Variant calling: Part 1

Robert Bukowski, Qi Sun, Minghui Wang

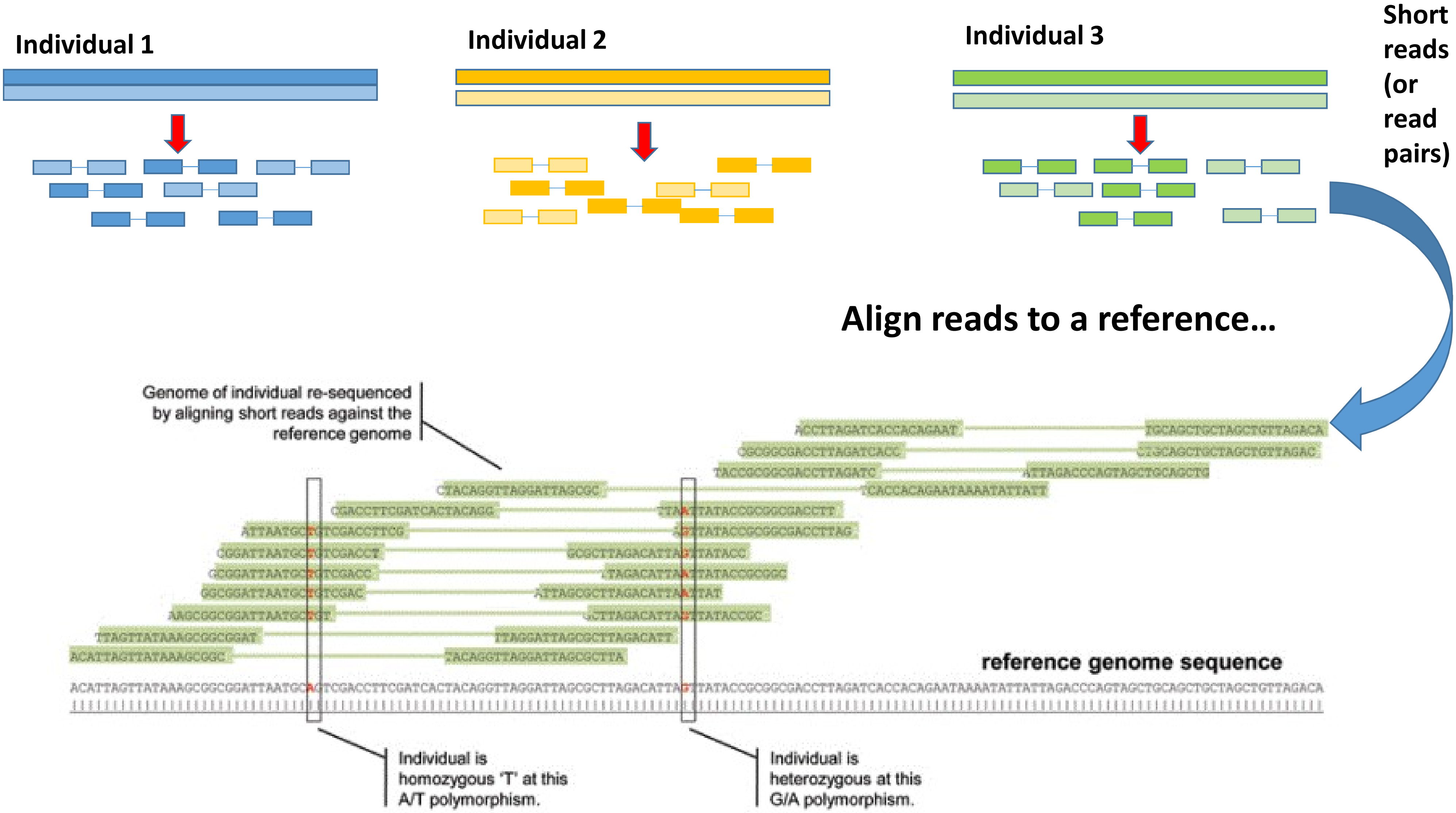
Bioinformatics Facility

Institute of Biotechnology

Slides: http://cbsu.tc.cornell.edu/lab/doc/Variant_workshop_Part1.pdf

Exercise instructions: http://cbsu.tc.cornell.edu/lab/doc/Variant_exercise1.pdf

Workshop contact: bukowski@cornell.edu



Align reads to a reference…

Genome of individual re-sequenced by aligning short reads against the reference genome

Individual is homozygous 'T' at this A/T polymorphism.

Individual is heterozygous at this G/A polymorphism.

Expected output: table of genotypes

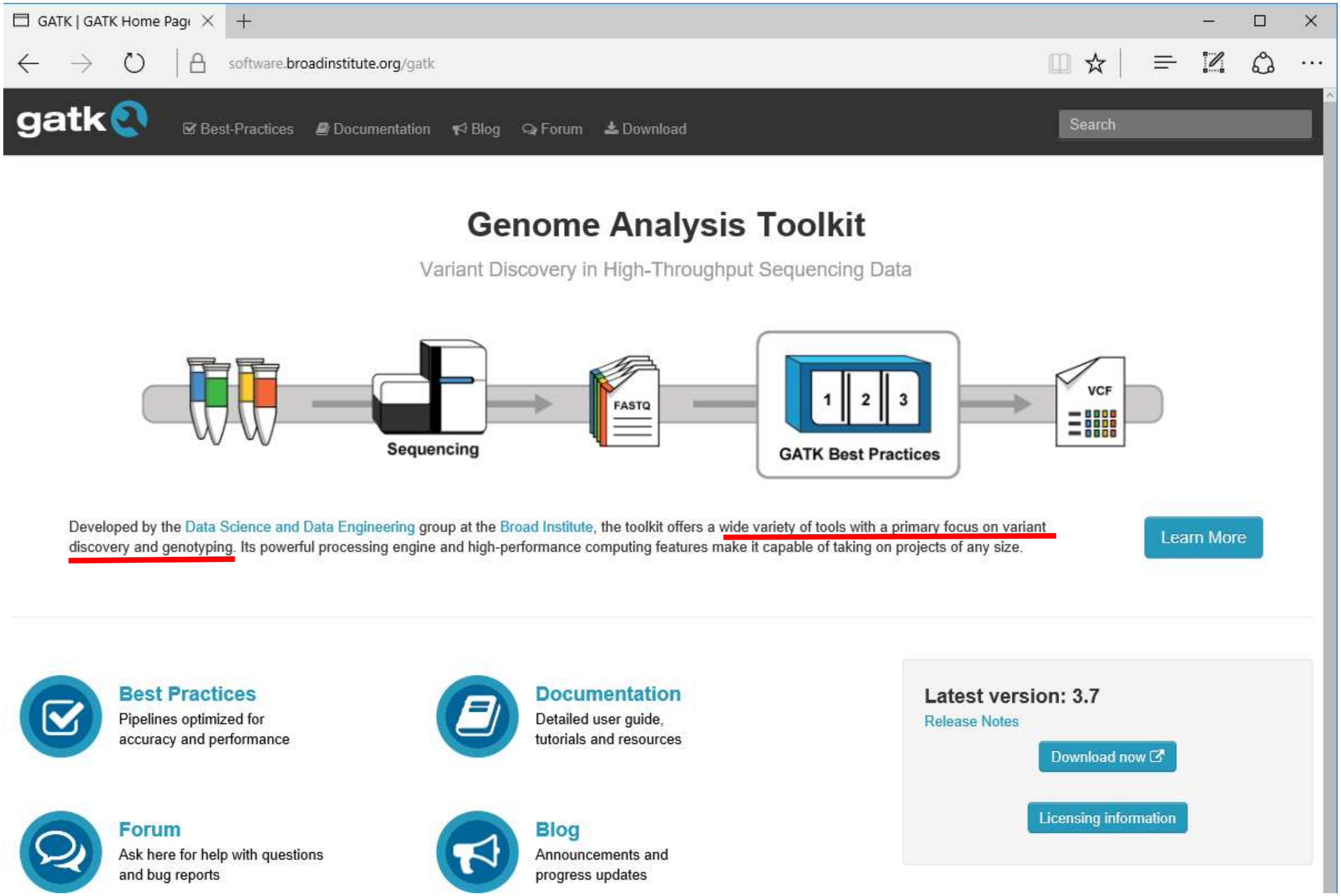
Variant site chr and position	Indiv1	Indiv2	Indiv3	….
site1	AA	AA	AC	…
site2	GT	missing	TT	…
…	…	…	…	…
siteN	CC	CC	AA	…

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 – next week

State of the art: GATK from Broad Institute



Genome Analysis Toolkit

Variant Discovery in High-Throughput Sequencing Data

Developed by the Data Science and Data Engineering group 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.

Best Practices
Pipelines optimized for accuracy and performance

Documentation
Detailed user guide, tutorials and resources

Forum
Ask here for help with questions and bug reports

Blog
Announcements and progress updates

Latest version: 3.7
Release Notes

GATK

- Developed in conjunction with 1000 (human) genomes project
- Package of command-line tools (written in **Java**)
- GATK pipelines rely on another Java package, **PICARD** (also from Broad) for processing of alignment files
- Contains multiple tools for
  - NGS data processing
  - Genotyping and variant discovery
  - Variant filtering and evaluation
    - Still very specific to organism under study – some harder than others
  - Massively parallel processing on HPC clusters
- Ever evolving and adapting to emerging sequencing technologies
- GATK development led a to protocol referred to as **<u>Best practices</u> for calling variants** with the GATK

Where to go for detailed documentation of GATK and PICARD tools

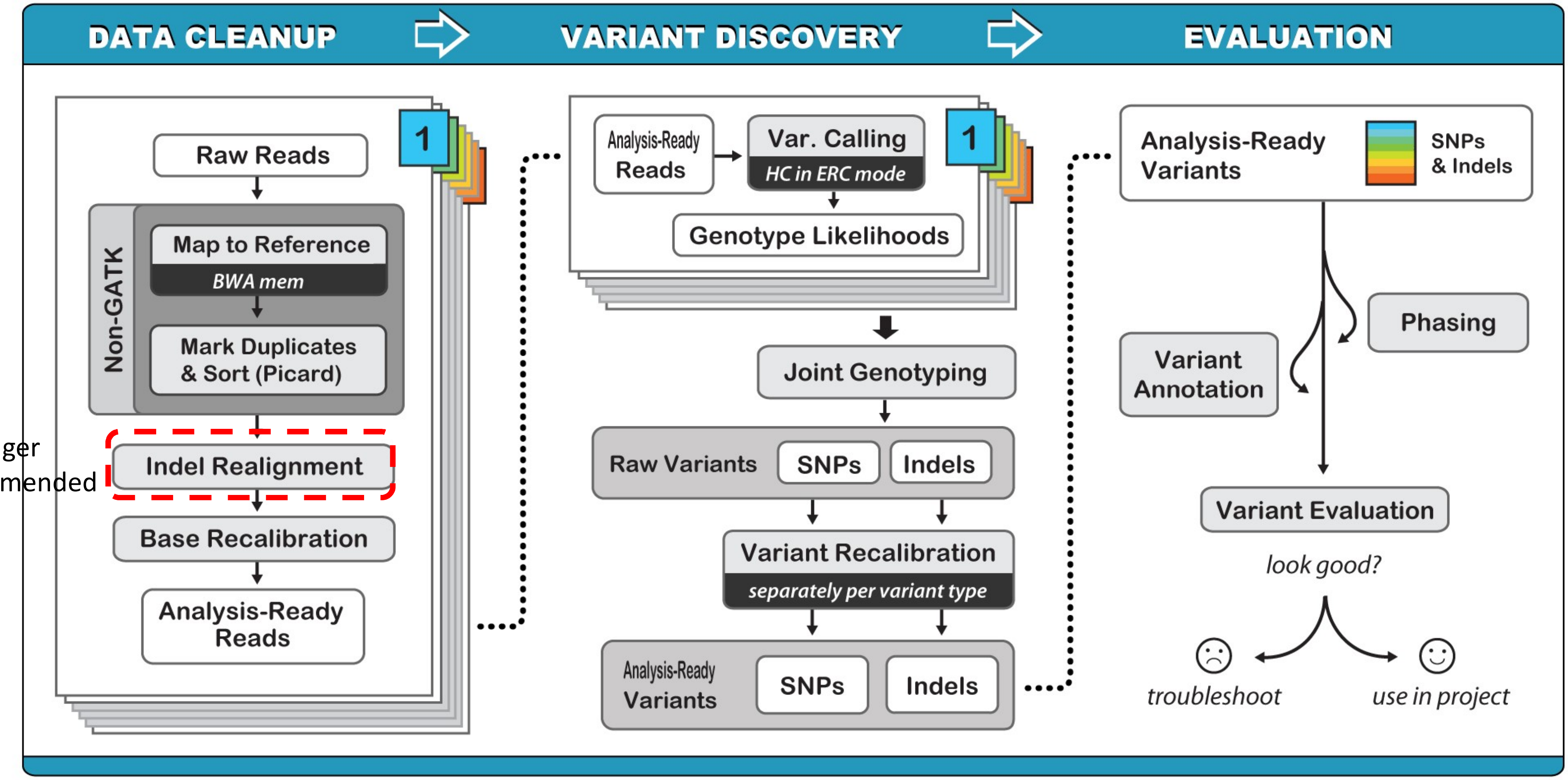
GATK

https://www.broadinstitute.org/gatk/guide/tooldocs/

PICARD

http://broadinstitute.github.io/picard/

Best Practices for DNA-Seq variant calling



DATA CLEANUP

Raw Reads

Non-GATK

Map to Reference

BWA mem

Mark Duplicates & Sort (Picard)

Indel Realignment

Base Recalibration

Analysis-Ready Reads

1

VARIANT DISCOVERY

Analysis-Ready Reads

Var. Calling

HC in ERC mode

Genotype Likelihoods

1

Joint Genotyping

Raw Variants

SNPs

Indels

Variant Recalibration

separately per variant type

Analysis-Ready Variants

SNPs

Indels

EVALUATION

Analysis-Ready Variants

SNPs & Indels

Variant Annotation

Phasing

Variant Evaluation

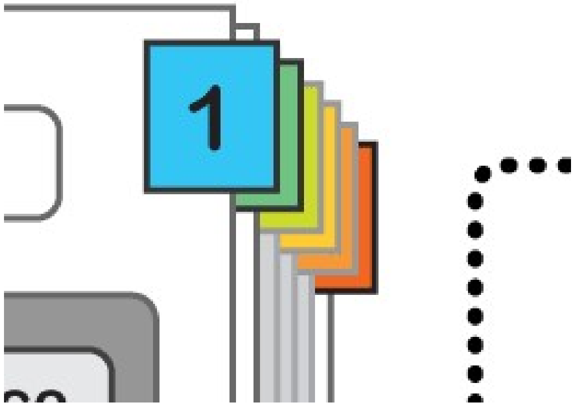
look good?

troubleshoot

use in project

No longer recommended

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 and one Illumina lane**

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

Reads from one sample may be in

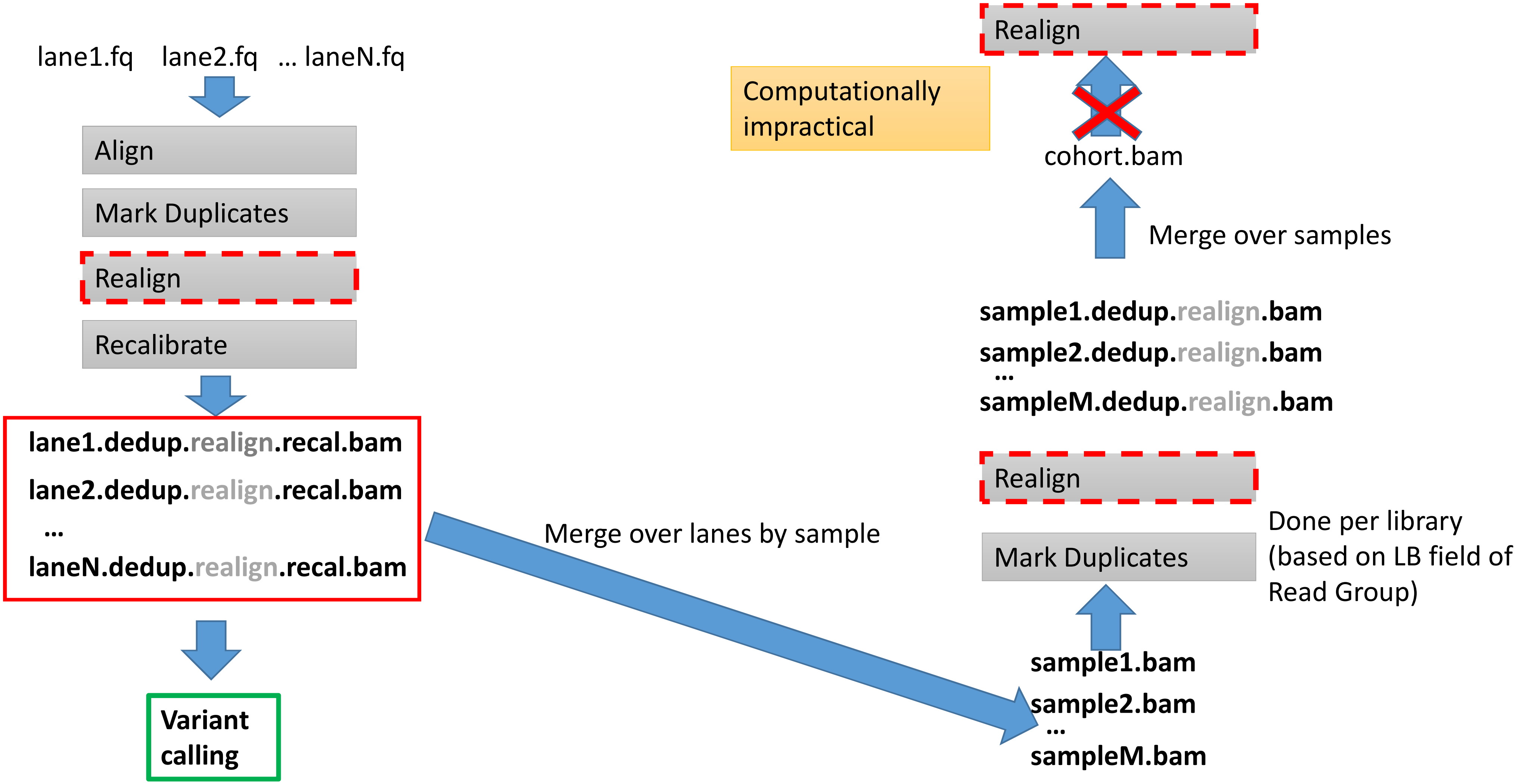
- One file, OR…..
- Multiple files

Generally, read pre-processing is done separately for each FASTQ file or pair, especially if files contain a lot of data. However:

- **Mark Duplicates** works best if given **all reads from a given library** (sometimes scattered among files)
- **Indel realignment** works best with all reads from all samples (cohort)

Meeting these optimal conditions is usually not practical (large computational cost), so compromises have to be made

Typical read preparation pipeline: one sample in a lane



lane1.fq lane2.fq … laneN.fq

Align

Mark Duplicates

Realign

Recalibrate

lane1.dedup.realign.recal.bam

lane2.dedup.realign.recal.bam

…

laneN.dedup.realign.recal.bam

Variant calling

Merge over lanes by sample

sample1.bam

sample2.bam

…

sampleM.bam

Mark Duplicates

Realign

Done per library (based on LB field of Read Group)

sample1.dedup.realign.bam

sample2.dedup.realign.bam

…

sampleM.dedup.realign.bam

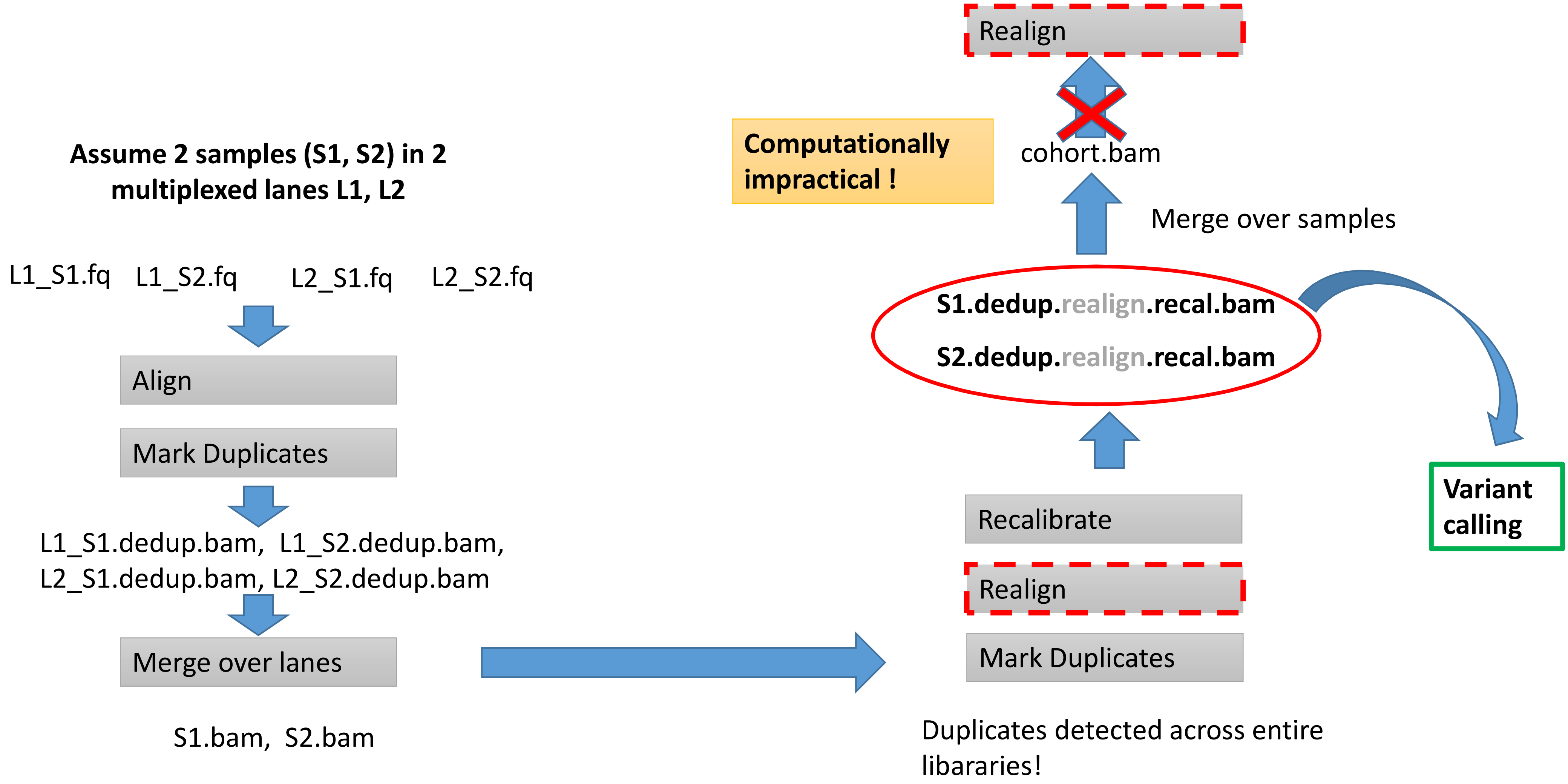
Merge over samples

cohort.bam

Realign

Computationally impractical

Typical read preparation pipeline: multiplexed lanes



Assume 2 samples (S1, S2) in 2 multiplexed lanes L1, L2

L1_S1.fq L1_S2.fq L2_S1.fq L2_S2.fq

Align

Mark Duplicates

L1_S1.dedup.bam, L1_S2.dedup.bam,
L2_S1.dedup.bam, L2_S2.dedup.bam

Merge over lanes

S1.bam, S2.bam

Mark Duplicates

Realign

Recalibrate

Duplicates detected across entire libararies!

S1.dedup.realign.recal.bam

S2.dedup.realign.recal.bam

Variant calling

Merge over samples

cohort.bam

Realign

Computationally impractical !

Input: reads in FASTQ format

FASTQ format: 4 lines per read (“@name”, sequence, “+”, quality string)

```
@61DFRAAXX100204:1:100:10494:3070
ACTGCATCCTGGAAAGAATCAATGGTGGCCGGAAAGTGTTTTTCAAATACAAGAGTGACAATGTGCCCTGTTGTTT
+
ACCCCCCCCCCCCCCCCCCCCCCCCCCCCCBC?CCCCCCCCC@@CACCCCCACCCCCCCCCCCCCCCCCCCCCCCC
```

ASCII code of a letter in quality string **- 33** equals **Phred quality score** of the corresponding base.
older Illumina platforms used **64** instead of **33**

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

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

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:

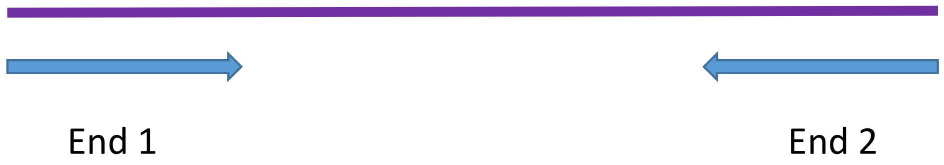
First sequence in “left” file

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

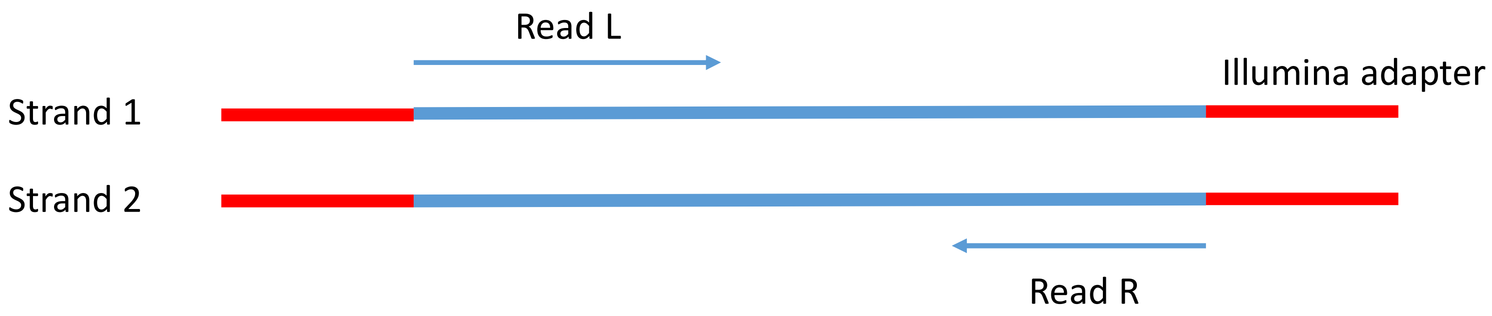
First sequence in “right” file

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

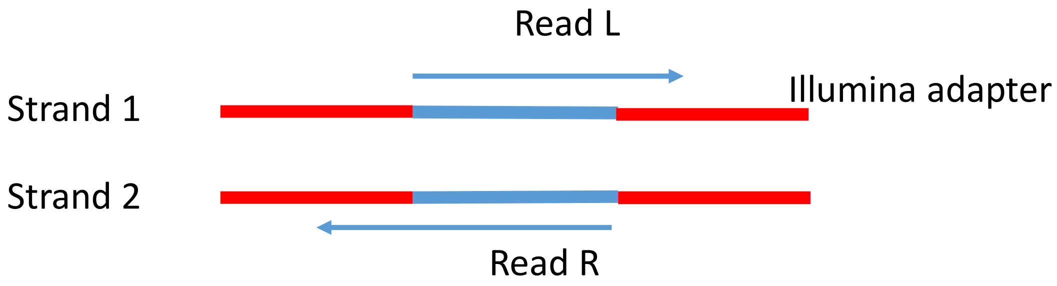
The two ends come from **opposite strands** of the fragment being sequenced



Sequencing long fragment



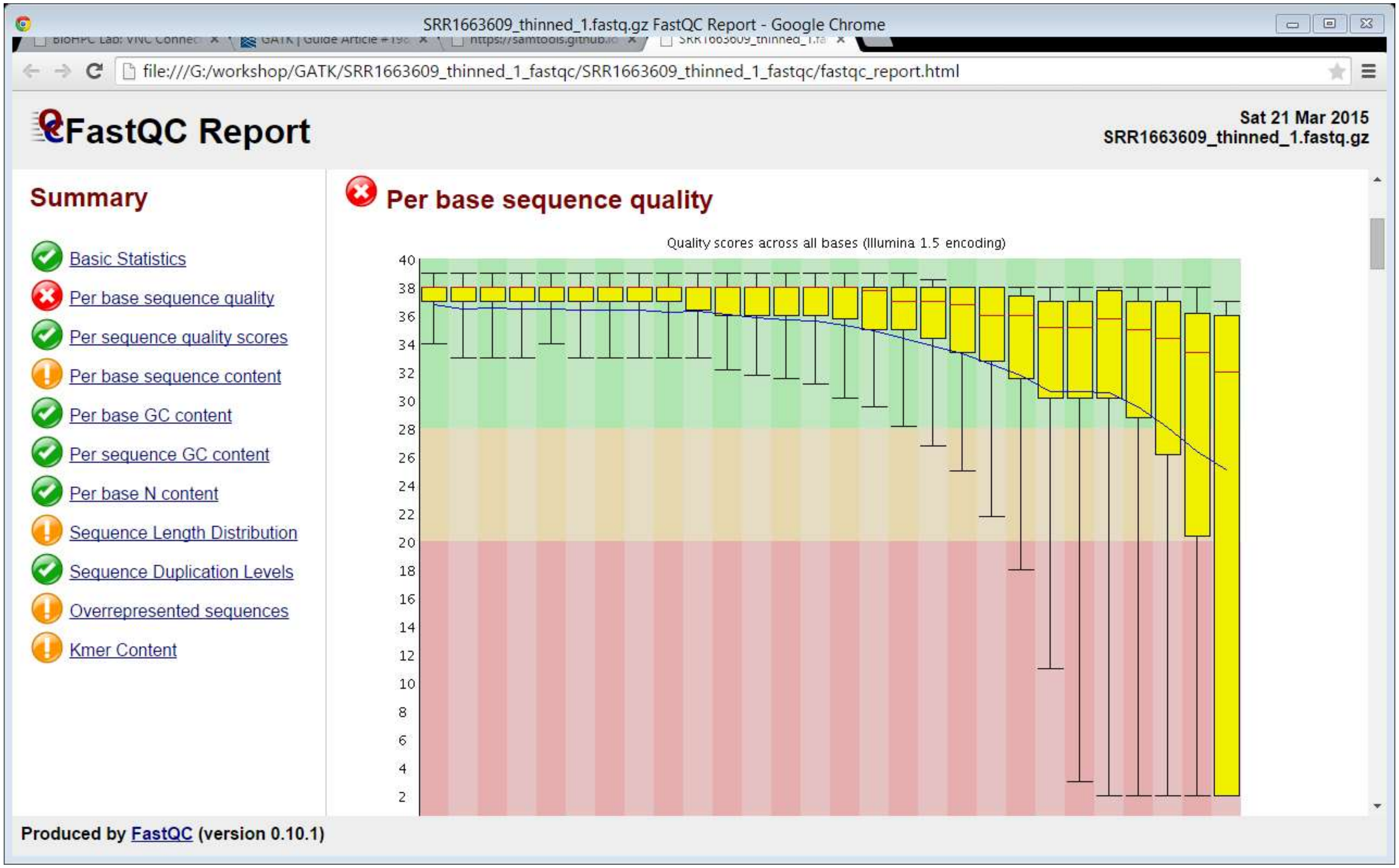
Sequencing short fragment



Read-through: sequenced reads cut into adapters

Adapter remnants should be removed

Read quality assessment with fastqc



SRR1663609_thinned_1.fastq.gz FastQC Report - Google Chrome

file:///G:/workshop/GATK/SRR1663609_thinned_1_fastqc/SRR1663609_thinned_1_fastqc/fastqc_report.html

FastQC Report

Sat 21 Mar 2015
SRR1663609_thinned_1.fastq.gz

Summary

- ✓ Basic Statistics
- ✗ Per base sequence quality
- ✓ Per sequence quality scores
- ! Per base sequence content
- ✓ Per base GC content
- ✓ Per sequence GC content
- ✓ Per base N content
- ! Sequence Length Distribution
- ✓ Sequence Duplication Levels
- ! Overrepresented sequences
- ! Kmer Content

✗ Per base sequence quality

Quality scores across all bases (Illumina 1.5 encoding)

Produced by FastQC (version 0.10.1)

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

Pre-alignment read clean-up

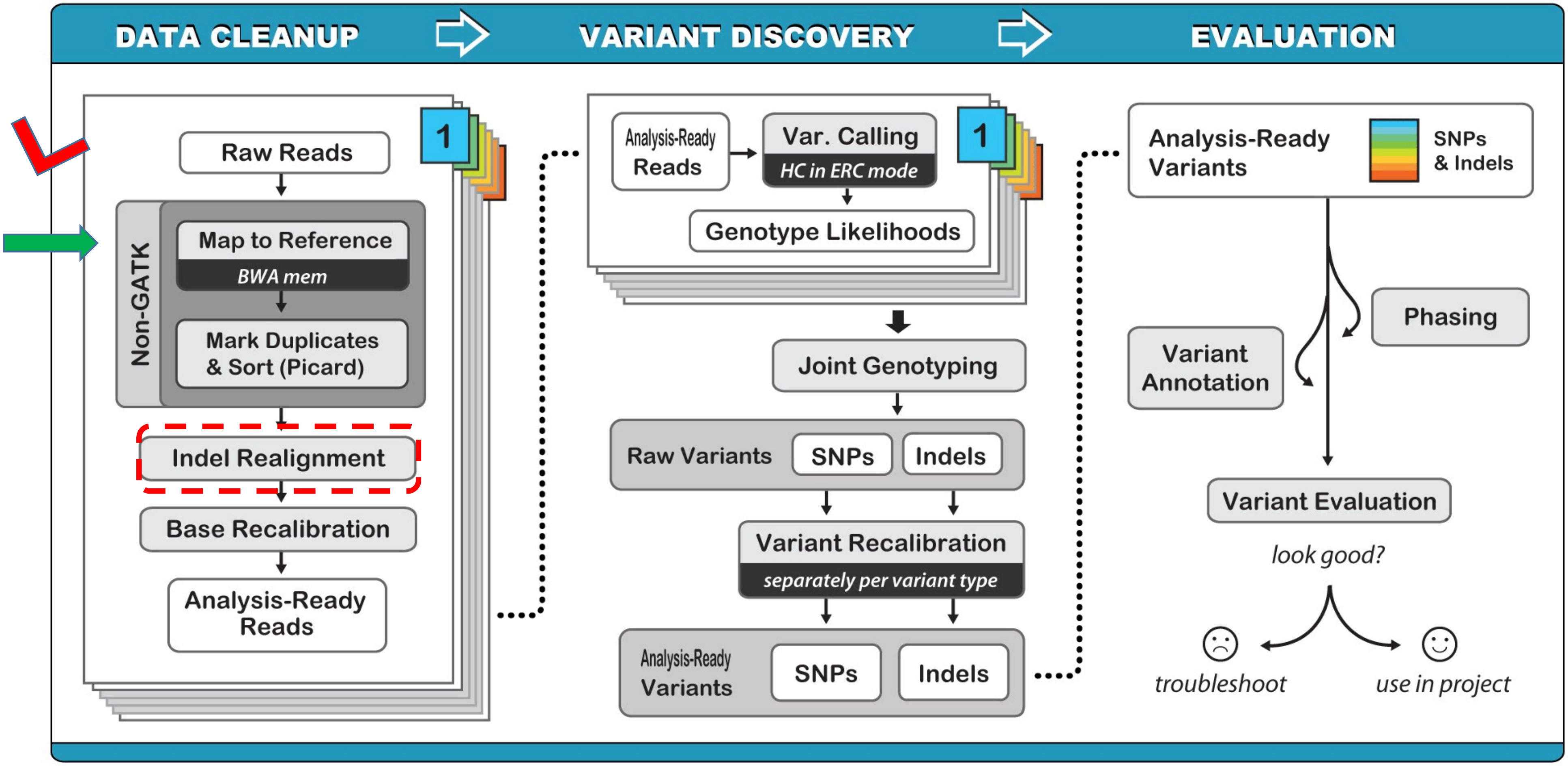
Trimmomatic: A flexible read trimming tool for Illumina NGS data (Bolger et al., http://www.usadellab.org/cms/?page=trimmomatic)

```
java -jar  trimmomatic.jar PE -threads 2 -phred33  \
reads_1.fastq.gz  reads_2.fastq.gz  \
reads_P_1.fastq.gz   reads_U_1.fastq.gz  \
reads_P_2.fastq.gz   reads_U_2.fastq.gz  \
ILLUMINACLIP:TruSeq3-PE.fa:2:30:10  SLIDINGWINDOW:4:5  LEADING:5 TRAILING:5 MINLEN:25
```

Filtering operations (in order specified) performed on each read:

- Remove Illumina adapters (those in file `TruSeq3-PE.fa`) using “palindrome” algorithm (will keep only one copy of a “read-through”)
- Clip read when average base quality over a 4bp sliding window drops below 5
- Clip leading and trailing bases if base quality below 5
- Skip read if shorter than 25bp

“Best Practices” for DNA-Seq variant calling



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**

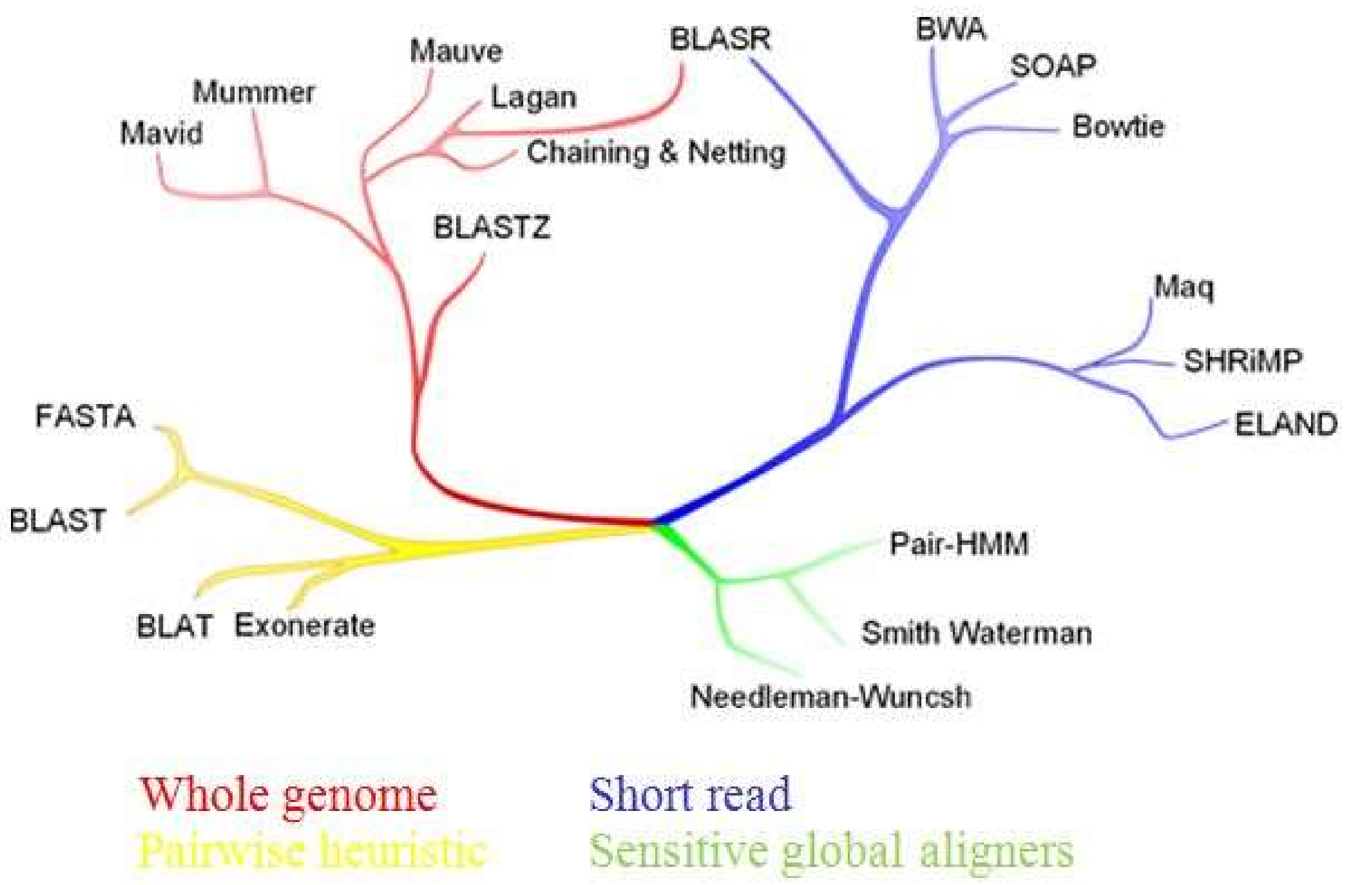
- Some reads cannot be mapped unambiguously in a single location (have low **Mapping Quality**)
  - if reads too short
  - reads originating from paralogs or repetitive regions
  - Having paired-end (PE) data helps

- 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

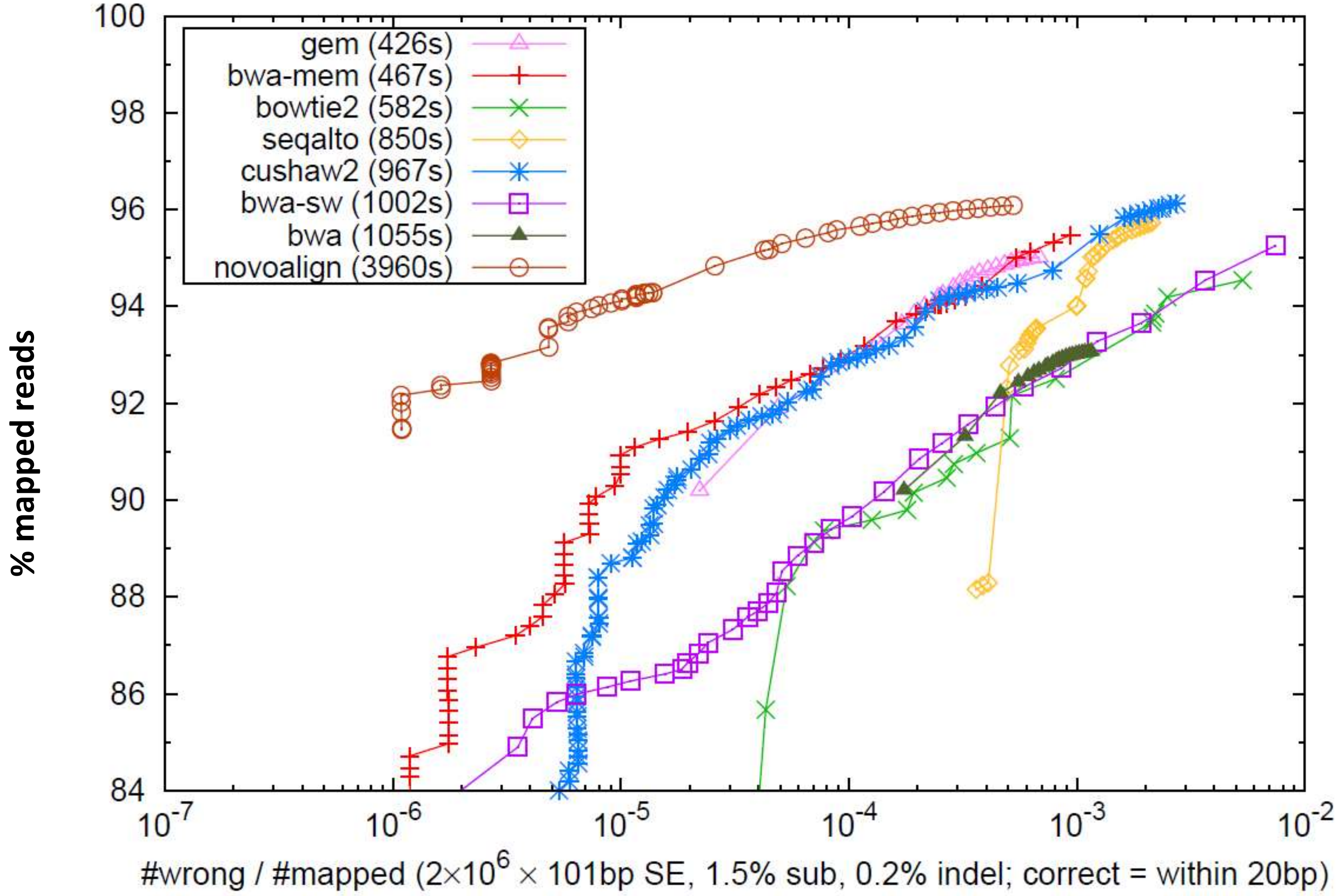
Picking good aligner is important

Aligner phylogeny



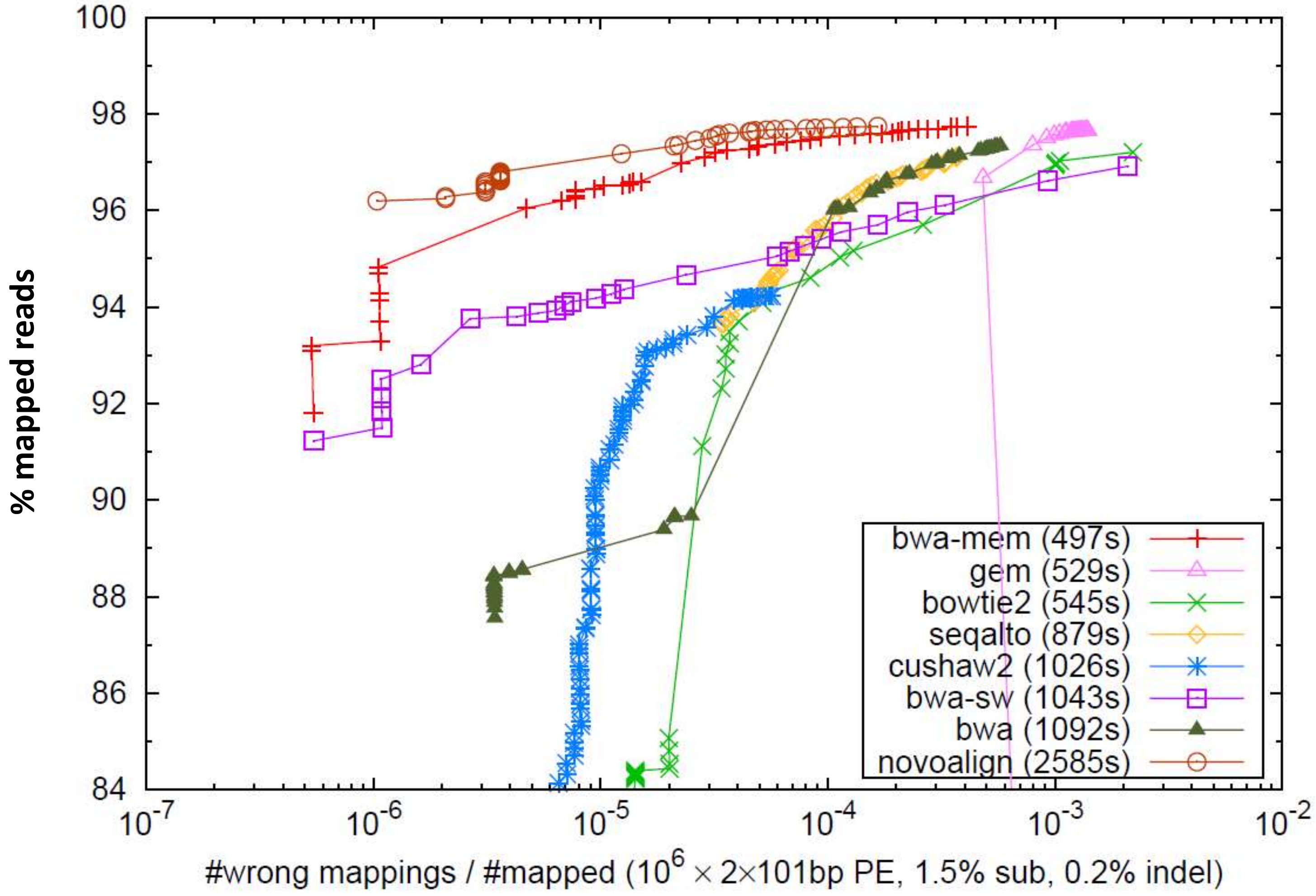
From: Konrad Paszkiewicz, University of Exeter, http://evomics.org/2014/01/alignment-methods/

Performance of various aligners on simulated short reads (SE) from human genome



From: Li (Broad Institute), http://arxiv.org/pdf/1303.3997v2.pdf

Performance of various aligners on simulated short reads (PE) from human genome



From: Li (Broad Institute), http://arxiv.org/pdf/1303.3997v2.pdf

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: index reference genome

First things first: **Index reference genome**

```
bwa index genome.fa
```

Will create a bunch of BWA index files: `genome.fa.ann, genome.fa.bwt, genome.fa.fai, genome.fa.pac, genome.fa.sa`

```
samtools faidx genome.fa

java -jar $PICARDDIR/picard.jar CreateSequenceDictionary R=genome.fa  O=genome.dict
```

Will create two auxiliary files, `genome.fa.fai` and `genome.dict` containing summary information about lengths of chromosomes and where they start. Both files are needed by GATK (not by BWA aligner)

This step has to be done only once for each reference genome. The index files may be stored in a separate directory and reused.

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**.

SAM to BAM conversion, sorting and indexing

SAM = Sequence Alignment/Map
BAM = Binary Alignment/Map

SAM format is wasteful (text files take a lot of space on disk) – better to convert it to a more compact, binary format called **BAM**. Typically, we also **sort** the alignments over genomic coordinate and **index** them:

Using samtools

```
samtools view –Sb sample1.sam > sample1.bam
samtool sort sample1.bam –o sample1.sorted.bam
samtools index sample1.sorted.bam
```

Using PICARD

```
java -jar SortSam.jar INPUT=sampl1.sam \
OUTPUT=sample1.sorted.bamSORT_ORDER=coordinate

java -jar BuildBamIndex.jar INPUT=
```

Indexing will create a small file called `sample1.sorted.bam.bai` (or `sample1.sorted.bai`)

It is a **"table of contents"** to quickly point from genomic coordinates to overlapping alignment records

Shortcut: avoid generating large SAM files

```
bwa mem [options] reads_1.fq.gz reads_2.fq.gz | samtools view –Sb - >  sample.bam
```

Pipe operator: passes STDOUT of command on the left to STDIN of command on the right

"-" tells **`samtools`** to take input from STDIN

Output from **`bwa mem`** (a large text file in SAM format written to STDOUT) is **piped** into the **`samtools`** command which converts it into (much smaller) file in **BAM** format "on the fly".

No more large SAM file to store and handle!

Back to BWA mem command: define Read Group

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

What will this option do?

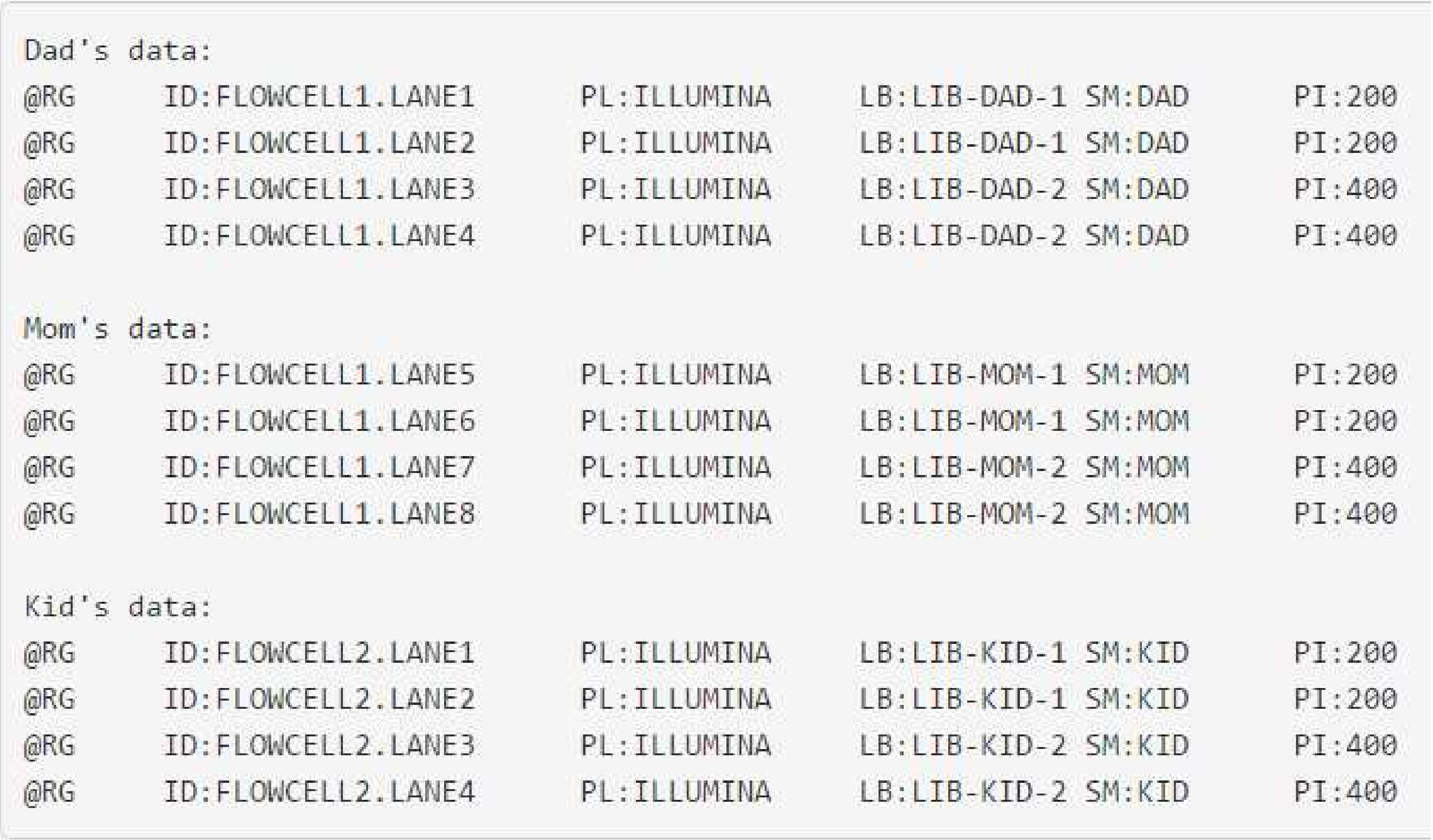
The SAM/BAM file header will contain a line (TAB-delimited) defining the group:

@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)	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** and **one library**
A libray may be sequenced multiple times (on different 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**

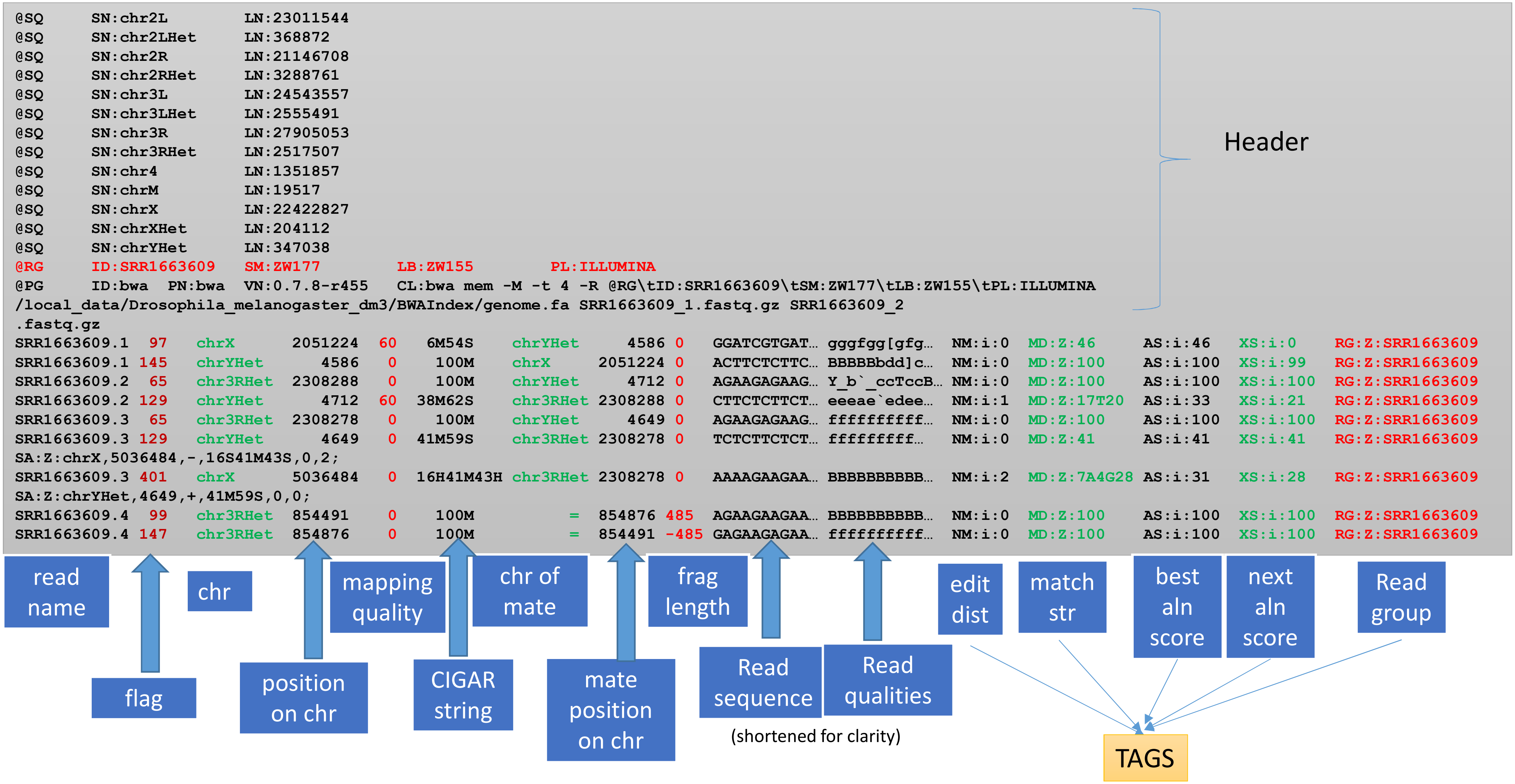
```
@RG    ID:HL5WNCCXX_2_A    SM:A    LB:mylib       PL:ILLUMINA
@RG    ID:HL5WNCCXX_3_A    SM:A    LB:mylib       PL:ILLUMINA

@RG    ID:HL5WNCCXX_2_B    SM:B    LB:mylib       PL:ILLUMINA
@RG    ID:HL5WNCCXX_3_B    SM:B    LB:mylib       PL:ILLUMINA
```

Forgot to add Read Group at alignment step? No problem, just use PICARD tool:

```
java -jar $PICARDDIR/picard.jar \
AddOrReplaceReadGroup \
INPUT=input.bam \
OUTPUT=input_with_rgroup.bam  \
SORT_ORDER=coordinate \
RGSM=my_sample \
RGPU=none \
RGID=my_groupID \
RGLB=my_library \
RGPL=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… ffffffffff…  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… ffffffffff…  NM:i:0  MD:Z:100    AS:i:100   XS:i:100   RG:Z:SRR1663609
```

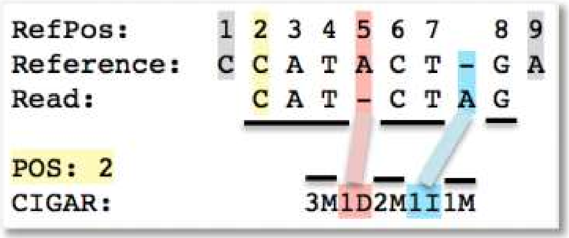
(shortened for clarity)

Anatomy of a SAM file

- **Position**: 1-based position of the first read base on the chromosome

- **Mapping Quality**: phred probability the read is in the wrong place (i.e., the higher MAPQ the better)

- **CIGAR**: Compact Idiosyncratic Gapped Alignment Report – shows how many indels, how many bases soft- or hard-clipped
  - **100M** whole read aligned (no clips), no indels
  - **16H41M43H** 16 bp clipped from the beginning of the read, 43 bp clipped from the end, 41 remaining bases aligned with no indels
  - **52S48M** 52 bp soft-clipped from the beginning (i.e., these bases are still shown, but do not take part in alignment), the other 48 aligned without indel
  - **3M1D2M1I1M** 3 bases aligned followed by 1 base deleted, 2 next ones aligned, 1 base inserted and the last one aligned



```
RefPos:     1 2 3 4 5 6 7   8 9
Reference:  C C A T A C T - G A
Read:         C A T - C T A G

POS: 2
CIGAR:            3M1D2M1I1M
```

- **Fragment length**: distance in bp between positions of 1st bases of the two reads in a pair

Anatomy of a SAM file, cnt

Tags: some universal, others supplied by a particular aligner and specific to it
Here are the ones produced by **BWA mem**:

TAG	Example	What it means
NM	NM:i:1	Number of mismatches (integer value)
MD	MD:Z:17T20	17 matches, then some other base in place of T, then 20 more matches (counting from beginning of read)
AS	AS:i:100	Alignment score 100 (integer)
XS	XS:i:21	Second-best alignment score 21 (integer)
RG	RG:Z:SRR166309	Read Group ID
SA	SA:Z:chrYHet,4649,+,41M59S,0,0	Location and tags of second-best hit

What is “flag”?

Let’s covert the “flag” number to **binary** representation. For example,

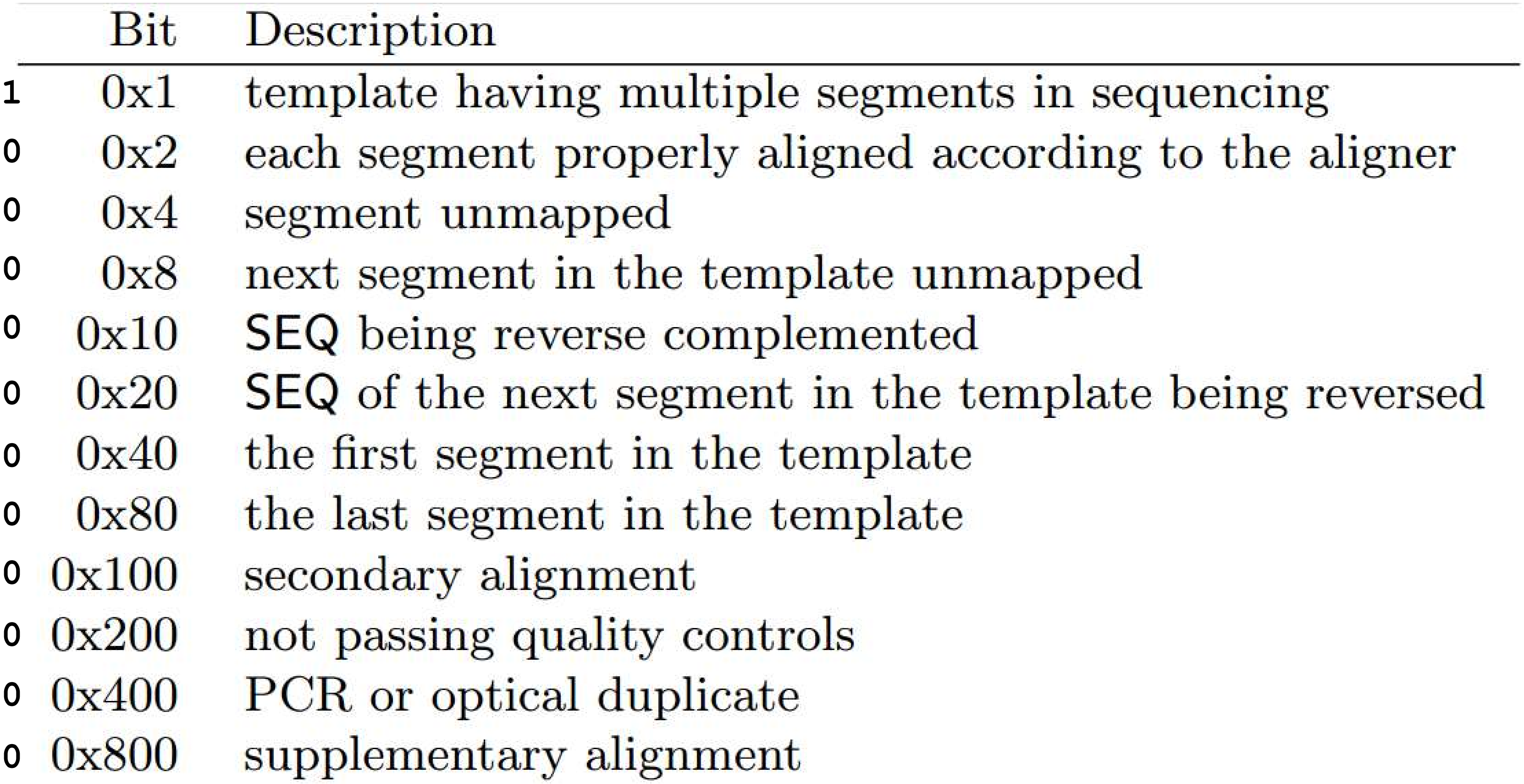
Flag (decimal)	Flag (hex)	Flag (binary)
`145`	`0x91`	`10010001`
`129`	`0x81`	`10000001`
`97`	`0x61`	`1100001`

The positon (counted from right to left) in binary number corresponds to some property of this read’s alignment.

An “1” in a given position says the read **has** the corresponding property

A “0” means the read **does not have** the corresponding property

What bit flags mean



Binary	Hex	
	Bit	Description
000000000001	0x1	template having multiple segments in sequencing
000000000010	0x2	each segment properly aligned according to the aligner
000000000100	0x4	segment unmapped
000000001000	0x8	next segment in the template unmapped
000000010000	0x10	SEQ being reverse complemented
000000100000	0x20	SEQ of the next segment in the template being reversed
000001000000	0x40	the first segment in the template
000010000000	0x80	the last segment in the template
000100000000	0x100	secondary alignment
001000000000	0x200	not passing quality controls
010000000000	0x400	PCR or optical duplicate
100000000000	0x800	supplementary alignment

Flag decoded

145 = 00010010001

- Read is a part of a fragment in sequencing (00000000001) – they all do, because our data is all PE reads

- Read aligns to reference as reverse complement (00000010000)

- Read is the second end of the fragment (00010000000)

In alignment of SE reads, the only flags possible are 0 (mapped on forward strand), 16 (mapped to reverse strand), or 4 (unmapped)

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)
```

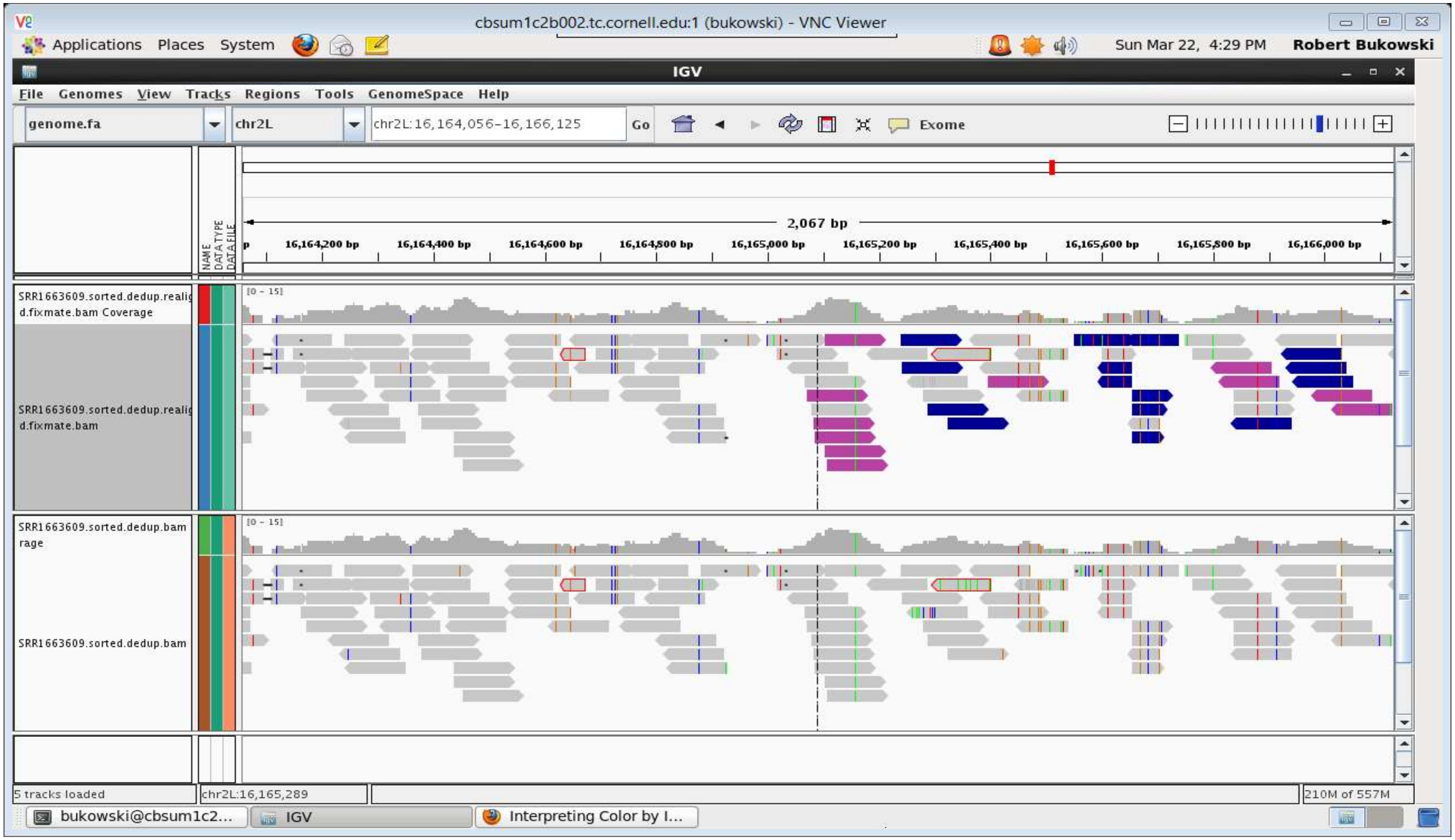
Looking into a BAM file: IGV viewer

Look at multiple BAM files

Zoom in and out

Various color-coding schemes

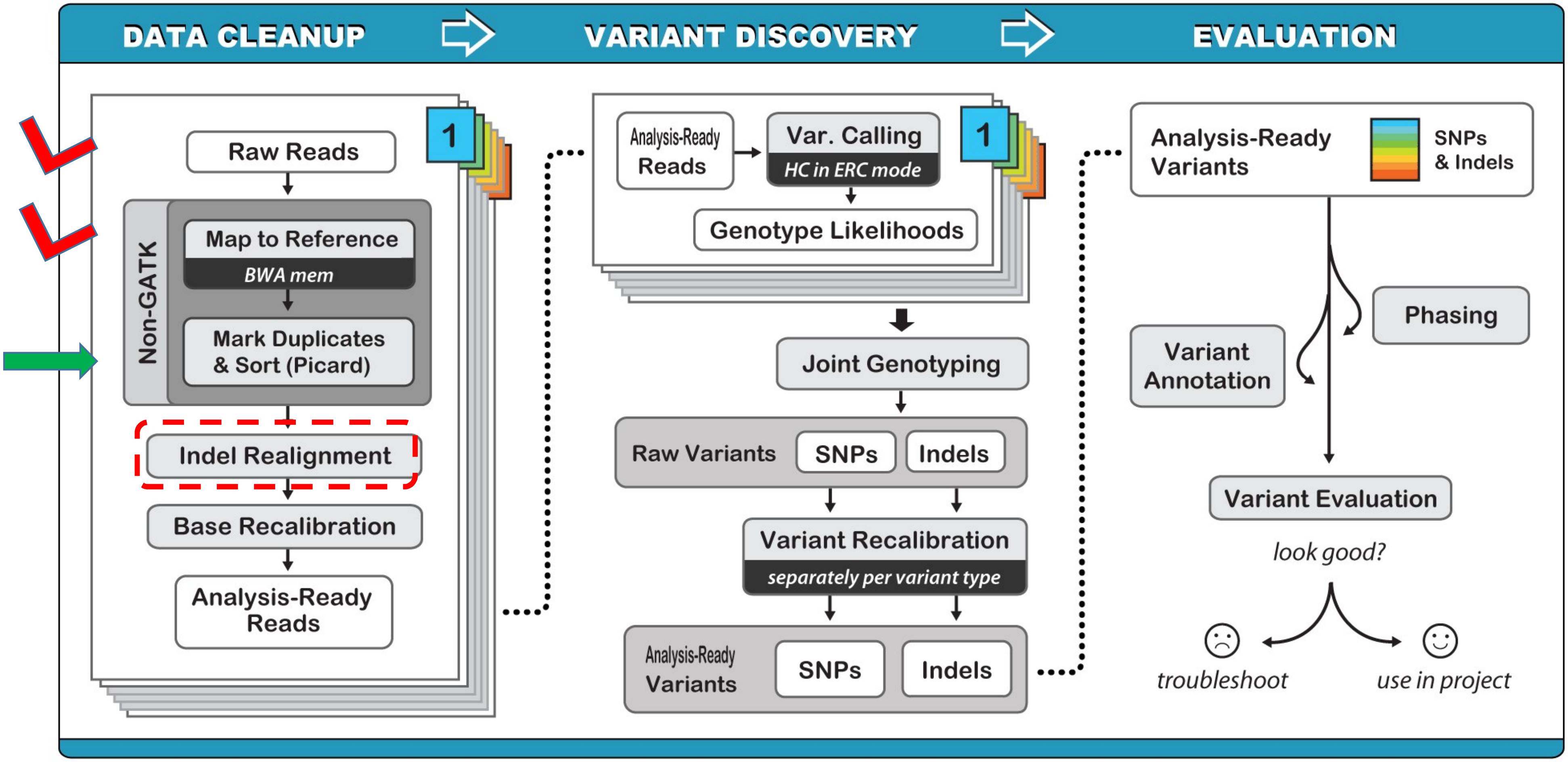
Can load genome annotation track


cbsum1c2b002.tc.cornell.edu:1 (bukowski) - VNC Viewer
Applications Places System
Sun Mar 22, 4:29 PM
Robert Bukowski
IGV
File Genomes View Tracks Regions Tools GenomeSpace Help
genome.fa
chr2L
chr2L:16,164,056-16,166,125
Go
Exome
2,067 bp
16,164,200 bp
16,164,400 bp
16,164,600 bp
16,164,800 bp
16,165,000 bp
16,165,200 bp
16,165,400 bp
16,165,600 bp
16,165,800 bp
16,166,000 bp
NAME
DATA TYPE
DATA FILE
SRR1663609.sorted.dedup.realigned.fixmate.bam Coverage
[0 - 15]
SRR1663609.sorted.dedup.realigned.fixmate.bam
SRR1663609.sorted.dedup.bam Coverage
[0 - 15]
SRR1663609.sorted.dedup.bam
5 tracks loaded
chr2L:16,165,289
210M of 557M
bukowski@cbsum1c2...
IGV
Interpreting Color by I...

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

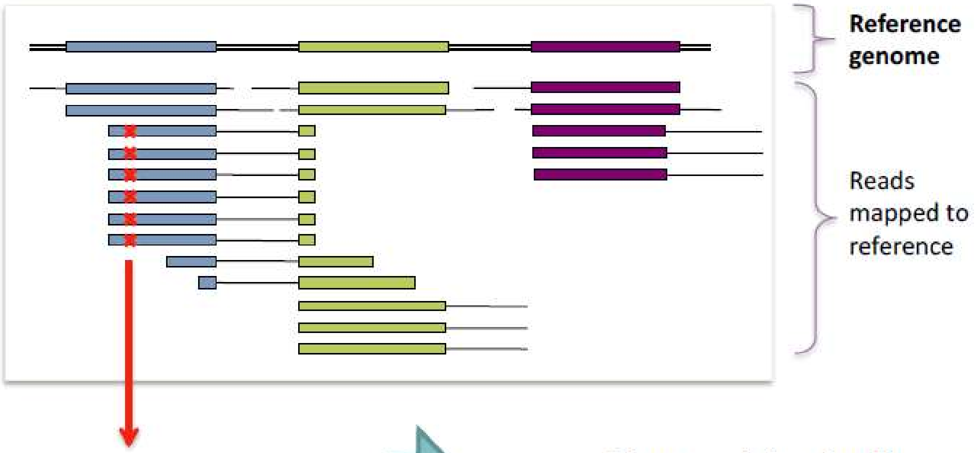
  - Very similar in sequence (except sequencing errors).
  - Identified where the first, say, 50 bases are identical between two reads and the read's coordinates are close

- **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

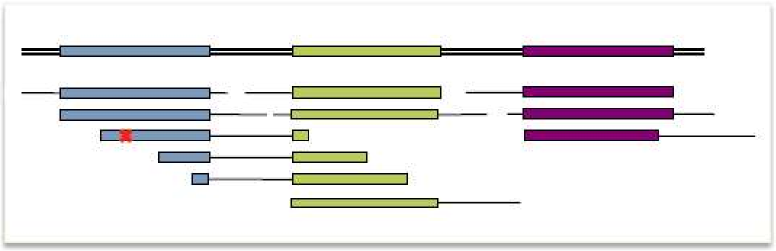


Reference genome

Reads mapped to reference

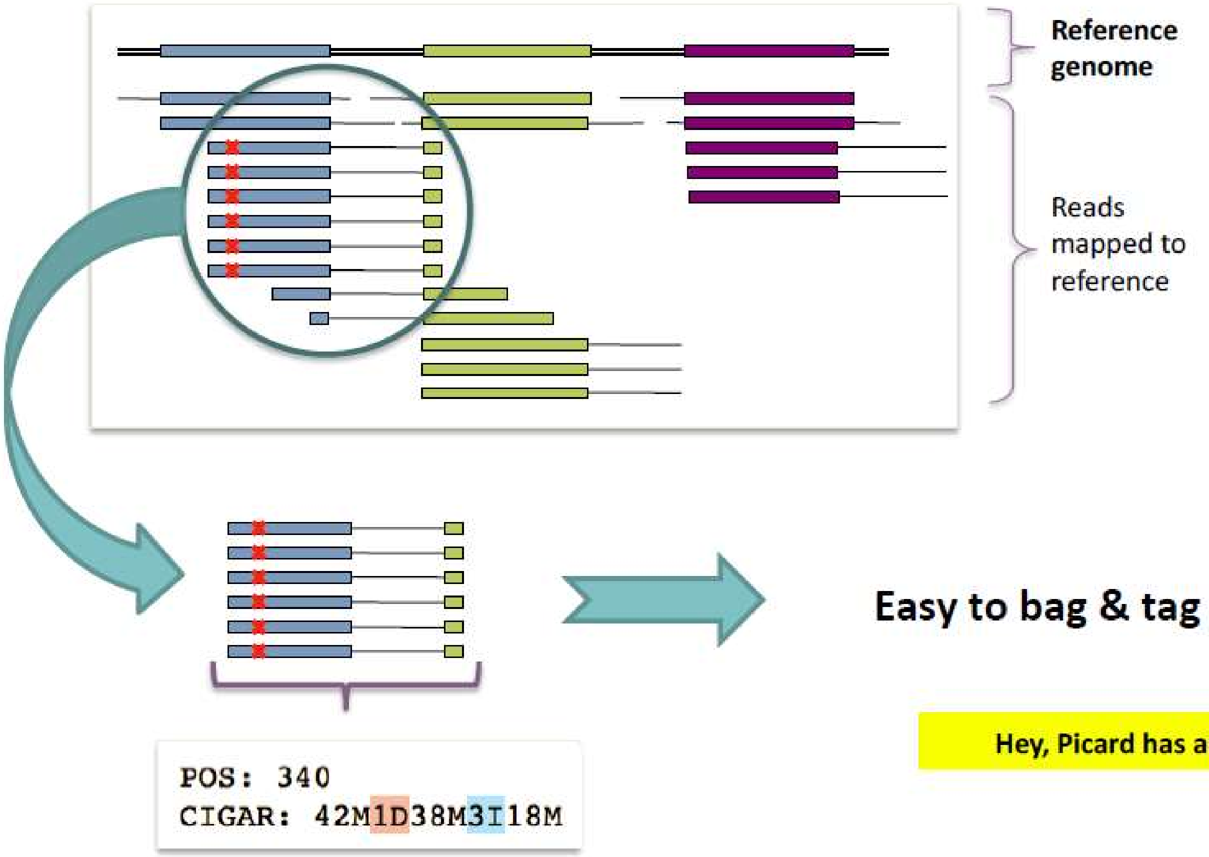
FP variant call (bad)

After marking duplicates, the GATK will only see :



… and thus be more likely to make the right call

How removing (marking) duplicates works



```
POS: 340
CIGAR: 42M1D38M3I18M
```

Easy to bag & tag

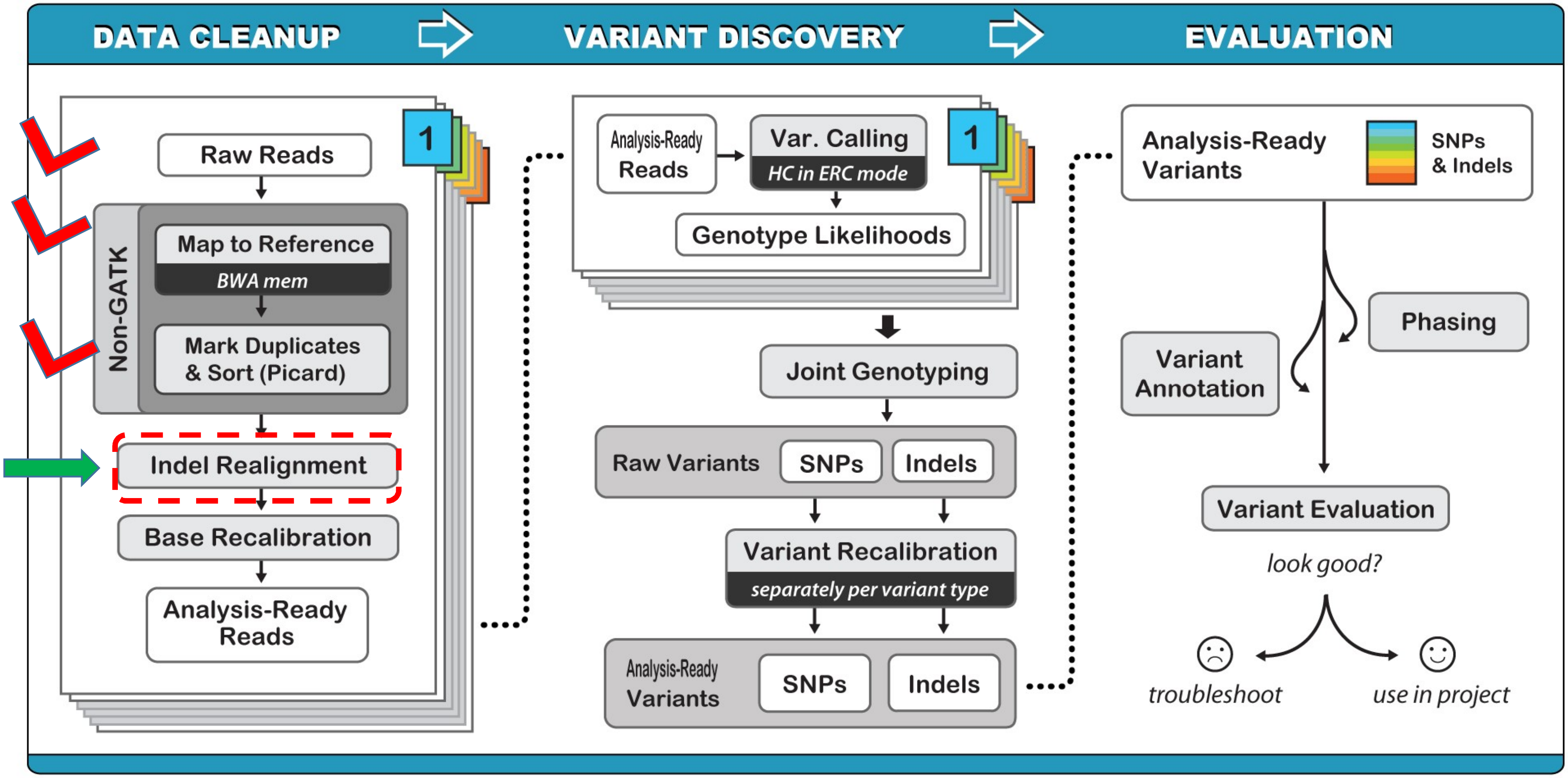
Hey, Picard has an app for that!

Removing (marking) duplicates with PICARD

```
java -jar $PICARDDIR/picard.jar \
MarkDuplicates \
INPUT=sample1.sorted.bam \
OUTPUT=sample1.sorted.dedup.bam \
METRICS_FILE=sample1.sorted.dedup.metrics.txt
```

- 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



Ambiguity of alignment at indel sites

```
Reference  CTTTAGTTTCTTTT----CTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC

           CTTTAGTTTCTTTT----GCCGCTTTCTTTCTTTCTT
           CTTTAGTTTCTTTT----GCCGCTTTCTTTCTTTCTT
Reads      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

But we can try to shift things around a bit:

```
Reference  CTTTAGTTTCTTTT----CTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC

           CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTT
           CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTT
Reads      CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
           CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
           CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
           CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
```

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

This looks better !

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

Ambiguity of alignment: around adjacent SNPs

Reference

```
AAGCGTCG
```

Reads

```
AAGCGTCG
AAGCGTCG
AAGCGTCG
AAGCTACG
AAGCTACG
AAGCTACG
```

What is better: **3 adjacent SNPs** or an **insertion**?

Reference

```
AAG---CGTCG
```

Reads

```
AAG---CGTCG
AAG---CGTCG
AAG---CGTCG
AAGCTACG
AAGCTACG
AAGCTACG
```

Ambiguity of alignment: around homo-polymer runs flanked by adjacent SNPs

Reference

```
...CCCATTTTTTTCTAAAAGCTGGCAT...

    CCCATTTTTTCTAAAAGCTGGCAT...
    CCCATTTTTTCTAAAAGCTGGCAT...
    CCCATTTTTTCTAAAAGCTGGCAT...
...CCCATTTTTTCTAAAA
...CCCATTTTTTCTAAAA
...CCCATTTTTTCTAAAA
```

Reads

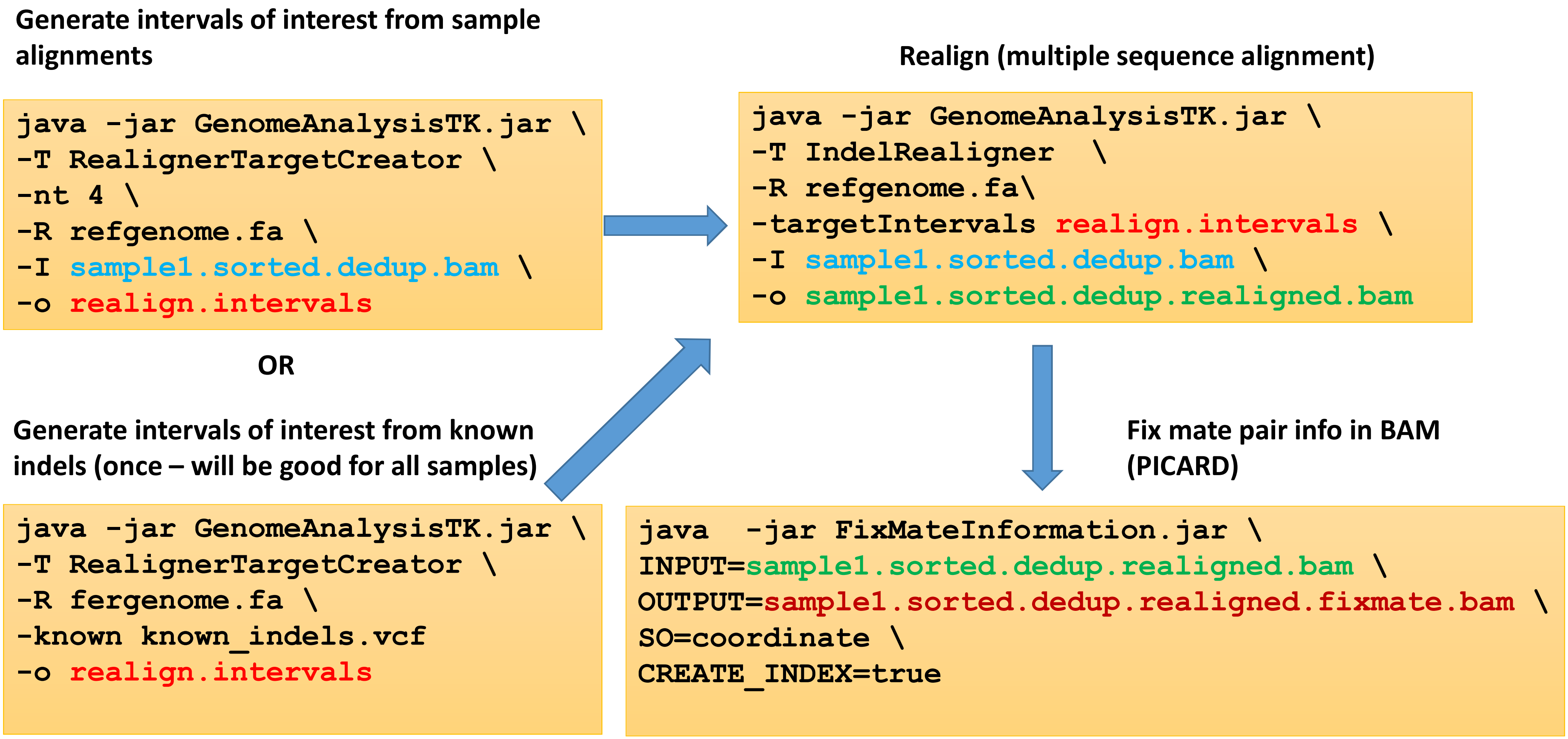
Reference

```
...CCCATTTTTTTCTAAAAGCTGGCAT...

   CCCA-TTTTTTCTAAAAGCTGGCAT...
   CCCA-TTTTTTCTAAAAGCTGGCAT...
   CCCA-TTTTTTCTAAAAGCTGGCAT...
...CCCA-TTTTTTCTAAAA
...CCCA-TTTTTTCTAAAA
...CCCA-TTTTTTCTAAAA
```

Reads

Remedy: local realignment



Generate intervals of interest from sample alignments

```
java -jar GenomeAnalysisTK.jar \
-T RealignerTargetCreator \
-nt 4 \
-R refgenome.fa \
-I sample1.sorted.dedup.bam \
-o realign.intervals
```

OR

Generate intervals of interest from known indels (once – will be good for all samples)

```
java -jar GenomeAnalysisTK.jar \
-T RealignerTargetCreator \
-R fergenome.fa \
-known known_indels.vcf
-o realign.intervals
```

Realign (multiple sequence alignment)

```
java -jar GenomeAnalysisTK.jar \
-T IndelRealigner  \
-R refgenome.fa\
-targetIntervals realign.intervals \
-I sample1.sorted.dedup.bam \
-o sample1.sorted.dedup.realigned.bam
```

Fix mate pair info in BAM (PICARD)

```
java  -jar FixMateInformation.jar \
INPUT=sample1.sorted.dedup.realigned.bam \
OUTPUT=sample1.sorted.dedup.realigned.fixmate.bam \
SO=coordinate \
CREATE_INDEX=true
```

Local realignment: when is it needed?

Local re-alignment is time-consuming!

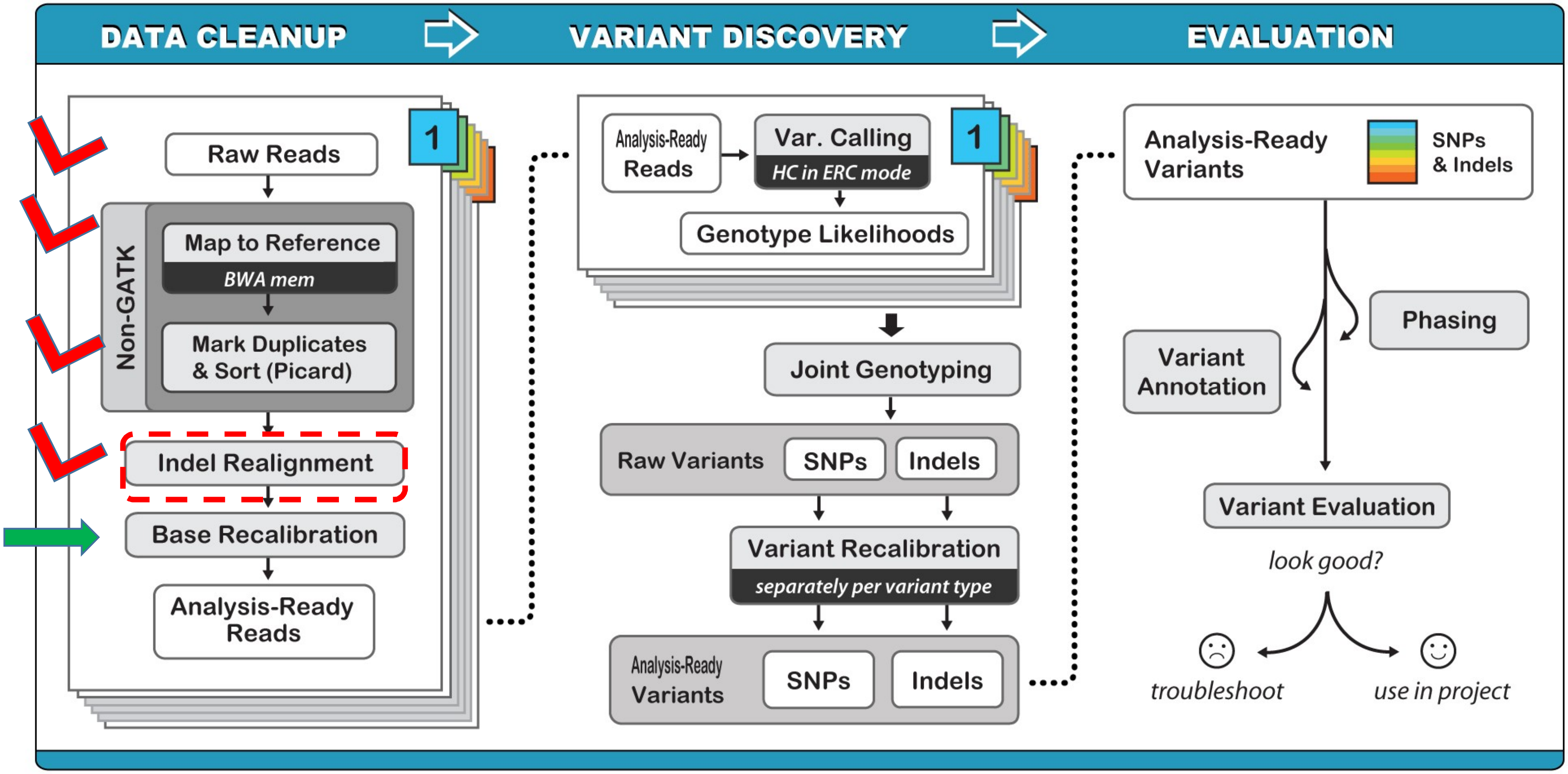
Re-alignment no longer recommended if the genotyping method used downstream involves **local haplotype assembly**

HaplotypeCaller (from GATK)
FreeBayes
re-alignment implicit in the assembly algorithm

Still needed if the genotypes called from allelic depths at individual sites

UnifiedGenotyper (GATK)
samtools

“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

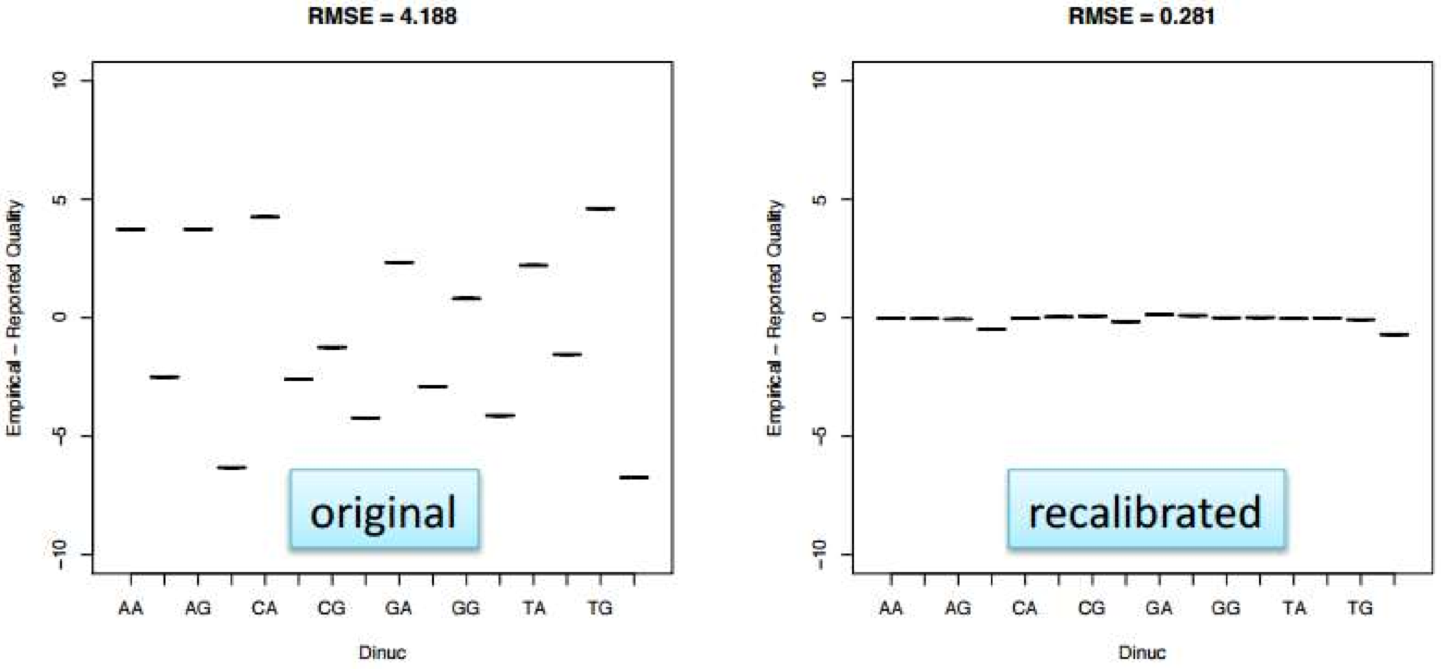
- Scan all reads (i.e., bases) again (in a BAM file)
- For each base
    - Classify into a bin
    - Apply bin-specific correction to base quality scores (based on the database collected in previous step)

Caveats:

- Local realignment should be done before recalibration
- 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

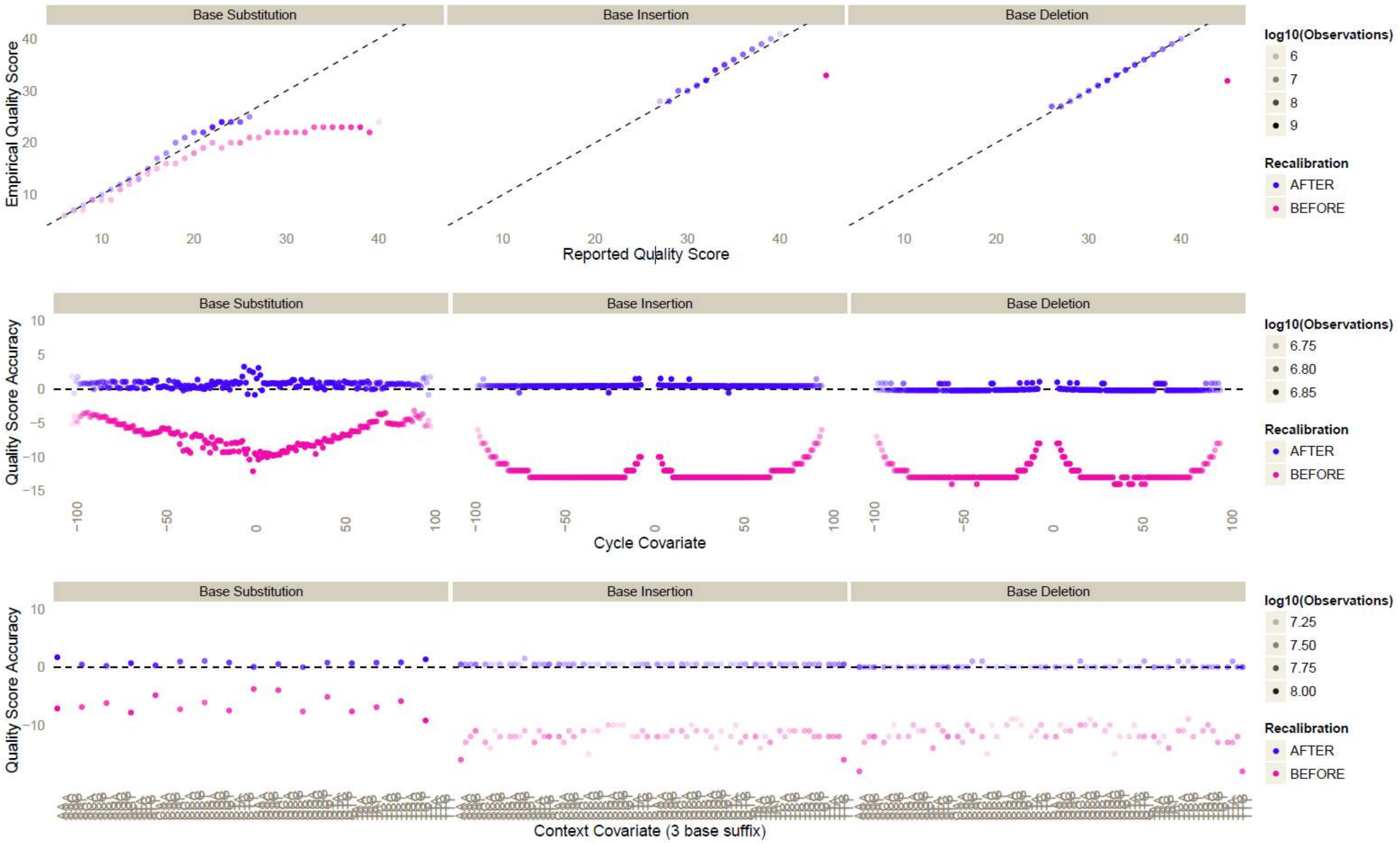
Collect mismatch statistics in bins

```
java -jar GenomeAnalysisTK.jar  \
-T BaseRecalibrator  \
-R refgenome.fasta\
-knownSites known_snps_indels.vcf  \
-I sample1.sorted.dedup.realigned.fixmate.bam \
-o sample1.sorted.dedup.realigned.fixmate.recal_data.table \
-cov ReadGroupCovariate  \
-cov QualityScoreCovariate \
-cov CycleCovariate
```

Recalibrate base qualities in the BAM file

```
java -jar GenomeAnalysisTK.jar  \
-T PrintReads  \
-R refgenome.fasta \
-BQSR sample1.sorted.dedup.realigned.fixmate.recal_data.table \
-I sample1.sorted.dedup.realigned.fixmate.bam \
-o sample1.sorted.dedup.realigned.fixmate.recal.bam
```

This is what recalibration results may look like



Running things in parallel

Alignment

Multithreading in BWA mem works well up to 10-15 CPUs. On a machine with 24 CPUS, run 2 BWA mem jobs concurrently, each on 10 threads (**`bwa mem –t 10`** …) .

Mark Duplicates

Realign

Recalibrate

Multithreading non-existent or not too efficient – best to execute this part of pipeline as multiple independent jobs (one per lane or sample/lane), run in parallel on one or multiple machines. Required memory and disk access bandwidth will determine the optimal number of concurrent jobs per machine. Experiment!

Running alignment preparation in parallel

Alignment

Multithreading in BWA mem works well up to 10-15 CPUs. On a machine with 24 CPUS, run 2 BWA mem jobs concurrently, each on 10 threads (**`bwa mem –t 10`** …) .

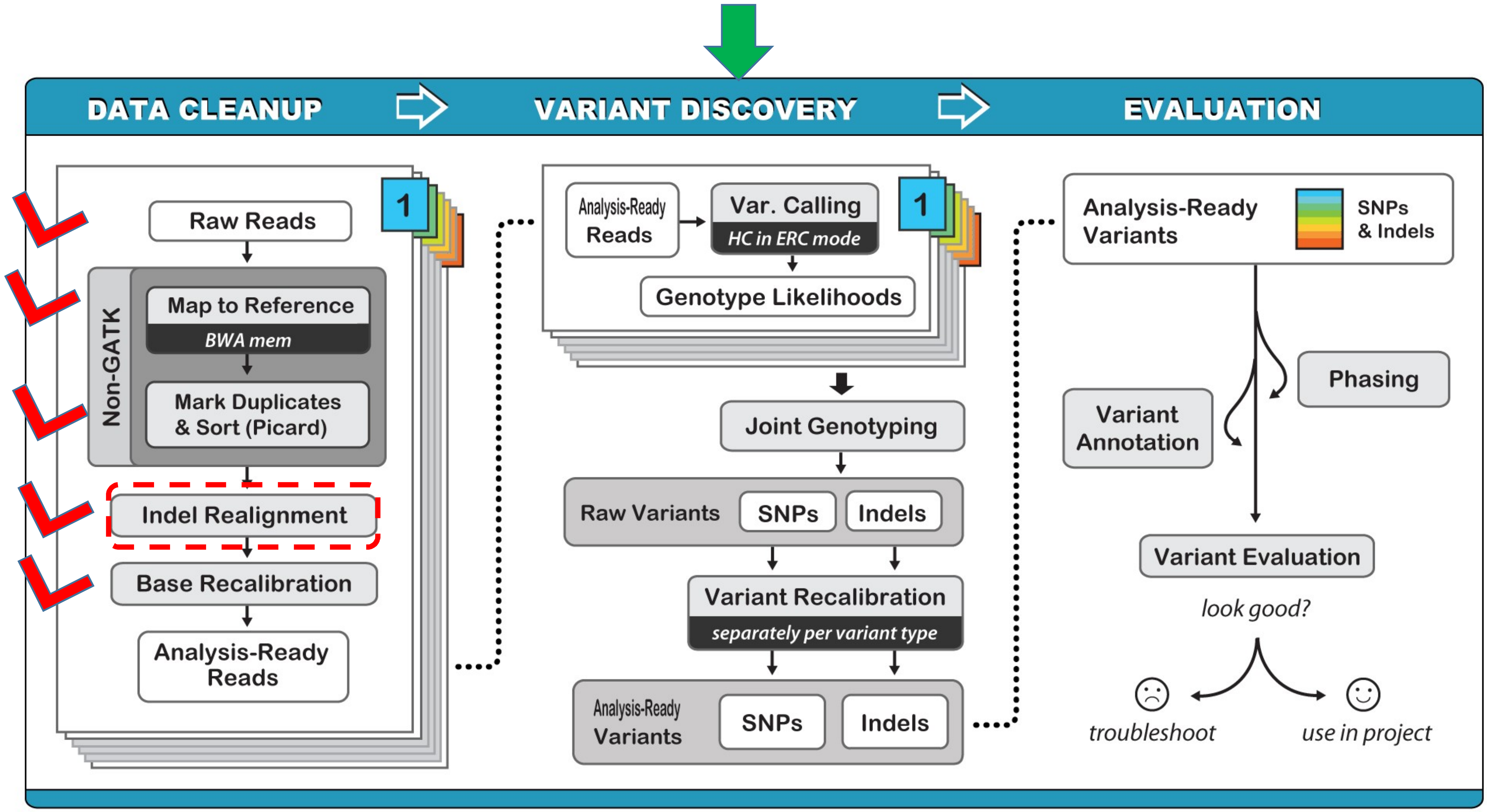
Mark Duplicates

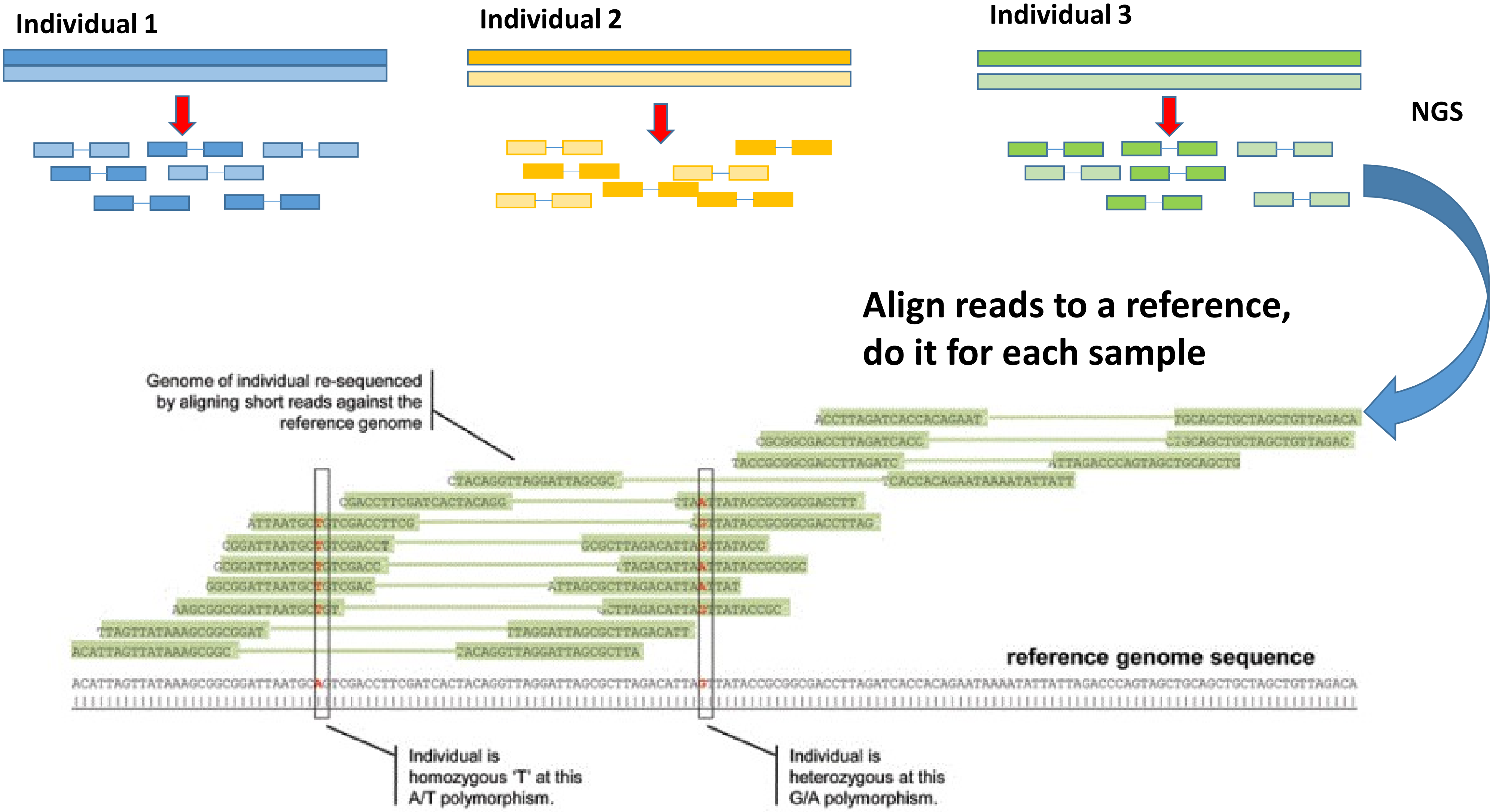
Realign

Recalibrate

- Multithreading non-existent or not too efficient
- best to execute this part of pipeline as multiple independent jobs (one per lane or sample/lane), run in parallel on one or multiple machines.
- Required memory and disk access bandwidth will determine the optimal number of concurrent jobs per machine. Experiment!
- These steps take **a long time** (may be comparable with alignment itself!)

“Best Practices” for DNA-Seq variant calling





Individual 1

Individual 2

Individual 3

NGS

Align reads to a reference,
do it for each sample

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.

Objectives…..

For each site on the genome determine

- Whether or not there is **any variation** at this site (across all samples)
- If there is variation, **assign a genotype** (for diploids: pair of alleles) to each sample
- Decide which sites can be considered **variant** from all samples' perspective and report these sites (positions, alleles, sample genotypes, …)

How it's done?

<u>The old days:</u>

Call variant base on thresholds (e.g., ratio of reference/non-reference bases)
Assign genotypes based on other thresholds

<u>Now-days: probabilistic framework</u>

- Calculate and report probabilities (or **likelihoods**) of various sample genotypes (given read alignments)
- Calculate and report **probability** of a site being a variant (given read alignments)
- Input: read alignments, read **base quality scores** (preferably **recalibrated**), read mapping quality

How to describe variants: Variant Call Format (VCF)

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)

How to describe variants: 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

How to describe variants: 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:

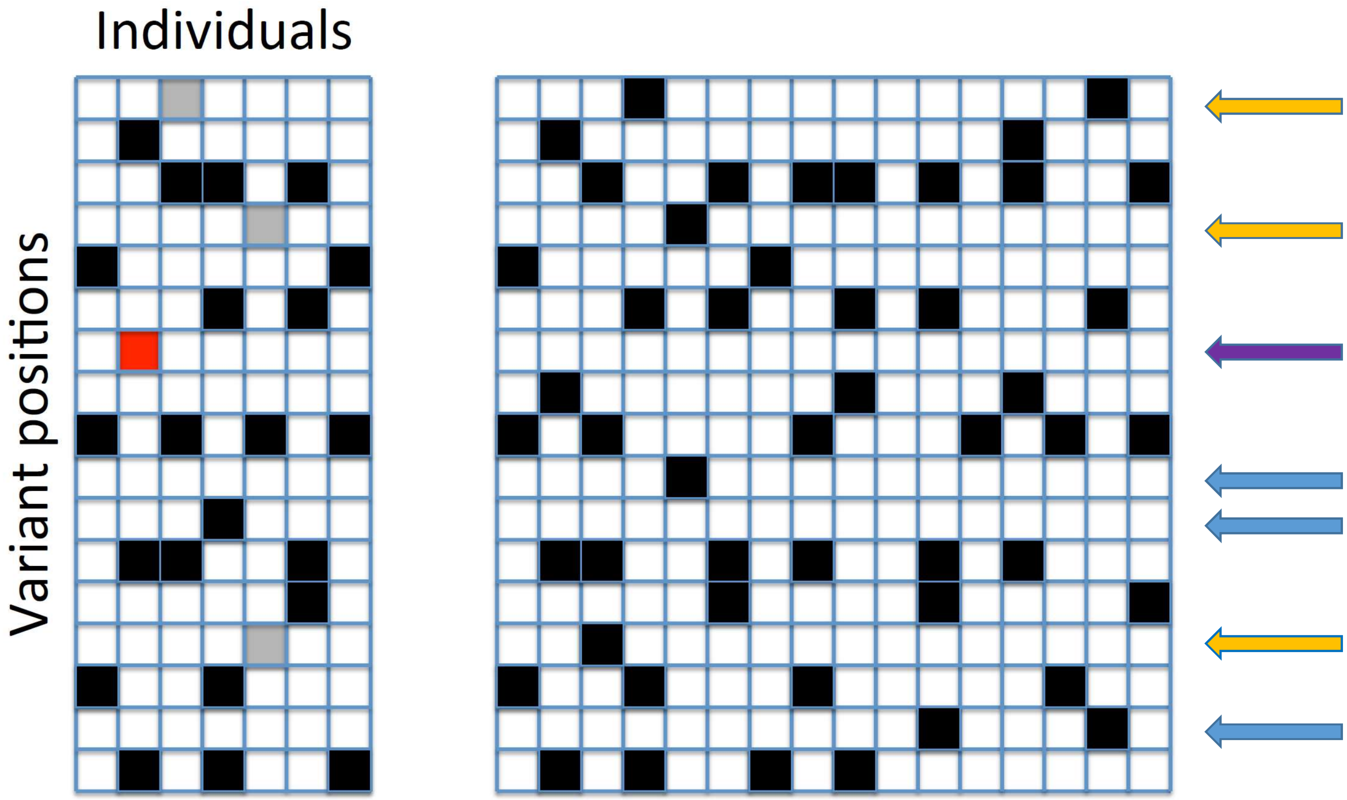
```
AC=2;AF=0.333;AN=6;DP=16;FS=0.000;GQ_MEAN=16.00;GQ_STDDEV=10.54;MLEAC=2;MLEAF=0.333;MQ=25.00;MQ0=0;NCC=1;QD=22.51;SOR=3.611
```

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

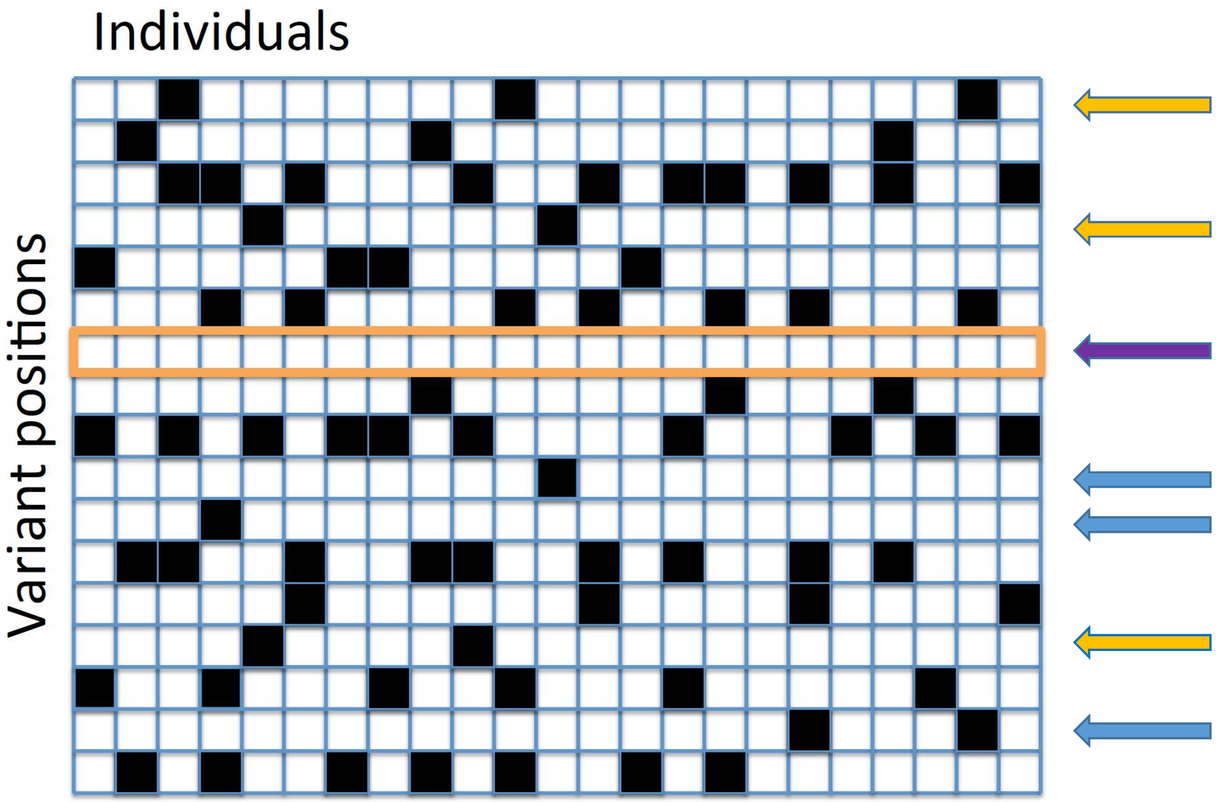
```
…
##INFO=<ID=AC,Number=A,Type=Integer,Description="Allele count in genotypes, for each ALT allele, in the same order as listed">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency, for each ALT allele, in the same order as listed">
##INFO=<ID=AN,Number=1,Type=Integer,Description="Total 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">
…
```

Two batches of individuals considered separately

□ Negatives (ref homozygotes) ■ Positives (non-ref allele present) ▨ False negatives 🟥 False positives



Two batches of individuals considered together



Sample-by-sample or joint (cohort-level) variant calling?

“Seeing” reads from multiple samples (mapped to a region of reference genome) allows smarter decisions about which alleles are real and which are sequencing or alignment errors…

More confidence in variant calling

Multiple samples data allow calling a variant even if individual sample calls are of low quality

Joint calling is better, but….

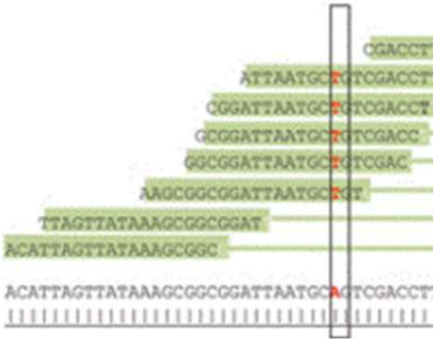
Scales badly with the number of samples

“N+1” problem: what if one more (or a few more) individuals are added?
Need to repeat the calling! (in finite time….)

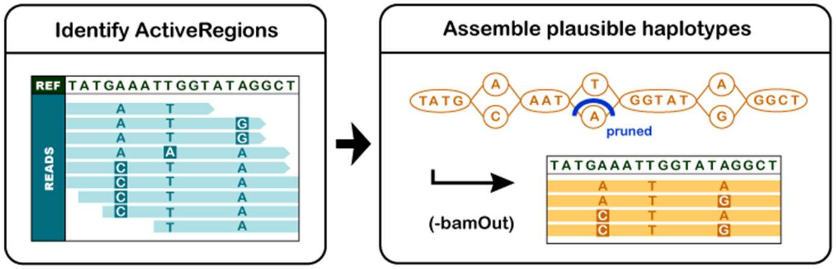
Variant detection, PL calculation

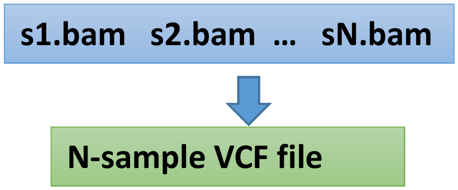
Joint calling strategy

Variants found from read pileup; PL from allelic read depths and base qualities



Variants found from locally assembled haplotypes; PL from read alignment to haplotypes





s1.bam s2.bam … sN.bam

N-sample VCF file

Software:

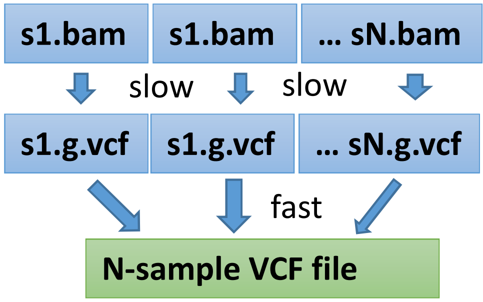
- ❑ **GATK (UnifiedGenotyper)**
- ❑ **Sentieon (commercial GATK)**
- ❑ **samtools**

Software:

- ❑ **GATK (HaplotypeCaller)**
- ❑ **Sentieon (commercial GATK)**
- ❑ **FreeBayes** (Garrison E, Marth G. arXiv:1207.3907 [q-bio.GN] 2012)

RECOMMENDED

“N+1” problem: adding one or a few more samples means re-call of everything!



s1.bam s1.bam … sN.bam

slow slow

s1.g.vcf s1.g.vcf … sN.g.vcf

fast

N-sample VCF file

Software: **none**

Software:

- ❑ **GATK (HaplotypeCaller in gVCF mode)**
- ❑ **GATK (GenotypeGVCFs)**
- ❑ **Sentieon (commercial GATK)**

RECOMMENDED

No **“N+1” problem**: process extra BAMs into **g.vcf** and repeat the (**fast**) **joint calling**

`*.g.vcf`:

- Per-sample intermediate files summarizing read depths and genotype likelihoods for each site of genome
- Derived from BAM files (in fact – replace them)
- Format somewhat similar to VCF, but with invariant regions represented as single records

```
#CHROM  POS     ID      REF        ALT         QUAL     FILTER      INFO                    FORMAT                  SAMPLE_NAME
4       2503    .       G       <NON_REF>        .        .       END=2504        GT:DP:GQ:MIN_DP:PL      0/0:2:0:2:0,0,0
4       2505    .       C       <NON_REF>        .        .       END=2517        GT:DP:GQ:MIN_DP:PL      0/0:2:6:2:0,6,90
4       2518    .       C       T,<NON_REF>      0.01     .       DP=1;ExcessHet=3.0103;MLEAC=0,0;MLEAF=0.000,0.000;RAW_MQ=441.00
GT:AD:DP:GQ:PGT:PID:PL:SB     0/0:1,0,0:1:3:0|1:2518_C_T:0,3,26,3,26,26:0,1,0,0
4       2519    .       A       <NON_REF>        .        .       END=2531        GT:DP:GQ:MIN_DP:PL      0/0:2:6:2:0,6,90
4       2532    .       G       C,<NON_REF>      20.00    .       DP=1;ExcessHet=3.0103;MLEAC=1,0;MLEAF=0.500,0.000;RAW_MQ=441.00
GT:AD:DP:GQ:PGT:PID:PL:SB     1/1:0,1,0:1:3:0|1:2518_C_T:45,3,0,45,3,45:0,0,0,1
4       2533    .       A       <NON_REF>        .        .       END=2543        GT:DP:GQ:MIN_DP:PL      0/0:2:6:2:0,6,90
```

- Much smaller than BAM files
- Used later to make **joint calls** on the cohort (instead of the original BAM files)

This approach addresses the **“N+1” problem**: process one extra BAM into **g.vcf** and repeat the (fast) joint calling

SNP and Indel calling is a large-scale Bayesian modeling problem

=1

Reported as PL in our VCF example

Bayesian model

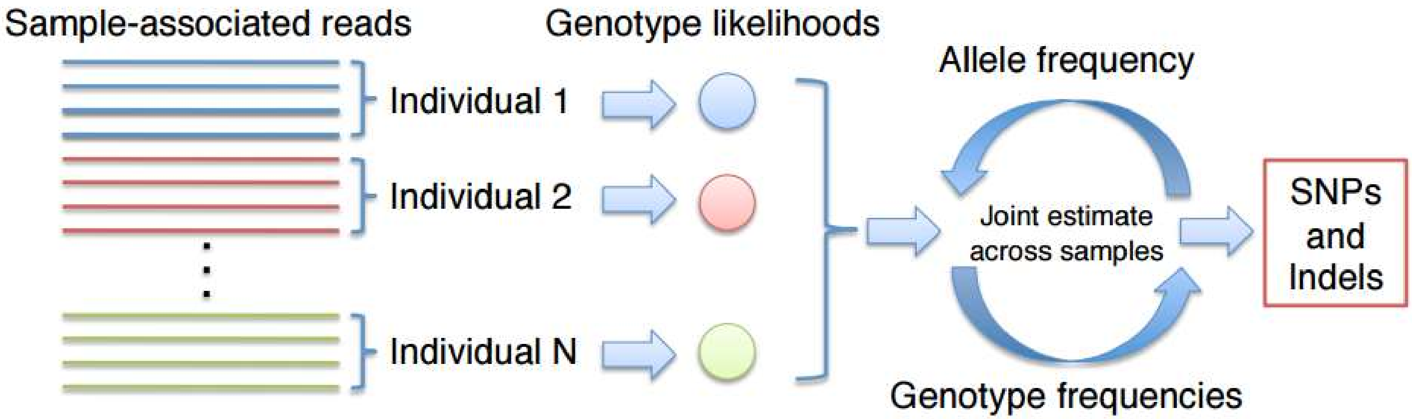
$$\Pr\{G|D\} = \frac{\overbrace{\Pr\{G\}}^{\text{Prior of the genotype}}\,\overbrace{\Pr\{D|G\}}^{\text{Likelihood of the genotype}}}{\Sigma_i \Pr\{G_i\}\Pr\{D|G_i\}}, \quad \text{[Bayes' rule]}$$

$$\Pr\{D|G\} = \prod_j \left(\frac{\Pr\{D_j|H_1\}}{2} + \frac{\Pr\{D_j|H_2\}}{2}\right) \quad \text{where } \overbrace{G = H_1 H_2}^{\text{Diploid assumption}}$$

$$\Pr\{D|H\} \quad \text{is} \quad \text{the haploid likelihood function}$$

- Inference: what is the genotype G of each sample given read data D for each sample?
- Calculate via Bayes' rule the probability of each possible G
- Product expansion assumes reads are independent
- Relies on a likelihood function to estimate probability of sample data given proposed haplotype

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



Pr{D|G}

For each site, obtain distribution of count of non-reference allele:

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

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)

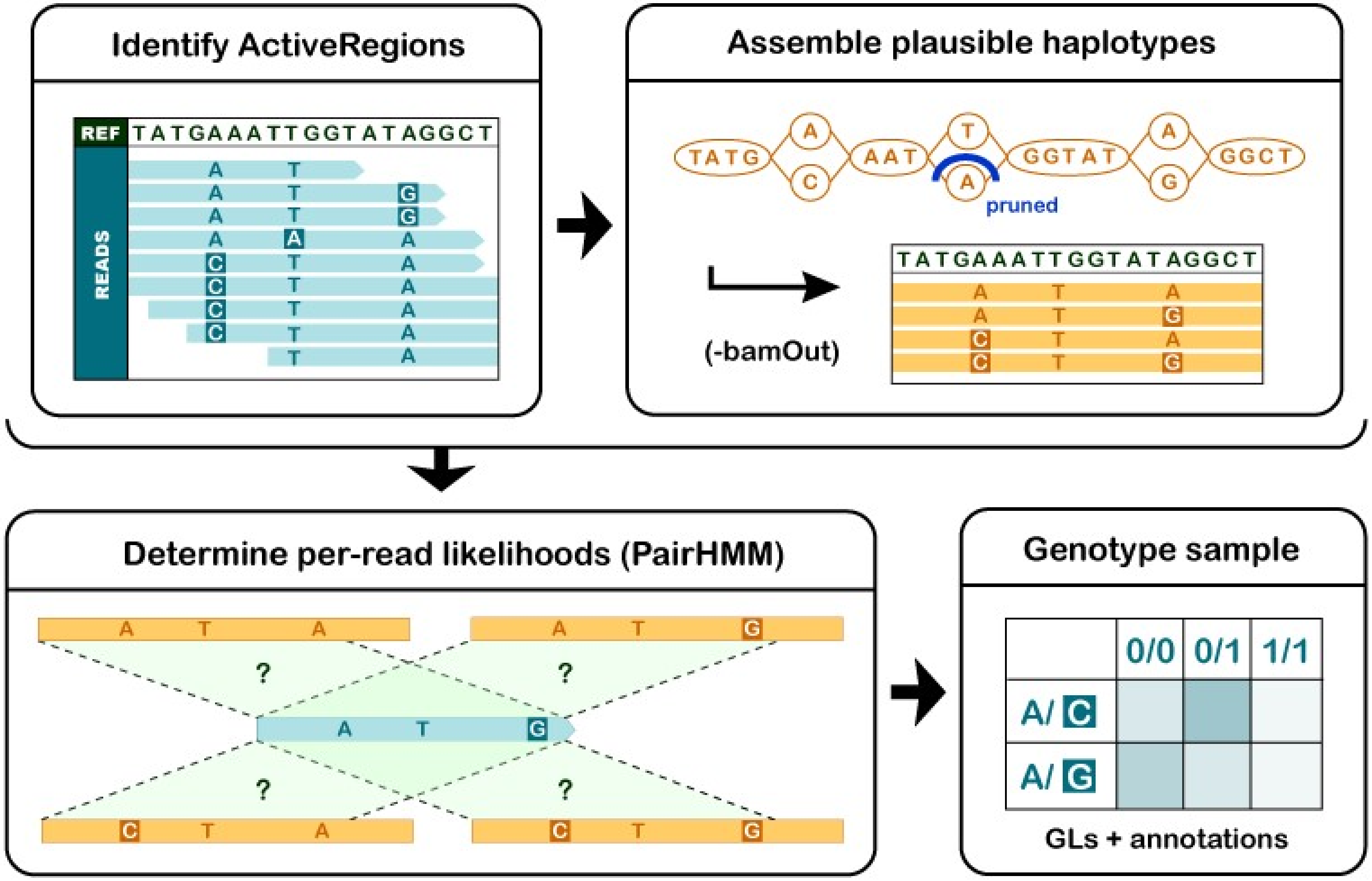
Haplotype likelihood function in UnifiedGenotyper

$$\begin{aligned}\Pr\{D_j|H\} &= \Pr\{D_j|b\},\ \text{[single base pileup]}\\ \Pr\{D_j|b\} &= \begin{cases} 1-\epsilon_j & D_j = b,\\ \epsilon_j & \text{otherwise.}\end{cases}\end{aligned}$$

From base quality score

Substitution-specifc rates (confusion matrix) may also be used here

Haplotype likelihood function in HaplotypeCaller



Identify ActiveRegions

Assemble plausible haplotypes

Determine per-read likelihoods (PairHMM)

Genotype sample

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

Haplotype caller: what does it do?

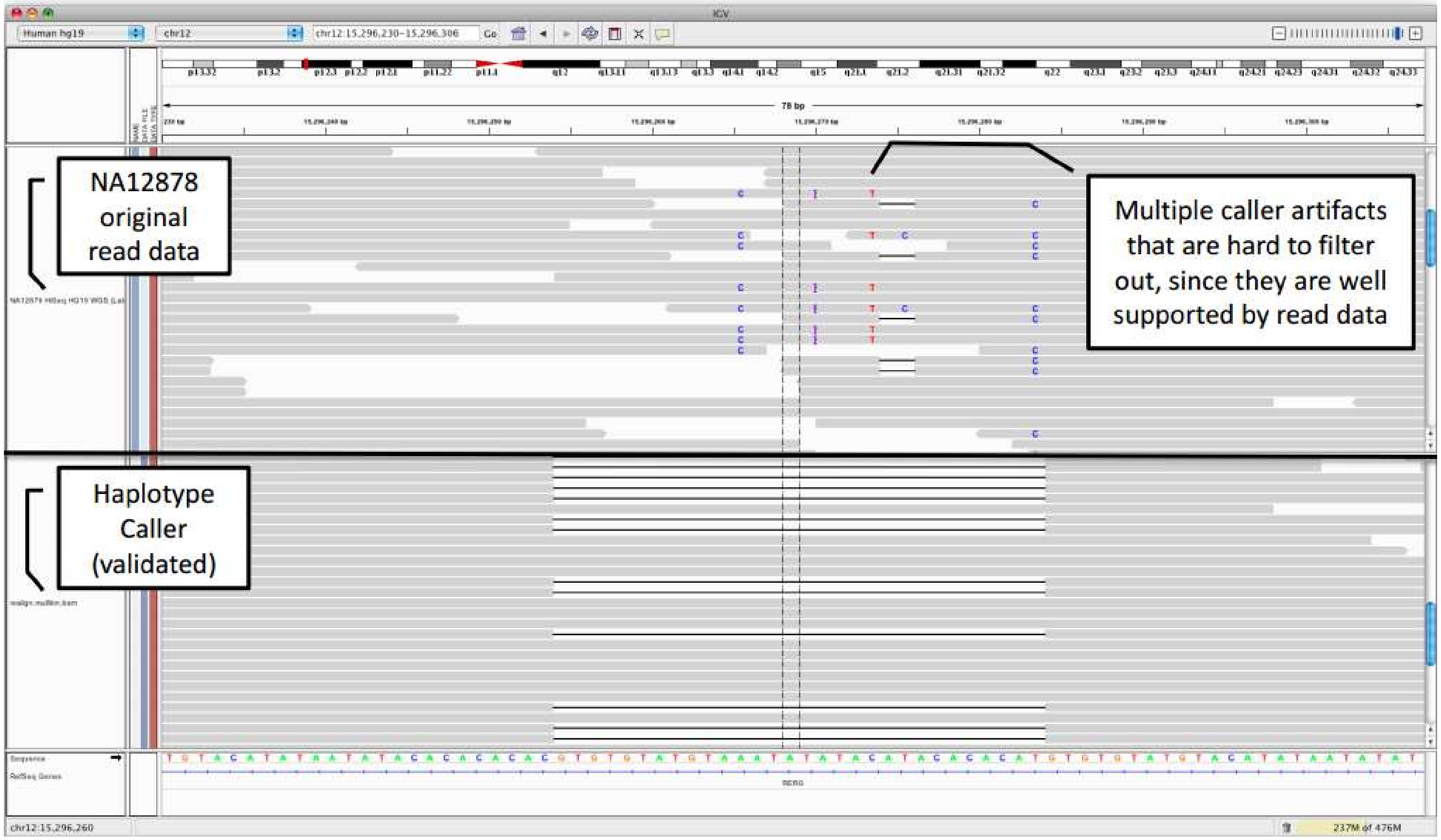
1. **Define active regions**
The program determines which regions of the genome it needs to operate on, based on the presence of significant evidence for variation.

2. **Determine haplotypes by re-assembly of the active region**
For each ActiveRegion, the program builds a De Bruijn-like graph to reassemble the ActiveRegion, and identifies what are the possible haplotypes present in the data. The program then realigns each haplotype against the reference haplotype using the Smith-Waterman algorithm in order to identify potentially variant sites.

3. **Determine likelihoods of the haplotypes given the read data P{Dj|H}**
For each ActiveRegion, the program performs a pairwise alignment of each read against each haplotype using the PairHMM algorithm. This produces a matrix of likelihoods of haplotypes given the read data. These likelihoods are then marginalized to obtain the likelihoods of alleles for each potentially variant site given the read data.

4. **Assign sample genotypes**
For each potentially variant site, the program applies Bayes' rule, using the likelihoods of alleles given the read data to calculate the likelihoods of each genotype per sample given the read data observed for that sample. The most likely genotype is then assigned to the sample.

Artifact SNPs and small indels caused by large indel is only recovered by local assembly

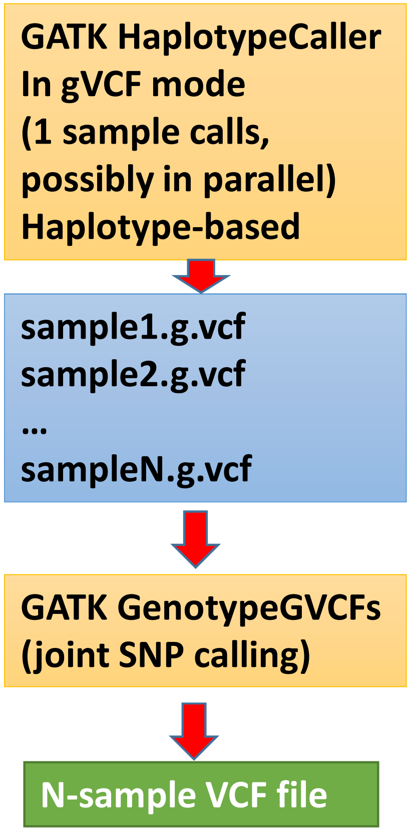

IGV
Human hg19
chr12
chr12:15,296,230-15,296,306
Go
78 bp
15,296,240 bp
15,296,250 bp
15,296,260 bp
15,296,270 bp
15,296,280 bp
15,296,290 bp
15,296,300 bp
Sequence
RefSeq Genes
T G T A C A T A T A A T A T A C A C A C A C A C G T G T G T A T G T A A A T A T A T A C A T A C A C A C A T G T G T G T A T G T A C A T A T A A T A T A T
chr12:15,296,260
237M of 476M

NA12878 original read data

Haplotype Caller (validated)

Multiple caller artifacts that are hard to filter out, since they are well supported by read data

HaplotypeCaller in gVCF mode….



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

```
java -jar GenomeAnalysisTK.jar \
     -T HaplotypeCaller \
     -R genome.fa \
     -I sample1.sorted.dedup.realigned.fixmate.recal.bam  \
     --emitRefConfidence GVCF \
     -o sample1.g.vcf
```

Slow

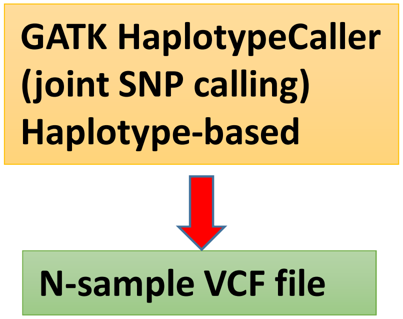
….Followed by joint variant calling with GenotypeGVCFs

Run **once** after all *.g.vcf files are obtained

```
java -Xmx2g -jar GenomeAnalysisTK.jar \
   -T GenotypeGVCFs \
   -R genome.fa \
   --variant sample1.g.vcf \
   --variant sample2.g.vcf \
   --variant sample3.g.vcf \
   --variant sample4.g.vcf \
   -stand_call_conf 30 \
   -o 4samples.vcf
```

Fast

Running HaplotypeCaller (variant-only mode)



```
java -jar GenomeAnalysisTK.jar \
     -T HaplotypeCaller \
     -R genome.fa \
     -I sample1.sorted.dedup.realigned.fixmate.recal.bam  \
     -I sample2.sorted.dedup.realigned.fixmate.recal.bam  \
     -I sample3.sorted.dedup.realigned.fixmate.recal.bam  \
     -I sample4.sorted.dedup.realigned.fixmate.recal.bam  \
     -L chr2R \
     -stand_call_conf 30 \
     -o 4samples_joint_call.chr2R.vcf
```

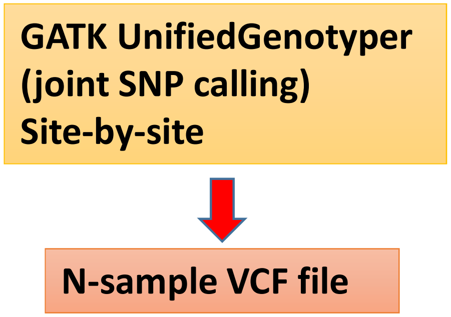
May be parallelized by genome region (using –L option)
i.e., different regions run on different processors

Note:

Haplotype assembly uses reads from all samples (rather than one at a time), and so…

…resulting VCF file will not be exactly equivalent to that obtained from gVCF mode runs followed by GenotypeGVCFs

Running UnifiedGenotyper



GATK UnifiedGenotyper
(joint SNP calling)
Site-by-site

N-sample VCF file

```
java -Djava.io.tmpdir=$TMP -jar GenomeAnalysisTK.jar \
     -T UnifiedGenotyper \
     -R genome.fa \
     -I sample1.sorted.dedup.realigned.fixmate.recal.bam  \
     -I sample2.sorted.dedup.realigned.fixmate.recal.bam  \
     -I sample3.sorted.dedup.realigned.fixmate.recal.bam  \
     -I sample4.sorted.dedup.realigned.fixmate.recal.bam  \
     -L chr2R \
     -stand_call_conf 30 \
     -o 4samples.UG.chr2R.vcf
```

May be parallelized by genome region (using –L option)
i.e., different regions run on different processors

Broad recommends running HaplotypeCaller-based pipeline instead if this

However, still recommended for
Pooled sample cases
High ploidy cases

Options to pay attention to in HaplotypeCaller, GenotypeGVCFs, and UnifiedGenotyper

`-stand_call_conf [number]`

Variants with quality score (QUAL) less than **`[number]`** will not be written to VCF file. **Good to set this low** – better have too many raw variants than too few. Can always filter VCF file later. Default: 30.

`--num_cpu_threads_per_data_thread [number]`

Run on **`[number]`** threads (CPU cores). Default: 1 The program scales reasonably up to 8-10 threads.

`-dcov [int]`

Read depth at any locus will be capped at **`[number]`**; the goal is to provide even distribution of read start positions while removing excess coverage. For truly unbiased down-sampling, use **`-dfrac`** Defaults are usually high (250) – can be reduced to speed things up.

Alternatives to GATK

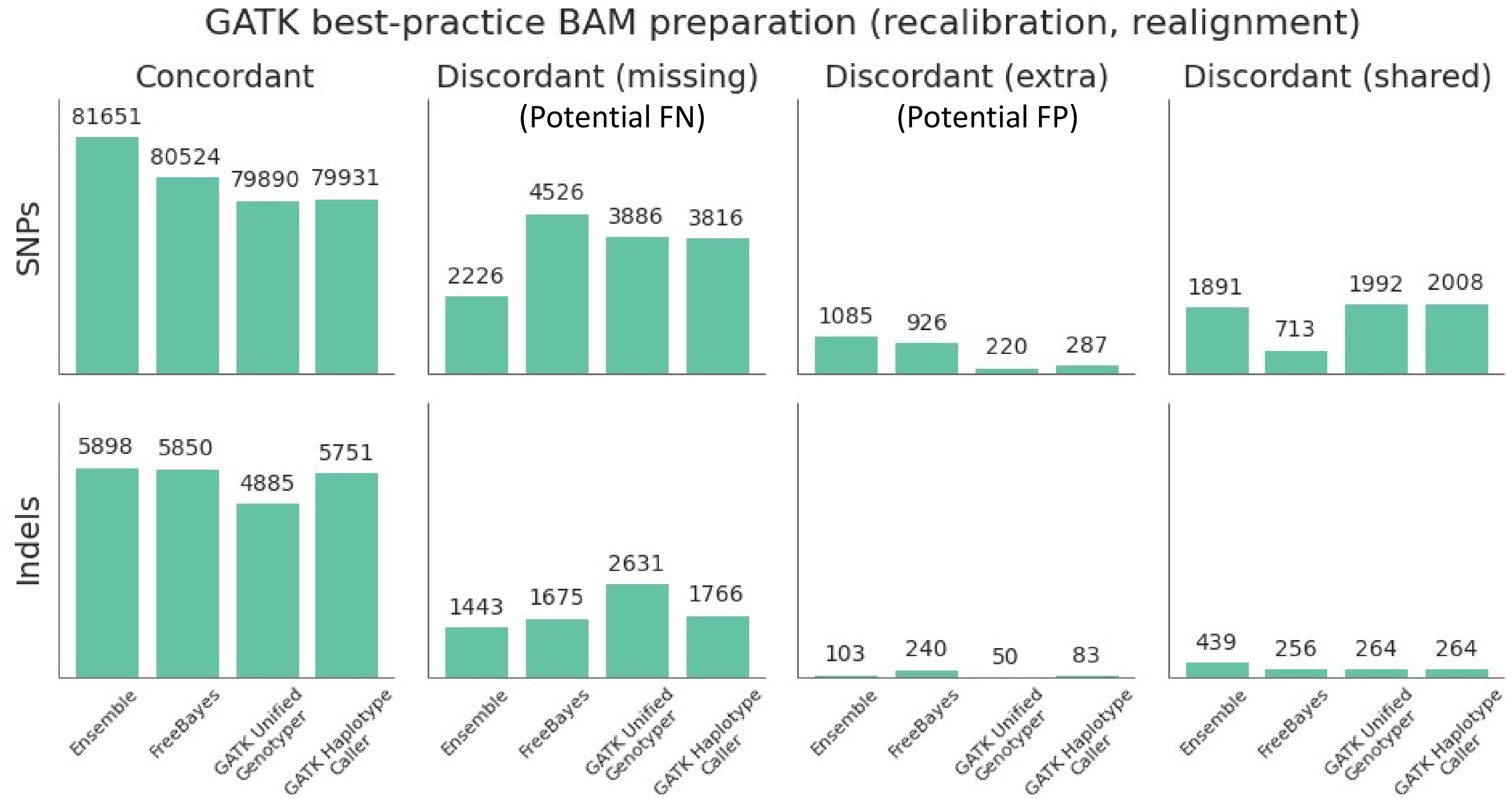
FreeBayes (Erik Garrison et al., https://github.com/ekg/freebayes)

- 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 – order of magnitude faster than GATK HaplotypeCaller**!
- Still suffers from "N+1" problem

Sentieon (http://sentieon.com)

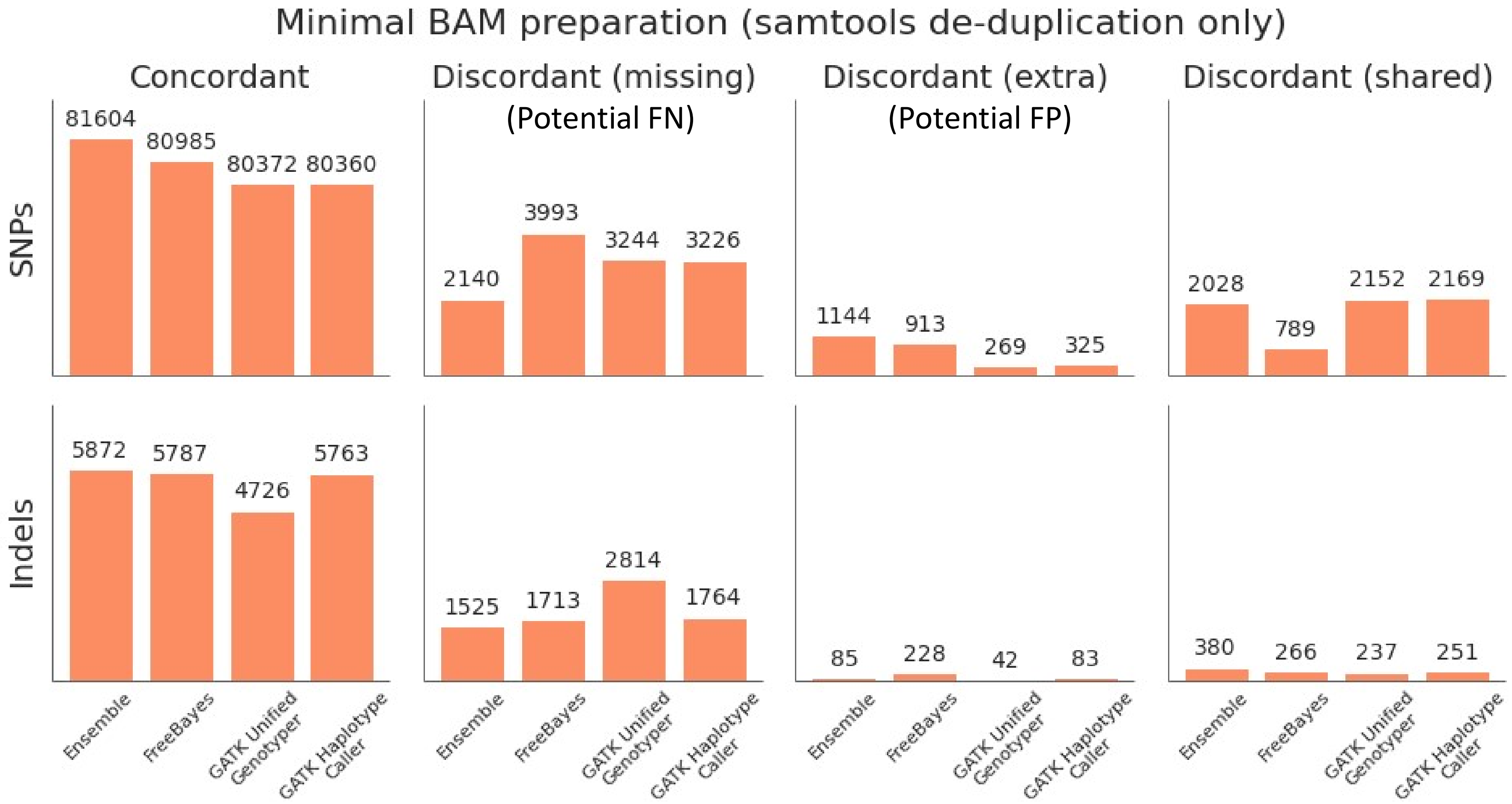
- Commercial version of GATK (currently equivalent to GATK 3.5)
- **10-30 times faster than GATK on most parts of the pipeline**
- Command syntax different from GATK (although functionality the same)
- Available on BioHPC Lab for $50/week (need to recover license cost)
    - License: 300 CPU cores of can run simultaneously (across all machines) at any time

Comparison of GATK and FreeBayes (how many known NA12878 SNPs/indels are called correctly/incorrectly)



Source: http://bcb.io/2013/10/21/updated-comparison-of-variant-detection-methods-ensemble-freebayes-and-minimal-bam-preparation-pipelines/

Comparison of GATK and FreeBayes (how many known NA12878 SNPs/indels are called correctly/incorrectly)



Source: http://bcb.io/2013/10/21/updated-comparison-of-variant-detection-methods-ensemble-freebayes-and-minimal-bam-preparation-pipelines/

- FreeBayes not much different than GATK HaplotypeCaller
  - somewhat better in detecting concordant SNPs and indels
  - somewhat worse with False Positives and False Negatives

- Expensive BAM file pre-processing (re-alignment around indels and base quality score recalibration) seems to have little impact, especially for haplotype-based callers (FreeBayes and HaplotypeCaller)

- However, Broad still recommends running base quality score recalibration

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

Simple linux tools help analyze a VCF file

Count variants:

```
grep –v ”#” hc.chr2R.vcf | wc –l
```

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

```
head -1000 hc.chr2R.vcf | grep “#” > new_file.vcf
grep –v ”#” hc.chr2R.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 “#” hc.chr2R.vcf | awk ’{if($6>100) print}’ > good_variants
```

Useful tool: VariantFiltration – hard filtering on various criteria

Example:

```
java -jar GenomeAnalysisTK.jar  \
-T VariantFiltration \
-R genome.fa \
-filter "MQ0 >= 4 && ((MQ0 / (1.0 * DP)) > 0.1)"  \
-filter “FS>=10.0"   \
-filter “AN>=4" \
-filter "DP>100 || DP<4" \
-filterName HARD_TO_VALIDATE  \
-filterName SNPSBFilter \
-filterName SNPNalleleFilter  \
-filterName SNPDPFilter  \
-cluster 3 \
-window 10  \
--variant chr2R.4samples.vcf \
-o chr2R.4samples.filtered.vcf
```

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

Whenever any of the “-filter” conditions satisfied, the corresponding “-filterName” 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)

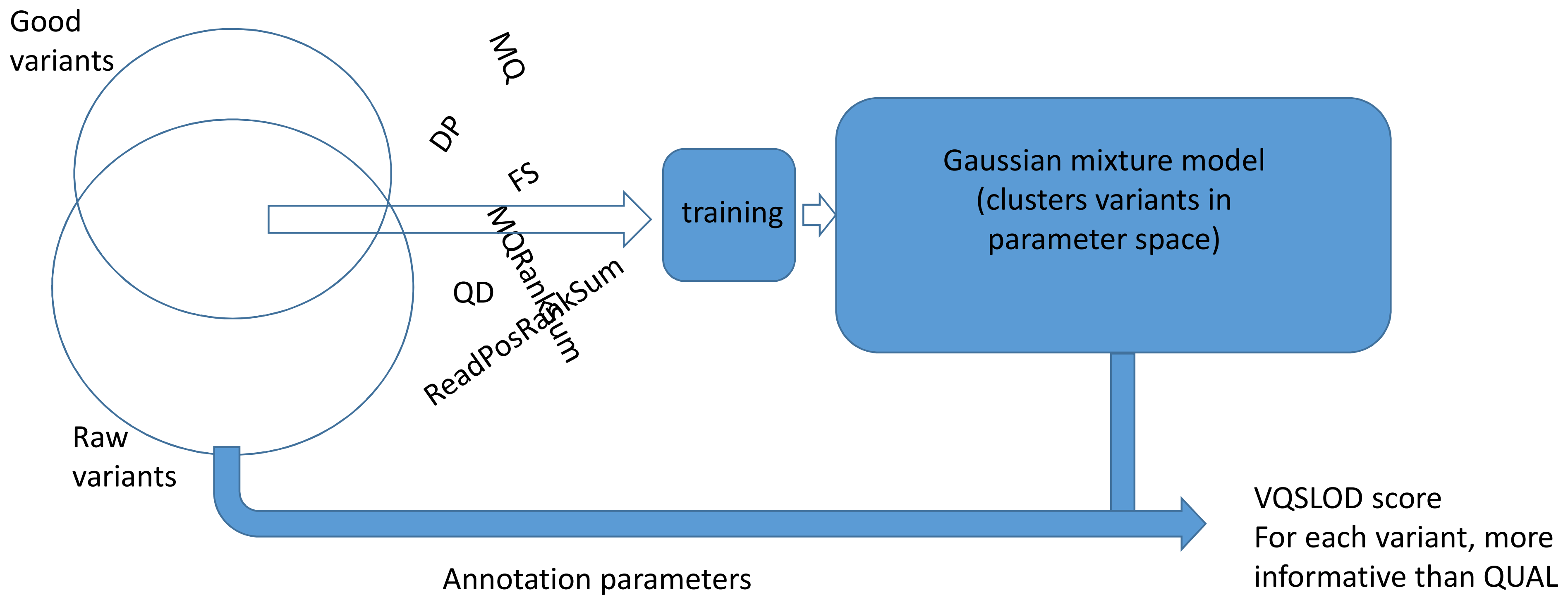
QD
QUAL/(depth of non-reference reads) – should be large (e.g, >2)

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)

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



Good variants

Raw variants

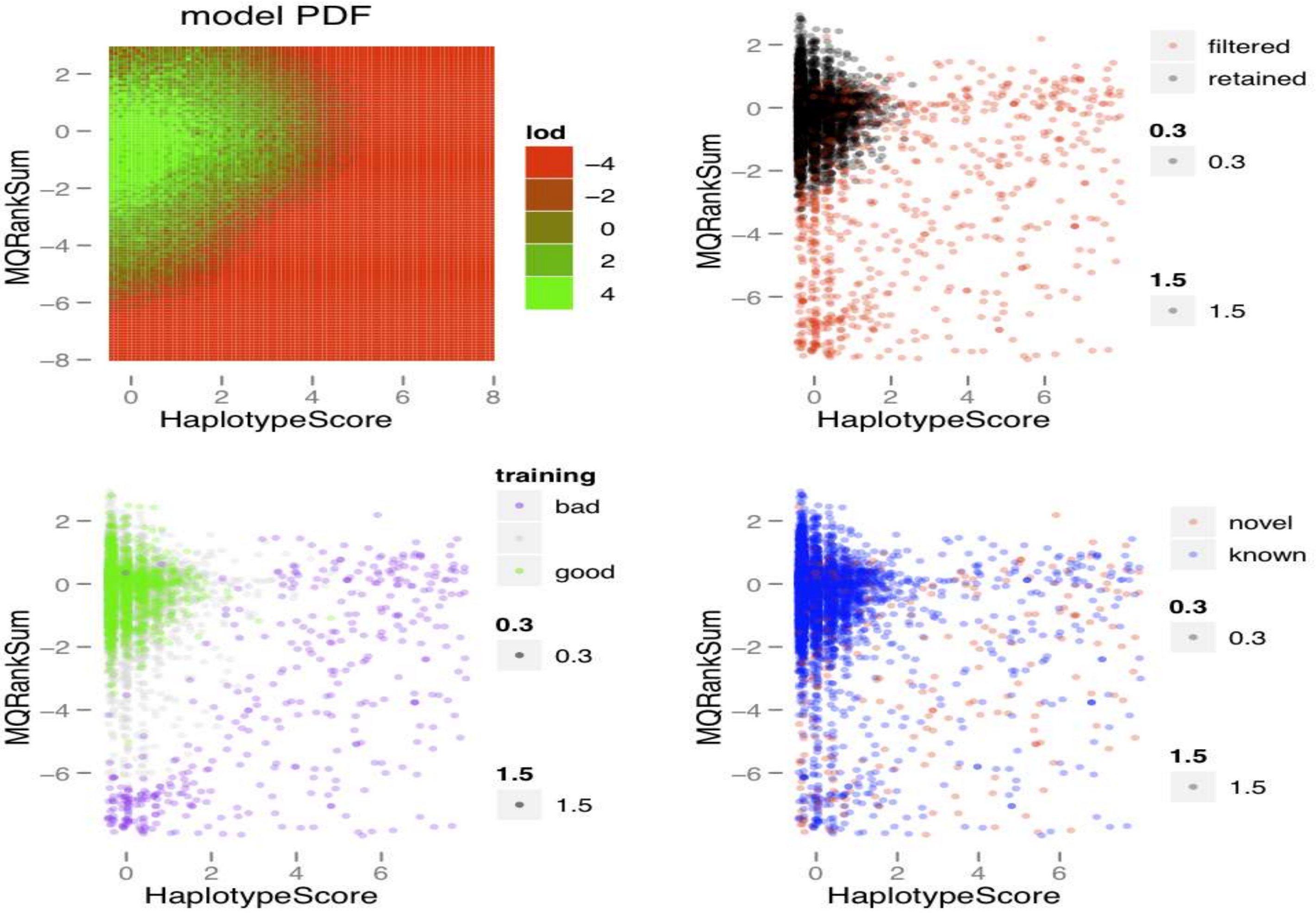
training

Gaussian mixture model (clusters variants in parameter space)

Annotation parameters

VQSLOD score
For each variant, more informative than QUAL

2D cross-section though cluster of variants in multi-D parameters space



Useful tool: VariantEval – summary stats and comparison of callsets

```
java -Xmx2g -jar GenomeAnalysisTK.jar \
   -R genome.fa \
   -T VariantEval \
   -o file1.file2.comp.gatkreport \
   --eval:set1 file1.vcf  \
   --comp file2.vcf
```

Will summarize various properties of variants in **file1.vcf**

- Classes of variants
- Indel characteristics
- Ti/Tv
- Multi-allelic variants
- ….

Will compare to variants in file2.vcf

- Common variants and extra variants in file1.vcf (**compared to file2.vcf**)
- Concordance rate

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