

DTU





**DTU Health Technology
Bioinformatics**

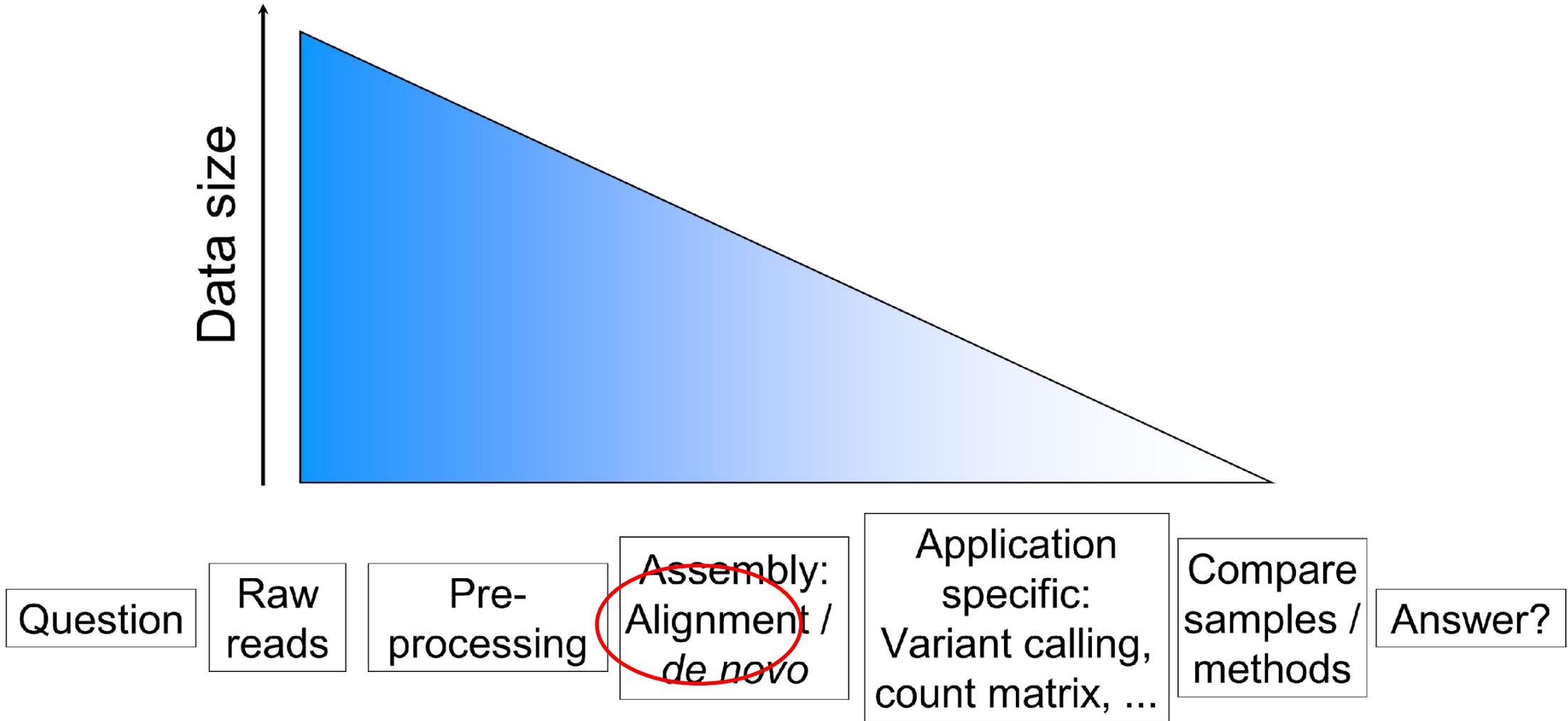
Alignment

*Gabriel Renaud
Associate Professor
Section of Bioinformatics
Technical University of Denmark
gabre@dtu.dk*

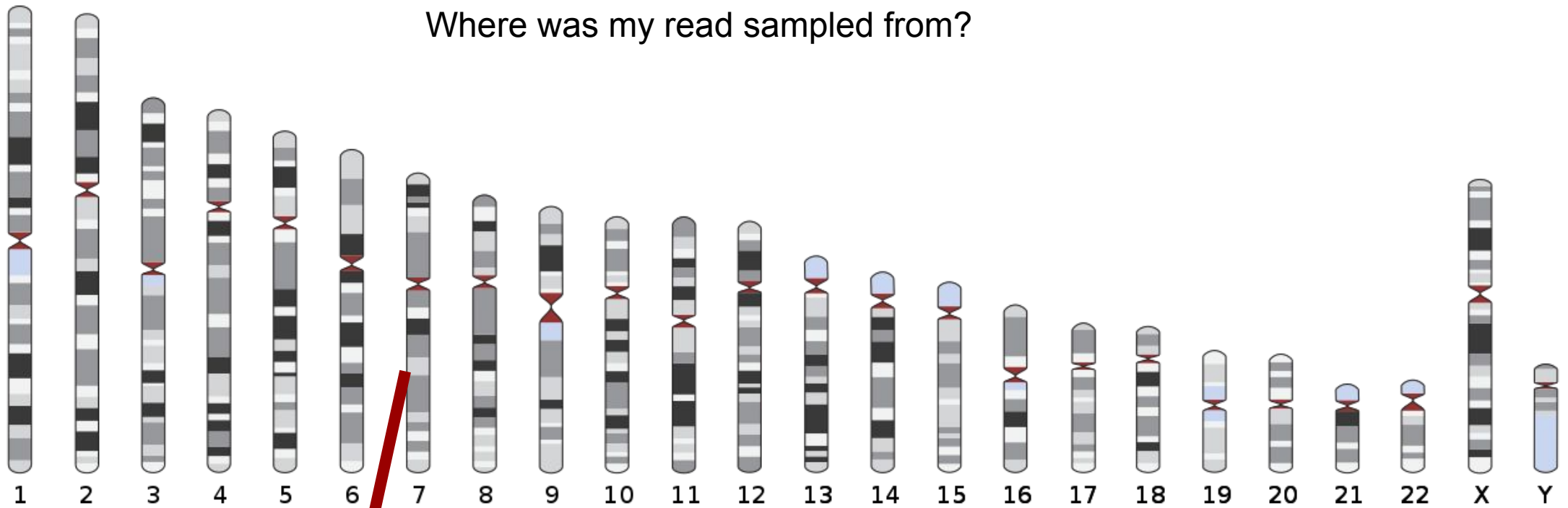
Menu

- Alignment approaches
- Burrows-Wheeler Transform
- More about coverage and depth
- Storing sequence alignments

Generalized NGS analysis



Where was my read sampled from?



Alignment:

my read

GGGGCTGCCGCCCCCATAGCA-GGACCTGGTCACAAGCAGC

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

chr7: 102,204,582

GGGGCTGCCGCCCCCATAGCAAGGACCTGGTCAC-AGCAGC



What is an alignment?

Alignment = story

seq1 : CAAGACTAACCTGAA
seq2 : CATGATAGCACTGCA

events?

seq1

CAAGACTAACCTGAA

seq2

CATGATAGCACTGCA



What is an alignment?

Alignment = story

seq1 : CAAGACTAACCTGAA
 seq2 : CATGATAGCACTGCA

6 events

seq1

CAAGACTAACCTGAA

CATGACTAACCTGAA

CATGATAACCTGAA

CATGA_TAGACTGAA

CATGA_TAGCACCTGAA

CATGA_TAGCACTGAA

CATGA_TAGCA_CTGCA

CATGATAGCACTGCA

seq2

What is an alignment?

Alignment = story

seq1 : CAAGACTAACCTGAA

seq2 : CATGATAGCACTGCA

D		C	A	T	G	A	T	A	G	C	A	C	T	G	C	A
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	-26	-28	-30
C	-2	1	-1	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21	-23	-25	-27
A	-4	-1	2	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24
A	-6	-3	0	1	-1	-1	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21
G	-8	-5	-2	-1	2	0	-2	-4	-4	-6	-8	-10	-12	-14	-16	-18
A	-10	-7	-4	-3	0	3	1	-1	-3	-5	-5	-7	-9	-11	-13	-15
C	-12	-9	-6	-5	-2	1	2	0	-2	-2	-4	-4	-6	-8	-10	-12
T	-14	-11	-8	-5	-4	-1	2	1	-1	-3	-3	-5	-3	-5	-7	-9
A	-16	-13	-10	-7	-6	-3	0	3	1	-1	-2	-4	-5	-4	-6	-6
A	-18	-15	-12	-9	-8	-5	-2	1	2	0	0	-2	-4	-6	-5	-5
C	-20	-17	-14	-11	-10	-7	-4	-1	0	3	1	1	-1	-3	-5	-6
C	-22	-19	-16	-13	-12	-9	-6	-3	-2	1	2	2	0	-2	-2	-4
T	-24	-21	-18	-15	-14	-11	-8	-5	-4	-1	0	1	3	1	-1	-3
G	-26	-23	-20	-17	-14	-13	-10	-7	-4	-3	-2	-1	1	4	2	0
A	-28	-25	-22	-19	-16	-13	-12	-9	-6	-5	-2	-3	-1	2	3	3
A	-30	-27	-24	-21	-18	-15	-14	-11	-8	-7	-4	-3	-3	0	1	4

Needleman-Wunsch: best story?

- Dynamic programming
- Find the best path in the matrix

What is an alignment?

Alignment = story

seq1 : CAAGACTAACCTGAA
 seq2 : CATGATAGCACTGCA

5 events

seq1

CAAGACTAACCTGAA

CATGACTAACCTGAA

CATGATAACCTGAA

CATGA_TAGCCTGAA

CATGA_TAGCACTGAA

CATGA_TAGCACTGCA

seq2

CATGATAGCACTGCA

What is an alignment?

Alignment = story

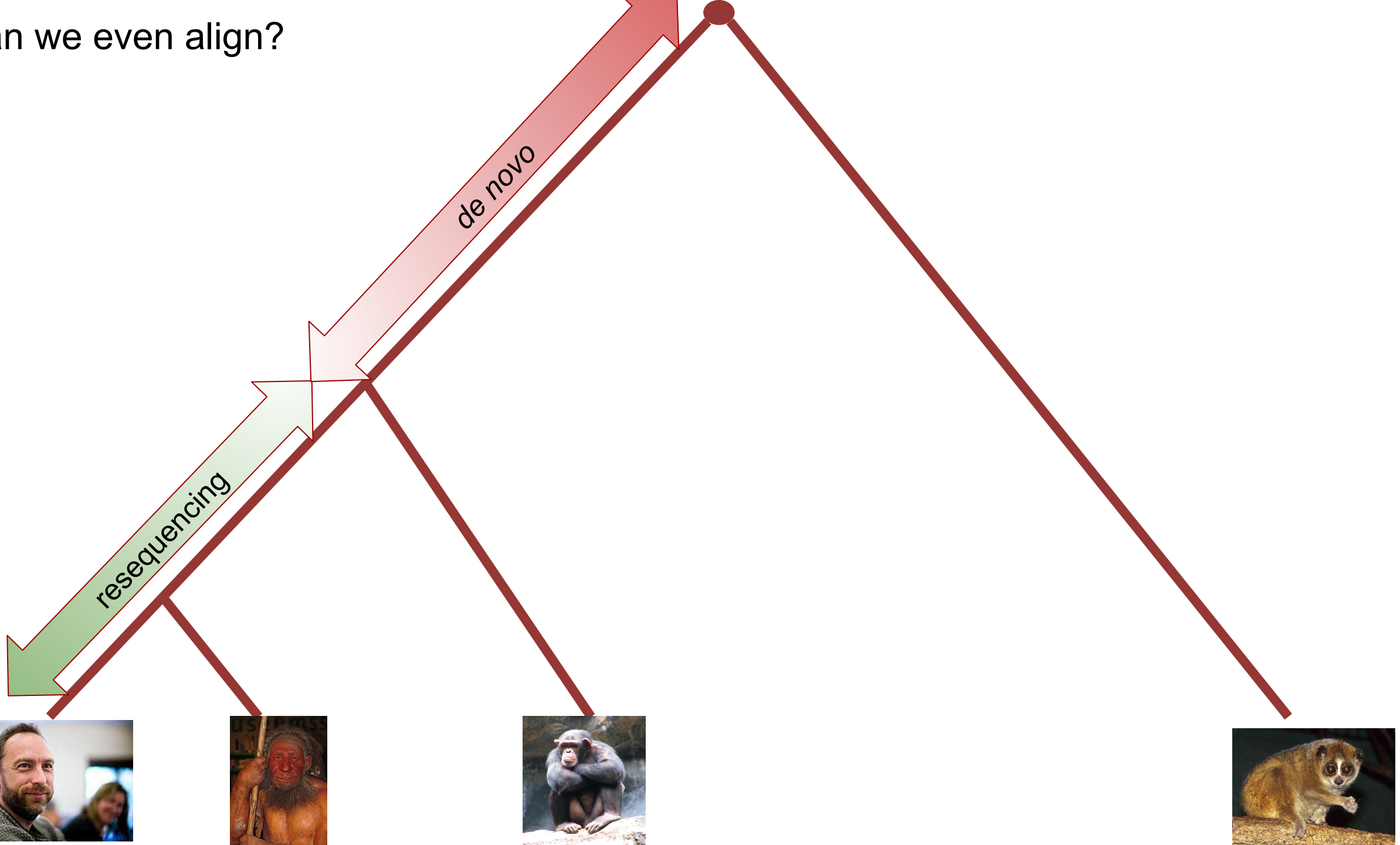
- 2 sequences can have a lot of alignments
- Not every alignment is equally likely
- Important to quantify the likelihood of seeing that alignment
- Be skeptical when you hear: “This is the alignment!”

What is an alignment?

Types of alignment:

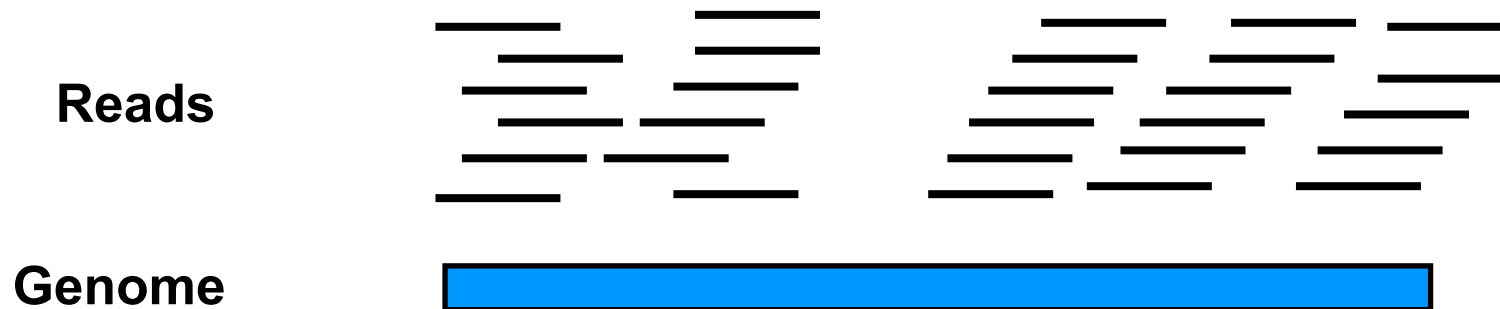
- “Short” read alignment
- Whole sequence alignment
- Whole genome/chromosome alignment
- Multiple sequence alignment

Can we even align?



Alignment/Mapping

- Assemble your reads by aligning them to a closely related reference genome
- High sequence similarity between individuals makes this possible



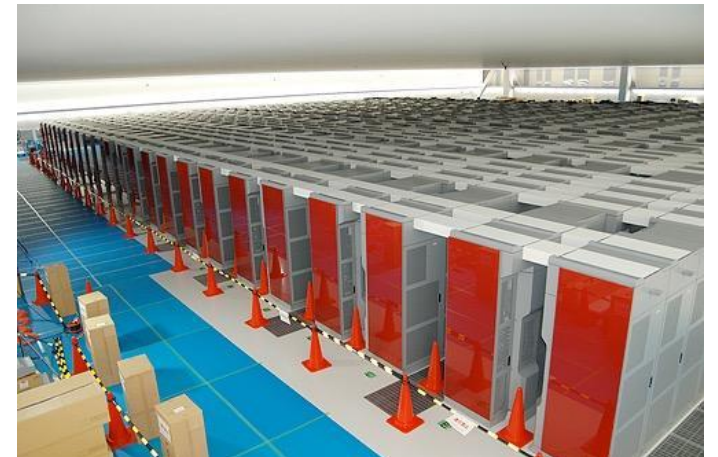
Sounds easy?

- Some pitfalls:
 - Divergence between sample and reference genome
 - Repeats in the genome
 - Recombination and re-arrangements
 - Poor reference genome quality
 - Read errors
 - Regions not in the ref. genome
 - Surprise sample



Simplest solution

- Exact string matches: Reference: ...ACGTGCGGACGCTGAACGTGA...
(does not work)
Read: GCGCACGCTGTAC
- We need to allow mismatches/indels
(Smith-Waterman, Needleman-Wunsch)
- One of the worlds fastest computer (*K computer - RIKEN*)
- 20M reads 100 nt reads vs. human genome ~ 1 month
- We search each read vs. the entire reference



How about BLAST?

- Basic Local Alignment Search Tool
- Build list of “words” common to the reference+query

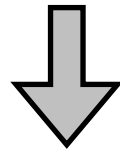
seq: CAAGACTAACCTGAA

- Sensitive, great for finding remote homologs:
 - given a human protein, find the mouse one
- Way too slow for large number of short reads

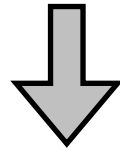
CAAGACT
 AAGACTA
 AGACTAA

Smart solution

1. Use algorithm to quickly find *possible* matches

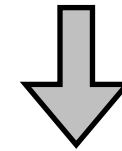


Drastically reduced search space

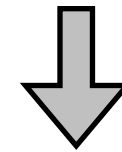


2. Allow us to perform slow/precise alignment for possible matches (Smith-Waterman)

3.2Gb



X possible matches

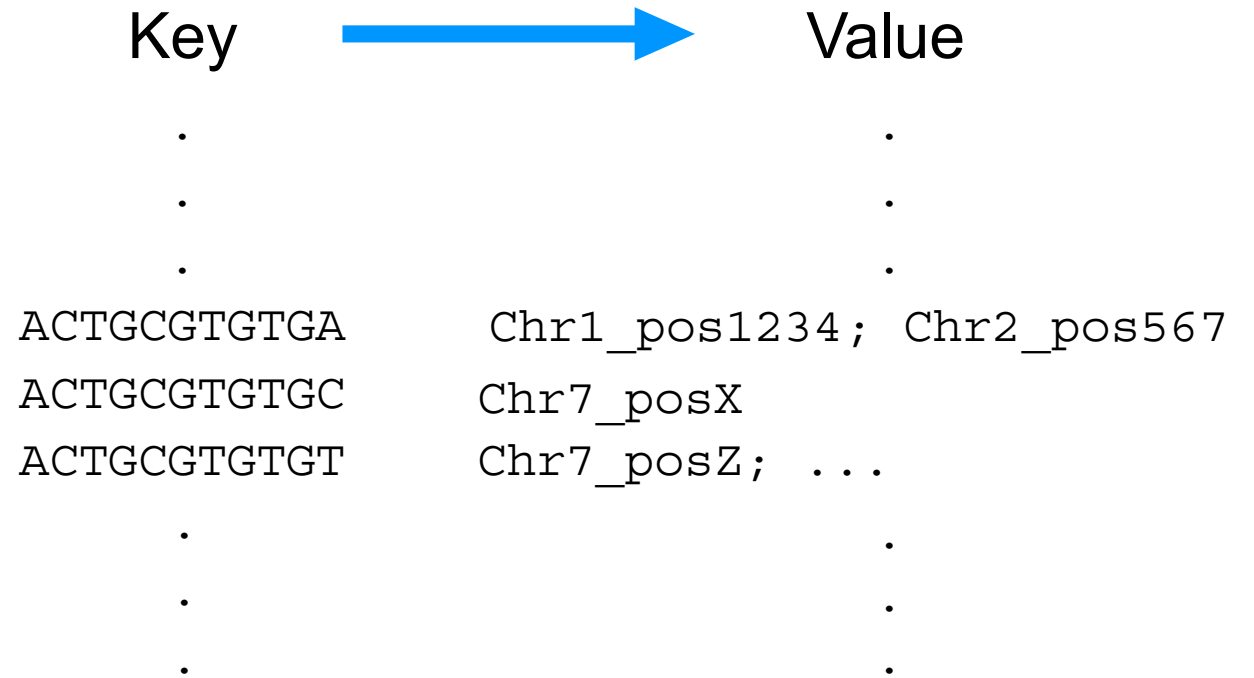


1 best match

Hash based algorithms

Lookups in hashes are *fast!*

1. Index the reference using *k*-mers.
2. Search reads vs. hash *k*-mers
3. Perform alignment of entire read around seed
4. Report best alignment



Also known as *Seed and extend*

Spaced seeds

- Key/ k -mer is called a seed
- BLAST uses $k=11$ and all must be matches
- Smarter: Spaced seeds (only care about '1' in seed, '0' = wildcards)
 - Higher sensitivity
 - One can use several seeds

11111111111

L = 11, 11 matches

111010010100110111

L = 18, 11 matches

Multiple seeds & drawbacks

- One could require multiple short seeds
 - Instead of extending around each seed, extend around positions with several seed matches
- Drawbacks of hash-based approaches:
 - Lots(!) of RAM to keep index in memory (hg ~48Gb!)

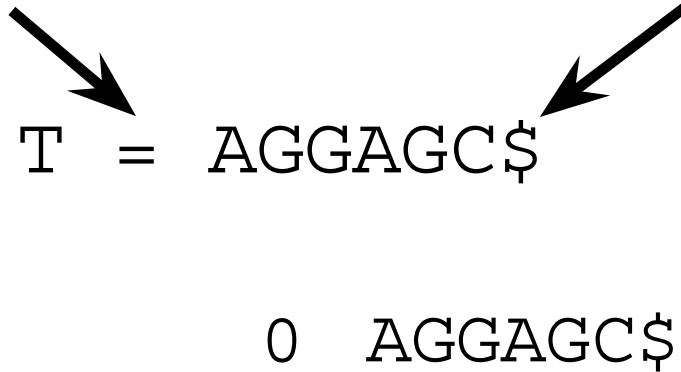
Burrows-Wheeler Transform

- Hash based aligners require lots of memory and are only reasonable fast
- Can we make it better/faster?
- Burrows Wheeler Transformation (BWT)
- BWT was originally created for compression

BWT: Create index

Genome

Marks end-of-string, lexicographically smallest



 T = AGGAGC\$

 0 AGGAGC\$

1. Create all possible shifts of the string

(move first base to end)

BWT: Create index

Genome

Marks end-of-string, lexicographically smallest

T = AGGAGC\$

1. Create all possible shifts of the string

(move first base to end)

0 AGGAGC\$

1 GGAGC\$A

BWT: Create index

Genome

Marks end-of-string, lexicographically smallest

T = AGGAGC\$

1. Create all possible shifts of the string

(move first base to end)

0	AGGAGC\$
1	GGAGC\$A
2	GAGC\$AG

BWT: Create index

Genome

Marks end-of-string, lexicographically smallest

T = AGGAGC\$

1. Create all possible shifts of the string

(move first base to end)

0	AGGAGC\$
1	GGAGC\$A
2	GAGC\$AG
3	AGC\$AGG

BWT: Create index

Genome

Marks end-of-string, lexicographically smallest

T = AGGAGC\$

1. Create all possible shifts of the string

(move first base to end)

0	AGGAGC\$
1	GGAGC\$A
2	GAGC\$AG
3	AGC\$AGG
4	GC\$AGGA
5	C\$AGGAG
6	\$AGGAGC

BWT: Create index

Genome

Marks end-of-string, lexicographically smallest

T = AGGAGC\$

2. Sort the strings lexicographically to create BWT matrix and Suffix Array

0	AGGAGC\$
1	GGAGC\$A
2	GAGC\$AG
3	AGC\$AGG
4	GC\$AGGA
5	C\$AGGAG
6	\$AGGAGC

\$	A	G	G	A	G	C
A	G	C	\$	A	G	G
A	G	G	A	G	C	\$
C	\$	A	G	G	A	G
G	A	G	C	\$	A	G
G	C	\$	A	G	G	A
G	G	A	G	C	\$	A

BWT matrix

BWT: Create index

Genome

Marks end-of-string, lexicographically smallest

T = AGGAGC\$

BWT (T) = CG\$GGAA

\$	A	G	G	A	G	C
A	G	C	\$	A	G	G
A	G	G	A	G	C	\$
C	\$	A	G	G	A	G
G	A	G	C	\$	A	G
G	C	\$	A	G	G	A
G	G	A	G	C	\$	A

BWT matrix

BWT: Create index

Genome

Marks end-of-string, lexicographically smallest

T = AGGAGC\$

- Reversible
- BTW(T) is easier to compress than T due to repeated characters tend to cluster ex:

BWT (T) = CG\$GGAA

Ringeren_I_Ringe_ringer_ringere_end_ringeren_ringer_i_Ringsted\$
 \$d__ _nIiernerdenrgtrr_gggggnnnnnnn_RrrrRrReeeiiiiiiiieeeee____gs

- try bzip2

BWT: *T*-rank

$T = \text{AGGAGC\$}$

T-ranking:

of times the base occurred previously in T

$A_0 \quad G_0 \quad G_1 \quad A_1 \quad G_2 \quad C_0 \quad \$$

<i>F</i>						<i>L</i>
\$	A	G	G	A	G	C
A	G	C	\$	A	G	G
A	G	G	A	G	C	\$
C	\$	A	G	G	A	G
G	A	G	C	\$	A	G
G	C	\$	A	G	G	A
G	G	A	G	C	\$	A

BWT: *T*-rank

$T = \text{AGGAGC\$}$

T-ranking: # of times the base occurred previously in *T*

$A_0 \quad G_0 \quad G_1 \quad A_1 \quad G_2 \quad C_0 \quad \$$

<i>F</i>						<i>L</i>
\$	A_0	G_0	G_1	A_1	G_2	C
A_1	G_2	C_0	\$	A_0	G_0	G_1
A_0	G_0	G_1	A_1	G_2	C_0	\$
C_0	\$	A_0	G_0	G_1	A_1	G_2
G_1	A_1	G_2	C_0	\$	A_0	G_0
G_2	C_0	\$	A_0	G_0	G_1	A_1
G_0	G_1	A_1	G_2	C_0	\$	A_0

Notice that individual base-rank is the same in *F* and *L*



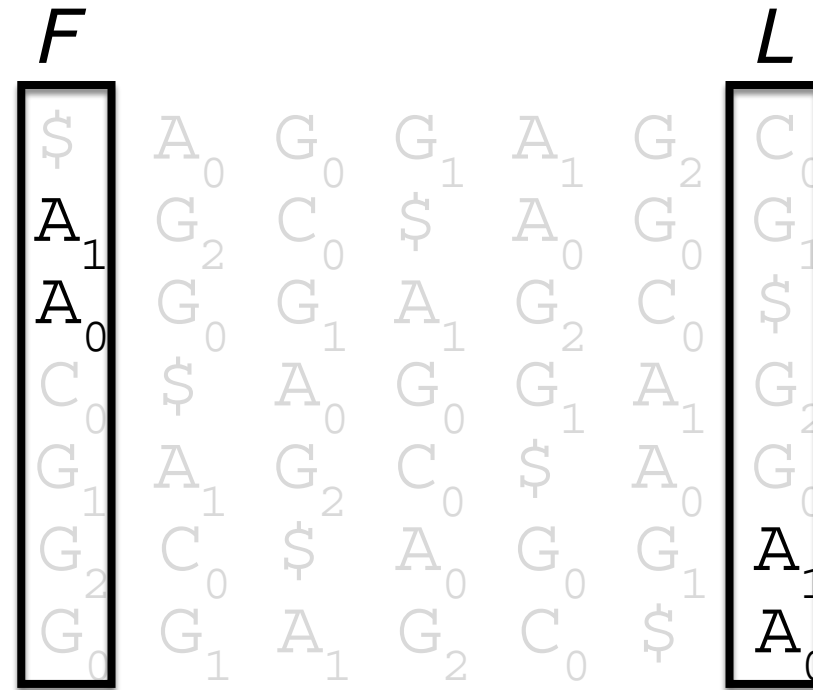
Rank will always be the same in *F* and *L*

BWT: *T*-rank

$T = \text{AGGAGC\$}$

T-ranking: # of times the base occurred previously in *T*

$A_0 \quad G_0 \quad G_1 \quad A_1 \quad G_2 \quad C_0 \quad \$$



Notice that individual base-rank is the same in *F* and *L*



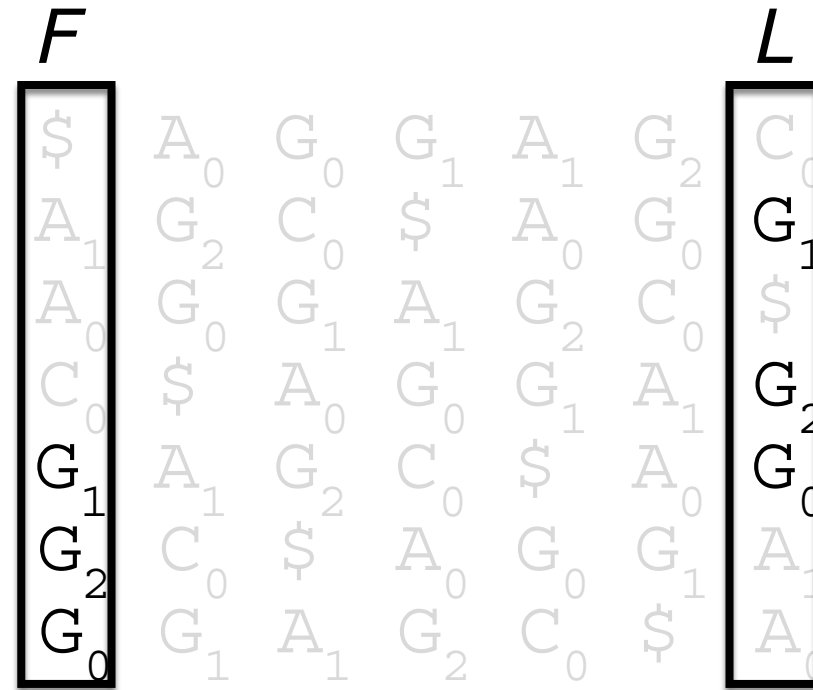
Rank will always be the same in *F* and *L*

BWT: *T*-rank

$T = \text{AGGAGC\$}$

T-ranking: # of times the base occurred previously in *T*

$A_0 \quad G_0 \quad G_1 \quad A_1 \quad G_2 \quad C_0 \quad \$$



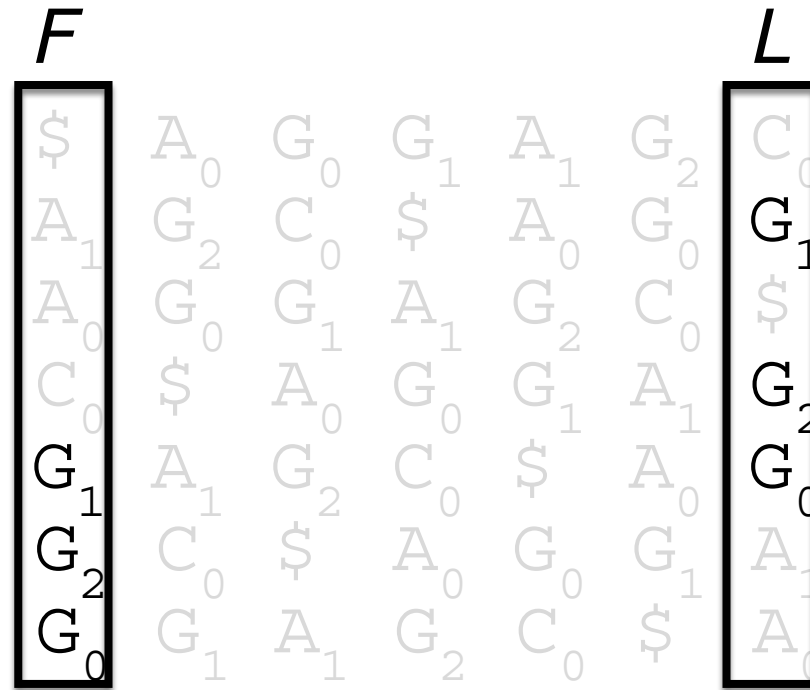
Notice that individual base-rank is the same in *F* and *L*



Rank will always be the same in *F* and *L*

BWT: *T*-rank

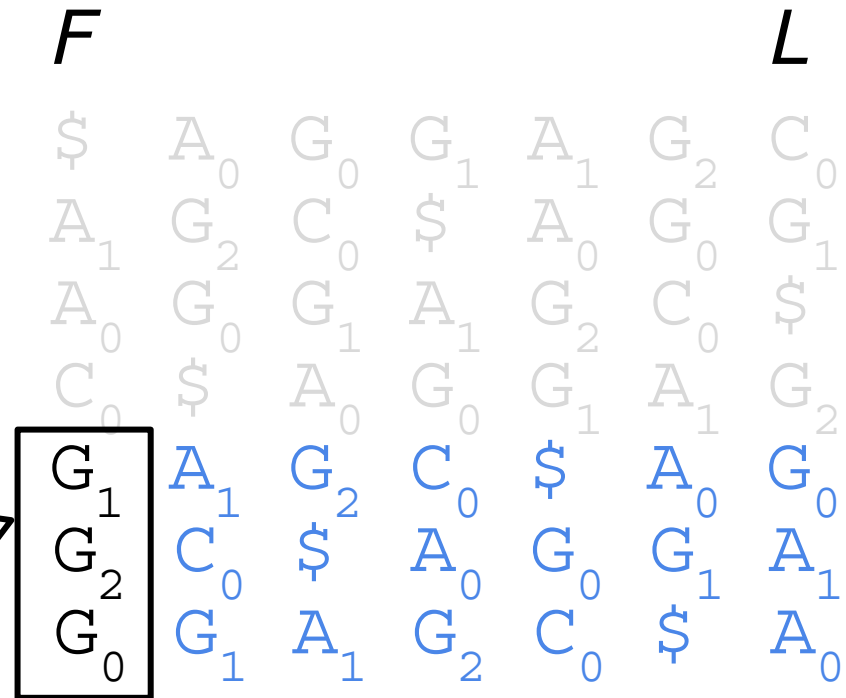
Why does this generalize?



BWT: *T*-rank

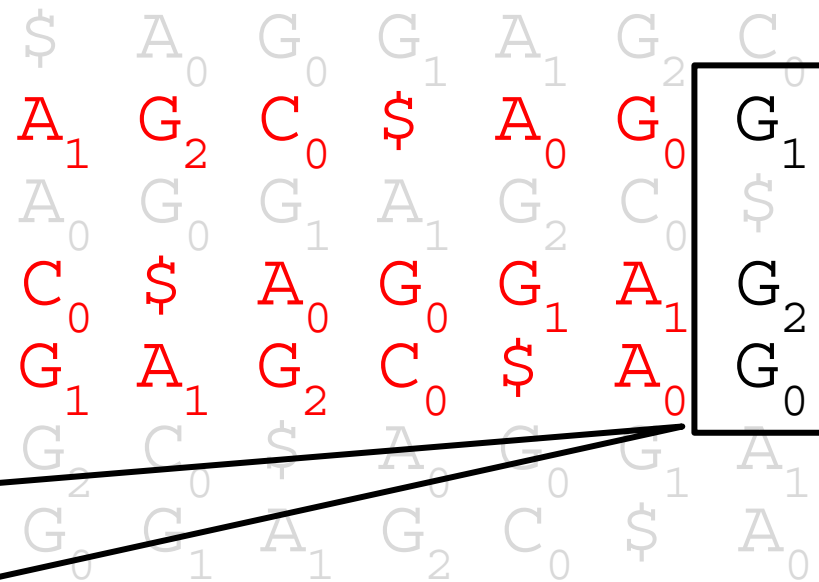
Why does this generalize?

We are all the same letter, our order is determined by what is right of us (blue).



BWT: T-rank

Why does this generalize?

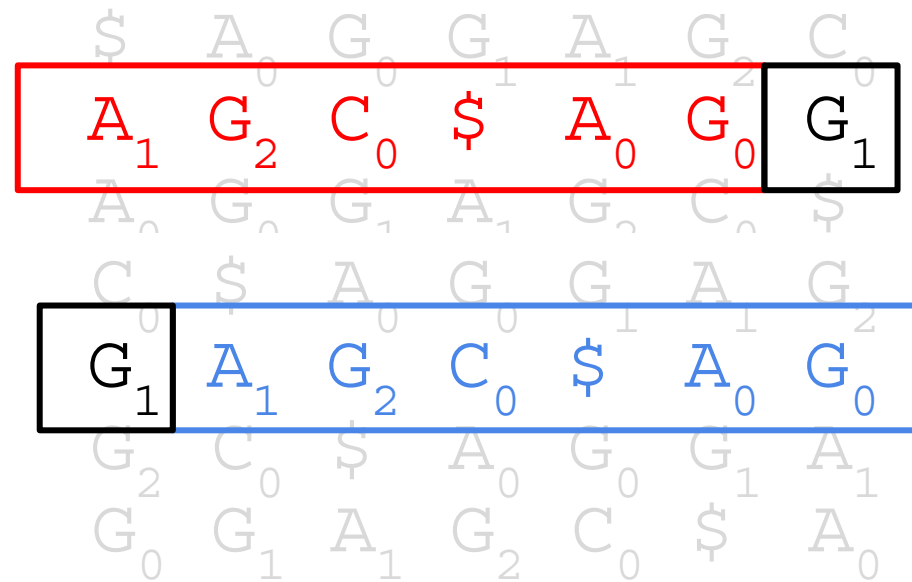


We are all the same letter, our order is determined by what is left of us (red).

BWT: *T*-rank

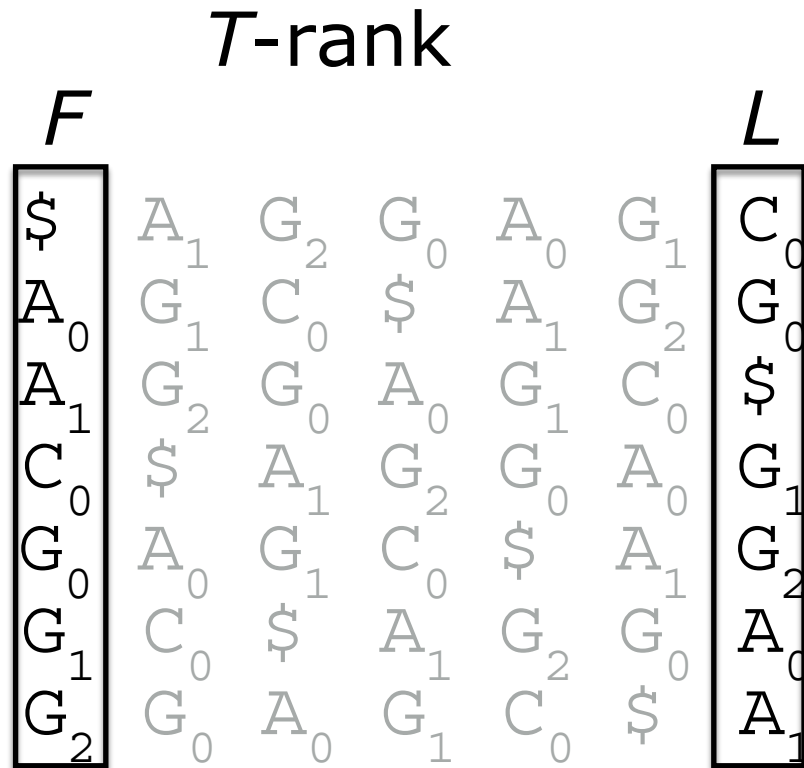
Why does this generalize?

- The string left (red) of the G1 and right (blue) of G1 are identical
- They are sorted
- Therefore the order is the same



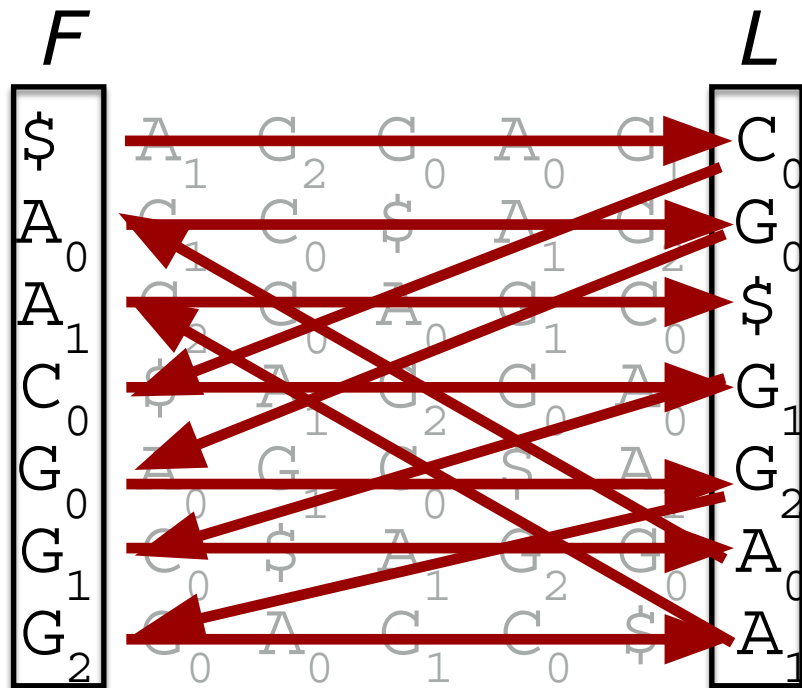
BWT: *T*-rank

How to reverse the BTW back the original string *T* ?



BWT is reversible

LF-mapping: LF can be used to recreate the original genome



$C_0 G_1 A_0 G_0 G_2 A_1 \$$

Reversed:

$A_1 G_2 G_0 A_0 G_1 C_0$

$T = AGGAGC\$$

F can be represented = 2x A, 1x C, 3x G
we need $|\Sigma|$ integers

We therefore only need to store L

Why are we talking about BTW for alignments?

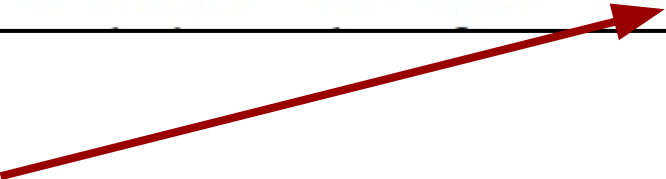
- In 1994, Michael Burrows and David Wheeler created the Burrows-Wheeler Transform (BWT)
 - A reversible transformation of the genome
- Full-text index in Minute space (FM) index
 - *Paolo Ferragina, and Giovanni Manzini. "Opportunistic data structures with applications." Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on. IEEE, 2000.*
 - Implementations: BWA, bowtie and SOAP2
 - How fast? First an intro to “big O” notation

Brief intro to “Big O” notation

The CALCULATED procedure in Figure 3 gives a better, though not optimal, bound. It is conceptually equivalent to the one described in Figure 4, which is simpler to understand. We use the BWT of the reverse (not complemented) reference sequence to test if a substring of W is also a substring of X . Note that to do this test with BWT string B alone would make CALCULATED an $O(|W|^2)$ procedure, rather than $O(|W|)$ as is described in Figure 3.



?

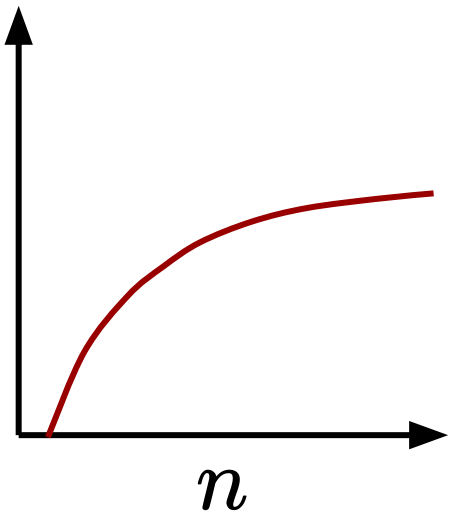


from: Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*. 2009 Jul 15;25(14):1754-60.

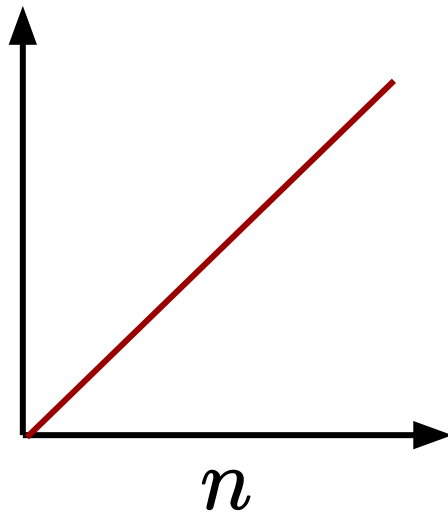
Brief intro to “Big O” notation

- If I have n sequences, how does the amount of time required by the program increase?

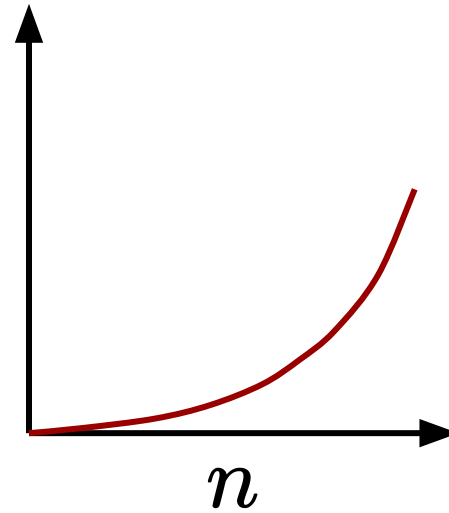
$\log(n)$



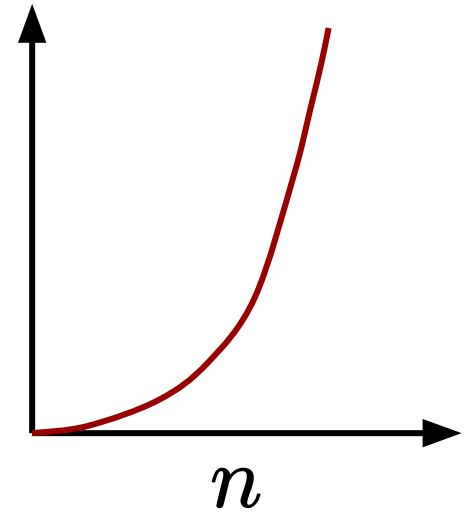
n



n^2



2^n



Brief intro to “Big O” notation

- If I have n sequences, how does the amount of time required by the program increase?

Number of steps

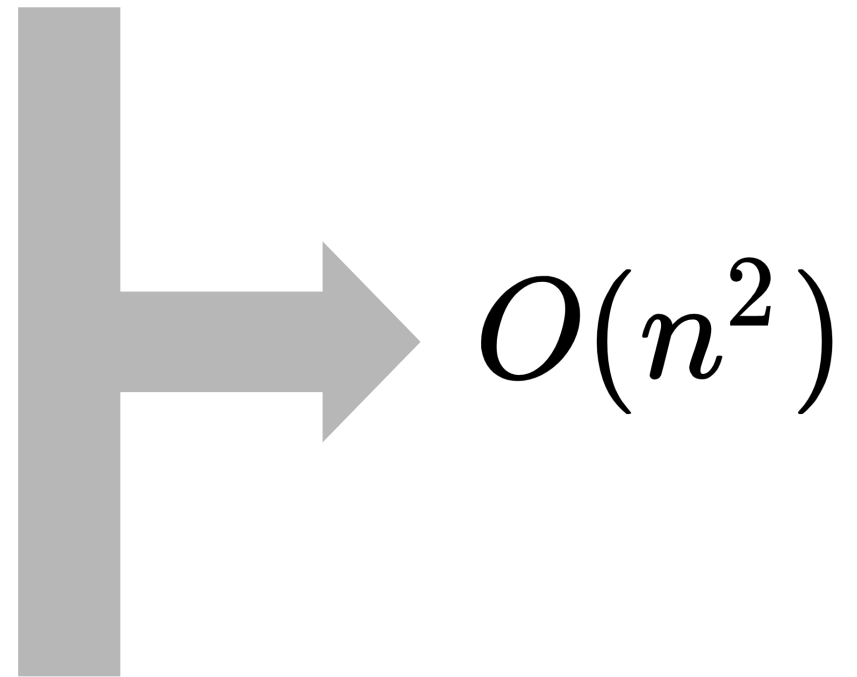
$$3n^2 + 6n + 1$$

$$25n^2 + 5n + 4$$

$$9n^2$$

$$1000n^2$$

$$n^2 - 9$$

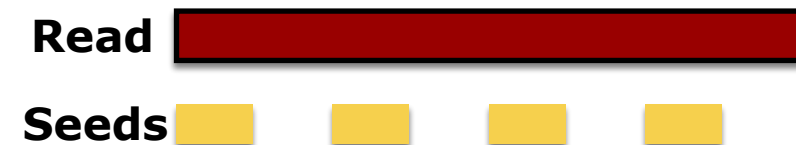


BWT for alignment

- The FM index uses $O(|L|)$ memory
- Can be searched in $O(|pattern|)$ steps
- Entire FM-index is 1.5Gb for human genome
- FM-index
 - We also need certain other data structures that we did not cover
- Human genome can be effectively indexed and searched using 3Gb RAM!

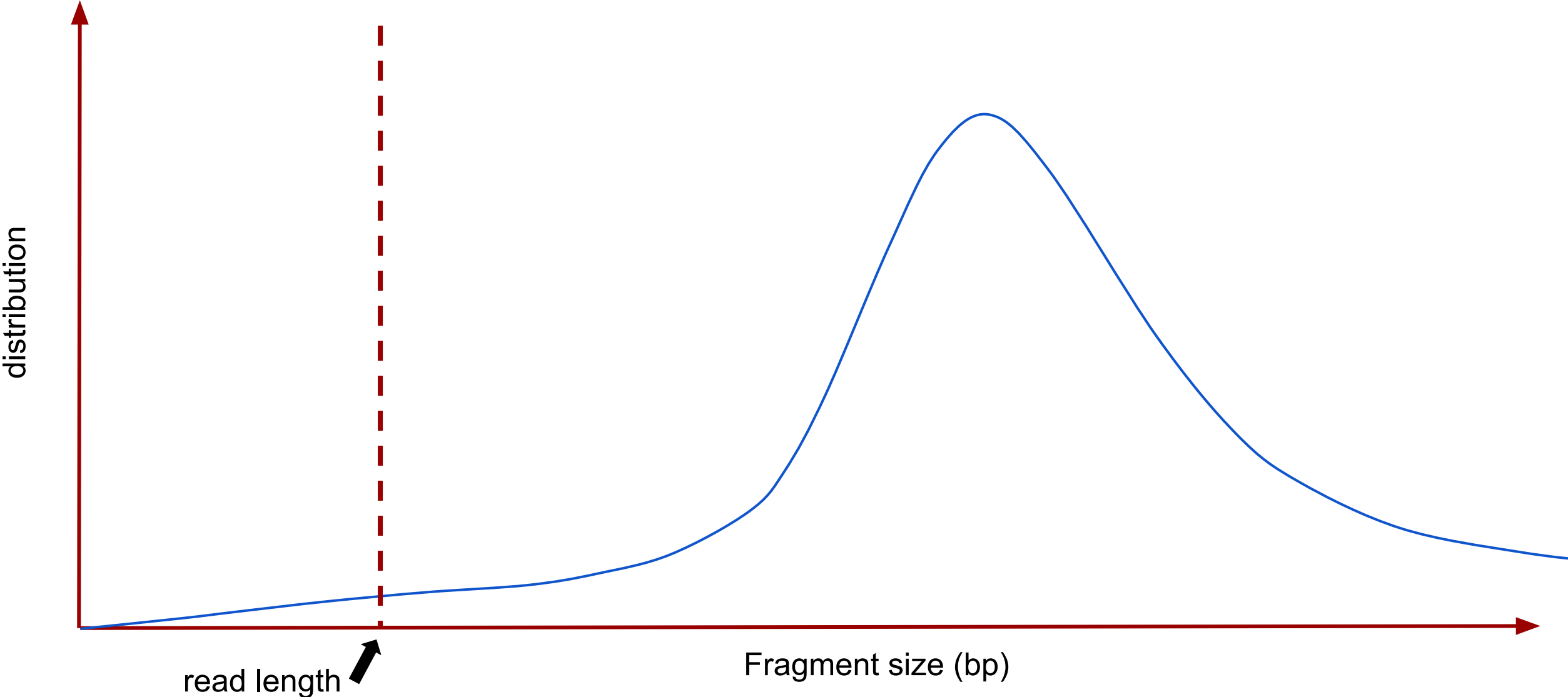
Implementation in BWA

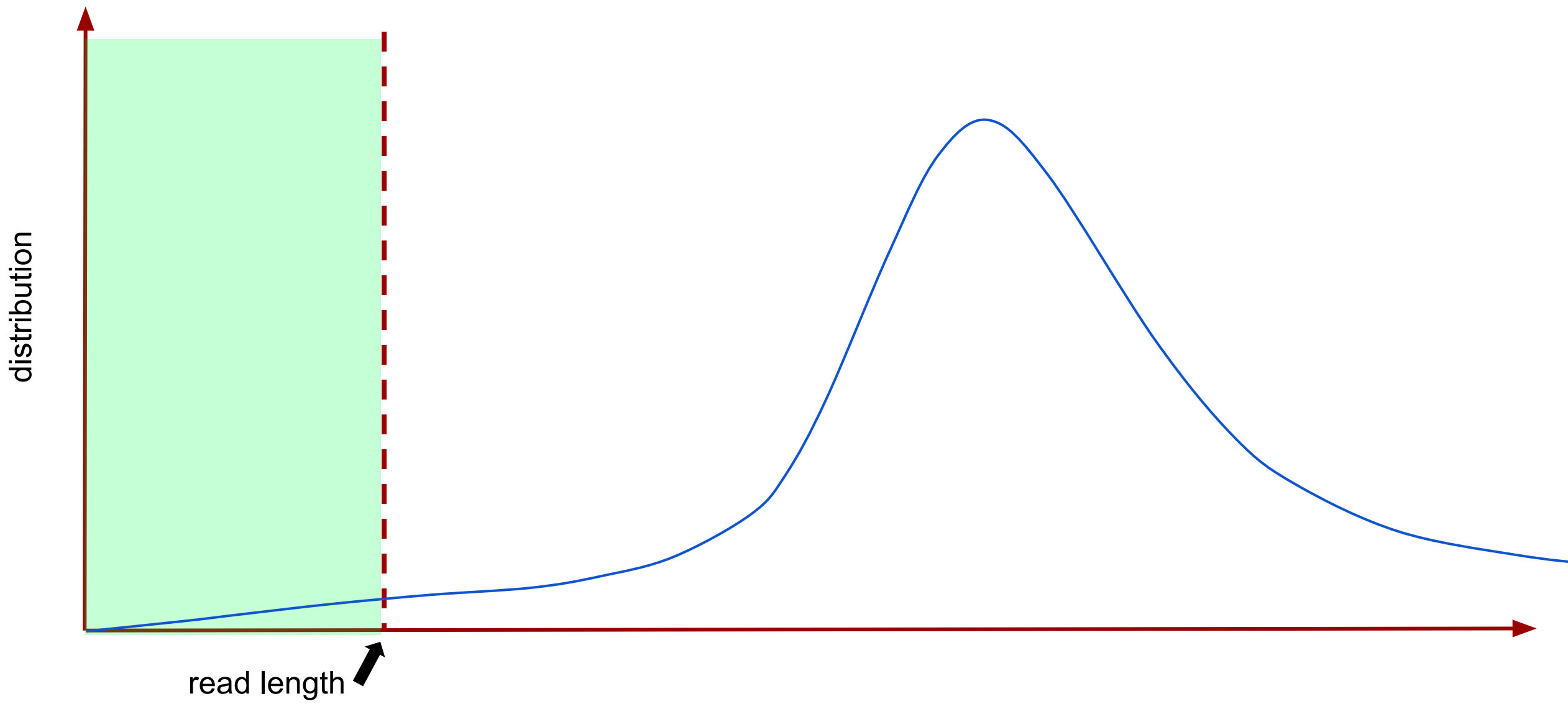
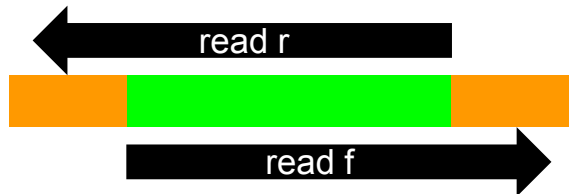
- Burrows Wheeler Aligner (BWA) can use:
 - bwa aln: First ~30nt of read as seed
 - Extend around positions with seed match
 - For short reads
 - bwa mem: Multiple short seeds across the read
 - Extend around positions with several seed matches
 - For longer reads

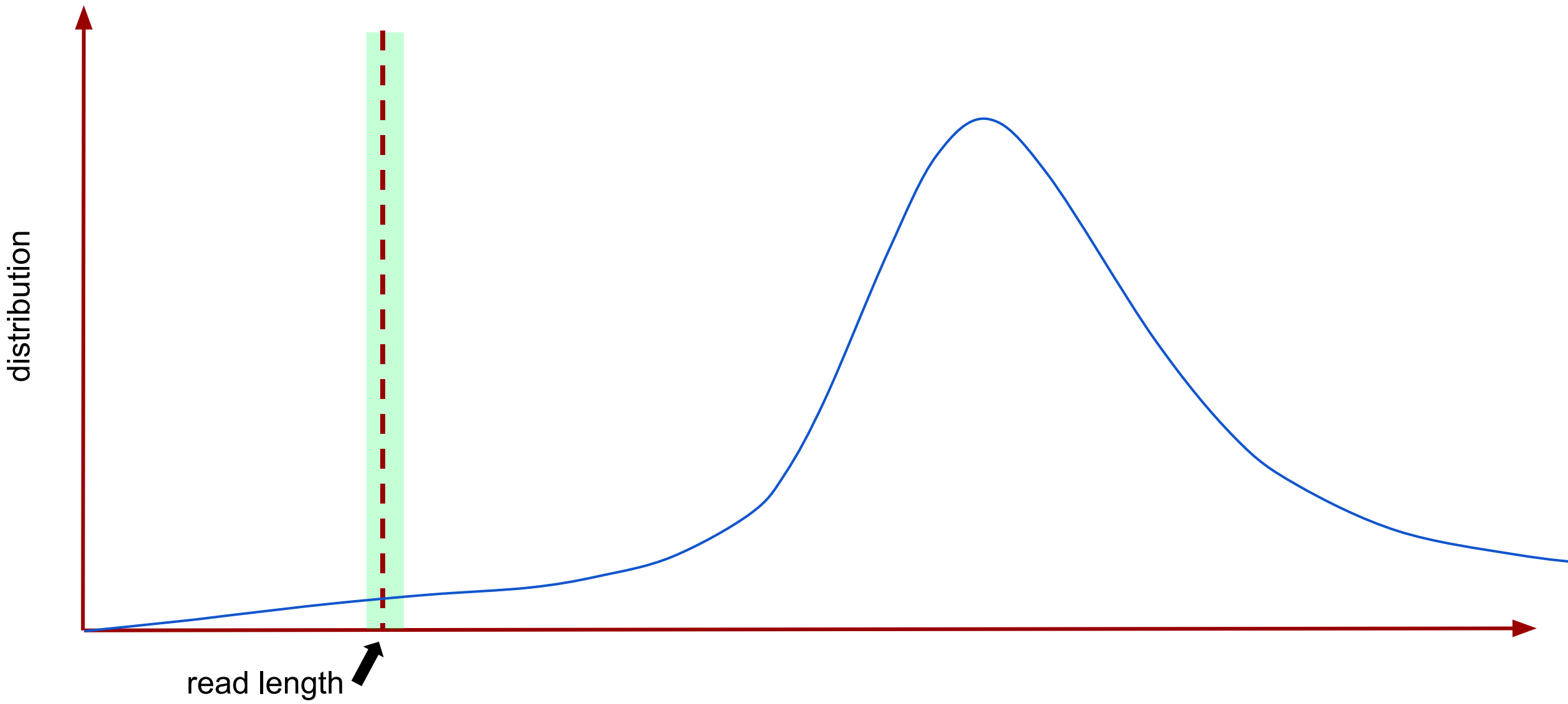
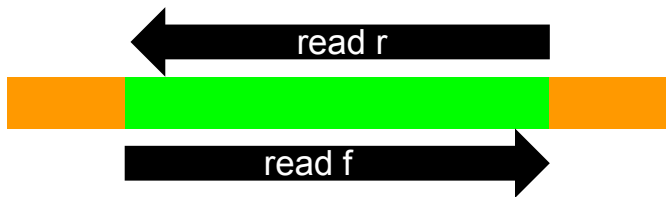


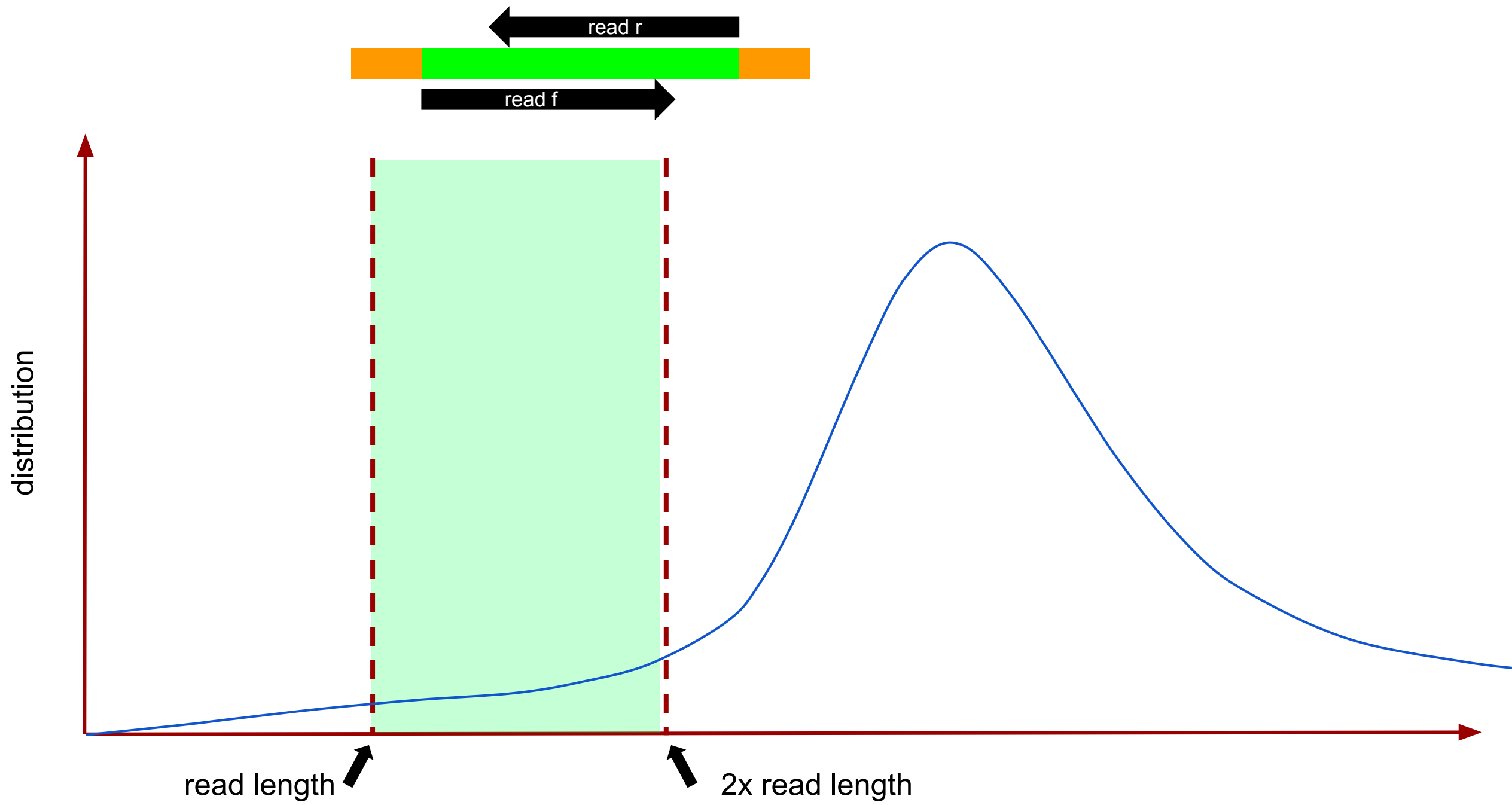
Notes about mapping quality and paired-end

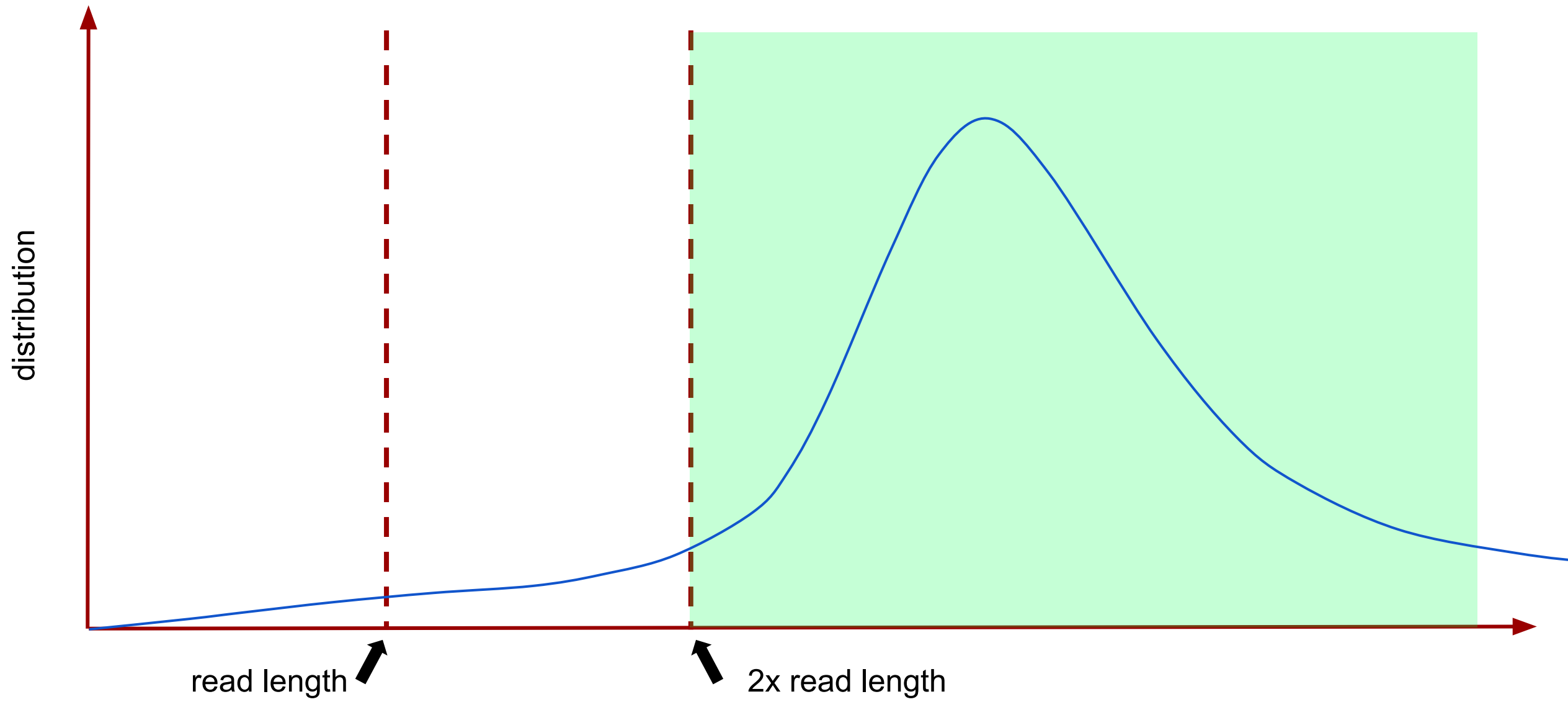
Insert size distribution

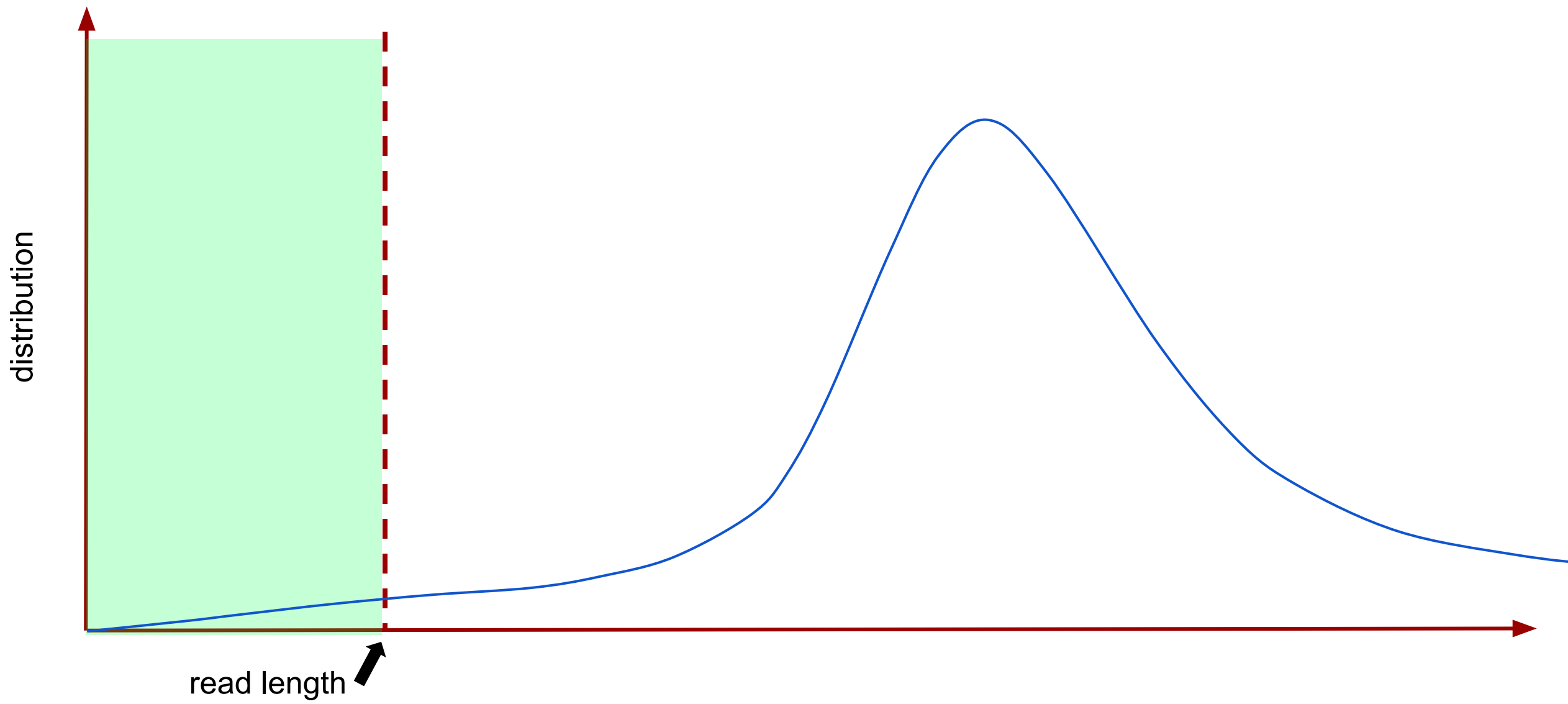








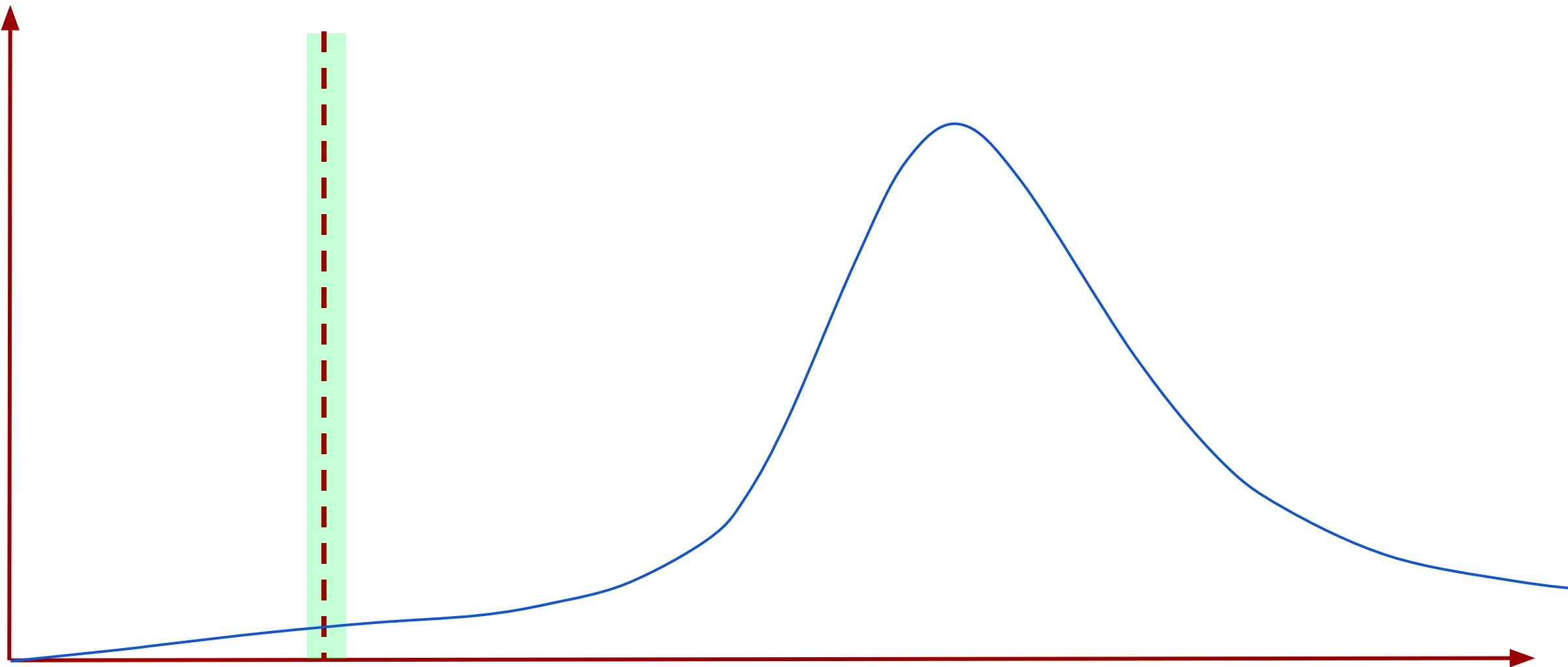




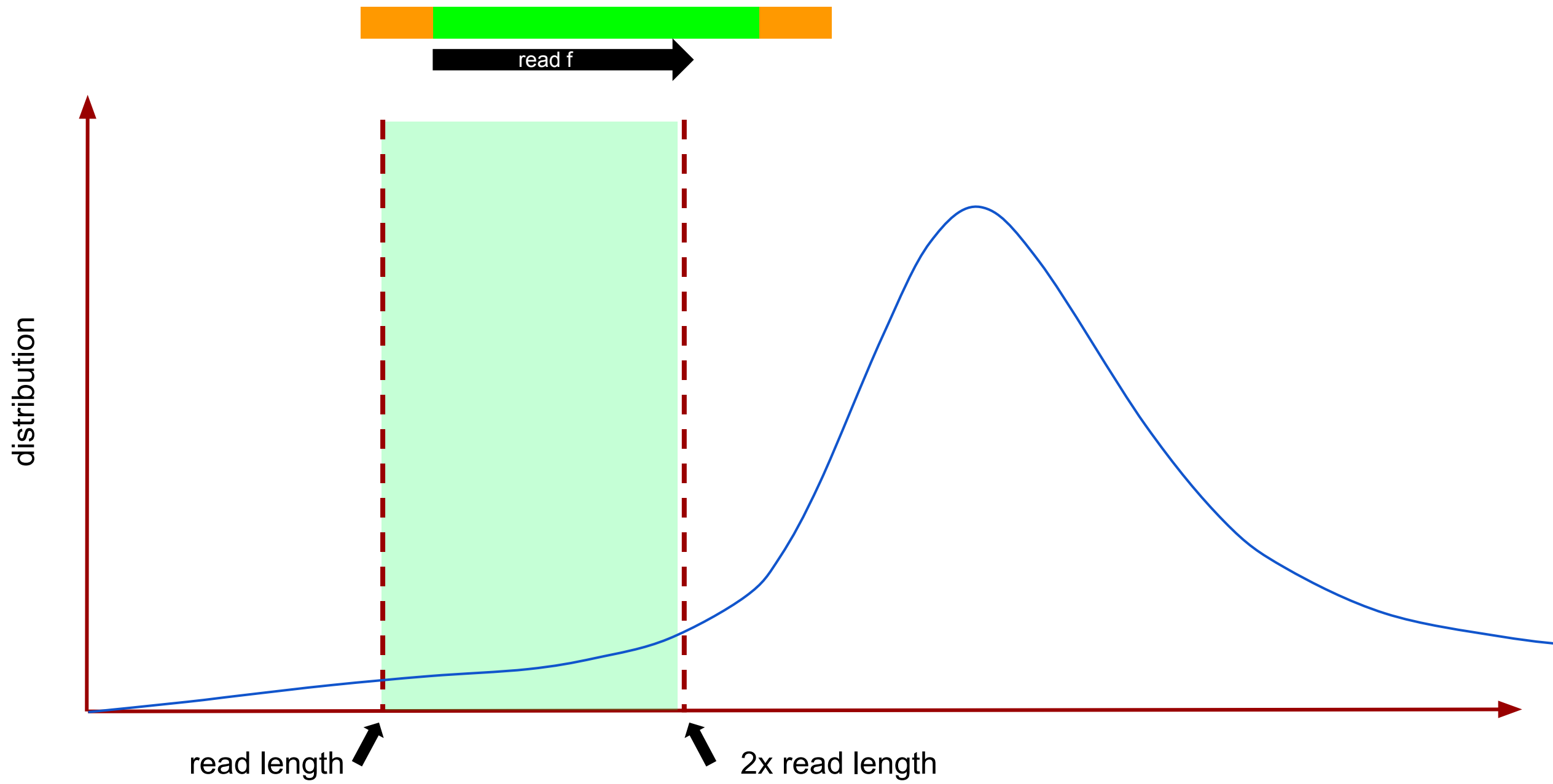


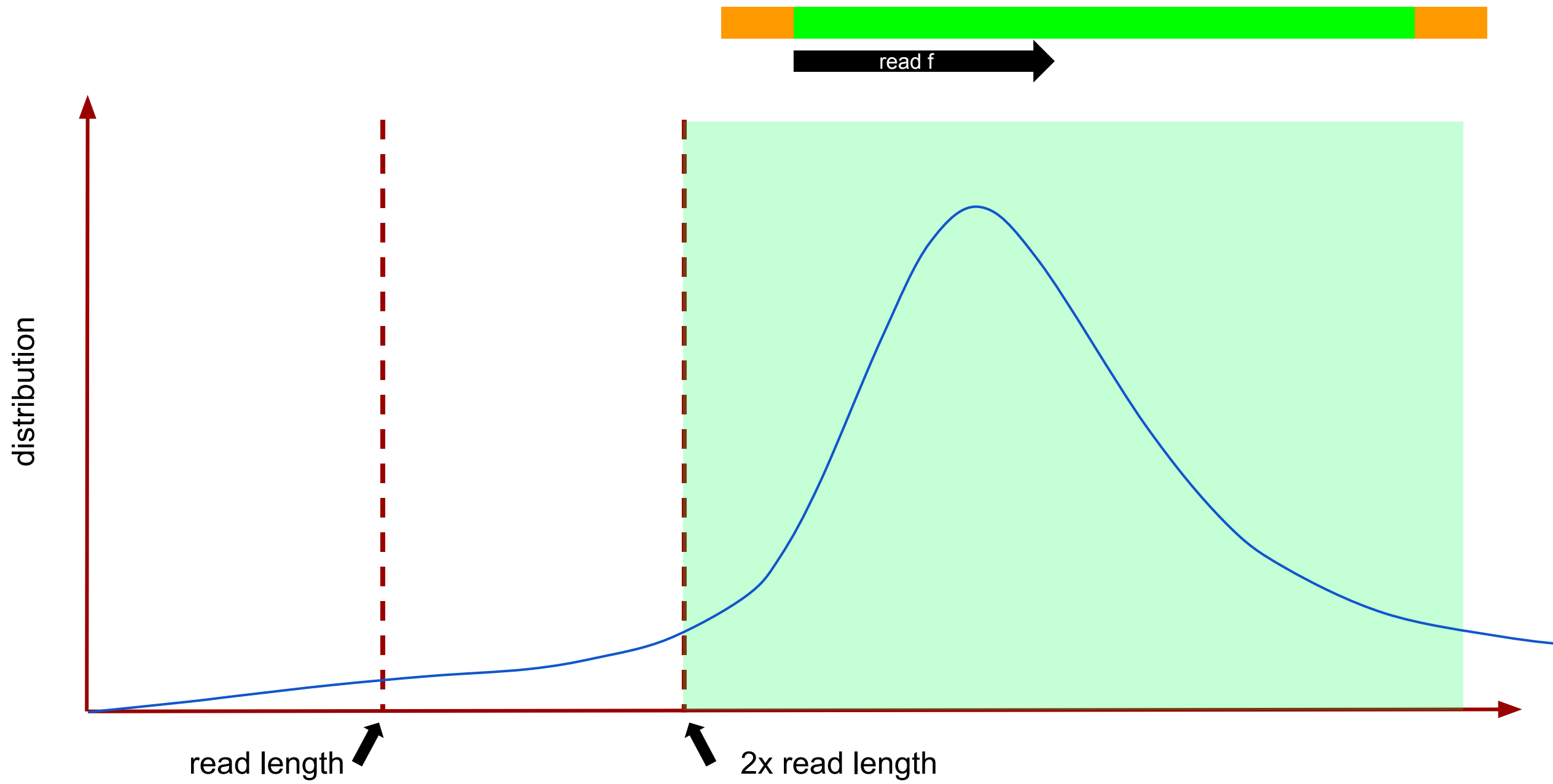
read f

distribution



read length





Intro to mapping quality

What happens when a sequence has multiple hits to the genome?
Depends on the aligner, Burrows-Wheeler Aligner (BWA) does the following:

- Assign to the genomic location with the best score
- Use other matches to compute the probability of mismapping on a log scale:

$$\text{MAPQ} = -10 \log (P[\text{mismapping}])$$

$$\text{e.g. MAPQ } 30 = P[\text{mismapping}] = 1/1000$$

Mapping quality

30

$p(\text{match}) = 0.001$



$p(\text{match}) = 0.99$



reference

6

$p(\text{match}) = 0.3$



$p(\text{match}) = 0.99$



reference

3

$p(\text{match}) = 0.98$



$p(\text{match}) = 0.99$

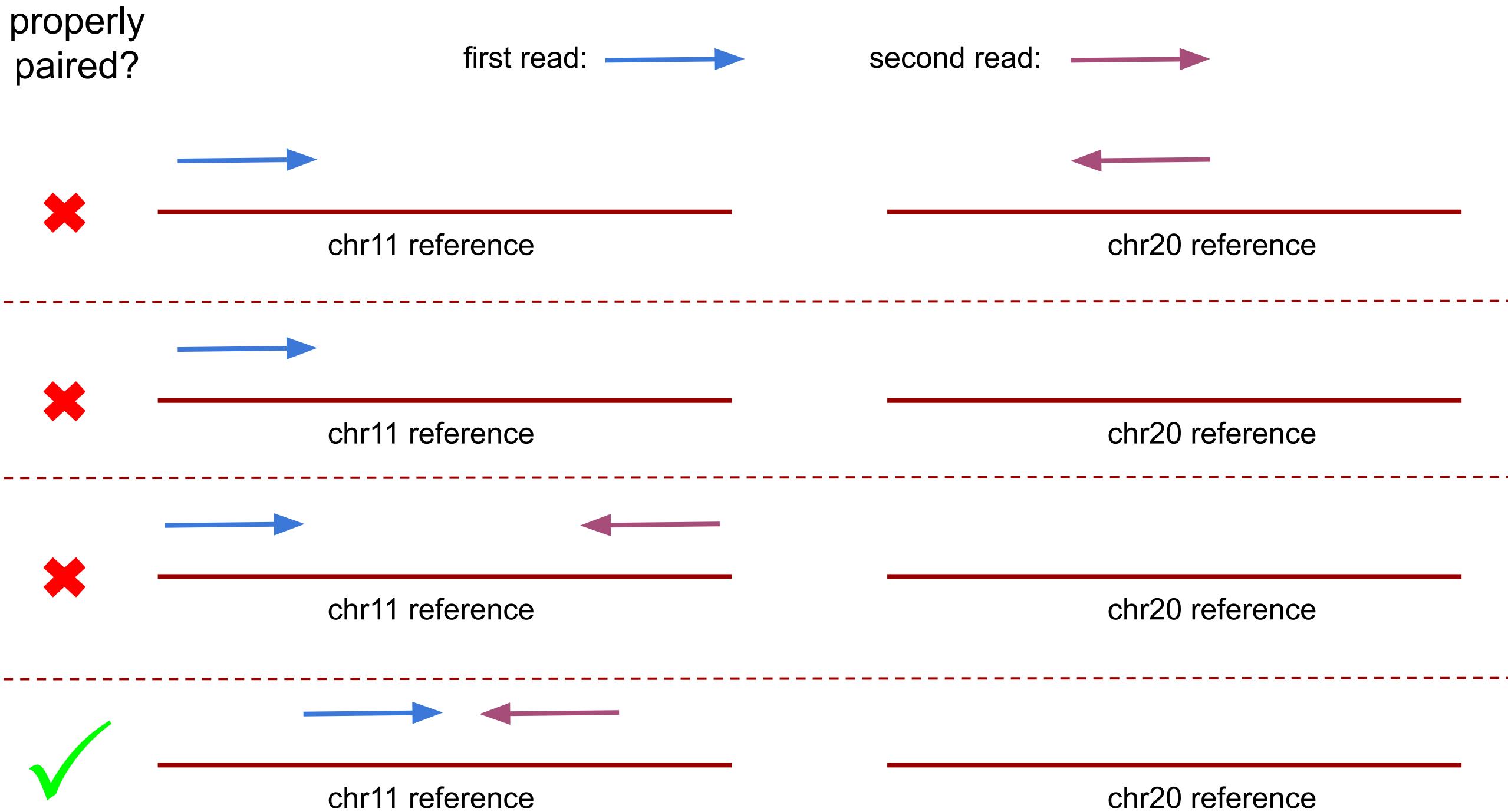


reference



Intro to “proper pairs” vs “unpaired”

- Some aligners add an extra flag to indicate that 2 paired reads were found:
 - on the same chromosome
 - facing each other (one + strand, the other - strand)
 - within a “reasonable” distance

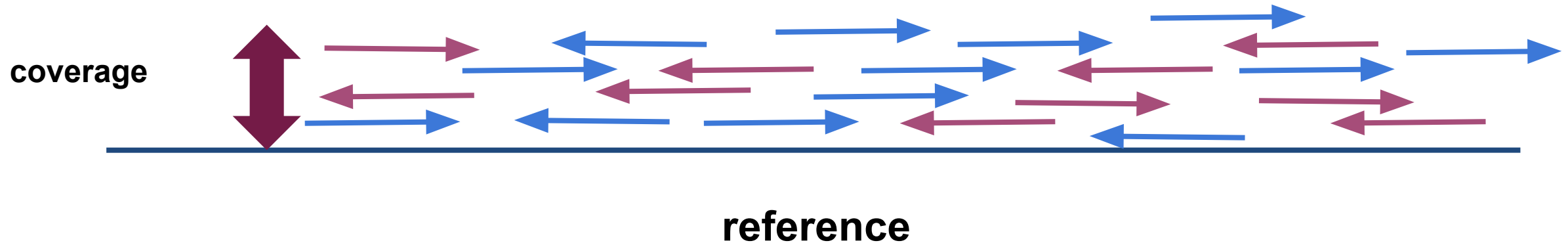


mapping quality vs mappability

- Mapping quality is often (poorly) approximated for speed
- Use another technique to avoid spurious mappings: genomic mappability
- Mapping quality is per read
- Mappability is for a genomic region

Coverage

- Coverage/depth is how many times that your data covers the genome (on average)



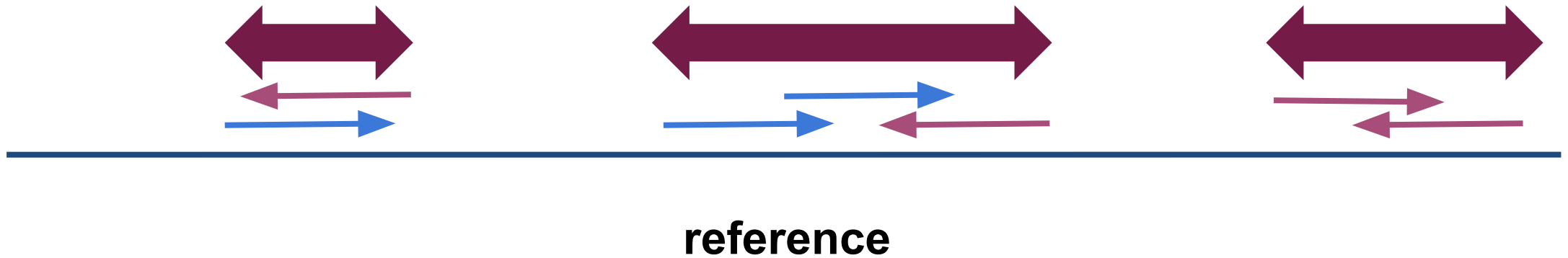
Coverage

- Coverage/depth is how many times that your data covers the genome (on average)
- Example:
 - N : Number of reads: 5M
 - L : Read length: 100
 - G : Genome size: 5Mbp
 - $C = 5M * 100 / 5M = 100X$
 - On average there are 100 reads covering each position in the genome

$$C = N \times \frac{L}{G}$$

Breadth of coverage

- breadth of coverage: fraction of the genome covered (1X or more)



SAM/BAM format

- Sequence Alignment / Map format
- BAM = Binary SAM and zipped - always convert to BAM
- Two sections
 - Header: All lines start with “@”
 - Alignments: All other lines

@SQ SN:mito_ref LN:16569

@PG ID:bwa PN:bwa VN:0.7.17 CL:bwa samse human_MT.fa test_cut.sai input.fq.gz

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

The header section

```

@SQ SN:mito_ref LN:16569
@PG ID:bwa PN:bwa VN:0.7.17 CL:bwa samse human MT.fa test cut.sai input.fq.gz
HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFJJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

The alignment section

```
@SQ SN:mito_ref LN:16569
@PG ID:bwa PN:bwa VN:0.7.17 CL:bwa samse human_MT.fa test_cut.sai input.fq.gz
```

Contains information like:

- What command line was used to generate this SAM/BAM file?
- What does the reference genomes look like? (chromosome names+length)

@SQ SN:mito_ref LN:16569

@PG ID:bwa PN:bwa VN:0.7.17 CL:bwa samse human_MT.fa test_cut.sai input.fq.gz

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFJJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

HN3BX:8:2115:3985:4426	16	mito_ref	5265	37	12M	*	0	0	ATTATCGAAGAA	JJJJJAFFFAAA	NM:i:0	MD:Z:12	RG:Z:sample1	X0:i
HN3BX:8:1102:2678:3157	0	mito_ref	9373	0	9M	*	0	0	AATGATGAC	AAFFFJJJJ	NM:i:1	MD:Z:7G1	RG:Z:sample1	X0:i
HN3BX:8:1217:1588:1335	16	mito_ref	7241	15	9M	*	0	0	ATACACCAC	JJJJFFFAA	NM:i:0	MD:Z:9	RG:Z:sample2	X0:i
HN3BX:8:2216:6248:1342	16	mito_ref	14440	0	5M	*	0	0	ATACT	FFFAA	NM:i:0	MD:Z:5	RG:Z:sample1	X0:i
HN3BX:8:1222:8146:4237	0	mito_ref	1994	0	8M	*	0	0	AACCTACC	AAFFFJJJJ	NM:i:0	MD:Z:8	RG:Z:sample1	X0:i

Read IDs (should be unique)

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFJJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

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

ex:

- 0 = single-end read, mapped, read mapping to + strand
- 16 = single-end read, mapped, read mapping to - strand

```

HN3BX:8:2115:3985:4426 15 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9173 0 9M * 0 0 AATGATGAC AAFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 15 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 15 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1594 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i
  
```

name of chromosome (ex: chr1, chr2 ...)

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFJJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

coordinate on chromosome


```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFJJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 AACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

mapping quality

ex:

MQ = prob of correct mapping

37 = 99.98%

15 = 96.84%

0 = 00.00%

HN3BX:8:2115:3985:4426	16	mito_ref	5265	37	12M	0	0	ATTATCGAAGAA	JJJJJAFFFAAA	NM:i:0	MD:Z:12	RG:Z:sample1	X0:i
HN3BX:8:1102:2678:3157	0	mito_ref	9373	0	9M	0	0	AATGATGAC	AAFFFJJJJ	NM:i:1	MD:Z:7G1	RG:Z:sample1	X0:i
HN3BX:8:1217:1588:1335	16	mito_ref	7241	15	9M	0	0	ATACACCAC	JJJJFFFAA	NM:i:0	MD:Z:9	RG:Z:sample2	X0:i
HN3BX:8:2216:6248:1342	16	mito_ref	14440	0	5M	0	0	ATACT	FFFAA	NM:i:0	MD:Z:5	RG:Z:sample1	X0:i
HN3BX:8:1222:8146:4237	0	mito_ref	1994	0	8M	0	0	AACCTACC	AAFFFJJJJ	NM:i:0	MD:Z:8	RG:Z:sample1	X0:i

CIGAR

Number of operations:

matches (**M**)

insertions (**I**)

deletions (**D**)

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 1 M * 0 ATTATCGAAGAA JJJJJAFFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 M * 0 AATGATGAC AAFFFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 M * 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 M * 0 AACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 M * 0 AACCTACC AAFFFFJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

chromosome of the other pair (not used here)

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M 0 0 ATTATCGAAGAA JJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M 0 0 AATGATGAC AAFFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M 0 0 AACCTACC AAFFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

position on the chromosome of the other pair (not used here)

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFJJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 AATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

length between pairs (not used here)

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AFFFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AFFFFJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i
  
```

sequence of the reads

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

quality scores of the reads

```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFFAAZ NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFFFJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```

optional flags, tell us:

- how many mismatches in alignment
- total number of matches
- read group


```

HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X:i:i
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFJJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X:i:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X:i:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X:i:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X:i:i

```

multiplexing: to which read group
 does the read belong to?

How to reconstruct the alignment? it's complicated...

```

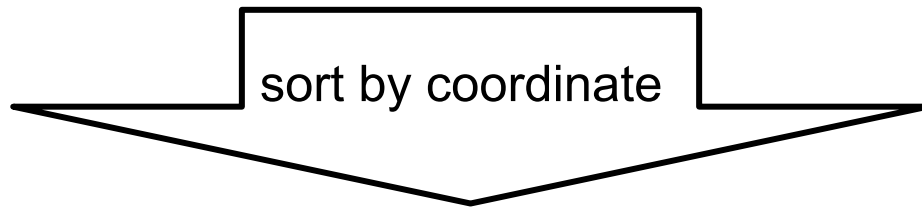
HN3BX:8:2115:3985:4426 16 mito_ref 5265 31 12M 0 0 ATTATCGAAGAA JJJJJAFFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i:1
HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M 0 0 AATGATGAC AAFFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i:1
HN3BX:8:1217:1588:1335 16 mito_ref 7241 11 9M 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i:1
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 AACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i:1
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFFJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i:1
  
```

Ref+ : 9373 AATGATGGC 9381
 | | | | | | | |
 Qry+ : 1 AATGATGAC 9

```

HN3BX:8:1102:2678:3157 0  mito_ref 9373  0  9M * 0 0 AATGATGAC  AAFFFJJJJ  NM:i:1 MD:Z:7G1  RG:Z:sample1 X0:i
HN3BX:8:2115:3985:4426 16  mito_ref 5265  37 12M * 0 0 ATTATCGAAGAA  JJJJJJAFFFAAA  NM:i:0 MD:Z:12  RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16  mito_ref 7241  15  9M * 0 0 ATACACCAC  JJJJFFFAA  NM:i:0 MD:Z:9  RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16  mito_ref 14440  0  5M * 0 0 ATACT  FFFAA  NM:i:0 MD:Z:5  RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0  mito_ref 1994  0  8M * 0 0 AACCTACC  AAFFFJJJJ  NM:i:0 MD:Z:8  RG:Z:sample1 X0:i

```



```

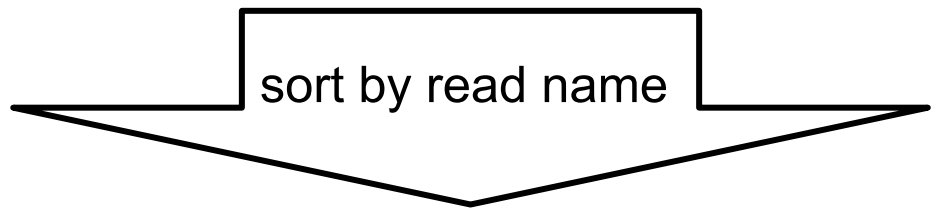
HN3BX:8:1222:8146:4237 0  mito_ref 1994  0  8M * 0 0 AACCTACC  AAFFFJJJJ  NM:i:0 MD:Z:8  RG:Z:sample1 X0:i
HN3BX:8:2115:3985:4426 16  mito_ref 5265  37 12M * 0 0 ATTATCGAAGAA  JJJJJJAFFFAAA  NM:i:0 MD:Z:12  RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16  mito_ref 7241  15  9M * 0 0 ATACACCAC  JJJJFFFAA  NM:i:0 MD:Z:9  RG:Z:sample2 X0:i
HN3BX:8:1102:2678:3157 0  mito_ref 9373  0  9M * 0 0 AATGATGAC  AAFFFJJJJ  NM:i:1 MD:Z:7G1  RG:Z:sample1 X0:i
HN3BX:8:2216:6248:1342 16  mito_ref 14440  0  5M * 0 0 ATACT  FFFAA  NM:i:0 MD:Z:5  RG:Z:sample1 X0:i

```

```

HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i

```



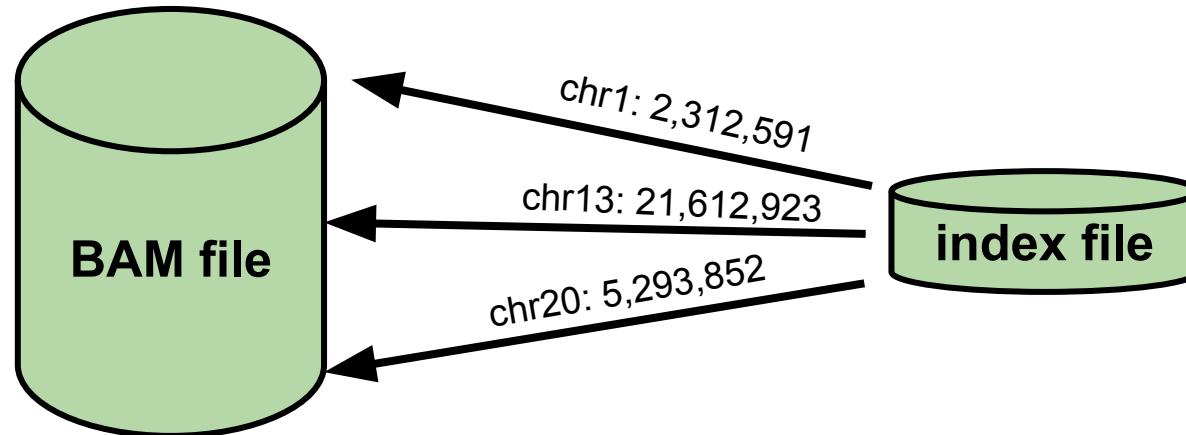
```

HN3BX:8:1102:2678:3157 0 mito_ref 9373 0 9M * 0 0 AATGATGAC AAFFFJJJJ NM:i:1 MD:Z:7G1 RG:Z:sample1 X0:i
HN3BX:8:1217:1588:1335 16 mito_ref 7241 15 9M * 0 0 ATACACCAC JJJJFFFAA NM:i:0 MD:Z:9 RG:Z:sample2 X0:i
HN3BX:8:1222:8146:4237 0 mito_ref 1994 0 8M * 0 0 AACCTACC AAFFFJJJJ NM:i:0 MD:Z:8 RG:Z:sample1 X0:i
HN3BX:8:2115:3985:4426 16 mito_ref 5265 37 12M * 0 0 ATTATCGAAGAA JJJJJJAFFFAAA NM:i:0 MD:Z:12 RG:Z:sample1 X0:i
HN3BX:8:2216:6248:1342 16 mito_ref 14440 0 5M * 0 0 ATACT FFFAA NM:i:0 MD:Z:5 RG:Z:sample1 X0:i

```

BAM indexing

- I want all reads mapping to chrX
- Go through the entire BAM file until we reach chrX
- Better idea: keep a small file telling you where chromosome/coord start:

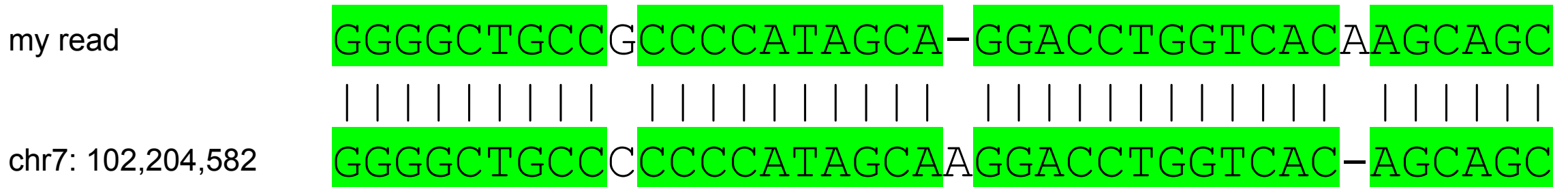


- Need a BAM file sorted by coordinate
- Is usually a .bai or .csi file

BAM vs CRAM

- Idea: why store the original reads? You only the differences to a reference
- Can be 30-60% smaller
- Good for long term storage
- BAM can be converted to CRAM
- Can be indexed

Alignment to a graph, not just a reference



If there are too many mismatches+indels, we might miss the alignment

Exercise time!

http://teaching.healthtech.dtu.dk/22126/index.php/Alignment_exercise