Variant calling: Part 1

Robert Bukowski, Qi Sun, Minghui Wang

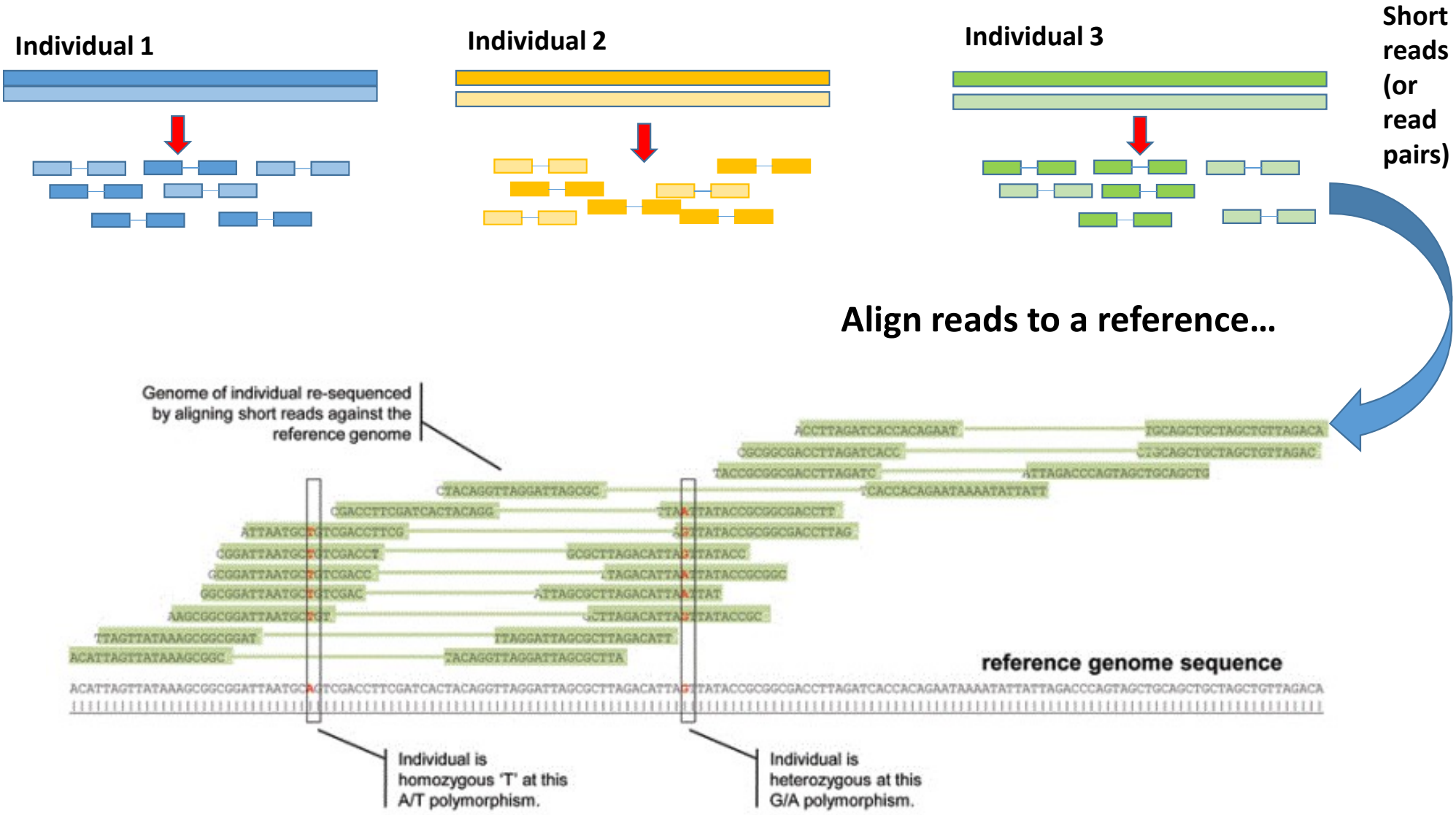
Bioinformatics Facility

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


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

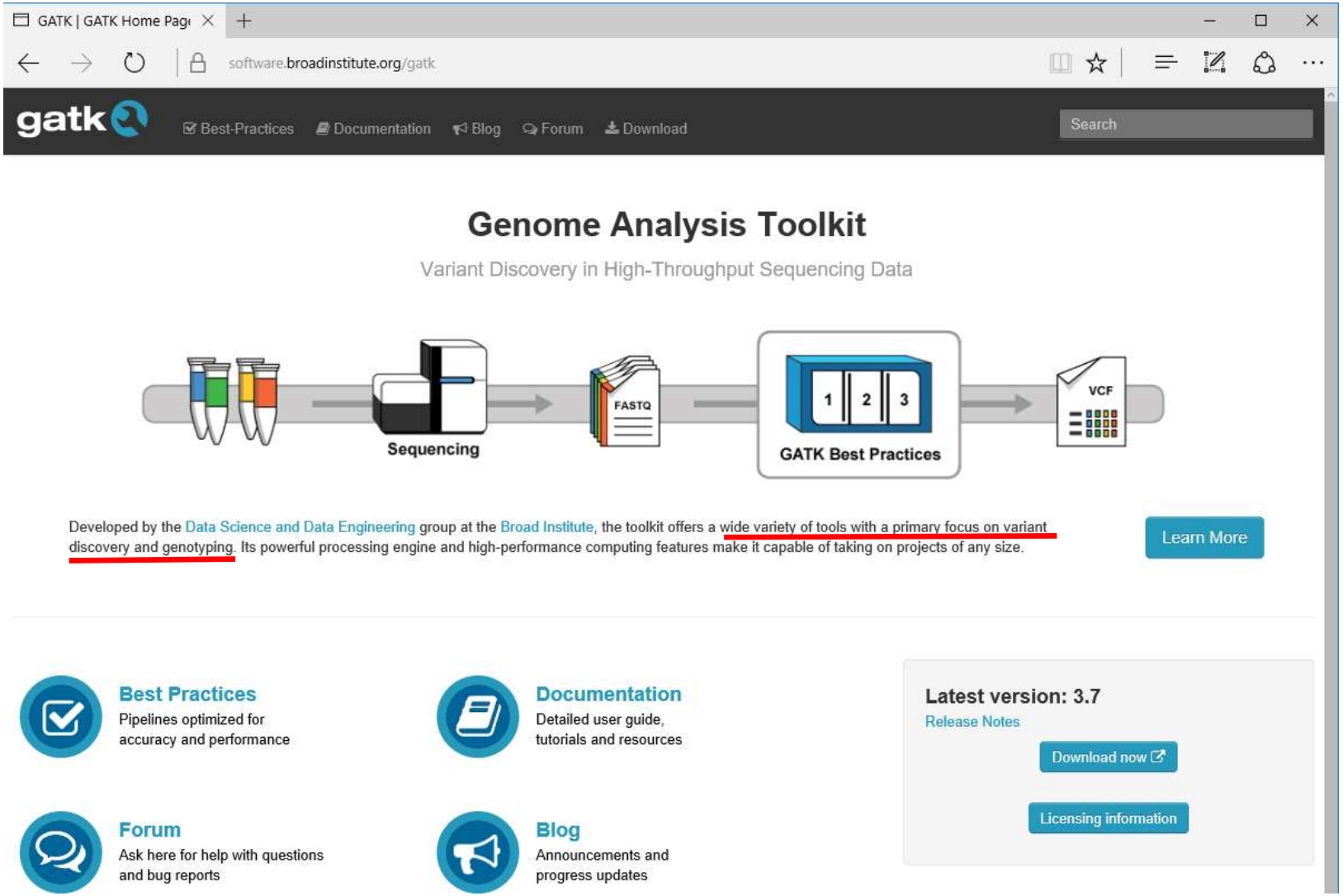
Variant site chr and position	Indiv1	Indiv2	Indiv3	….
site1	AA	AA	AC	…
site2	GT	missing	TT	…
…	…	…	…	…
siteN	CC	CC	AA	…

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



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 **Best practices for calling variants** with the GATK

Where to go for detailed documentation of GATK and PICARD tools

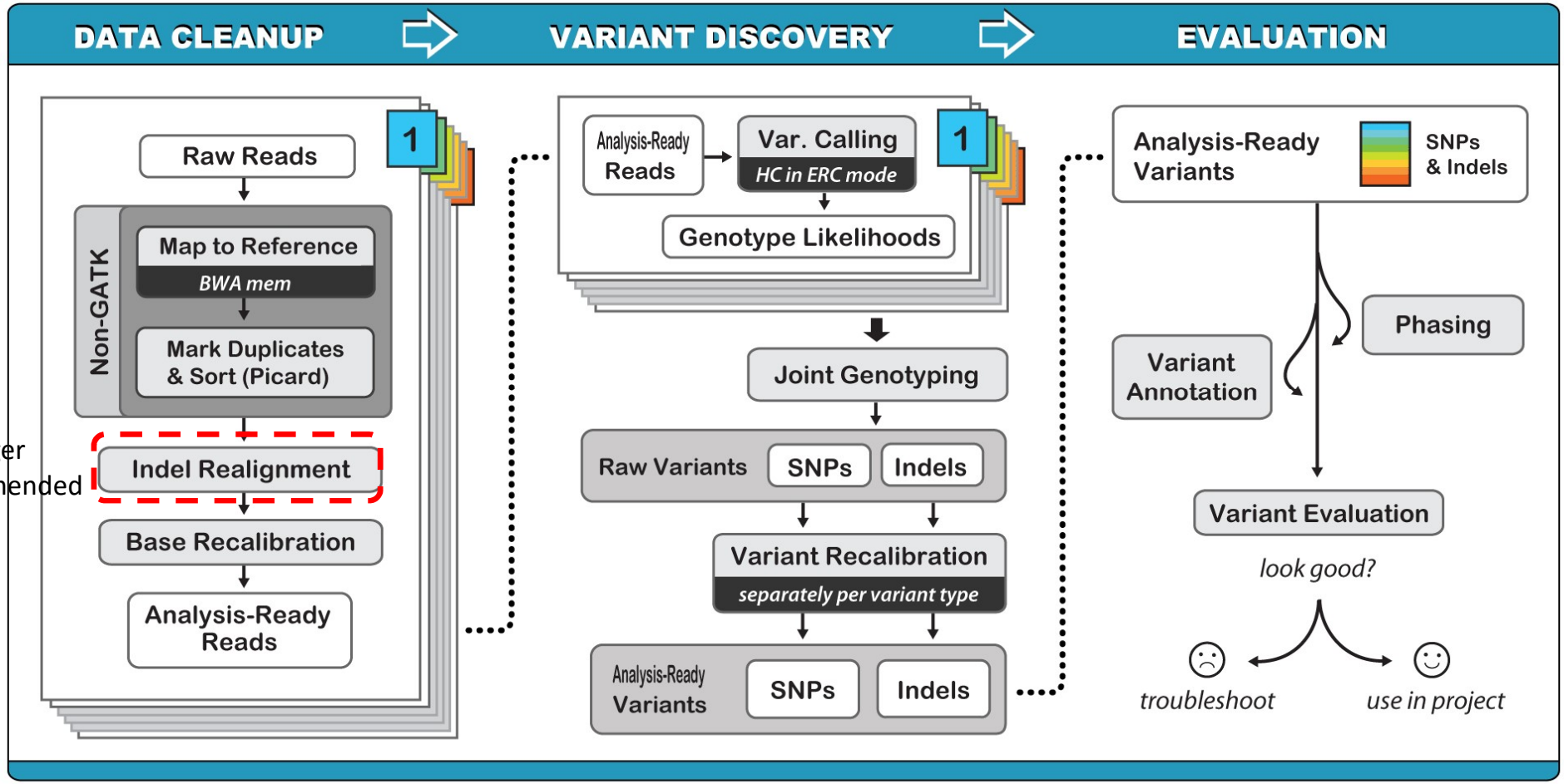
GATK

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

PICARD

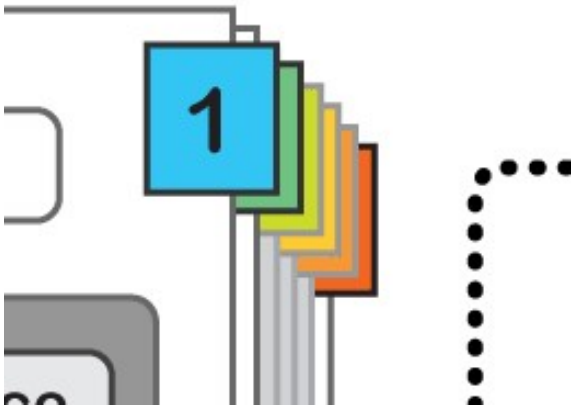
http://broadinstitute.github.io/picard/

Best Practices for DNA-Seq variant calling



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**

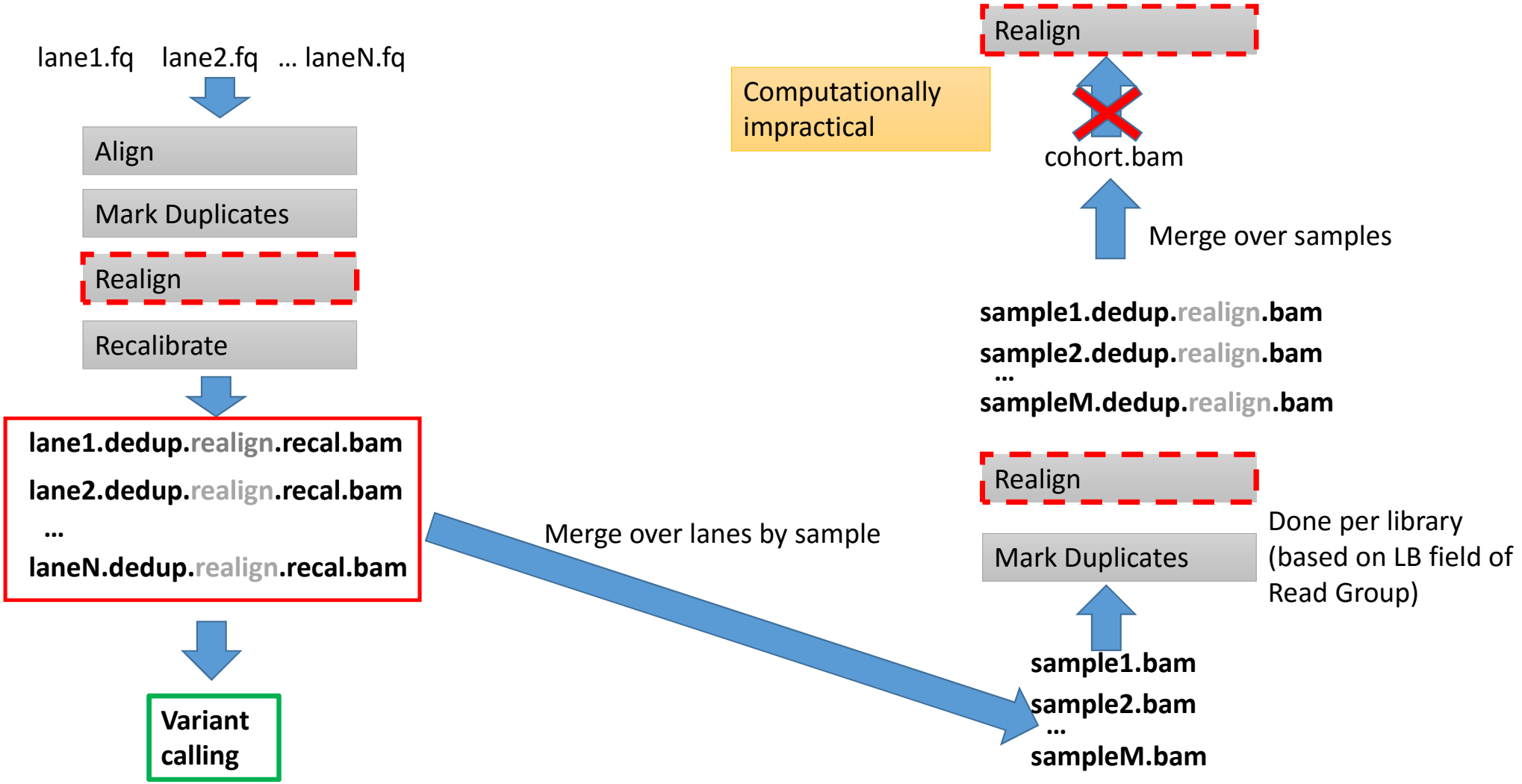
- A lane may contain a single sample, OR…
- A lane may contain reads from multiple samples (multiplexing)

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

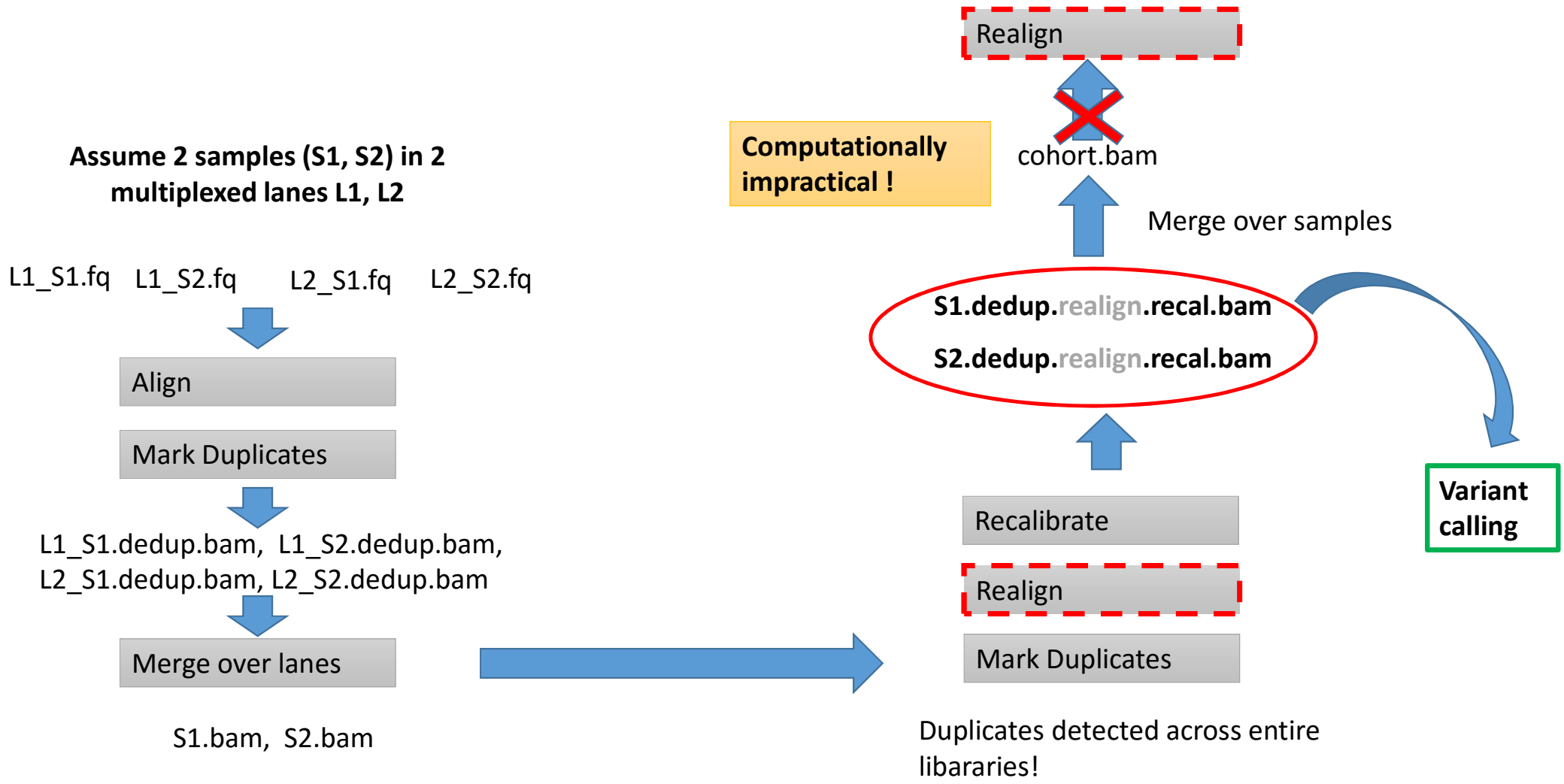
Generally, read pre-processing is done separately for each FASTQ file or pair, especially if files contain a lot of data. However:
- **Mark Duplicates** works best if given **all reads from a given library** (sometimes scattered among files)
- **Indel realignment** works best with all reads from all 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



Typical read preparation pipeline: multiplexed lanes



Input: reads in FASTQ format

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

```
@61DFRAAXX100204:1:100:10494:3070
ACTGCATCCTGGAAAGAATCAATGGTGGCCGGAAAGTGTTTTTCAAATACAAGAGTGACAATGTGCCCTGTTGTTT
+
ACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBC?CCCCCCCCC@@CACCCCCACCCCCCCCCCCCCCCCCCCCCC
```

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 $\mathbf{10^{-3.4}}$.

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?CCCCCCCCC@@CACCCCCACCCCCCCCCCCCCCCCCCCCCC
```

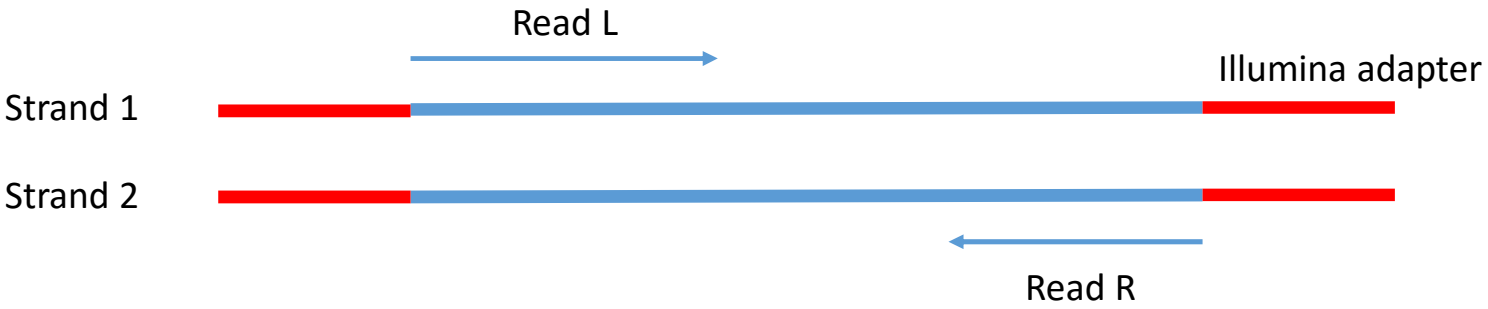
First sequence in "right" file

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

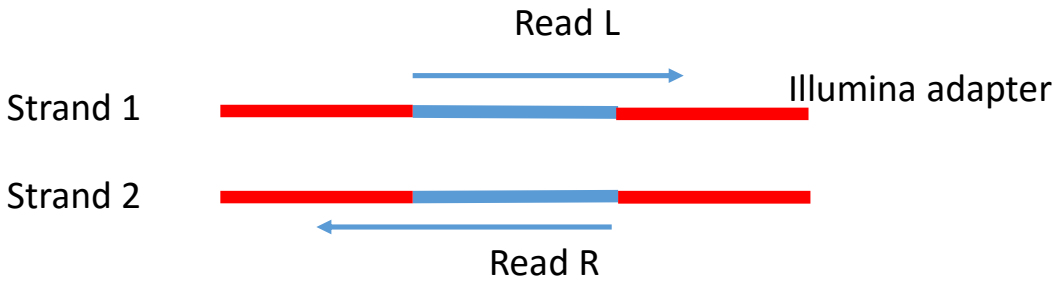
The two ends come from **opposite strands** of the fragment being sequenced

End 1 End 2

Sequencing long fragment



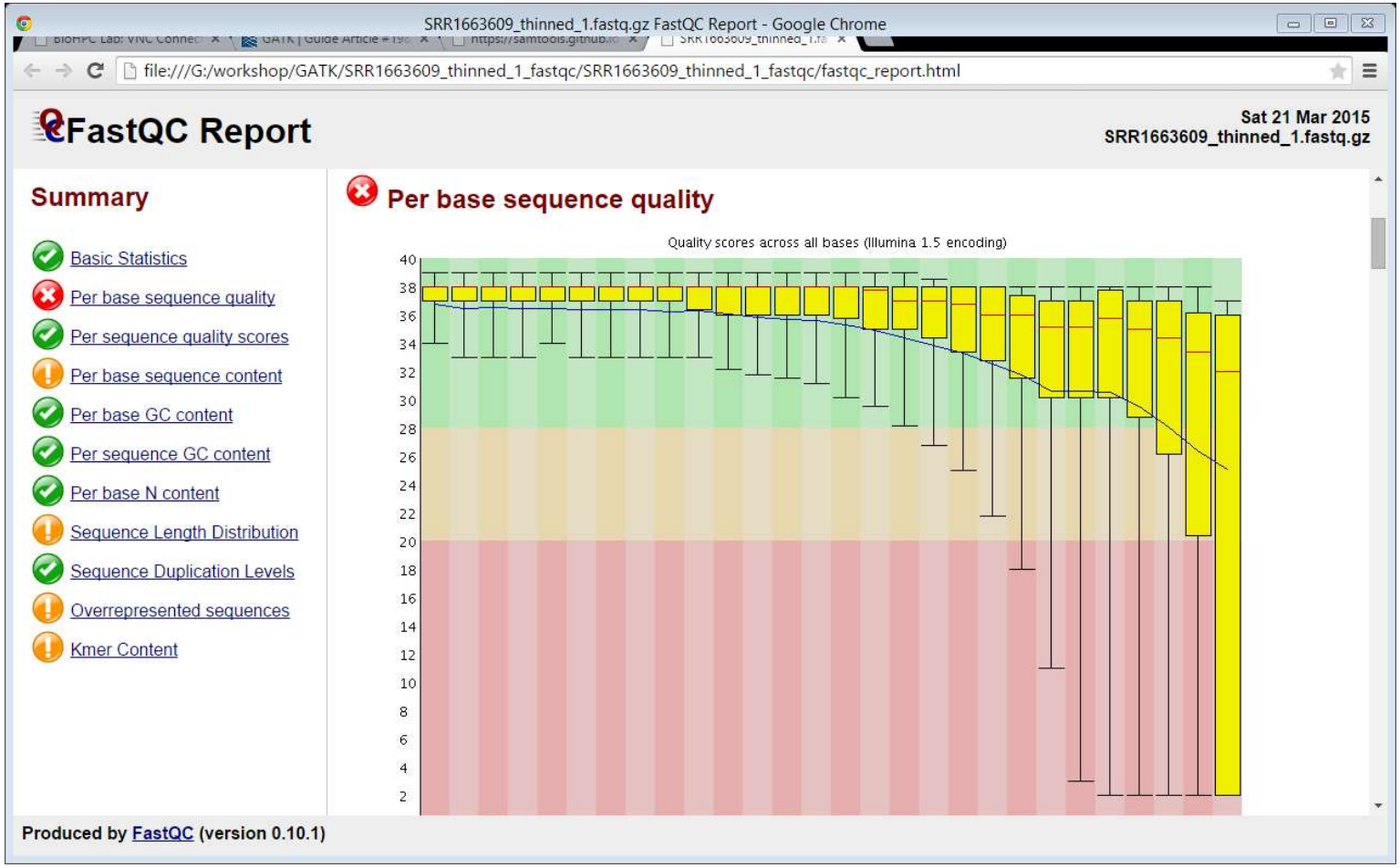
Sequencing short fragment



Read-through: sequenced reads cut into adapters

Adapter remnants should be removed

Read quality assessment with fastqc



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

Pre-alignment read clean-up

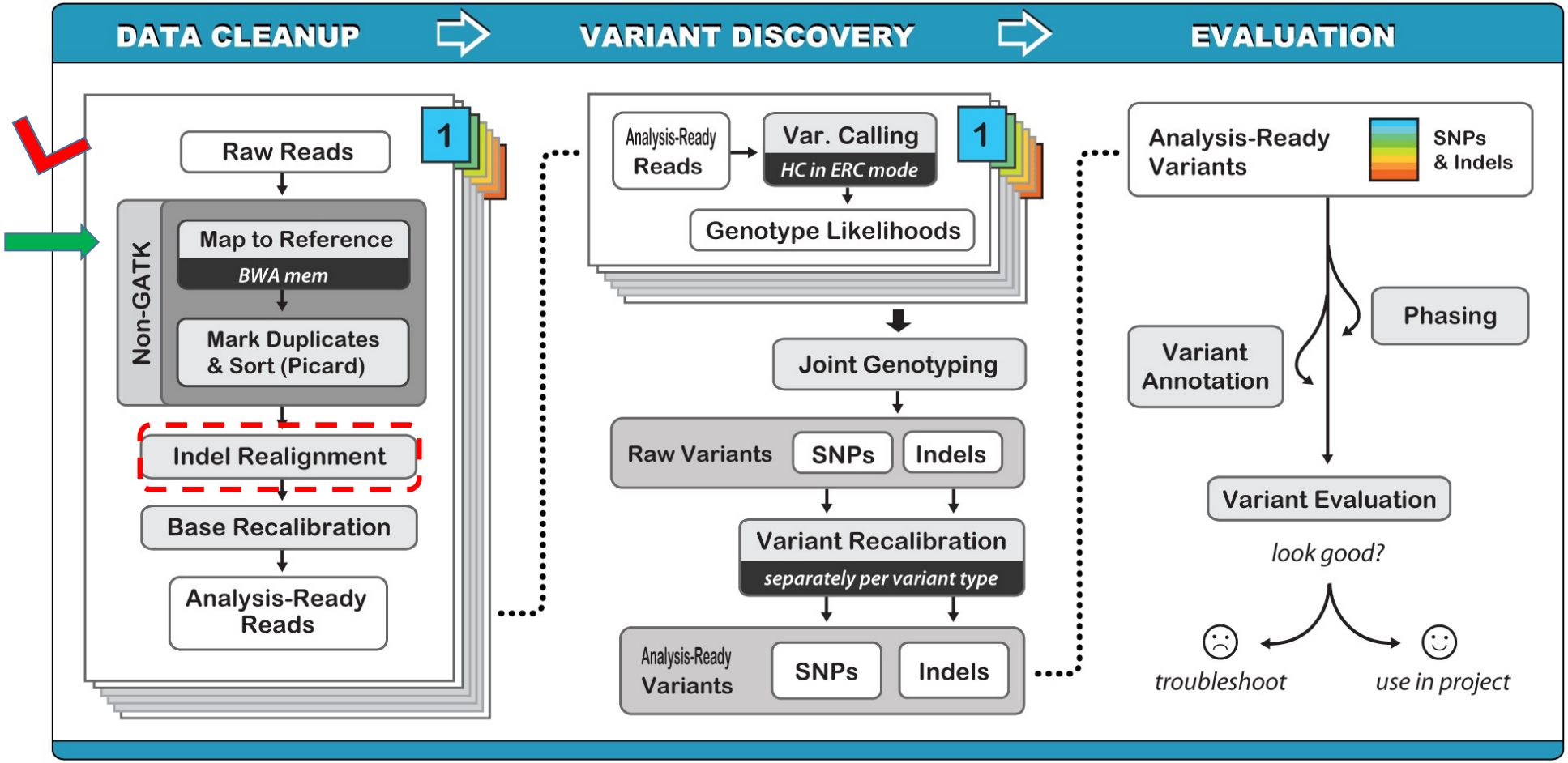
Trimmomatic: A flexible read trimming tool for Illumina NGS data (Bolger et al., http://www.usadellab.org/cms/?page=trimmomatic)

```
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**

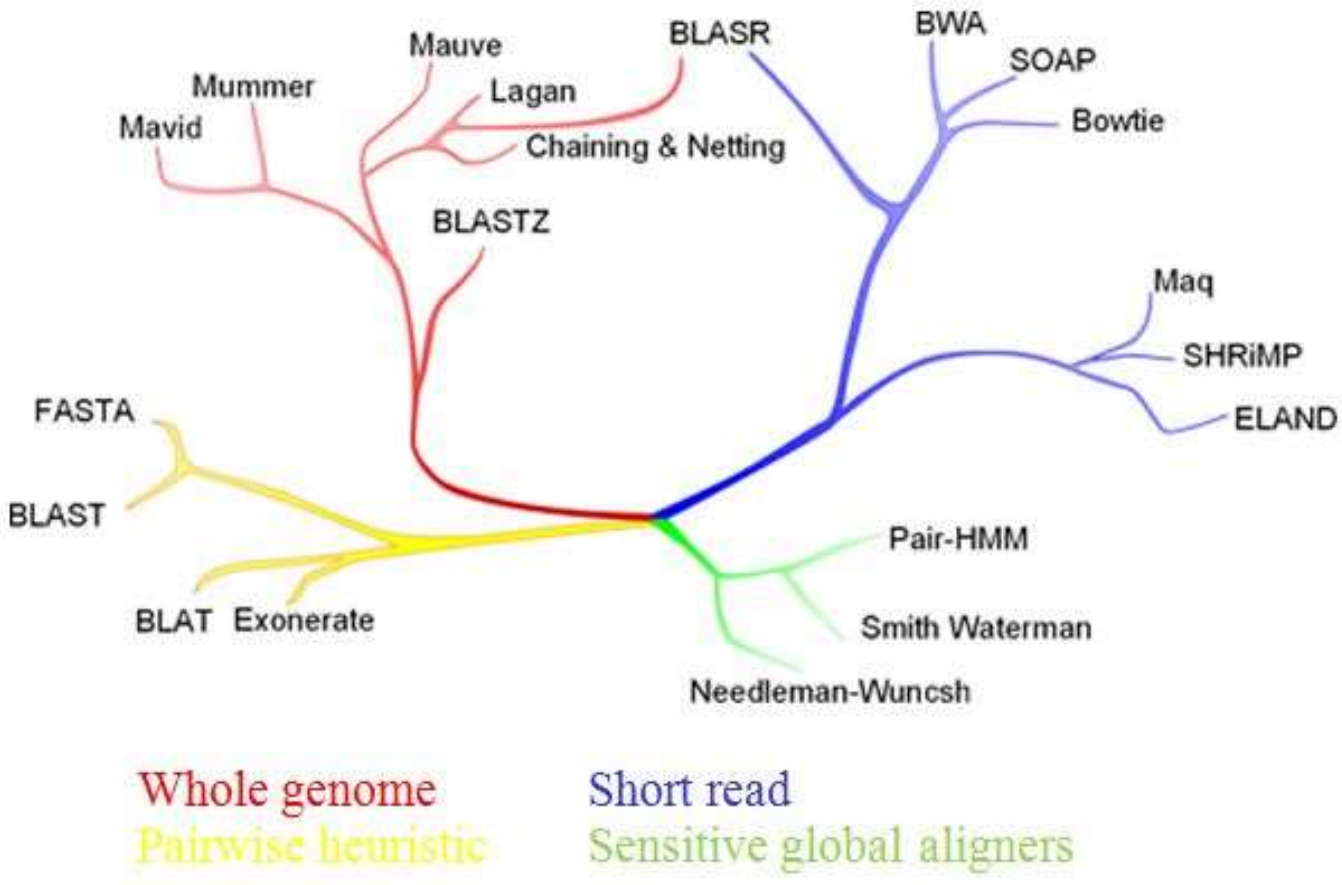
- Some reads cannot be mapped unambiguously in a single location (have low **Mapping Quality**)
  - if reads too short
  - reads originating from paralogs or repetitive regions
  - Having paired-end (PE) data helps

- 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

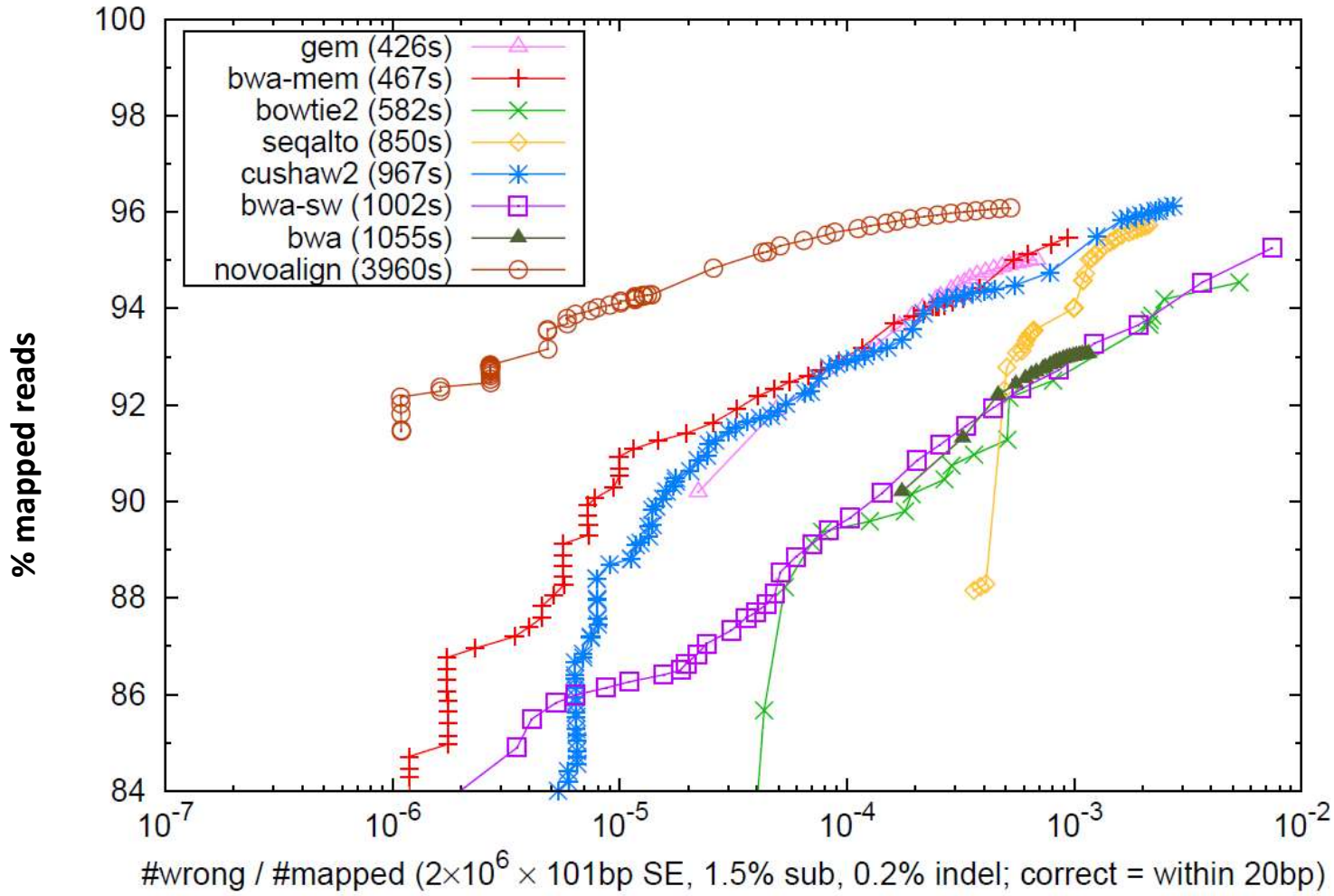
Picking good aligner is important

Aligner phylogeny



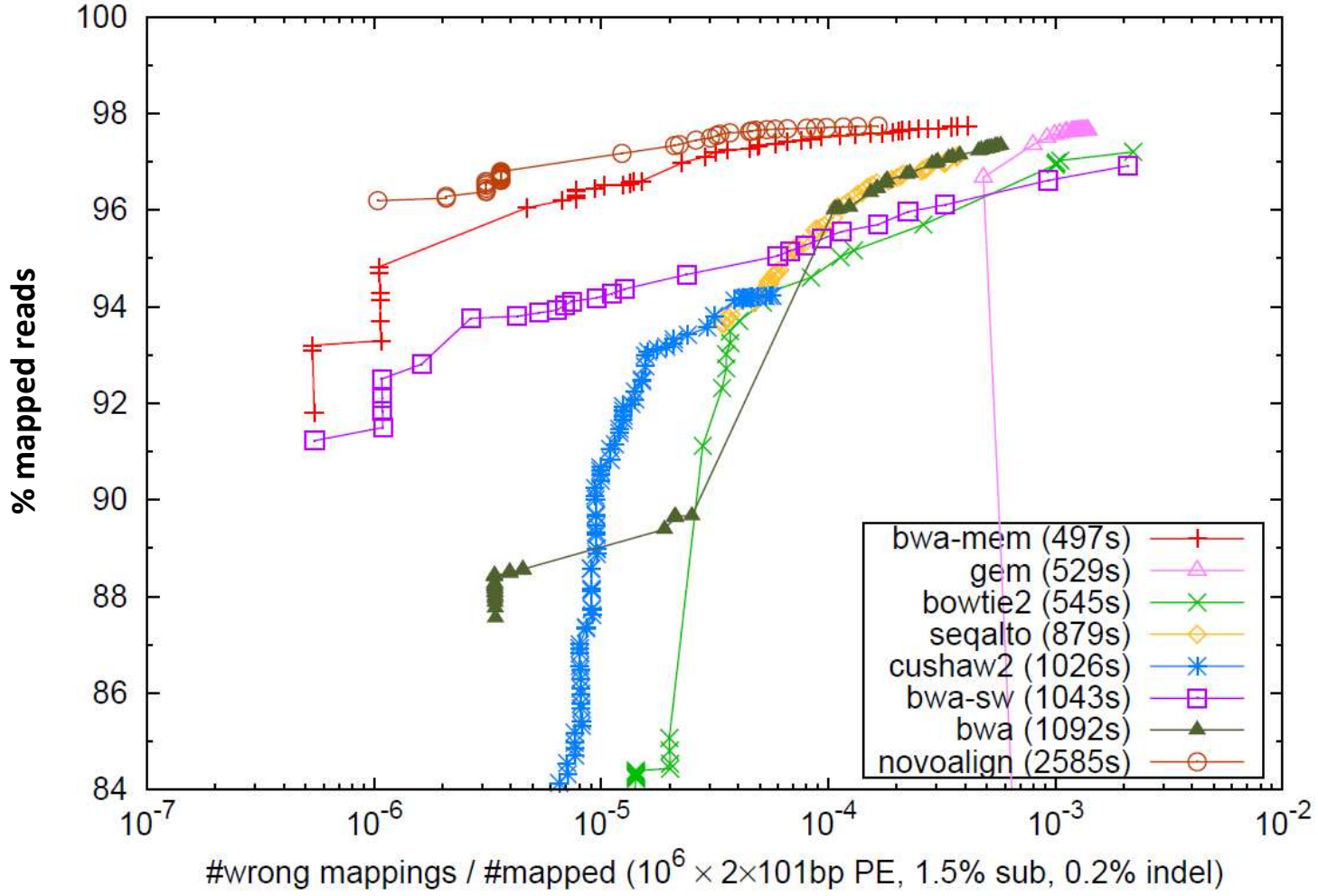
From: Konrad Paszkiewicz, University of Exeter, http://evomics.org/2014/01/alignment-methods/

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… fffffffffff… 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… fffffffffff… NM:i:0  MD:Z:100     AS:i:100 XS:i:100  RG:Z:SRR1663609
```

Header

read name

chr

flag

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

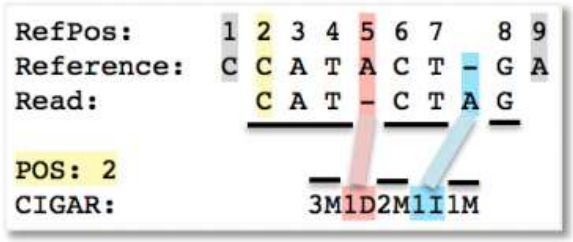
TAGS

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



- **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 (000000000001) – 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)
```

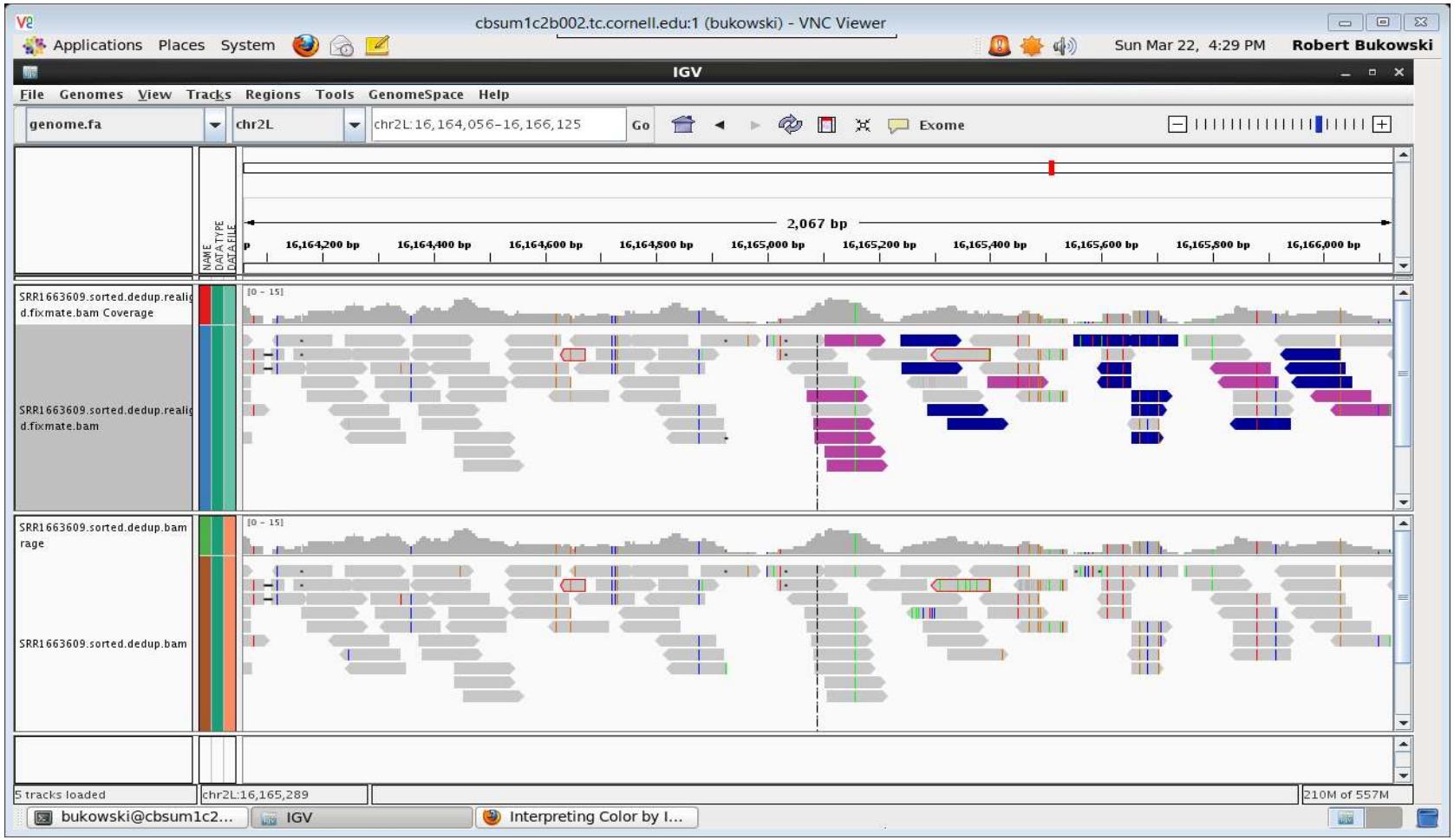
Looking into a BAM file: IGV viewer

Look at multiple BAM files

Zoom in and out

Various color-coding schemes

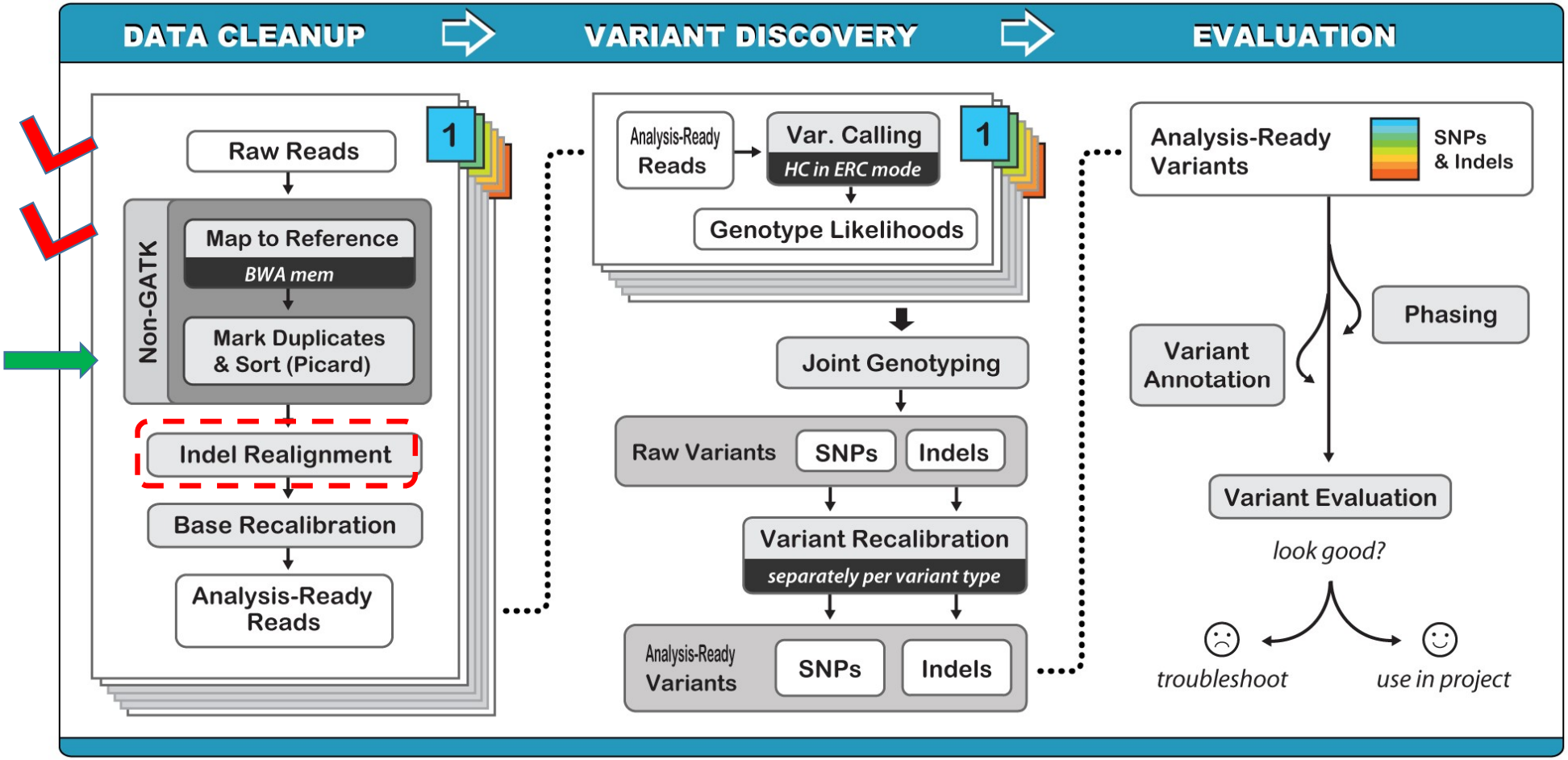
Can load genome annotation track



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

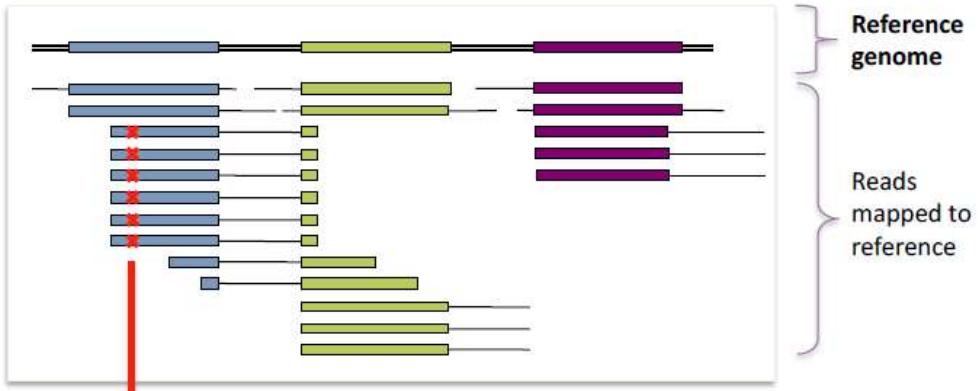
  - Very similar in sequence (except sequencing errors).
  - Identified where the first, say, 50 bases are identical between two reads and the read's coordinates are close

- **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



FP variant call (bad)

After marking duplicates, the GATK will only see :

… and thus be more likely to make the right call

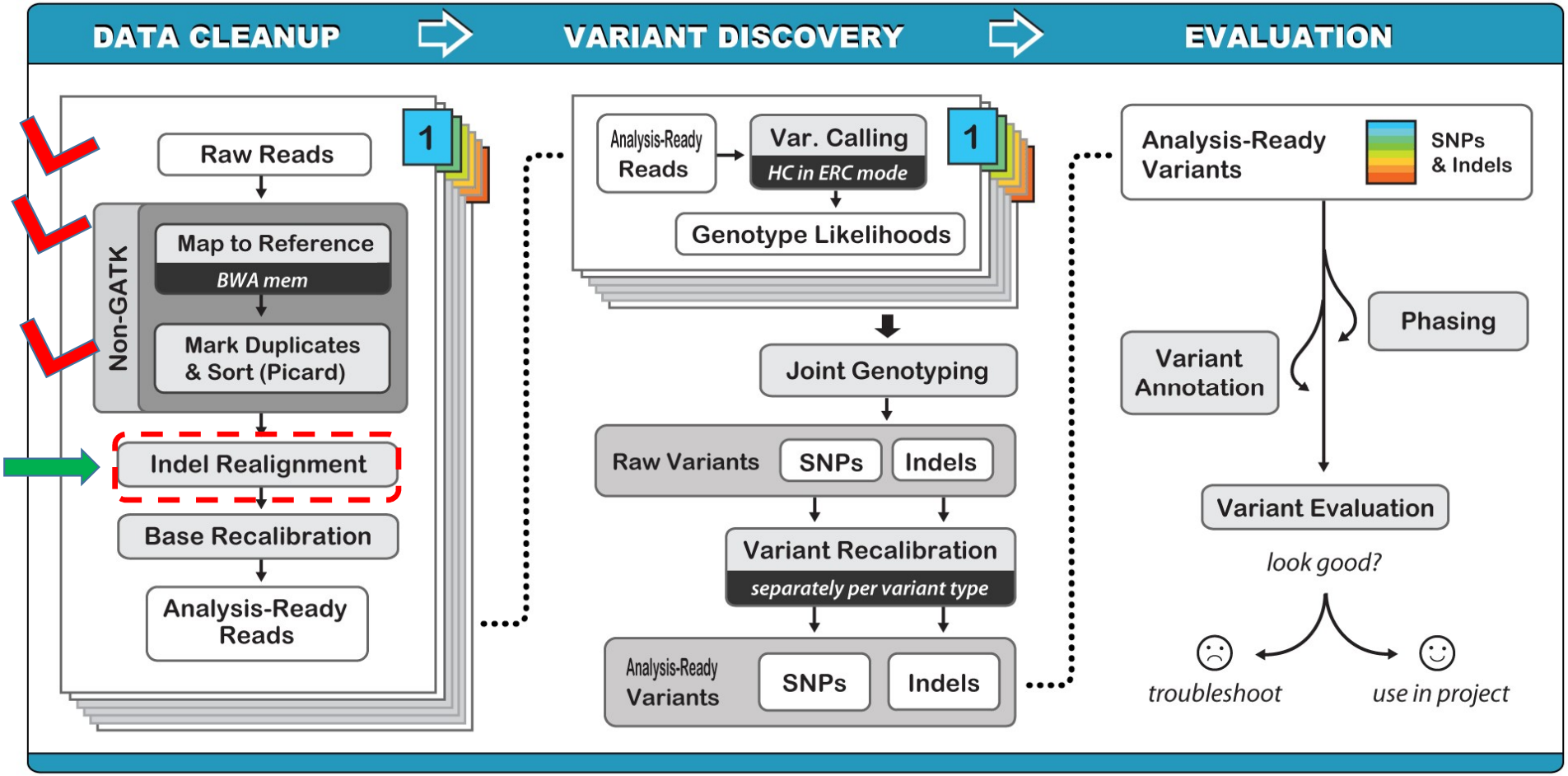
How removing (marking) duplicates works

Removing (marking) duplicates with 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

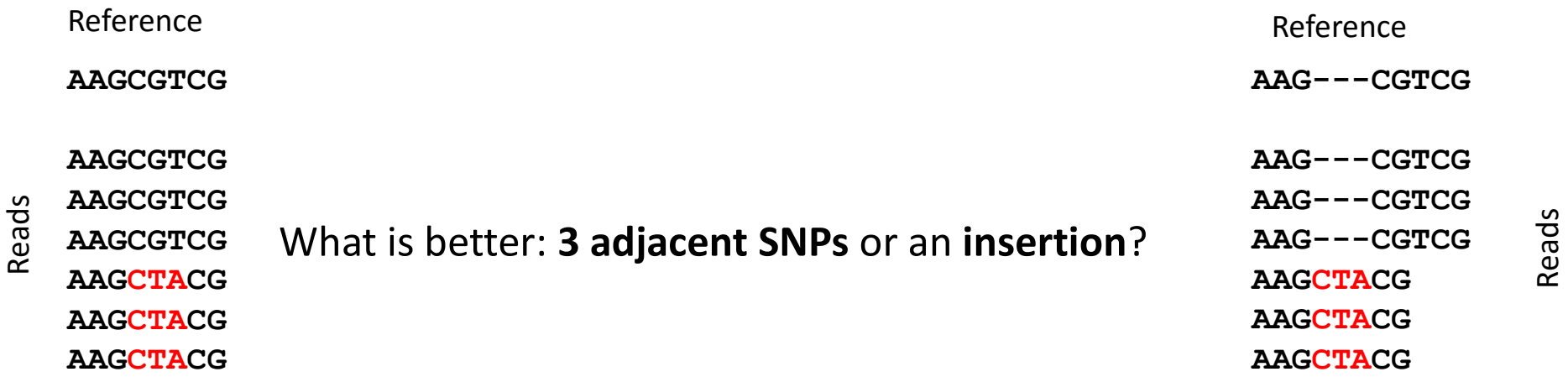
```
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
CTTTAGTTTCTTTTGCCGCTTTCTTTCTTTCTTTTTTTTTAAGTCTCCCTC
```

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

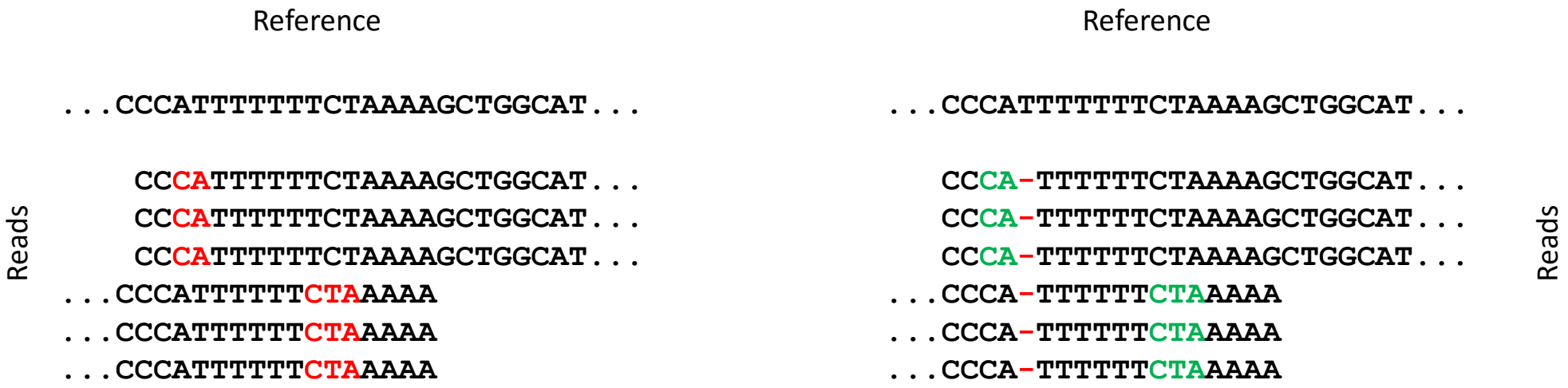
This looks better !

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

Ambiguity of alignment: around adjacent SNPs



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



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!

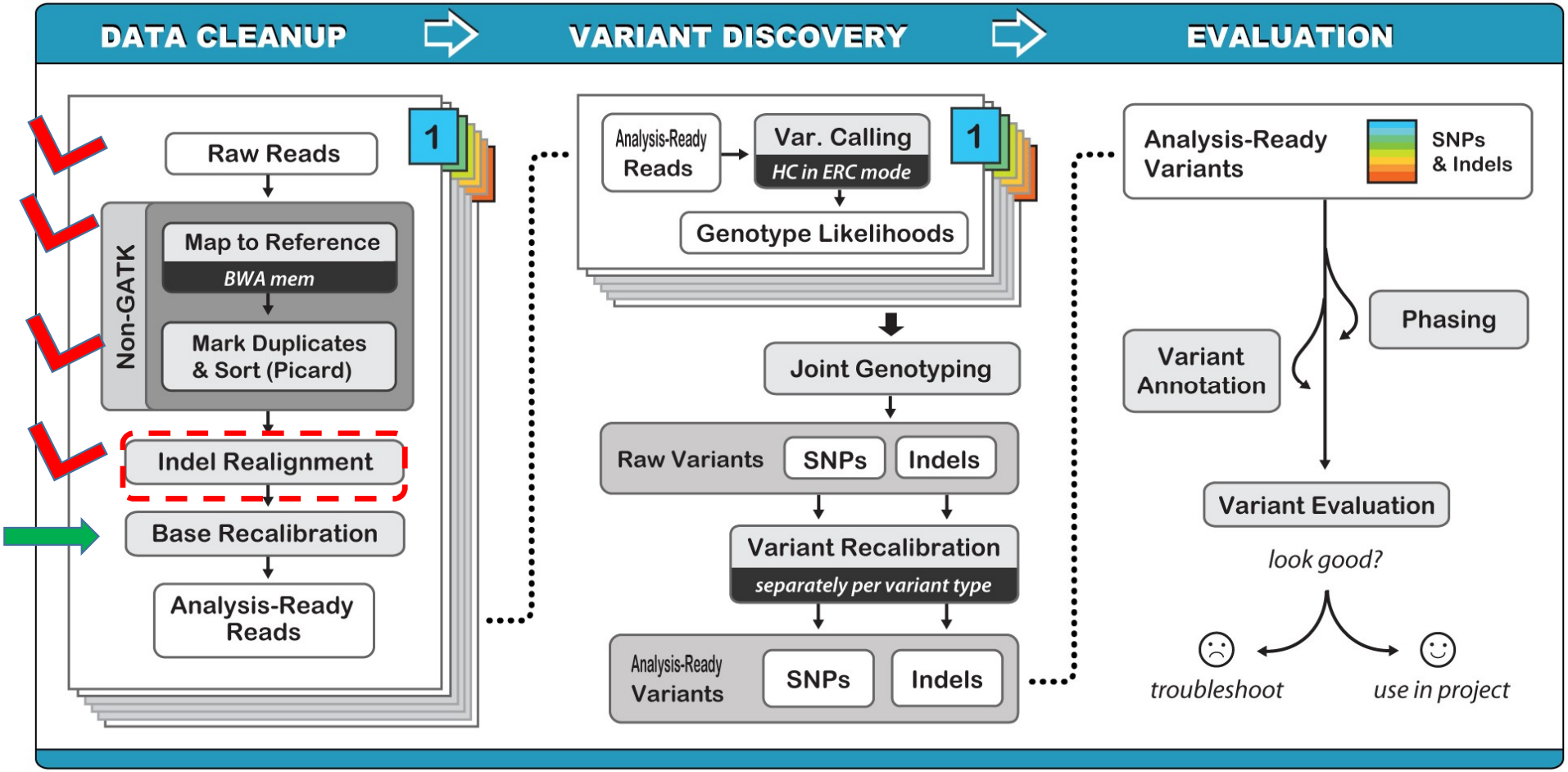
Re-alignment no longer recommended if the genotyping method used downstream involves **local haplotype assembly**

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

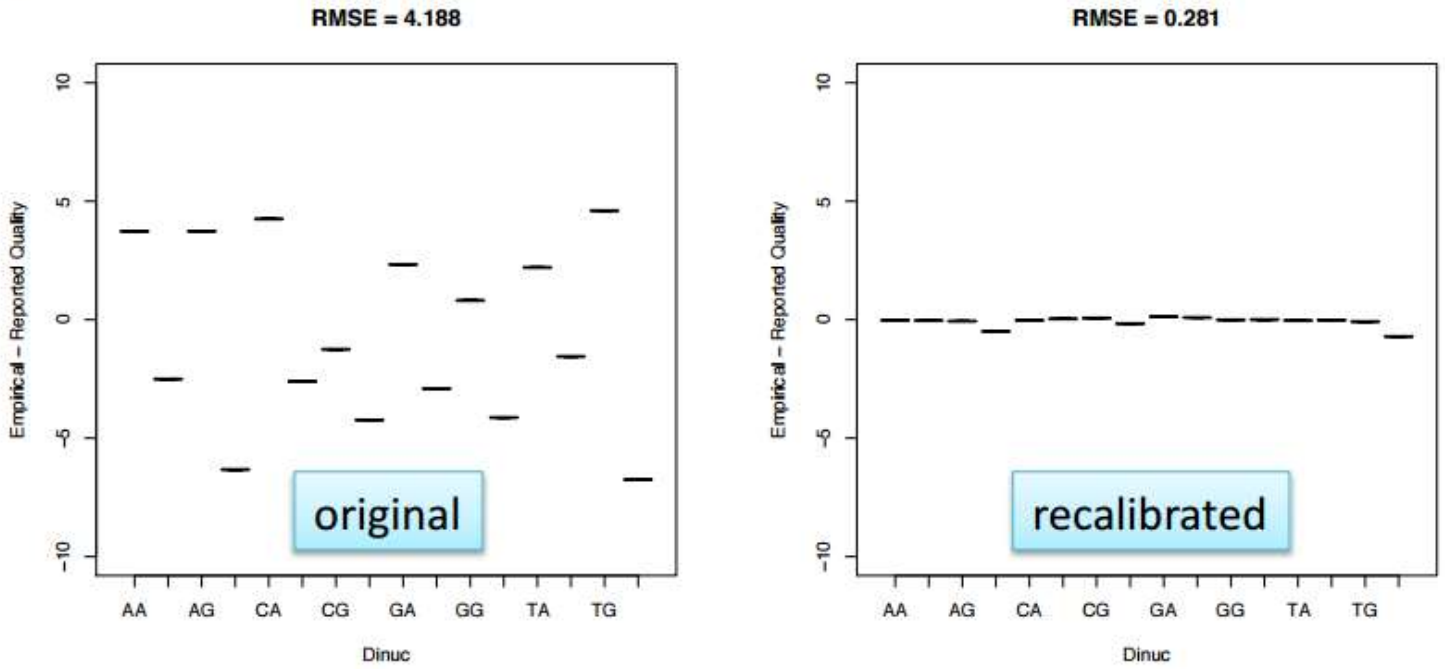
- Define “bins” in terms of covariates:
  - Lane
  - Original quality score
  - Machine cycle (position on read)
  - Sequencing context (what bases are around)
- Scan all aligned reads (i.e., bases) in a given read group
  - Classify each base to a “bin”; decide whether it is a mismatch
- In each bin
  - 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

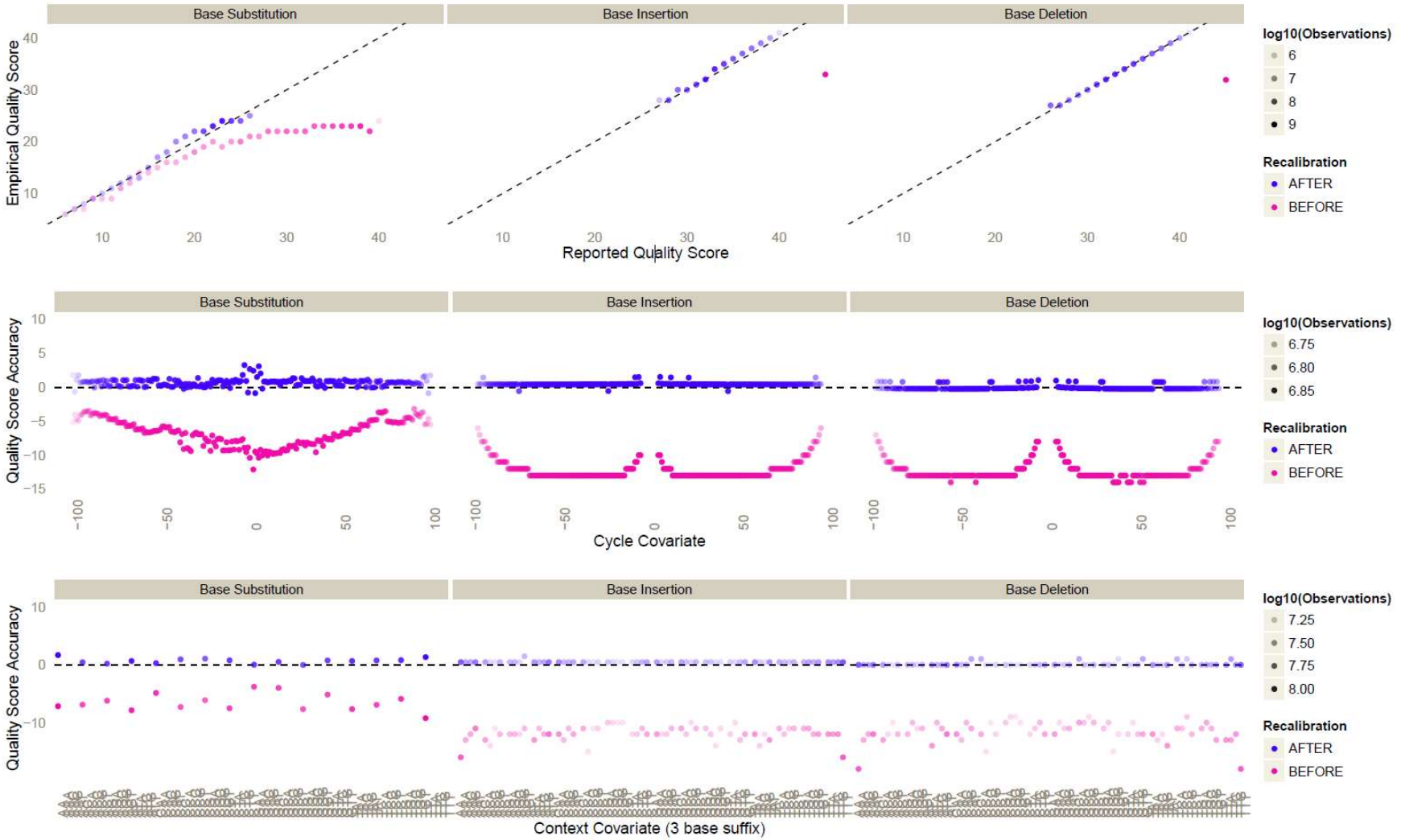
Collect mismatch statistics in bins

```
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!