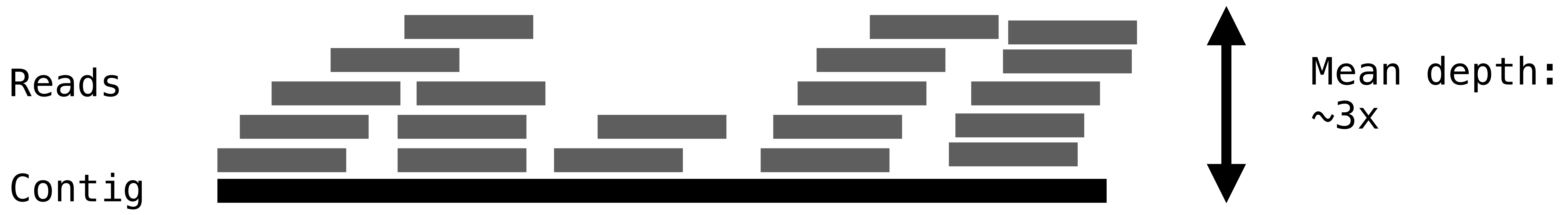
High-throughput alignments

SAM and BAM

Jakob Nissen, 2021-10-21

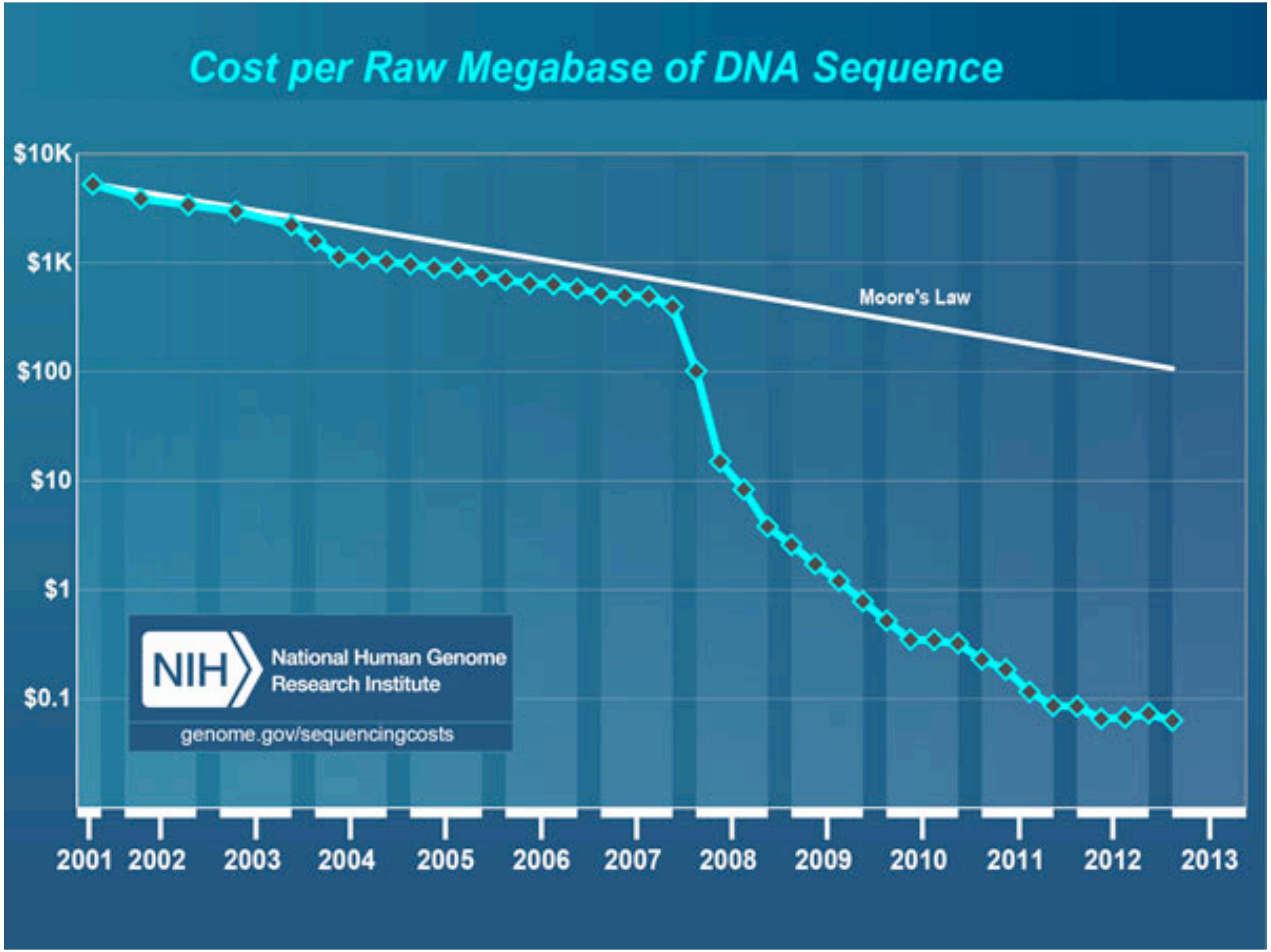
Overview

- Previously: Small reads -> assembly -> genome
- But genome assembly is not the most common use of small reads!
    - Genome assembly is computationally very hard
    - It's been studied for decades, still error prone
- Usually, you align reads to an existing genome. Much easier!



What's the problem with using S/W for this?

Remember the most abused slide?



Problems

- S/W algorithm scales with length of reference and query
  - Align to human chromosome 1 - 250M bases!
- Easily millions of reads
- FASTA output format: Completely infeasible!

```
>read
------------------------------- [ 250 mill ... ]
>chrom1
NNNNNATTCGGAGTCGTATTAGGGAGAGCGA [ 250 mill ... ]
```

Solution 1: Seed and extend

```
                          T A G G T C G T G A T G A T G A G G C
00000000000000000000000000110010101101101110001110001110001010 01
```

- Remember how kmers could be represented by a machine integer?
- Matching two kmers is a single CPU instruction, < 1 nanosecond
- Alignment approach: Match kmers between reference and query, and use them as *seeds*.
    - Around each seed, align using S/W
- Most common algorithms are BLAST (1990). BWA (2009?), minimap2 (2018), KMA (2018)
    - All of these use seed & extend

Solution 2: SAM format

- We need a more efficient file format for alignments than FASTA!
- Sequence Alignment/Map format
    - Complicated format - we'll just go through the important parts
- First a header
    - A sequence of header rows
        - Header row: @ + 2 characters + \t + fields
            - Field: 2 characters + : + data

```
@HD	VN:1.6	SO:coordinate
@SQ	SN:SEQHEADER	LN:1501
```

Solution 2: SAM format

- Then each alignment on its own line.
- Fields separated by a tab (\t)
  - Query identifier
  - Sam flags
  - Reference identifier
  - 1-based leftmost mapping position
  - Mapping quality
  - CIGAR string
  - Ref of next query if query is grouped
  - Pos of next query if query is grouped
  - Template length
  - Query sequence
  - Sequence ASCII quality
  - Optional fields

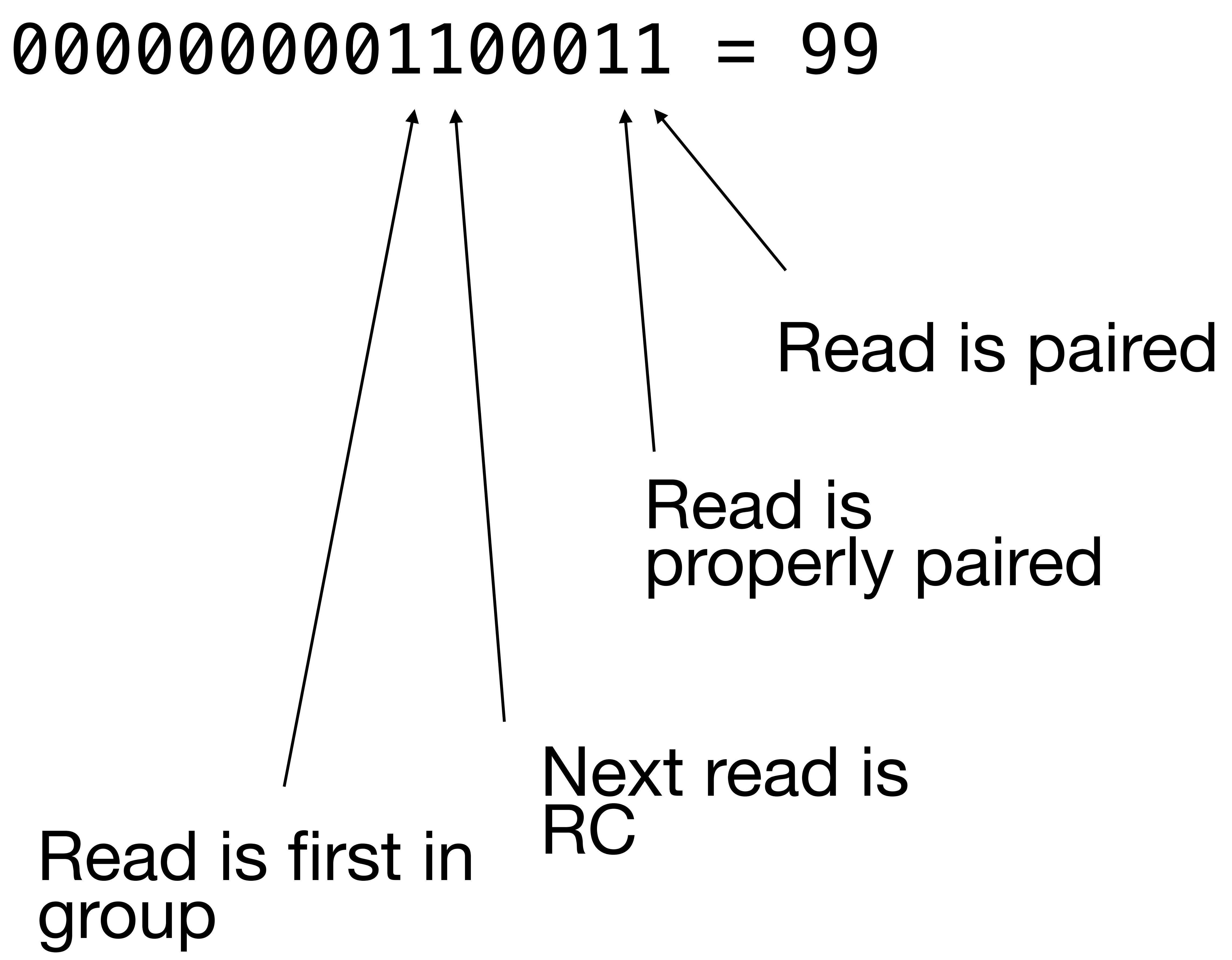
But what is this?

We won't go into these

SAM flags

- Each alignment has a list of boolean (true / false) statement
  - Read is paired
  - Read is properly paired
  - Read is not mapped
  - Next read in read group not mapped
  - Read is reverse complemented
  - Next read in read group is reverse complemented
  - Is first read in read group
  - Read is last read in read group
  - Secondary alignment
  - Alignment fails quality checks
  - PCR or optical duplicate
  - Supplementary alignment

Encode it as a number in binary!

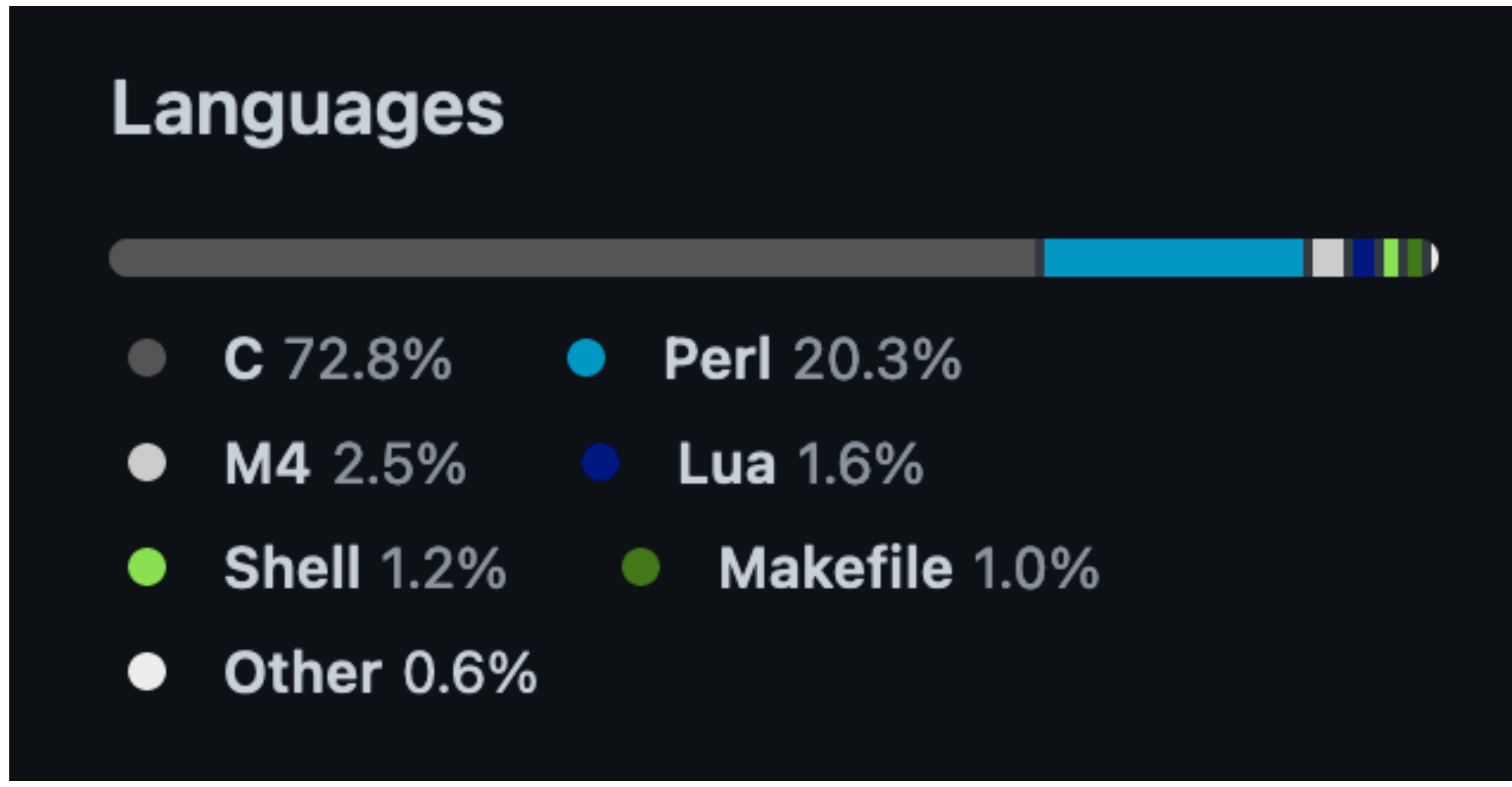


BAM format

- Binary Alignment Format
- The binary equivalent of SAM - Has a 1:1 correspondence with SAM
- More efficient storage than text
- Compressed in BGZF format
- Everyone uses BAM, no-one uses SAM.
    - Including you, in the exercise

`Two language problem...`

- BAM files are manipulated with the `samtools` program



```
$ samtools view subset.bam | grep 99 | head -1
```

XAM.jl

- Often you see:
  - Q: "How do I extract all primary alignment that map to this region where..."?
  - A: Use samtools to filter, then pass into grep, then awk, then re-add the header
- This awkwardness doesn't scale to more complicated workflows or really complex read filtering / processing
- XAM.jl: A Julia package for processing / reading / writing SAM and BAM
- XAM.jl is kind of crappy right now, but we will only do basic stuff today
  - We will re-write XAM at some point and make it good

Questions?

Exercise 5