Analysing re-sequencing samples

Anna Johansson

WABI / SciLifeLab

What is resequencing?

- You have a reference genome
  - represents one individual
- You generate sequence from other individuals
  - same species
  - closely related species
- How representative of your genome?
- You want to identify variation
  1) map millions of reads to reference genome
  2) SNPs / indels / structural variation

What accuracy is required?

- Is the result of sequencing the final answer or will it be used for something else?
- What is the importance of reducing false positives and false negatives relative to sequencing cost?

Example 1: identification of new mutations

- e.g. comparison of tumour vs. normal tissue or comparison of parents vs offspring
- sensitivity to false positives and false negatives is high
- mutations extremely rare
- FP rate >1 per Mb will swamp signal
- samples may be precious

Example 2: SNP discovery

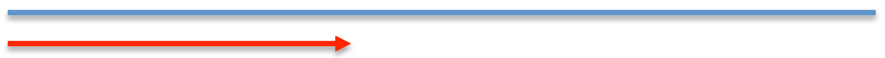
- Sequencing multiple individuals in order to design a SNP array
- High tolerance to false positives and false negatives (they will be validated by array)
- Does not need to be comprehensive – lower coverage acceptable
- Only interested in identifying markers to (e.g.) analyze population structure

Example 3: selection mapping

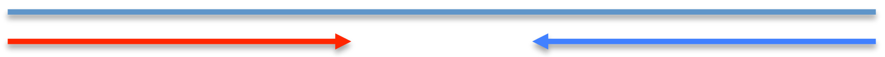
- Sequencing multiple individuals in order to scan genetic variation for signals of selection
- Looking for regions with reduced levels of SNP variation
- low false positive rate important
  - or selective sweeps will be obscured by noise

Types of reads

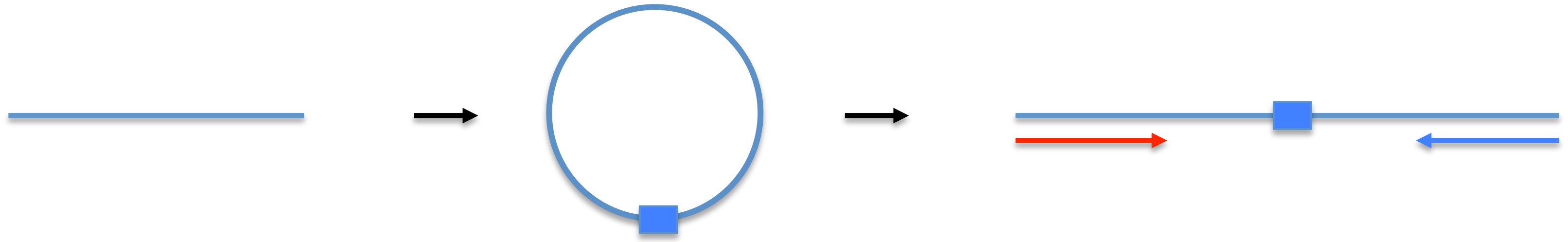
- fragment



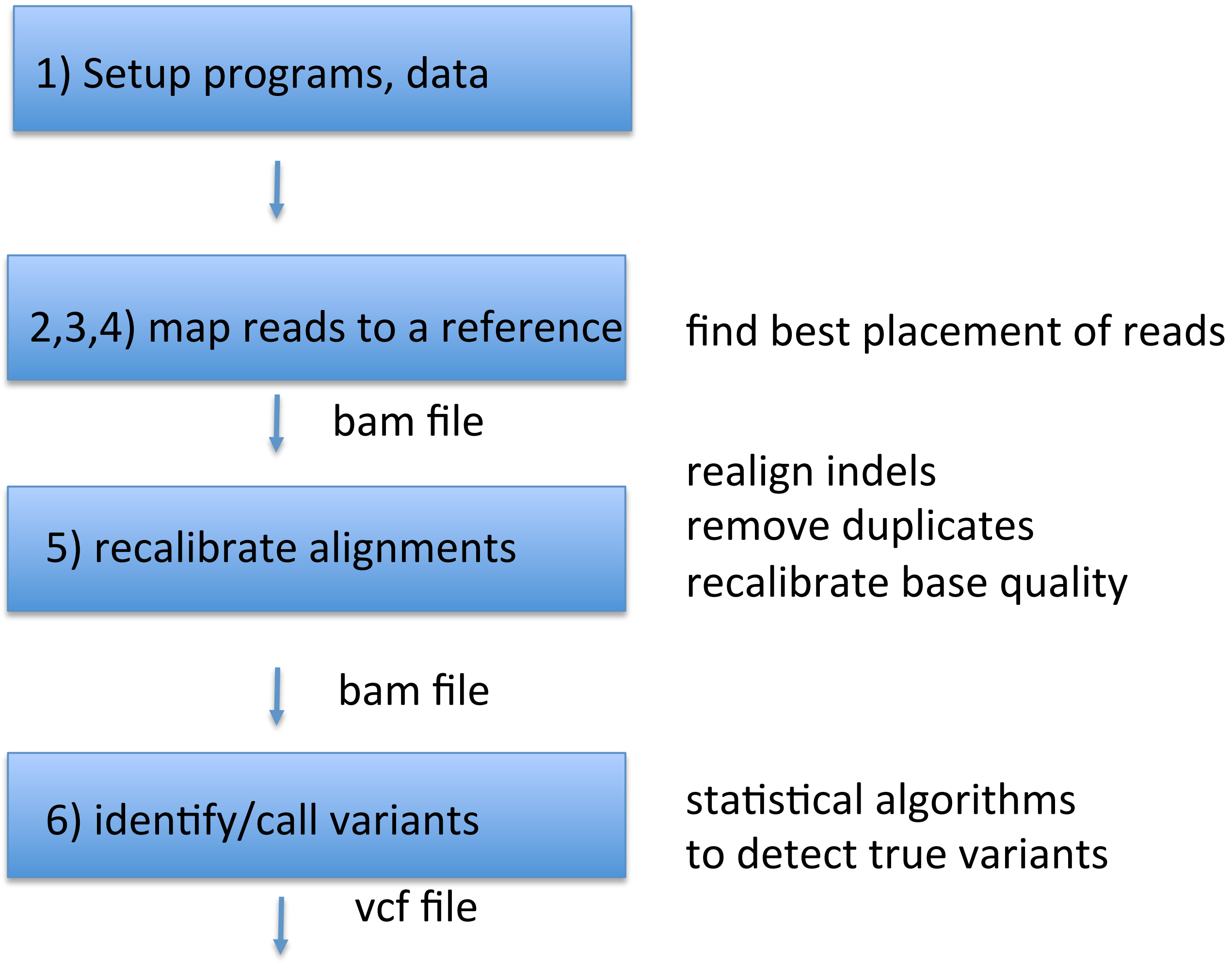
- paired-end



- mate pair (jumping libraries)



Steps in resequencing



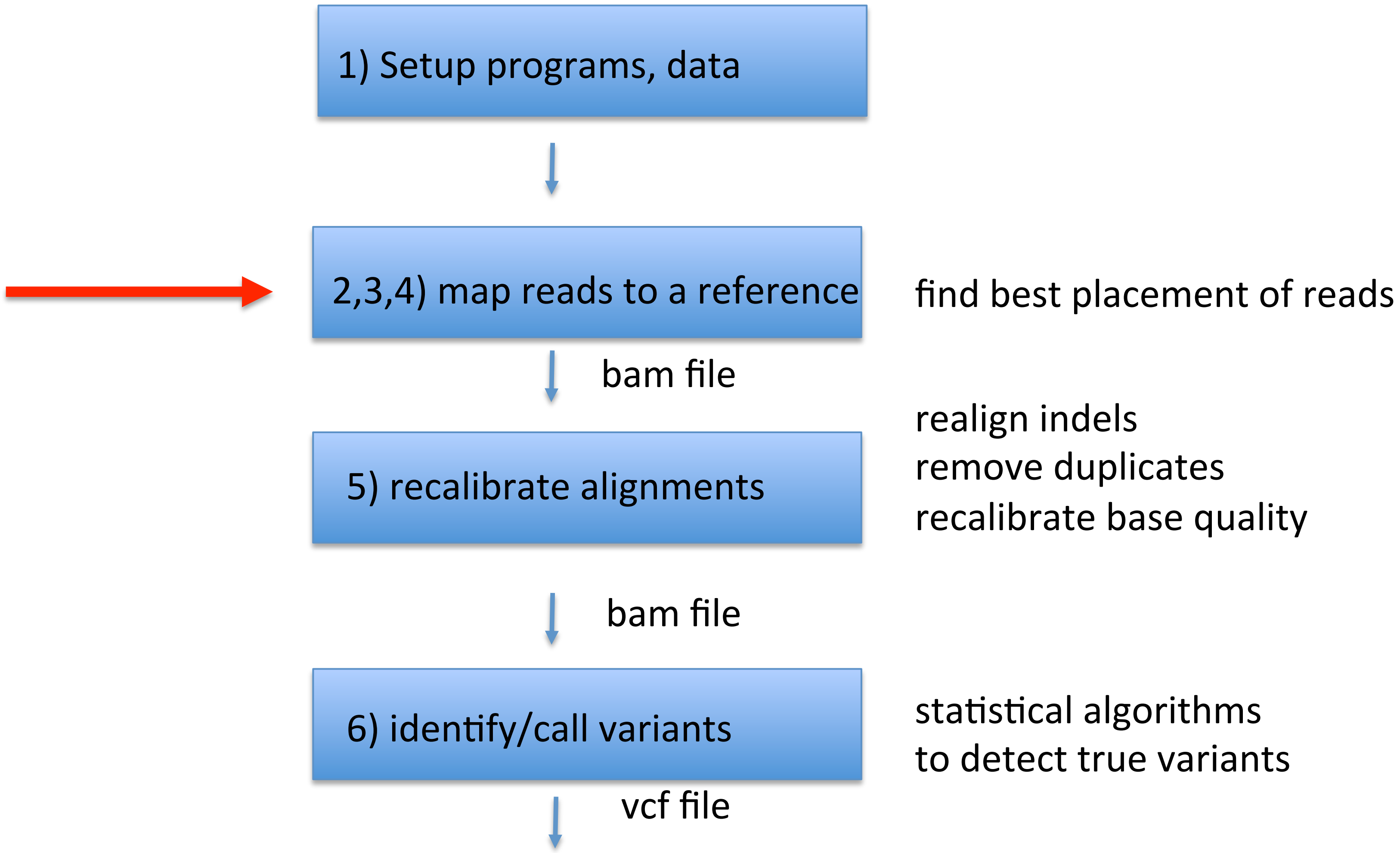
Benefits of each library type

- Fragments
  - fastest runs (one read per fragment)
  - lowest cost
- Paired reads
  - More data per fragment
  - improved mapping and assembly
  - same library steps, more data

Benefits of each library type

- Mate pairs (jumping libraries)
  - Paired end separation limited by fragment size
  - only way to make long jumps
  - Very useful for
    - assembly and alignment across duplications and low-complexity DNA
    - identification of large structural variants
    - phasing of SNPs
  - More DNA required
  - Not all platforms can read second strand

Steps in resequencing



Step 2: Map reads

- Maq (http://maq.sourceforge.net/)
  - nongapped
- BWA (http://bio-bwa.sourceforge.net/)
  - Burroughs-Wheeler aligner
  - gapped
  - successor to Maq
- bowtie (http://bowtie-bio.sourceforge.net/index.shtml)
  - fast + memory efficient
- Mosaik (http://bioinformatics.bc.edu/marthlab/)
  - Smith-Waterman

Step 2: Map reads

- Maq (http://maq.sourceforge.net/)
  - nongapped
- BWA (http://bio-bwa.sourceforge.net/)
  - Burroughs-Wheeler aligner
  - gapped
  - successor to Maq
- bowtie (http://bowtie-bio.sourceforge.net/index.shtml)
  - fast + memory efficient
- Mosaik (http://bioinformatics.bc.edu/marthlab/)
  - Smith-Waterman

mapping algorithm tricks

- simple brute force
- hash tables
- suffix trees
- Burroughs-Wheeler transform

brute force

```
TCGATCC
x
GACCTCATCGATCCCACTG
```

brute force

```
 TCGATCC
 x
GACCTCATCGATCCCACTG
```

brute force

```
  TCGATCC
  x
GACCTCATCGATCCCACTG
```

brute force

```
   TCGATCC
   x
GACCTCATCGATCCCACTG
```

brute force

```
    TCGATCC
    ||x
GACCTCATCGATCCCACTG
```

brute force

```
     TCGATCC
     x
GACCTCATCGATCCCACTG
```

brute force

```
      TCGATCC
      x
GACCTCATCGATCCCACTG
```

brute force

```
       TCGATCC
       |||||||
GACCTCATCGATCCCACTG
```

hash tables

build an index of the reference sequence for fast access

seed length 7

```
0    5    10   15

GACCTCATCGATCCCACTG
GACCTCA               →   chromosome 1, pos 0
 ACCTCAT              →   chromosome 1, pos 1
  CCTCATC             →   chromosome 1, pos 2
   CTCATCG            →   chromosome 1, pos 3
    TCATCGA           →   chromosome 1, pos 4
     CATCGAT          →   chromosome 1, pos 5
      ATCGATC         →   chromosome 1, pos 6
       TCGATCC        →   chromosome 1, pos 7
        CGATCCC       →   chromosome 1, pos 8
         GATCCCA      →   chromosome 1, pos 9
```

hash tables

build an index of the reference sequence for fast access

```
TCGATCC ?

                0    5    10   15

                GACCTCATCGATCCCACTG
                GACCTCA                →    chromosome 1, pos 0
                 ACCTCAT               →    chromosome 1, pos 1
                  CCTCATC              →    chromosome 1, pos 2
                   CTCATCG             →    chromosome 1, pos 3
                    TCATCGA            →    chromosome 1, pos 4
                     CATCGAT           →    chromosome 1, pos 5
                      ATCGATC          →    chromosome 1, pos 6
                       TCGATCC         →    chromosome 1, pos 7
                        CGATCCC        →    chromosome 1, pos 8
                         GATCCCA       →    chromosome 1, pos 9
```

hash tables

build an index of the reference sequence for fast access

```
TCGATCC = chromosome 1, pos 7

                0    5    10   15

                GACCTCATCGATCCCACTG
                GACCTCA              →    chromosome 1, pos 0
                 ACCTCAT             →    chromosome 1, pos 1
                  CCTCATC            →    chromosome 1, pos 2
                   CTCATCG           →    chromosome 1, pos 3
                    TCATCGA          →    chromosome 1, pos 4
                     CATCGAT         →    chromosome 1, pos 5
                      ATCGATC        →    chromosome 1, pos 6
                       TCGATCC       →    chromosome 1, pos 7
                        CGATCCC      →    chromosome 1, pos 8
                         GATCCCA     →    chromosome 1, pos 9
```

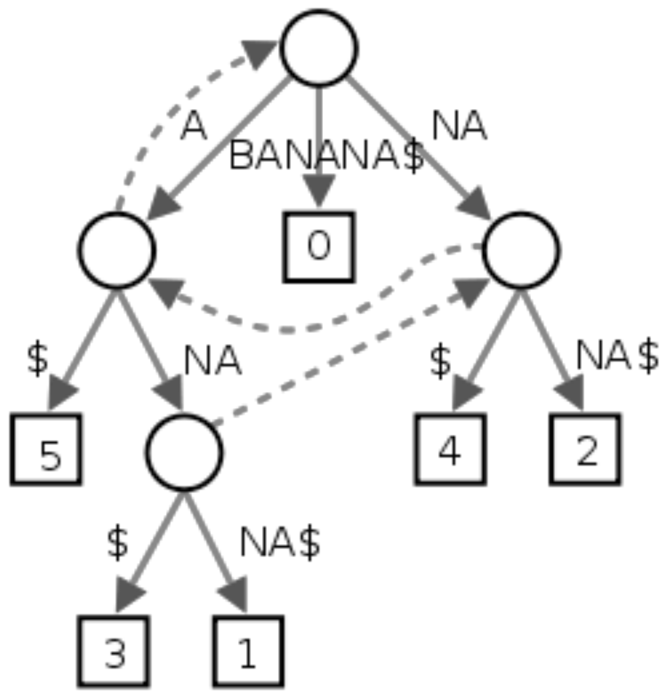
hash tables

Used by MAQ, Eland, SOAP, SHRiMP, ZOOM, partially by Mosaik

Problem: Indexing big genomes/lists of reads requires lots of memory

suffix trees

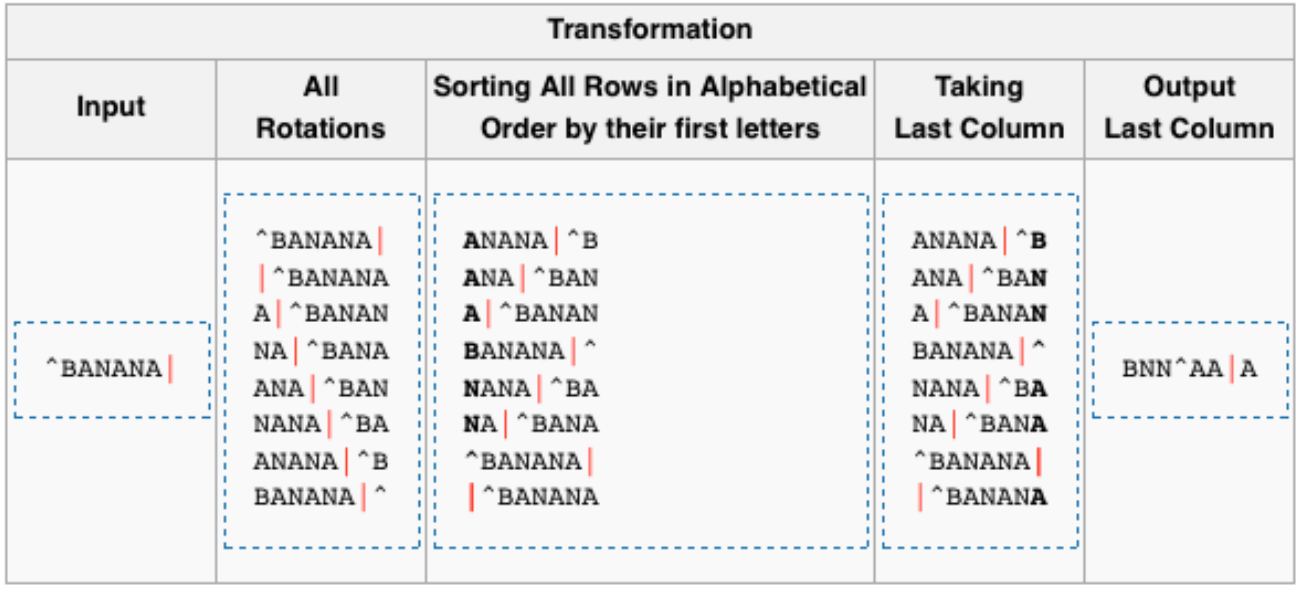
suffix tree for BANANA



breaks sequence into parts
(e.g. B, A, NA)
allows efficient searching of substrings in a sequence

Advantage: alignment of multiple identical copies of a substring in the reference is only needed to be done once because these identical copies collapse on a single path

Burroughs-Wheeler transform



Transformation				
Input	All Rotations	Sorting All Rows in Alphabetical Order by their first letters	Taking Last Column	Output Last Column
^BANANA|	^BANANA| |^BANANA A|^BANAN NA|^BANA ANA|^BAN NANA|^BA ANANA|^B BANANA|^	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	ANANA|^B ANA|^BAN A|^BANAN BANANA|^ NANA|^BA NA|^BANA ^BANANA| |^BANANA	BNN^AA|A

algorithm used in computer science for file compression
original sequence can be reconstructed
identical characters more likely to be consecutive → reduces memory required

Mapping algorithms

- Bowtie and BWA exploit suffix tree and BW transform
- Increases speed and decreases memory needed
- Standard output is now SAM or SAM binary (BAM) format

SAM format

HEADER SECTION

```
@HD VN:1.0 SO:coordinate
@SQ SN:1 LN:249250621 AS:NCBI37 UR:file:/data/local/ref/GATK/human_g1k_v37.fasta M5:1b22b98cdeb4a9304cb5d48026a85128
@SQ SN:2 LN:243199373 AS:NCBI37 UR:file:/data/local/ref/GATK/human_g1k_v37.fasta M5:a0d9851da00400dec1098a9255ac712e
@SQ SN:3 LN:198022430 AS:NCBI37 UR:file:/data/local/ref/GATK/human_g1k_v37.fasta M5:fdfd811849cc2fadebc929bb925902e5
@RG ID:UM0098:1 PL:ILLUMINA PU:HWUSI-EAS1707-615LHAAXX-L001 LB:80 DT:2010-05-05T20:00:00-0400 SM:SD37743 CN:UMCORE
@RG ID:UM0098:2 PL:ILLUMINA PU:HWUSI-EAS1707-615LHAAXX-L002 LB:80 DT:2010-05-05T20:00:00-0400 SM:SD37743 CN:UMCORE
@PG ID:bwa VN:0.5.4
```

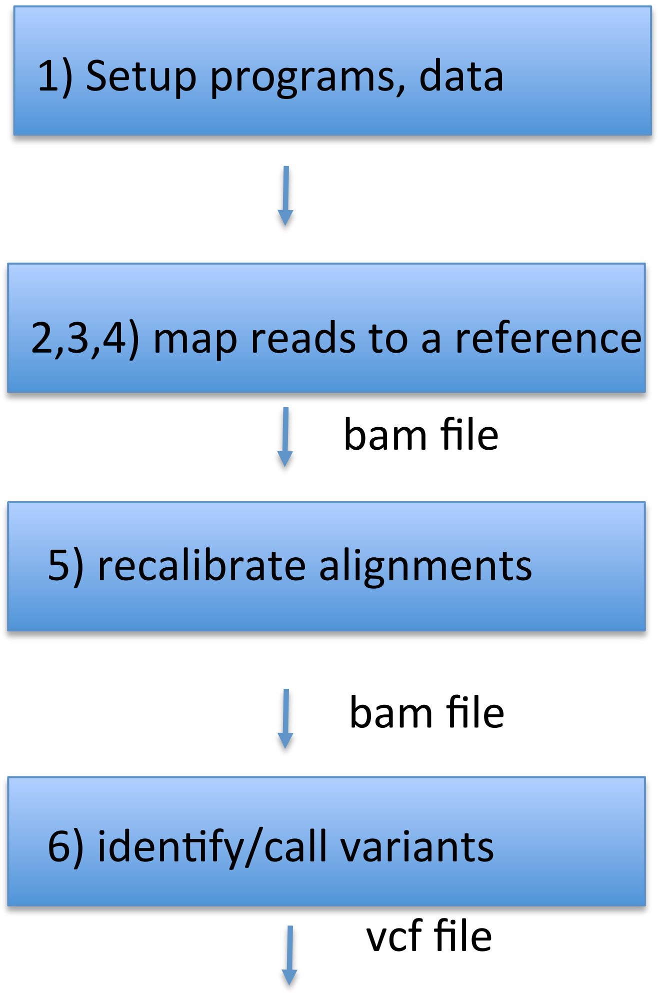
ALIGNMENT SECTION

```
8_96_444_1622 73 scaffold00005 155754 255 54M * 0 0 ATGTAAAGTATTTCCATGGTACACAGCTTGGTCGTAATGTGATTGCTGAGCCAG
BC@B5)5CBBCCBCCCBC@@7C>CBCCBCCC;57)8(@B@B>ABBCBC7BCC=> NM:i:0

8_80_1315_464 81 scaffold00005 155760 255 54M = 154948 0 AGTACCTCCCTGGTACACAGCTTGGTAAAAATGTGATTGCTGAGCCAGACCTTC B?@?
BA=>@>>7;ABA?BB@BAA;@BBBBBBAABABBBCABAB?BABA?BBBAB NM:i:0

8_17_1222_1577 73 scaffold00005 155783 255 40M1116N10M * 0 0 GGTAAAAATGTGATTGCTGAGCCAGACCTTCATCATGCAGTGAGAGACGC BB@BA??
>CCBA2AAABBBBBBB8A3@BABA;@A:>B=,;@B=A:BAAAA NM:i:0 XS:A:+ NS:i:0

8_43_1211_347 73 scaffold00005 155800 255 23M1116N27M * 0 0 TGAGCCAGACCTTCATCATGCAGTGAGAGACGCAAACATGCTGGTATTTG
#>8<=<@6/:@9';@7A@@BAAA@BABBBABBB@=<A@BBBBBBBBCCBB NM:i:2 XS:A:+ NS:i:0

8_32_1091_284 161 scaffold00005 156946 255 54M = 157071 0 CGCAAACATGCTGGTAGCTGTGACACCACATCAACAGCTTGACTATGTTTGTAA
BBBBB@AABACBCA8BBBBBABBBB@BBBBBBA@BBBBBBBBBA@:B@AA@=@@ NM:i:0
```

query name   ref. seq.   position   query seq.

quality. seq.

Steps in resequencing

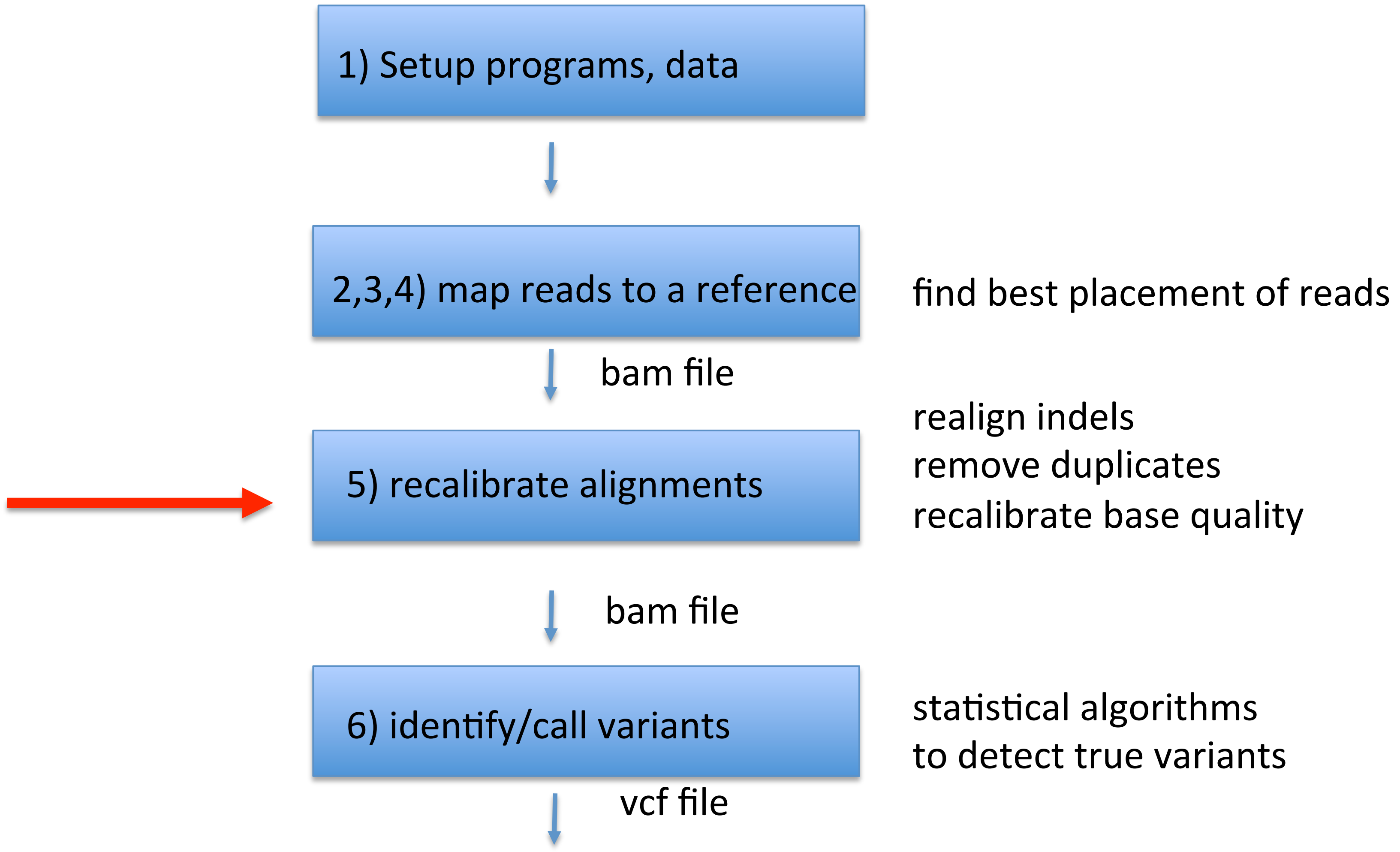

1) Setup programs, data
2,3,4) map reads to a reference
bam file
5) recalibrate alignments
bam file
6) identify/call variants
vcf file

find best placement of reads

realign indels
remove duplicates
recalibrate base quality

statistical algorithms
to detect true variants

Steps in resequencing



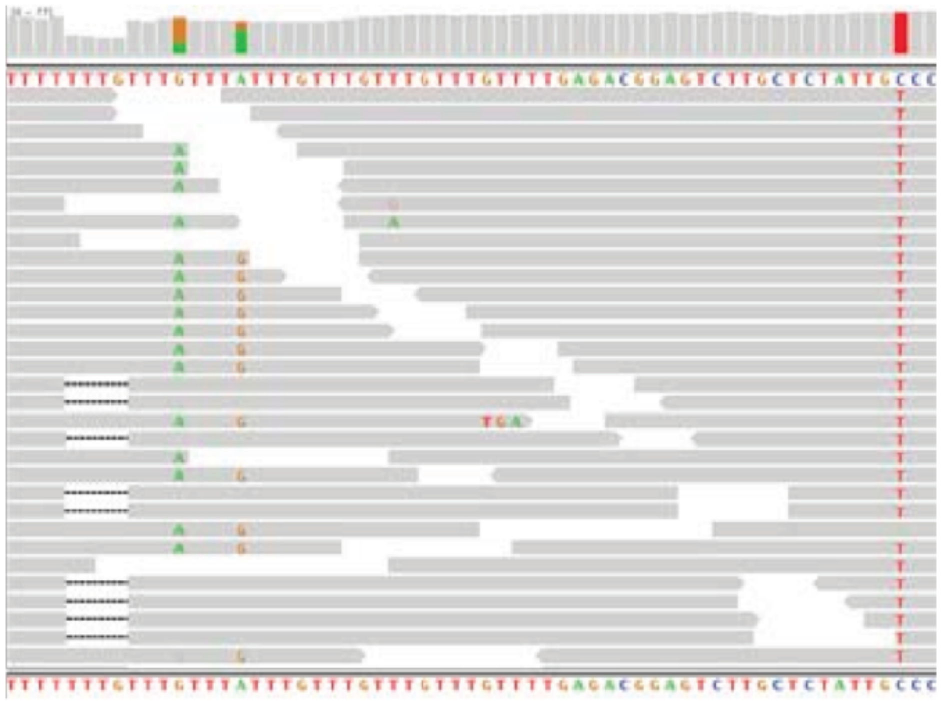
software

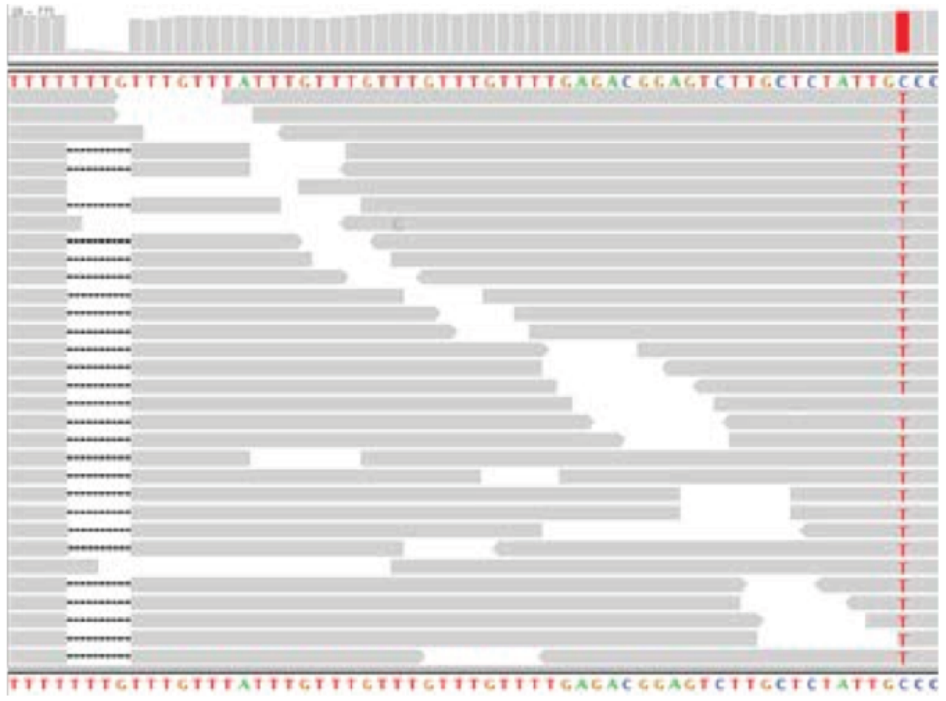
- Some very useful programs for manipulation of short reads and alignments:
- SAM Tools (http://samtools.sourceforge.net/)
  - provides various utilities for manipulating alignments in the SAM and BAM format, including sorting, merging, indexing and generating alignments in a per-position format.
- Picard (http://picard.sourceforge.net/)
  - comprises Java-based command-line utilities that manipulate SAM and BAM files
- Genome Analysis Toolkit (http://www.broadinstitute.org/gatk/)
  - GATK offers a wide variety of tools, with a primary focus on variant discovery and genotyping as well as strong emphasis on data quality assurance.
- Integrative Genomics viewer (http://www.broadinstitute.org/igv/)
  - IGV is very useful for visualizing mapped reads

step 2: recalibration

- 2.1 realign indels
- 2.2 remove duplicates
- 2.3 recalibrate base quality

2.1: local realignment


HiSeq data, raw BWA alignments

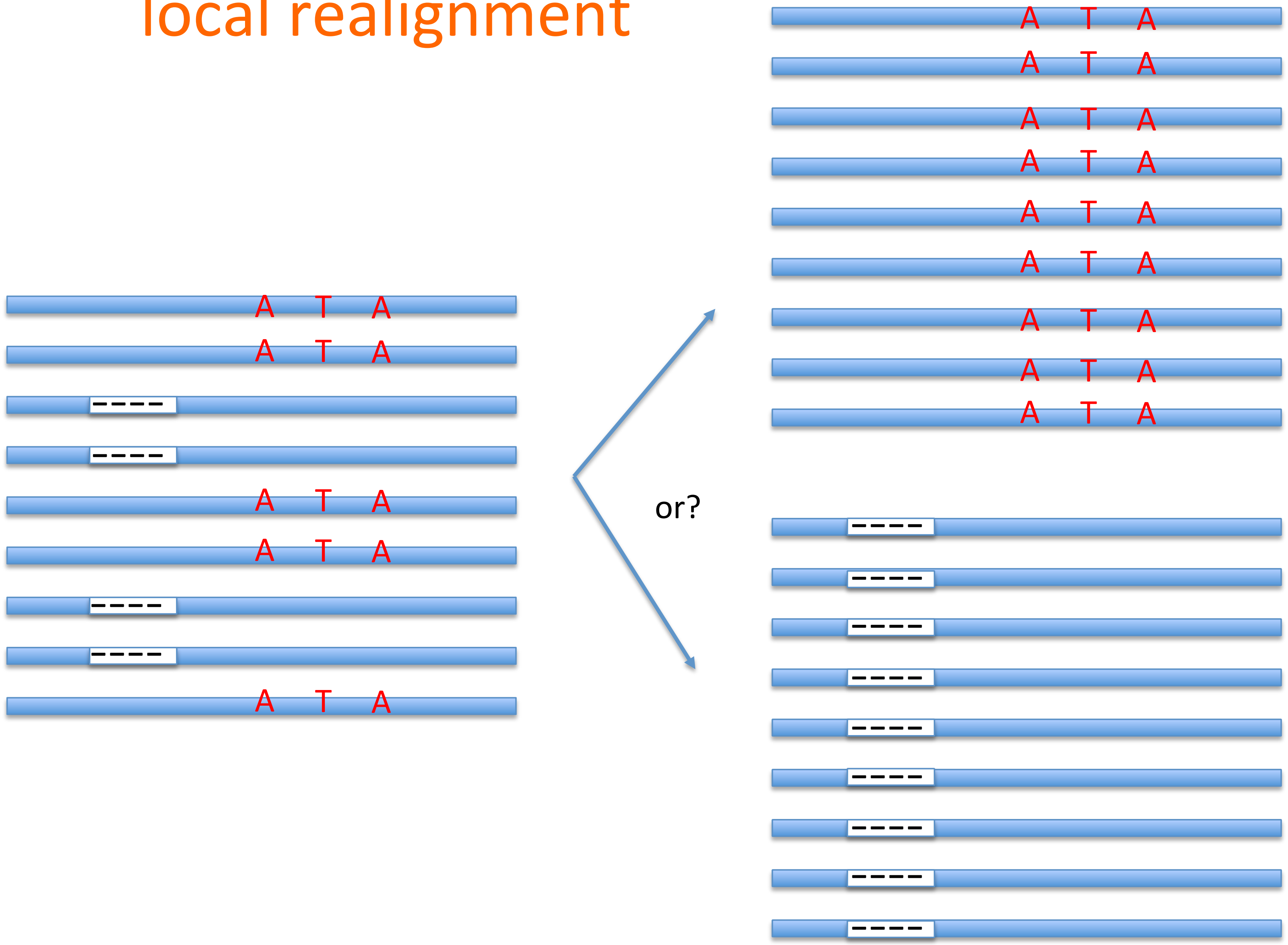

HiSeq data, after MSA

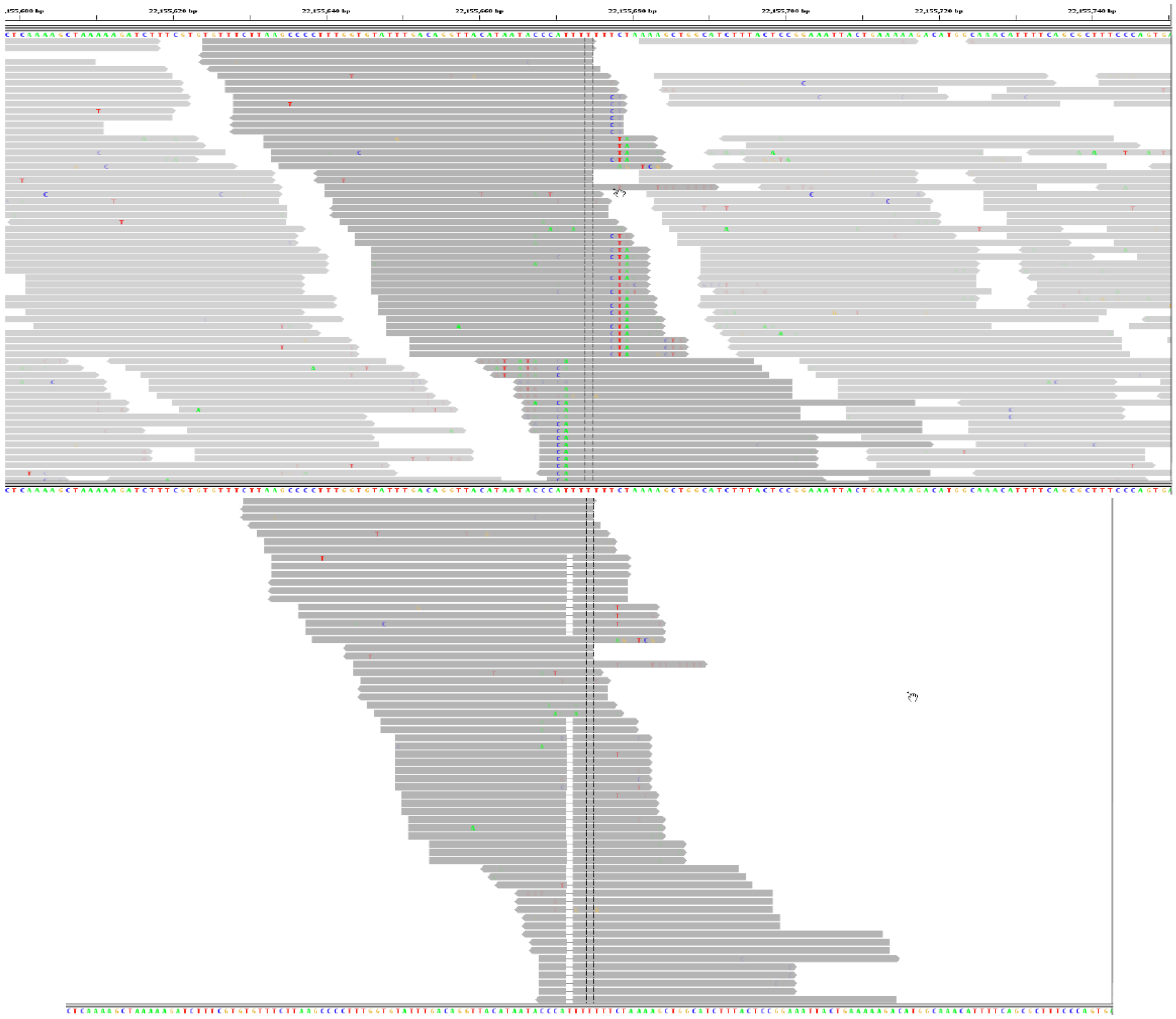
can be performed using GATK commands: `RealignerTargetCreator` followed by `IndelRealigner`

local realignment

- mapping is done one read at a time
- single variants may be split into multiple variants around indels
- solution: realign these regions taking all reads into account

local realignment





155,600 bp
22,155,620 bp
22,155,640 bp
22,155,660 bp
22,155,680 bp
22,155,700 bp
22,155,720 bp
22,155,740 bp

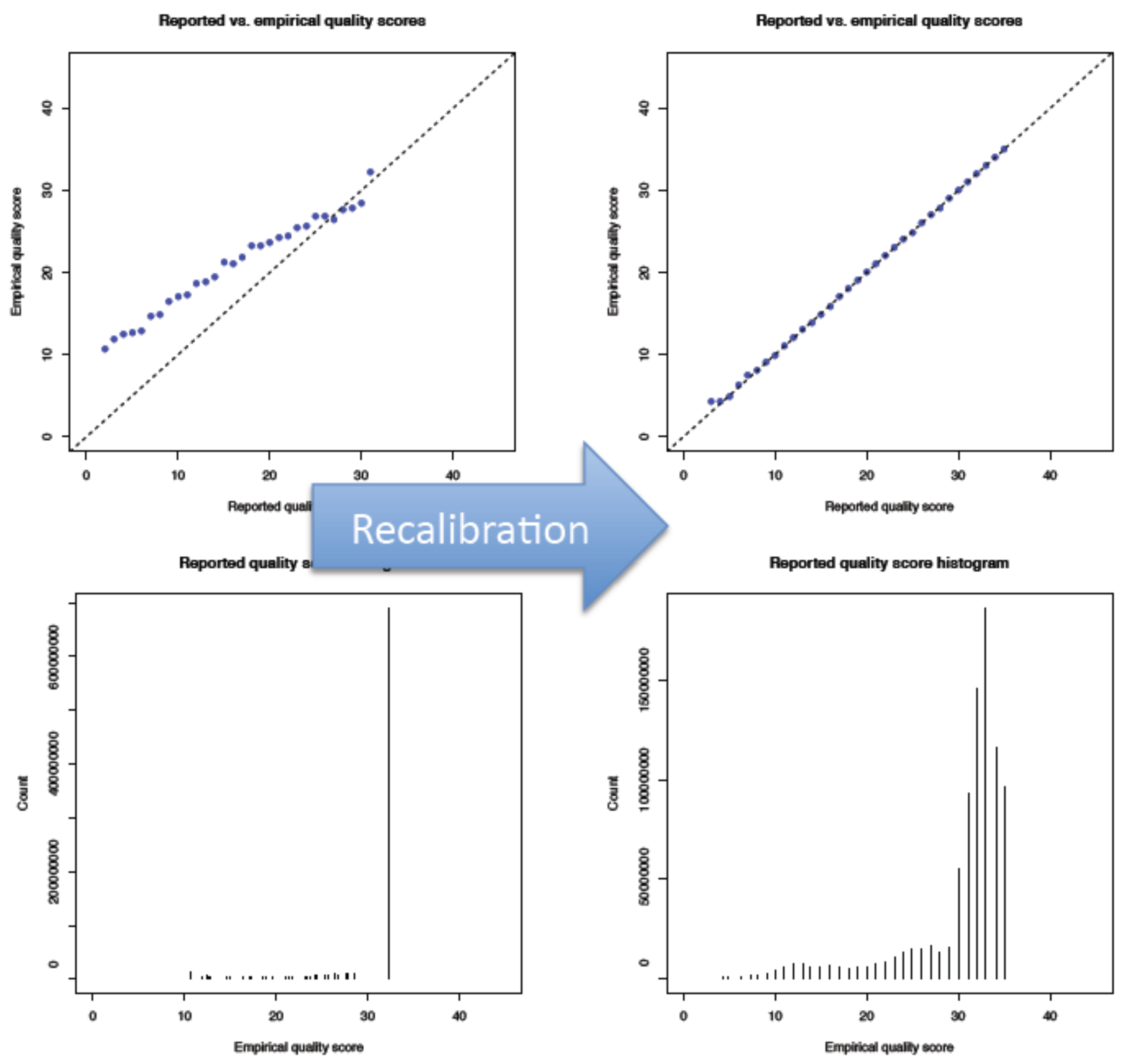
2.2 PCR duplicates

- When two or more reads originate from same molecule (artificial duplicates)
  - not independent observations
  - skew allele frequency and read depth
  - errors double counted
- PCR duplicates occur
  - during library prep, or
  - optical duplicates (one cluster read as two)
- mark or remove

Identify PCR duplicates

- Single or paired reads that map to identical positions
- Picard `MarkDuplicates`
- Optical duplicates occur close to each other on sequencer
- If low coverage, then duplicates are likely artifacts
- If high coverage, then more duplicates are real

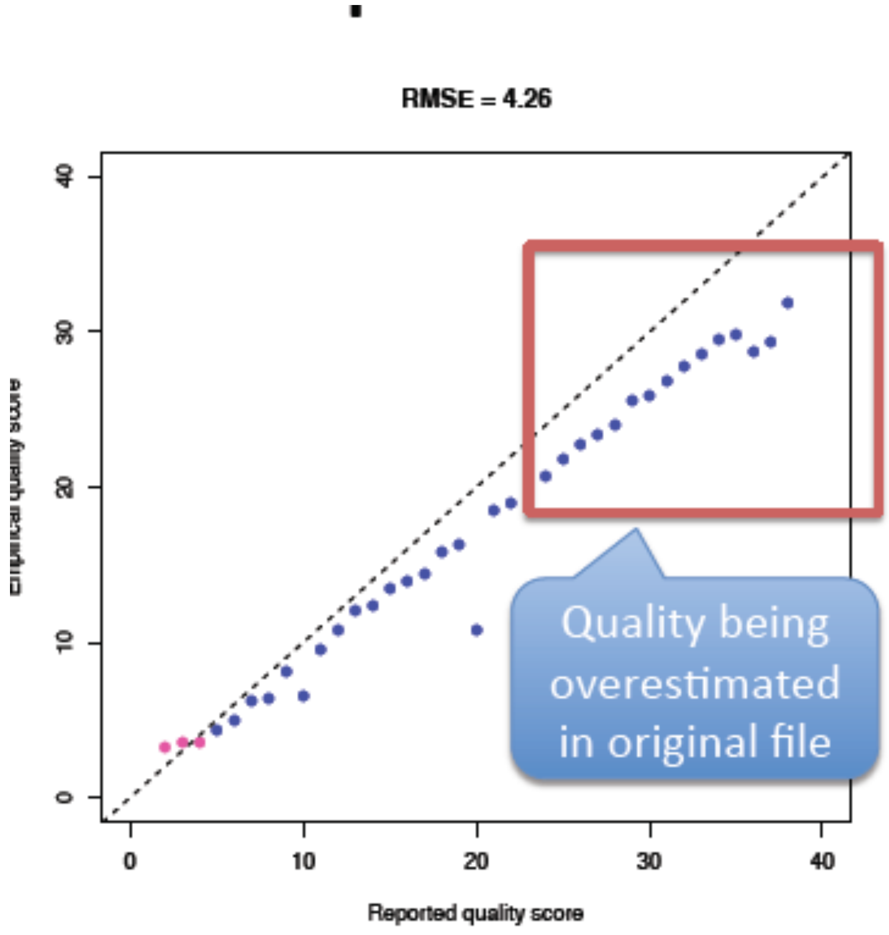
2.3 base quality recalibration

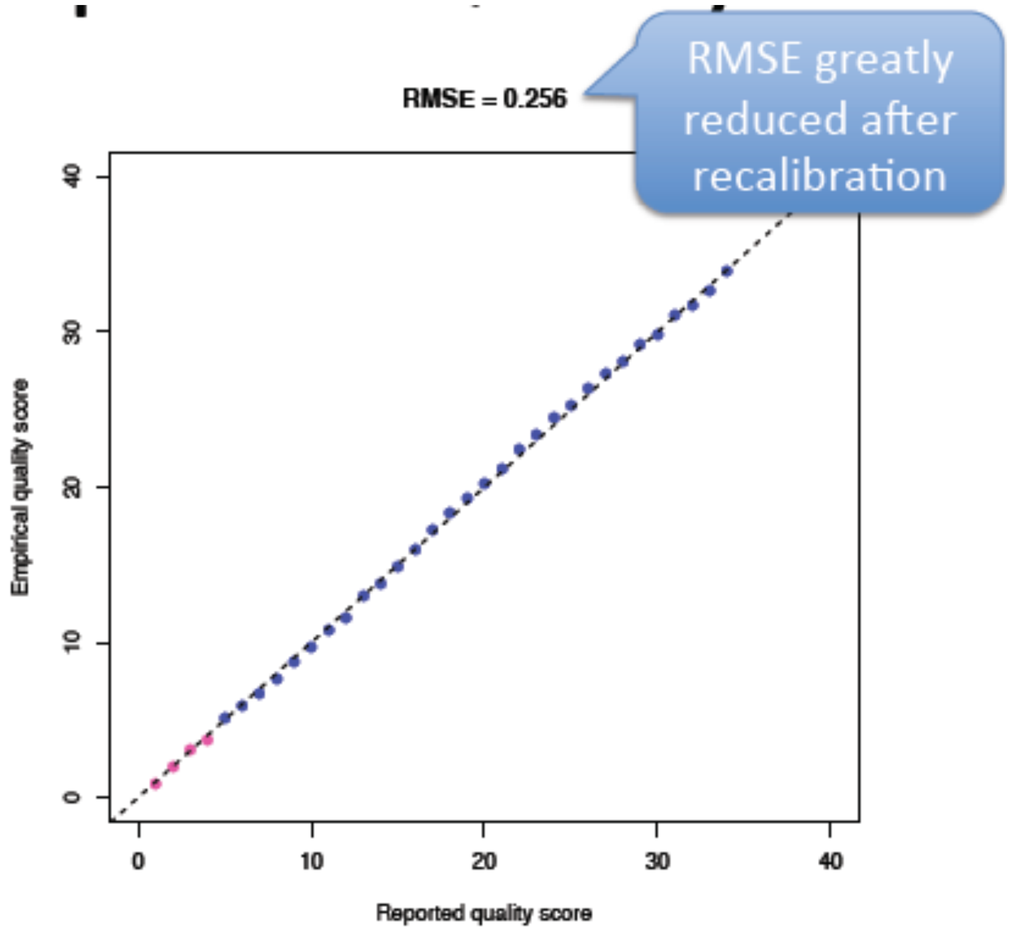


Recalibration Method

- Bin each base by
  - read group
  - called quality
  - position in read
  - local dinucleotide context
- score observed quality per bin
  - # of mismatches +1 / # of observed bases
- scale compared to reported quality

Reported vs empiral quality scores

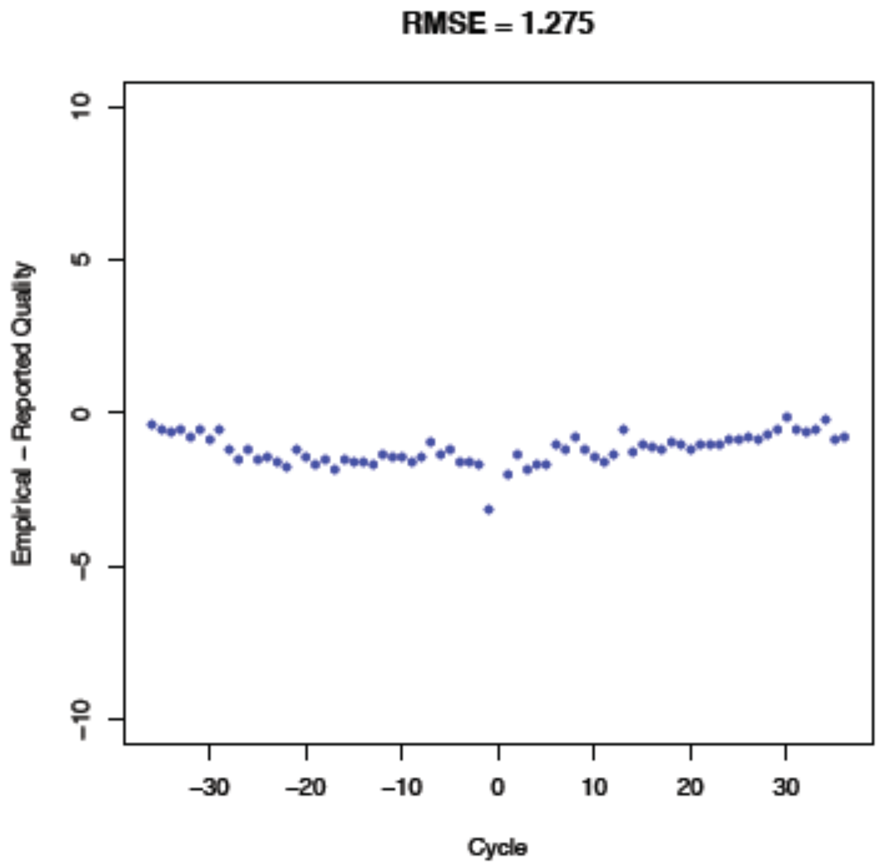


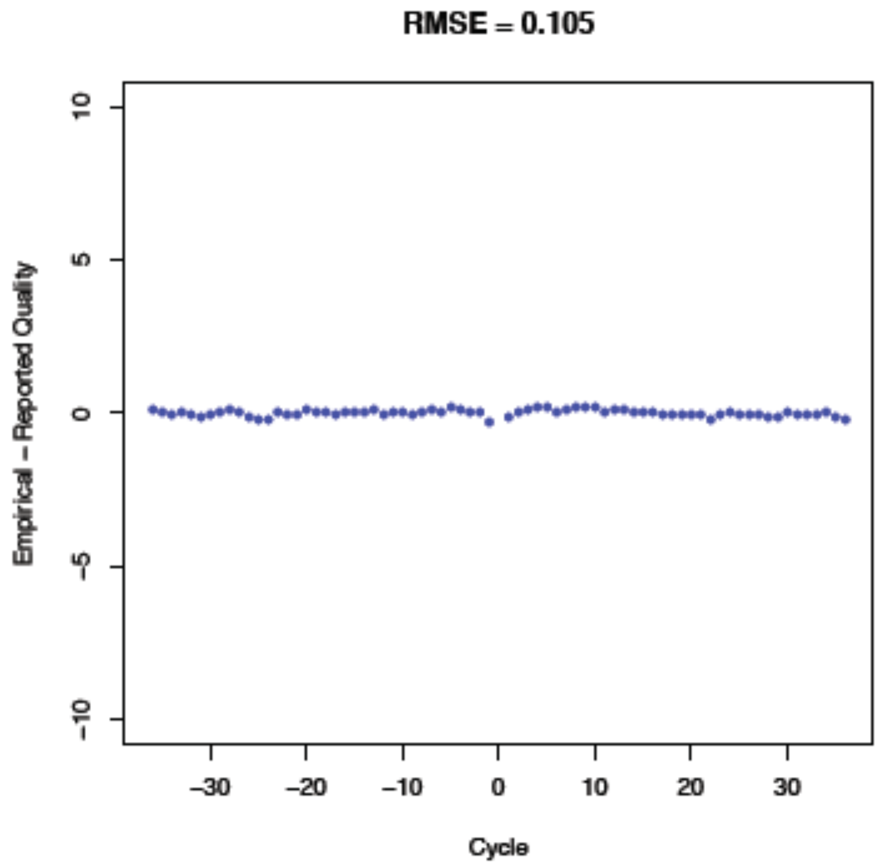


Before Recalibration

After Recalibration

Residual error by machine cycle

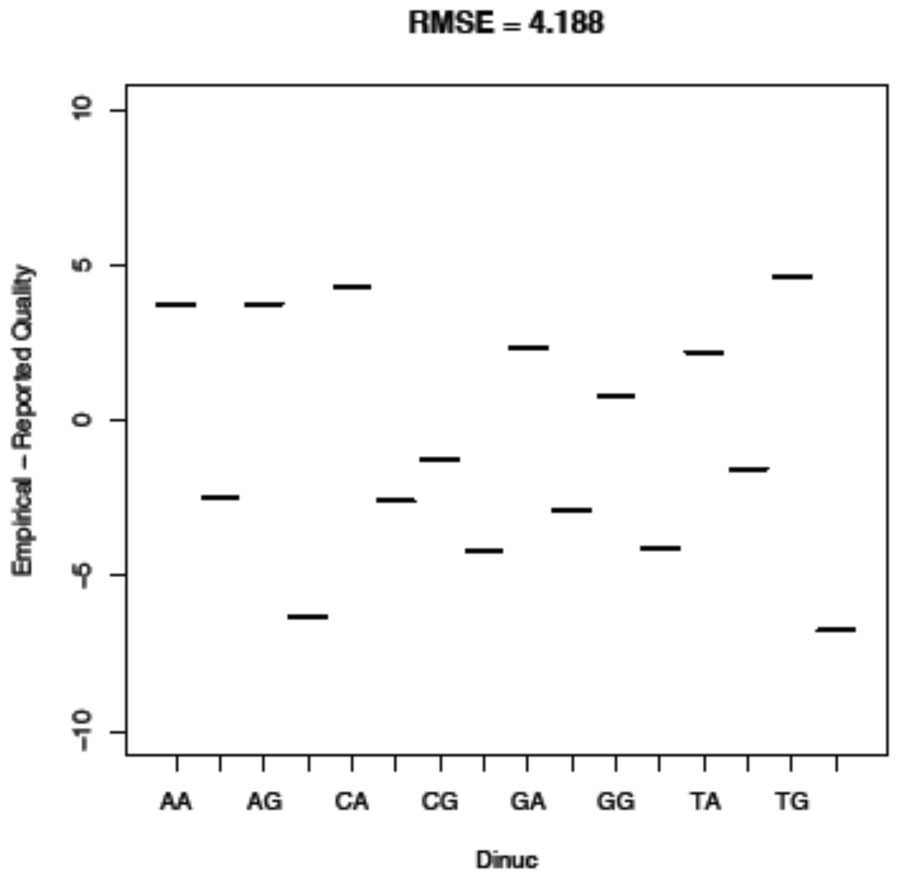


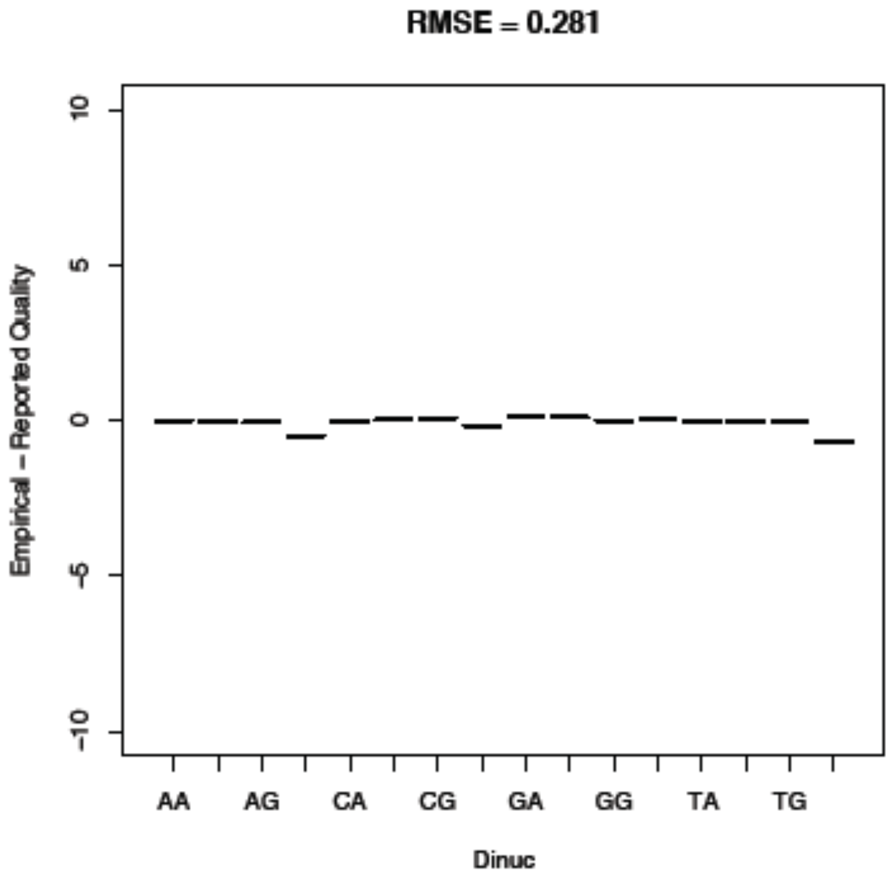


Before Recalibration

After Recalibration

Residual error by dinucleotide

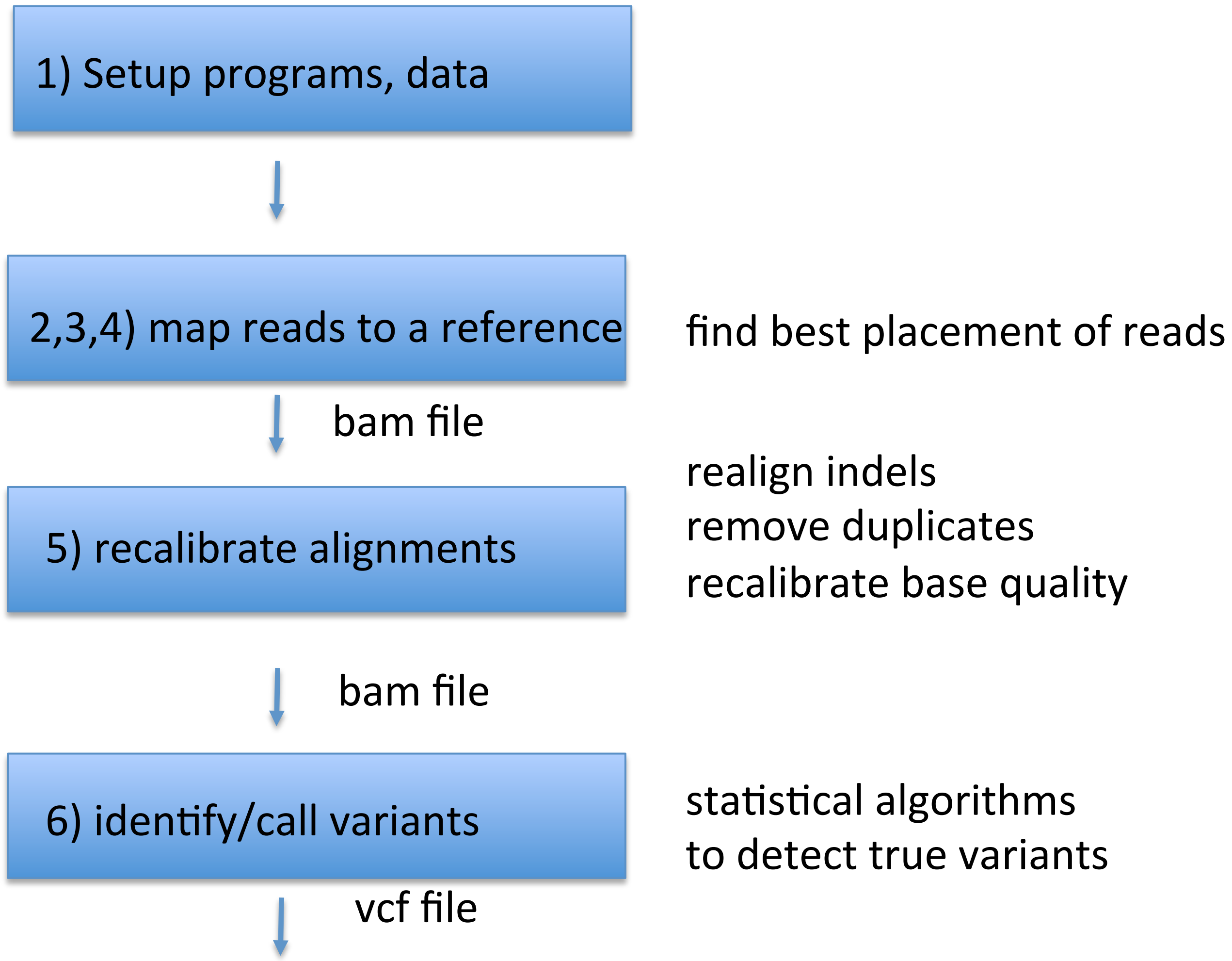




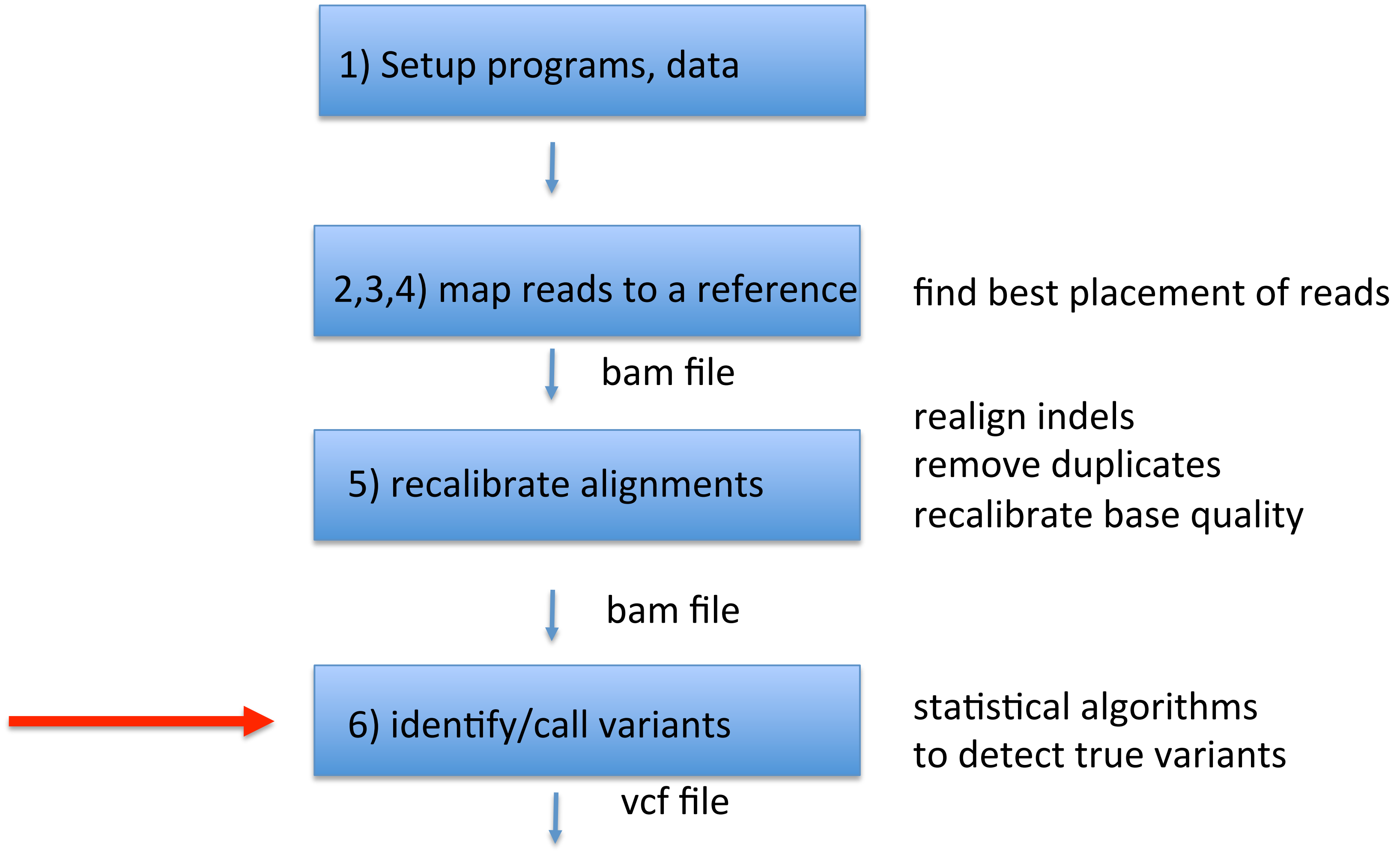
Before Recalibration

After Recalibration

Steps in resequencing



Steps in resequencing



Single Nucleotide Variant calling

- Genome Analysis Toolkit (http://www.broadinstitute.org/gatk/)
  - Integrated pipeline for SNP discovery (java)
- FreeBayes (http://bioinformatics.bc.edu/marthlab/FreeBayes)
  - Bayesian SNP calling (C++)

Both programs perform Bayesian population based SNP calling

simple pileup methods

```
acacagatagacatagacatagacagatgag
acacagatagacatagacatagacagatgag
acacacatagacatagacatagacagatgag
acacagatagacatagacatagacagatgag
acacagatagacatatacatagacagatgag
acacagatagacatatacatagacagatgag
acacagatagacatatacatagacagttgag
acacagatagacatagacatagacagatgag
acacagatagacatatacatagacagatgag
acacagatagacatagacatagacagatgag
```

simple pileup methods

- Count calls at each site
- Compare to threshold values
- Most appropriate for pooled data
- with barcoded data more information is available

individual barcodes

```
acacagatagacatagacatagacagatgag 4
acacagatagacatagacatagacagatgag 3
acacacatagacatagacatagacagatgag 6
acacagatagacatagacatagacagatgag 4
acacagatagacatatacatagacagatgag 2
acacagatagacatatacatagacagatgag 2
acacagatagacatatacatagacagttgag 1
acacagatagacatagacatagacagatgag 6
acacagatagacatatacatagacagatgag 1
acacagatagacatagacatagacagatgag 6
```

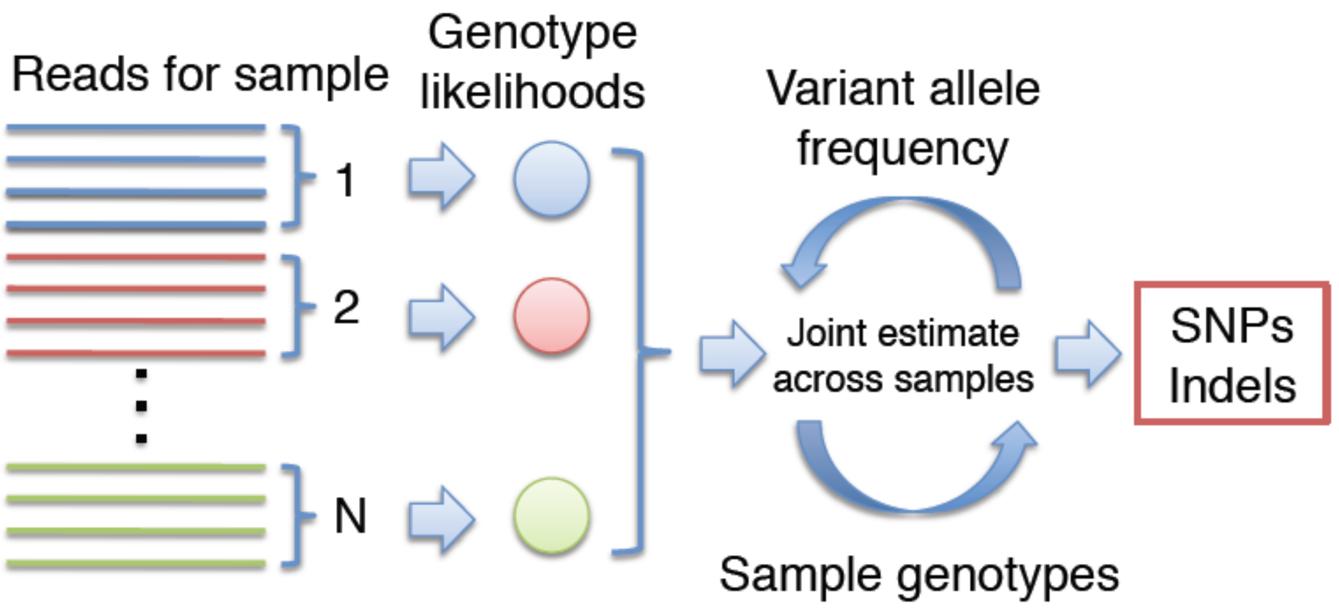
Bayesian methods

- Assign calls to specific genotypes
- Probability of genotype given data
- Makes better use of all data
- More appropriate with individual barcodes

population-based calling

- Variants at high frequency are more likely real
- Weak single sample calls are combined to discover variants among samples with high confidence
- "haplotype aware" calling also possible
  - infers haplotypes
  - uses info to impute variants

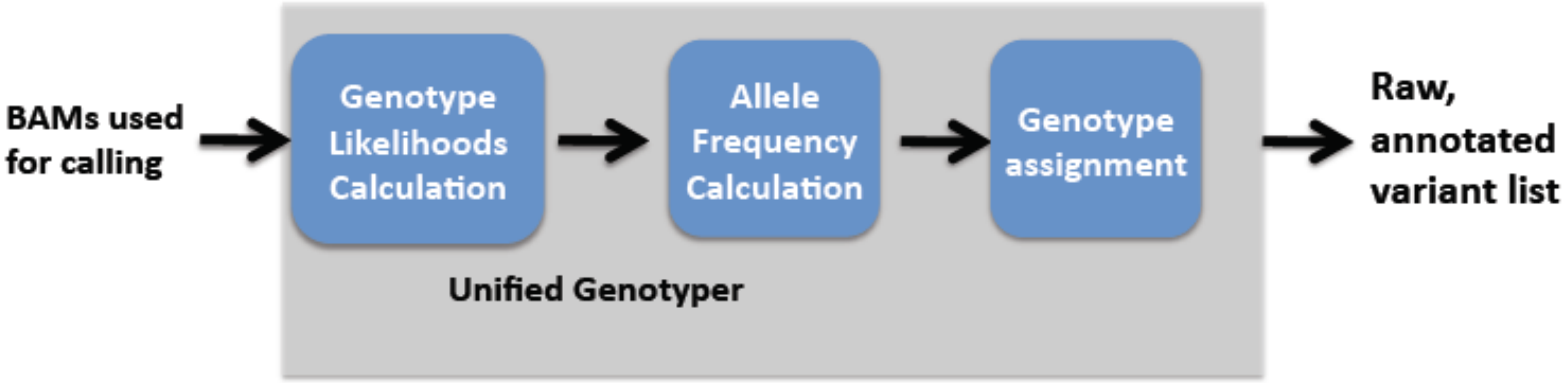
population-based calling



Simultaneous estimation of:

- Allele frequency (AF) spectrum: Pr{AF = i | D}
- The prob. that a variant exists: Pr{AF > 0 | D}
- Assignment of genotypes to each sample

GATK unified genotyper


BAMs used for calling
Genotype Likelihoods Calculation
Allele Frequency Calculation
Genotype assignment
Raw, annotated variant list
Unified Genotyper

Variant calling is multi-sample aware, and it incorporates three major steps:
• Computing, for each sample, for each genotype, likelihoods of data given genotypes.
• Computing, the allele frequency distribution to determine most likely allele count, and emit a variant call if determined.
• If a variant is emitted, assign a genotype to each sample.
– Genotypes can be further refined by haplotype based imputation
(e.g using Beagle, Mach, fastPHASE).

VCF format

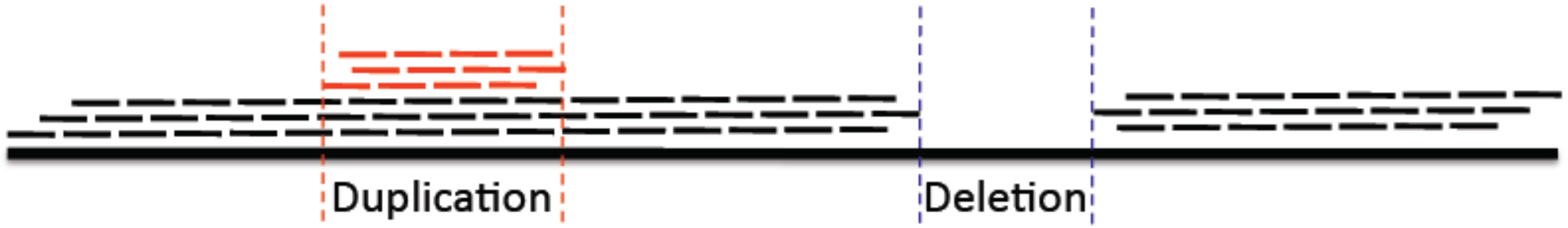
```
##fileformat=VCFv4.0 ##fileDate=20090805
##source=myImputationProgramV3.1
##reference=1000GenomesPilot-NCBI36
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=.,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT NA00001 NA00002 NA00003
20 14370 rs6054257 G A 29 PASS NS=3;DP=14;AF=0.5;DB;H2 GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51
1/1:43:5:.,.
20 17330 . T A 3 q10 NS=3;DP=11;AF=0.017 GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3 0/0:41:3
20 1110696 rs6040355 A G,T 67 PASS NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27
2|1:2:0:18,2 2/2:35:4
20 1230237 . T . 47 PASS NS=3;DP=13;AA=T GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20 1234567 microsat1 GTCT G,GTACT 50 PASS NS=3;DP=9;AA=G GT:GQ:DP 0/1:35:4 0/2:17:2 1/1:40:3
```

Discovery of structural variants

1) Read depth can identify copy number changes
2) Paired end spacing can mark regions of insertion, deletion or rearrangement
3) Long reads can be aligned at multiple places (split-read alignment) to find breakpoints
4) De novo assembly (global or local) can find novel insertions and define breakpoints

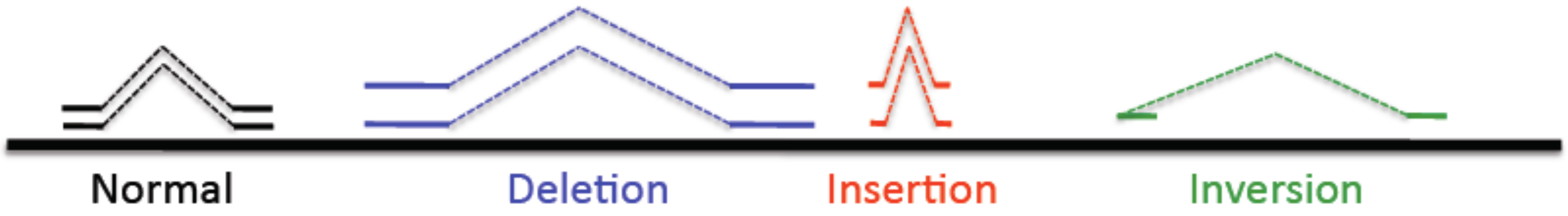
1) Read depth analysis

- Depth of coverage can be used to estimate copy number
- Samples may exhibit variation in depth indicative of polymorphic copy number variants
- How many copies of a duplication in the reference?
- How similar are the copies
- Difficult to distinguish homozygotes and heterozygotes.



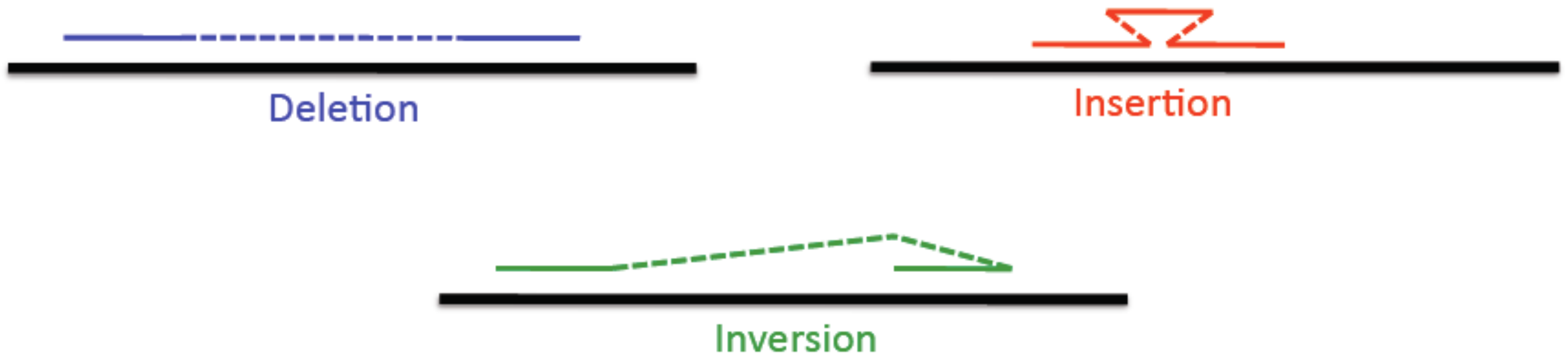
2) Paired end analysis

- Paired ends have a fixed length between them
- Genomic rearrangements cause them to vary
  - Deletion: reads will map too far apart
  - Insertion: reads will map too close
  - Inversion: reads in wrong orientation
- more reliable with long pairs



3) Split-read alignments

- Base-level breakpoint resolution
- Only works with long reads
  - short reads have many spurious splits
- Caveat: breakpoints may be duplicated
  - reads won't split if single alignment is good



4) *De novo* assembly to identify structural variants

- Assemble contigs
- Align to reference
- Look for insertions, deletions, rearrangements
- Use paired ends to identify potential events

Chosing names for files

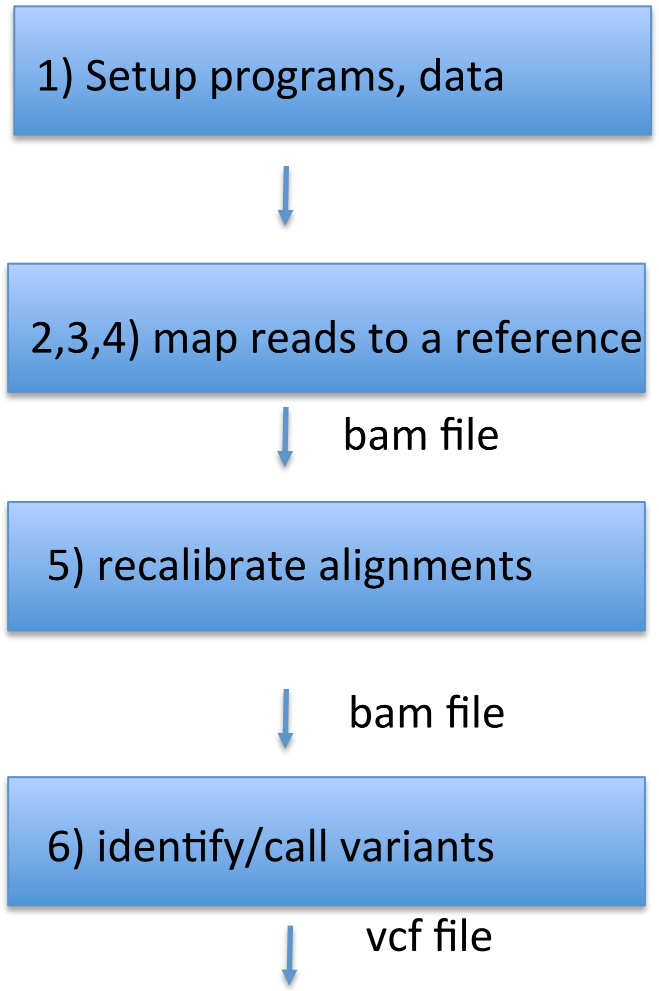
By using standardized names for your files it becomes easier to keep track of the different steps

- Initial file name according to some relevant information about the contents e.g. NA06984.ILLUMINA.low_coverage.17q
- For each step of the pipeline, create a new file with an extension that state what you have been doing, e.g.
  - NA06984.ILLUMINA.low_coverage.17q.merge.bam
  - NA06984.ILLUMINA.low_coverage.17q.merge.realign.bam
  - NA06984.ILLUMINA.low_coverage.17q.merge.realign.dedup.bam
  - NA06984.ILLUMINA.low_coverage.17q.merge.realign.dedup.recal.bam

Overview of excercise

1. Access to data and programs
2. Mapping (BWA)
3. Merging alignments (BWA)
4. Creating BAM files (Picard)
5. Processing files (GATK)
6. Variant calling and filtering (GATK)
7. Viewing data (IGV)

Steps in resequencing


1) Setup programs, data
2,3,4) map reads to a reference
bam file
5) recalibrate alignments
bam file
6) identify/call variants
vcf file

find best placement of reads

realign indels
remove duplicates
recalibrate base quality

statistical algorithms
to detect true variants

1) Access to data and programs

- Data comes from 1000 genomes pilot project
  - 81 low coverage (2-4 x) Illumina WGS samples
  - 63 Illumina exomes
  - 15 low coverage 454
- ~ 1 Mb from chromosome 17

- Tasks: align one sample to reference, process, reacalibration, variant calling and filtering

1) Access to data and programs

- Programs:
- BWA and samtools modules can be loaded:

```
module load bioinfo-tools
module load bwa
module load samtools
```

- picard and GATK are are set of java programs:

```
/bubo/sw/apps/bioinfo/GATK/1.5.21/
/bubo/sw/apps/bioinfo/picard/1.69/kalkyl/
```

2) Mapping

- Indexing sequences:
- perform BW transform on reference

```
bwa index -a bwtsw <reference sequence>
```

- make a fasta index for reference

```
samtools faidx <reference sequence>
```

- align each paired end separately

```
bwa aln <reference> <seq 1> >align1.sai
bwa aln <reference> <seq 2> >align2.sai
```

3) Merging alignments

- command to combine alignments from paired ends into a SAM file

```
bwa sampe <ref> <sai1> <sai2> <fq1> <fq2> >align.sam
```

<ref>	= reference sequence
<sai1>	= alignment of seq 1 of pair
<sai2>	= alignment of seq 2 of pair
<fq1>	= fastq reads seq 1 of pair
<fq2>	= fastq reads seq 2 of pair

4) Creating and editing BAM files

- create BAM file (samtools)

```
samtools import <reference.fai> <sam file> <bam file>
```

- sort BAM file (samtools)

```
samtools sort <input bam> <output bam rootname>
```

- index BAM file (samtools)

```
samtools index <input bam>
```

- add read groups to BAM (picard)

```
java -Xmx2g –jar /path/AddOrReplaceReadGroups.jar
INPUT=<sam file>
OUTPUT=<bam file>
... more options
```

- index new BAM file (picard)

```
java -Xmx2g –jar /path/BuildBamIndex.jar
INPUT=<bam file>
... more options
```

5) Processing files

- mark problematic indels (GATK)

```
java -Xmx2g -jar /path/GenomeAnalysisTK.jar
-I <bam file>
-R <ref file>
-T RealignerTargetCreator
-o <intervals file>
```

- realign around indels (GATK)

```
java -Xmx2g -jar /path/GenomeAnalysisTK.jar
-I <bam file>
-R <ref file>
-T IndelRealigner
-o <realigned bam>
-targetIntervals <intervals file>
```

5) Processing files

- mark duplicates (picard)

```
java -Xmx2g -jar /path/MarkDuplicates.jar
INPUT=<input bam>
OUTPUT=<marked bam>
METRICS_FILE=<metrics file>
```

- quality recalibration - compute covariation (GATK)

```
java -Xmx2g -jar /path/GenomeAnalysisTK.jar
-T CountCovariates
-I <input bam>
-R <ref file>
-knownSites <vcf file>
-cov ReadGroupCovariate
-cov CycleCovariate
-cov DinucCovariate
-cov QualityScoreCovariate
-recalFile <calibration csv>
```

5) Processing files

NEXT:

repeat steps 2-5 for another sample!

5) Processing files

- merge BAM from multiple samples (picard)

```
java -Xmx2g -jar /path/MergeSamFiles.jar
INPUT=<input bam 1> INPUT=<input bam 2> .. INPUT=<input bam N>
OUTPUT=<output bam>
```

- unified genotyper (GATK)

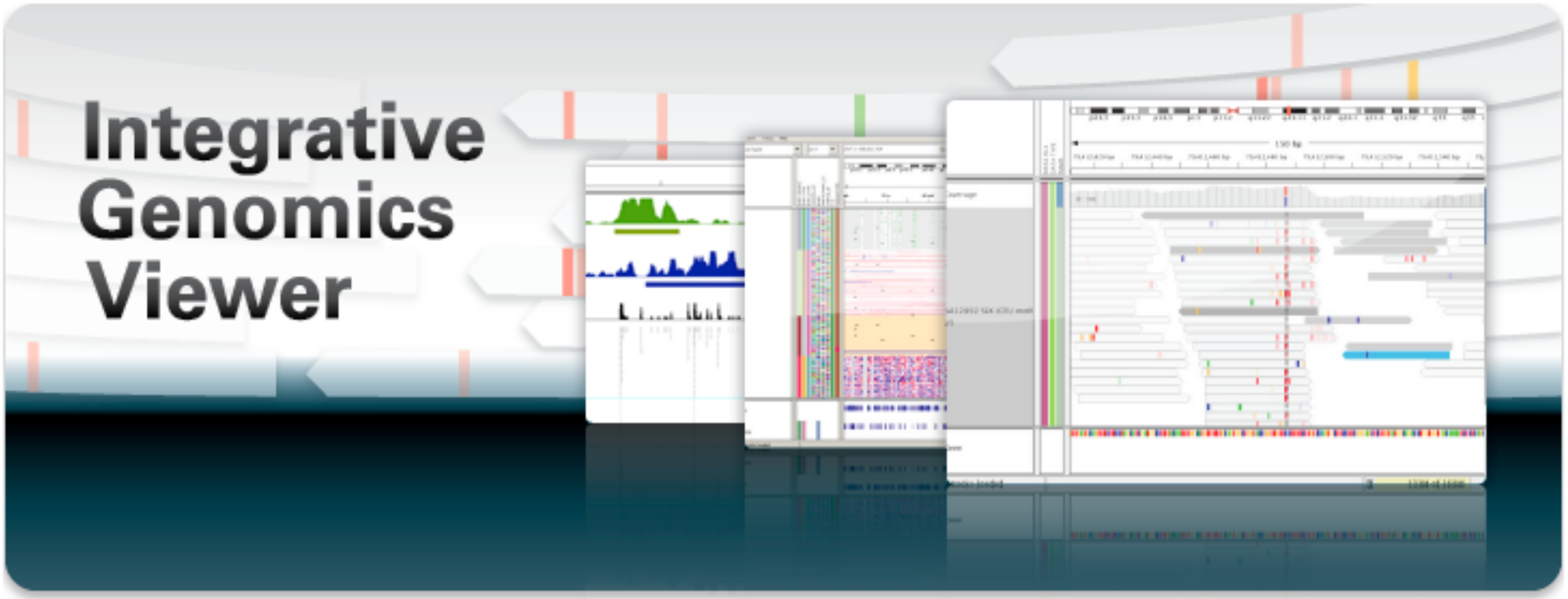
```
java -Xmx2g -jar /path/GenomeAnalysisTK.jar
-T UnifiedGenotyper
-R <ref file>
-I <merged bam>
-o <filename.vcf>
-glm BOTH
```

6) Variant calling and filtering

- variant filtering

```
java -Xmx2g -jar /path/GenomeAnalysisTK.jar
-T VariantFiltration
-R <reference>
-V <input vcf>
-o <output vcf>
--filterExpression "QD<2.0" --filterName QDfilter
--filterExpression "MQ<40.0" --filterName MQfilter
--filterExpression "FS>60.0" --filterName FSfilter
--filterExpression "HaplotypeScore>13.0" --filterName HSfilter
--filterExpression "MQRankSum<-12.5" --filterName MQRSfilter
--filterExpression "ReadPosRankSum<-8.0" --filterName RPRSfilter
```

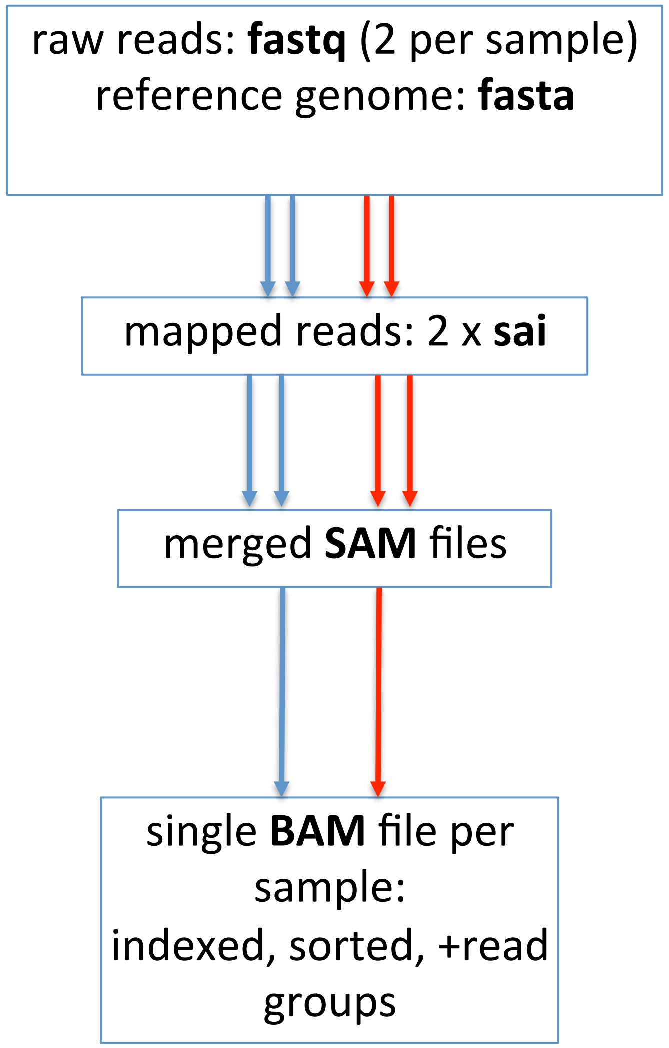
7) Viewing data with IGV



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

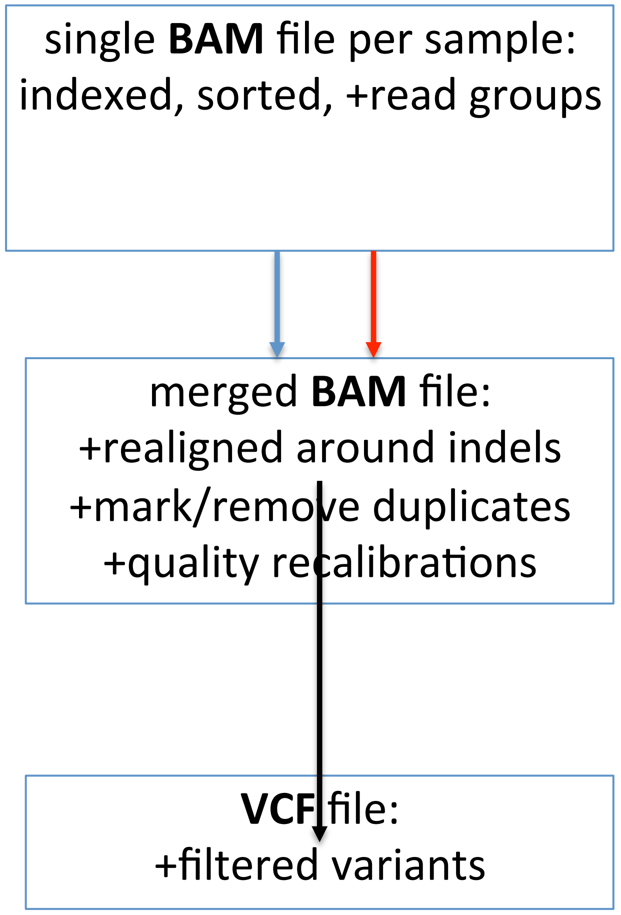
pipeline (1)

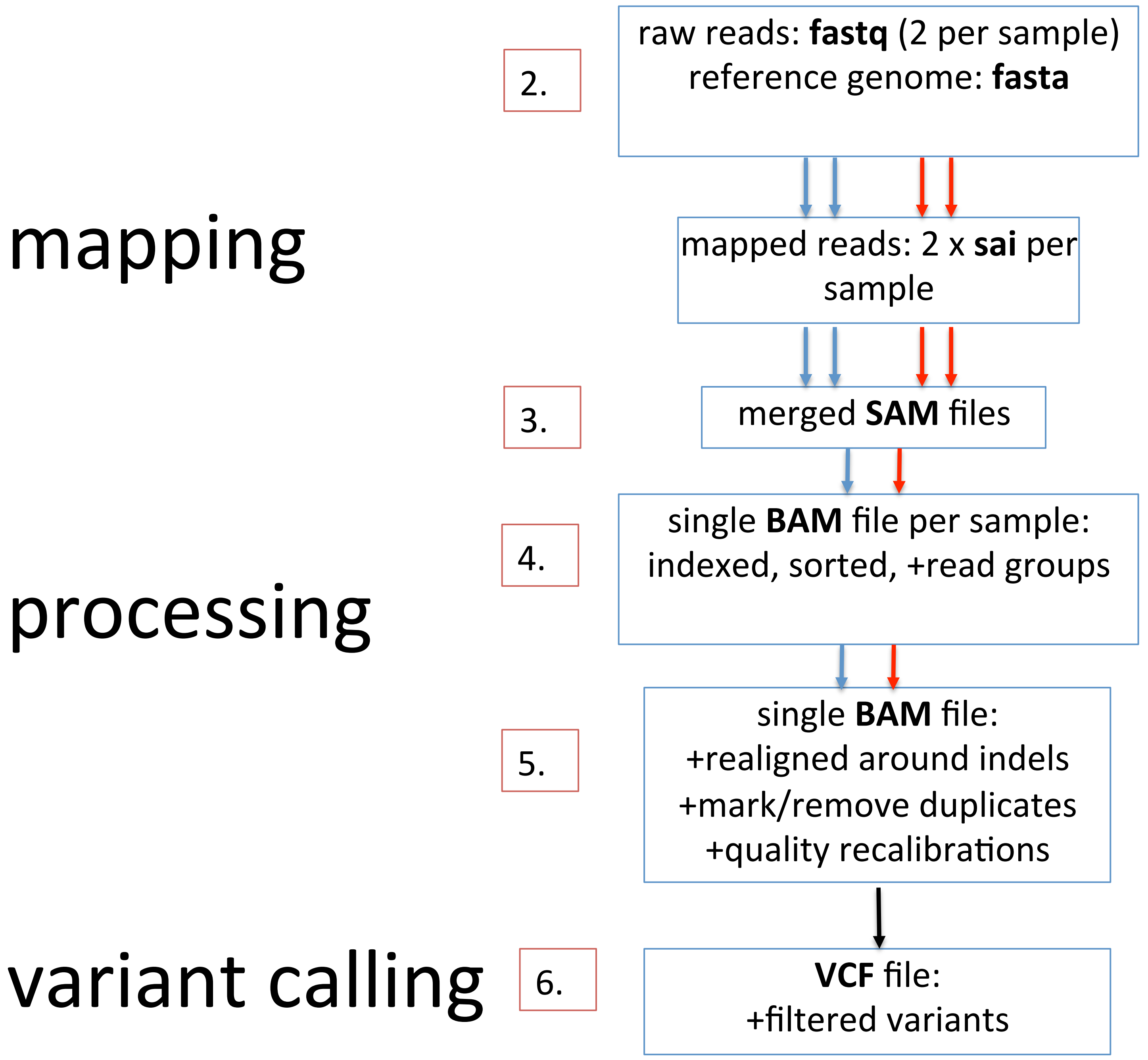
2. Mapping
   - `bwa index`
   - `samtools faidx`
   - `bwa aln`
3. Merging alignments
   - `bwa sampe`
4. Creating BAM files
   - `samtools import`
   - `samtools index`
   - `samtools sort`
   - `picard AddOrReplaceReadGroups`
   - `picard BuildBamIndex`


raw reads: **fastq** (2 per sample)
reference genome: **fasta**
mapped reads: 2 x **sai**
merged **SAM** files
single **BAM** file per sample: indexed, sorted, +read groups

pipeline (2)

5. Processing files (GATK)
   - `GATK RealignerTargetCreator`
   - `GATK IndelRealigner`
   - `picard MarkDuplicates`
   - `GATK CountCovariates`
   - `picard MergeSamFiles`
6. Variant calling and filtering (GATK)
   - `GATK UnifiedGenotyper`
   - `GATK VariantFiltration`
7. Viewing data (IGV)




mapping
processing
variant calling
2.
raw reads: **fastq** (2 per sample)
reference genome: **fasta**
mapped reads: 2 x **sai** per sample
3.
merged **SAM** files
4.
single **BAM** file per sample: indexed, sorted, +read groups
5.
single **BAM** file:
+realigned around indels
+mark/remove duplicates
+quality recalibrations
6.
VCF file:
+filtered variants

Thanks!

+ this presentation was made by Matt Webster
+ special thanks to Mike Zody for some slides