53#c)
- Adapter removal not that important in alignment-based methods

Alignment is fundamentally hard……

- Genomes being re-sequenced not sufficiently similar to reference
  - Not enough reads will be mapped
  - Reads originating from parts of genome absent from reference will align somewhere anyway, leading to **false SNPs**

- Some reads cannot be mapped unambiguously in a single location (have low **Mapping Quality**)
  - if reads too short
  - reads originating from paralogs or repetitive regions
  - Having paired-end (PE) data helps

- Alignment of some reads may be ambiguous even if placement on reference correct (SNPs vs indels)
  - Need local multi-read re-alignment or local haplotype assembly (expensive!)

- Sequencing errors
  - Easier to handle and/or build into variant-calling models

Choosing a good aligner is important

Ambiguity of alignment at indel sites

Reference `CTTTAGTTTCTTTT----CTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC`

Reads
```
CTTTAGTTTCTTTT----GCCGCTTTCTTTCTTTCTT
CTTTAGTTTCTTTT----GCCGCTTTCTTTCTTTCTT
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
```

For these reads, aligner preferred to make a few SNPs rather than insertion

For these reads, insertion was a better choice

Aligner, like BWA, works on one read (fragment) at a time, does not see a bigger picture…)

But we can try to shift things around a bit:

Reference `CTTTAGTTTCTTTT----CTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC`

Reads
```
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTT
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTT
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
```

This looks better !

Only seen after aligning all (at least some) reads!

Strategies to deal with indels

Local multiple-sequence re-alignment of all reads spanning an putative indel (GATK 3)

- performed prior to variant calling
- computationally expensive

Local read assembly into haplotypes (**`HaplotypeCaller`** in GATK 3, 4)

- naturally gets rid of reads with sequencing errors
- used along whole genome (not only indels)
- computationally expensive
- standard in modern variant calling pipelines

BWA mem – aligner of choice in GATK

- **BWA** = Burrows Wheeler Aligner (uses BW transform to compress data)
- **MEM** = Maximal Exact Match (how alignment "seeds" are chosen)

- **Performs local alignment** (rather than end-over-end)
  - Can clip ends of reads, if they do not match
  - Can split a read into pieces, mapping each separately (the best aligned piece is then the primary alignment)

- **Performs gapped alignment**

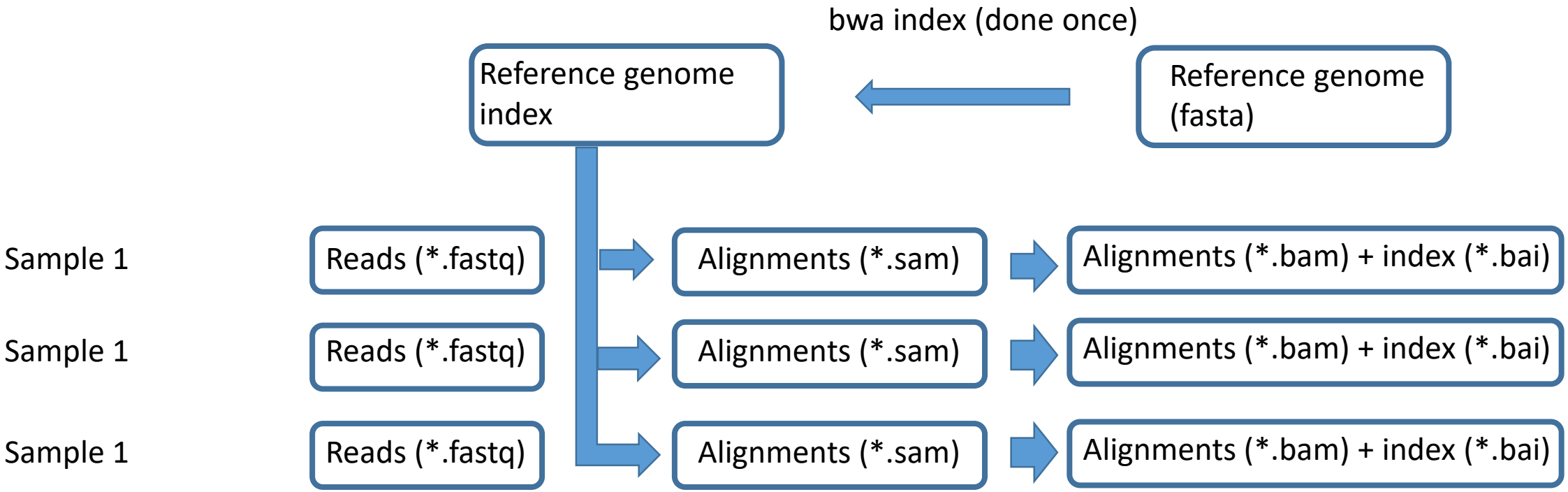
- **Utilizes PE reads** to improve mapping

- **Reports only one alignment** for each read
  - If ambiguous, one of the equivalent best locations is chosen at random
  - Ambiguously mapped reads are reported with low **Mapping Quality**
- Works well for reads 70bp to several Mbp

- Time scales linearly with the size of query sequence (at least for exact matches)
- Moderate memory requirement (few GB of RAM to hold reference genome)

Li H. and Durbin R. To cite BWA: Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler Transform. Bioinformatics, 25:1754-60. [PMID: 19451168]

Running BWA mem



Programs involved:

bwa mem

samtools view
samtools sort
samtools index

Alignment

Alignment post-processing steps

Running BWA mem: align your reads

For PE reads:

```
bwa mem -M -t 4 \
-R '@RG\tID:C6C0TANXX_2\tSM:ZW177\tLB:ZW177lib\tPL:ILLUMINA' \
./genome_index/genome.fa    \
sample1reads_1.fastq.gz  sample1reads_2.fastq.gz  >  sample1.sam
```

(SE version the same – just specify one read file instead of two)

What does it all mean:

- **`-M`**: if a read is split (different parts map to different places) mark all parts other than main as "secondary alignment" (technicality, but important for GATK which ignores secondary alignments)
- **`-R`**: add **Read Group** description (more about it in a minute)
- **`-t 4`**: run of 4 CPU cores. If CPUs available, bwa mem scales well up to about 12 CPU cores.
- **`./genome_index/genome.fa`**: points to BWA index files (**`genome.fa.*`**)
- Output (i.e., alignments) will be written to the file **`sample1.sam`**. As the name suggests, it will be in **SAM format**.

BWA mem command: define Read Group

```
-R '@RG\tID:C6C0TANXX_2\tSM:ZW177\tLB:ZW177lib\tPL:ILLUMINA'
```

What will this option do?

The SAM/BAM file header will contain a line (TAB-delimited) defining the group:

`@RG`	`ID:C6C0TANXX_2`	`SM:ZW177`	`LB:ZW177lib`	`PL:ILLUMINA`
	Unique ID of a collection of reads sequenced together, typically: Illumina lane +(barcode or sample)+library	Sample name	DNA prep Libray ID	Sequencing platform

Each alignment record will be marked with **Read Group ID** (here: `C6C0TANXX_2`), so that programs in downstream analysis know where the read is from.

Read groups, sample and library IDs are important for GATK operation!

Each **READ GROUP** contains reads from **one sample, one library, one flowcell_lane**
A library may be sequenced multiple times (on different flowcell_lanes)
Sample may be sequenced multiple times, on different lanes and from different libraries

```
Dad's data:
@RG     ID:FLOWCELL1.LANE1      PL:ILLUMINA     LB:LIB-DAD-1 SM:DAD      PI:200
@RG     ID:FLOWCELL1.LANE2      PL:ILLUMINA     LB:LIB-DAD-1 SM:DAD      PI:200
@RG     ID:FLOWCELL1.LANE3      PL:ILLUMINA     LB:LIB-DAD-2 SM:DAD      PI:400
@RG     ID:FLOWCELL1.LANE4      PL:ILLUMINA     LB:LIB-DAD-2 SM:DAD      PI:400

Mom's data:
@RG     ID:FLOWCELL1.LANE5      PL:ILLUMINA     LB:LIB-MOM-1 SM:MOM      PI:200
@RG     ID:FLOWCELL1.LANE6      PL:ILLUMINA     LB:LIB-MOM-1 SM:MOM      PI:200
@RG     ID:FLOWCELL1.LANE7      PL:ILLUMINA     LB:LIB-MOM-2 SM:MOM      PI:400
@RG     ID:FLOWCELL1.LANE8      PL:ILLUMINA     LB:LIB-MOM-2 SM:MOM      PI:400

Kid's data:
@RG     ID:FLOWCELL2.LANE1      PL:ILLUMINA     LB:LIB-KID-1 SM:KID      PI:200
@RG     ID:FLOWCELL2.LANE2      PL:ILLUMINA     LB:LIB-KID-1 SM:KID      PI:200
@RG     ID:FLOWCELL2.LANE3      PL:ILLUMINA     LB:LIB-KID-2 SM:KID      PI:400
@RG     ID:FLOWCELL2.LANE4      PL:ILLUMINA     LB:LIB-KID-2 SM:KID      PI:400
```

Read Group assignment: multiplexed lanes

One flowcell: **HL5WNCCXX**, **two lanes** (2 and 3), each with **samples A and B (2-plex)** from library **my_lib**

@RG	ID:HL5WNCCXX_2_A	SM:A	LB:mylib	PL:ILLUMINA
@RG	ID:HL5WNCCXX_3_A	SM:A	LB:mylib	PL:ILLUMINA
@RG	ID:HL5WNCCXX_2_B	SM:B	LB:mylib	PL:ILLUMINA
@RG	ID:HL5WNCCXX_3_B	SM:B	LB:mylib	PL:ILLUMINA

Anatomy of a SAM file

```
@SQ     SN:chr2L        LN:23011544
@SQ     SN:chr2LHet     LN:368872
@SQ     SN:chr2R        LN:21146708
@SQ     SN:chr2RHet     LN:3288761
@SQ     SN:chr3L        LN:24543557
@SQ     SN:chr3LHet     LN:2555491
@SQ     SN:chr3R        LN:27905053
@SQ     SN:chr3RHet     LN:2517507
@SQ     SN:chr4         LN:1351857
@SQ     SN:chrM         LN:19517
@SQ     SN:chrX         LN:22422827
@SQ     SN:chrXHet      LN:204112
@SQ     SN:chrYHet      LN:347038
@RG     ID:SRR1663609   SM:ZW177         LB:ZW155        PL:ILLUMINA
@PG     ID:bwa  PN:bwa  VN:0.7.8-r455   CL:bwa mem -M -t 4 -R @RG\tID:SRR1663609\tSM:ZW177\tLB:ZW155\tPL:ILLUMINA
/local_data/Drosophila_melanogaster_dm3/BWAIndex/genome.fa SRR1663609_1.fastq.gz SRR1663609_2
.fastq.gz
SRR1663609.1   97   chrX      2051224  60   6M54S     chrYHet     4586 0   GGATCGTGAT… gggfgg[gfg… NM:i:0  MD:Z:46     AS:i:46   XS:i:0    RG:Z:SRR1663609
SRR1663609.1  145   chrYHet      4586   0    100M     chrX     2051224 0   ACTTCTCTTC… BBBBBbdd]c… NM:i:0  MD:Z:100    AS:i:100  XS:i:99   RG:Z:SRR1663609
SRR1663609.2   65   chr3RHet  2308288   0    100M     chrYHet     4712 0   AGAAGAGAAG… Y_b`_ccTccB… NM:i:0  MD:Z:100    AS:i:100  XS:i:100  RG:Z:SRR1663609
SRR1663609.2  129   chrYHet      4712  60  38M62S     chr3RHet 2308288 0   CTTCTCTTCT… eeeae`edee… NM:i:1  MD:Z:17T20  AS:i:33   XS:i:21   RG:Z:SRR1663609
SRR1663609.3   65   chr3RHet  2308278   0    100M     chrYHet     4649 0   AGAAGAGAAG… fffffffff… NM:i:0  MD:Z:100    AS:i:100  XS:i:100  RG:Z:SRR1663609
SRR1663609.3  129   chrYHet      4649   0  41M59S     chr3RHet 2308278 0   TCTCTTCTCT… fffffffff… NM:i:0  MD:Z:41     AS:i:41   XS:i:41   RG:Z:SRR1663609
SA:Z:chrX,5036484,-,16S41M43S,0,2;
SRR1663609.3  401   chrX      5036484   0  16H41M43H chr3RHet 2308278 0   AAAAGAAGAA… BBBBBBBBBB… NM:i:2  MD:Z:7A4G28 AS:i:31   XS:i:28   RG:Z:SRR1663609
SA:Z:chrYHet,4649,+,41M59S,0,0;
SRR1663609.4   99   chr3RHet  854491    0    100M          =   854876 485   AGAAGAAGAA… BBBBBBBBBB… NM:i:0  MD:Z:100    AS:i:100  XS:i:100  RG:Z:SRR1663609
SRR1663609.4  147   chr3RHet  854876    0    100M          =   854491 -485  GAGAAGAGAA… fffffffff… NM:i:0  MD:Z:100    AS:i:100  XS:i:100  RG:Z:SRR1663609
```

Header

read name

flag

chr

position on chr

mapping quality

CIGAR string

chr of mate

mate position on chr

frag length

Read sequence

Read qualities

(shortened for clarity)

edit dist

match str

best aln score

next aln score

Read group

TAGS

Looking into a BAM file: samtools

BAM files are binary – special tool is needed to look inside

Examples:

`samtools view –h myfile.bam | more`
prints the file in SAM format (i.e., human-readable) to screen page by page; skip –h to omit header lines

`samtools view –c myfile.bam`
prints the number of records (alignments) in the file; for BWA mem it may be larger than the number of reads!

`samtools view –f 4 myfile.bam`
Extracts records with a given flag – here: flag 4 (unmapped); prints them to screen

Type **`samtools`**, or go to http://samtools.sourceforge.net/ for more options

`samtools flagstat myfile.bam`
Displays basic alignment stats based on flag

```
samtools flagstat
SRR1663609.sorted.dedup.realigned.fixmate.bam
10201772 + 0 in total (QC-passed reads + QC-failed reads)
74334 + 0 secondary
0 + 0 supplimentary
679571 + 0 duplicates
9685912 + 0 mapped (94.94%:-nan%)
10127438 + 0 paired in sequencing
5063719 + 0 read1
5063719 + 0 read2
8747736 + 0 properly paired (86.38%:-nan%)
9500218 + 0 with itself and mate mapped
111360 + 0 singletons (1.10%:-nan%)
252790 + 0 with mate mapped to a different chr
89859 + 0 with mate mapped to a different chr (mapQ>=5)
```

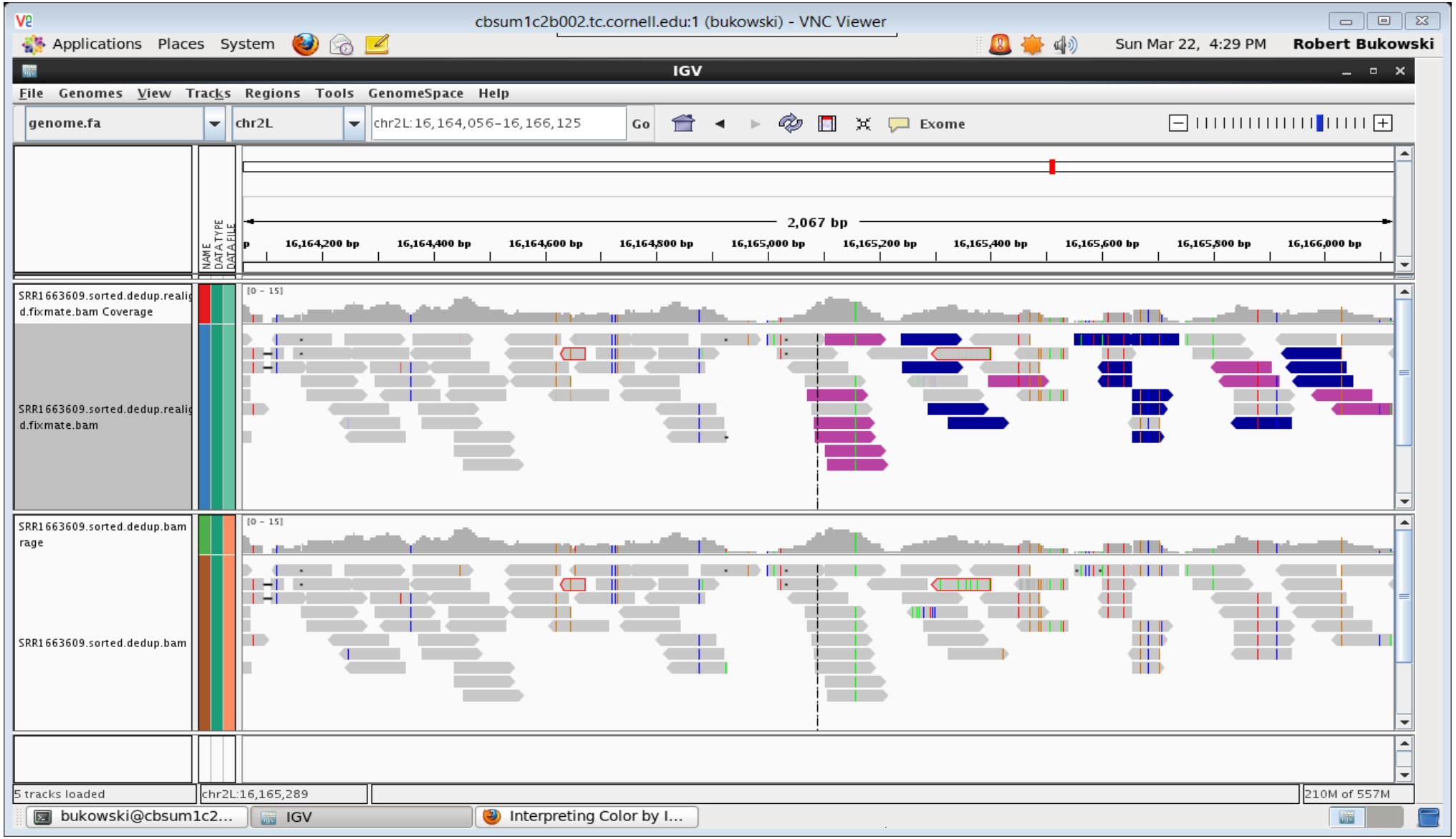
Looking into a BAM file: IGV viewer

Look at multiple BAM files

Zoom in and out

Various color-coding schemes

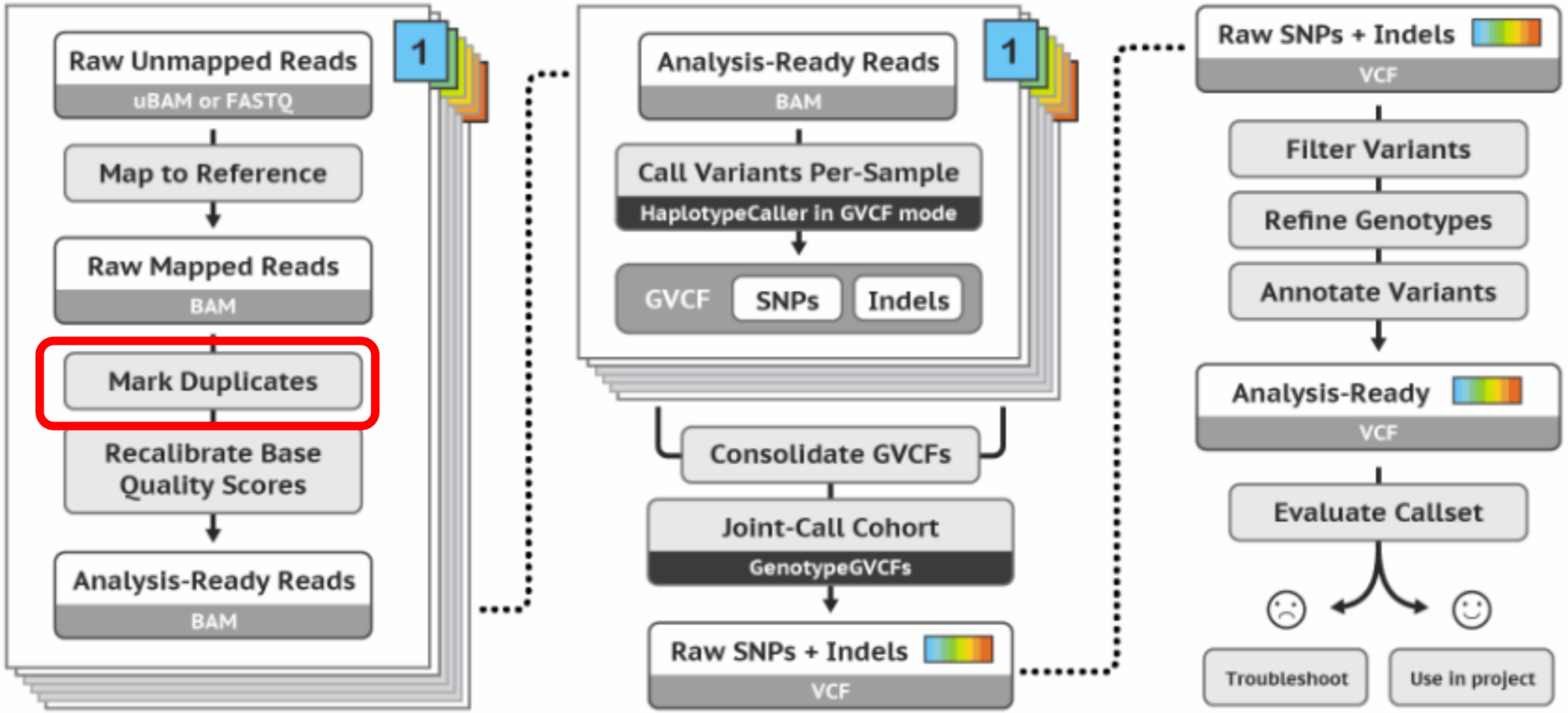
Can load genome annotation track



IGV is a Java program available on BioHPC machines. Can be installed on laptop, too.

http://www.broadinstitute.org/igv/home

“Best Practices” for DNA-Seq variant calling



Duplicate reads (fragments)

- **Optical duplicates:** (Illumina) generated when a single cluster of reads is part of two adjacent tiles' on the same slide and used to compute two read calls separately

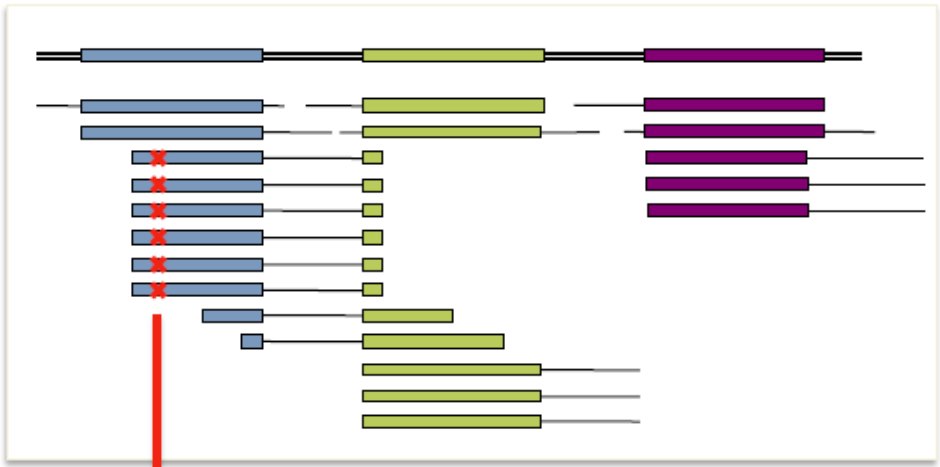
  - Very similar in sequence (except sequencing errors).
  - Identified where the first, say, 50 bases are identical between two reads and the read’s coordinates are close

- **Library duplicates (aka PCR duplicates):** generated when the original sample is pre-amplified to such extent that initial unique targets are PCR replicated prior to library preparation and will lead to several independent spots on the Illumina slide.

  - do not have to be adjacent on the slide
  - share a very high level of sequence identity
  - align to the same place on reference
  - identified from alignment to reference

Why duplicates are bad for variant calling

✖ = sequencing error propagated in duplicates

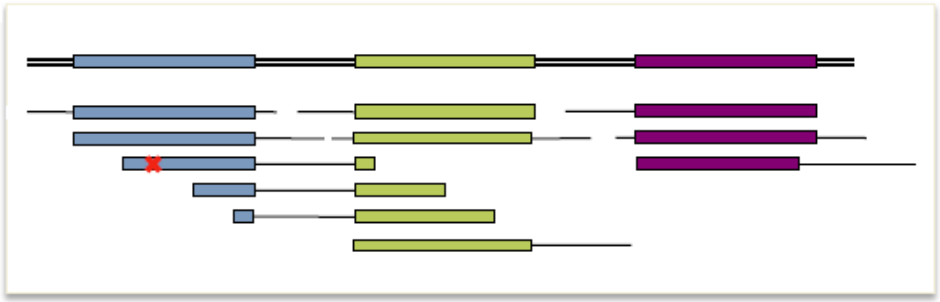


Reference genome

Reads mapped to reference

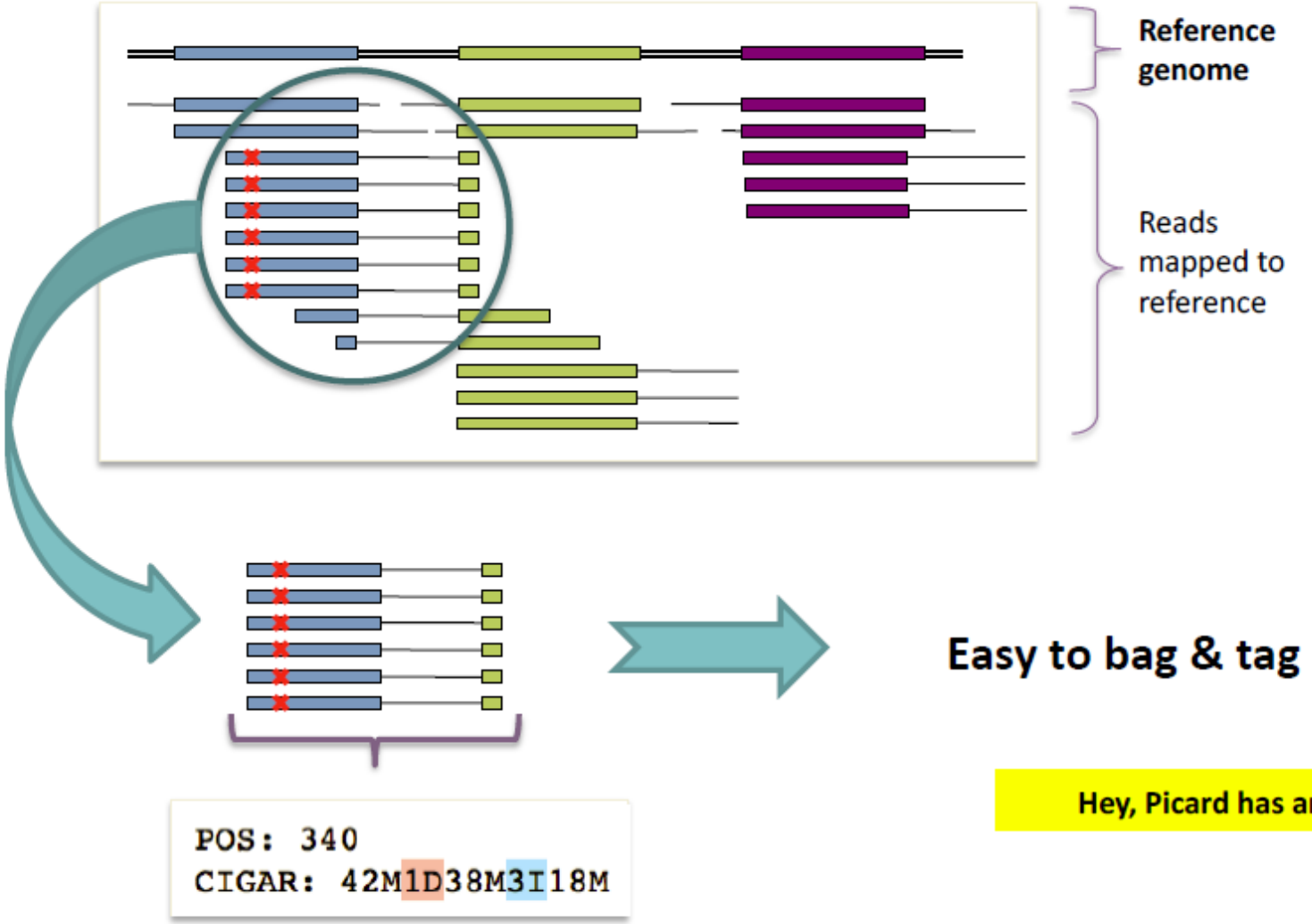
FP variant call (bad)

After marking duplicates, the GATK will only see :



… and thus be more likely to make the right call

How removing (marking) duplicates works



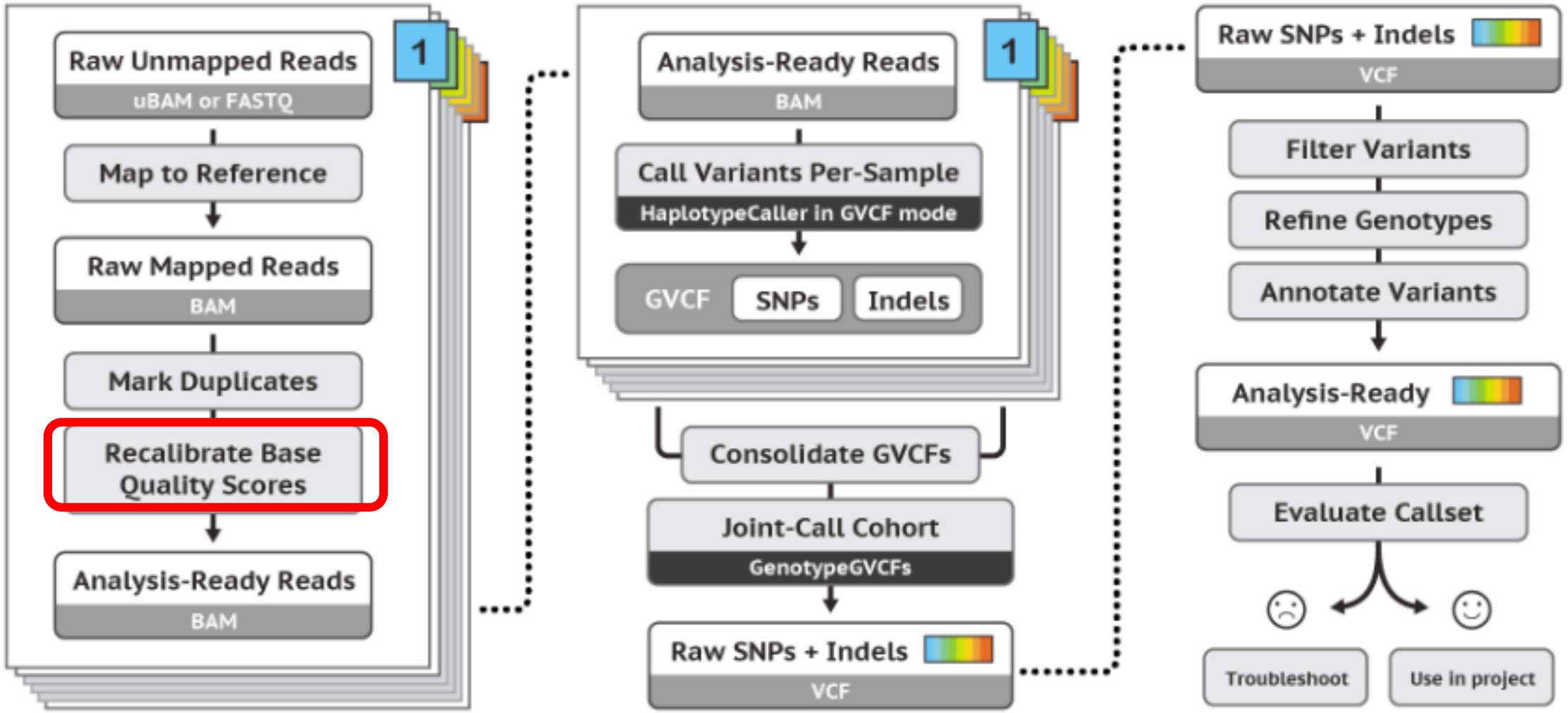
Removing (marking) duplicates with GATK4

```
gatk MarkDuplicatesSpark \
-I sample1.bam \
-O sample1.sorted.dedup.bam \
-M sample1.sorted.dedup.metrics.txt
```

- Tool will also sort BAM over coordinate and produce index (i.e., *.bai fle)
- The metrics file will contain some stats about the de-duping
- In the resulting BAM file, only one fragment from each duplicate group survives unchanged, other duplicate fragments are given a flag 0x400 and will not be used downstream.
- Optimally, detection and marking of duplicate fragments should be done **per library**, i.e., over all read groups corresponding to a given library.
- In practice, often sufficient to do it per lane (read group).

“Best Practices” for DNA-Seq variant calling

?



Base quality score recalibration

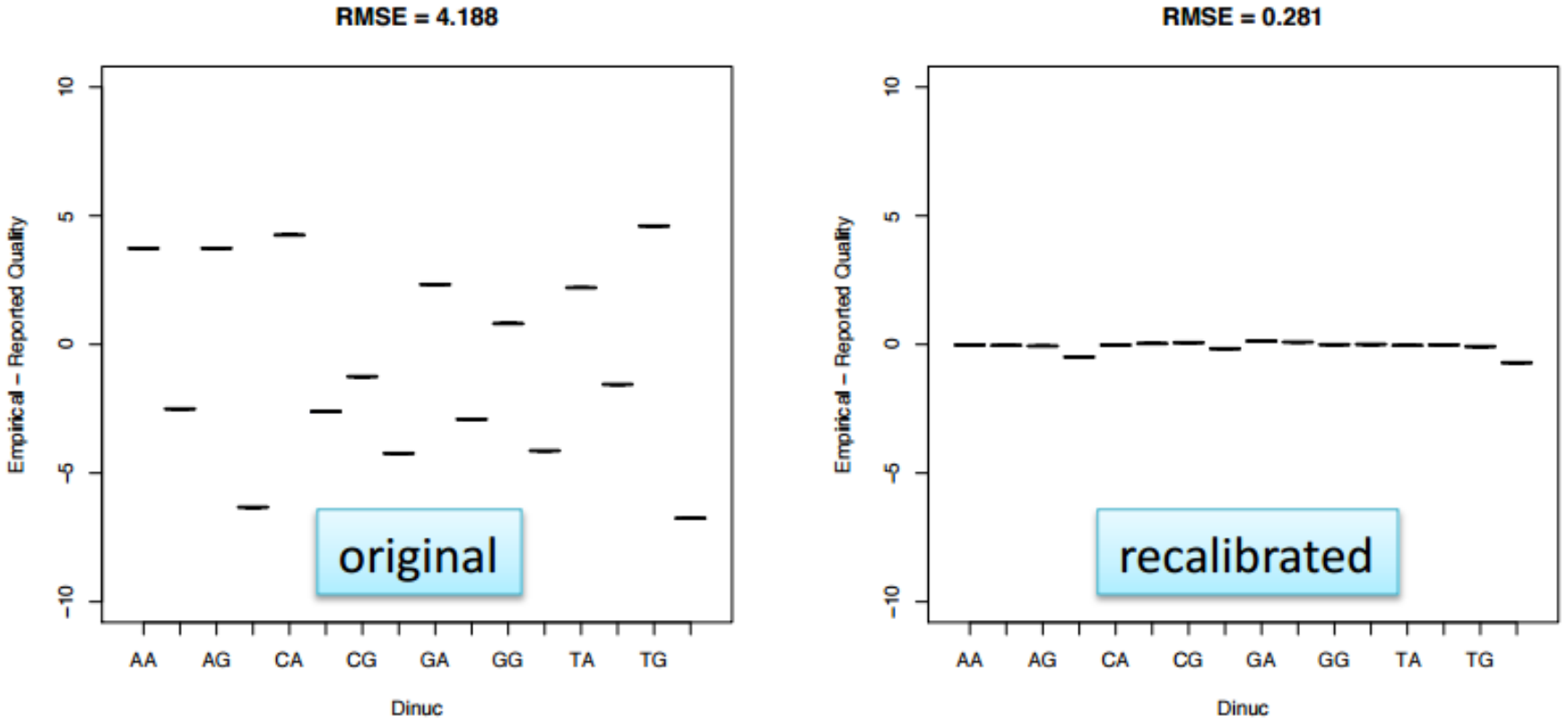
- Define “bins” in terms of covariates:
  - Lane
  - Original quality score
  - Machine cycle (position on read)
  - Sequencing context (what bases are around)
- Scan all aligned reads (i.e., bases) in a given read group
  - Classify each base to a “bin”; decide whether it is a mismatch
- In each bin
  - count the number of mismatches (where read base != reference base)
  - Calculate **empirical quality score** from **#mismatches/#all_observed_bases**; compare to original
- Compile a **database** of corrections

- Scan all reads (i.e., bases) again (in a BAM file)
- For each base
  - Classify into a bin
  - Apply bin-specific correction to base quality scores (based on the database collected in previous step)

Caveat:
- Known variation (SNPs and indels) have to be excluded (not a source of errors)

Base quality scores reported by a sequencer may be inaccurate and biased

Example: Bias in the qualities reported depending on nucleotide context



https://www.broadinstitute.org/gatk/guide/topic?name=methods

Base quality score recalibration: good or bad?

Implicit assumption behind recalibration: sequencing error rate higher than SNP rate

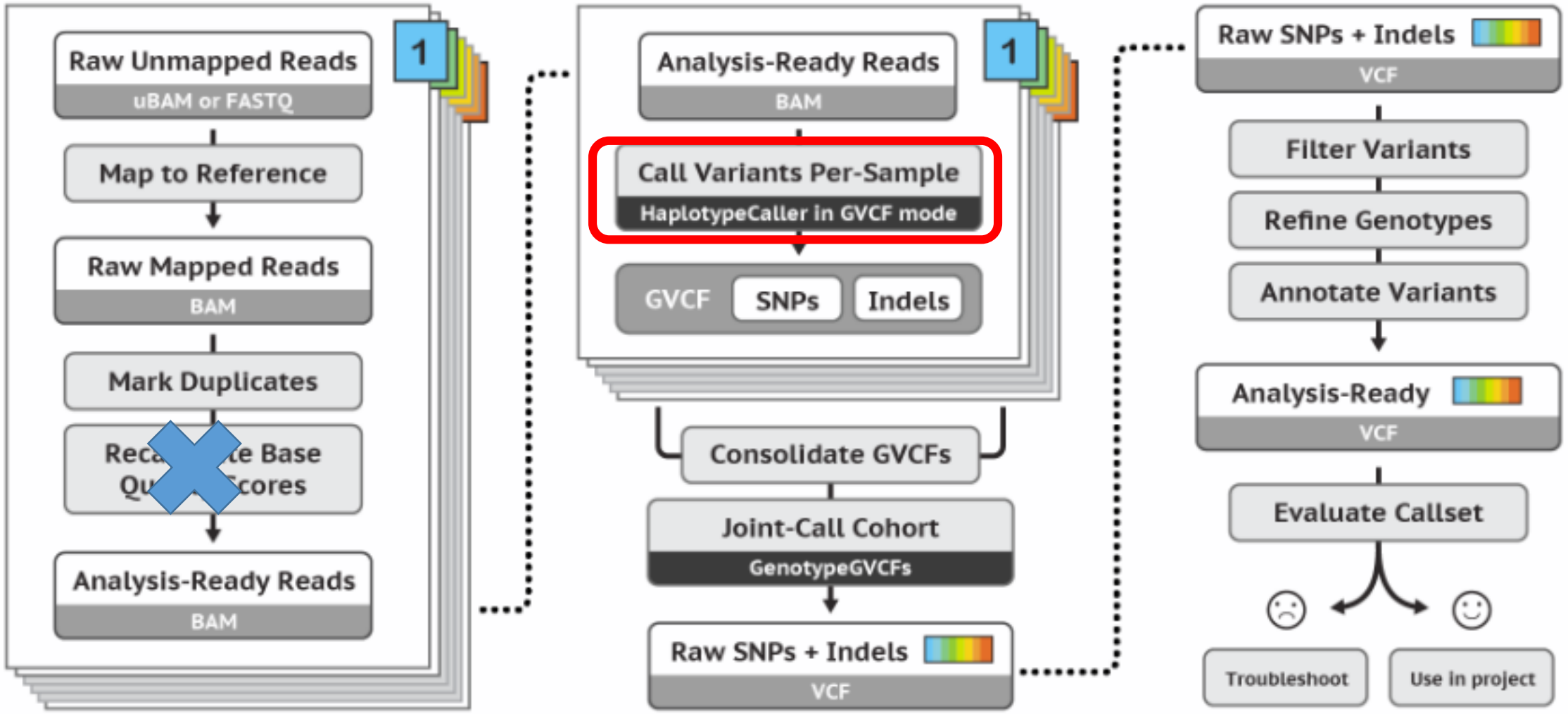
applicable only to populations with very little diversity (humans)

in most (non-human) cases, ‘empirical errors’ are not sequencing errors (either real variants or misalignments)

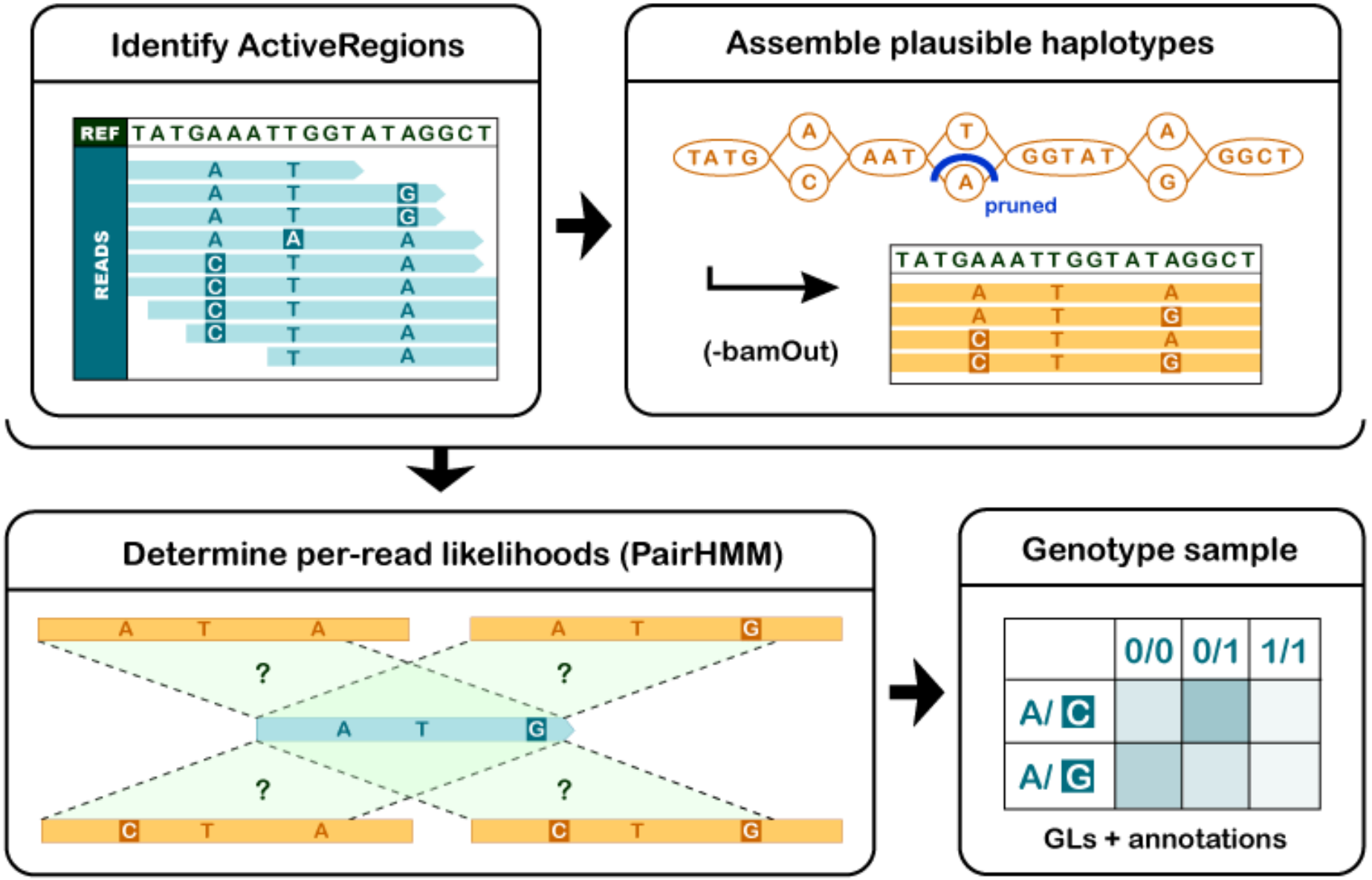
‘known variants’ not always available

Conclusion: do not recalibrate (unless dealing with human genomes)

“Best Practices” for DNA-Seq variant calling



HaplotypeCaller (in GVCF mode): extract variation from alignments for each sample



$P\{D_j|H\}$ determined from PairHMM scores of reads alignments to haplotypes (based on base qualities)

Genomic Variant Call (g.vcf) file: result of HaplotypeCaller

```
#CHROM  POS      ID  REF    ALT      QUAL FILTER   INFO              FORMAT                    NA12878
20  10001567     .   A   <NON_REF>    .    .    END=10001616     GT:DP:GQ:MIN_DP:PL  0/0:38:99:34:0,101,1114
20  10001617     .   C   A,<NON_REF> 493.77  .
BaseQRankSum=1.632;ClippingRankSum=0.000;DP=38;ExcessHet=3.0103;MLEAC=1,0;MLEAF=0.500,0.00;MQRankSum=0.000;RA
W_MQ=136800.00;ReadPosRankSum=0.170    GT:AD:DP:GQ:PL:SB   0/1:19,19,0:38:99:522,0,480,578,538,1116:11,8,13,6
20  10001618     .   T   <NON_REF>    .    .    END=10001627     GT:DP:GQ:MIN_DP:PL  0/0:39:99:37:0,105,1575
20  10001628     .   G   A,<NON_REF> 1223.77  .
DP=37;ExcessHet=3.0103;MLEAC=2,0;MLEAF=1.00,0.00;RAW_MQ=133200.00   GT:AD:DP:GQ:PL:SB
1/1:0,37,0:37:99:1252,111,0,1252,111,1252:0,0,21,16
20  10001629     .   G   <NON_REF>    .    .    END=10001660     GT:DP:GQ:MIN_DP:PL  0/0:43:99:38:0,102,1219
20  10001661     .   T   C,<NON_REF> 1779.77  .
DP=42;ExcessHet=3.0103;MLEAC=2,0;MLEAF=1.00,0.00;RAW_MQ=151200.00   GT:AD:DP:GQ:PL:SB
1/1:0,42,0:42:99:1808,129,0,1808,129,1808:0,0,26,16
20  10001662     .   T   <NON_REF>    .    .    END=10001669     GT:DP:GQ:MIN_DP:PL  0/0:44:99:43:0,117,1755
20  10001670     .   T   G,<NON_REF> 1773.77  .
DP=42;ExcessHet=3.0103;MLEAC=2,0;MLEAF=1.00,0.00;RAW_MQ=151200.00   GT:AD:DP:GQ:PL:SB
1/1:0,42,0:42:99:1802,129,0,1802,129,1802:0,0,25,17
20  10001671     .   G   <NON_REF>    .    .    END=10001673     GT:DP:GQ:MIN_DP:PL  0/0:43:99:42:0,120,180
20  10001674     .   A   <NON_REF>    .    .    END=10001674     GT:DP:GQ:MIN_DP:PL  0/0:42:96:42:0,96,1197
```

Positional information: chromosome, start, end (if non-variant block)

Non-reference allele info; <NON_REF> stands for any non-reference allele

Genotype (GT): may be 0/0, 0/1, 1/1, 0/2, 1/2, 2/2, … where '0' is REF allele and 1, 2, … are ALT alleles in order listed

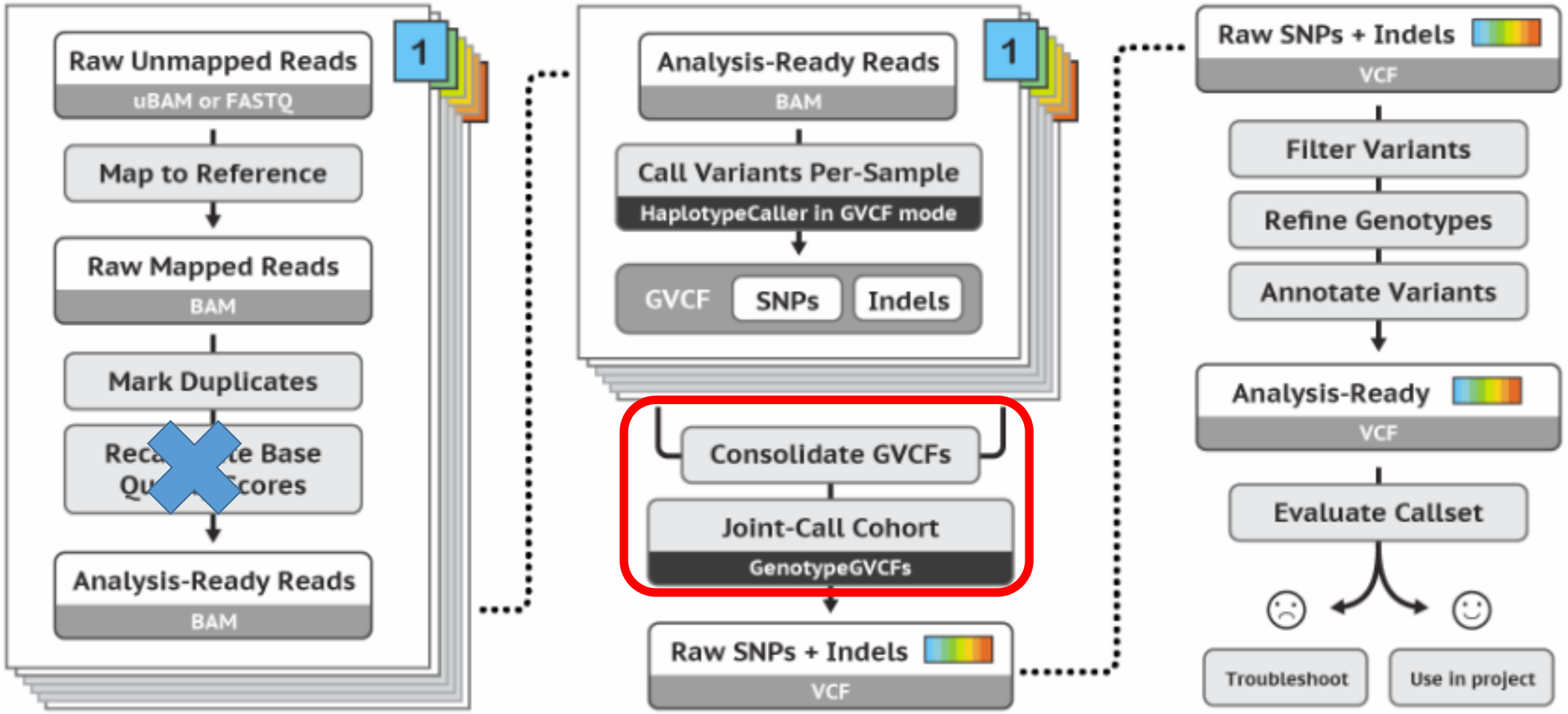
Genotype likelihoods (PL): example: **0,120,180** means that **0/1** is 10^{-12} times less likely than **0/0**

All symbols defined in the **header** of the g.vcf file (e.g., entries in INFO field for variant sites)

GVCF header: excerpts

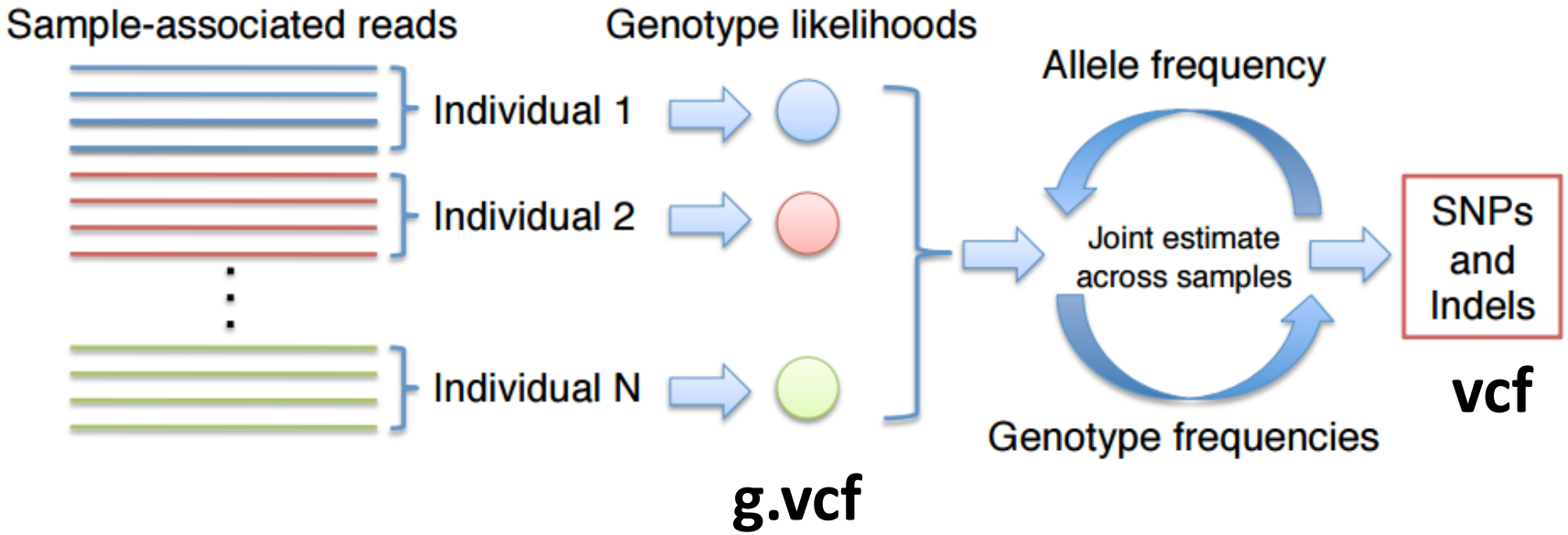
##fileformat=VCFv4.2
##ALT=<ID=NON_REF,Description="Represents any possible alternative allele at this location">
##FILTER=<ID=LowQual,Description="Low quality">
##FORMAT=<ID=AD,Number=R,Type=Integer,Description="Allelic depths for the ref and alt alleles in the order listed">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Approximate read depth (reads with MQ=255 or with bad mates are filtered)">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##GVCFBlock55-56=minGQ=55(inclusive),maxGQ=56(exclusive)
##GVCFBlock56-57=minGQ=56(inclusive),maxGQ=57(exclusive)
##GVCFBlock57-58=minGQ=57(inclusive),maxGQ=58(exclusive)
##INFO=<ID=BaseQRankSum,Number=1,Type=Float,Description="Z-score from Wilcoxon rank sum test of Alt Vs. Ref base qualities">
##INFO=<ID=ClippingRankSum,Number=1,Type=Float,Description="Z-score From Wilcoxon rank sum test of Alt vs. Ref number of hard clipped bases">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Approximate read depth; some reads may have been filtered">
##INFO=<ID=DS,Number=0,Type=Flag,Description="Were any of the samples downsampled?">
##INFO=<ID=END,Number=1,Type=Integer,Description="Stop position of the interval">
##INFO=<ID=ExcessHet,Number=1,Type=Float,Description="Phred-scaled p-value for exact test of excess heterozygosity">
##INFO=<ID=InbreedingCoeff,Number=1,Type=Float,Description="Inbreeding coefficient as estimated from the genotype likelihoods per-sample when compared against the Hardy-Weinberg expectation">
##INFO=<ID=MLEAC,Number=A,Type=Integer,Description="Maximum likelihood expectation (MLE) for the allele counts (not necessarily the same as the AC), for each ALT allele, in the same order as listed">
##INFO=<ID=MLEAF,Number=A,Type=Float,Description="Maximum likelihood expectation (MLE) for the allele frequency (not necessarily the same as the AF), for each ALT allele, in the same order as listed">
##INFO=<ID=MQ,Number=1,Type=Float,Description="RMS Mapping Quality">
##INFO=<ID=MQRankSum,Number=1,Type=Float,Description="Z-score From Wilcoxon rank sum test of Alt vs. Ref read mapping qualities">
##INFO=<ID=RAW_MQ,Number=1,Type=Float,Description="Raw data for RMS Mapping Quality">
##INFO=<ID=ReadPosRankSum,Number=1,Type=Float,Description="Z-score from Wilcoxon rank sum test of Alt vs. Ref read position bias">
##contig=<ID=20,length=63025520,assembly=GRCh37>
##source=HaplotypeCaller

“Best Practices” for DNA-Seq variant calling



Variation across cohort

Multi-sample calling integrates per sample likelihoods to jointly estimate allele frequency of variation



For each site, obtain distribution of count of non-reference allele (**AC**):

Pr{AC=i | D} ← Per sample **Genotype Likelihoods** + **Prior**

Prior: **Pr{AC=i} = Het/i** (where Het is population heterozygosity; or define your own prior)

QUAL = -10*log Pr{AC=0| D} (reported in VCF file)

From BAM files to population variants

gatk HaplotypeCaller
In gVCF mode
(1 sample calls,
preferably in parallel)

sample1.g.vcf
sample2.g.vcf
…
sampleN.g.vcf

allsample.g.vcf

gatk GenotypeGVCFs
(joint variant calling)

N-sample VCF file

Run for each sample (on a multi-CPU machine, **run a few simultaneously**)

```
gatk HaplotypeCaller \
     -R genome.fa \
     -I sample1.sorted.dedup.recal.bam  \
     --minimum-mapping-quality 30 \
     -ERC GVCF \
     -O sample1.g.vcf
```

Slow

….Followed by combining g.vcf files

```
gatk CombineGVCFs \
   -R genome.fa \
   --variant sample1.g.vcf \
     ……
   --variant sampleN.g.vcf
   -O allsample.g.vcf
```

Fast

….and by joint variant calling with GenotypeGVCFs

```
gatk GenotypeGVCFs \
   -R genome.fa \
   -V allsample.g.vcf \
   -stand-call-conf 5 \
   -O allsample.vcf
```

Fast

Variant Call Format (VCF)

Similar to **g.vcf**, but used to describe sites deemed **variant** across a cohort

HEADER LINES: start with "##", describe all symbols found later on in FORMAT and ANNOTATIONS, e.g.,

```
##fileformat=VCFv4.1
##FORMAT=<ID=AD,Number=.,Type=Integer,Description="Allelic depths for the ref and alt alleles in the order listed">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Approximate read depth (reads with MQ=255 or with bad mates are filtered)">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
……
```

SITE RECORDS:

#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	ZW155	ZW177
chr2R	2926	.	C	A	345.03	PASS	[ANNOTATIONS]	GT:AD:DP:GQ:PL	0/1:4,9:13:80:216,0,80	0/0:6,0:6:18:0,18,166
chr2R	9862	.	TA	T	180.73	.	[ANNOTATIONS]	GT:AD:DP:GQ:PL	1/1:0,5:5:15:97,15,0	1/1:0,4:4:12:80,12,0
chr2R	10834	.	A	ACTG	173.04	.	[ANNOTATIONS]	GT:AD:DP:GQ:PL	0/0:14,0:14:33:0,33,495	0/1:6,3:9:99:105,0,315

ID: some ID for the variant, if known (e.g., dbSNP)
REF, ALT: reference and alternative alleles (on forward strand of reference)
QUAL = -10*log(1-p), where p is the probability of variant being present given the read data
FILTER: whether the variant failed a filter (filters defined by the user or program processing the file)

Variant Call Format (VCF)

```
[HEADER LINES]
#CHROM  POS    ID  REF   ALT    QUAL    FILTER     INFO            FORMAT            ZW155                 ZW177
chr2R   2926   .    C     A     345.03    PASS   [ANNOTATIONS]   GT:AD:DP:GQ:PL  0/1:4,9:13:80:216,0,80   0/0:6,0:6:18:0,18,166
chr2R   9862   .   TA     T     180.73     .     [ANNOTATIONS]   GT:AD:DP:GQ:PL  1/1:0,5:5:15:97,15,0     1/1:0,4:4:12:80,12,0
chr2R 10834    .    A  ACTG    173.04     .     [ANNOTATIONS]   GT:AD:DP:GQ:PL  0/0:14,0:14:33:0,33,495       ./.
```

GT (genotype):

0/0	reference homozygote
0/1	reference-alternative heterozygote
1/1	alternative homozygote
0/2, 1/2, 2/2, etc.	- possible if more than one alternative allele present
./.	missing data

AD: allele depths

DP: total depth (may be different from sum of AD depths, a the latter include only reads significantly supporting alleles)

PL: genotype likelihoods (phred-scaled), normalized to the best genotype, e.g.,

PL(0/1) = -10*log[Prob(data|0/1) / Prob(data|best_genotype)]

GQ: genotype quality – this is just PL of the second-best genotype

Variant Call Format (VCF)

[HEADER LINES]									
#CHROM	POS	ID	REF	ALT	QUAL	FILTER	INFO	FORMAT	ZW155	ZW177
chr2R	2926	.	C	A	345.03	PASS	[ANNOTATIONS]	GT:AD:DP:GQ:PL	0/1:4,9:13:80:216,0,80	0/0:6,0:6:18:0,18,166
chr2R	9862	.	TA	T	180.73	.	[ANNOTATIONS]	GT:AD:DP:GQ:PL	1/1:0,5:5:15:97,15,0	1/1:0,4:4:12:80,12,0
chr2R	10834	.	A	ACTG	173.04	.	[ANNOTATIONS]	GT:AD:DP:GQ:PL	0/0:14,0:14:33:0,33,495	0/1:6,3:9:99:105,0,315

[ANNOTATIONS]: all kinds of quantities and flags that characterize the variant; supplied by the variant caller (different callers will do it differently)

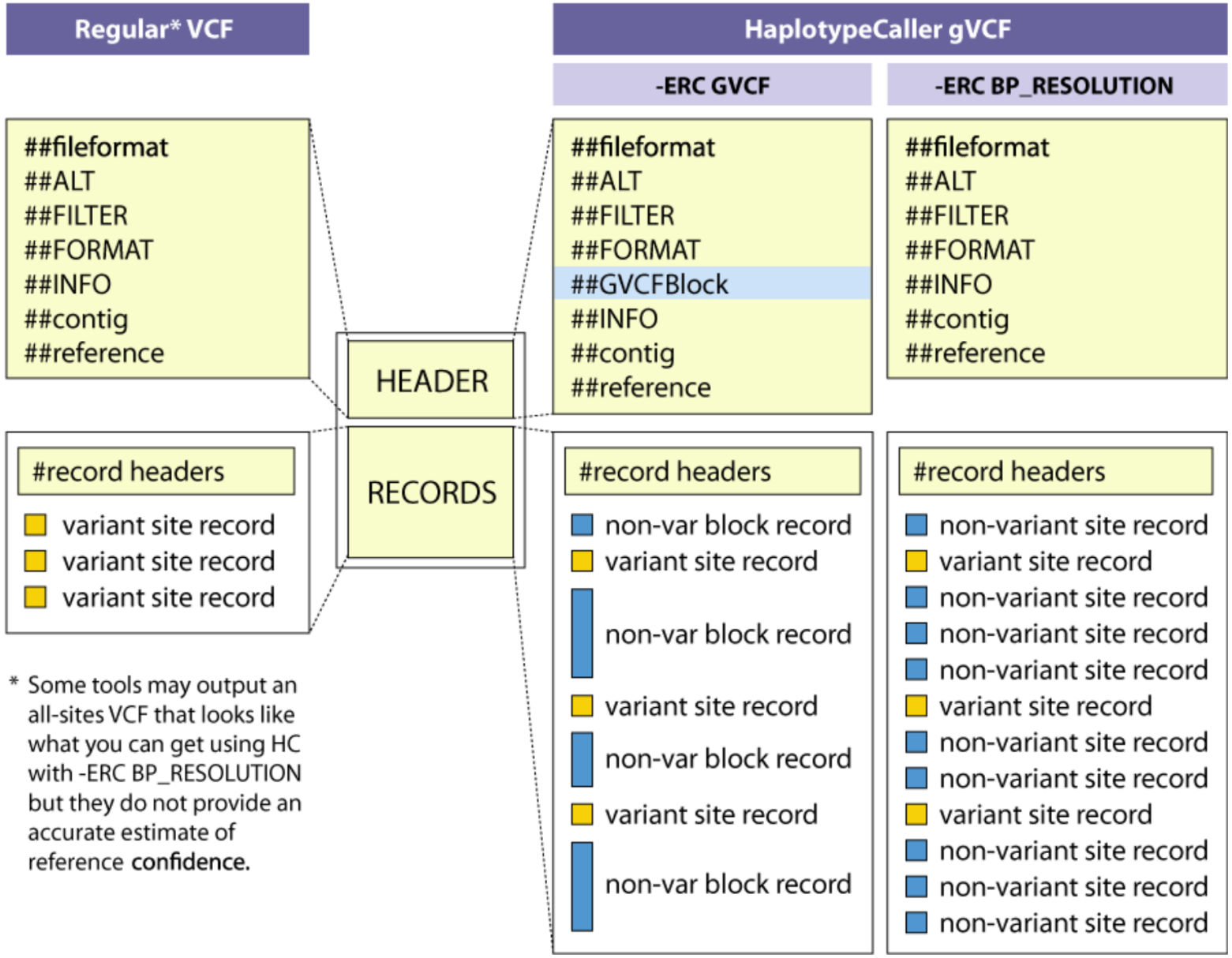
Example:

```
AC=2;AF=0.333;AN=6;DP=16;FS=0.000;GQ_MEAN=16.00;GQ_STDDEV=10.54;MLEAC=2;MLEAF=0.33
3;MQ=25.00;MQ0=0;NCC=1;QD=22.51;SOR=3.611
```

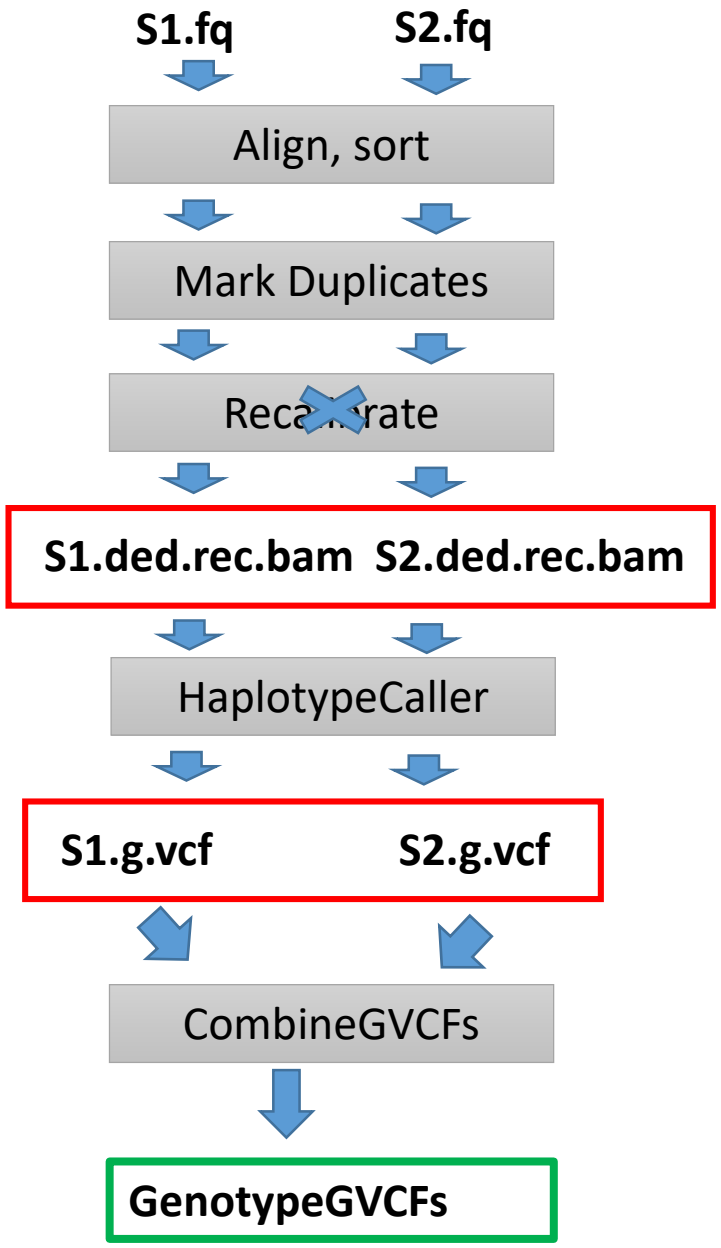
All ANNOTATION parameters are defined in the **HEADER LINES** on top of the file

```
…
##INFO=<ID=AC,Number=A,Type=Integer,Description="Allele count in genotypes, for each ALT allele, in the same order as listed">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency, for each ALT allele, in the same order as listed">
##INFO=<ID=AN,Number=1,Type=Integer,Description="Total number of alleles in called genotypes">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Approximate read depth; some reads may have been filtered">
##INFO=<ID=FS,Number=1,Type=Float,Description="Phred-scaled p-value using Fisher's exact test to detect strand bias">
##INFO=<ID=GQ_MEAN,Number=1,Type=Float,Description="Mean of all GQ values">
##INFO=<ID=MQ,Number=1,Type=Float,Description="RMS Mapping Quality">
##INFO=<ID=NCC,Number=1,Type=Integer,Description="Number of no-called samples">
##INFO=<ID=QD,Number=1,Type=Float,Description="Variant Confidence/Quality by Depth">
##INFO=<ID=SOR,Number=1,Type=Float,Description="Symmetric Odds Ratio of 2x2 contingency table to detect strand bias">
…
```

VCF versus GVCF format



Running things in parallel



Process individual datasets in parallel

Process genomic regions (e.g., chromosomes) in parallel (after alignment)

Some pipeline stages allow multithreading, i.e., processing within one dataset may be sped up by running on multiple CPUs, e.g.:

BWA alignment on 10 threads:

```
bwa mem -t 10 [other options]
```

HaplotypeCaller on 4 threads:

```
gatk HaplotypeCaller --native-pair-hmm-threads 4 [other options]
```

Most GATK4 tools have multithreaded versions (add **`Spark`** at end of tool name, like **`HaplotypeCallerSpark`**) – some still in BETA stage…

Caution:

- total number of requested threads should never exceed the number of CPUs on the machine!
- Using too many threads or running too many simultaneous jobs my decrease performance. Experiment!
- BWA and **`HaplotypeCaller`** scale decently on up to 8-10 threads

Technical considerations

Set **PATH** to see the latest GATK (execute once in terminal or in the beginning of a script):

```
export PATH=/programs/gatk-4.1.4.0:$PATH
```

Use “--java-options” to control Java virtual machine, e.g., give java 8GB of RAM to work in:

```
gatk --java-options “-Xmx8g” [other options]
```

Specify scratch directory (important for most tools), e.g.,

```
gatk HaplotypeCaller --tmp-dir /workdir/$USER/tmp [other options]
```

Compressed (gzipped) versions of all FASTQ and VCF files can be used with all commands

For **GenotypeGVCFs**, use permissive variant emission threshold (you can filter bad variants later)

```
gatk GenotypeGVCFs -stand_call_conf 5 [other options]
```

Alternatives to GATK

bcftools, samtools (Sanger Institute, Broad Institute, http://www.htslib.org/doc/bcftools.html)

FreeBayes (Erik Garrison et al., https://github.com/ekg/freebayes)

- Haplotype-based variant detection (no re-alignment around indels needed)
- Better (than GATK's) Bayesian model, directly incorporating a number of metrics, such as read placement bias and allele balance
- **In our tests – a few times faster than GATK HaplotypeCaller**
- Still suffers from "N+1" problem

Sentieon (http://sentieon.com)

- Commercial version of GATK (currently equivalent to GATK 3.8)
- **10-30 times faster than GATK on most parts of the pipeline**
- Command syntax different from GATK (although functionality the same)
- Available on BioHPC Lab for $50/week (need to recover license cost)
  - License: 300 CPU cores of can run simultaneously (across all machines) at any time

What to do with a freshly obtained set of called variants?

Simple linux tools help analyze a VCF file

Count variants:

```
grep –v ”#” all.vcf | wc –l
```

Extract sites located between positons 10000 and 20000 on chromosome chr2R and save them in a new VCF file:

```
head -1000 all.vcf | grep “#” > new_file.vcf
grep –v ”#” all.vcf | \
awk ’{if($1==”chr2R” && $2 >=10000 && $2 <=20000) print}’ >> new_file.vcf
```

Extract variants with quality (QUAL) greater than 100 (the resulting file will have no header!):

```
grep –v “#” all.vcf | awk ’{if($6>100) print}’ > good_variants
```

Useful tool: VariantFiltration – hard filtering on various criteria

Example:

```
Gatk VariantFiltration \
-R genome.fa \
--filter-expression "MQ0 >= 4 && ((MQ0 / (1.0 * DP)) > 0.1)"  \
--filter-expression “FS>=10.0"   \
--filter-expression “AN>=4" \
--filter-expression "DP>100 || DP<4" \
--filter-name HARD_TO_VALIDATE  \
--filter-name SNPSBFilter \
--filter-name SNPNalleleFilter  \
--filter-name SNPDPFilter  \
-cluster 3 \
-window 10  \
-V all.vcf \
-O all.filtered.vcf
```

Filtering options for SNPs may be different than for indels (see exercise)

Whenever any of the “-filter” conditions satisfied, the corresponding **`--filter-name`** will be added to the **`FILTER`** field in VCF.

Commonly used filtering parameters (from GATK)

DP

Total depth of read coverage at the site (shouldn't be too low)

MQ0

Number of zero mapping quality reads spanning the site (should be low)

MQ

RMS mapping quality of reads spanning the site

FS

P-value (phred-scaled) of the strand bias contingency table (should be low)

QD

QUAL/(depth of non-reference reads) – should be large (e.g, >2)

ReadPosRankSum

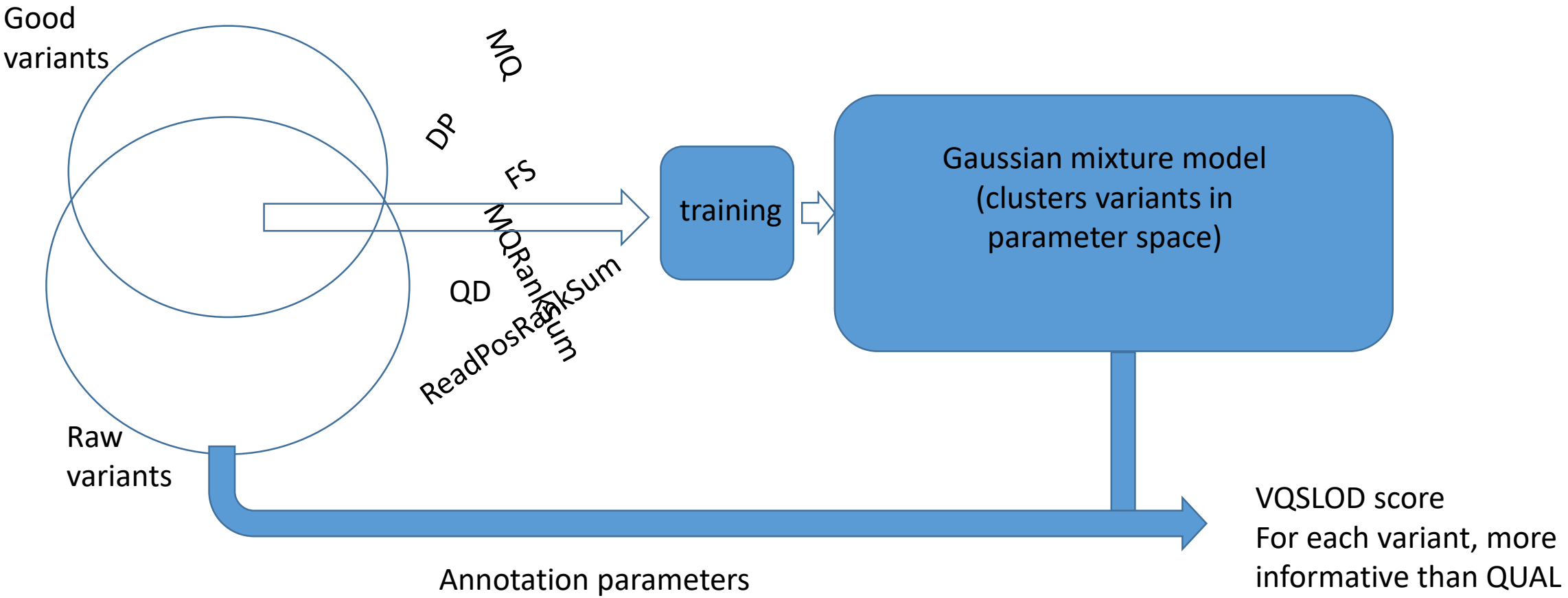
Parameter showing how close the variant site is to ends of reads (typically more positive for good variants) – available only for heterozygous sites

MQRankSum

Parameter comparing mapping qualities of reads carrying an alternative allele to reference reads – available only for heterozygous sites (typically more positive for good variants).

Variant Quality Score Recalibration (VSQR)
Tools: VariantRecalibrator, ApplyVQSR

Machine learning model, recommended instead of hard (threshold-based) filtering when a set of true, reliable variants is available.



Other VCF analysis and manipulation package: vcftools

vcftools (A. Auton, A. Amrcketta, http://vcftools.sourceforge.net/)

Obtain basis VCF statistics (number of samples and variant sites):

```
vcftools --vcf hc.chr2R.vcf
```

Extract subset of variants (chromosome chr2R, between positions 1M and 2M) and write tem a new VCF file

```
vcftools –vcf hc.chr2R.vcf --chr chr2R --from-bp 1000000 --to-bp 2000000
--recode –recode-INFO-all -c > subset.vcf
```

Get allele frequencies for all variants and write them to a file

```
vcftools --vcf hc.chr2R.vcf --freq -c > hc.chr2R.freqs
```

Compare two VCF files (will print out various kinds of compare info in files **`hc.ug.compare.*`**):

```
vcftools --vcf hc.chr2R.vcf --diff ug.chr2R.vcf --out hc.ug.compare
```

Vcftools can also compute

- LD statistics
- Fst between populations

Word of caution

All call optimization effort in GATK directed towards detection and removal of sequencing errors and small alignment errors

Reference genome assumed to be adequate (similar to those of re-sequenced individuals), i.e., reads assumed to be decently mapped to right locations possibly with small alignment ambiguities

Elaborate GATK pipeline will not help in case of massive misalignments (reads mapping to completely wrong locations) resulting from large diversity

What to do then?

Filter raw set of variants (most of them wrong) based on data for a large population (if you have one)

Identity by Descent (IBD): exploit local identity within pairs of samples
local Linkage Disequilibrium (LD): true variant should be in LD with nearby ones

Use methods based on multiple references covering population diversity (pangenomes, graph methods, PHG)