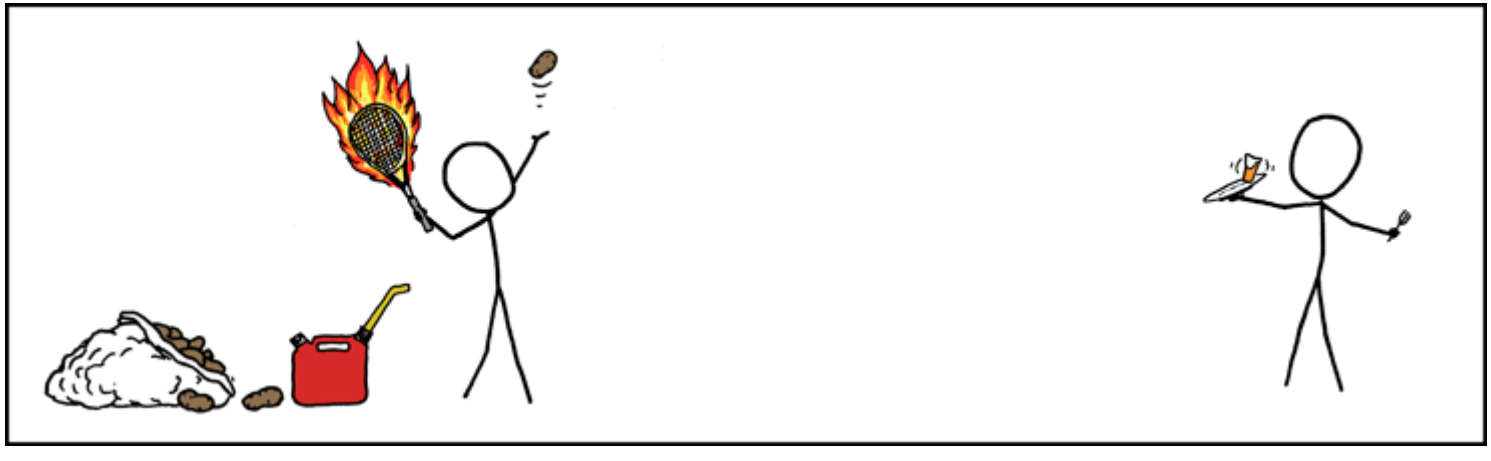
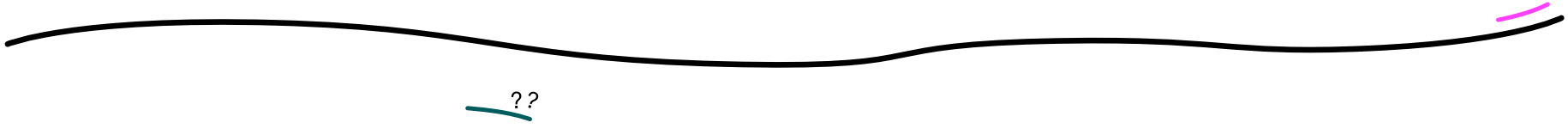
MIMM4750G

Short read mapping



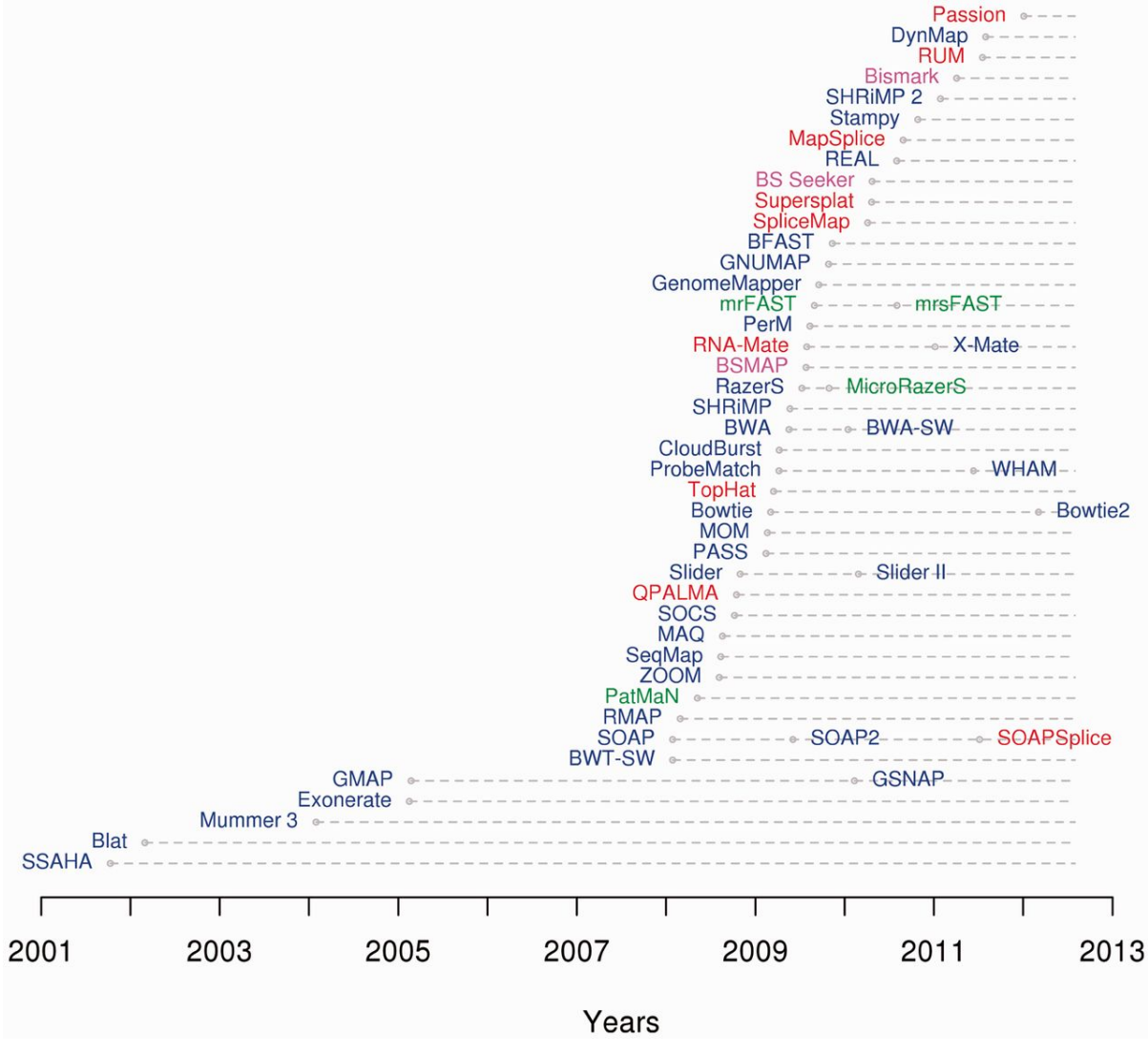
Two problems of alignment

- When we've been aligning sequences, we've been starting with a set of short, homologous sequences - the problem is figuring out where there might be insertions and deletions.
- What if we need to align a short sequence to a very long one, *e.g.*, a genome?
- We have to solve this *mapping* problem (where is the homologous region, if any?) before we can tackle the other one.



NGS and alignment

- The computing time of pairwise alignment is about $O(mn)$ where m and n are the sequence lengths.
- The development of NGS platforms created a huge challenge for existing alignment methods — too much data!
- **New alignment programs were needed.**
- In 2012, there were over 60 different programs available.
- Hash functions played an important role in many of these methods



Timeline of mapper software from NA Fonseca *et al* 2012, Bioinformatics 28: 3169.

Hash functions

- A hash function is some algorithm that can take a datum of any size (usually enormous) and quickly reduce it down to a value of a much smaller, fixed size.
- Since the range of possible hash *values* is much smaller than the range of possible *keys*, there is some chance that two or more keys will *collide* to the same value.
- For example, a function that determines if a string has an even or odd number of characters is a hash function (but a pretty useless one).

Hash functions

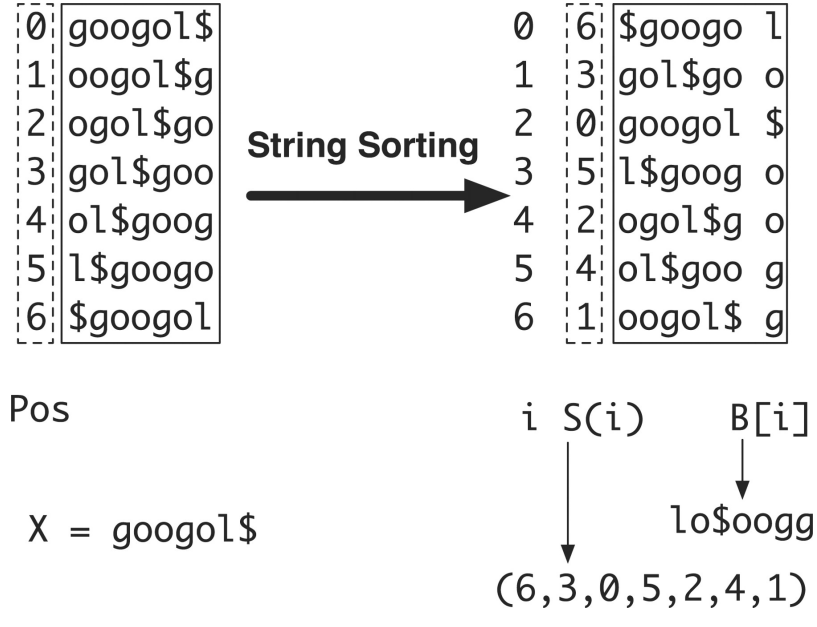
- MD5 is a hash function that is widely used to verify the contents of a file that has been transferred over the network (it is no longer considered secure).
- MD5 produces a 128-bit hash value. This is usually represented as a hexadecimal string of length 32. A hexadecimal value (`0` to `f`) represents four bits (*e.g.*, `1101` equals `d`).
- Chromosome 21 of the human genome (assembly hg38) has the MD5 hash value `eefe014d35decf90afde7b37e9954554`.

Hash indexing

- A hash value can be used to solve the first problem of alignment.
- Break the genome down into fragments.
- Generate the hash values for each fragment.
- Use the hash value as the lookup key for the location of that fragment in the genome.
- We can then rapidly "look up" the location of a matching fragment from our query sequence using this index.

Burrows-Wheeler transform

- Storing the hash index required a lot of memory!
- The next generation of short read mappers (BWA, Bowtie, SOAPv2) replace the hash function with the Burrows-Wheeler transform (BWT)



BWT

- The transform of string S, $BWT(S)$, is easier to compress
- Reversing $BWT(S)$ back to S is fast.
- Transform makes finding exact matches to another string T *really* fast.

Constructing an index

- `bowtie2-build` constructs the BWT table from one or more reference sequences
- `-f` flag indicates references are in a FASTA file
- `-q` flag sets program to quiet mode (otherwise it prints a *lot* of messages!)

```
art@Kestrel:~$ bowtie2-build -q -f Zika-reference.fa zika
Building a SMALL index
art@Kestrel:~$ ls -l
total 16436
-rw-rw-r-- 1 art art 4198153 Mar 14 14:14 zika.1.bt2
-rw-rw-r-- 1 art art    2704 Mar 14 14:14 zika.2.bt2
-rw-rw-r-- 1 art art      17 Mar 14 14:14 zika.3.bt2
-rw-rw-r-- 1 art art    2699 Mar 14 14:14 zika.4.bt2
-rw-rw-r-- 1 art art   10991 Mar  7 21:06 Zika-reference.fa
-rw-rw-r-- 1 art art 4198153 Mar 14 14:14 zika.rev.1.bt2
-rw-rw-r-- 1 art art    2704 Mar 14 14:14 zika.rev.2.bt2
```

Running bowtie2

- Mapping paired-end Illumina MiSeq reads from a Zika virus infection to the reference genome:

```
art@Kestrel:~$ bowtie2 -x zika -1 Zika-envelope.n1E4.R1.fastq.gz -2 Zika
10000 reads; of these:
  10000 (100.00%) were paired; of these:
    10000 (100.00%) aligned concordantly 0 times
    0 (0.00%) aligned concordantly exactly 1 time
    0 (0.00%) aligned concordantly >1 times
    ----
    10000 pairs aligned concordantly 0 times; of these:
      0 (0.00%) aligned discordantly 1 time
    ----
    10000 pairs aligned 0 times concordantly or discordantly; of these:
      20000 mates make up the pairs; of these:
        19979 (99.89%) aligned 0 times
        21 (0.10%) aligned exactly 1 time
        0 (0.00%) aligned >1 times
0.10% overall alignment rate
```

Coping with pathogen diversity

- Note that we could only map 21 out of 20000 reads - that sucks!
- Viruses (and sometimes bacteria) evolve so rapidly that there can be many genetic differences between a sample and the reference.
- Most mappers can only a small number of differences between a read and the reference!
- We can ask the mapper to *locally* align the read so that runs of mismatched bases at the 5' or 3' ends are not penalized.

Soft clipping

- To allow soft clipping (local alignment) in `bowtie2`, we use the `--local` option:

```
art@Kestrel:~$ bowtie2 -x zika -1 Zika-envelope.n1E4.R1.fastq.gz -2 Zika
10000 reads; of these:
  10000 (100.00%) were paired; of these:
    8639 (86.39%) aligned concordantly 0 times
    1361 (13.61%) aligned concordantly exactly 1 time
    0 (0.00%) aligned concordantly >1 times
    ----
    8639 pairs aligned concordantly 0 times; of these:
      0 (0.00%) aligned discordantly 1 time
    ----
    8639 pairs aligned 0 times concordantly or discordantly; of these:
      17278 mates make up the pairs; of these:
        17269 (99.95%) aligned 0 times
        9 (0.05%) aligned exactly 1 time
        0 (0.00%) aligned >1 times
13.65% overall alignment rate
```

Iterative mapping

- We are still only mapping about 14% of the reads to the reference.
- We need a reference that is more closely related to the consensus of our sample.
- Take the reads that *did* map to the reference and use them to "evolve" the genome into a new reference.

```
art@Kestrel:~$ python adapt-ref.py local.sam Zika-reference.fa adapted.f
NC_012532.1, original length 10794
Reads cover interval of length 1500
Updated reference with 221 differences
```

Iterative mapping

- Just one round of revising the reference sequence improves the mapping efficiency from 15% to 99%!

```
art@Kestrel:~$ bowtie2 -x adapted -1 Zika-envelope.n1E4.R1.fastq.gz -2 Z
10000 reads; of these:
  10000 (100.00%) were paired; of these:
    90 (0.90%) aligned concordantly 0 times
    9910 (99.10%) aligned concordantly exactly 1 time
    0 (0.00%) aligned concordantly >1 times
    ----
    90 pairs aligned concordantly 0 times; of these:
      0 (0.00%) aligned discordantly 1 time
    ----
    90 pairs aligned 0 times concordantly or discordantly; of these:
      180 mates make up the pairs; of these:
        180 (100.00%) aligned 0 times
        0 (0.00%) aligned exactly 1 time
        0 (0.00%) aligned >1 times
99.10% overall alignment rate
```

SAM to BAM

- The SAM output file is very useful, but it is HUGE!
- A BAM file is simply a binary archive of a SAM file.
- Since it is binary, it does not have to waste space making the data readable by a human.
- This conversion can be performed with *samtools*: "a library and software package for parsing and manipulating alignments in the SAM/BAM format"

samtools

- About a 3-fold compression of file contents:

```
art@Kestrel:~$ samtools view -S -b remapped.sam > remapped.bam
[samopen] SAM header is present: 1 sequences.
art@Kestrel:~$ ls -l remapped.*
-rw-rw-r-- 1 art art 2763590 Mar 14 17:25 remapped.bam
-rw-rw-r-- 1 art art 8840818 Mar 14 15:24 remapped.sam
```

- And **not** human readable:

```
art@Kestrel:~$ head -n1 remapped.bam
□□ï¿œ□BC□ï¿œeï¿œï¿œj□1□ï¿œï¿œeï¿œ>ï¿œï¿œï¿œ□ï¿œï¿œ_□ï¿œ\mUh]ï¿œo%ï¿œï¿œ□

                                                    sï¿œ9ï¿œï¿œxÝ□>
                                                                -ï¿
                                                        Nn.ï¿œï¿œ□□ï
```

Demonstrate Hutton Tablet

The James Hutton Institute