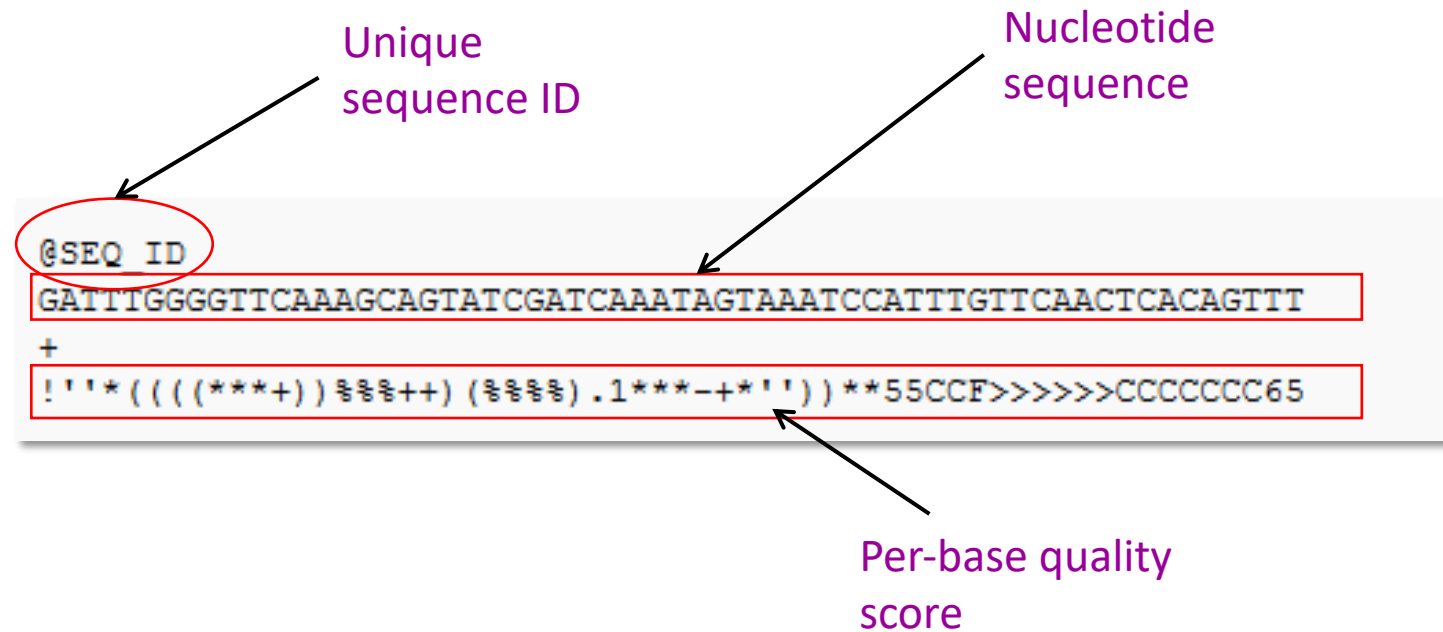


# Working with Genomic file formats on the Linux command line

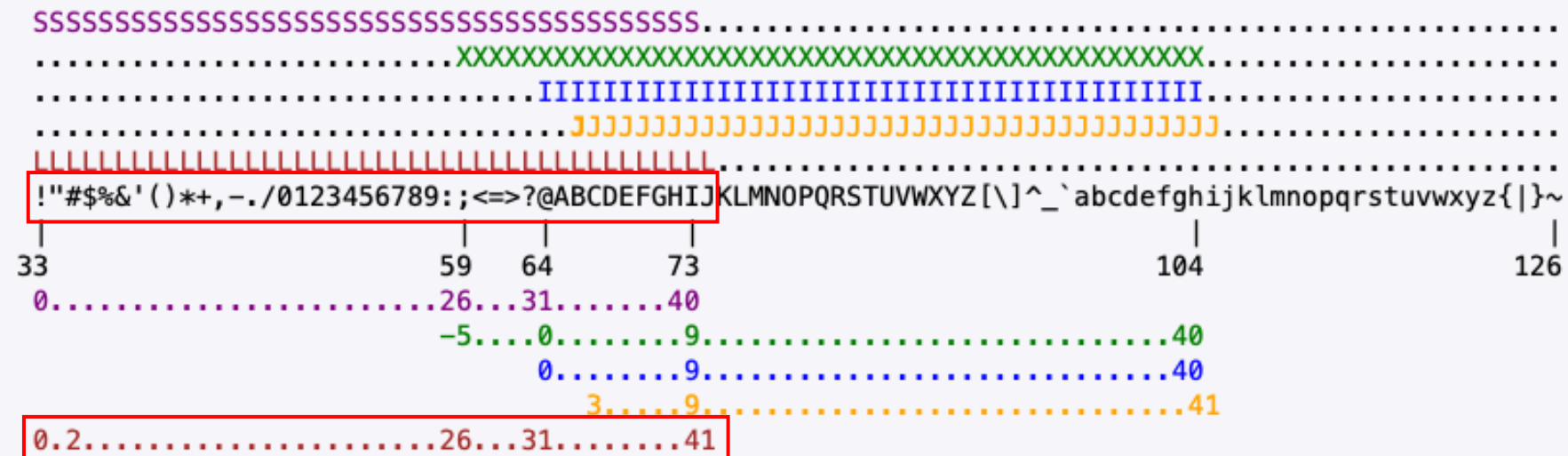
Oxford Biomedical Data Science Training Programme

# FASTQ files



Different platforms use different symbols (ASCII characters) to encode quality scores

# FASTQ files

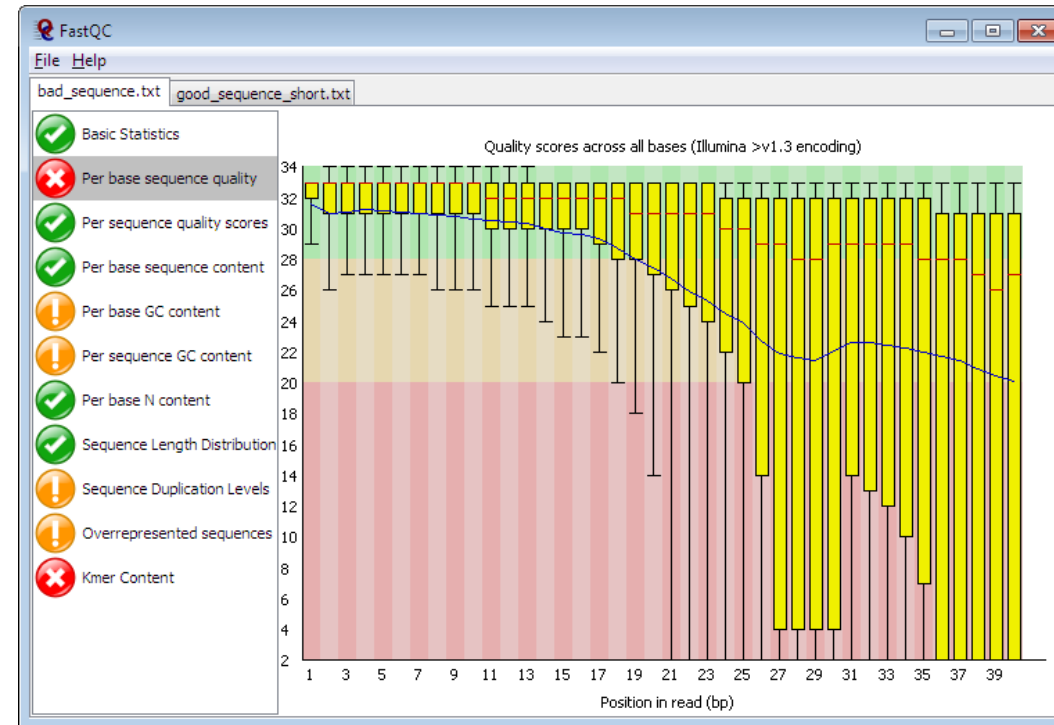


S - Sanger Phred+33, raw reads typically (0, 40)  
X - Solexa Solexa+64, raw reads typically (-5, 40)  
I - Illumina 1.3+ Phred+64, raw reads typically (0, 40)  
J - Illumina 1.5+ Phred+64, raw reads typically (3, 41)  
with 0=unused, 1=unused, 2=Read Segment Quality Control Indicator (bold)  
(Note: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

# FastQC - Read Quality Control

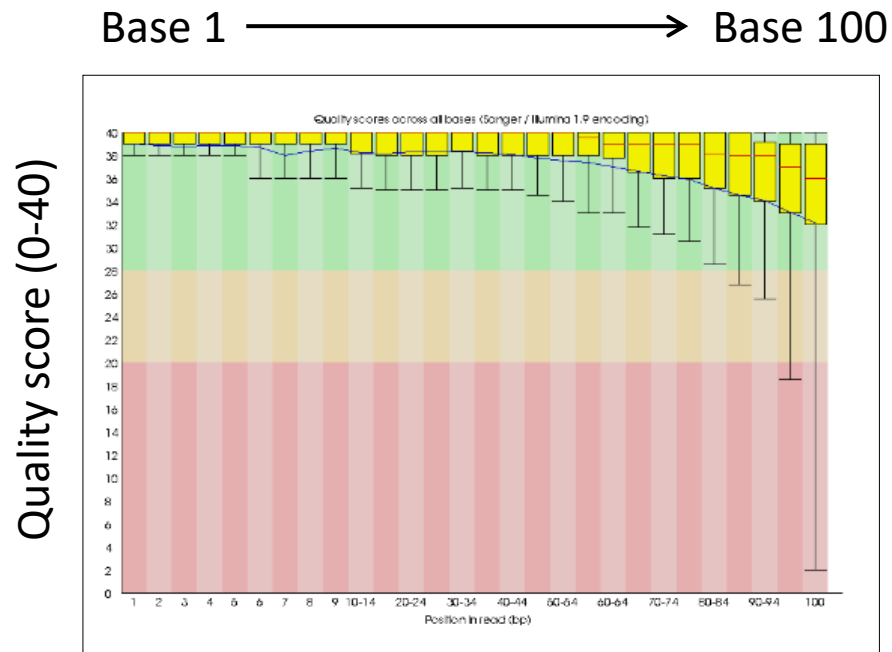


- Traffic light overview
- Graphical summaries
- HTML report



<http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

# Per-base sequence quality



## Error probability

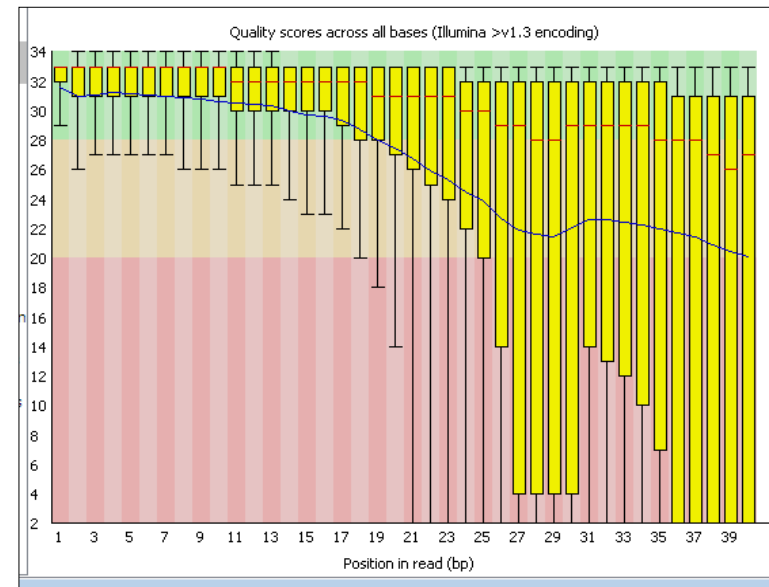
Q40 = 1 in 10,000

Q30 = 1 in 1000

Q20 = 1 in 100

Q10 = 1 in 10

## Possible to trim reads by quality

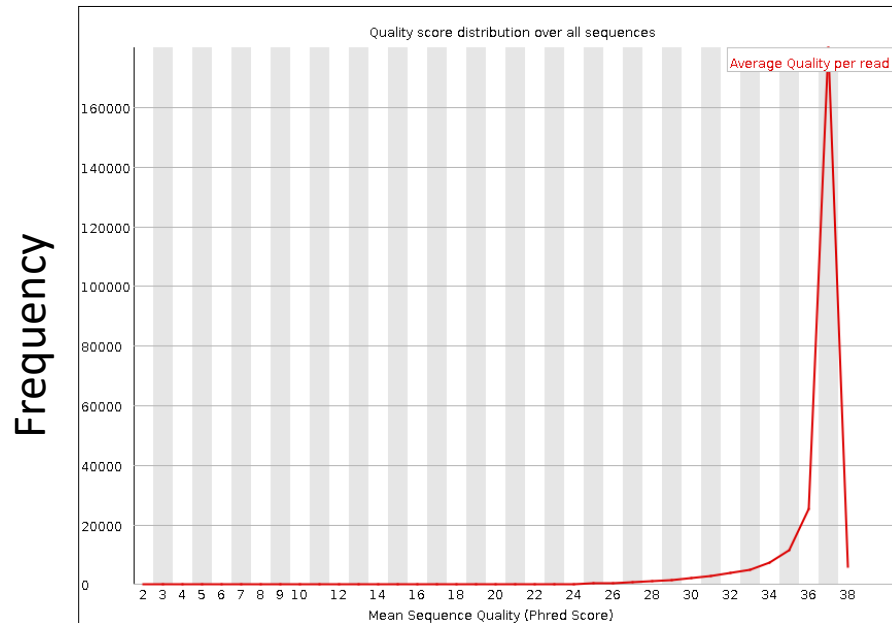


Quality score > 30 = Good (green zone)

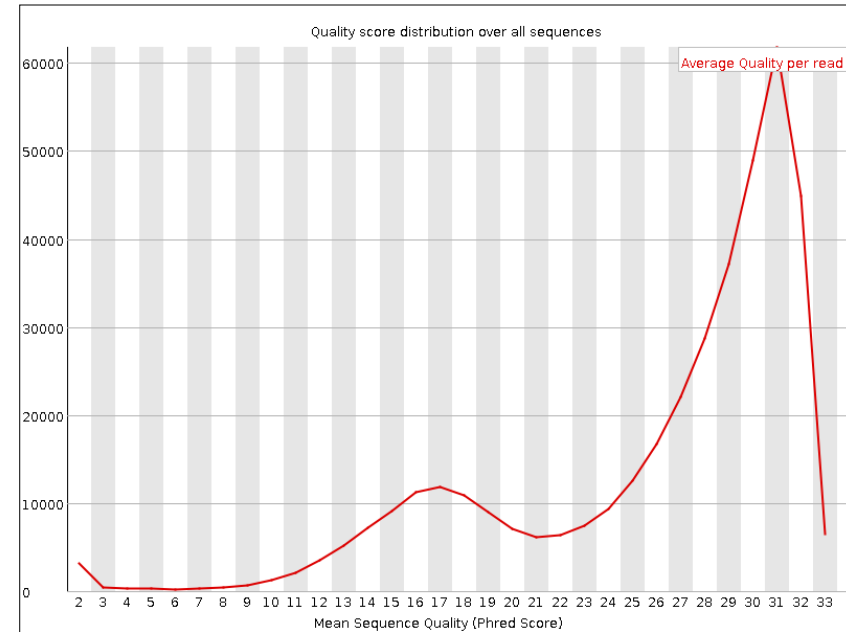
Quality score > 20 = OK (orange zone)

Quality score < 20 = Poor (red zone)

# Average sequence quality



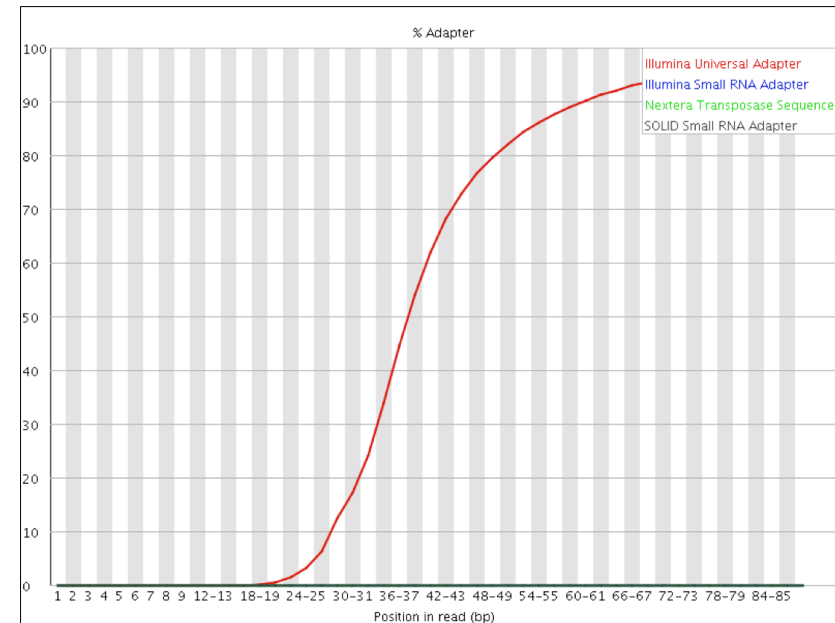
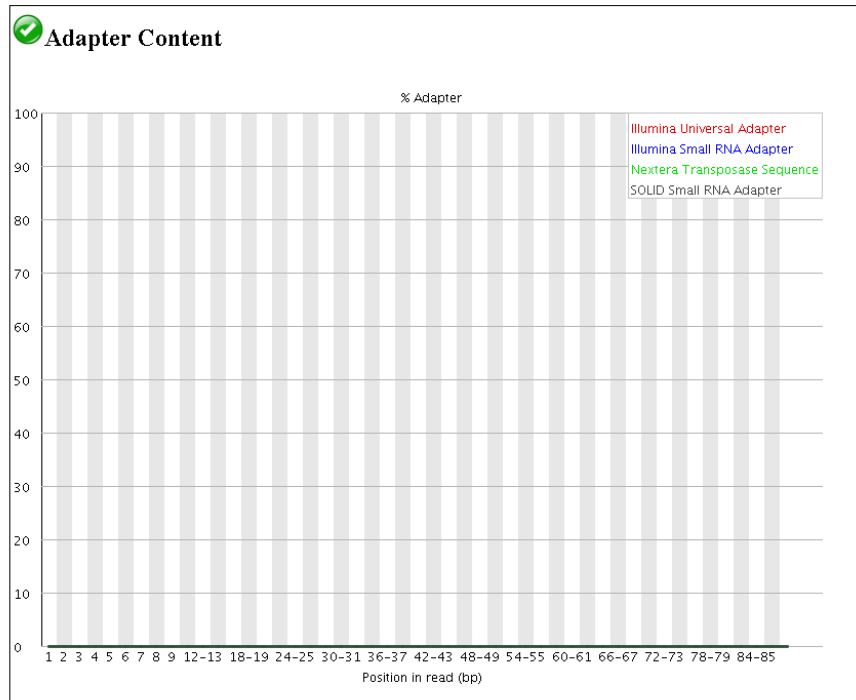
Average sequence quality = 37



Average sequence quality = 31 with additional low quality sequences at 17

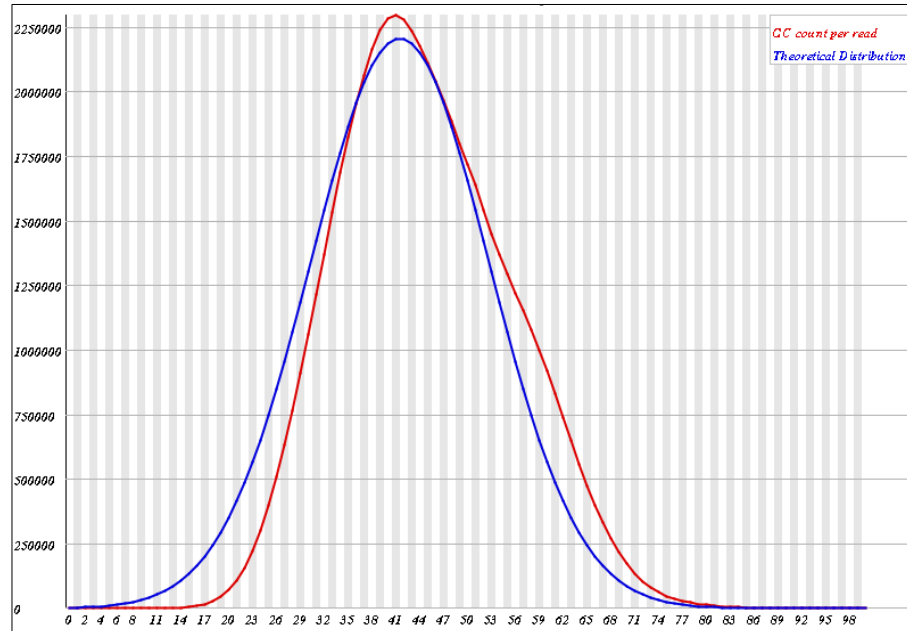
- Low quality reads can be removed by trimming by quality
- Reads trimmed below a minimum length are discarded
- Trimmomatic/Trim Galore/Cutadapt

# Sequencing adapter contamination



- If read length is greater than insert size then the sequencer will begin to sequence into the adapter
- Adapters can be removed by trimming using Trimmomatic etc.

# GC content



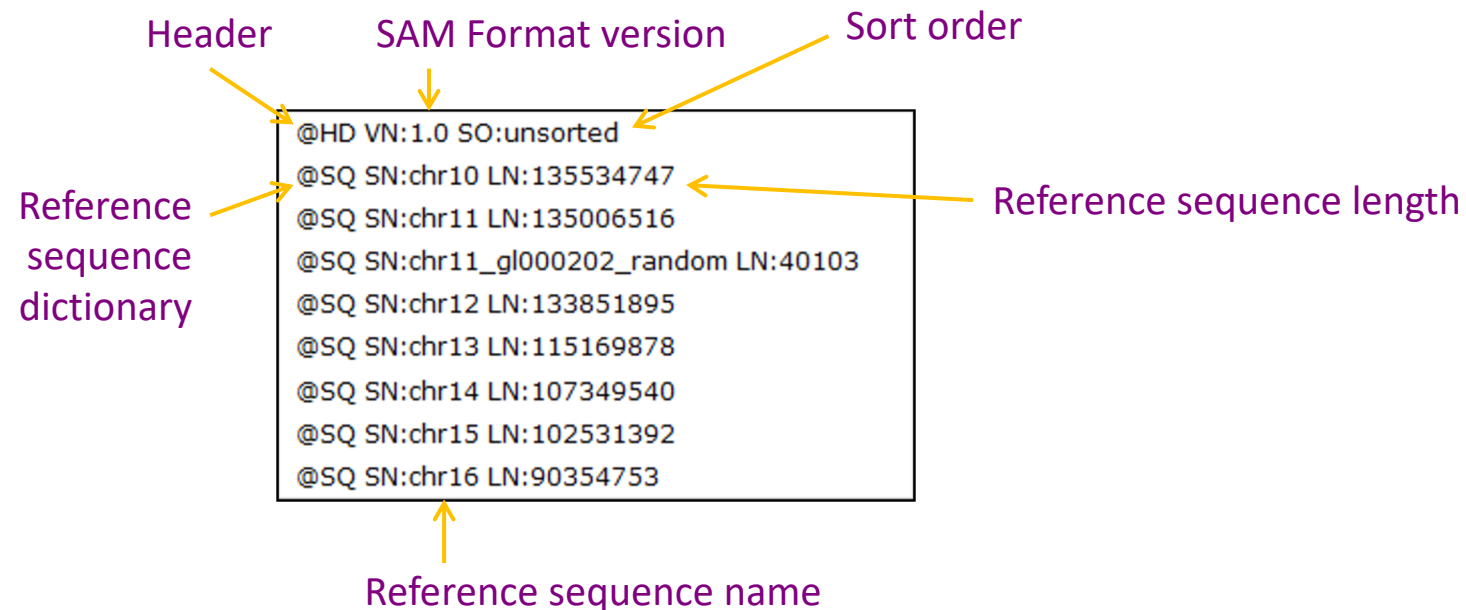
GC count per read  
Theoretical distribution

- May see some bias in exome/amplicon data as many functional regions of the genome have increased GC content compared to genome average



# SAM format

- Sequencing Alignment Map
- Contains alignment information
- Standardised text file format – headers and entries



# SAM format

1	2	3	4	5	6	7	8	9	10	11
SRR502532.86625	0	chrX	95360010	255	27M	*	0	0	CACTTGTAATAATTTAGCTGTGCTAAAT	@>CAAC@77A4@ACCB9BCA)CC>77?
SRR502532.87075	0	chr6	115961470	255	27M	*	0	0	TAATTTTAGCAAAAAACACAACACT	B?ABCBB<BB;B5+=B5ACC?CBCCCC
SRR502532.87077	16	chr14	59150304	255	27M	*	0	0	CACTTGGGCCAATGAAACAAAGGTGAA	CBB>BA-?C87;(C75BCCCCC6:CBA
SRR502532.87232	16	chr3	111799551	255	27M	*	0	0	ATGAATGCAGCTGTTGATTATAGGAA	BCCCC=5CC959>C7:CB;?CB<@CCB
SRR502532.87586	0	chrX	71644230	255	27M	*	0	0	TAATTATTTCTTGATAATTCCTAGC	@6=A>-ACBCCB@CB?9:?B4CCCACC
SRR502532.86839 COLUMBO:4:14:78:209/1	4	*	0	0	*	*	0	0	TGAAAGGGACAGTATATTGGTGGGGGG	BA2=ACC@AA=5>3BAAB@;C5C=@@

1. QNAME: Query template NAME
2. FLAG: bitwise FLAG
3. RNAME: Reference sequence NAME
4. POS: 1-based leftmost mapping POSition
5. MAPQ: MAPping Quality
6. CIGAR: CIGAR string
7. RNEXT: Ref. name of the mate/next segment
8. PNEXT: Position of the mate/next segment
9. TLEN: observed Template LENgth
10. SEQ: segment SEQuence
11. QUAL: ASCII of Phred-scaled base QUALity + 33

# SAM bitwise flags

Bit	Description
0x1	template having multiple segments in sequencing
0x2	each segment properly aligned according to the aligner
0x4	segment unmapped
0x8	next segment in the template unmapped
0x10	SEQ being reverse complemented
0x20	SEQ of the next segment in the template being reversed
0x40	the first segment in the template
0x80	the last segment in the template
0x100	secondary alignment
0x200	not passing quality controls
0x400	PCR or optical duplicate
0x800	supplementary alignment

## Decoding SAM flags

This utility makes it easy to identify what are the properties of a read based on its SAM flag value, or conversely, to find what the SAM Flag value would be for a given combination of properties.

To decode a given SAM flag value, just enter the number in the field below. The encoded properties will be listed under Summary below, to the right.

SAM Flag:  Explain

Switch to mate Toggle first in pair / second in pair

### Find SAM flag by property:

### Summary:

To find out what the SAM flag value would be for a given combination of properties, tick the boxes for those that you'd like to include. The flag value will be shown in the SAM Flag field above.

- ☐ read paired
- ☐ read mapped in proper pair
- ☐ read unmapped
- ☐ mate unmapped
- ☐ read reverse strand
- ☐ mate reverse strand
- ☐ first in pair
- ☐ second in pair
- ☐ not primary alignment
- ☐ read fails platform/vendor quality checks
- ☐ read is PCR or optical duplicate
- ☐ supplementary alignment

<https://broadinstitute.github.io/picard/explain-flags.html>

# SAM bitwise flags

SAM flag = 67

**Summary:**

read paired (0x1)

read mapped in proper pair (0x2)

first in pair (0x40)

<https://broadinstitute.github.io/picard/explain-flags.html>

# BAM/CRAM Format

## BAM:

- Binary Alignment Map
- BGZF compressed
- Sorted and indexed
- Random access
- Gunzip compatible
- EOF marker

## CRAM:

- Reference-based compression
- 40-50% space saving over BAM
- BAM compatible
- Sorted and indexed
- Random access

# Samtools

- Suite of programs for interacting with NGS sequencing data
- Based on HTSlib C library
- Runs on Linux command line
- Works with SAM/BAM/CRAM files

```
samtools view example.bam | head
```

```
samtools idxstats example.bam > example.stats
```

# Exercise 1 – SAM/BAM files

- Load Samtools (should be in your Python Conda environment)
- Convert example.sam to example.bam (in shared/week1/rnaseq)
- Index the BAM file
- View the SAM header from example.bam
- View all of the alignments from example.bam on chr7










# BED file format

The first three required BED fields are:

1. **chrom** - The name of the chromosome (e.g. chr3, chrY, chr2\_random) or scaffold (e.g. scaffold10671).
2. **chromStart** - The starting position of the feature in the chromosome or scaffold. The first base in a chromosome is numbered 0.
3. **chromEnd** - The ending position of the feature in the chromosome or scaffold. The *chromEnd* base is not included in the display of the feature. For example, the first 100 bases of a chromosome are defined as *chromStart*=0, *chromEnd*=100, and span the bases numbered 0-99.

The 9 additional optional BED fields are:

4. **name** - Defines the name of the BED line. This label is displayed to the left of the BED line in the Genome Browser window when the track is open to full display mode or directly to the left of the item in pack mode.
5. **score** - A score between 0 and 1000. If the track line *useScore* attribute is set to 1 for this annotation data set, the *score* value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). This table shows the Genome Browser's translation of BED score values into shades of gray:

shade									
score in range	≤ 166	167-277	278-388	389-499	500-611	612-722	723-833	834-944	≥ 945

6. **strand** - Defines the strand - either '+' or '-'.
7. **thickStart** - The starting position at which the feature is drawn thickly (for example, the start codon in gene displays).
8. **thickEnd** - The ending position at which the feature is drawn thickly (for example, the stop codon in gene displays).
9. **itemRgb** - An RGB value of the form R,G,B (e.g. 255,0,0). If the track line *itemRgb* attribute is set to "On", this RGB value will determine the display color of the data contained in this BED line. NOTE: It is recommended that a simple color scheme (eight colors or less) be used with this attribute to avoid overwhelming the color resources of the Genome Browser and your Internet browser.
10. **blockCount** - The number of blocks (exons) in the BED line.
11. **blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to *blockCount*.
12. **blockStarts** - A comma-separated list of block starts. All of the *blockStart* positions should be calculated relative to *chromStart*. The number of items in this list should correspond to *blockCount*.



# GFF format

## GFF format

Index ▷

GFF (General Feature Format) lines are based on the GFF standard file format. GFF lines have nine required fields that *must* be tab-separated. If the fields are separated by spaces instead of tabs, the track will not display correctly. For more information on GFF format, refer to <http://www.sanger.ac.uk/resources/software/gff/>.

If you would like to obtain browser data in GFF (GTF) format, please refer to [Genes in gtf or gff format](#) on the Wiki.

Here is a brief description of the GFF fields:

1. **seqname** - The name of the sequence. Must be a chromosome or scaffold.
2. **source** - The program that generated this feature.
3. **feature** - The name of this type of feature. Some examples of standard feature types are "CDS", "start\_codon", "stop\_codon", and "exon".
4. **start** - The starting position of the feature in the sequence. The first base is numbered 1.
5. **end** - The ending position of the feature (inclusive).
6. **score** - A score between 0 and 1000. If the track line *useScore* attribute is set to 1 for this annotation data set, the *score* value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). If there is no score value, enter ".".
7. **strand** - Valid entries include '+', '-', or '.' (for don't know/don't care).
8. **frame** - If the feature is a coding exon, *frame* should be a number between 0-2 that represents the reading frame of the first base. If the feature is not a coding exon, the value should be '.'.
9. **group** - All lines with the same group are linked together into a single item.

# GTF format

## GTF format

[Index ▸](#)

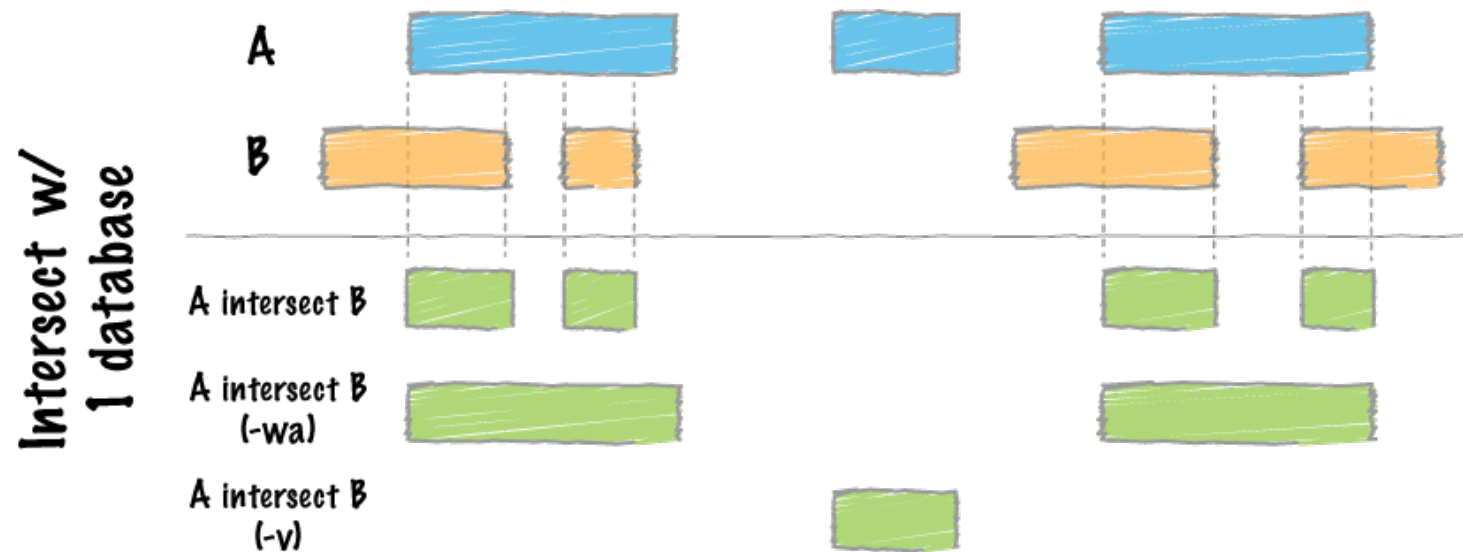
GTF (Gene Transfer Format) is a refinement to GFF that tightens the specification. The first eight GTF fields are the same as GFF. The *group* field has been expanded into a list of *attributes*. Each attribute consists of a type/value pair. Attributes must end in a semi-colon, and be separated from any following attribute by exactly one space.

The attribute list must begin with the two mandatory attributes:

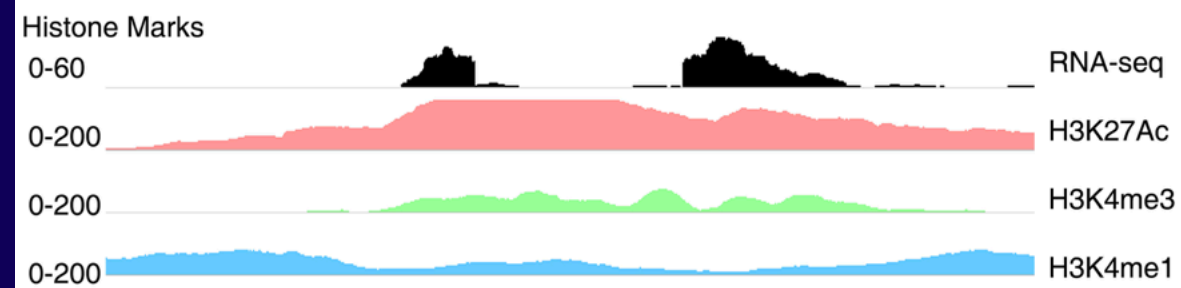
- **gene\_id value** - A globally unique identifier for the genomic source of the sequence.
- **transcript\_id value** - A globally unique identifier for the predicted transcript.

# Bedtools

- Intersect, merge, count, complement and shuffle genomic intervals
- Supports BAM, BED, GTF & VCF formats



# Wiggle files



- Designed for genome browser display of dense, continuous data
  - GC percent, probability scores and transcriptome data
- Composed of declaration lines and data lines

variableStep format

```
variableStep chrom=chr2 span=10  
300701 12.5  
300711 15.5
```

Declaration line

Data lines

fixedStep format

```
fixedStep chrom=chr3 start=400601 step=100  
11  
22  
33
```

Declaration line

Data lines

<https://genome.ucsc.edu/goldenPath/help/wiggle.html>

# BigWig format

- Indexed binary wiggle format
  - Compressed
  - Random access
- Create from Wig files using wigToBigWig
  - Part of UCSC tools
  - Memory intensive

# Exercise 2 – BED/GTF and Wig files

- Bedtools online tutorial
  - Bedtools should be installed within your Python Conda environment
  - <http://quinlanlab.org/tutorials/bedtools/bedtools.html>