# What bests describes your experience with sequence and alignment data?
# Ex: FASTA, FASTQ, SAM, BAM file formats

a) This is the first time I've heard about them!
b) This sounds vaguely familiar but tell me more!
c) I've played around with a few files, and I'd like to learn more!
d) I use these files in my analyses all the time and I'm excited to share my expertise!

# Sequence & Alignment Data

Buffalo Chapters 10 & 11

https://github.com/vsbuffalo/bds-files

## Goals for today

1) Define, compare, and contrast 4 common file formats
2) Evaluate quality and attributes of the data within each file format
3) Identify key steps, methods, and best practices for quality control while working with these data types

# Sequence Data

FASTA and FASTQ file formats

# FASTA – uses

- FASTA file format developed: http://fasta.bioch.virginia.edu/fasta_www2/fasta_list2.shtml
- FASTA files store many data types:
  - reference genome files
  - protein and transcript sequences
  - alignments

# FASTA – format

Each sequence in a FASTA file is represented by 2 parts:

1. The description/identifier line starting with >
2. The sequence (can be a single line or interleaved)

```
head -n 10 egfr_flanks.fasta
```

# FASTA – format

```
>ENSMUSG00000020122|ENSMUST00000138518
> ENSMUSG00000020122|ENSMUST00000125984
>ENSMUSG00000020122|ENSMUST00000125984|epidermal growth factor receptor
>ENSMUSG00000020122|ENSMUST00000125984|Egfr
>ENSMUSG00000020122|ENSMUST00000125984|11|ENSFM00410000138465
```

- ❑ How might this cause issues when you're trying to parse the information on the identifier line?
- ❑ What steps should you take when building a new fasta file?
- ❑ What steps should you take when working with a new fasta file?

# FASTQ – format

FASTQ stores the FASTA-formatted sequence as well as the associated quality scores

```
head -n 10 contam.fastq
```

# FASTQ – format

FASTQ stores the FASTA-formatted sequence as well as the associated quality scores

```
head -n 10 contam.fastq
```

Each entry typically has 4 parts:

1. A header line beginning with @ containing the record identifier and other information.
2. The DNA sequence (can be on 1 or many lines)
3. A line beginning with just +
4. Quality scores for each base encoded in ASCII format which are same length as sequence.

# Chat Check-in

Let's look at `untreated1_chr4.fq`

There can be issues when parsing FASTQ files on different ASCII characters

@ denotes the header line

```
grep -c "^@" untreated1_chr4.fq
```

1. Does this result make sense? Why or why not?
2. If it doesn't make sense, what's causing this result?

# FASTQ – handling tip

A better solution is to use bioawk to determine the number of FASTQ entries:

```
module load bioawk
bioawk -cfastx 'END{print NR}' untreated1_chr4.fq
```

https://bioinformaticsworkbook.org/Appendix/Unix/bioawk-basics.html#gsc.tab=0

# FASTQ – Interpreting quality scores

## ASCII TABLE

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | [NULL] | 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 1 | 1 | [START OF HEADING] | 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 2 | 2 | [START OF TEXT] | 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 3 | 3 | [END OF TEXT] | 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 4 | 4 | [END OF TRANSMISSION] | 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 5 | 5 | [ENQUIRY] | 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 6 | 6 | [ACKNOWLEDGE] | 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 7 | 7 | [BELL] | 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 8 | 8 | [BACKSPACE] | 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 9 | 9 | [HORIZONTAL TAB] | 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 10 | A | [LINE FEED] | 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 11 | B | [VERTICAL TAB] | 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 12 | C | [FORM FEED] | 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 13 | D | [CARRIAGE RETURN] | 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 14 | E | [SHIFT OUT] | 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 15 | F | [SHIFT IN] | 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 16 | 10 | [DATA LINK ESCAPE] | 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 17 | 11 | [DEVICE CONTROL 1] | 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 18 | 12 | [DEVICE CONTROL 2] | 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 19 | 13 | [DEVICE CONTROL 3] | 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 20 | 14 | [DEVICE CONTROL 4] | 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 21 | 15 | [NEGATIVE ACKNOWLEDGE] | 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 22 | 16 | [SYNCHRONOUS IDLE] | 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 23 | 17 | [ENG OF TRANS. BLOCK] | 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 24 | 18 | [CANCEL] | 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 25 | 19 | [END OF MEDIUM] | 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 26 | 1A | [SUBSTITUTE] | 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 27 | 1B | [ESCAPE] | 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 28 | 1C | [FILE SEPARATOR] | 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 29 | 1D | [GROUP SEPARATOR] | 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 30 | 1E | [RECORD SEPARATOR] | 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 31 | 1F | [UNIT SEPARATOR] | 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

# FASTQ – Interpreting quality scores

Platform specific conversions!

**The quality score is the probability that a given base call is incorrect**

| Name | ASCII character range | Offset | Quality score type | Quality score range |
|---|---|---|---|---|
| Sanger, Illumina (versions 1.8 onward) | 33–126 | 33 | PHRED | 0–93 |
| Solexa, early Illumina (before 1.3) | 59–126 | 64 | Solexa | 5–62 |
| Illumina (versions 1.3–1.7) | 64–126 | 64 | PHRED | 0–62 |

# FASTQ – Interpreting quality scores example

We can use python to translate the following quality score string from a FASTQ file to a decimal value

```
python
>>> qual = "JJJJJJJJJJJJGJJJJJIIJJJJJIGJJJJJIJJJJJJJIJIJJJJHHHHHFFFDFCCC"
>>> [ord(b) for b in qual]
```

Use the table to convert these Illumina 1.8 data to PHRED quality scores:

```
>>> phred = [ord(b)-33 for b in qual]
```

Follow the equation for PHRED scores to calculate the probability that each base is incorrect:

```
>>> pr_phred = [10**(-q/10) for q in phred]
```

# Chat Check-in

What do you notice about the quality scores across the sequence?

# BREAK OUT DISCUSSION

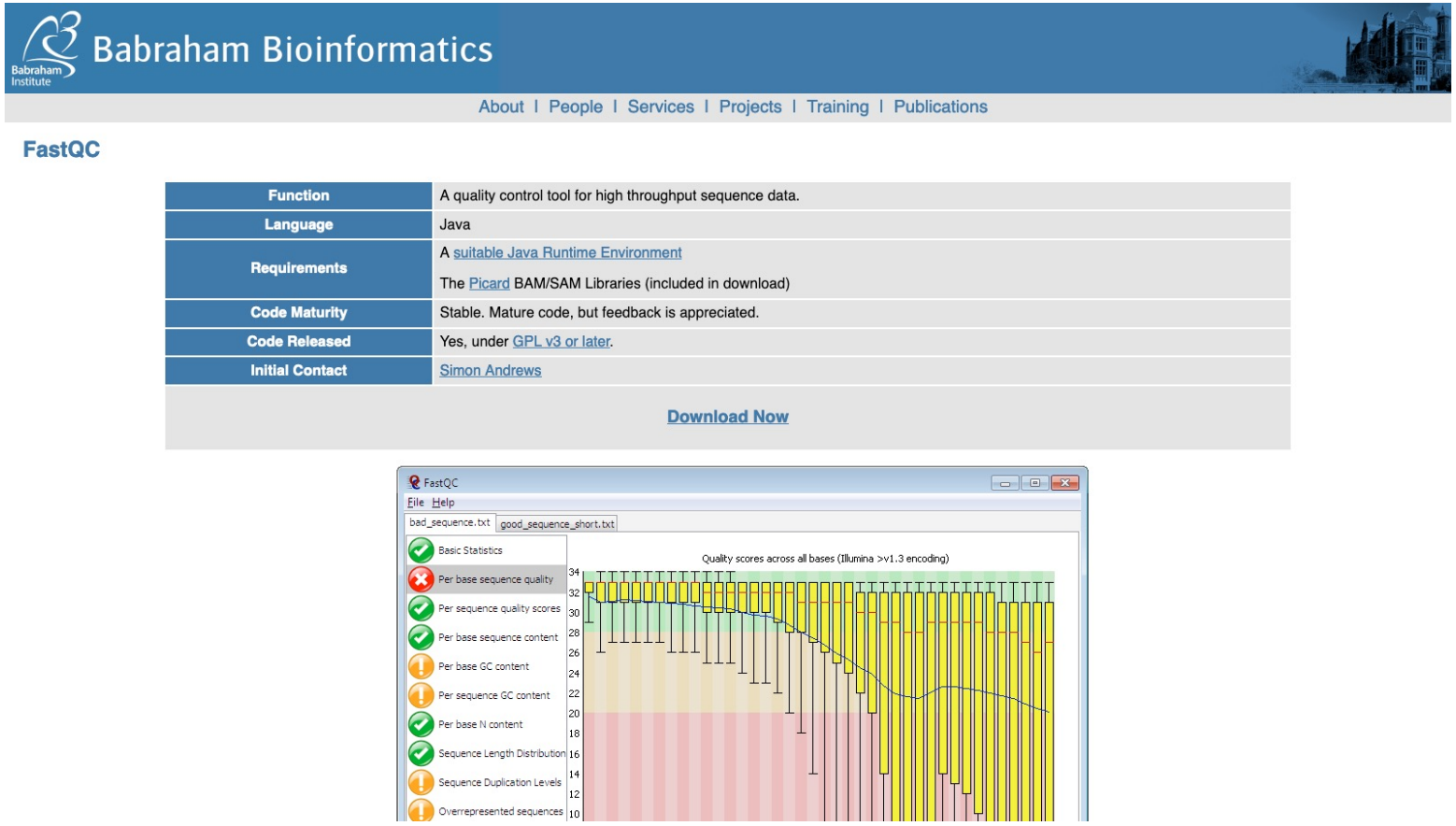
Watch this video:
https://www.youtube.com/watch?annotation_id=annotation_2285758 61&feature=iv&src_vid=womKfikWlxM&v=fCd6B5HRaZ8

1. Does this help explain the trend we saw in the previous example?
2. What is the advantage of generating sequencing foci through bridge amplification?
3. Why would you sequence paired end reads in tandem rather than simultaneously?
4. What would happen if templates in a sequencing focus fell out of phase?

# FASTQ – Evaluating quality within a file

`FastQC` is a program that summarizes and creates ways for visual inspection of a FASTQ file

https://www.bioinformatics.babraham.ac.uk/projects/fastqc/



# FASTQ – Evaluating quality within a file

```
module load fastqc
fastqc untreated1_chr4.fq
```

# FASTQ -- Evaluating quality within a file

```
module load sickle
sickle se -f untreated1_chr4.fq -t sanger
          -o untreated1_chr4_sickle.fq
```

```
module load seqtk
seqtk trimfq untreated1_chr4.fq >
             untreated1_chr4_trimfq.fq
```

What was the effect of the trimming on overall quality within the fastq?

# Alignment Data

SAM and BAM files

# Alignment Data – uses

Many applications of next-generation sequencing involve mapping reads to a reference genome

Can you describe a few?

# Alignment Data – uses

- Mapping reads produces an alignment of these reads onto a reference coordinate system
- Alignments are most frequently formatted as what are known as "SAM" and "BAM" files
- These files can be massive in size, and efficient code to extract information is a must!

# SAM – format

**@SQ entries** include information about sequences in the alignment including their name (SN) and length (LN)

**@RG entries** include information about the data and sample generated including the batch of sequence (ID), the individual sequenced (SM), and the platform used for sequencing (PL)

**@PG entries** include metadata about the programs used to create and process the SAM/BAM files

```
head -n 10 celegans.sam
```

# SAM, BAM – handling header metadata

`samtools` is a package designed to efficiently handle, parse, and format sam and bam files

```
module load samtools
samtools view -H celegans.sam
samtools view -H celegans.bam
```

How is the output of these commands different from:

```
head celegans.sam
```

# SAM, BAM – handling header metadata

`samtools` can also be paired with other unix commands with the pipe

```
samtools view -H celegans.bam | grep "^@RG"
```

# SAM, BAM – alignment format

Now that we have a good feel for the header, let's take a look at the alignment portion of SAM/BAM files:

```
samtools view celegans.sam | head -n 1
```

This is a little difficult to make sense of, so let's replace tabs with returns so can individually inspect each field:

```
samtools view celegans.sam | tr '\t' '\n’ |
head -n 11
```

# SAM, BAM – alignment format

1. I_2011868_2012306_0:0:0_0:0:0_2489 (sequence read name)
2. 83 (bit flag)
3. I (reference name, e.g., chromosome I)
4. 2012257 (leftmost base position of the sequence)
5. 40 (mapping quality)
6. 50M (CIGAR string)
7. = (RNEXT, reference name)
8. 2011868 (PNEXT, reference position)
9. -439 (TLEN, template length for paired-end reads)
10. CAAAAAATTTTGAAAAAAAAAATTGAATAAAAATTCACGGATTTCTGGCT
11. 22222222222222222222222222222222222222222222222

# SAM, BAM – alignment format

1. I_2011868_2012306_0:0:0_0:0:0_2489 (sequence read name)
2. **83 (bit flag)**
3. I (reference name, e.g., chromosome I)
4. 2012257 (leftmost base position of the sequence)
5. 40 (mapping quality)
6. **50M (CIGAR string)**
7. = (RNEXT, reference name)
8. 2011868 (PNEXT, reference position)
9. -439 (TLEN, template length for paired-end reads)
10. CAAAAAATTTTGAAAAAAAAAATTGAATAAAAATTCACGGATTTCTGGCT
11. 22222222222222222222222222222222222222222222222

# SAM, BAM – bitwise flags

Encode many attributes of the sequences in an alignment:

- unmapped
- paired-end
- aligned in reverse complement
- QC failure
- etc...

- How can we tell what a bitwise flag means?

```
samtools flags 83
```

- What if we want to find a flag for a particular set of attributes?

```
samtools flags paired,read1,qcfail
```

# SAM, BAM – CIGAR Strings

- Similar to bitwise flags
- 0-based indexing
- Base-pair index position + operation
- The total should add up to the length of a given sequence

*Table 11-2. CIGAR operations*

| Operation | Value | Description |
|---|---|---|
| M | 0 | Alignment match (note that this could be a sequence match or mismatch!) |
| I | 1 | Insertion (to reference) |
| D | 2 | Deletion (from reference) |
| N | 3 | Skipped region (from reference) |
| S | 4 | Soft-clipped region (soft-clipped regions are present in sequence in SEQ field) |
| H | 5 | Hard-clipped region (not in sequence in SEQ field) |
| P | 6 | Padding (see section 3.1 of the SAM format specification for detail) |
| = | 7 | Sequence match |

# SAM, BAM – CIGAR Strings Example

Based on the table, what does the following CIGAR String tell you about a given sequence?

43S6M1I26M

- 43S
- 6M
- 1I
- 26M

*Table 11-2. CIGAR operations*

| Operation | Value | Description |
|---|---|---|
| M | 0 | Alignment match (note that this could be a sequence match or mismatch!) |
| I | 1 | Insertion (to reference) |
| D | 2 | Deletion (from reference) |
| N | 3 | Skipped region (from reference) |
| S | 4 | Soft-clipped region (soft-clipped regions are present in sequence in SEQ field) |
| H | 5 | Hard-clipped region (not in sequence in SEQ field) |
| P | 6 | Padding (see section 3.1 of the SAM format specification for detail) |
| = | 7 | Sequence match |

# SAM, BAM – Evaluating and Handling Example

We want to know what the relative proportions of mapped and properly paired reads to unmapped reads occur in an alignment of human data

1. What are the bitwise flags for both mapped/properly paired and unmapped reads? (Proper_Pair, UNMAP)
2. Now let's use these flags and the view command in samtools to find the relative proportion:

```
samtools view -f 4 NA12891_CEU_sample.bam | wc -l
samtools view -f 2 NA12891_CEU_sample.bam | wc -l
```

# More things you can do with samtools!

- ❑ Extraction of particular regions of an alignment
- ❑ Viewing of alignment
- ❑ Variant calling
- ❑ etc...

There’s further description in Buffalo Chapter 11!

# Questions?

Thank you for your time and attention!