Variant Calling
(using High-throughput Sequencing Data)

October 2015

ABOUT THE COURSE

INF-BIOX121

- The tough challenges:
  - High Throughput Sequencing
  - Biological Applications
  - Informatics
- What is the secret for 18 hours with me
  - to be fun for you?
  - to be fun for me?
  - you to learn as much as possible?
- An attitude that works well in all walks of life
  - take initiative for problem solving
  - then ask for help

Who is teaching?

- People
  - Tim – bioinformatics – HTS since 2009
  - Even and Srinidhi – both took the course last year
- What we know about HTS
- I have run this course many times
  - its going not going to be easy
  - its probably going to be messy
  - but you will go away having learnt a lot, we hope ☺

Our goal for the participants

- Finish the course with an understanding of all major concepts
- Know how to run a variant calling pipeline
- Ready to make informed choice about whether you may need a simpler or more complex pipeline
- Focus is on Variant Calling and functional annotation
  - Previous courses have expressed need for more downstream analysis
  - So we have included more of this
- You might experience some overlap with other modules of the course, but repetition can be beneficial

Themes running across the different modules

- Technical / bioinformatic
  - Study design
  - Mapping principles
  - Alignment principles
  - Alternative splicing
  - Model system vs non-model system organisms

- The bioinformatic “process”
  - Reproducibility
  - Best practices
  - Testing
  - Documenting
  - Statistics and hypothesis testing
  - Summary statistics and visualisation
  - Sanity checking / validation of results
  - Finding data and munging it incl. public databases

Biologist advice to a biologist

- 1st advice: DON'T PANIC
  2nd advice: DON'T PANIC
  3rd advice: DON'T

- Do the unix part of tutorial of Unix and perl primer for biologists: http://korflab.ucdavis.edu/Unix_and_Perl/ , alternatively:
  - http://linuxcommand.org/learning_the_shell.php#contents
  - http://www.tuxfiles.org/linuxhelp/cli.html

Yyou are not the first to have problems
  - search for it on the net
  - or, if you can't find the answer, ask in a forum

- Learn to write a very simple script

- Make a list of your favourite commands as you learn them.

Often when something is difficult to understand it is often either because:

- It is not clear what is being done
- Or because it is not clear why it should be done

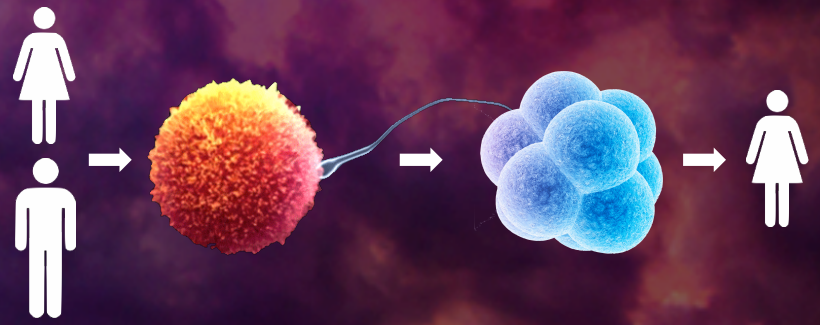
So stop me and ask if the WHAT or the WHY is unclear!!!

Some thoughts about scientific problems vs. techniques

- Always keep in mind what the basic goal is and what the basic issues are with attaining that goal.
- Learn to switch from the broad picture to the details and back again.
- In my opinion it is mistake to focus too much on specific techniques. Become good at different techniques, but learn them because they help you solve problems.
- Don't go looking for things you can apply a technique to or at least be careful to not do only this.
    - Antony van Leeuwenhoek is a good counter-example

An exception to not focussing on techniques
Antony van Leeuwenhoek (1632-1723)

Humans and other multicellular organisms



A reproducing system which is:

* Self-assembling
* Self-repairing
* Self-operating
* Self-upgrading
* Self-aware
* Social and spiritual

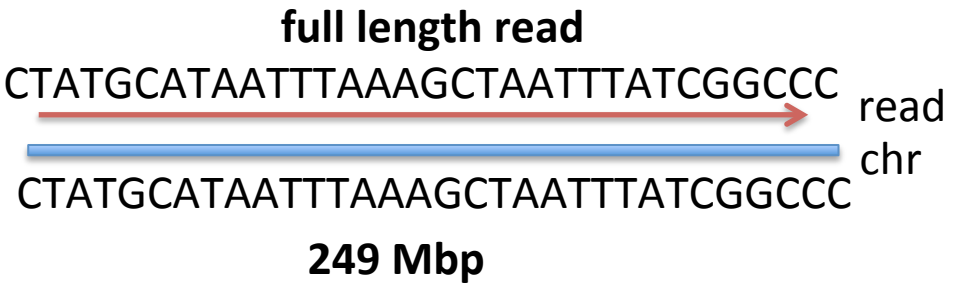
The full information underlying this system is stored in the DNA of EVERY cell

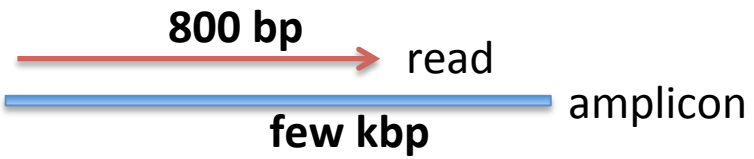
So **why** variant calling?

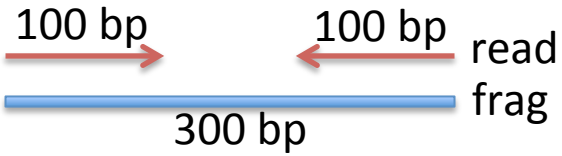
- To known the DNA sequence
- Why is the DNA sequence of interest?
  - Because it is the genetic blueprint, driving a large fraction of the higher level phenotypes (protein, network, cell, organ, behaviour, disease)
  - Because it is product of evolution and thus contains information:
    - about the evolutionary process
    - about the populations which have evolved

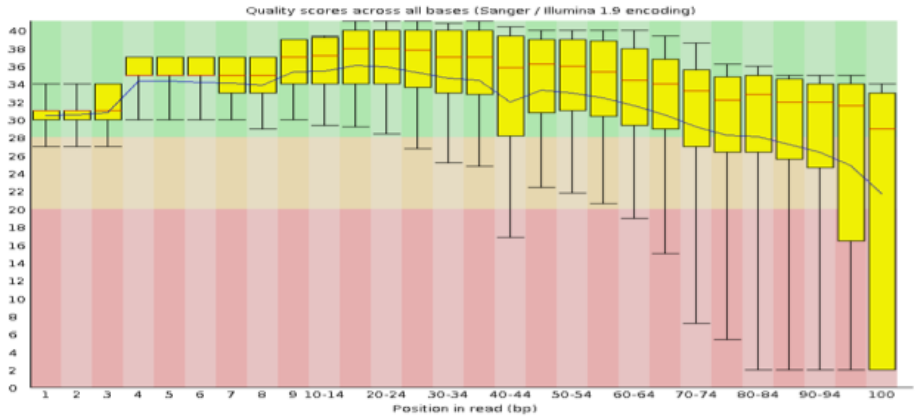
In a perfect world – Perfect sequencing

- Perfect sequencing:
  - single molecule (no PCR)
  - **full** length
  - no deterioration of quality
- The real world has improved but is not perfect:
  - Sanger
    - PCR
    - length: some kb
    - limited number of reads
    - high quality
  - HTS (Illumina)
    - PCR
    - 100-300 bp
    - Paired-end option (PE)
    - billions of reads
    - high quality, but deteriorating along read









Looking for a ping pong ball in a stadium

- With a hand torch!!

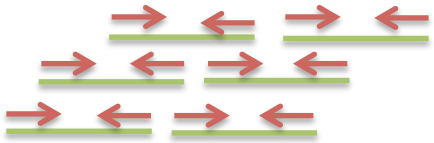
Or with the lights on?

A quick overview of the HTS workflow

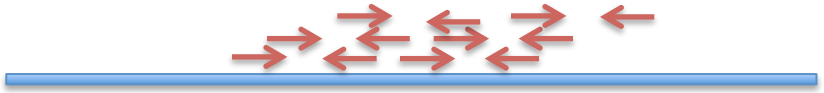
Fragment sample

Capture

Sequence

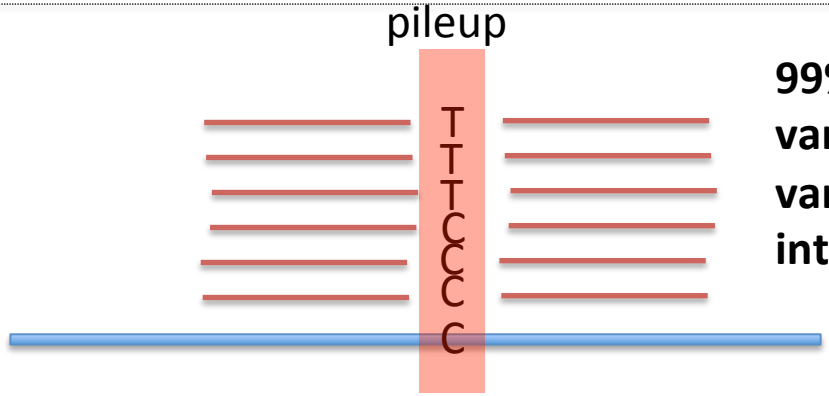


Map



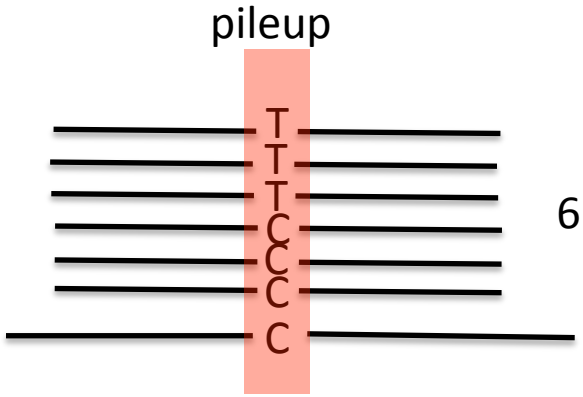
Align

Variant call



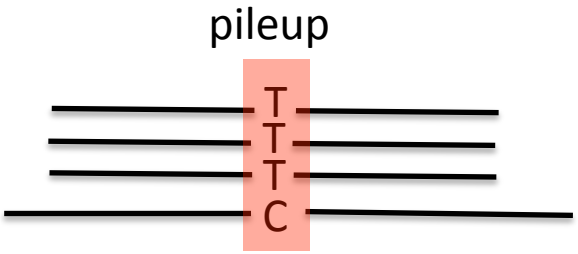
99% of sites are NOT variant, but it is the variant sites that are interesting

Variant sites



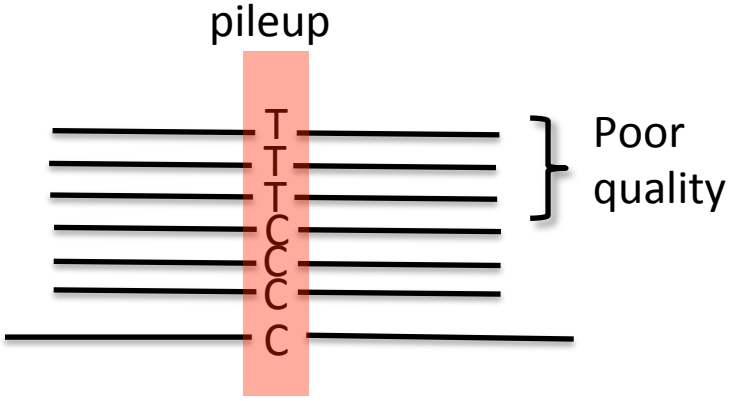
The common and easy case

- Good mapping of reads
- Good base qualities
- Good depth



Poor depth

- May not have sampled both alleles
- Could be C/T or T/T



Poor quality

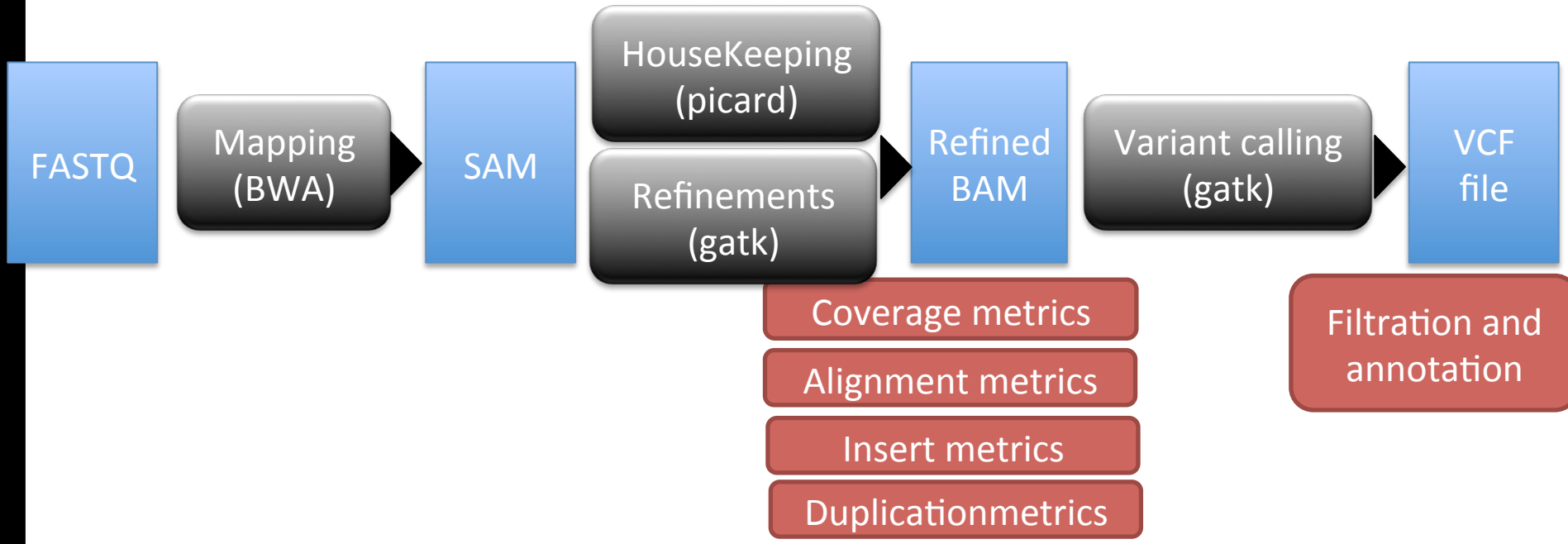
- of base calls
- of read mapping

Can lead to false variant calls: site could be ref

C/C and Ts are just base call errors

Our pedagogical approach

Multiple iterations with increasing complexity

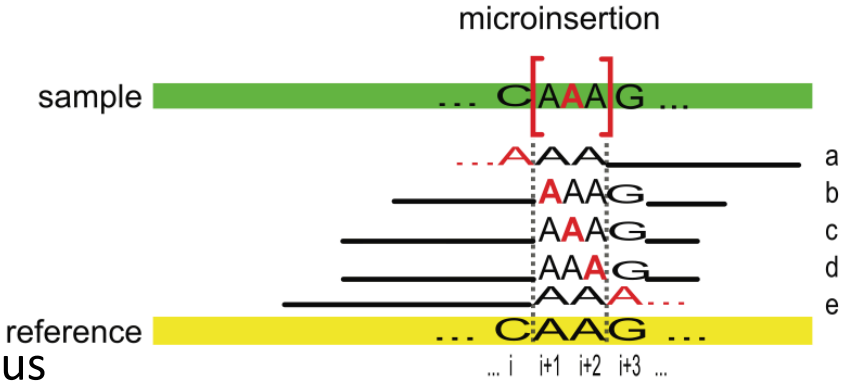


What do we aim to achieve with this approach

- Hoped for advantages of approach:
  - simple to understand most basic pipeline and easy to run
  - practice in each iteration
- Don't worry if you do not understand everything at the first iteration
- With bioinformatic pipelines it is generally a good idea to quickly create a simple version before make it more complex
  - prototyping
  - Check out Tom Wujec's TED talk "**The marshmellow challenge**"
- We will focus on single sample analysis, and will discuss more complicated setups later
  - multi-sample
  - pooled

Why do we need a “complex” pipeline?

- The biggest problem with variant calling is large numbers of FPs and FNs:
  - Mainly driven by bad alignments
  - FNs often create FPs
  - Can be systematic across samples, thus creating consistent SNPs across samples that are just machine artifacts.



- FPs and FNs, may result in:
  - the signal in our data drowning in noise ➔ no result
  - false results ➔ erroneous result

- You may find that reviewers will object to a pipeline which is not state-of-the-art, which creates trouble for you in the review process eventhough you may actually have a result.

Choice of example dataset

- Human data – arbitrary

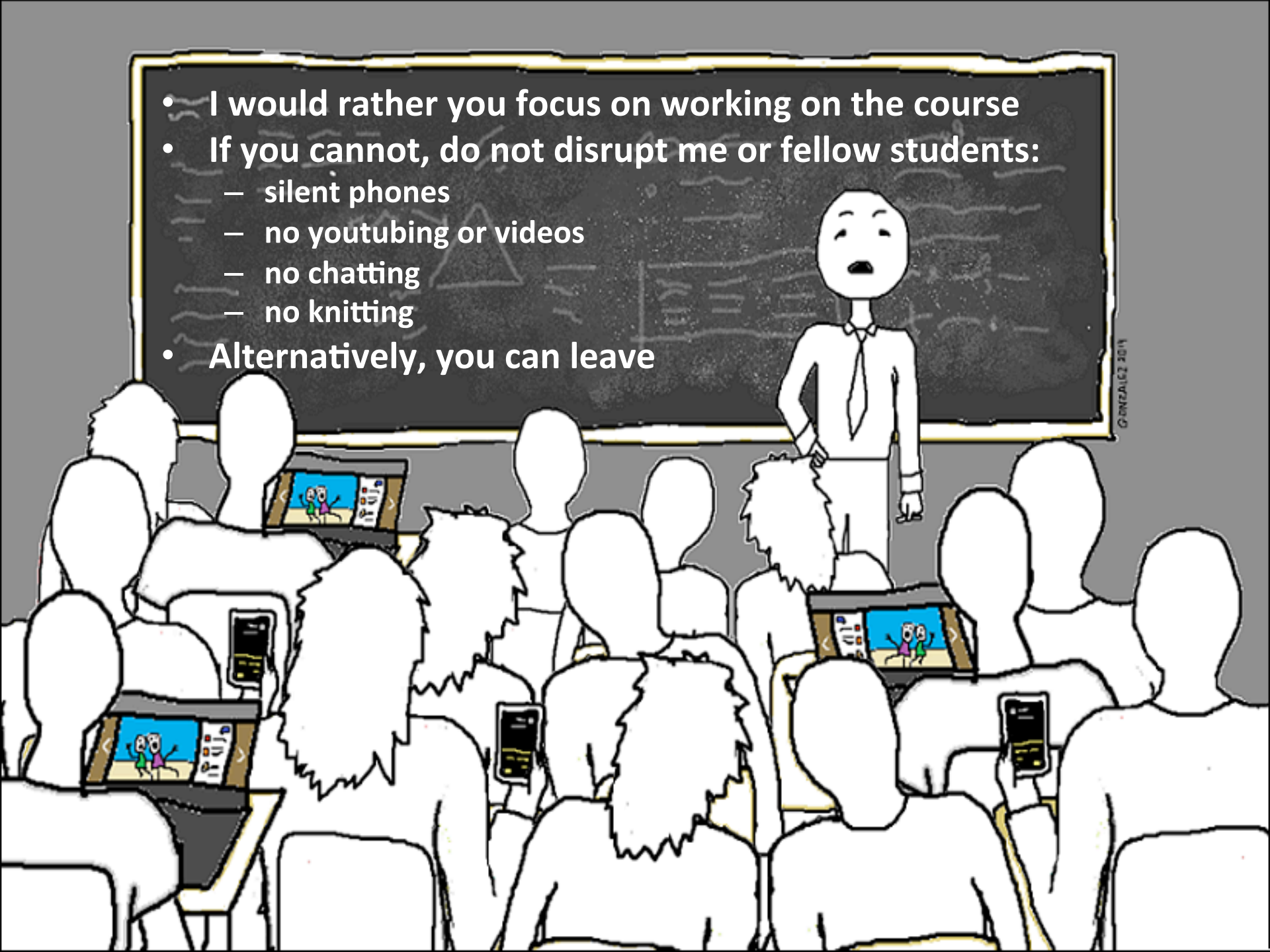
- What if you work on other species?
  - if you have a reference genome and reads from a sample you can run the most basic pipelines
  - For more advanced pipeline you may have to spend some effort finding or generating auxiliary files in correct format
    - dbSNP
    - Genome annotation
  - You may find that they are already available for many species
    - Check out the GSA/GATK website
    - Check out http://getsatisfaction.com/gsa

GSA team at the Broad Institute

- A large fraction of the materials and software in this course are produced by the **Genome Sequencing and Analysis Group** team at the Broad Institute
- Information sources
  - http://www.broadinstitute.org/gsa/wiki
  - http://www.getsatisfaction.com/gsa
- People
  - Mark A. DePristo, Manager of Medical and Population Genetics Analysis
  - Eric Banks, Team Lead
  - Guillermo del Angel
  - Ryan Poplin
  - Kiran Garimella, Team Lead
  - Mauricio Carneiro
  - Chris Hartl
  - Khalid Shakir, Team Lead
  - Matthew Hanna
  - David Roazen
- Others at the Broad
  - Heng Li: samtools and bwa
  - Tim Fennell: picard
  - Alec Wysoker: picard
  - Brac Chapman: bcbio.variation
- And others outside the Broad
  - sources at bottom of slides

For each practical

- Most key concepts explained up front
  - sometimes concepts will be first introduced in the practical (this will be the exception)
- Run through of the practical in plenum
- Read carefully through the practicals. All the information you need for doing the practical should be in the file
- Practical session
  - Try to get it running
  - If fail or computation takes time ➔ go to exercisesResults directory
- You will be doing “copy and paste”
  - that is OK
  - but you must understand what you are copying and pasting
  - if you don’t ask for explanations
- Use of labels to indicate status: “finished” and “needing help”
- It is hard!
  - new concepts
  - new environment (Unix)
  - try to understand both **and take notes!**


• I would rather you focus on working on the course
• If you cannot, do not disrupt me or fellow students:
– silent phones
– no youtubing or videos
– no chatting
– no knitting
• Alternatively, you can leave

Other dos and donts

- Do
  - Sign the attendance sheet
  - Remember that there are wide differences in competance
  - Consolidate your knowledge if you get ahead
  - Talk and get help from your neighbours

- Do **not** ask questions in plenum if you get ahead, instead consolidate what you have learnt (tool websites)

Unix attitude

- **Get used to having to know where you are and to tell the computer where things are**

```
java -Xmx2G -jar ${swDir}/picard-tools-1.67/MarkDuplicates.jar \
INPUT=aln.posiSrt.clean.bam \
OUTPUT=aln.posiSrt.clean.dedup.bam \
METRICS_FILE=aln.posiSrt.clean.dedup.bam_metrics.duplication.txt
```

- **Be as precise as you are in your field**

../vc/exerSandbox/03_advancedPipeline

- **Pay attention to detail and changes**

```
-rw-rw-r-- 1 timothyh timothyh  26M Oct  2 14:30 aln.posiSrt.clean.dedup.bam
-rw-rw-r-- 1 timothyh timothyh 145K Oct  2 14:30 aln.posiSrt.clean.dedup.bai
```

- **String commands together if you can**

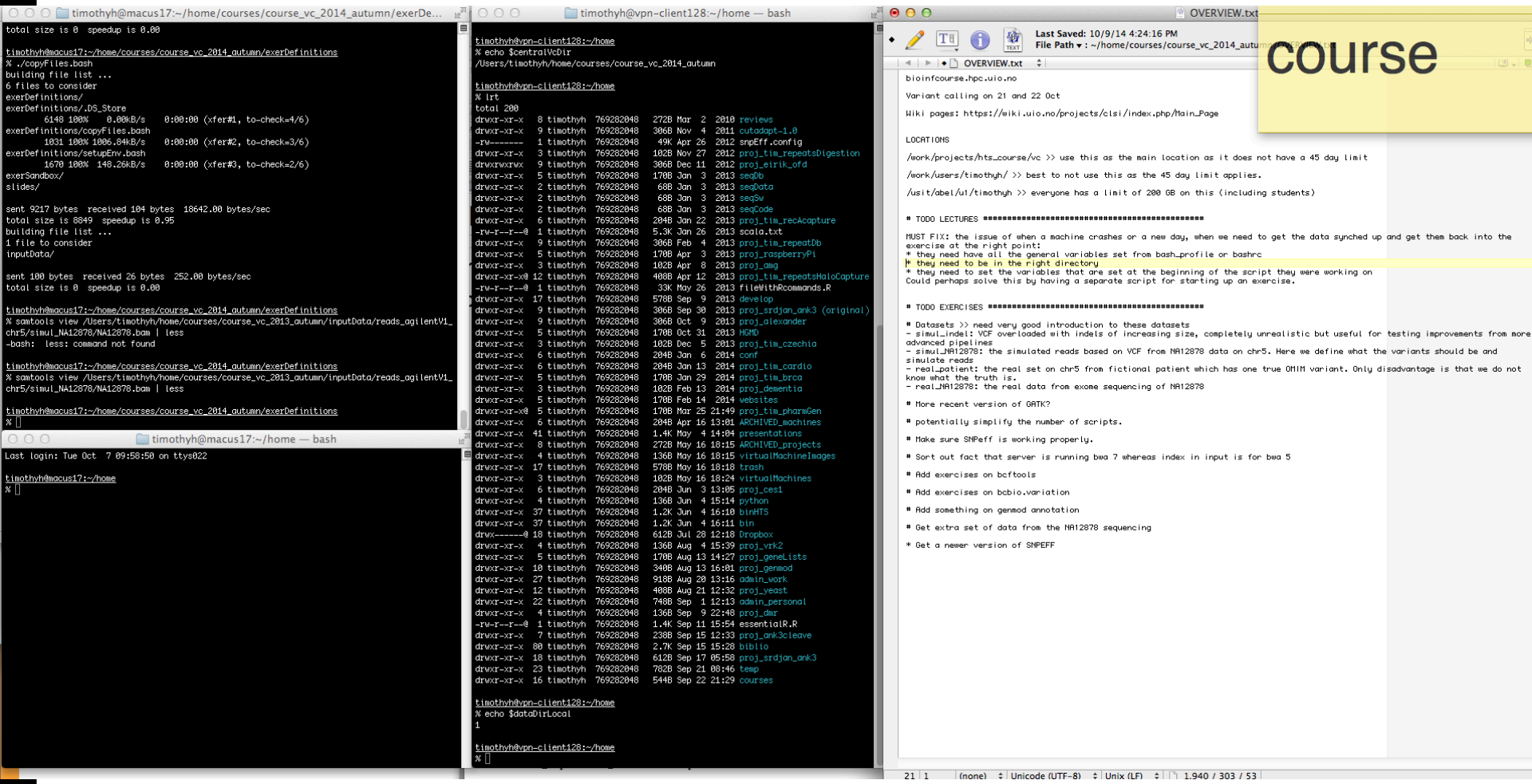
```
cat ${dataDir}/reads_agilentV1_chr5/simul_indels/deletions.vcf | grep -v "^#" | wc -l
```

Some useful unix shortcuts

- Ctrl + a: go to beginning of line
- Ctrl + e: go to end of line
- Ctrl + k: remove text after pointer
- Ctrl + u: remove text before pointer
- Arrow up and down to scroll through history
- Ctrl + r: reverse search history

Organising your desktop

- Keep an electronic notepad of file paths
- Useful to have multiple terminals



SOFTWARE AND DATASET – SETUP AND INTRODUCTION

Software and data installation – mini practical

- Use the terminal or a file explorer to take a look at the central repository
  - **exerDefinitions** contains all the practicals
  - **exerResults** contains pre-computed results of practicals
  - **inputData** contains all the datasets we will be working on
- Software
  - for documentation of how the tools work for a lot of commands you get a quick help by typing the following at the command line: **command** or **command –h**
  - **If this does not work, the best is to google the tool and read the online documentation.**
- Lets take a quick look at the data >> mini practical (see next slide)

Introduction of datasets in **inputData**

- reads_exomeCapt_chr5 in fastq format (**reads_agilentV1_chr5**)
  - real reads from exome capture (**real_patient**)
  - simulated: known mutations (mainly indels) and simulated reads (**simul_indels**) – same regions as real dataset
  - simulated: known mutations (mainly indels) and simulated reads (**simul_NA12878**) – same regions as real dataset

- reference data (**human_g1k_v37_chr5**)
  - **agilentV1** >> definition of capture tiles in different formats
  - **gatkBundle** >> reference data in fasta format and vcf files of known variants (dbSNP, 1000 genomes, hapmap)

- Formats >> we will return to these later

The reference genome

Or the species of your choice.

Some kind of reference is necessary for variant calling.

It is possible to use closely related species as a reference.

Naming and ordering of chromosome/contigs

	Hg18 (UCSC)	B36 (NCBI)
Contig prefix	chr	none
Mitochondrial contig	chrM	MT
Contig order	chrM, chr1, chr2, ..., chrX, chrY	1, 2, ..., X, Y, MT

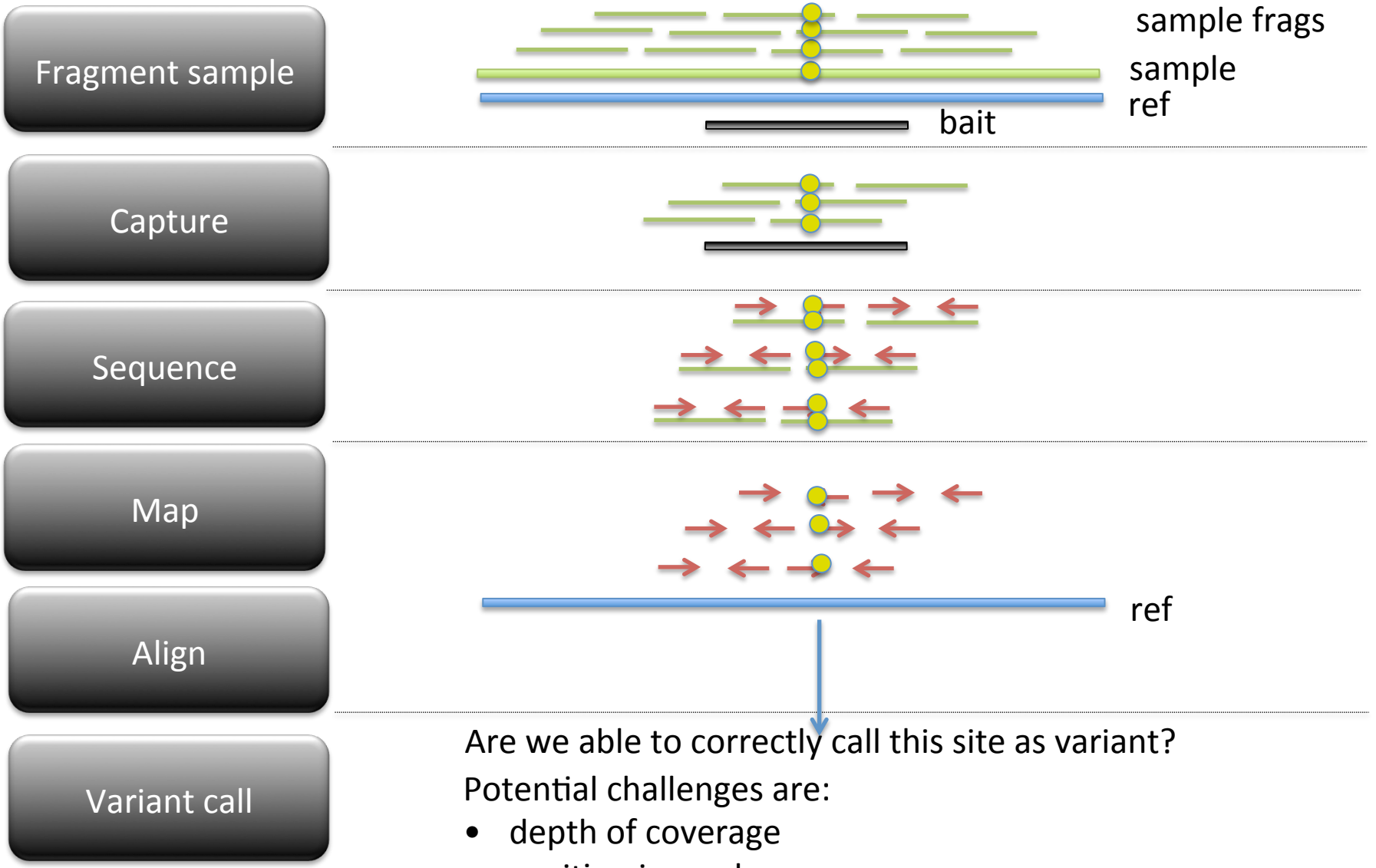
- Genome references
  - Fasta file: must have .fasta extension + respect naming and order
  - Fai file (created by samtools faidx): contig, size, location, basesPerLine ➔ for efficient random access
  - Dict file (created by Picard CreateSequenceDictionary): SAM style header describing the contents of the fasta file ➔ for names and length of original file
- ROD (reference ordered data)
  - GATK supports several common file formats for reading ROD data: VCF, UCSC formatted dbSNP, BED
- dbSNP files
  - Must also be ROD
  - Generated by GSA from the dbSNP db using a bit of bash, awk and a perl script: sortByRef.pl. Full details: http://www.broadinstitute.org/gsa/wiki/index.php/The_DBSNP_rod
- All of the above delivered for human as part of the **GATK resource bundle**
  - Other species may also be available
  - Help on generating for another species see GATK wiki or getsatisfaction.com/gsa

The FASTQ reads: Motivating the simulation

- Why simulate in variant calling?
  - with real data we do not know the "truth"
  - in a simulation, we can define the "truth" and then "hide" it and test how well our methods can recover this "truth"
  - in a real dataset, most variants are single nucleotide polymorphisms and these are usually very easy to correctly call

- Drawbacks of simulation: usually will not accurately model all aspects of real data

The simulation



The simulated dataset

Placed in true agilent tiles on chr 5

- SNPs – 100 of each type >> 600 in total
  - 3 strands (1 hetero, 2 hetero, 3 homo)
  - 2 positions in tile

Tiles: 1-600 SNPs
Position 0-14MB)

- Deletions – 5 of each type >> 1800 in total
  - 3 strands (1 hetero, 2 hetero, 3 homo)
  - 2 positions in tile (at edge or well inside)
  - size: 1 to 60

Tiles: 601-2400
Deletions - increasing in size in batches of 30 (14 to 72 MB)

- Insertions – 5 of each type >> 1800 in total
  - 3 strands (1 hetero, 2 hetero, 3 homo)
  - 2 positions in tile (at edge or well inside)
  - size: 1 to 60

Tiles: 2401-4200
Insertions - increasing in size in batches of 30 (72 to 131 MB)

- **A highly unrealistic set of variation but useful for studying the intricacies of variant calling**

Practical 01 – setting up your home directory

- We do this real SLOOOOOOOOW……
- We need to create a place where you can do the exercises in your home directory
- Start two terminal windows
- First setup the environment, you need:
  - in a terminal: **source /data/vc/exerDefinitions/setupEnv.bash**
  - **Nota Bene: you should always run this when you start a new terminal**
  - test it: **echo $localVcDir**
  - you should output something like **/usit/abel/u1/timothyh/vc**
- Get your copy of the files you need from the central location:
  - in a terminal: **/data/vc/exerDefinitions/copyFiles.bash**
  - The files were copied to this location: **echo $localVcDir**
  - Check out the contents of this directory: **ls $localVcDir**
  - You should see a list of files which are your copies of the inputData, slides and exercises (exerDefinitions) and a location where you should do all exercises (exerSandbox)

GENETICS 101

Basic concepts – Take notes

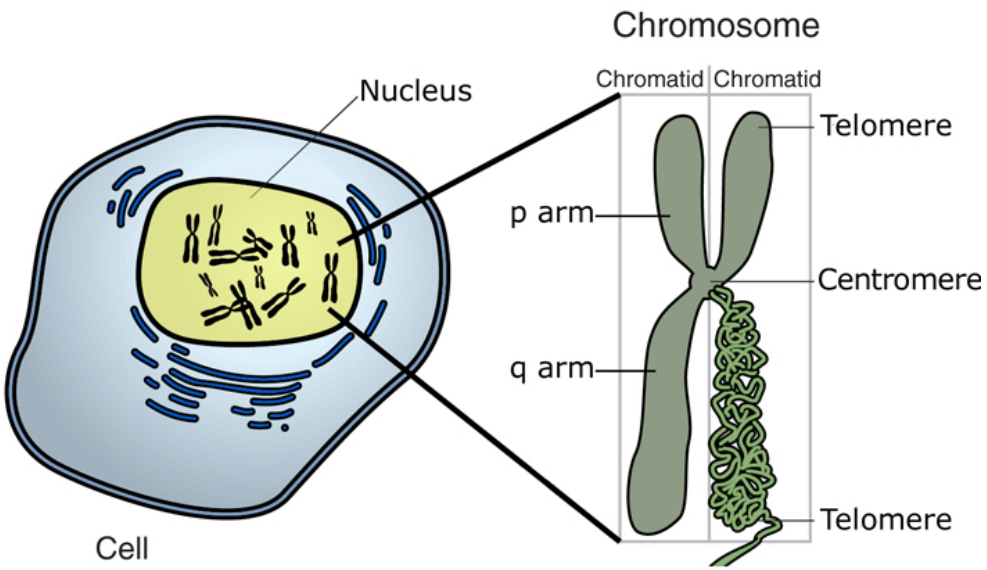
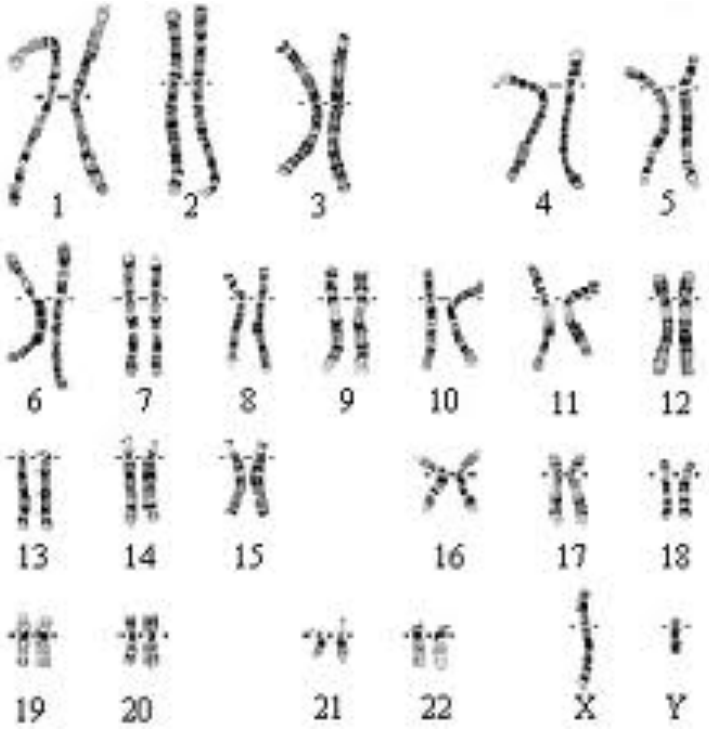
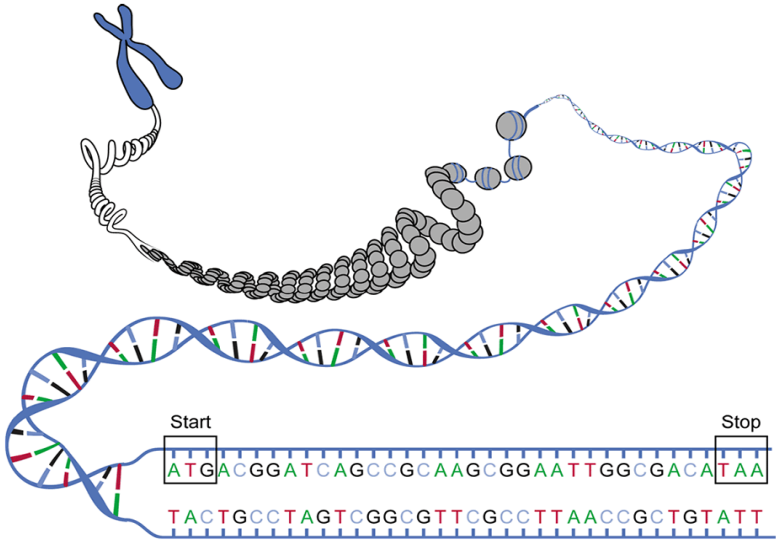


Image adapted from: National Human Genome Research Institute.

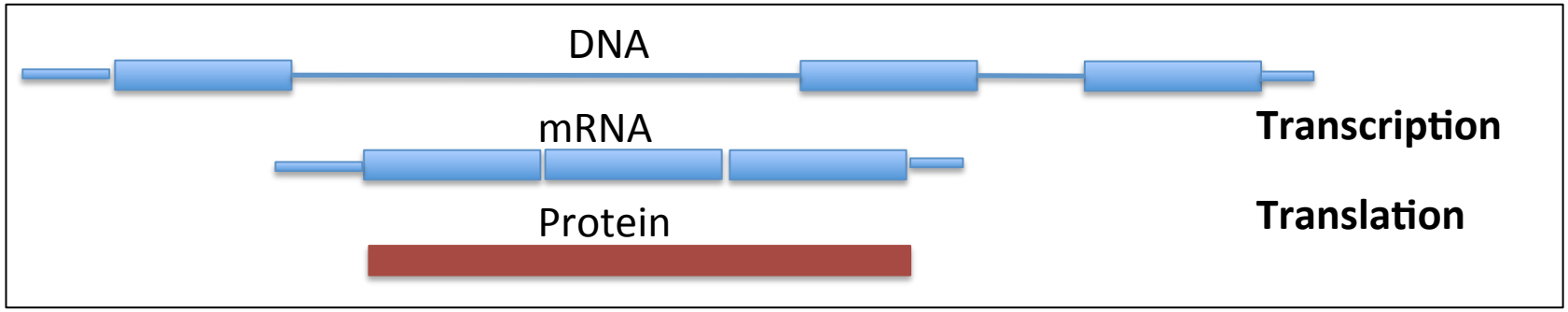




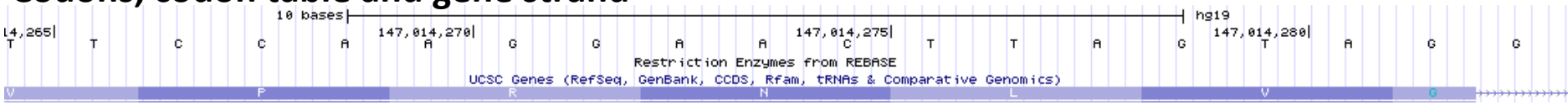
- Cells
- Chromosomes (ploidy)
- DNA complementarity and strands
- Variant site (SNPs are the most simple)
- Genotype: homozygote / heterozygote
- Haplotype

What is a gene?

Central Dogma

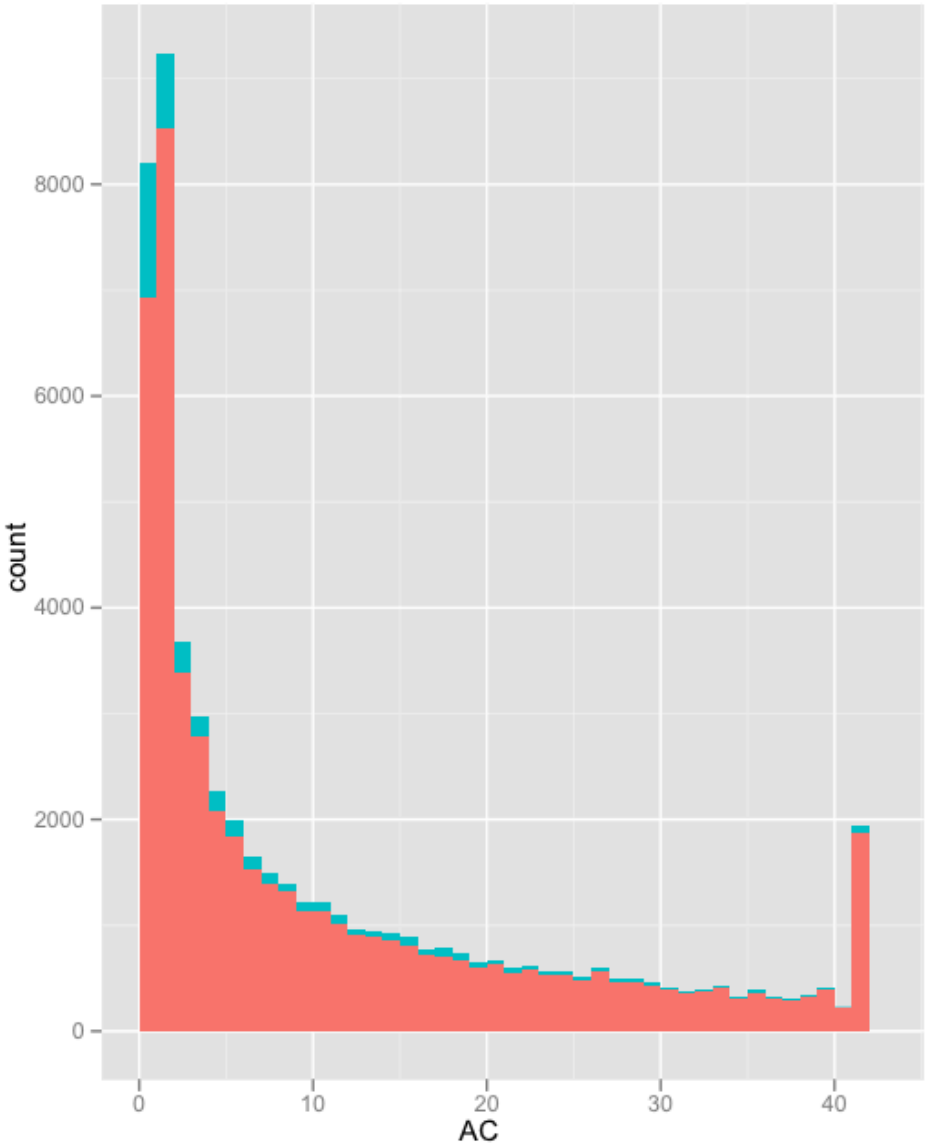


Codons, codon table and gene strand



- There are large numbers of genes in multicellular organisms
- Not all organism have multiple exons per gene and thus do not need splicing
- Outside genes are regulatory elements that control when and where genes are expressed

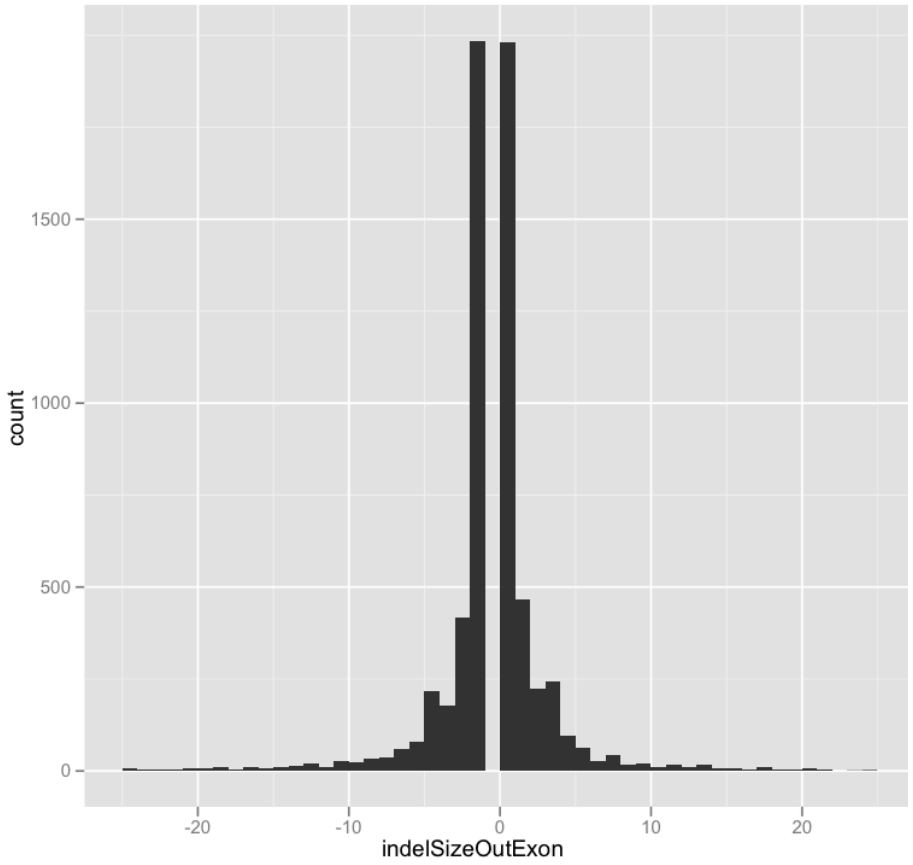
Distribution of Allele Count across 21 exomes

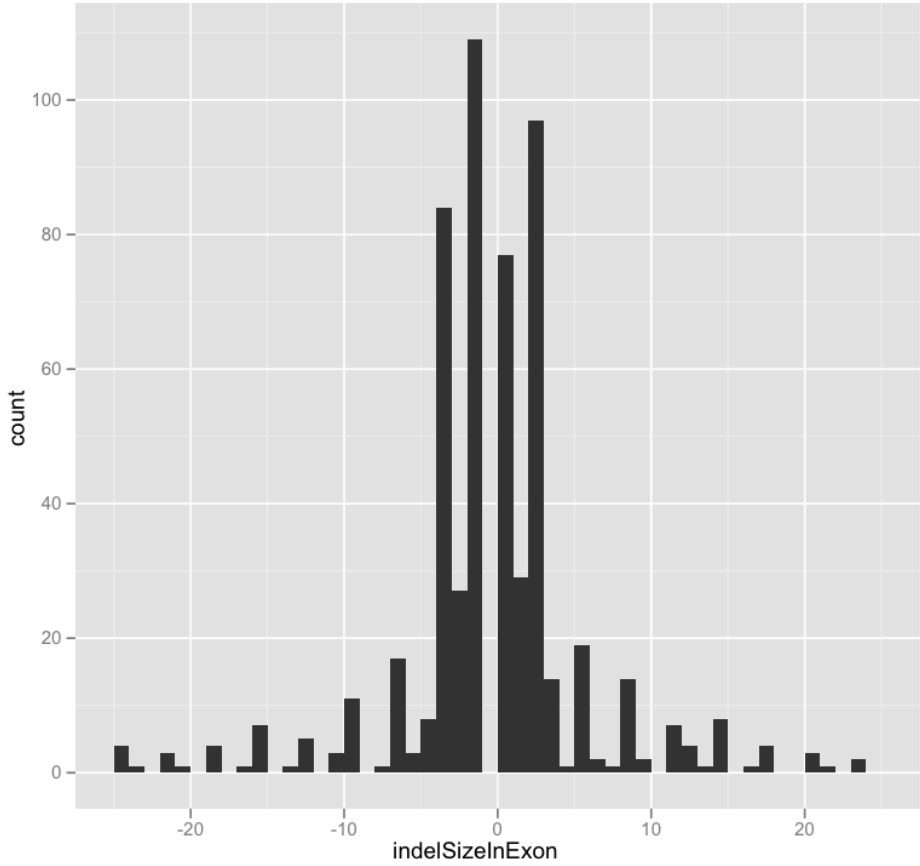


21 individual exomes (of diploid humans) i.e. 42 alleles

SNP numbers and indel size distribution

- Sequencing of human exome
  - 23,602 SNPs in coding exons (approx. 25M bp size)
  - 40,621 SNPs outside coding exons (approx. 25M bp size)





Notice both numbers and pattern in indel figures

MAPPING AND ALIGNMENT 101

Mapping

- Read mapping is the process of locating where in the reference genome a read originates from
- We typically have millions of 100-150 bp sequence reads which we need to map to a reference genome (often billions of bases)
- In addition, there will often not be an exact match between the read and the reference due to:
  - base call errors in individual reads
  - variation in the sample relative to the reference.
- One way of locating the origin of a read is to scan the whole genome looking for a match but this method is obviously very time consuming
  - later in the course we will study an algorithm which speeds this up

After mapping - Alignment

```
Ref    A A A C A A T T A A G T
                 |
Sample         AAAT

Sample A A A C A A A T A A T T A A G T
```

```
Ref    A A A C - - - - A A T T A A G T

Sample A A A C A A A T A A T T A A G T
       ------------------------------>
```

Correct alignment

```
Sample read A A A C A A A T A A T T
```

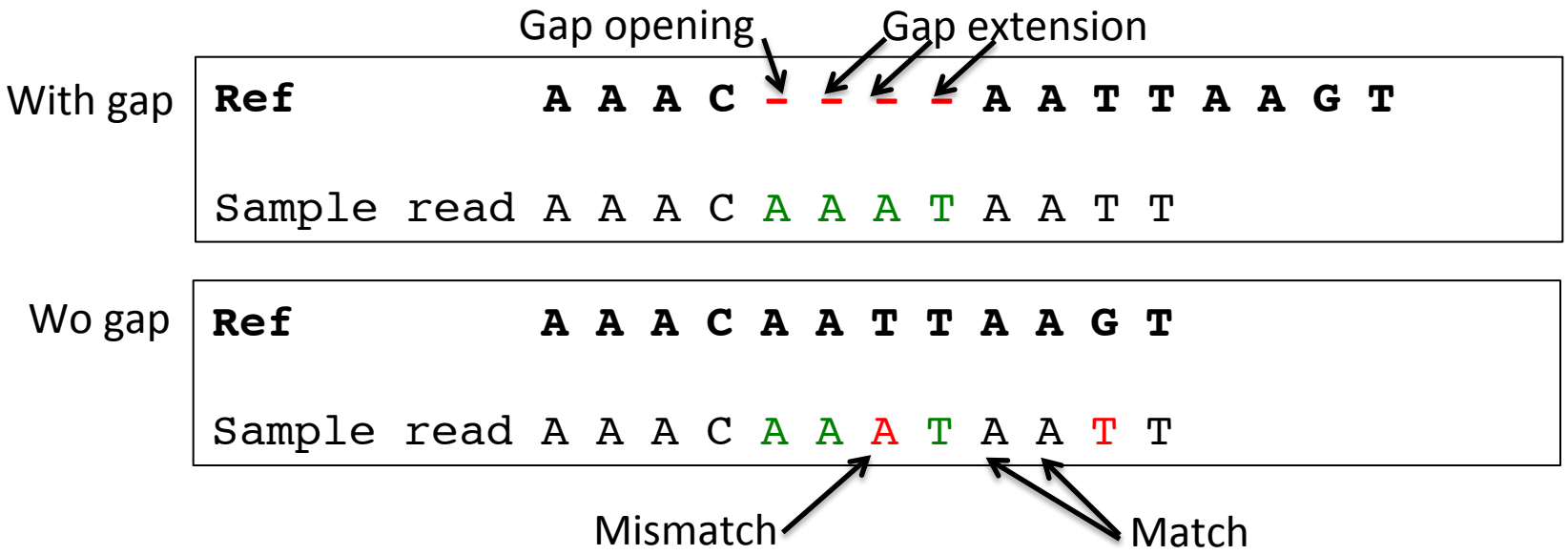
Correct alignment

```
Ref          A A A C - - - - A A T T A A G T

Sample read A A A C A A A T A A T T
```

Possible alignment

```
Ref          A A A C A A T T A A G T

Sample read A A A C A A A T A A T T
```

Alignment



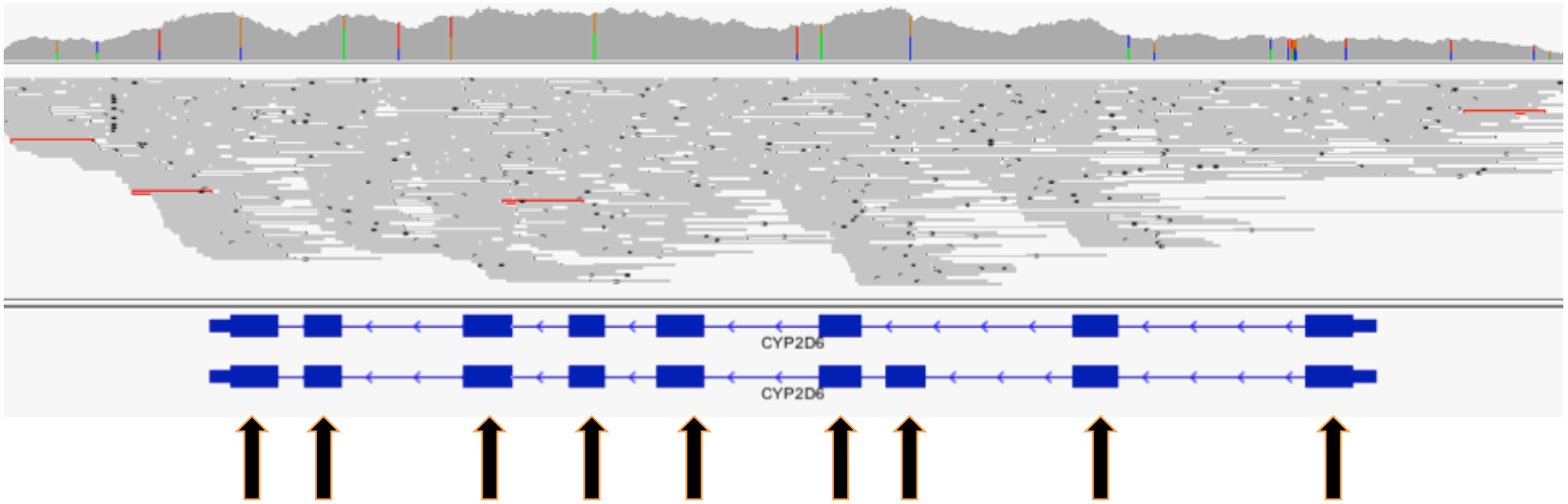
- Key component of alignment algorithm is the scoring
  - negative contribution to score
    - opening a gap
    - extending a gap
    - mismatches
  - positive contribution to score
    - matches
- The exact score contributions determine which alignment is chosen
- **Smith-Waterman** is an algorithm for finding optimal alignment given a scoring scheme without exhaustively enumerating and scoring all possible alignments

EXOME CAPTURE – ESSENTIALS

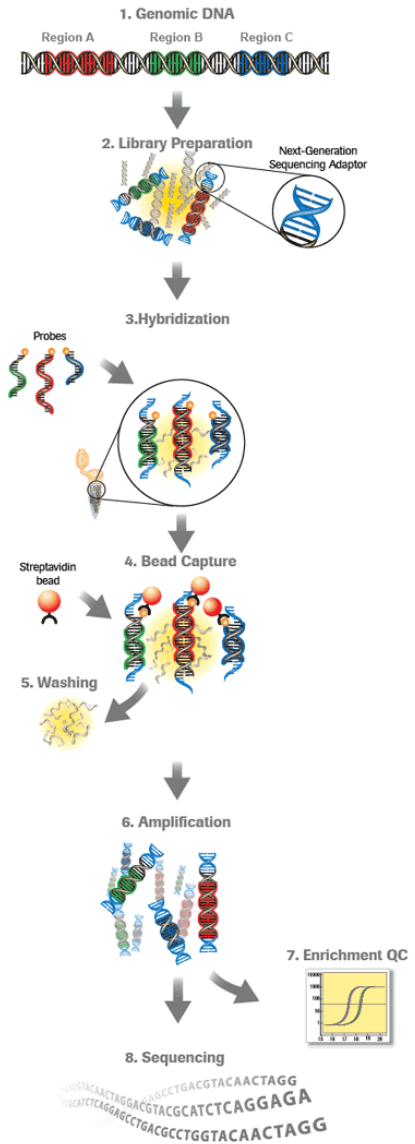
Sequencing a whole genome

- We could sequence a whole genome BUT only a small fraction is actually genes
  - in human less than 2% of the genome codes for proteins

- It saves a lot of sequencing to “capture” the regions we are interested in and sequence them



An overview of exome capture



Sonication

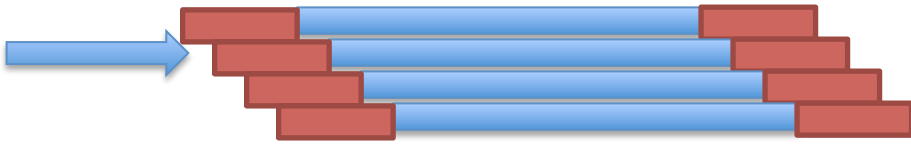
Library prep (sequencing adaptors on)

Oligonucleotide: AGCTAGATA
Sample DNA: ATCTATGATCT

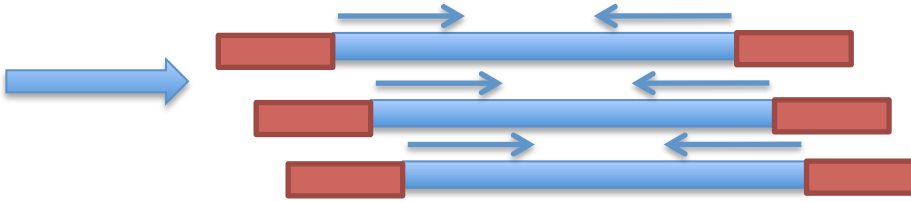
Hybridisation to probes

Bead capture

Amplification



Sequencing



Practical 02 – calling variants in the blind

- We look at the read datasets in local copy
  - **Done together with instructor**

- Open script 020_basicPipeline.bash (this should be your own copy under ${HOME}/vc/exerDefinitions)

- Explanation and execution of commands together with instructor

IGV mini practical

- Open a new terminal and start IGV
- Let us load up some data and explore some of the features of IGV
- We will load two datasets:
  - The dataset that you generated in the previous exercise
    - ${HOME}/vc/exerSandbox/02_basicPipeline
  - Another more realistic simulated dataset
    - ${HOME}/vc/inputData/reads_agilentV1_chr5/simul_NA12878
- Some the things we will do
  - loading files (locally or from server)
  - navigating (zoom in and out, back and forward)
  - getting access to details (on hover or click)
  - changing the way you view things (squish/expand, colours, pairs)
  - moving tracks and panels around and resizing
  - sessions
  - marking regions of interest
  - changing preferences like soft-clipped bases

SEQUENCE – ESSENTIALS

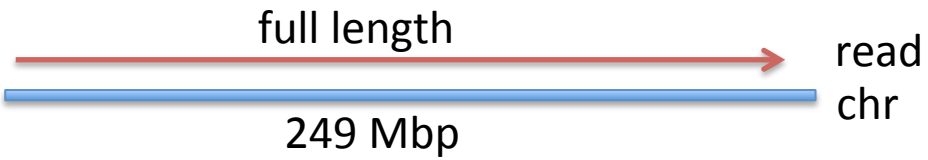
Sequencing

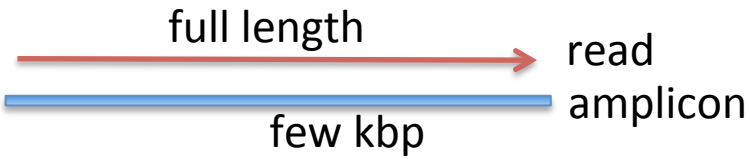
- Here we will be assuming that sequencing was covered in a previous module of the course

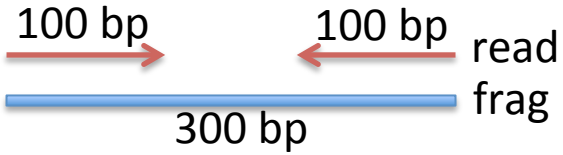
FASTQ FORMAT – ESSENTIALS

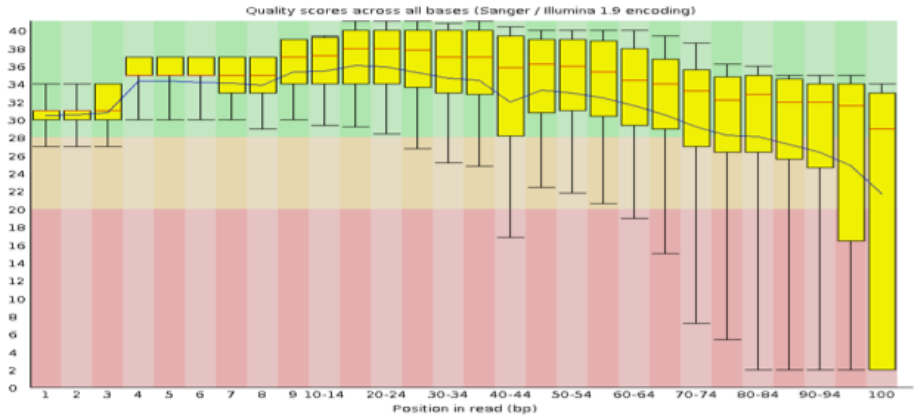
In a perfect world – Perfect sequencing

- Perfect sequencing:
  - single molecule (no PCR)
  - **full** length
  - no deterioration of quality
- While we are waiting:
  - Sanger
    - PCR
    - length: some kb
    - limited number of reads
    - high quality
  - HTS (Illumina)
    - PCR
    - 100 bp PE
    - billions of reads
    - high quality, but deteriorating along read









Fastq format – fasta with qualities

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((****+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

- p = the probability that the corresponding base call is wrong
- Qualities $$Q_{\text{sanger}} = -10 \log_{10} p$$
  - p = 0.1 ➔ Q = 10
  - p = 0.01 ➔ Q = 20
  - P = 0.001 ➔ Q = 30
- Encoding: Sanger/Phred format can encode a quality score from 0 to 93 using ASCII 33 to 126: Q + 33 ➔ ASCII code

Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
32	20	040		Space	64	40	100	@	@
33	21	041	!	!	65	41	101	A	A
34	22	042	"	"	66	42	102	B	B
35	23	043	#	#	67	43	103	C	C
36	24	044	$	$	68	44	104	D	D
37	25	045	%	%	69	45	105	E	E
38	26	046	&	&	70	46	106	F	F
39	27	047	'	'	71	47	107	G	G
40	28	050	(	(	72	48	110	H	H
41	29	051	)	)	73	49	111	I	I
42	2A	052	*	*	74	4A	112	J	J
43	2B	053	+	+	75	4B	113	K	K
44	2C	054	,	,	76	4C	114	L	L
45	2D	055	-	-	77	4D	115	M	M
46	2E	056	.	.	78	4E	116	N	N
47	2F	057	/	/	79	4F	117	O	O
48	30	060	0	0	80	50	120	P	P
49	31	061	1	1	81	51	121	Q	Q
50	32	062	2	2	82	52	122	R	R
51	33	063	3	3	83	53	123	S	S
52	34	064	4	4	84	54	124	T	T
53	35	065	5	5	85	55	125	U	U
54	36	066	6	6	86	56	126	V	V
55	37	067	7	7	87	57	127	W	W
56	38	070	8	8	88	58	130	X	X
57	39	071	9	9	89	59	131	Y	Y
58	3A	072	:	:	90	5A	132	Z	Z
59	3B	073	;	;	91	5B	133	[	[
60	3C	074	<	<	92	5C	134	\	\
61	3D	075	=	=	93	5D	135	]	]
62	3E	076	>	>	94	5E	136	^	^
63	3F	077	?	?	95	5F	137	_	_

Source: http://en.wikipedia.org/wiki/FASTQ_format

Different quality formulas and encodings

- Beware of different versions

```
 SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.....................................................
 ..........................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX......................
 ..............................IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII..........................
 ...............................JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ.........................
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
 |                         |    |        |                              |                     |
33                        59   64       73                            104                   126

S - Sanger, Illumina 1.8+     Phred+33,  raw reads typically (0, 41)
X - Solexa                    Solexa+64, raw reads typically (-5, 40)
I - Illumina 1.3+             Phred+64,  raw reads typically (0, 40)
J - Illumina 1.5+             Phred+64,  raw reads typically (3, 40)
   with 0=unused, 1=unused, 2=Read Segment Quality Control Indicator (bold)
   (Note: See discussion above).
```

- With sequence data recently produced, you do not need to worry
- For older data, be careful
- Tools such as BWA have an option for older formats

Illumina sequence identifiers

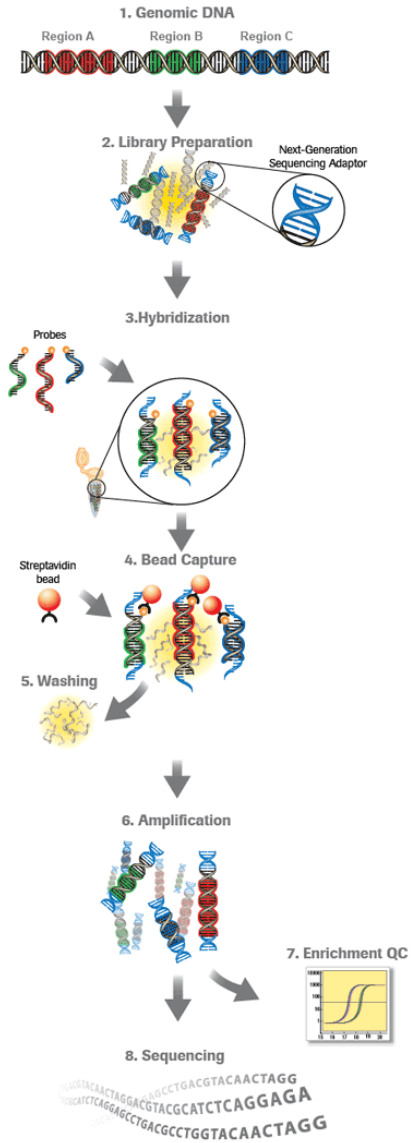
```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

```
@EAS139:136:FC706VJ:2:2104:15343:197393 1:Y:18:ATCACG
```

EAS139	the unique instrument name
136	the run id
FC706VJ	the flowcell id
2	flowcell lane
2104	tile number within the flowcell lane
15343	'x'-coordinate of the cluster within the tile
197393	'y'-coordinate of the cluster within the tile
1	the member of a pair, 1 or 2 *(paired-end or mate-pair reads only)*
Y	Y if the read is filtered, N otherwise
18	0 when none of the control bits are on, otherwise it is an even number
ATCACG	index sequence

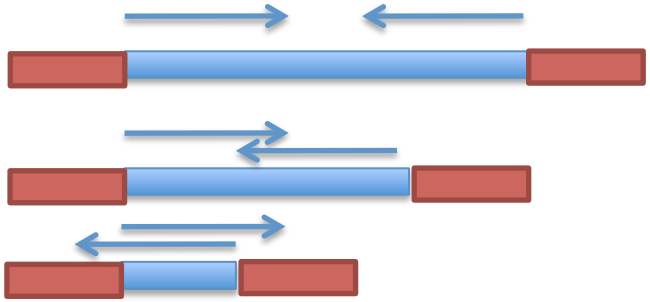
CLEANING UP FASTQ FILES (NOT CURRICULUM)

An overview of exome capture



Sonication

Library prep (sequencing adaptors on)



Problem: adaptor seq (red) in reads

Hybridisation to probes → Possible biases in sequences that hybridise

Bead capture → Possible biases in sequences that elute

Amplification → Possible biases in sequences that amplify

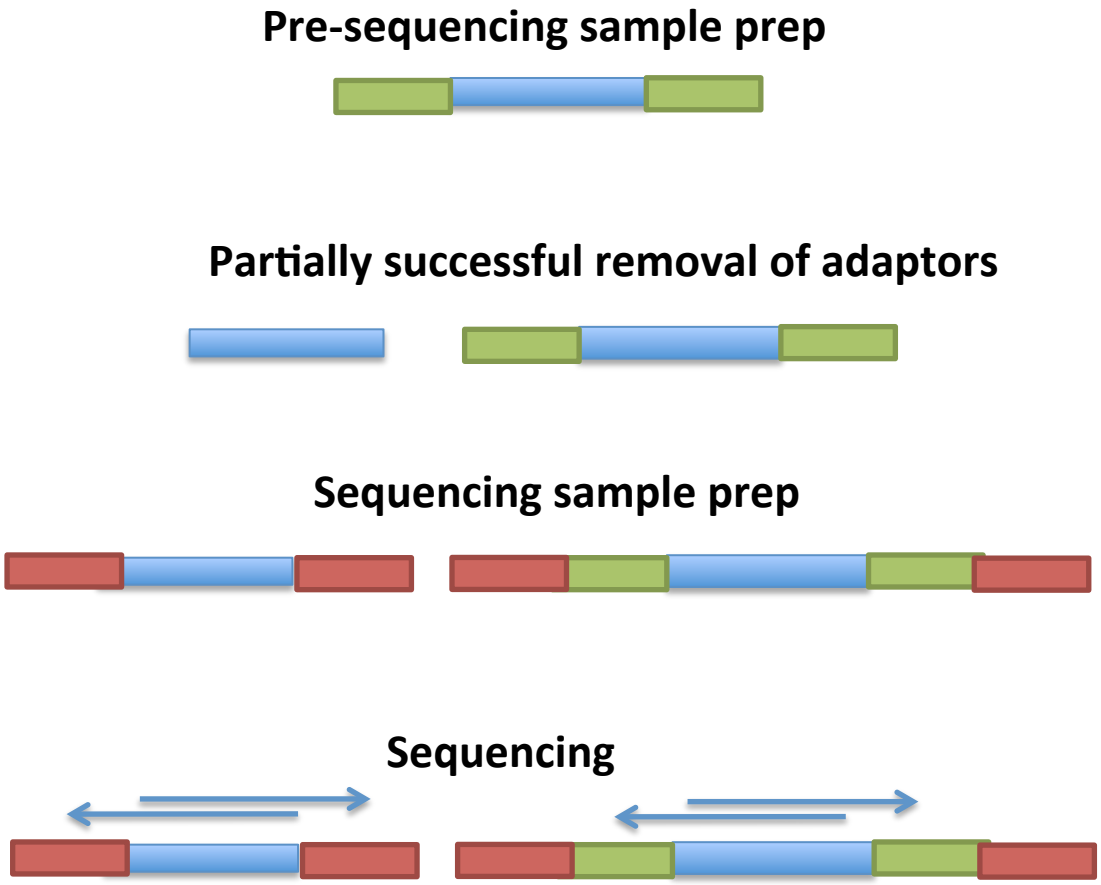
Sequencing → Possible biases in sequences that bridge PCR

The importance of clean up

Problem: adaptor seq in reads

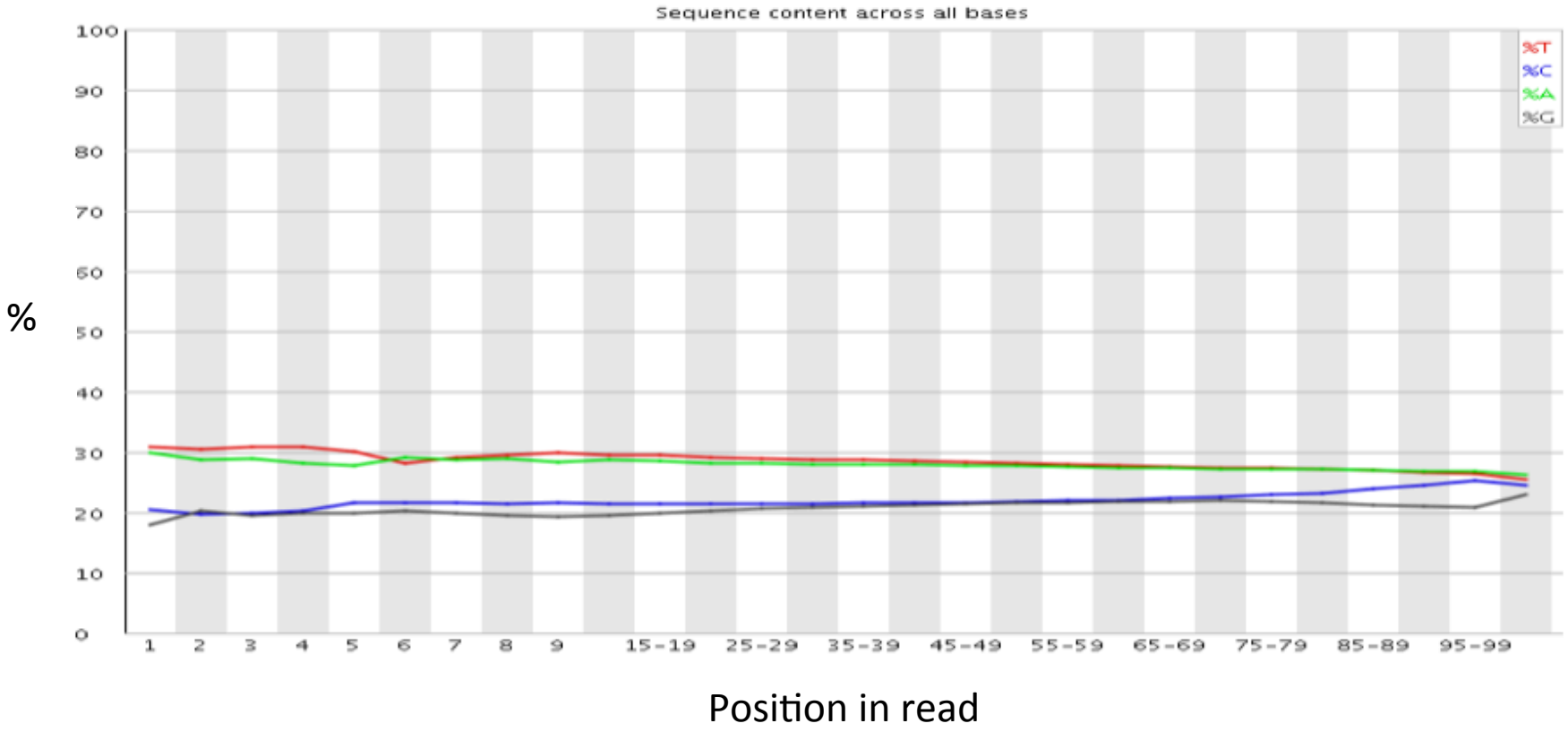
- There will be non-genomic sequences in reads
  - this will often prevent mapping
- Consequences
  - if short fragments are randomly distributed >> less reads mapping >> reduced coverage >> not a big problem as long as the fraction of reads affected is not too large
  - if short fragments are not randomly distributed (e.g. Halo capture) >> specific areas will suffer from reduced/no coverage >> **big problem**
  - **It all depends on the sample whether something really matters or not**

Chip-seq example

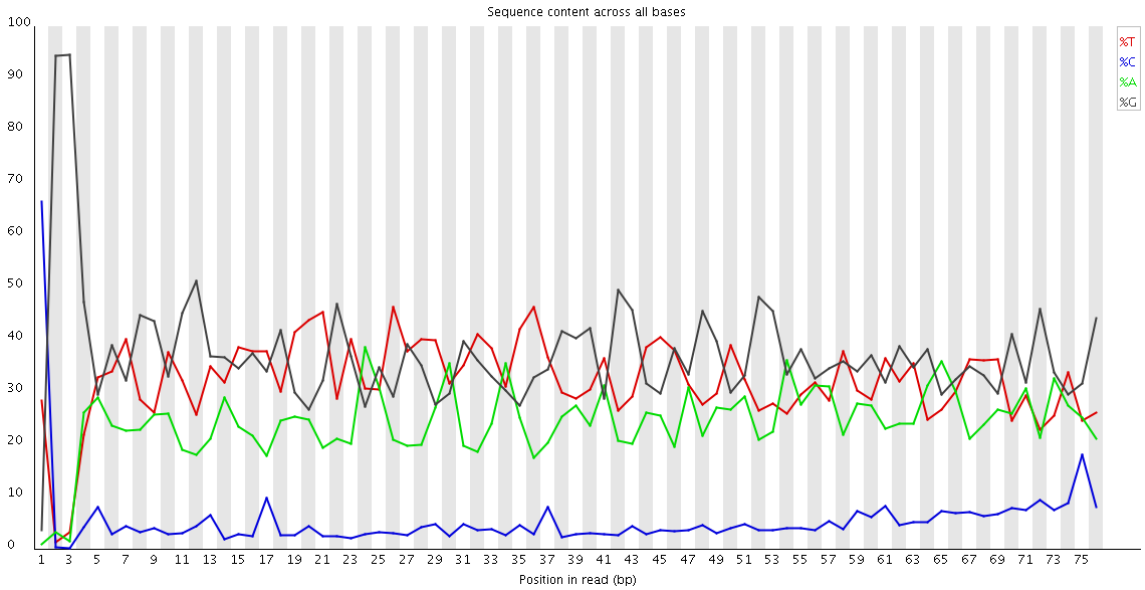


- If there are adaptors still attached to the genomic fragments in the material entering sequencing sample prep, the situation can be even more complicated.
- Fastqc report is essential to get a feel for whether clean-up is required

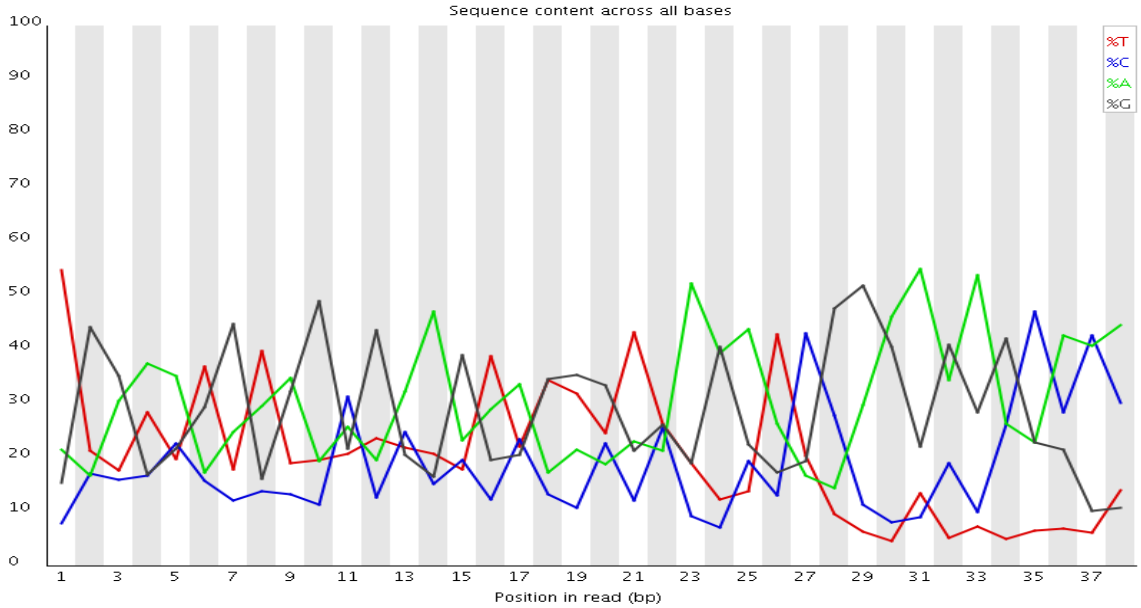
Per cycle sequence content



Per cycle sequence content



?



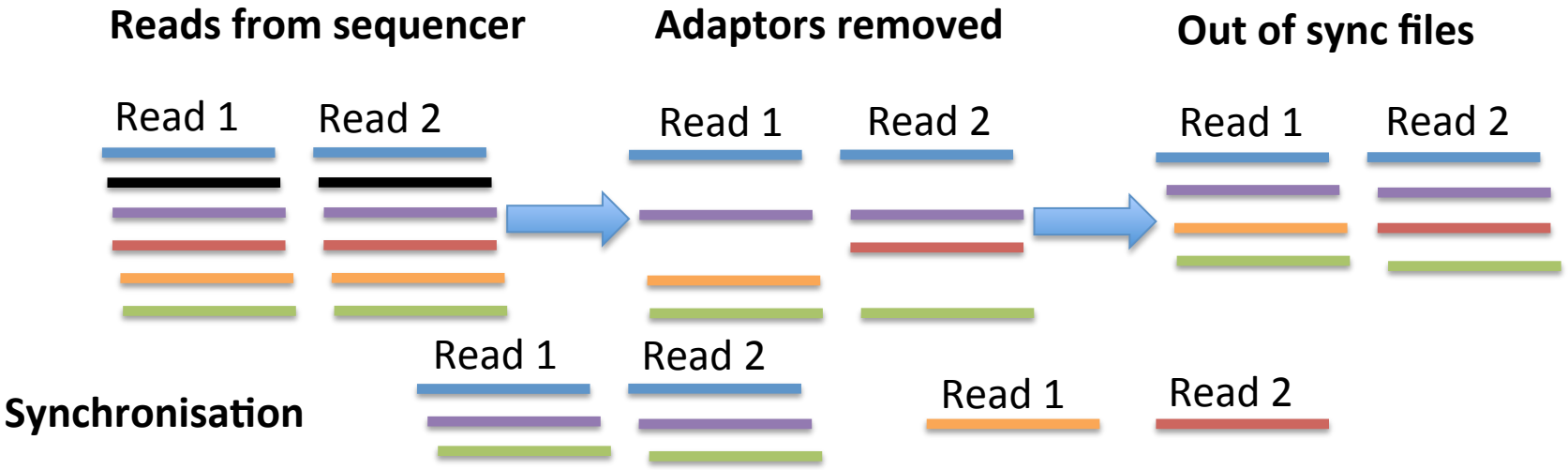
?

Manipulating fasta and fastq files

- Fastx toolkit: http://hannonlab.cshl.edu/fastx_toolkit/
- FASTQ trimmer
- FASTQ quality filter
- FASTQ quality trimmer
- Can do most of the obvious manipulations of fastq/a you may need

How to perform clean up

- Get an overview of the sample preparation both:
  - pre-sequencing
  - sequencing sample prep
- Run fastqc on the data
  - pay particular attention to the base distribution results
- Clean up using
  - fastx-toolkit
  - cutadapt
  - other tools
- Remember that you need to synchronise the R1 and R2 files (with PE)



NB: Synchronisation is very important even when not clipping off adaptors (files can be unsynchronised for other reasons)

MAPPING WITH BWA

Why mapping?

- The biggest difference with Sanger
  - we did not design and use primers for sequence amplification
  - we sonicated
  - >> we do not know where the reads “originate” from

- For each read
  - we need to determine its likely origin
  - how likely it is that we have correctly identified its origin

http://www.cs.jhu.edu/~langmea/resources/bwt_fm.pdf

- The Burrows-Wheeler Transform (BWT) is a way of permuting the characters of a string T into another string BWT (T)
- The permutation is reversible
- For example for T = **abaaba$**
- $ defined to be a cahracter that does not appear elsewhere in T and which is lexicographically prior to all other characters
- The rotations of T

$$
\begin{array}{ccccccc}
\$ & a & b & a & a & b & a \\
a & \$ & a & b & a & a & b \\
b & a & \$ & a & b & a & a \\
a & b & a & \$ & a & b & a \\
a & a & b & a & \$ & a & b \\
b & a & a & b & a & \$ & a \\
a & b & a & a & b & a & \$
\end{array}
$$

Lexicographic sorting of rows

Burrows-Wheeler Matrix BW M(T)

$$
\begin{array}{ccccccc}
\$ & a & b & a & a & b & a \\
a & \$ & a & b & a & a & b \\
a & a & b & a & \$ & a & b \\
a & b & a & \$ & a & b & a \\
a & b & a & a & b & a & \$ \\
b & a & \$ & a & b & a & a \\
b & a & a & b & a & \$ & a
\end{array}
$$

BWT(T)

BWT(T) = abba$aa

Relationship to suffix array

- Suffix array = sorted array of all suffixes of a string
- T = **abaaba$**

```
0  a b a a b a $
1  b a a b a $
2  a a b a $
3  a b a $
4  b a $
5  a $
6  $
```

```
6  $
5  a $
2  a a b a $
3  a b a $
0  a b a a b a $
4  b a $
1  b a a b a $
```

```
BWM:      SA: Suffixes given by SA:
$abaaba    6 $
a$abaab    5 a$
aaba$ab    2 aaba$
aba$aba    3 aba$
abaaba$    0 abaaba$
ba$abaa    4 ba$
baaba$a    1 baaba$
```

Correspond to the same ordering

Reversing the BW transform

- BWT(abaaba$) = abba$aa
- Seem like we have lost which **a** in BWT(T) corresponds to which **a** in T
- BUT we have not thanks to an important property of the BWT called **LF mapping**

$$
\begin{array}{ccccccc}
\$ & a & b & a & a & b & a \\
a & \$ & a & b & a & a & b \\
a & a & b & a & \$ & a & b \\
a & b & a & \$ & a & b & a \\
a & b & a & a & b & a & \$ \\
b & a & \$ & a & b & a & a \\
b & a & a & b & a & \$ & a
\end{array}
$$

$$T = a_0 b_0 a_1 a_2 b_1 a_3 \$$$

The subscript corresponds to the number of times the character has previously been seen in T

We call the subscript the rank
$ does not require rank as occurs only once

$$
\begin{array}{ccccccc}
\$ & a_0 & b_0 & a_1 & a_2 & b_1 & a_3 \\
a_3 & \$ & a_0 & b_0 & a_1 & a_2 & b_1 \\
a_1 & a_2 & b_1 & a_3 & \$ & a_0 & b_0 \\
a_2 & b_1 & a_3 & \$ & a_0 & b_0 & a_1 \\
a_0 & b_0 & a_1 & a_2 & b_1 & a_3 & \$ \\
b_1 & a_3 & \$ & a_0 & b_0 & a_1 & a_2 \\
b_0 & a_1 & a_2 & b_1 & a_3 & \$ & a_0
\end{array}
$$

The LF mapping states: the i^{th} occurrence of a character c in the last column has the same rank as the i^{th} occurrence of c in the first column

We will not prove why

How do we reverse BWT?

We re-rank them:
* Before we ranked according how many times occurred previously in T ie T ranking
* Now we re-rank according to how many times previouly seen in BWT(T) ie B ranking

F	L	$rank$
\$	a	0
a	b	0
a	b	1
a	a	1
a	\$	0
b	a	2
b	a	3

Start with \$

F	L	$rank$
\$	a	0
a	b	0
a	b	1
a	a	1
a	\$	0
b	a	2
b	a	3

Rows are rotations of T so last col of the first row contains the character to left of \$ in T: a

⬇

Rank array tells us this is a with rank 0

⬇

How do we get the character to the left of a_0?

⬇

We can do this if we know the row that begins with a_0

⬇

The LF tells us which row begins with a_0 : the first row containing an "a"

The LF mapping states: the i^{th} occurrence of a character c in the last column has the same rank as the i^{th} occurrence of c in the first column ⟷

Assuming first row is 0, we visit rows: 0, 1, 5, 3, 2, 6, 4 ⟶ $a_3 b_1 a_1 a_2 b_0 a_0 \$$ ⟵

Reversible means useful for compression

FM index – key to searching a string

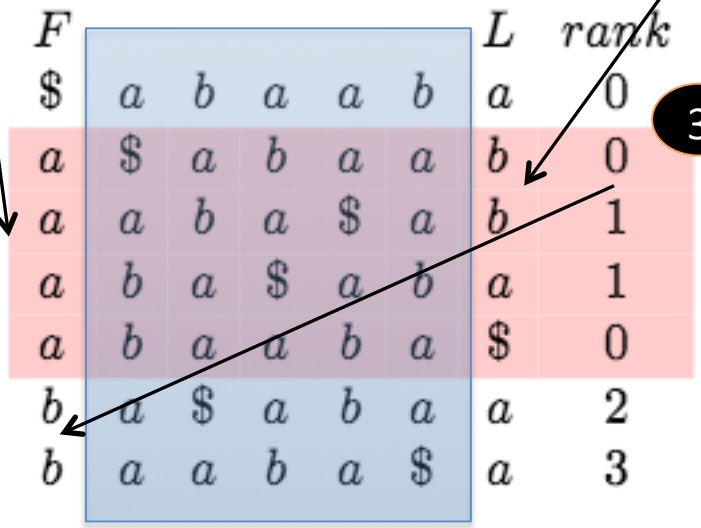
- Ferragina and Manzini: BWT can be used as a space-efficient index of T
- Searching for occurrences of P = aba
- The BWM is sorted, so any rows having P as a prefix will be consecutive

First find the rows beginning with the shortest **suffix** of P i.e. "a"
>> rows 1,2,3 and 4

1

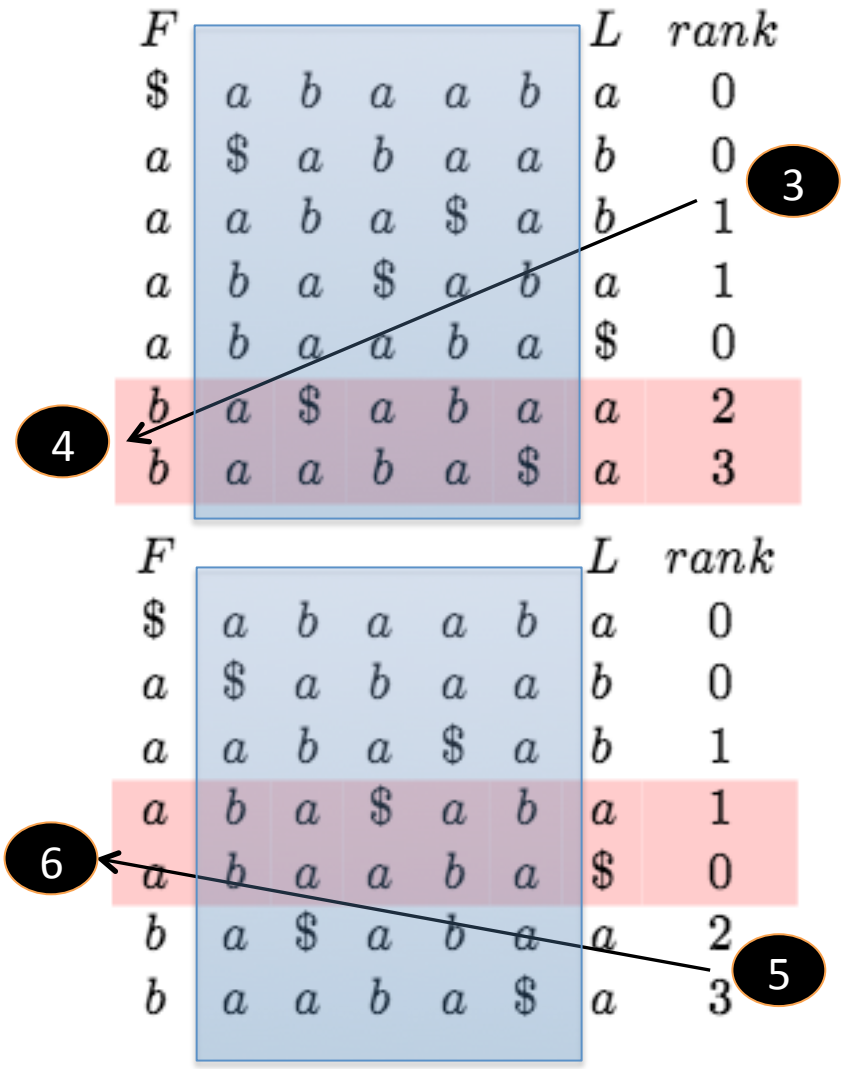
2 Second find all rows begin with the next longest **suffix** of P i.e. "ba"
>> these are rows 1 and 2 (**rows are rotations of T**)

3 The **LF mapping** and rank array tell us which rows have "ba" as a prefix: rows beginning with b0 and b1



F						L	$rank$
$	a	b	a	a	b	a	0
a	$	a	b	a	a	b	0
a	a	b	a	$	a	b	1
a	b	a	$	a	b	a	1
a	b	a	a	b	a	$	0
b	a	$	a	b	a	a	2
b	a	a	b	a	$	a	3

Searching T



The LF mapping and rank array tell us which rows have “ba” as a prefix: row begining with b0 and b1

Occurrences of “ba” are preceded by a2 and a3

Backward matching: apply LF mapping repeatedly to find range of rows **prefixed by successively longer proper suffixes of P until range is empty or run out of suffixes**

With only 3 backward matching (LF mapping and rotations) operations we have found the rows prefixed with occurrences of “aba”

Looking up offsets

So far we have discussed how to use the FM Index to determine whether and how many times a substring P occurs within T, but we have not discussed how to find *where* P occurrs, i.e. the substring's offset into T.

If our index included the suffix array $SA(T)$, we could simply look this up in $SA(T)$. For example, here was the range we ended up with after searching for $P = aba$ within $BWT(abaaba\$)$:

F						L	$SA(T)$
\$	a	b	a	a	b	a	6
a	\$	a	b	a	a	b	5
a	a	b	a	\$	a	b	2
a	b	a	\$	a	b	a	3
a	b	a	a	b	a	\$	0
b	a	\$	a	b	a	a	4
b	a	a	b	a	\$	a	1

$SA(T)$ tells us these matches occurred at T offsets 0 and 3.

Remember: the suffix array and BW matrix have a corresponding order

T=abaaba

Maybe that did not seem impressive.....

F						L	$rank$
\$	a	b	a	a	b	a	0
a	\$	a	b	a	a	b	0
a	a	b	a	\$	a	b	1
a	b	a	\$	a	b	a	1
a	b	a	a	b	a	\$	0
b	a	\$	a	b	a	a	2
b	a	a	b	a	\$	a	3

I could have found the offsets of "aba" in T. Yes, BUT:

- **either** you then scan T with "aba" >> exhausitve search
- **or** you have to store the whole BWM matrix

When the string you are searching is 3 billion bases long that means:

- **either** you check for a match at every 3 bn positions
- **or** you need to store a matrix that is 3bn x 3bn

If you use the BWT and suffix array:

- You only need to store the first and last columns of the BWM (not the blue section)
- You need to do X backward matching operations to locate a string of length X

There are many optimisation to:

- Decrease space requirements
- Decrease time requirements
- Allow for mismatches

Application to read mapping

- You need to build an index of the genome you wish to search
- Reads are 100 bp long
- But, the longer the string searched for the longer the search time (longer backtracking), thus an advantage to search with a shorter string.
- There are 4 different nucleotides and 3.0E09 bases of genome
  - 4**16 = 4.0E09
  - 4**17 = 17.0E09
- Assuming complex sequence, 16 to 17 bp should be enough for a unique match
- BUT
  - variation in sample relative to reference
  - base call errors
  - **THUS** 17 bp search string is not enough
- BWA searches with a "seed" of 32 and allows 2 differences in the seed
- **Reads with seed matches are aligned using Smith-Waterman**

The Fasta, BWA index and other indexes in inputData

- Log into the machines and run
  cd vc/inputData/human_g1k_v37_chr5/gatkBundle
  ls –l *fasta*
- The biggest file is **human_g1k_v37_chr5.fasta**
- Take a look at this file with **less**
- Next biggest is **human_g1k_v37_chr5.fasta.bwt** What do you think this is?
- Another big one is **human_g1k_v37_chr5.fasta.sa** What do you think this is?
- Files with extension: amb, ann, pac, bwt and sa are generated by indexing of .fasta file
- .dict and .fai are very simple files needed by Picard and other programs

What are desirable characteristics of a read mapper?

- Accurately predict the source of a read
  - in the normal range of base error rates
  - in the normal range of indel frequency and size
- But, not necessary to get the alignment exactly right as this can be done later using multiple sequence alignment (MSA)

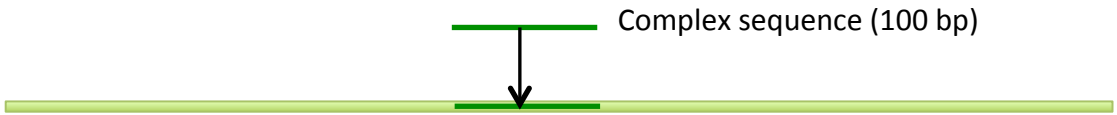
Reference	NNNNNCAAGNNNN
Sample	NNNNNCAAAGNNN

Reference	NNNNNCA_AGGNNN
Correct read align	NNNNNCAAAGNNNN
Reference	NNNNNCAAGGNNN
Alt. align	NNNNNCAAAGNNNN

- Produce an accurate estimate of the reliability of prediction

Factors complicating mapping

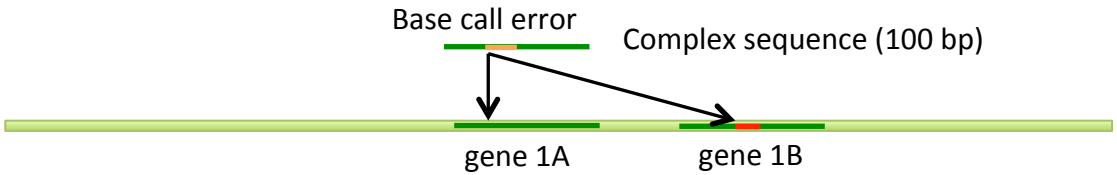
The simple case



Millions of reads

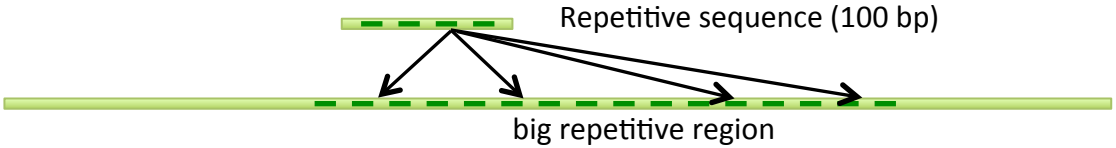
Billions of positions in human genome

Homologous regions



Risk of mismapping

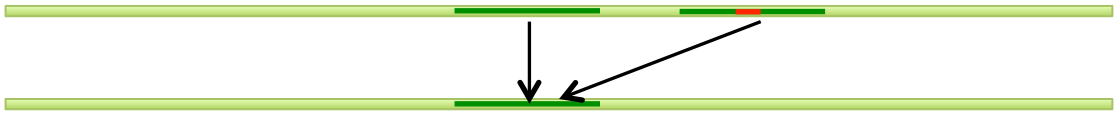
Repetitive region



Impossible to be map correctly

Structural variation (not in reference)

Duplication in sample which is not present in the reference



Definite "mismapping"

Different programs

- BWA
- Novoalign
- BOWTIE
- SOAP
- ....
- Most based on BWT: Burrows-Wheeler Transform
  - a very neat computer algorithm for finding the location of substrings within a string
    - can I find atgc in attgcatcgatcga.....
  - requires indexing of string / reference, but enables
    - rapid search, necessary when mapping billions of reads
    - manageable RAM footprint: 2.3 GB for single reads and 3GB for paired-end (for BWA), so runs on an ordinary computer

Mapping quality scores

- The mapping quality score is the Phred-scaled probability of the mapping being incorrect.

$$Q_{\text{sanger}} = -10 \log_{10} p$$

- Probability is computed from the qualities of the mismatched bases between read and reference and quality features of the second best hit (see Li, Ruan, and Durbin 2008)

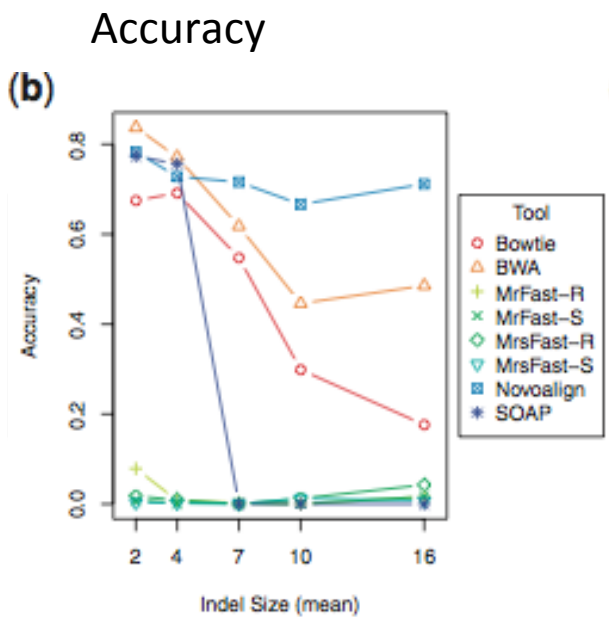
- All programs do not necessarily produce good estimates of mapping quality

- BWA provides good mapping qualites with slight overestimation of quality score:
  - empirical error rate 7x10e-06 for Q60 mappings

BWA

- Fast and accurate short read alignment with Burrows-Wheeler Transform
- But, in practice, makes changes to algorithm to adapt to biological reality and increase speed
- Paired-end mapping
- Like similar programs, randomly places a repetitive read across the multiple equally best positions and mapping quality 0
- Supports multi-threading (as do all BWT aligners)

Accuracy with varying indel sizes

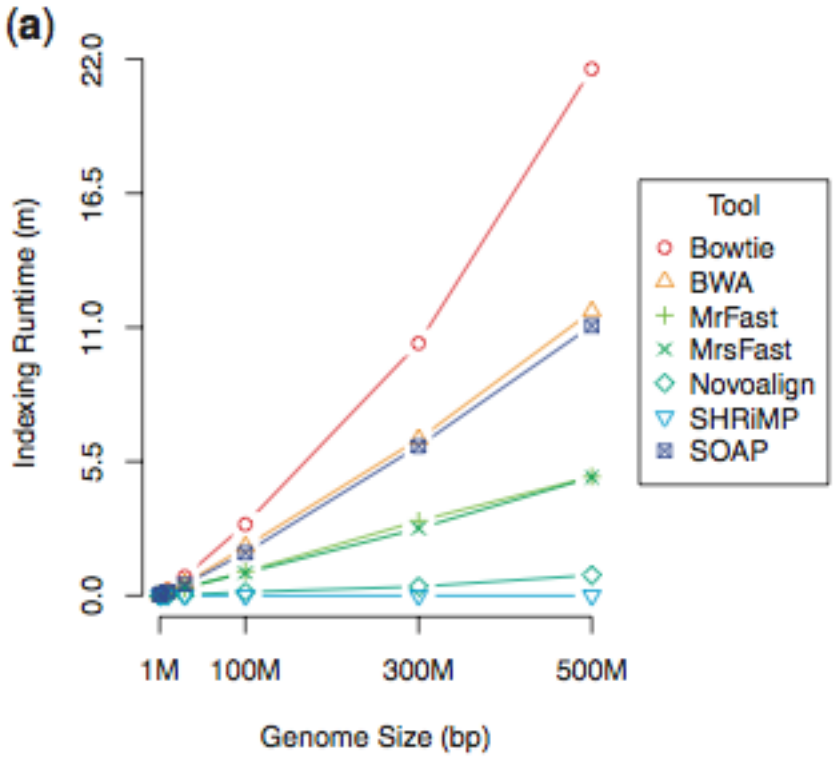


- BWA and Novoalign are best performing

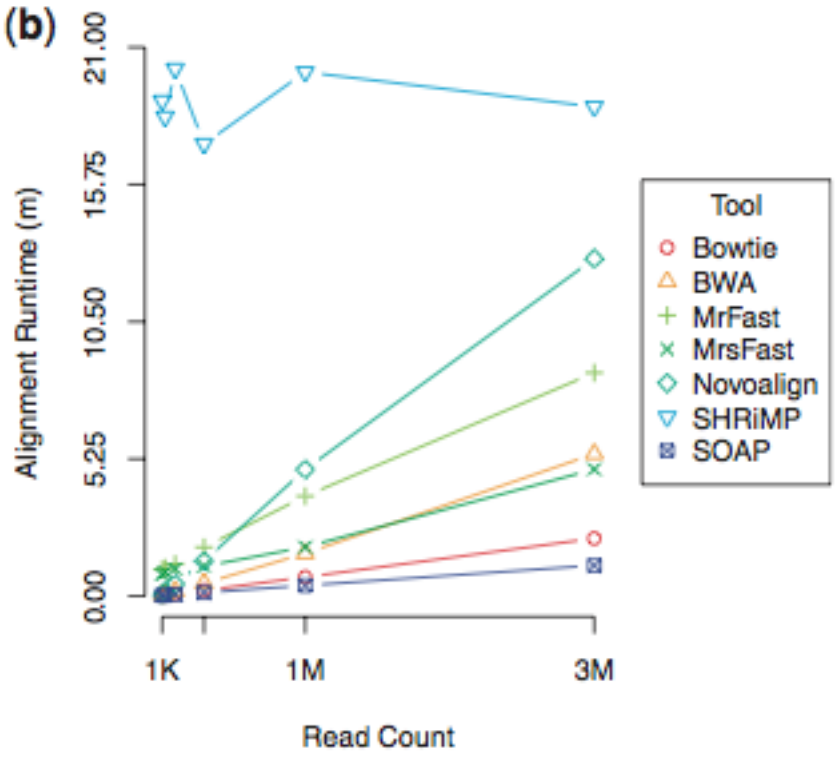
Source: Ruffalo et al. 2011

Runtime

Indexing



Mapping



500 MB genome

Source: Ruffalo et al. 2011

Words of caution about the comparison

- Note this results are for SR 50 bp reads
  - reads are often longer with Illumina
  - can be much longer with other technologies and will require different mapper (see Fast and accurate long-read alignment with Burrows-Wheeler transform)

- **But, in summary:**
  - **mappers are not all of the same quality**
  - **bwa and novoalign are amongst the best for Illumina reads**

Alignment errors after mapping

Base stacks

```
9    t   ttt
10   a   aaaC
11   a   aaaaa
12   a   aaaaaa
13   a   aaaaaa
14   c   cccTTT
15   a   aaaaaa
16   a   aaaaaa
17   t   AAtttt
18   t   tttttt
19   a   aaaaaa
20   a   aaaaaa
21   g   Tgggg
```

```
coor      12345678901234     567890123456789012345678
ref       aggttttataaaac----aattaagtctacagagcaacta
sample    aggttttataaaacAAATaattaagtctacagagcaacta
read1     aggttttataaaac    aaAtaa
read2      ggttttataaaac    aaAtaaTt
read3          ttataaaacAAATaattaagtctaca
read4              CaaaT    aattaagtctacagagcaac
read5                aaT    aattaagtctacagagcaact
read6                 T     aattaagtctacagagcaacta
read1     aggttttataaaacaaataa
read2      ggttttataaaacaaataatt
read3          ttataaaacaaataattaagtctaca
read4                caaataattaagtctacagagcaac
read5                  aataattaagtctacagagcaact
read6                   taattaagtctacagagcaacta
```

Incorrect

Correct

>> Can be solved by alignment: considering all mapping reads and reference together (as we shall see later)

Source: Heng Li, presentation at GSA workshop 2011


FASTQ
Mapping
(BWA)
SAM
HouseKeeping
& refinement
BAM
Variant calling
(gatk)
VCF
file

SAM FORMAT

What does the SAM file look like?

```
@SQ     SN:1    LN:249250621
@SQ     SN:2    LN:243199373
@SQ     SN:3    LN:198022430
@SQ     SN:4    LN:191154276
@SQ     SN:5    LN:180915260
@SQ     SN:6    LN:171115067
@SQ     SN:7    LN:159138663
@SQ     SN:8    LN:146364022
@SQ     SN:9    LN:141213431
@SQ     SN:10   LN:135534747
@SQ     SN:11   LN:135006516
.........
```

Header

Data lines
(one per read)

```
PCUS-319-EAS487_0001:7:1:1002:1094#0  pPr2  5   484690     29  76M     =   484585     -181  ATGCTTGGTGAAGCCGCTCACCAGCGACAGAAGGAAGGCGAA  ;3/<5;;:58?65<'=???<;@?BBA?BB=@@?@A  XT:A:U  NM:i:3  SM:i:29 AM:i:29 X0:i:1  X1:i:0  XM:i:3  XO:i:0  XG:i:0  MD:Z:465G53C11
PCUS-319-EAS487_0001:7:1:1002:1144#0  pPr1  5   141125710  60  76M     =   141125614  -172  GGGACATGACACACGGGGGCCACTCAGGTGGGAAGAATGAGA  =?B/;6=BBACBAA@BBAABBCBCCCCABCCCBBA  XT:A:U  NM:i:1  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:1  XO:i:0  XG:i:0  MD:Z:13A62
PCUS-319-EAS487_0001:7:1:1002:1144#0  pPR2  5   141125614  60  76M     =   141125710  172   TCAATGTCTGTCTCCCACTGGACTGTGAGCACCATACTAGGA  BBB@@BB@@BA???(<@;):=@9?@/5,6=?@=@@  XT:A:U  NM:i:0  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:76
PCUS-319-EAS487_0001:7:1:1002:1152#0  pPr1  18  61647004   60  76M     =   61646915   -165  CTTTGTTTATACAAAGACGGAGATATTCACCGAGGTTTCCAG  B?:>4BCA=BBBAAACBBBBB;BABBCBCBBCCCA  XT:A:U  NM:i:2  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:2  XO:i:0  XG:i:0  MD:Z:1765G52
PCUS-319-EAS487_0001:7:1:1002:1152#0  pPR2  18  61646915   60  76M     =   61647004   165   TTTTTAGTACAGAGGTTTGTTTGGAGAAAGCTCTTTGGGAGA  BCBCCACB>BCCACBBCBCBBBCBABACA=AB@?>  XT:A:U  NM:i:0  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:76
PCUS-319-EAS487_0001:7:1:1002:1173#0  pPr1  11  131794651  60  76M     =   131794549  -178  TTCAGAGATGATTTGTTACAATAGCTAGCATTATCCTATCTA  AA=BBABBBB@BBB@BA@AA?>>BBBA=B?BACCB  XT:A:U  NM:i:0  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:76
PCUS-319-EAS487_0001:7:1:1002:1173#0  pPR2  11  131794549  60  7M168M  =   131794651  178   GACAGCACAGAGCAGGGCTCACATCCCCACCCACCAGCTGGC  ?;>>87>:>@??;?93>9;97:8>88897;333::  XT:A:U  NM:i:2  SM:i:37 AM:i:37 X0:i:1  X1:i:0  XM:i:1  XO:i:1  XG:i:1  MD:Z:56C18
PCUS-319-EAS487_0001:7:1:1002:1177#0  pPr1  13  47061278   55  76M     =   47061179   -175  TCACTTGAACCCAGGAGGCAGAGATTTCAGTGAGCCAAGATC  ;6@<@@A@>AA<AAB<@AA@59<@@BB<5ABBBAB  XT:A:U  NM:i:0  SM:i:37 AM:i:18 X0:i:1  X1:i:0  XM:i:0  XO:i:0  XG:i:0  MD:Z:76
PCUS-319-EAS487_0001:7:1:1002:1177#0  pPR2  13  47061179   55  76M     =   47061278   175   CATGGTGAAACCCTGTCTTTACTAAAAATACAAAAAATTATC  @?A=?@?@@=A@A>=@@=<<BAA==???A@??A@A  XT:A:U  NM:i:0  SM:i:18 AM:i:18 X0:i:1  X1:i:3  XM:i:0  XO:i:0  XG:i:0  MD:Z:76
PCUS-319-EAS487_0001:7:1:1002:1205#0  pPR1  7   94285373   29  75M1S   =   94285448   151   CAGGGGTCTCCCAGCTCCCACCACCGGGGCAATTGCATTCTT  =45##################################  XT:A:M  NM:i:5  SM:i:29 AM:i:29 XM:i:5  XO:i:0  XG:i:0  MD:Z:58C5T1G0A2C4
```

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,255}`	Query template NAME
2	FLAG	Int	$[0, 2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0, 2^{29}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0, 2^{8}-1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next segment
8	PNEXT	Int	$[0, 2^{29}-1]$	Position of the mate/next segment
9	TLEN	Int	$[-2^{29}+1, 2^{29}-1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

Source: The SAM format specification

Inspecting one record

```
PCUS-319-EAS487_0001:7:1:1002:1094#0
pPr2
5
484690
29
76M
=
484585
-181
ATGCTTGGTGAAGCGCGTCACCAGCGACAGAAGGAAGGCGAA
;;;;;;;;3/<5;;:58?65<'=???<;@?BBA?BB=@@?@A
```

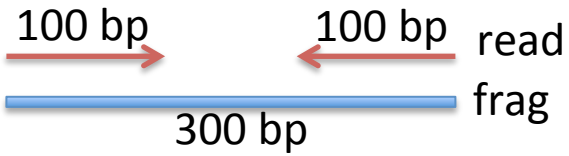
Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	[!-?A-~]{1,255}	Query template NAME
2	FLAG	Int	$[0,2^{16}-1]$	bitwise FLAG
3	RNAME	String	*|[!-()+-<>-~][!-~]*	Reference sequence NAME
4	POS	Int	$[0,2^{29}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0,2^{8}-1]$	MAPping Quality
6	CIGAR	String	*|([0-9]+[MIDNSHPX=])+	CIGAR string
7	RNEXT	String	*|=|[!-()+-<>-~][!-~]*	Ref. name of the mate/next segment
8	PNEXT	Int	$[0,2^{29}-1]$	Position of the mate/next segment
9	TLEN	Int	$[-2^{29}+1,2^{29}-1]$	observed Template LENgth
10	SEQ	String	*|[A-Za-z=.]+	segment SEQuence
11	QUAL	String	[!-~]+	ASCII of Phred-scaled base QUALity+33

Difference between 1-based and 0-based coordinates

```
NNCTGGTNNN
123456789 ==> specified as closed interval ==> coords 3-7 ==> length = 7 - 3 + 1
012345678 ==> specified as half-closed half-open ==> coords 2-7 ==> length = 7 - 2
```

- SAM (+ VCF and GFF) are 1-based
- BED are 0-based
- Can be very important when manipulating SNP coordinates >> be careful

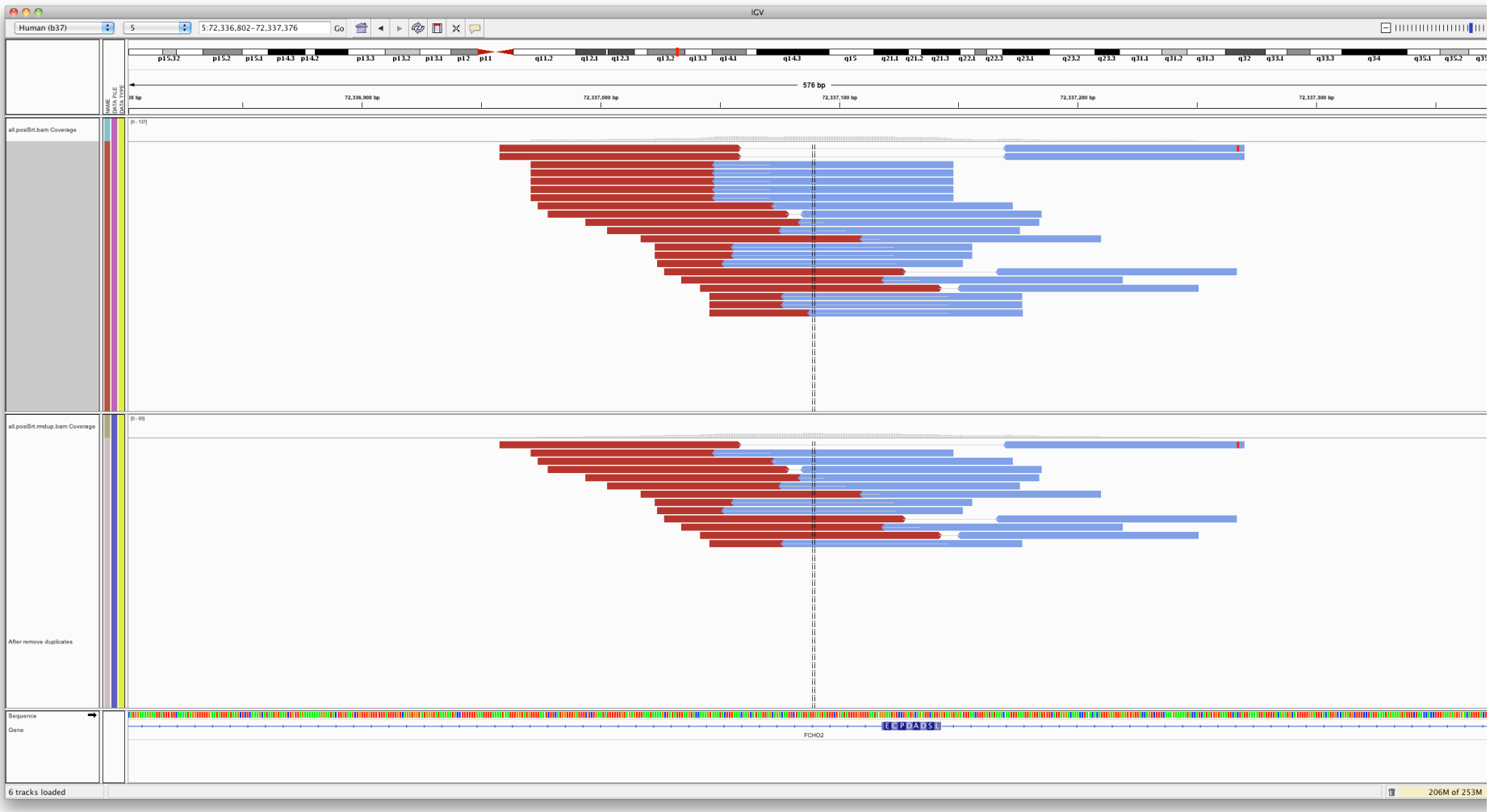
The FLAG column – a bit wise flag




```
p=0x1 (paired sequencing)
P=0x2 (properly paired after mapping)
u=0x4 (unmapped)
U=0x8 (mate unmapped)
r=0x10 (reverse)
R=0x20 (mate reverse)
1=0x40 (first read in pair)
2=0x80 (second read in pair)
s=0x100 (not primary) ==> if read has multiple mapppings one must be primary
f=0x200 (failure) ==> does not pass filter
d=0x400 (duplicate) ==> PCR or optical duplicate
```

- Translate from bit wise flag to readable codes by using **samtools view –X**
- **OR** take a look at **https://broadinstitute.github.io/picard/explain-flags.html**

Introducing flags - What is a duplicate?



About the SAM file produced by BWA

- It contains **all** the reads >> the Picard/GATK paradigm: information is annotated (and not filtered)
  - unique
  - ambiguous
  - unmapped
- It has a number of short comings
  - it takes a lot of space ➔ convert to BAM
  - the mates are not fully updated on each others existence ➔ fixmate
  - it is not sorted ➔ sort
  - it contains PCR duplicates ➔ mark or remove duplicates
  - it does not contain meta-data on the reads (sample, sequencer, etc)

What tools can I use to manipulate the data?

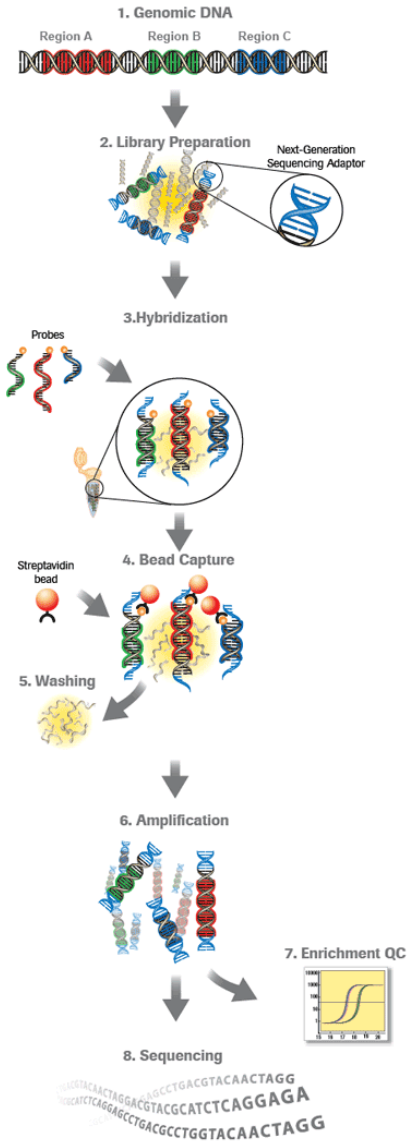
- Two alternative software packages for doing this
  - samtools
    - basic
    - easy to use
  - picard
    - advanced functionality
    - used by GATK team
- For simple tasks like viewing, indexing, sorting → samtools
- For more advanced tasks and preparing for downstream analysis → Picard due to GATKs pickyness

- These tools will be introduced in more detail in the practicals

- **Brief practical: Take a look at your SAM file**

COMPUTING ADVANCED METRICS – PICARD

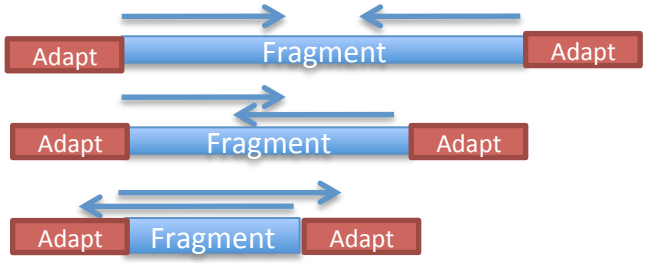
An overview of exome capture – weak points



Sonication

Library prep (sequencing adaptors on)

Problem: **error in sonication** >> adaptor seq in reads >> **unmapped reads**



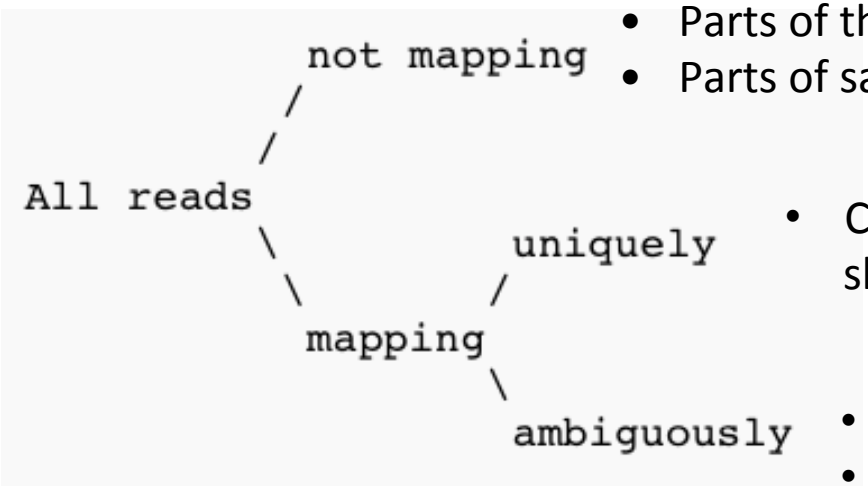
Hybridisation to probes → Possible biases in sequences that hybridise >> **coverage bias**

Bead capture → Possible biases in sequences that elute >> **coverage bias**

Amplification → Possible biases in sequences that amplify **>> sequence PCR duplicates**

Sequencing → Possible biases in sequences that bridge PCR >> **coverage bias**

Metrics - Basic read classification

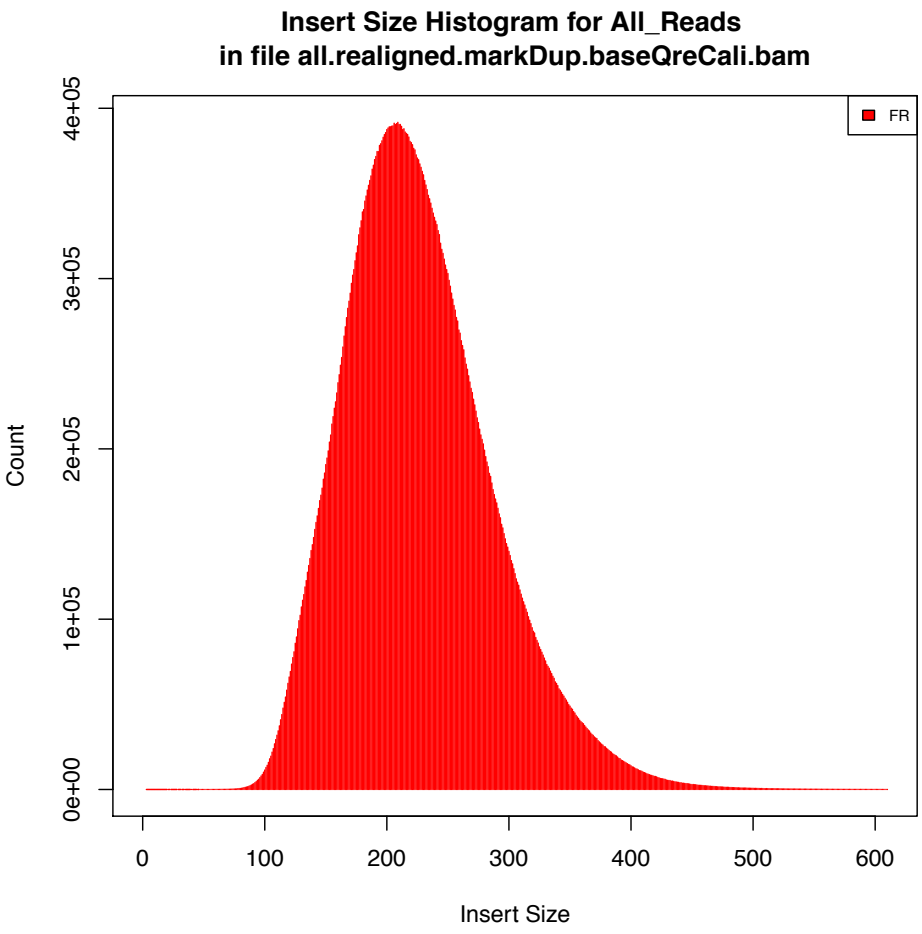


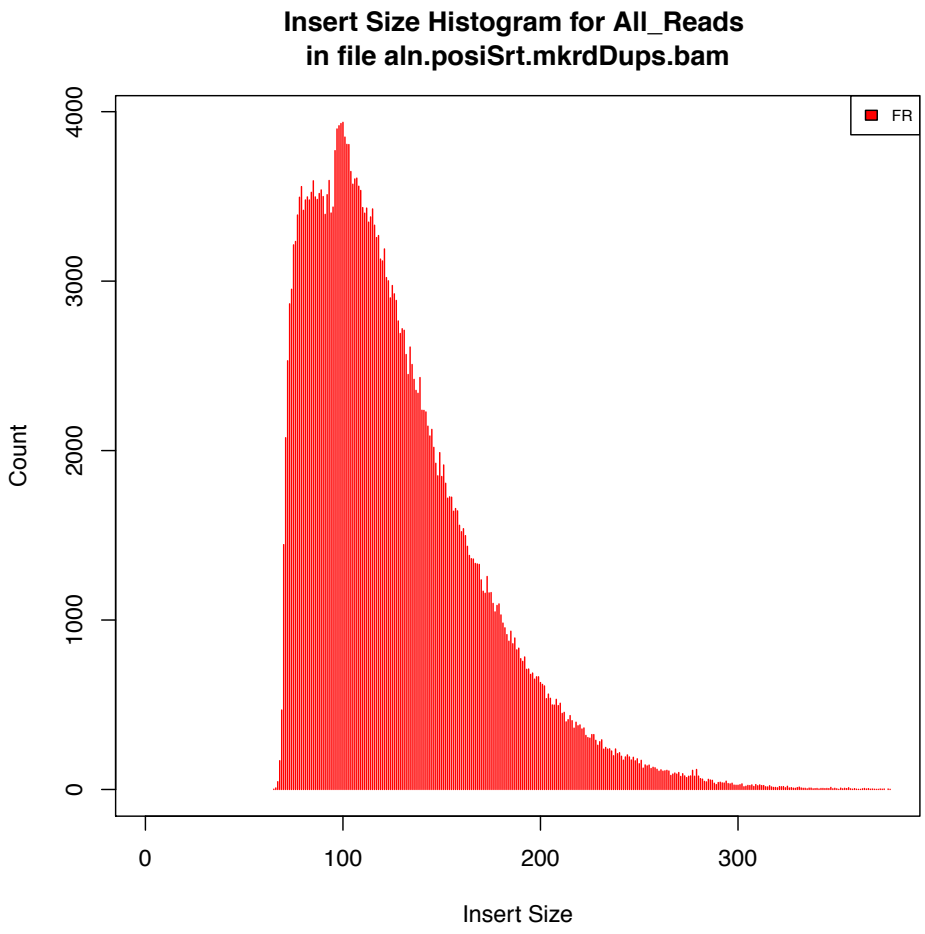
- Contamination from other species
- Reads containing non-genomic DNA e.g. adaptors
- PCR gunk
- Reads with sequencing errors
- Parts of the genome that are not assembled
- Parts of sample affected by structural variation

- Complex sequence with sufficient length should map uniquely

- Typical for repeats
- Also possible for homologous regions

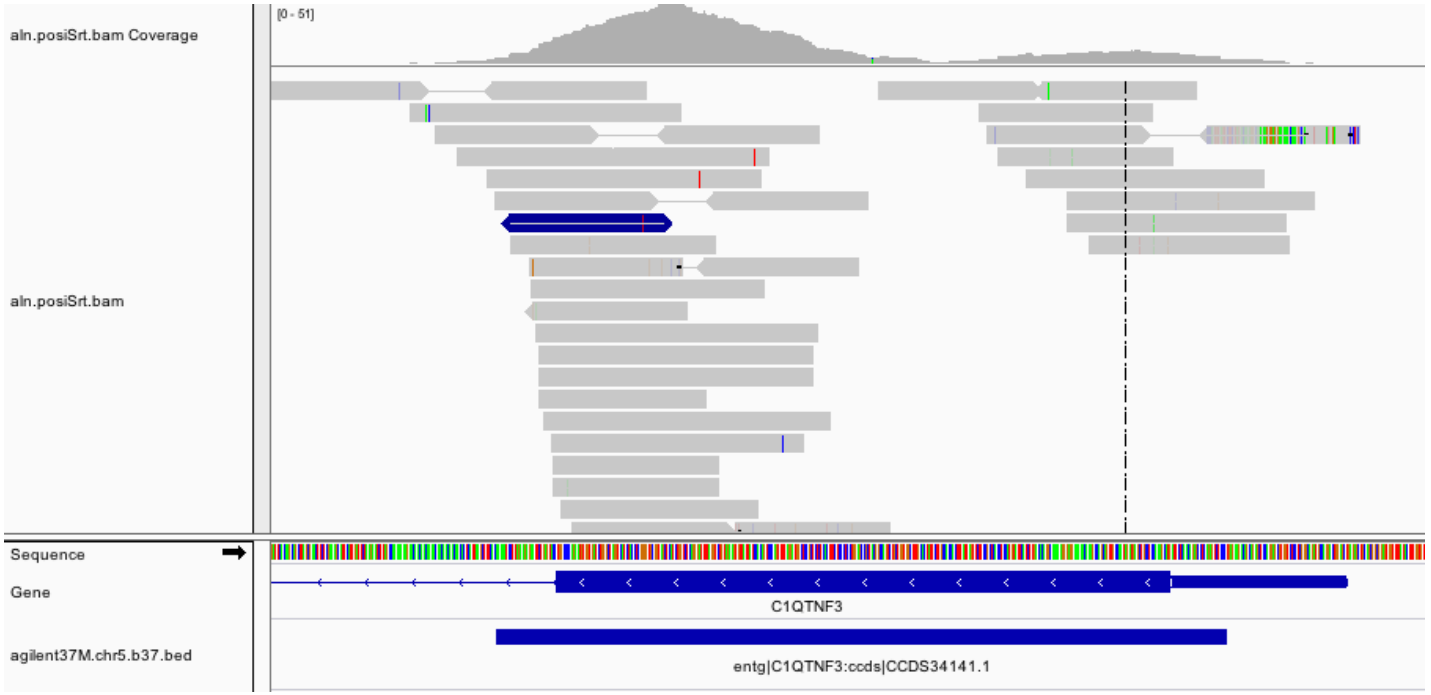
Metrics – Insert sizes



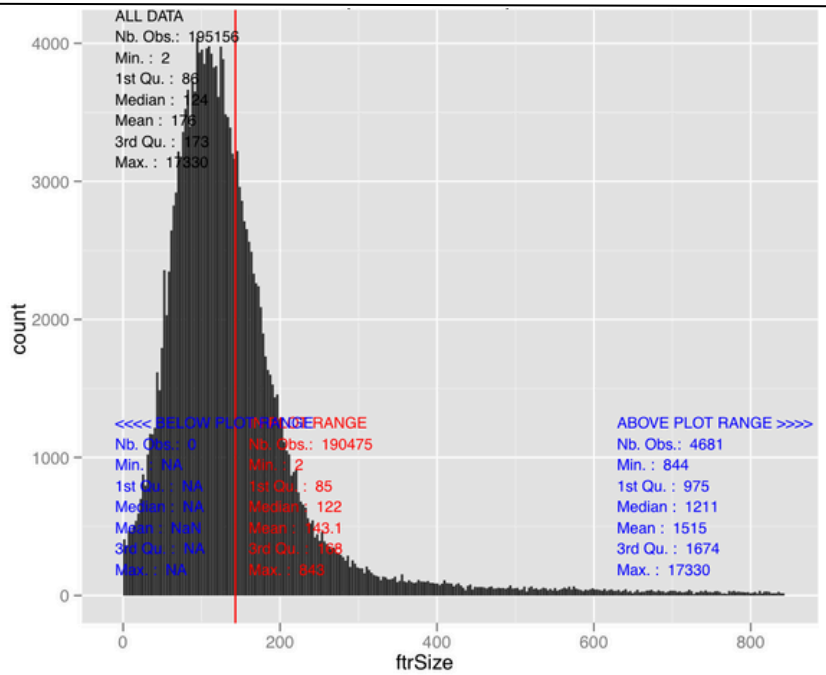


Metrics - Coverage

- Even if doing Whole Genome Sequencing (WGS) >> coverage issues
  - due to repetitive regions
  - due to properties of the DNA e.g. GC content
- Exome sequencing >> Capture by hybridisation



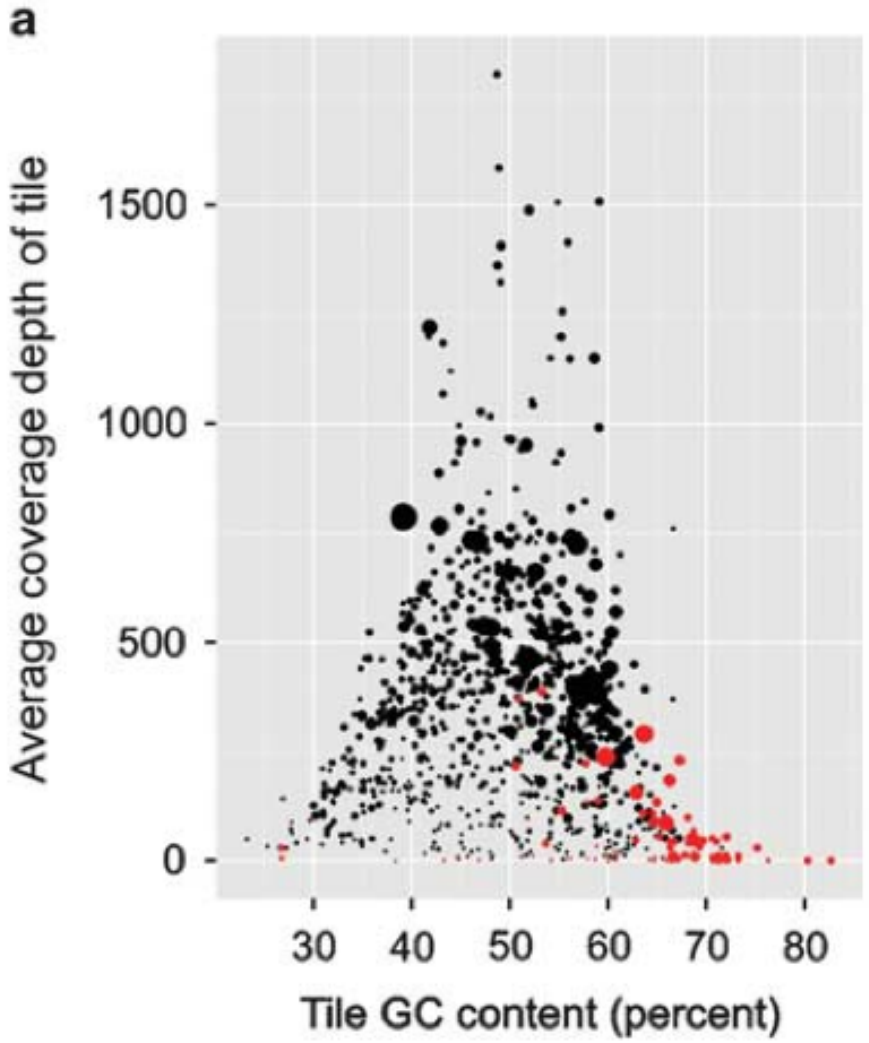
Exome capture – The nature of the target



CCDS exon length distribution

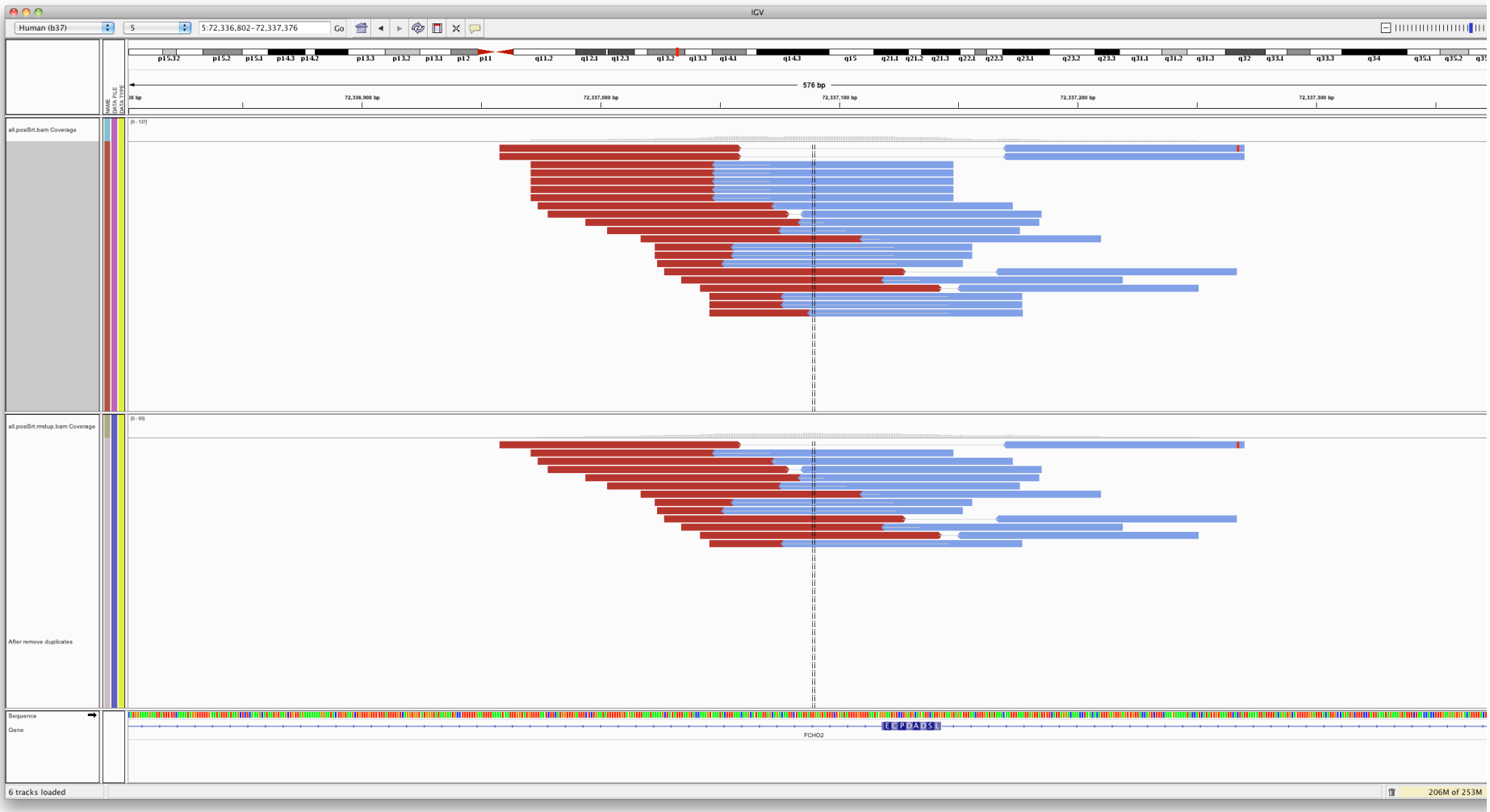
```
Exon(x) - the target:              IIIIIXXXXXXXXXXXXXRRRRRXXXXXXXIIIII
(R=repeat, I=intron)

Nimblegen (avoid repeats and tiles no longer than need to be)
The tiled (T) region (or bait):     TTTTTTTTTTTTT     TTTTTTTTTT
The oligos(O) (or probes):          00000             000000
                                      00000               0000
                                        00000               0000
                                          00000               0000
                                            00000

Agilent (no attempt to avoid repeats and no oligo overlap so tiles are always
multiple of 120 so often sequence well into introns)
The tiled (T) region (or bait):   TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
The oligos(O) (or probes):          000000000000000|0000000000000


There may be more than one tiled region per target
.e.g. if the target contains repeats.
Oligo spacing: distance between start positions of oligos
Oligo overlap: number of overlapping bases between oligos
```

Coverage metrics – Effect of GC content



ZERO_CVG_TARGETS_PCT	0.031204
FOLD_80_BASE_PENALTY	2.955833
PCT_TARGET_BASES_2X	0.930749
PCT_TARGET_BASES_10X	0.634677
PCT_TARGET_BASES_20X	0.334935
PCT_TARGET_BASES_30X	0.16685

Duplicate metrics



Duplicates potentially introduce variant calling errors

021_generatingReports.bash

- Time to see how we can do some of this in practice
- Walk through the fastq and SAM/BAM part of the practical
- Stop when we get to VCF part!
- **Walk through this together with instructor**

VCF FILE - BASICS

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        NA00
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
20     17330   .         T   A       3    q10    NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ 0|0:49:3:58,50
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
20     1230237 .         T   .       47   PASS   NS=3;DP=13;AA=T                   GT:GQ:DP:HQ 0|0:54:7:56,60
20     1234567 microsat1 GTCT G,GTACT 50   PASS   NS=3;DP=9;AA=G                    GT:GQ:DP    0/1:35:4
```

Meta data: definitions of tags used elsewhere in data lines

Header line

Data lines

Variant columns

Genotype columns

Source: The SAM format specification

Columns of data lines

- **CHROMO:** chromosome / contig
- **POS:** the reference position with the 1st base having position 1
- **ID:** an id; rs number if dbSNP variant
- **REF:** reference base.
  - The value in POS refers to the position of the first base in the string
  - for indels, the reference string must include the base before the event (and this must be reflected in POS)
- **ALT:** comma sepearated list of alternate non-ref alleles called on at least one of the samples
  - if no alternate alleles then the missing value should be used "."
- **QUAL:** phred-scaled quality score of the assertion made in ALT (whether variant or non-variant)
- **FILTER:** PASS if the position has passed all filters (defined in meta-data).
- **INFO:** additional information

Source: The SAM format specification

REF and ALT

Reference a t C g a >> C is reference base

		REF	ALT
Variant	a t **G** g a >> C is a G →	20 3 . C	G
Variant	a t **–** g a >> C is deleted →	20 **2** . TC	T
Variant	a t C**a**g a >> A is inserted →	20 3 . C	CA

Source: The SAM format specification

REF and ALT

Reference a t C g a >> C is reference base

				REF	ALT
Variant	a t **G** g a >> C is a G	→	20 3 . C G		
Variant	a t **–** g a >> C is deleted	→	**20** **2** . TC T		
				REF	**ALT**
To represent both in the same record		→	**20** **2** . TC **T,TG**		

Comparing VCF files: normalisation

- The VCF format is quite precise but still leaves room for representing one variant in multiple ways
  - VCF files need normalisation before comparison

- **Parsimony**
  - Pos: 5, Ref: ATC, Alt: AT
  - **Or** Pos: 6, Ref: TC, Alt: T >> most parsimonious
- **Left alignment**, suppose context: pos 8, ref: ATTTT, T deletion
  - Pos: 10, Ref: TT, Alt: T
  - **Or** Pos: 8, Ref: AT, Alt: A >> left aligned
- **MNP on separate lines**
  - 150 TCT CCC
  - Can be decomposed into two records: 150 T C AND 152 T C

- **One should also ensure that the same reference naming is used in both comparison files and that both files have the same sort order**

- More details at: https://github.com/chapmanb/bcbio.variation/wiki/Normalized-variant-representation and http://genome.sph.umich.edu/wiki/Variant_Normalization

021_generatingReports.bash (part II: VCF file)

- Time to take a look at the VCF part of the practical

- **Do this together**

Practical 022_workingWithFormats

- introduction to manipulation of:
  - SAM/BAM
  - VCF

- **Walk through this together**

APPENDIX

Overview of topics (not in chrono order)

- Software and datasets setup and intro
- Fastq format
- Read mapping (SAM/BAM format)
- IGV
- Variant calling (VCF format)
- Metrics reports (esp coverage – BED format)
- Alignment refinement
- Base quality score recalibration
- Variant annotation and functional filtration

Comparison of different programs (optional)

- Comparison analysis of algorithms for next-generation sequencing read alignment, Ruffalo et al. 2011
  - generate genome in silico with certain indel properties (size and frequency)
  - simulate reads from this genome with certain error rate
  - measure mapper accuracy: the true location of a read is considered the truth and a mapping is considered a prediction
- What to look out for on the next slide
  - Ignore the Fast tools as they are poor mappers (green in legend)
  - Only focus on the parts of the plots that cover realistic range of parameters
- In summary, BWA and Novoalign are both good mapping tools

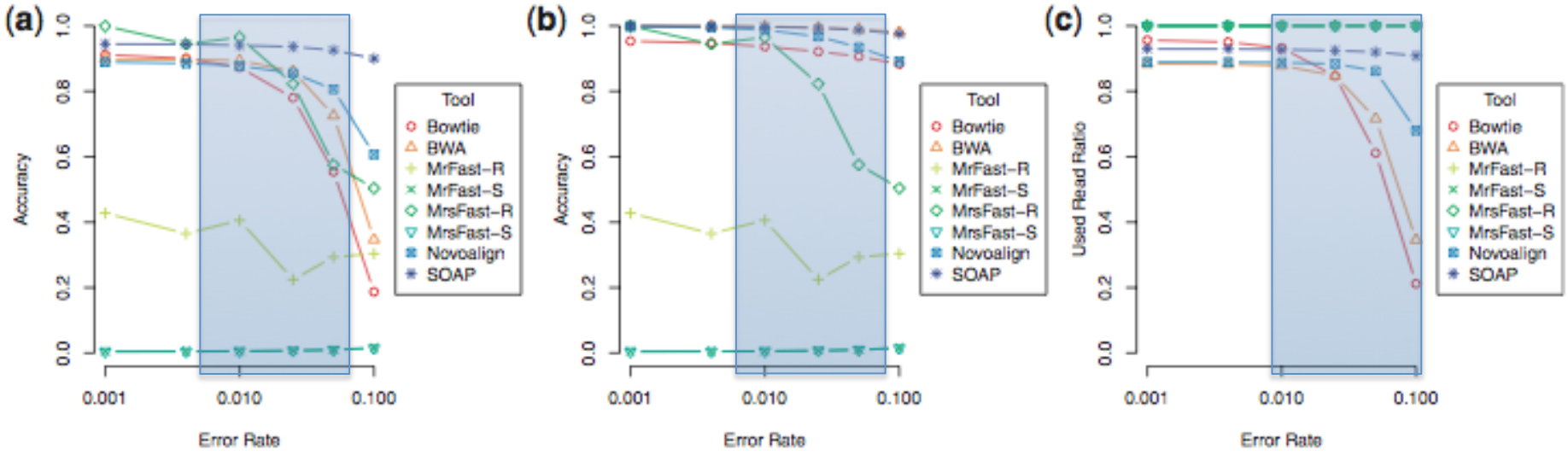
Accuracy with varying error rate

With mapping quality threshold 10

Accuracy

Accuracy

Used read ratio



- No indels
- Higher range of error rate is unrealistic
- Minor differences between good algorithms in range

Source: Ruffalo et al. 2011

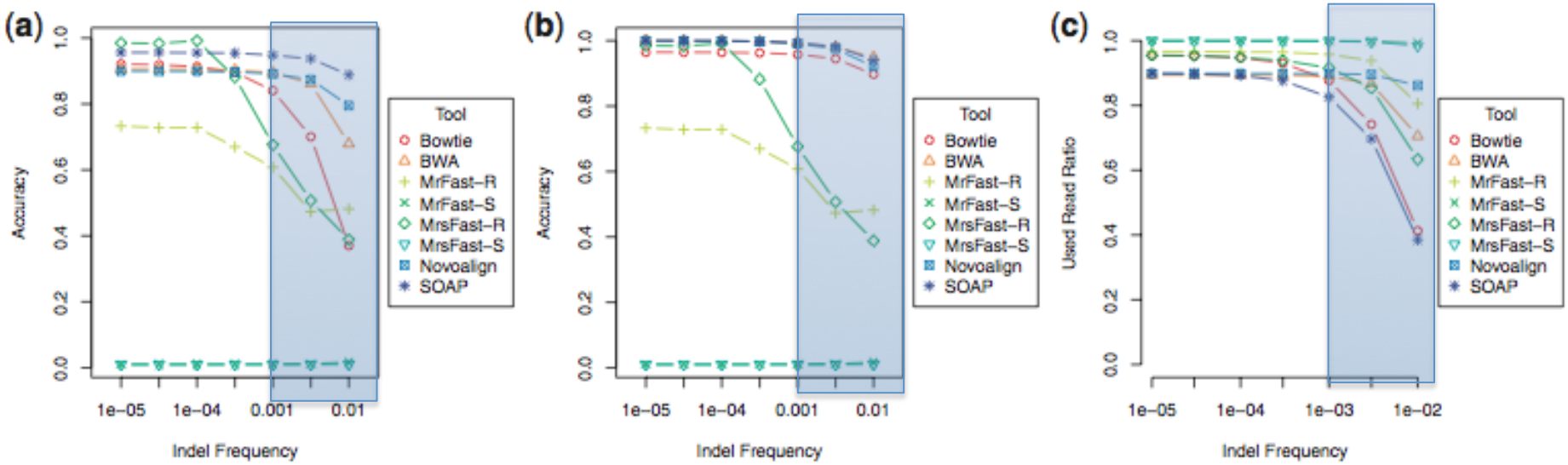
Accuracy with varying indel frequency

With mapping quality threshold 10

Accuracy

Accuracy

Used read ratio



- indel size fixed at 2 ➔ small
- error rate fixed at 0.01 ➔ somewhat high
- only lower indel frequencies are realistic ➔ small differences between programs in relevant range
- soap would have looked a lot worse if indel size had been bigger

Source: Ruffalo et al. 2011