



UNIVERSITY OF UTAH  
SCHOOL OF MEDICINE

Department of  
Human Genetics

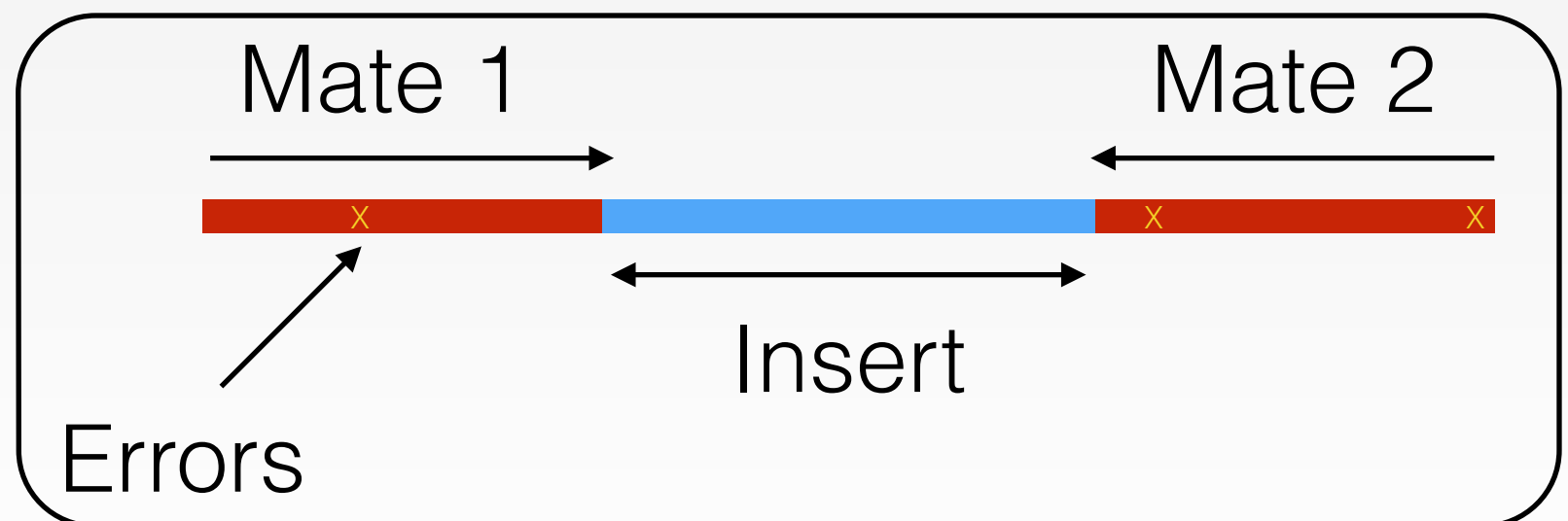
# Sequence mapping and assembly

Alistair Ward  
USTAR Center for Genetic Discovery  
University of Utah

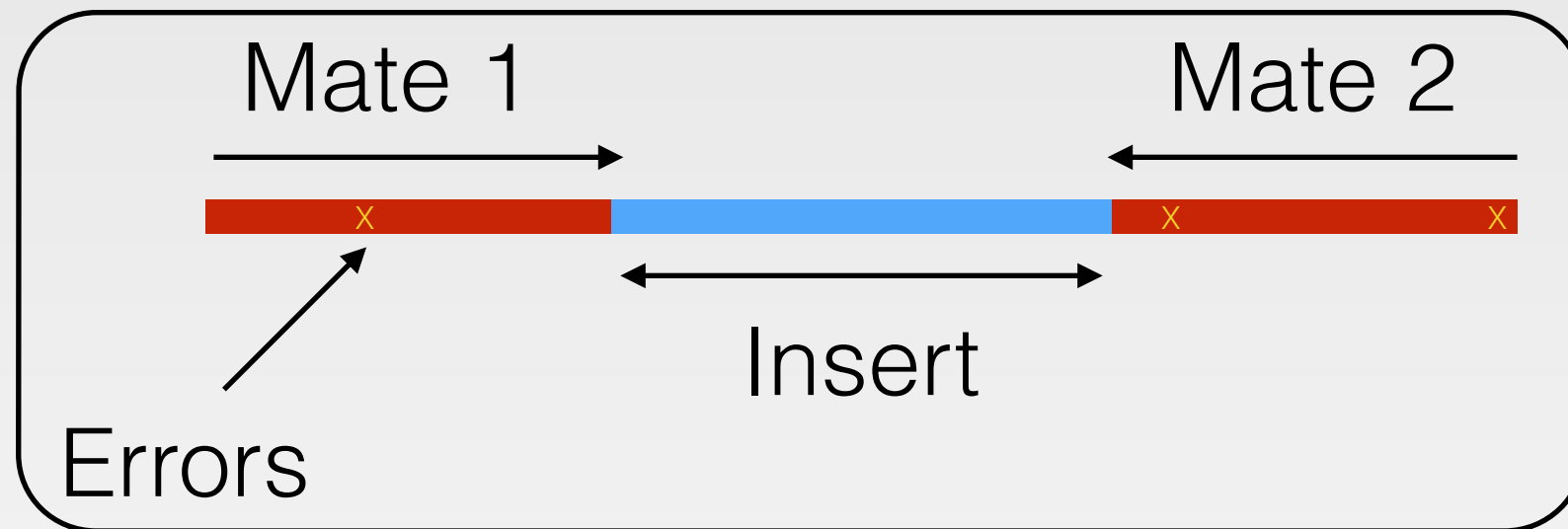
# Sequenced a genome?

- Fragmented a genome -> DNA library
- PCR amplification
- Sequence reads (ends of DNA fragment for mate pairs)
- We no longer have any positional information or relational information between fragments

We have millions/billions of sequenced DNA fragments



# Sequenced a genome?



Stored in a ***fastq*** file

@READ\_NAME/1 ← Unique read name - /1 indicates first mate  
GCACTGTGTGTGCTA ← Read sequence  
+  
IIHIBABIIIBI@BI ← Base qualities

# What we will cover

- Multiple strategies for making sense of the DNA sequences
- Importance of mapping - what is your endgame?
  - Mapping to a reference (resequencing):
    - Traditional mapping (detail) Mosaik, Bwa, Bowtie, Stampy
    - Split-read mapping Scissors, Pindel
    - Graph alignment glia
  - Assembly methods Cortex, Velvet, sga

# Mapping to a reference genome



- This is like a jigsaw puzzle
- Compare reads to a reference genome, accounting for genetic differences
- Two major approaches:
  - Hashing the reference
  - Burrows-Wheeler transform

# Hash based approach

Find all k-mers in the reference genome



GCACTGTGTGTGCTA

Store all positions in a hash table

# Break up reads

- Determine where a read can fit accounting for:
  - Sequencing errors,
  - True genetic differences with the reference
- Break read into hashes

ACACATGTACGTAGTCGTAGTGCTAGTCAGCT – read length n

ACACATGTACGTAGT – hash 1

CACATGTACGTAGTC – hash 2

ACATGTACGTAGTCG – hash 3

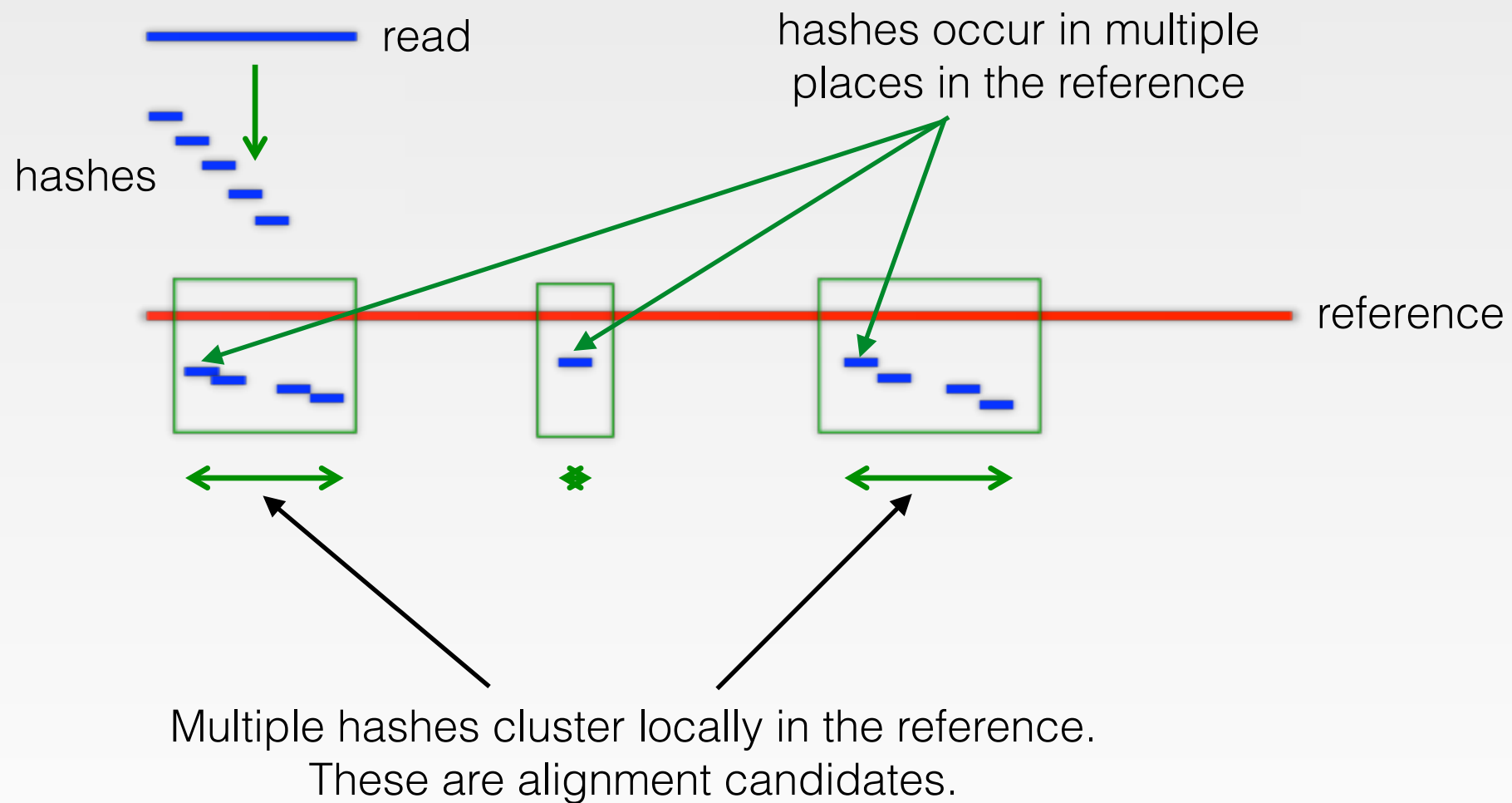
...

GTAGTGCTAGTCAGC – hash n-2

TAGTGCTAGTCAGCT – hash n-1

# Compare read to reference

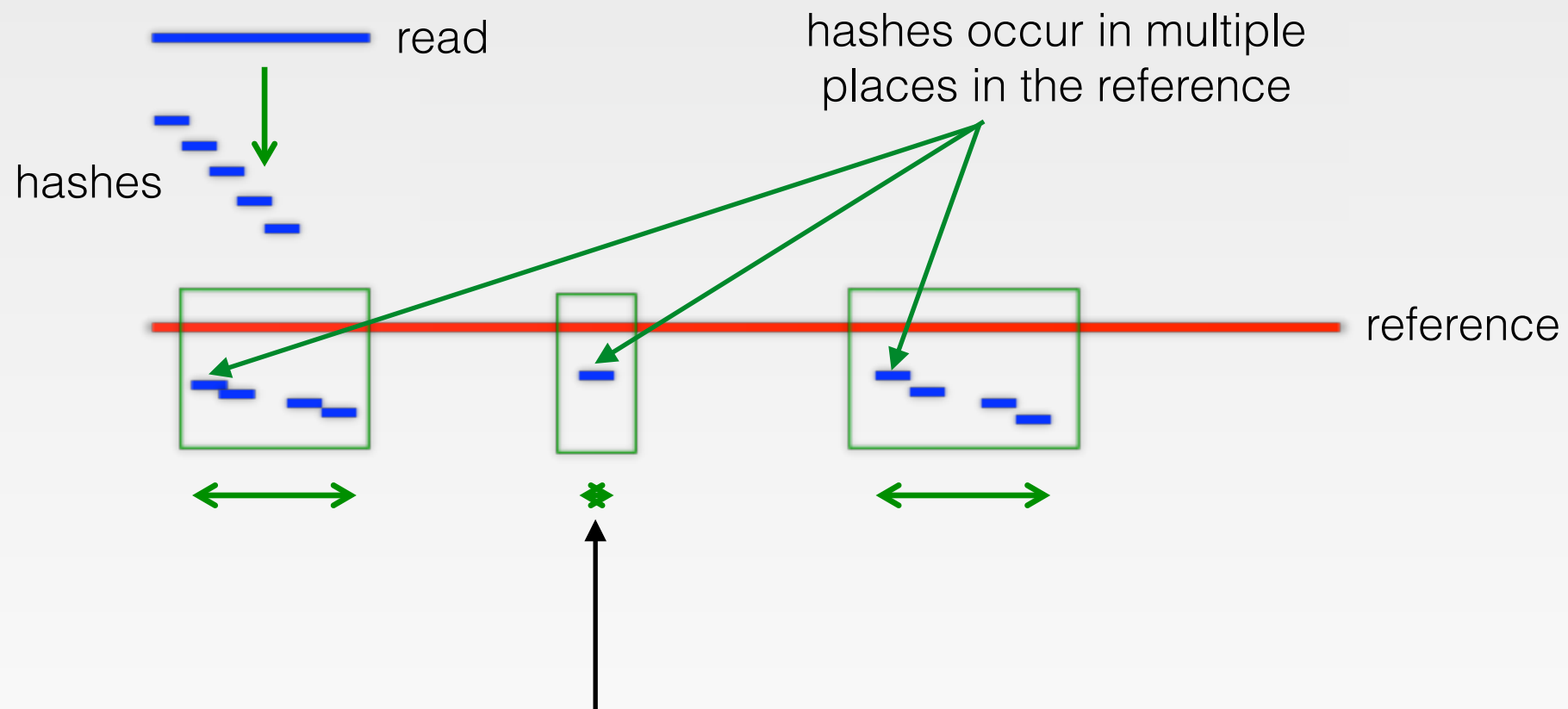
- Find where each hash lands in the reference:





# Compare read to reference

- Find where each hash lands in the reference:



Small clusters of hashes will appear all over the reference.  
These are not alignment candidates.

# Smith Waterman algorithm

- Find the optimal alignment for each candidate.
- Maximise similarity measure between two sequences

# Smith-Waterman example

- Generate a matrix with the sequences to compare
- Populate matrix with scores

$$M(i,0) = 0 \text{ for } 0 \leq i \leq m$$

$$M(0,j) = 0 \text{ for } 0 \leq j \leq n$$

$$M(i,j) = \max \begin{bmatrix} 0 \\ M(i-1, j-1) + s(a_i, b_j) \\ \max_{k \geq 1} \{M(i-k, j) + W_k\} \\ \max_{l \geq 1} \{M(i, j-l) + W_l\} \end{bmatrix}$$

	-	A	...	A
-	M(0,0)	M(1,0)	...	M(i,0)
A	M(0,1)	M(1,1)	...	M(i,1)
⋮	⋮	⋮	⋱	⋮
A	M(0,j)	M(1,j)	...	M(i,j)

# Smith-Waterman example

$M(i,0) = 0$  for  $0 \leq i \leq m$

$M(0,j) = 0$  for  $0 \leq j \leq n$

	-	A	C	A	C	A	C	T	A
-	0	0	0	0	0	0	0	0	0
A	0								
G	0								
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								

# Smith-Waterman example

$$M(i,j) = \max \begin{bmatrix} 0 \\ M(i-1, j-1) + s(a_i, b_j) \\ \max_{k \geq 1} \{M(i-k, j) + W_k\} \\ \max_{l \geq 1} \{M(i, j-l) + W_l\} \end{bmatrix}$$

$$M(i-1, j-1) + s(a_i, b_j)$$

$s(a_i, b_j) = +2$  if  $a = b$       Match  
 $s(a_i, b_j) = -1$  if  $a \neq b$       Mismatch

$$M(1, 1) = +2$$

	-	A	C	A	C	A
-	0	0	0	0	0	0
A	0	M(1,1)				
G	0					
C	0					
A	0					
C	0					
A	0					
C	0					
A	0					

# Smith-Waterman example

$$M(i,j) = \max \begin{bmatrix} 0 \\ M(i-1, j-1) + s(a_i, b_j) \\ \max_{k \geq 1} \{M(i-k, j) + W_k\} \\ \max_{l \geq 1} \{M(i, j-l) + W_l\} \end{bmatrix}$$

$$M(i-1, j-1) + s(a_i, b_j)$$

$s(a_i, b_j) = +2$  if  $a = b$       Match  
 $s(a_i, b_j) = -1$  if  $a \neq b$       Mismatch

$$M(1, 1) = +2$$

	-	A	C	A	C	A
-	0	0	0	0	0	0
A	0	2				
G	0					
C	0					
A	0					
C	0					
A	0					
C	0					
A	0					

# Smith-Waterman example

$$M(i,j) = \max \begin{bmatrix} 0 \\ M(i-1, j-1) + s(a_i, b_j) \\ \max_{k \geq 1} \{M(i-k, j) + W_k\} \\ \max_{l \geq 1} \{M(i, j-l) + W_l\} \end{bmatrix}$$

Insertion or deletion scoring

$$W_i = -1$$

	-	A	C	A	C	A
-	0	0	0	0	0	0
A	0	2	M(2,1)			
G	0					
C	0					
A	0					
C	0					
A	0					
C	0					
A	0					

# Smith-Waterman example

	-	A	C	A	C	A	C	T	A
-	0	0	0	0	0	0	0	0	0
A	0	2	1						
G	0								
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								



# Smith-Waterman example

	-	A	C	A	C	A	C	T	A
-	0	0	0	0	0	0	0	0	0
A	0	2	1	2					
G	0								
C	0								
A	0								
C	0								
A	0								
C	0								
A	0								

# Smith-Waterman example

	-	A	C	A	C	A	C	T	A
-	0	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0	2
G	0	1	1	1	1	1	1	0	1
C	0	0	3	2	3	2	3	2	1
A	0	2	2	5	4	5	4	3	4
C	0	1	4	4	7	6	7	6	5
A	0	2	3	6	6	9	8	7	8
C	0	1	4	5	8	8	11	10	9
A	0	2	3	6	7	10	10	10	12

# Traceback

- Start at highest value
- Diagonal line is a match/mismatch
- Up/down or left/right are indels




Sequence 1  
A-CACACTA

Sequence 2  
AGCACAC-A

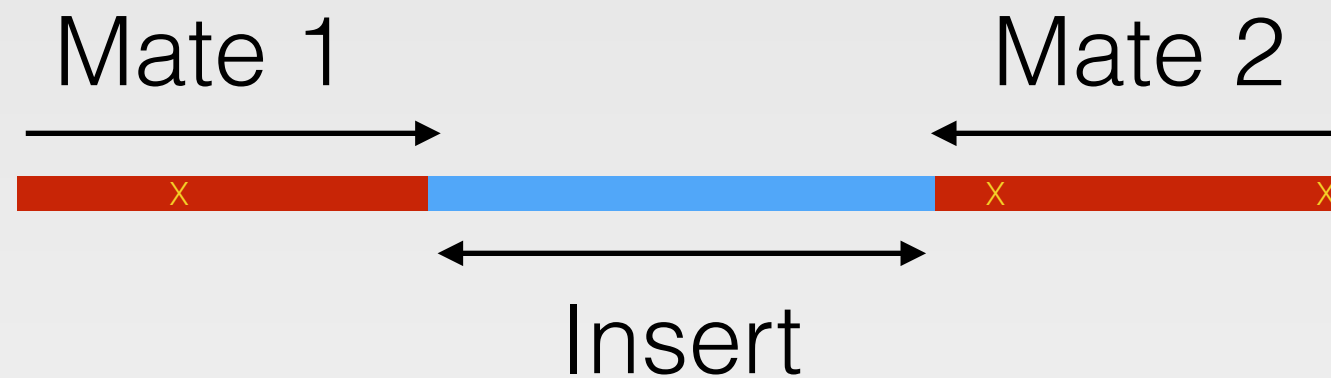
	-	A	C	A	C	A	C	T	A
-	0	0	0	0	0	0	0	0	0
A	0	2	1	2	1	2	1	0	2
G	0	1	1	1	1	1	1	0	1
C	0	0	3	2	3	2	3	2	1
A	0	2	2	5	4	5	4	3	4
C	0	1	4	4	7	6	7	6	5
A	0	2	3	6	6	9	8	7	8
C	0	1	4	5	8	8	11	10	9
A	0	2	3	6	7	10	10	10	12




# Paired end reads

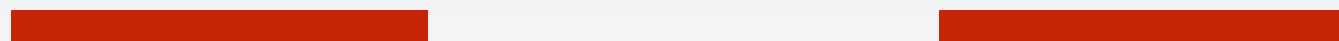


-  mapped uniquely
  -  mapped to multiple locations
  -  unmapped
-

# Paired end reads

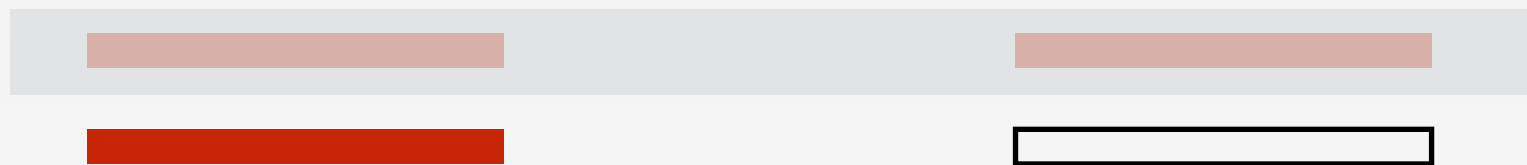
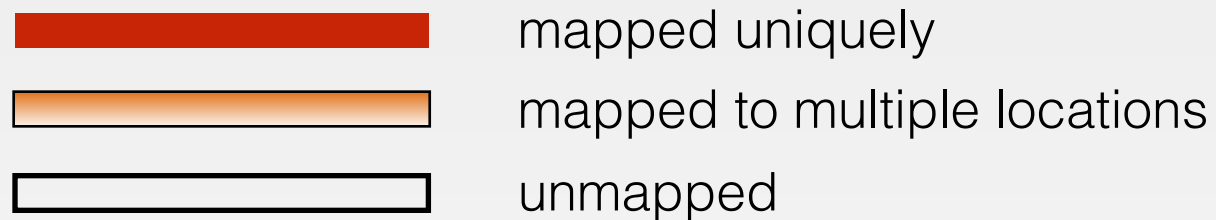
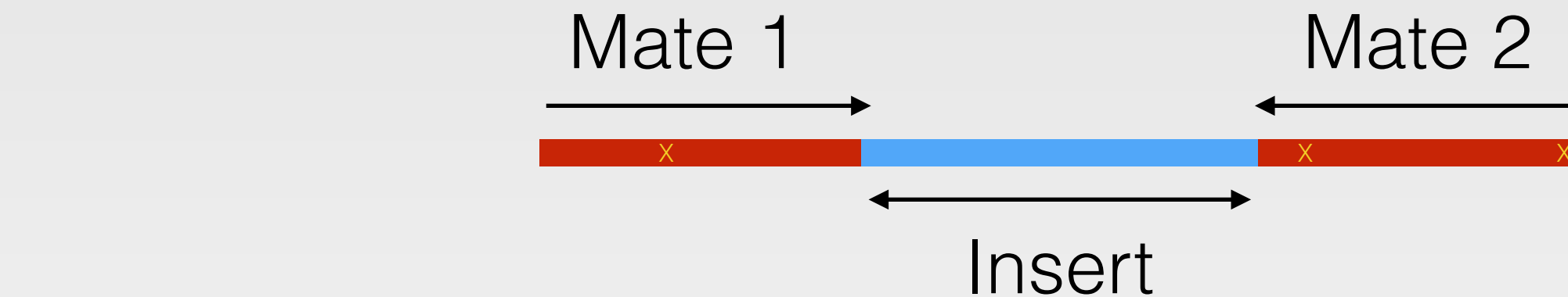


-  mapped uniquely
-  mapped to multiple locations
-  unmapped



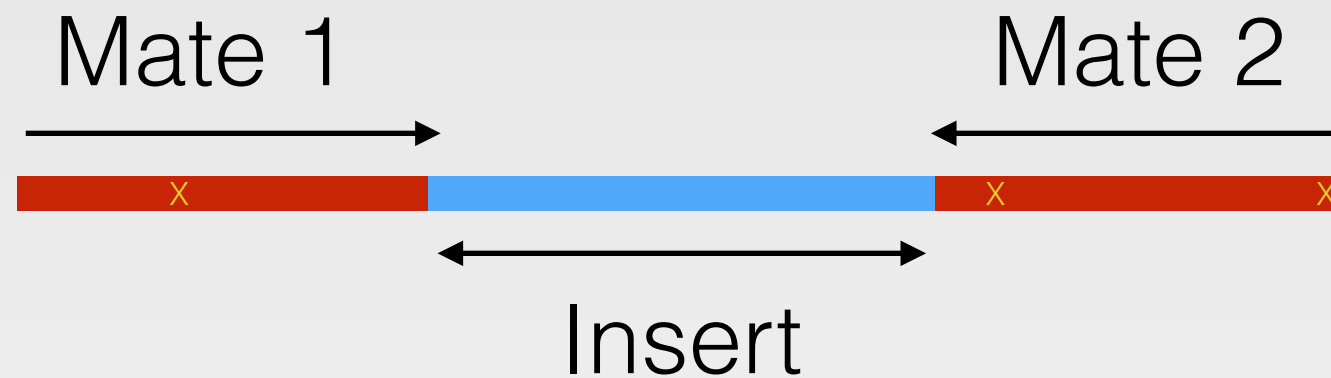
Both mates map uniquely




# Paired end reads

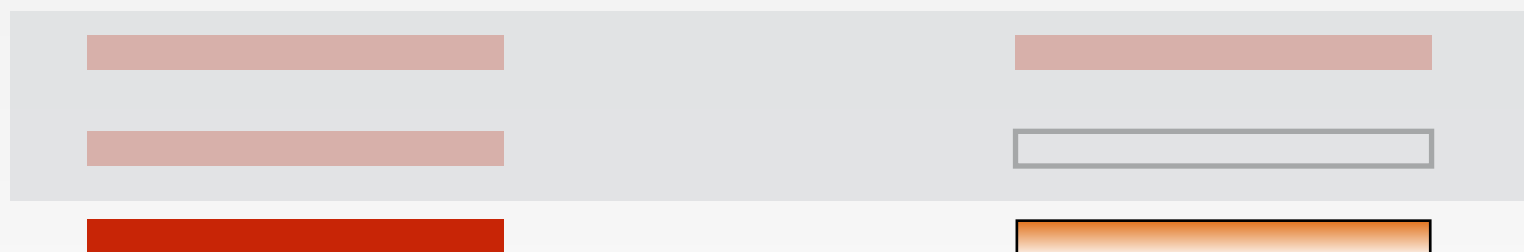


One mate maps uniquely, the other is unmapped

# Paired end reads

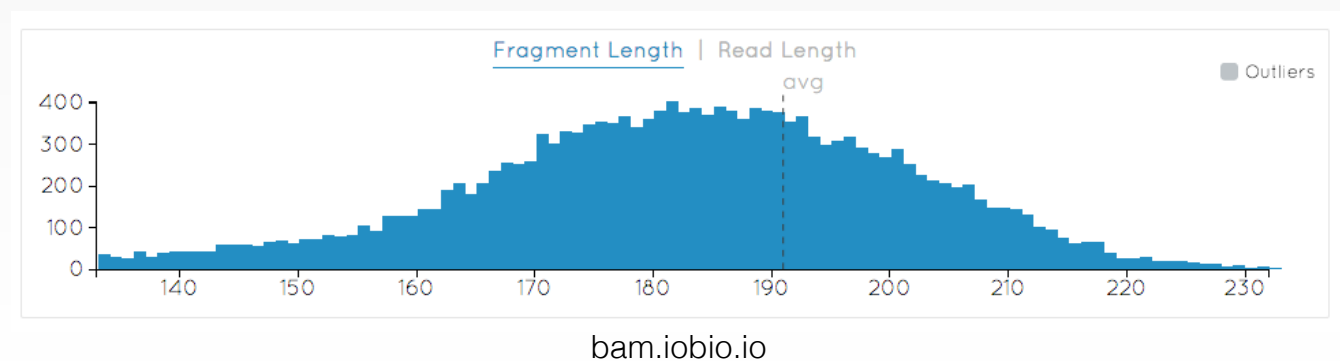


-  mapped uniquely
-  mapped to multiple locations
-  unmapped



One mate maps uniquely, the other maps to multiple locations

Use fragment length distribution to determine most likely location



# SAM format

<http://samtools.github.io/hts-specs/SAMv1.pdf>

Version (VN) and sort order (SO) -  
Important!

Reference sequence (SQ)  
and sequence length (LN)

```
@HD      VN:1.3  SO:coordinate
@SQ      SN:20  LN:63025520
@RG      ID:HG00096  SM:HG00096
@PG      ID:HG00096  PN:bwa  CL:/Users/AlistairNWard/Work/gkno/gkno_launcher/tools/bwa/bwa mem -t 8
```

Read group (RG) and  
sample (SM)

Programs (PG) that have  
been run on the data



# SAM format

<http://samtools.github.io/hts-specs/SAMv1.pdf>

Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	<code>[!-?A-~]{1,255}</code>	Query template NAME
2	FLAG	Int	<code>[0,2<sup>16</sup>-1]</code>	bitwise FLAG
3	RNAME	String	<code>\*  [!-( )+-&lt;&gt;-~] [!-~]*</code>	Reference sequence NAME
4	POS	Int	<code>[0,2<sup>31</sup>-1]</code>	1-based leftmost mapping POSition
5	MAPQ	Int	<code>[0,2<sup>8</sup>-1]</code>	MAPping Quality
6	CIGAR	String	<code>\*  ([0-9]+[MIDNSHPX=])+</code>	CIGAR string
7	RNEXT	String	<code>\* =  [!-( )+-&lt;&gt;-~] [!-~]*</code>	Ref. name of the mate/next read
8	PNEXT	Int	<code>[0,2<sup>31</sup>-1]</code>	Position of the mate/next read
9	TLEN	Int	<code>[-2<sup>31</sup>+1,2<sup>31</sup>-1]</code>	observed Template LENgth
10	SEQ	String	<code>\*  [A-Za-z=.]+</code>	segment SEQUENCE
11	QUAL	String	<code>[!-~]+</code>	ASCII of Phred-scaled base QUALity+33

1	2	3	4	5	6	7	8	9	10
SRR062634.14576120	163	20	899919	60	100M	=	900037	218	TTCCCCAGTAGCTGGGATTACAGGCATACGCCACCATC

?

?

# Flag

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




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

Secondary alignment - an alternative location for the read (sequence and quality strings are '\*')

Supplementary alignments - read is split into a set of alignments. All but one are represented as supplementary

# Flag

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



← → ↻ 🏠 [broadinstitute.github.io/picard/explain-flags.html](http://broadinstitute.github.io/picard/explain-flags.html)

This utility explains SAM flags in plain English.

Flag:

Explanation:

- ☒ 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

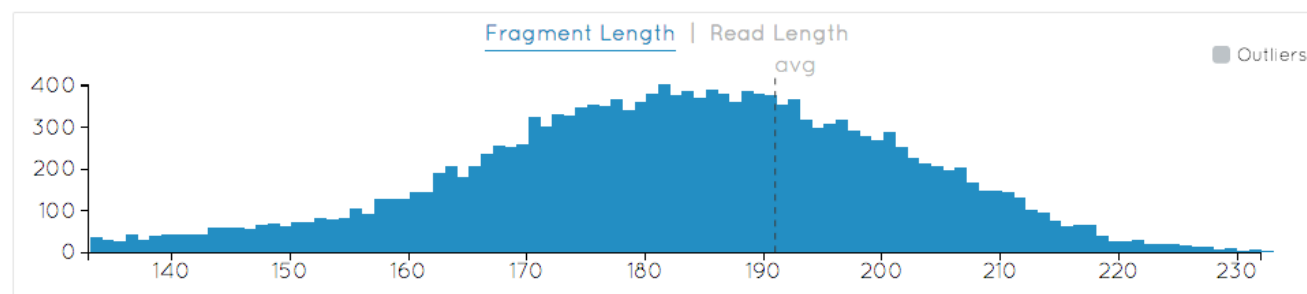
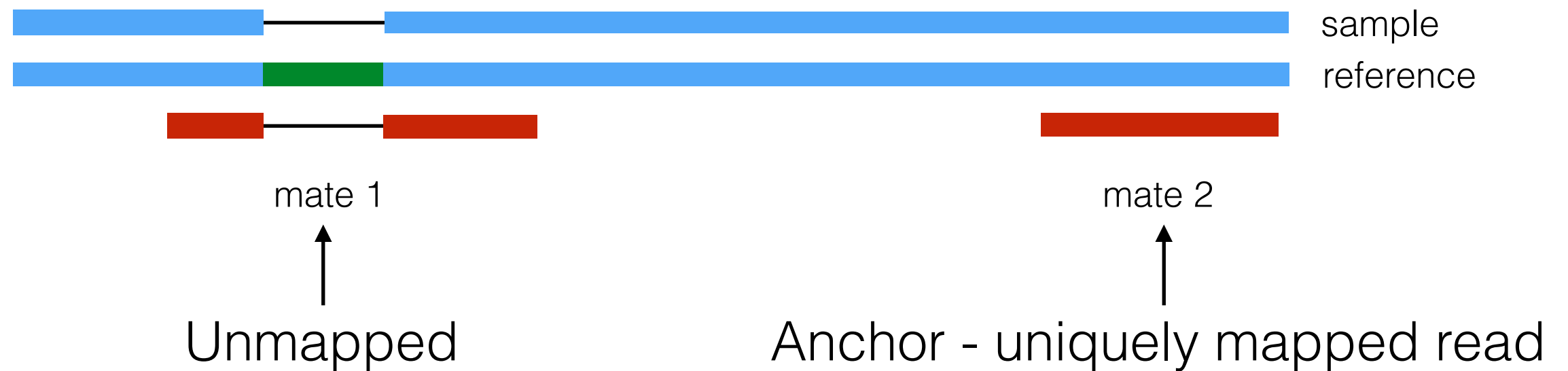
Summary:

- read paired
- read mapped in proper pair
- mate reverse strand
- second in pair

Secondary alignment  
quality strings are

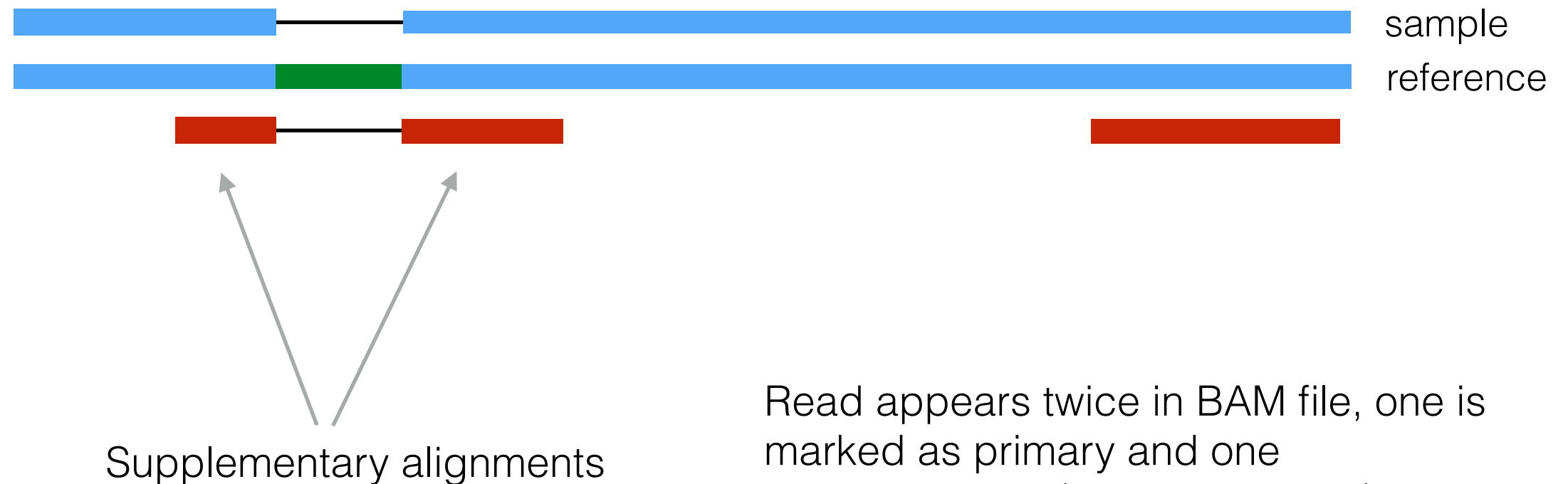
Supplementary alignments - read is split into a set of alignments. All but one are represented as supplementary

# Split read mapping



Estimate of fragment length -  
we have an idea of where to look

# Split read mapping



Read appears twice in BAM file, one is marked as primary and one supplementary (arbitrary choice)

Hard clipped reads

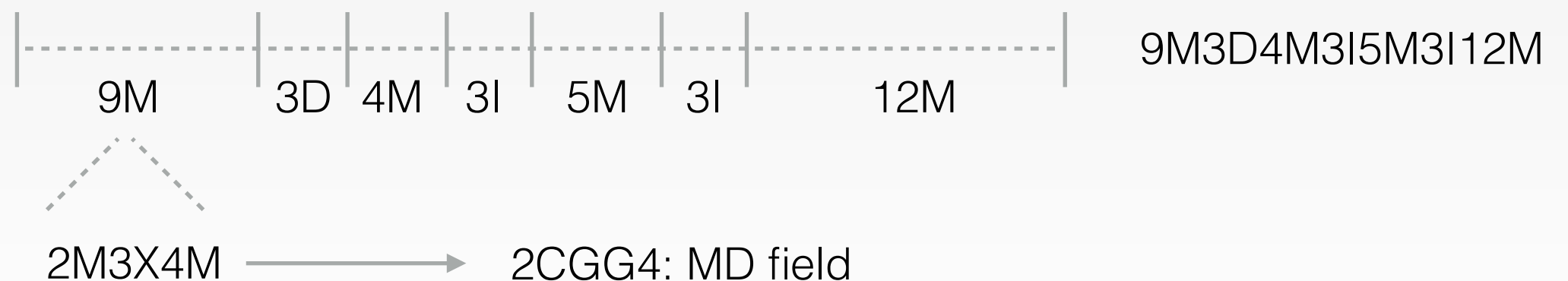
e.g. 30M70H and 30H70M

# CIGAR string

Op	BAM	Description
M	0	alignment match (can be a sequence match or mismatch)
I	1	insertion to the reference
D	2	deletion from the reference
N	3	skipped region from the reference
S	4	soft clipping (clipped sequences present in SEQ)
H	5	hard clipping (clipped sequences NOT present in SEQ)
P	6	padding (silent deletion from padded reference)
=	7	sequence match
X	8	sequence mismatch

Reference: ACTTTTCATCCCTAAA---CAACC---CTGTGTTTTCCC

Sample: AC**CGG**TCAT---TAA**TT**CAACC**CTT**CTGTG**AAA**TCCC



# CIGAR - Clipped reads

Reduced coverage



e.g. 20S80M

e.g. 84M16S

# Mapping quality

An important quantity attached to each mapped read:

The probability that a read is **incorrectly** placed

$$Q = -\log_{10}P$$

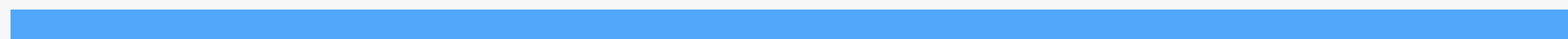
Q is the Phred score

Q = 30 means there is a 1 in 1000 chance  
that the read is misaligned



# Parameters

- Can you just use an aligner out of the box?
- Yes, but it is wise to understand what parameters are doing
- What are you looking for?



reference



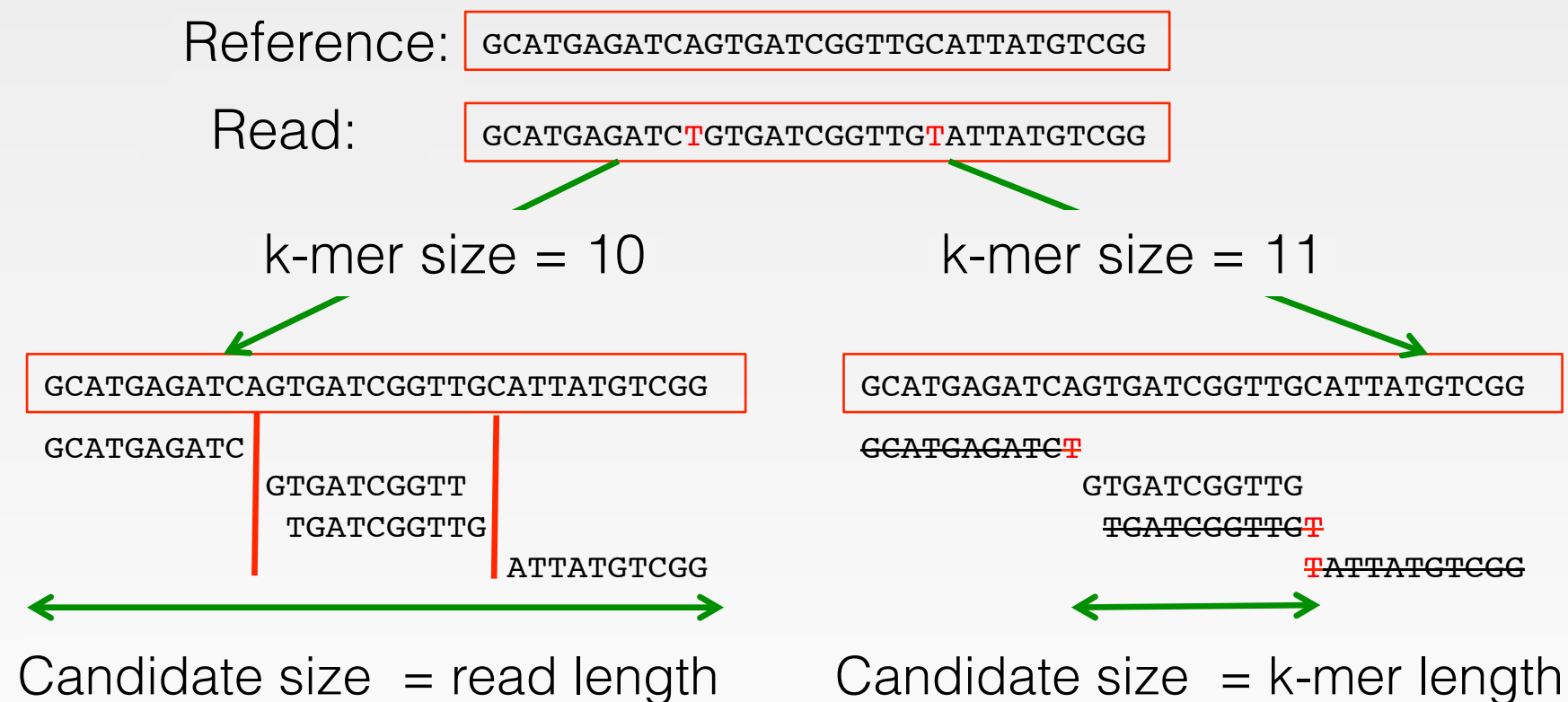
SNPs



Deletions

# Parameters - k-mer size

Short reads - choice of k-mer size is important



# Burrows-Wheeler

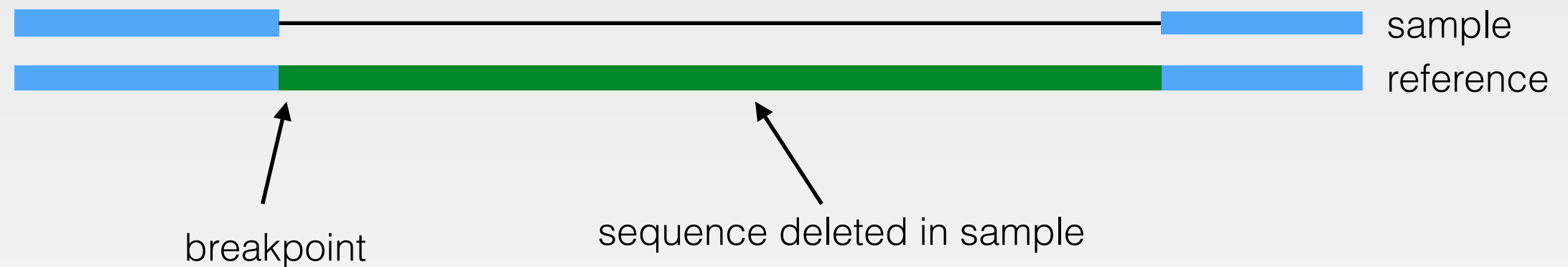
- Align the query sequence against the suffix tree of the reference
- Represent the suffix tree with an FM-index using the Burrows-Wheeler transform
  - Reduces the memory footprint

# Mapping pros and cons

- Vast majority of sample sequence can be accurately placed
- Small differences (variants or errors) can be easily identified
- Problems with:
  - Large scale differences - structural variation
  - Reference bias
  - Repetitive DNA
- How can we address these shortcomings?

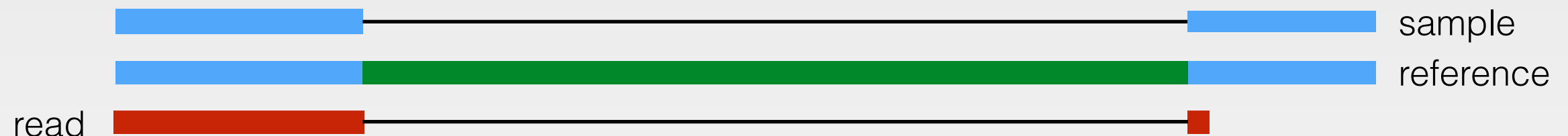
# Mapping across a deletion

- A read straddles a deletion



# Mapping across a deletion

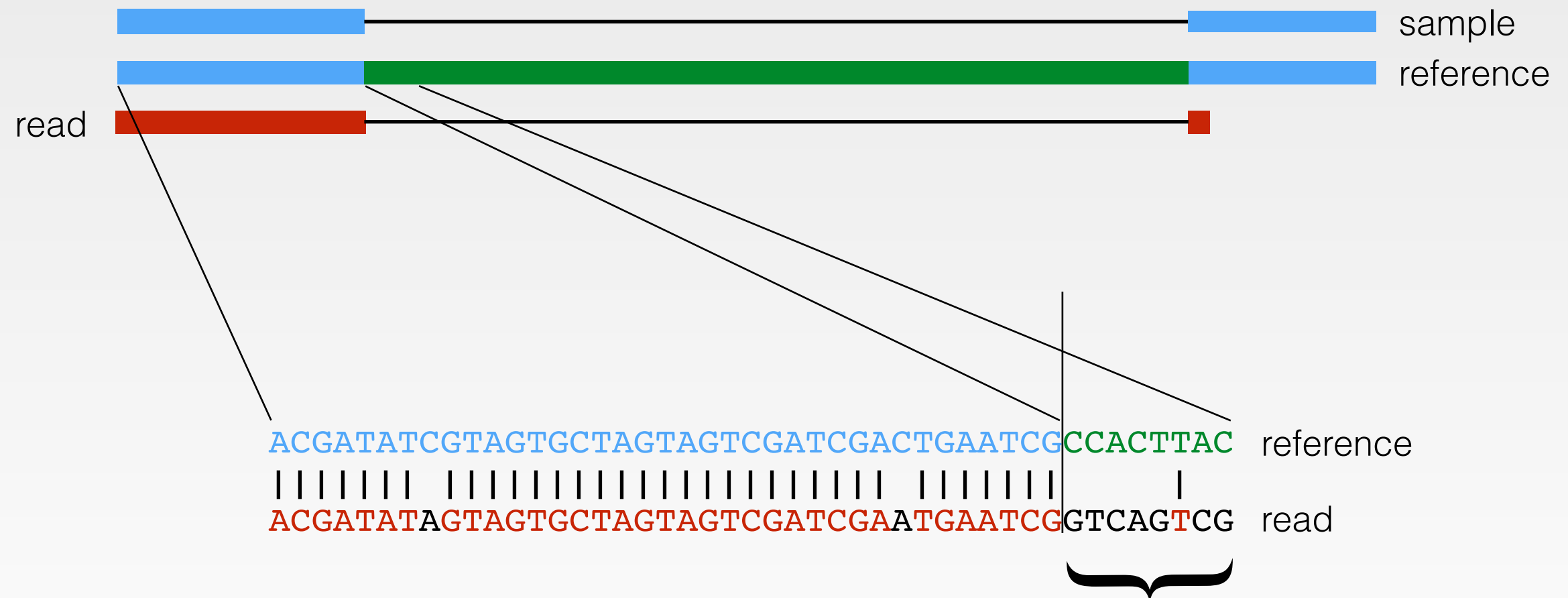
- A read straddles a deletion



What happens if we map this read to the reference?

# Successful mapping

- A read straddles a deletion



# Mapping across a deletion

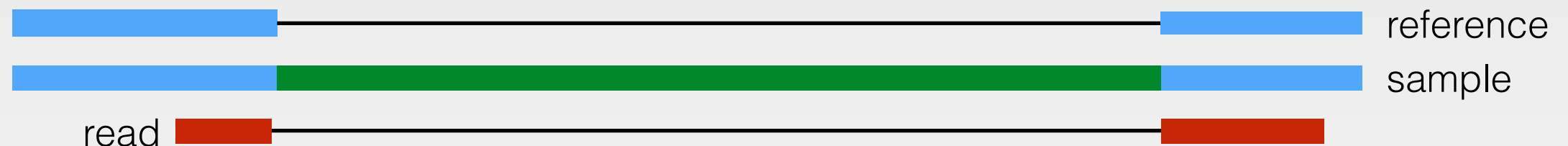
- A read straddles a deletion





# Failed mapping

- A read straddles a deletion

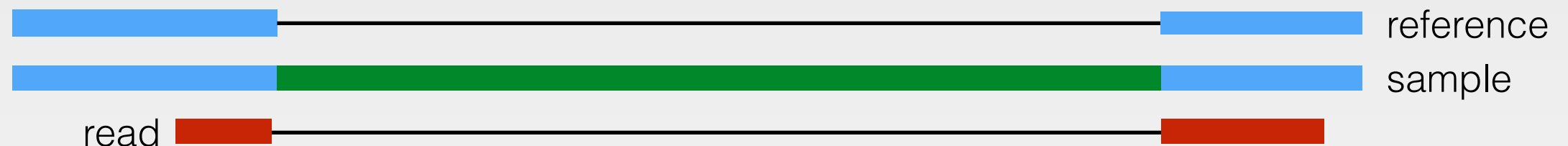


ACGATATCGTA	CTGACTGACTGACTGACTGGCGGCGTCTTGAGCC	reference
ACGATATAGTA	TCCCTGCGGCATACCTCACATTCAAGTCAGTCG	read

This read cannot be mapped

# Failed mapping

- A read straddles a deletion

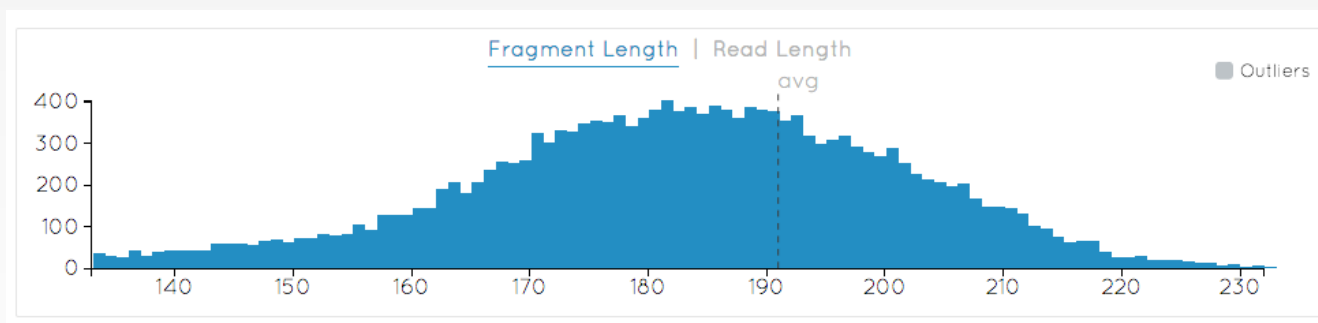
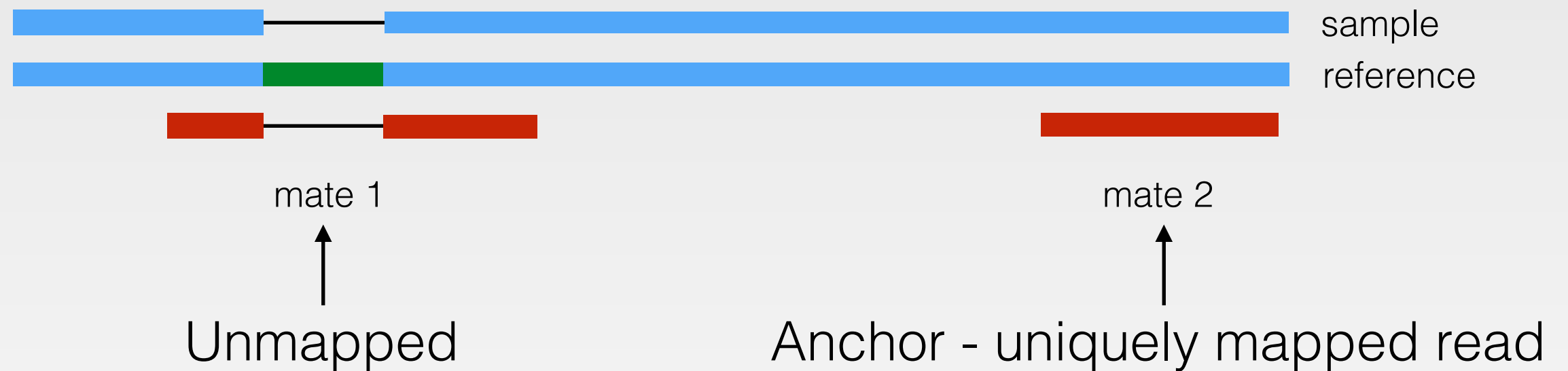


ACGATATCGTA	CTGACTGACTGACTGACTGGCGGCGTCTTGAGCC	reference
ACGATATAGTA	TCCCTGCGGCATACCTCACATTCAAGTCAGTCG	read

This read cannot be mapped

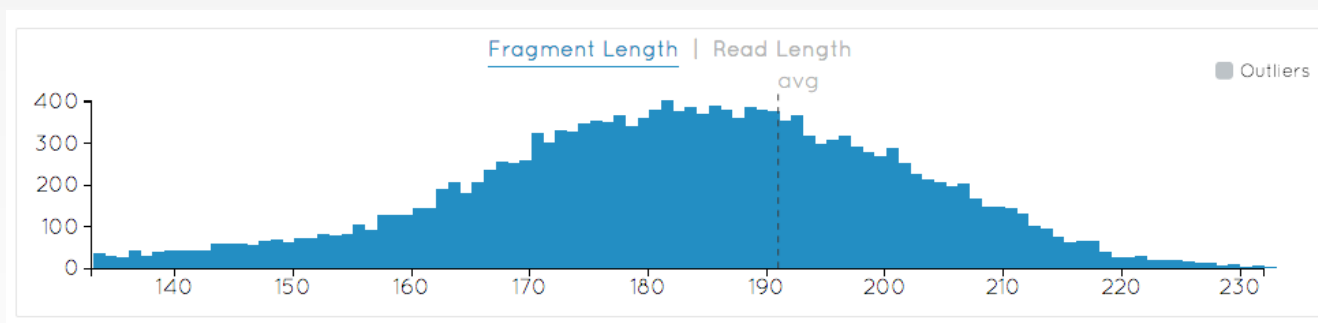
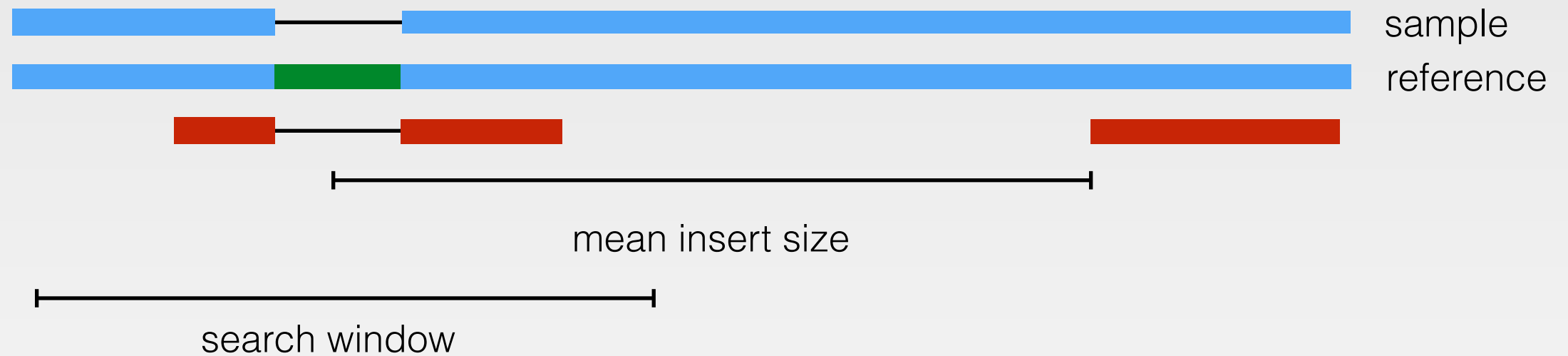
Or can it?

# Split read mapping



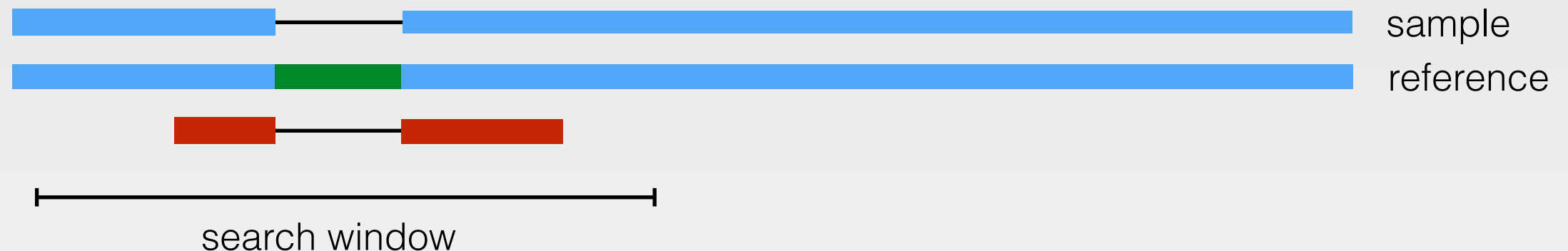
Estimate of fragment length -  
we have an idea of where to look

# Split read mapping



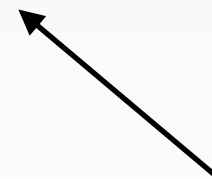
Estimate of fragment length -  
we have an idea of where to look

# Mapping across a deletion



Use Smith-Waterman algorithm across a window

Match: 30 (10)  
Mismatch: -60 (-9)  
Open gap: -60 (-15)  
Extend gap: -1 (-1)



Opening a gap is not penalized more than a mismatch

# Mapping strategies

Try to map the read assuming that the sample contains one of the following structural variants

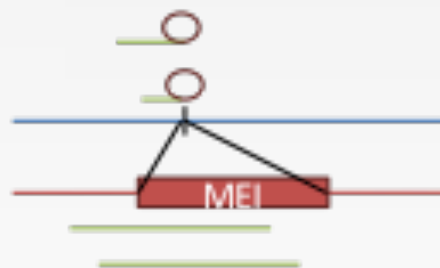
Sample contains a deletion



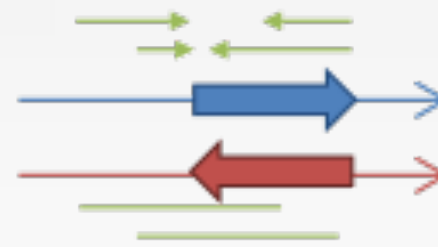
Sample contains inserted sequence



Sample contains mobile element



Sample contains an inversion



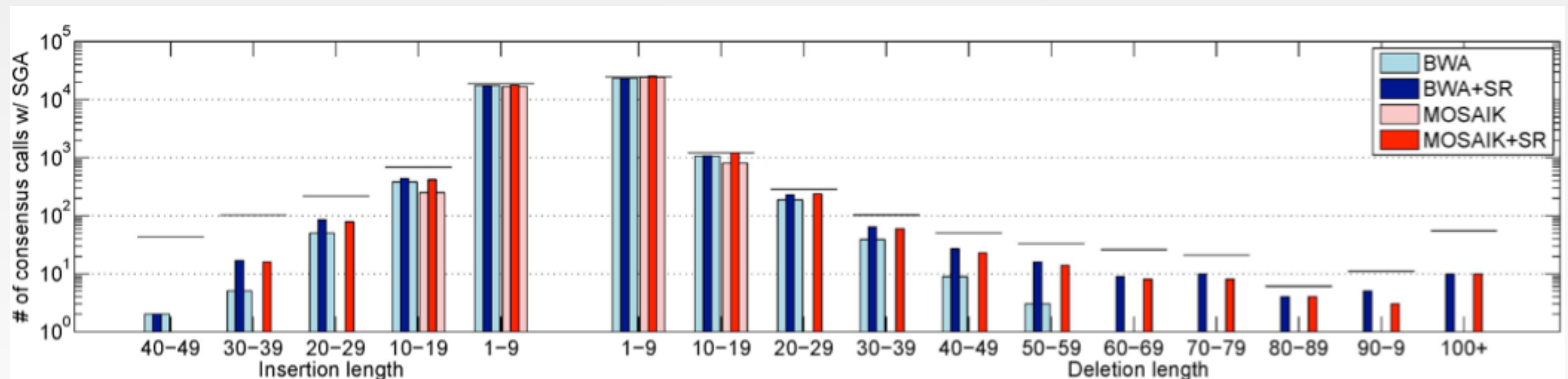
reference

sample

# Does it work

Call indels in AFR samples from 1000 Genomes Project

SR = split read

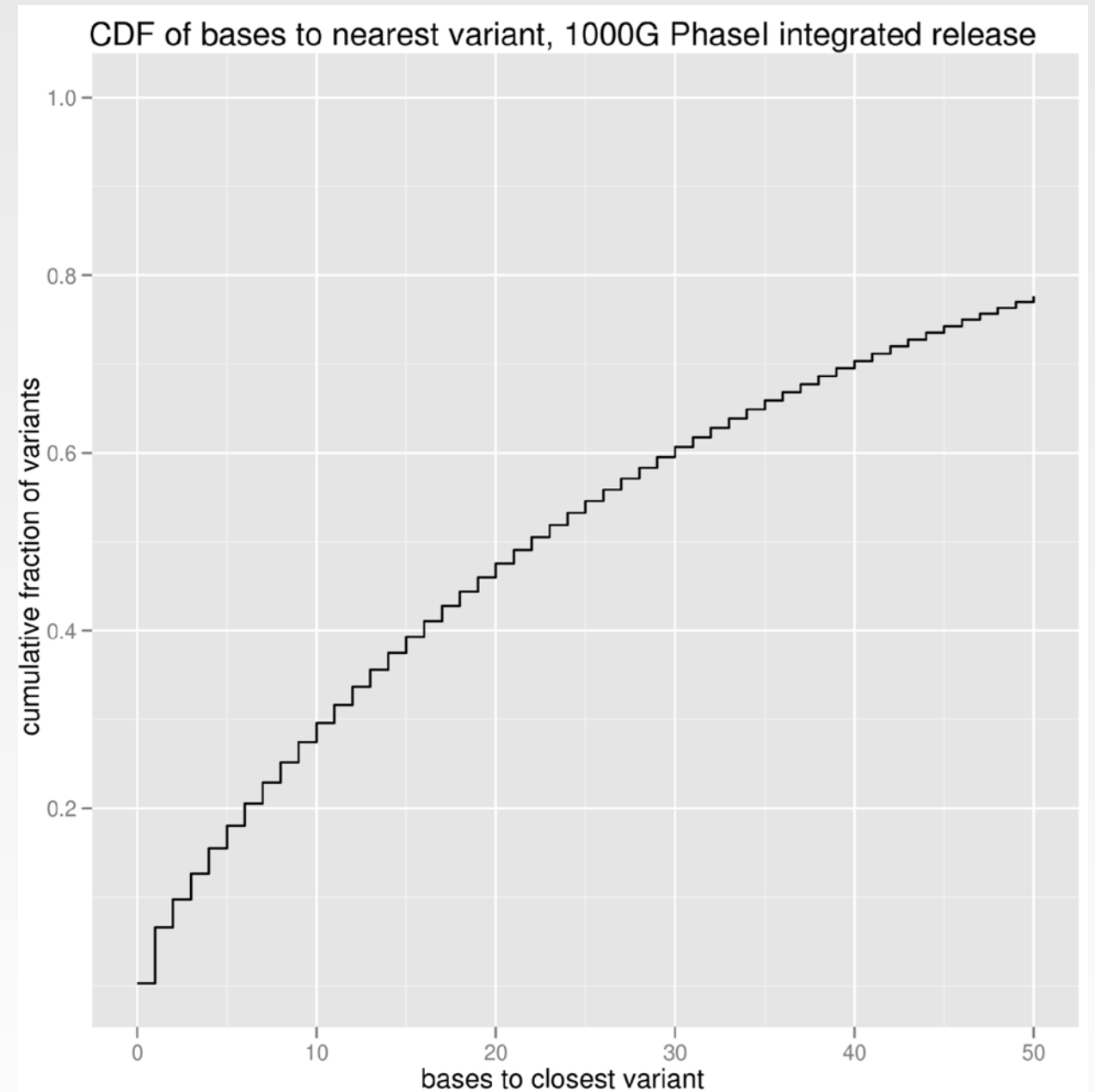


Scissors - unpublished

# Clusters of variants

What happens if multiple variants are in close proximity?

Are we leveraging all of our current knowledge when mapping?

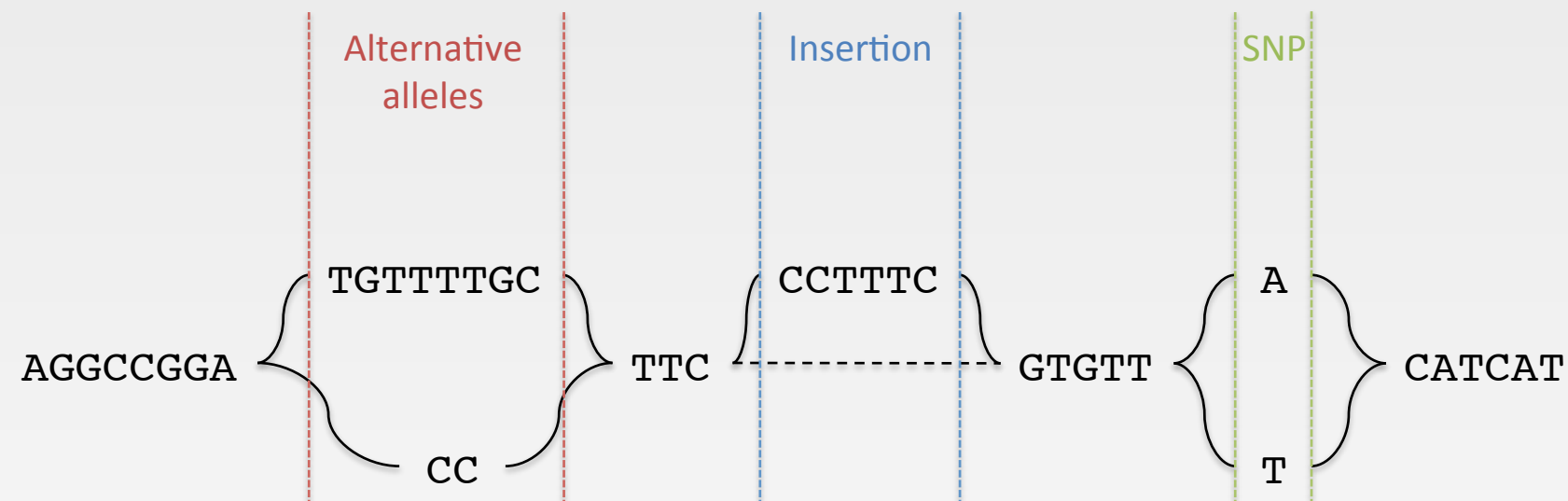


Erik Garrison - 1000 Genomes Phase I



# Variant graph

Replace linear reference with a graph

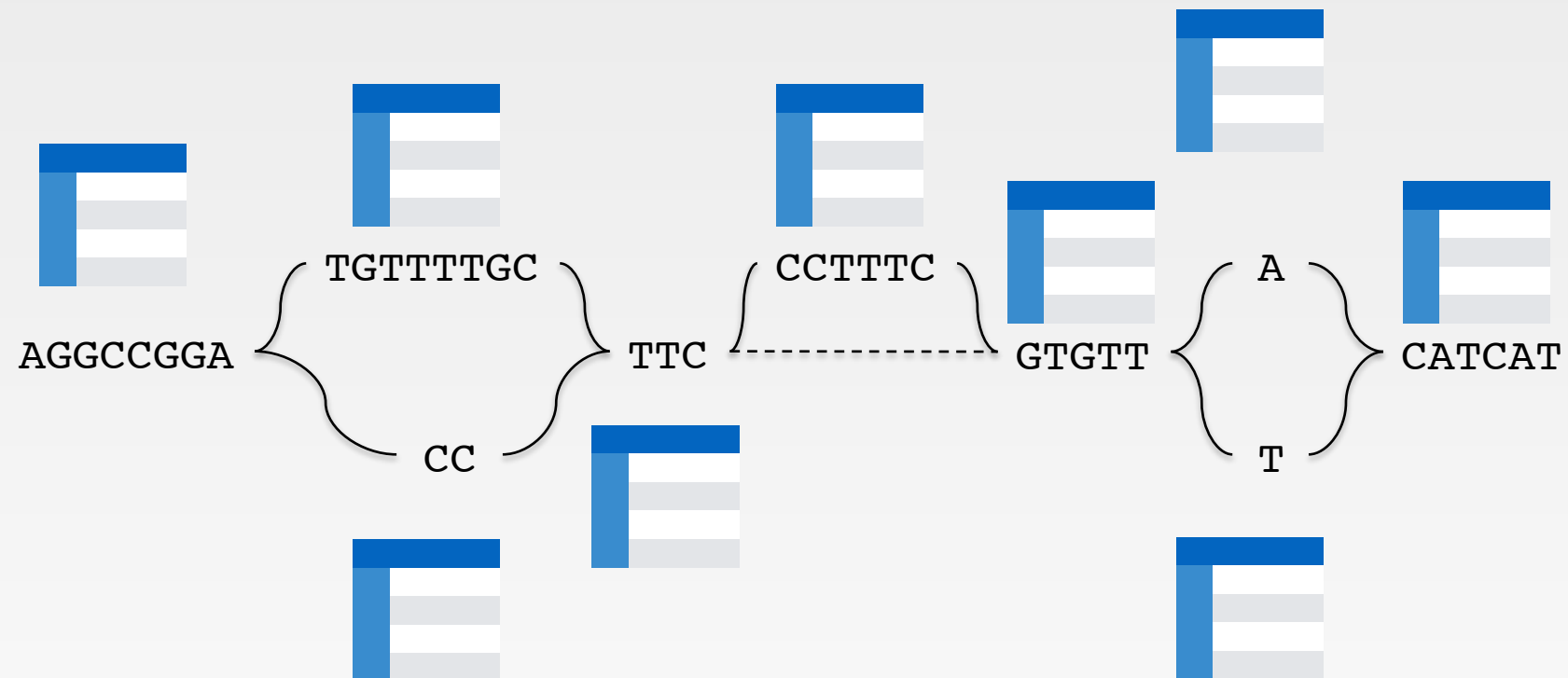


- Identify read pairs with a single uniquely mapped mate
- Generate a graph in the local region
- Map the unmapped or poorly mapped mate (lots of clipped bases/mismatches/gaps) to the graph



# Variant graph

Sample read: AGGC**T**GGACCTTCGTGTTTCATCAT  
AGGC**T**GGACCTTCGTGTTTCATCAT

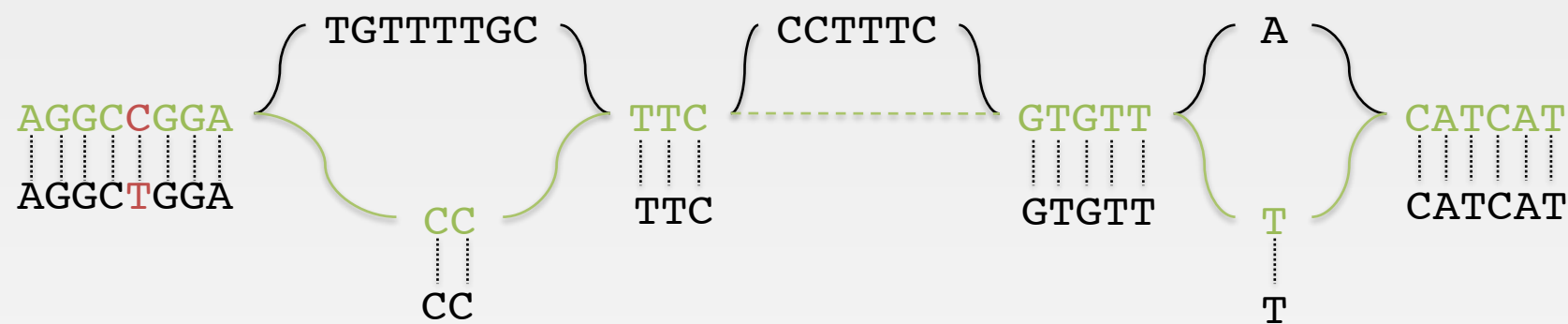


Compare read to all possible paths using a graph Smith-Waterman approach



# Variant graph

Sample read: AGGC**T**GGACCTTCGTGTTTCATCAT



By following the correct alleles, there is only a single mismatch in the alignment

- Significant reduction in reference bias
- Recover de novo variants clustered with other polymorphic sites

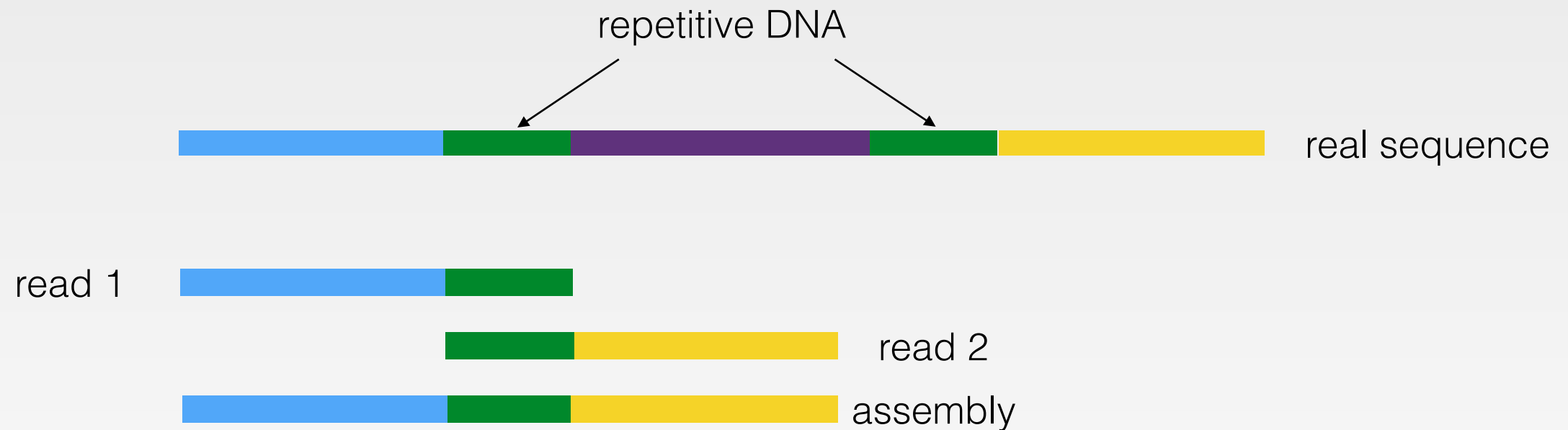


# Unresolved problems

- Mapping tries to match the reference, so inherently introduces a bias towards the reference
- We have to modify parameters based on the read content (e.g. deletions)
- Mapping to repetitive DNA is still problematic
- What if there is no or an incomplete reference for the sequenced organism?

# Assembly

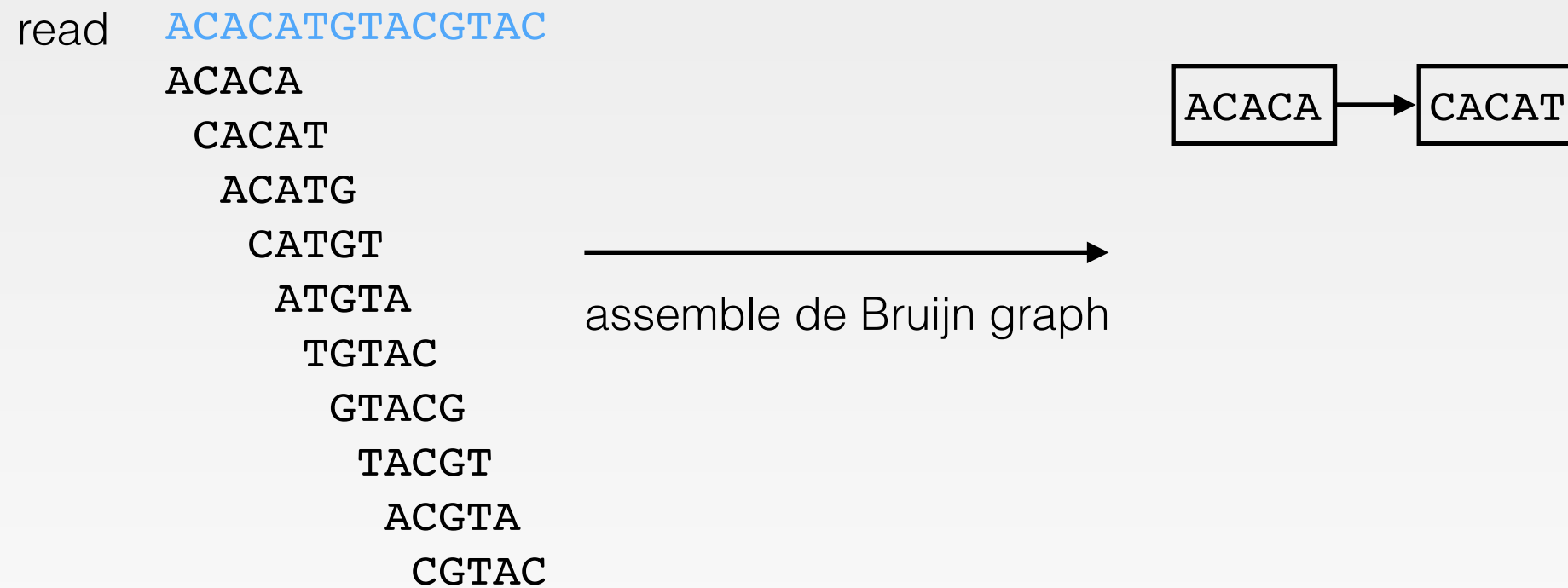
Can we just overlap the reads to create an assembly?



We can, if there isn't too much repetitive DNA  
BUT, >50% is repetitive

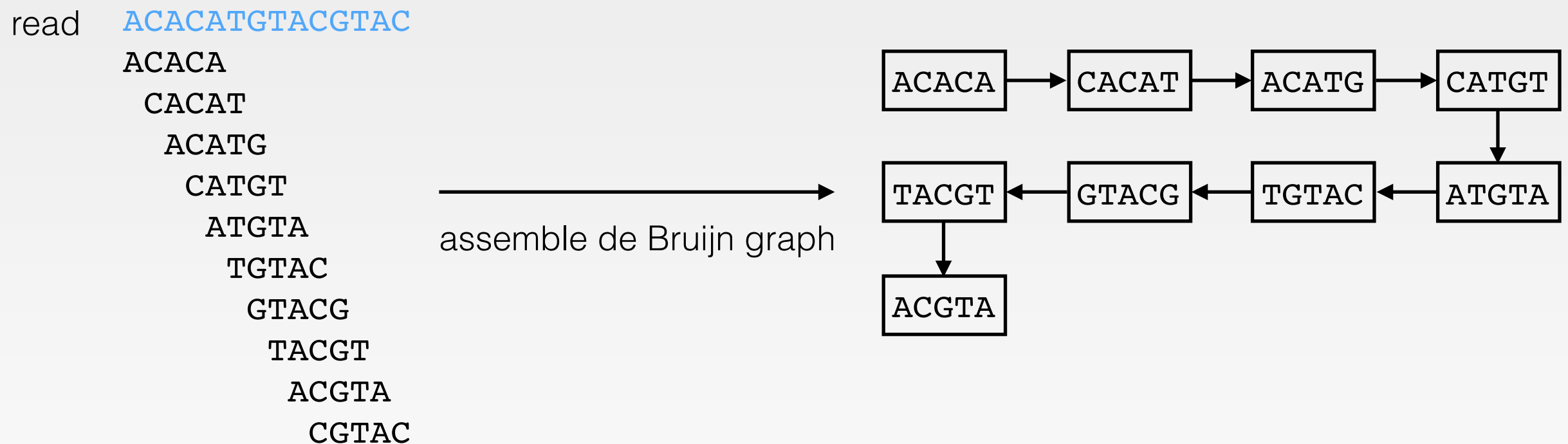
# Clean de Bruijn graph

Break reads into k-mers (of length e.g. 5)  
Each k-mer is a node in the graph



# Clean de Bruijn graph

Break reads into k-mers (of length 5)  
Each k-mer is a node in the graph



This is the de Bruijn graph representation of the read

# De Bruijn graph

Let's add one more base ('G') to the read

read ACACATGTACGTACG

ACACA

CACAT

ACATG

CATGT

ATGTA

TGTAC

GTACG

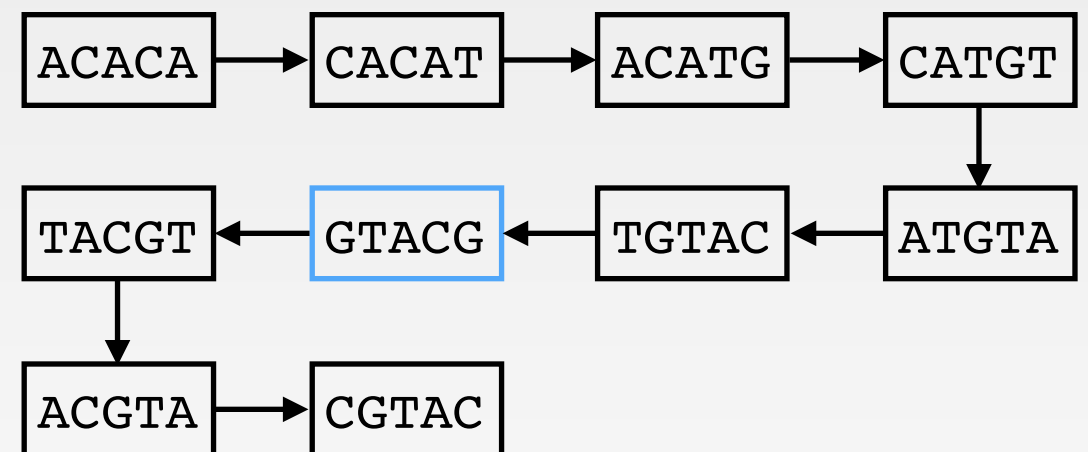
TACGT

ACGTA

CGTAC

GTACG

assemble de Bruijn graph



Final node

GTACG



# De Bruijn graph

Let's add one more base ('G') to the read

read ACACATGTACGTACG

ACACA

CACAT

ACATG

CATGT

ATGTA

TGTAC

GTACG

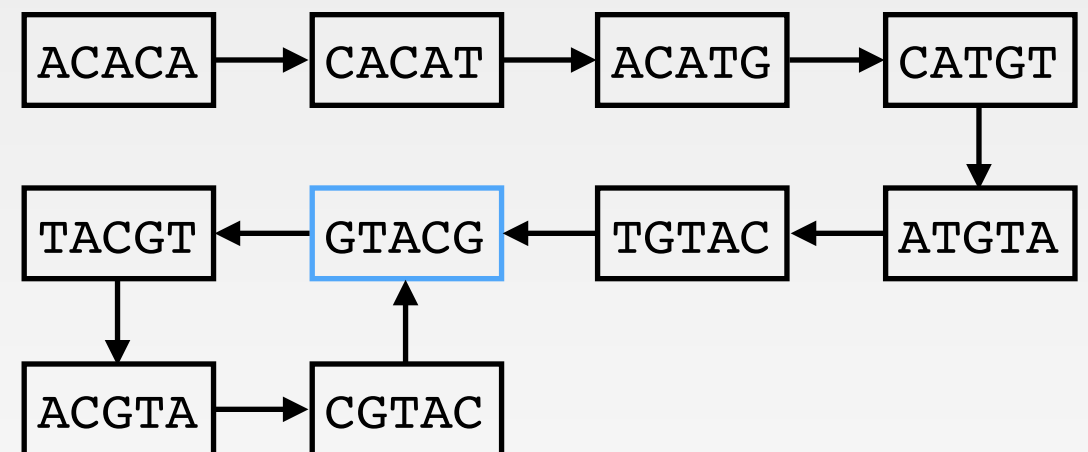
TACGT

ACGTA

CGTAC

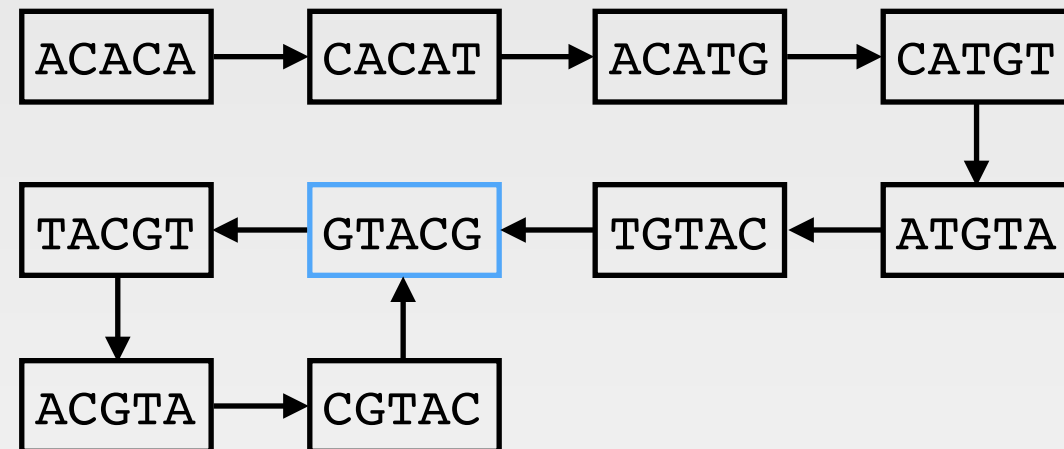
GTACG

assemble de Bruijn graph

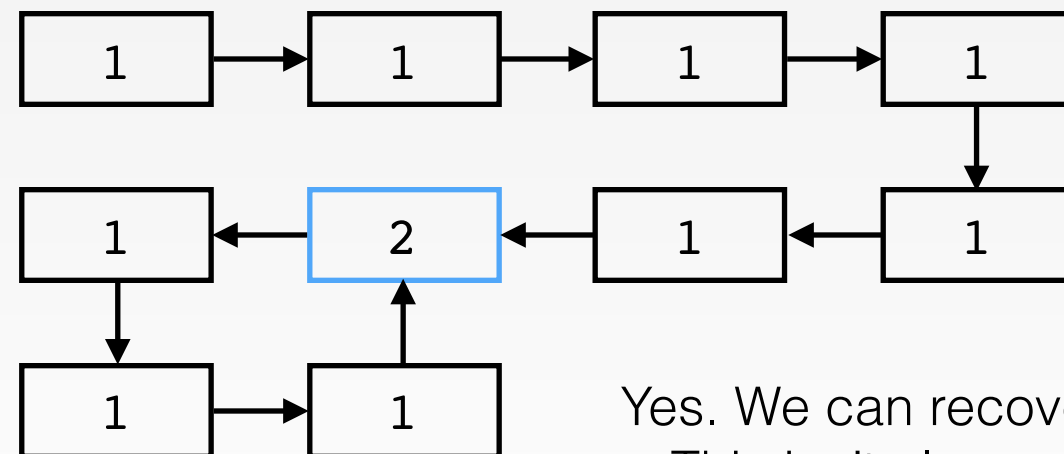


Our graph has a loop!  
Can we retrieve our read from the graph?

# Graph back to read



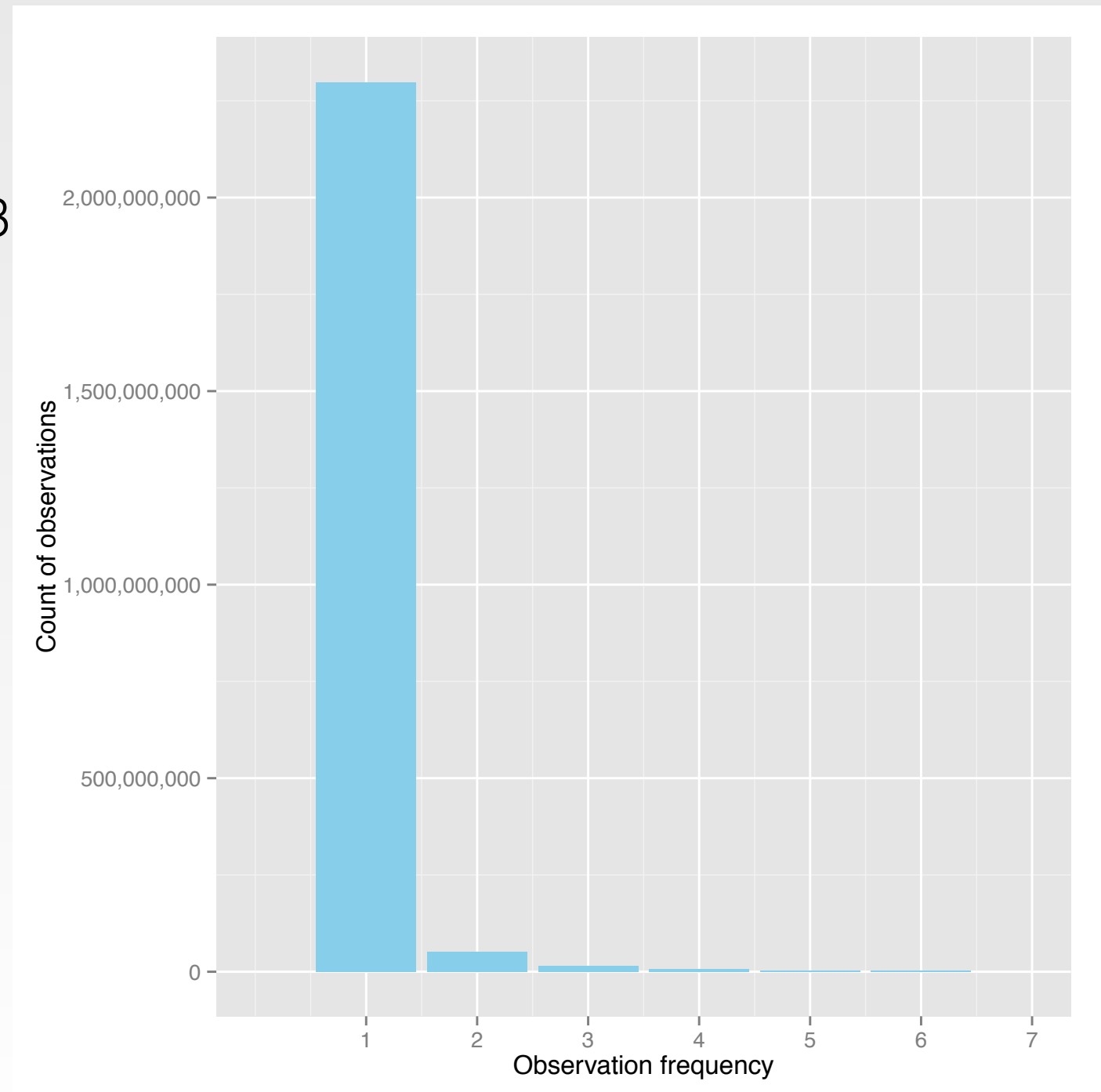
Record k-mer frequencies as graph is built



Yes. We can recover the read.  
This isn't always possible.  
(Sanger sequencing)

# Distribution of k-mers

- Consider using a k-mer length of 23
- There are  $4^{23} = 7 \times 10^{13}$  possible k-mers of length 23,  
(70,000,000,000,000 k-mers)
- The human genome only has  $3 \times 10^9$ bp
- Most k-mers are unique

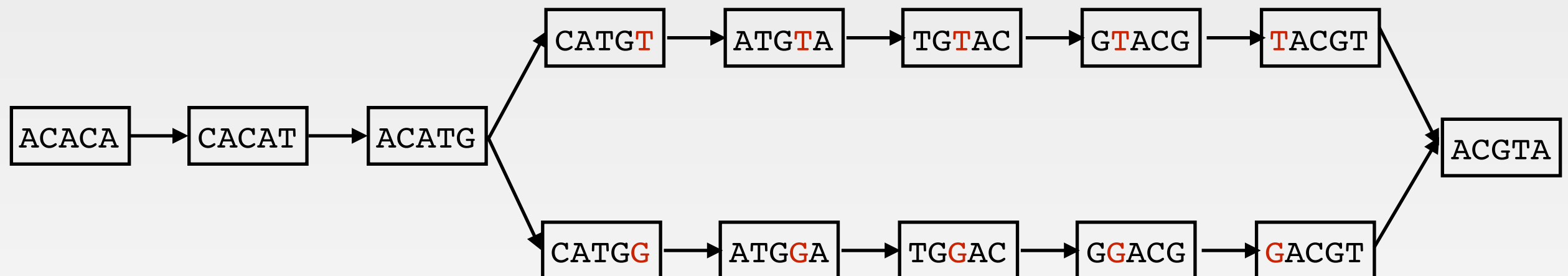


# Bubbles

Sample has a heterozygous SNP

ACACATG**T**ACGTAC

ACACATG**G**ACGTAC

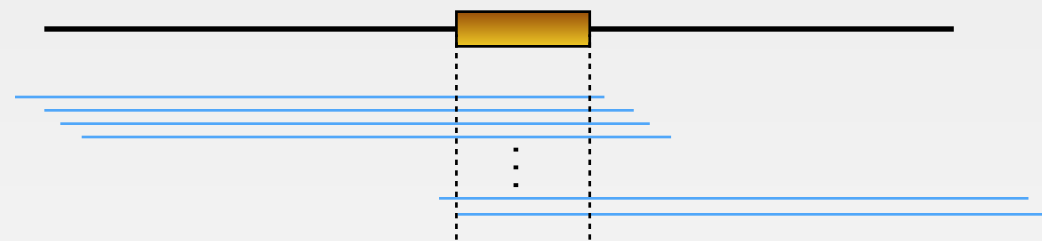


Is this a SNP or an error?

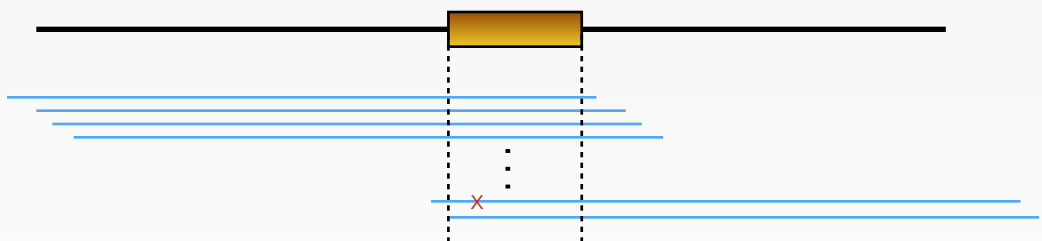
# k-mer frequency distribution

Errors

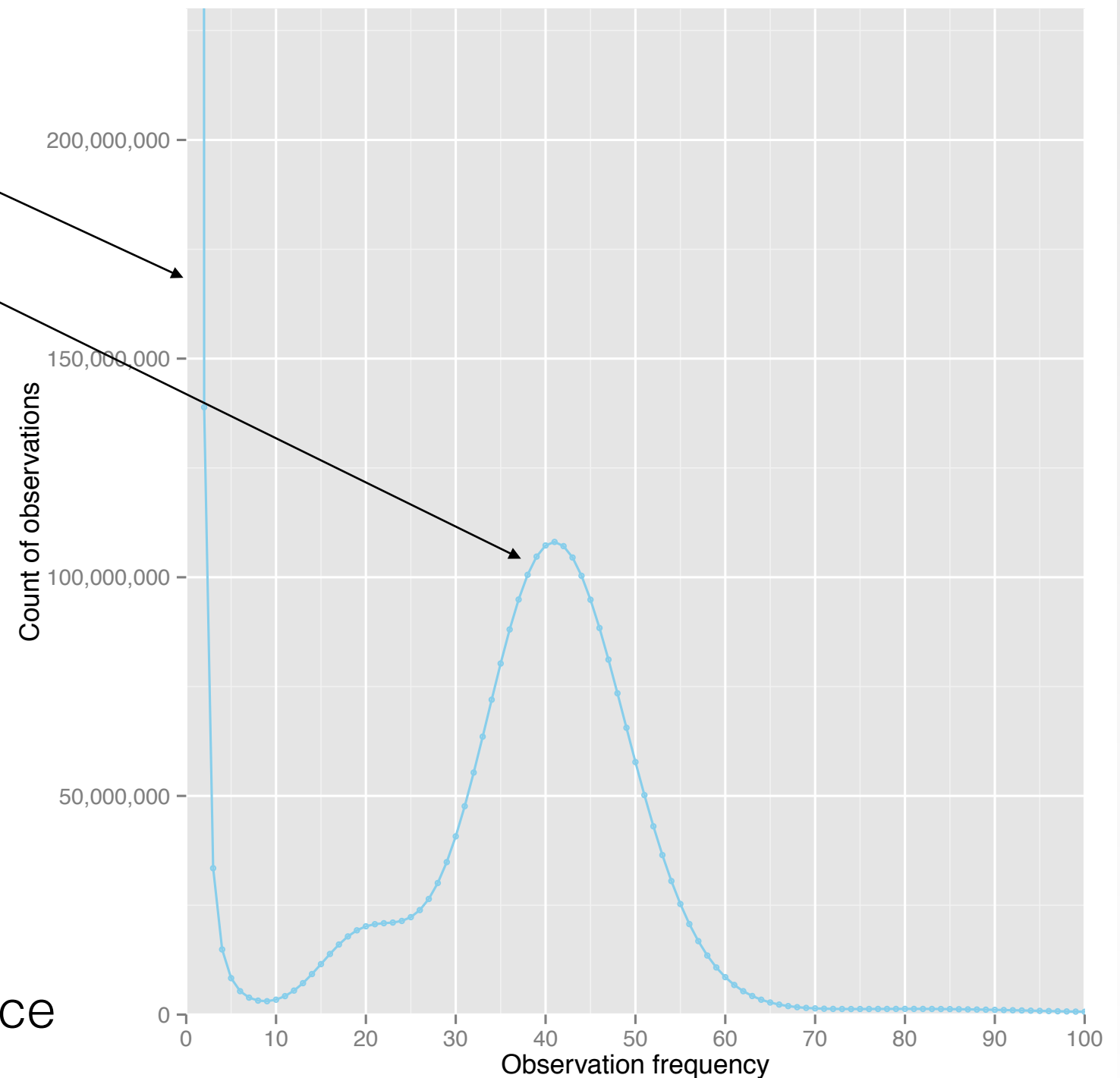
Peak at sequencing coverage



If the coverage is 40x, we observe the k-mer 40 times



k-mer with error, we only observe once



# Summary

- Many mapping strategies
  - Hash based mapping
  - Burrows-Wheeler transform
  - Split read mapping
  - Local graph alignment
- Overlap assembly
- de Bruijn graph assembly
- Reference free k-mer comparison
- Choose a strategy (or combination of strategies) based on the experiment and the available data

# Mapping tools

## Mappers:

Mosaik: <https://github.com/wanpinglee/MOSAIK>

\*BWA: <http://bio-bwa.sourceforge.net/>

STAMPY: <http://www.well.ox.ac.uk/project-stampy>

## Split-read aligners:

SCISSORS: <https://github.com/wanpinglee/scissors>

Pindel<sup>†</sup>: <http://gmt.genome.wustl.edu/pindel/current/>

Graph alignment: \*glia: <https://github.com/ekg/glia>

Reference free: \*RUFUS: <https://github.com/jandrewfarrel/RUFUS>

\*gkno: [https://github.com/gkno/gkno\\_launcher](https://github.com/gkno/gkno_launcher)

\*gotCloud: <http://genome.sph.umich.edu/wiki/GotCloud>