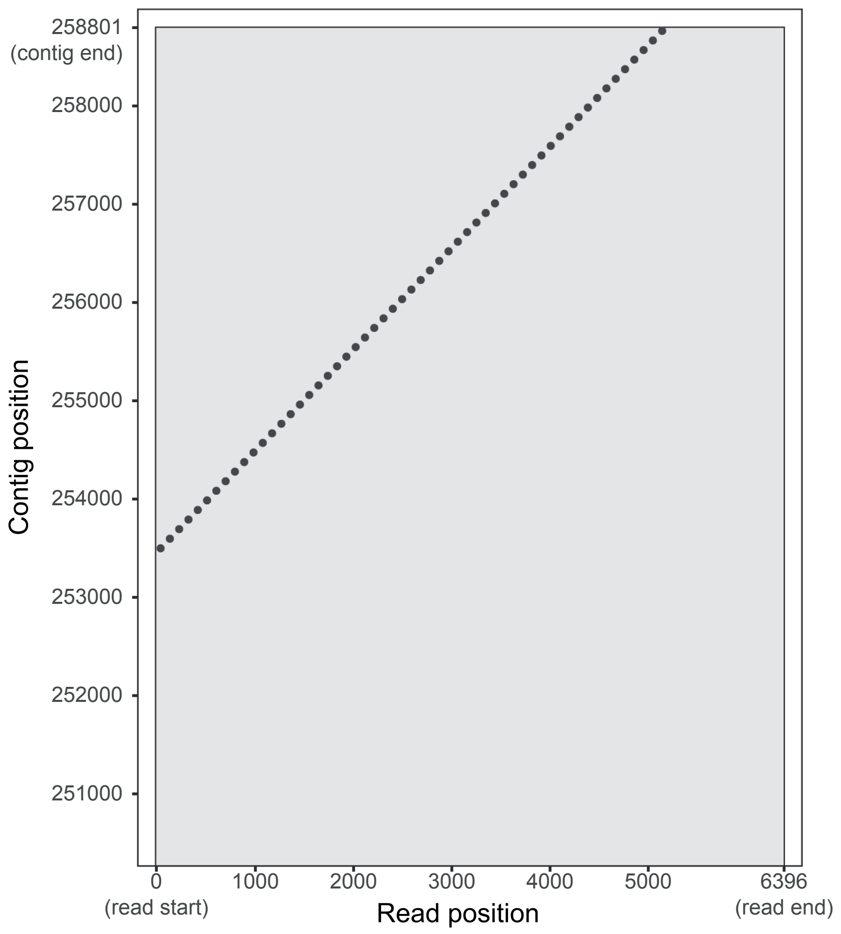
Semi-global read alignment in Unicycler-align

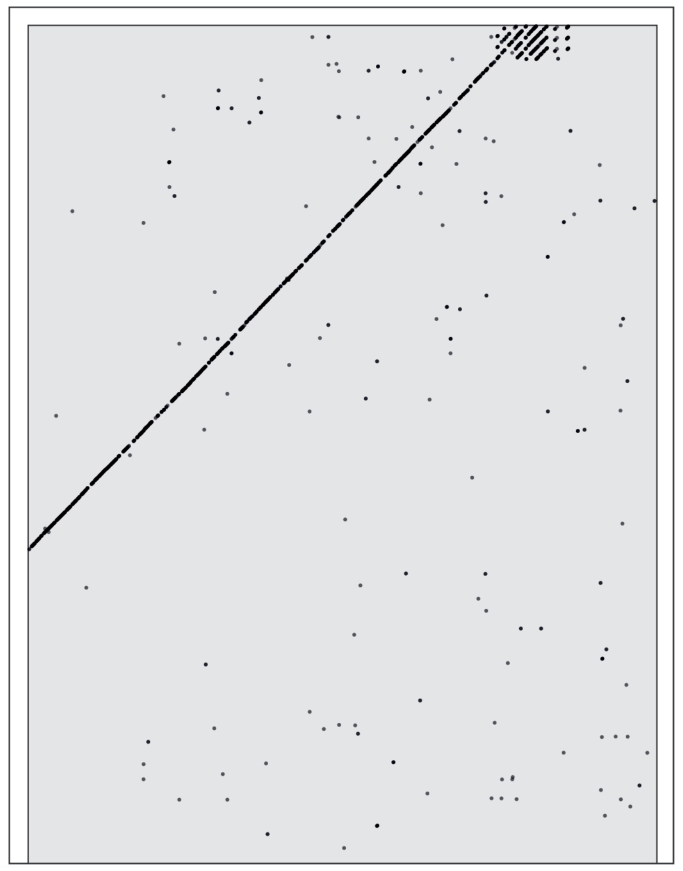
A. Minimap alignment



Minimap efficiently finds approximate alignments between contigs and reads. These alignments are local (not semi-global) and only provide start/end coordinates (not a base-by-base alignment). Unicycler-align uses this alignment to identify relevant regions of the read and contig for use in subsequent steps.

This example shows a read which aligns to a long contig, overlapping past the contig end.

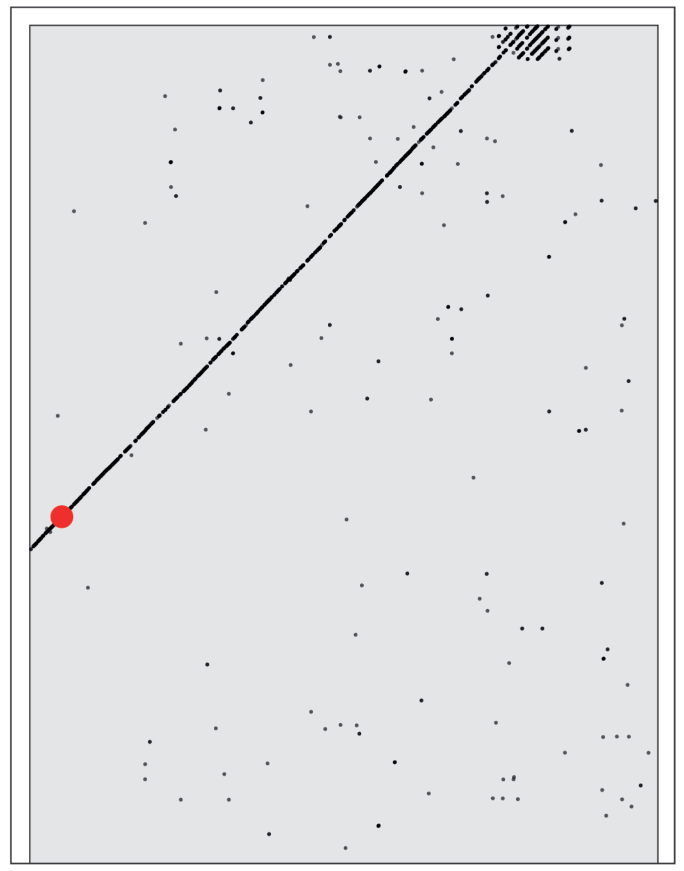
B. Common k-mers



A k-mer index (implemented as a hash table) identifies all common k-mers between the read and contig. Unicycler-align's default k-mer size is 10, a relatively small value which allows for alignment of low identity reads.

In this example, the contig is adjacent to a loop in the graph. The contig contains repetitive sequence at its end, and the read extends past the end of the contig into more of this repetitive sequence. This is shown in the dot plot by the dense region of common k-mers at the top right.

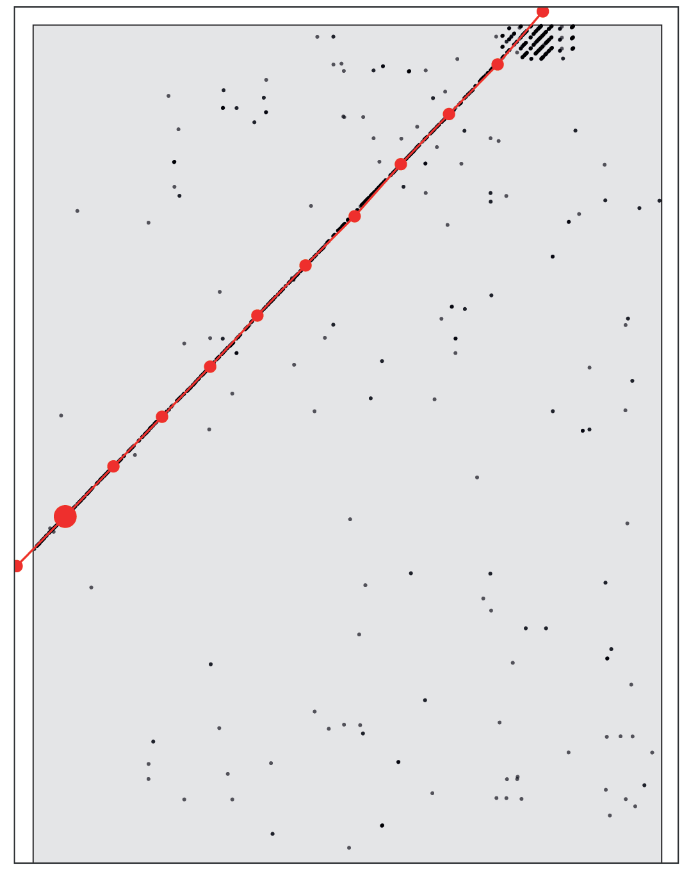
C. Alignment seed



Unicycler-align scores each common k-mer based on its neighbouring points. Nearby points that lie on or near the diagonal to the point contribute positively to the score, while nearby points away from the diagonal detract from the score.

This selects a point that is in a well matching region away from repeats – ideally one that is unambiguously on the alignment line.

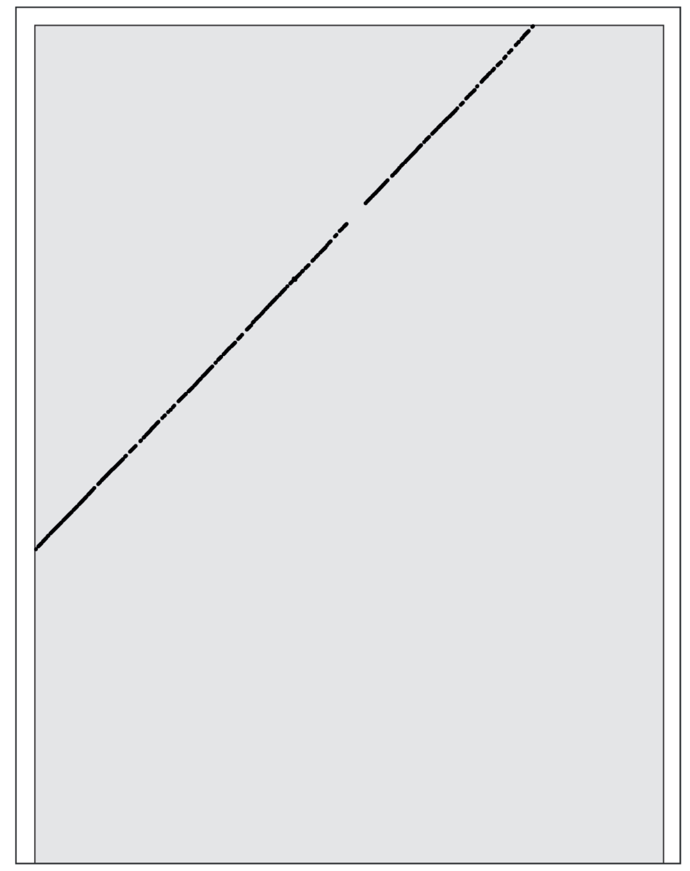
D. Line tracing



Starting with this point, Unicycler-align then traces a line outward in each direction. The line is made up of discreet segments, and the angle of each is selected with a hill climbing algorithm to maximise the proximity of the line to common k-mers.

As Unicycler-align performs semi-global alignment, the line tracing proceeds until a sequence end is reached, even in regions where the alignment is weak.

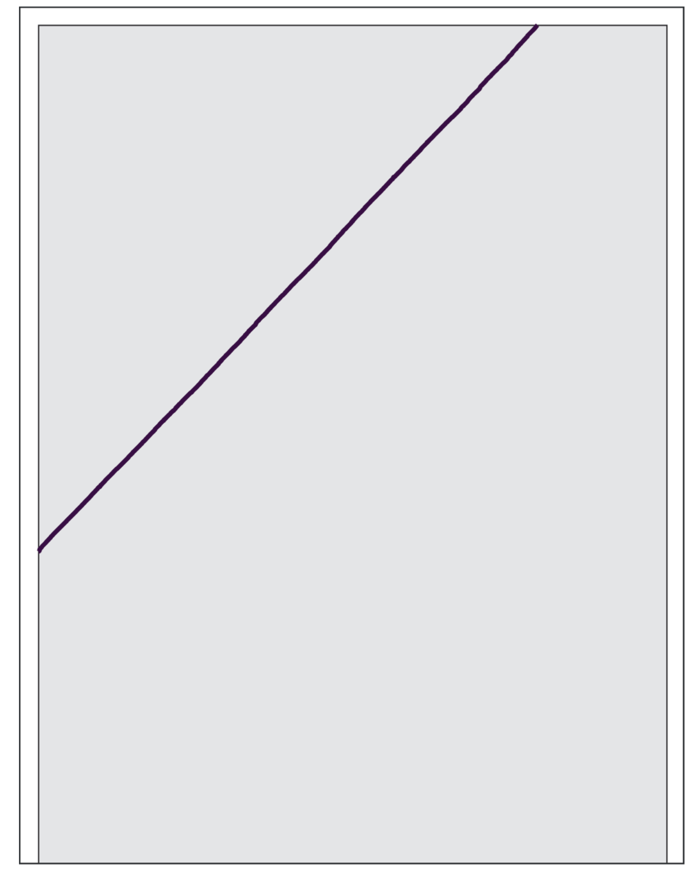
E. K-mer selection



Common k-mers near the alignment line are extracted and all others are discarded.

Ideally, this will result in a dense collection of k-mers over the entire alignment. However, some regions can be sparse due to poor read identity or inexact line tracing. In this example, a deletion-heavy region in the middle of the read caused the alignment to deviate from a straight line. As a result, the line trace briefly diverged from the true alignment, leaving a small gap in the selected k-mers.

F. SeqAn alignment



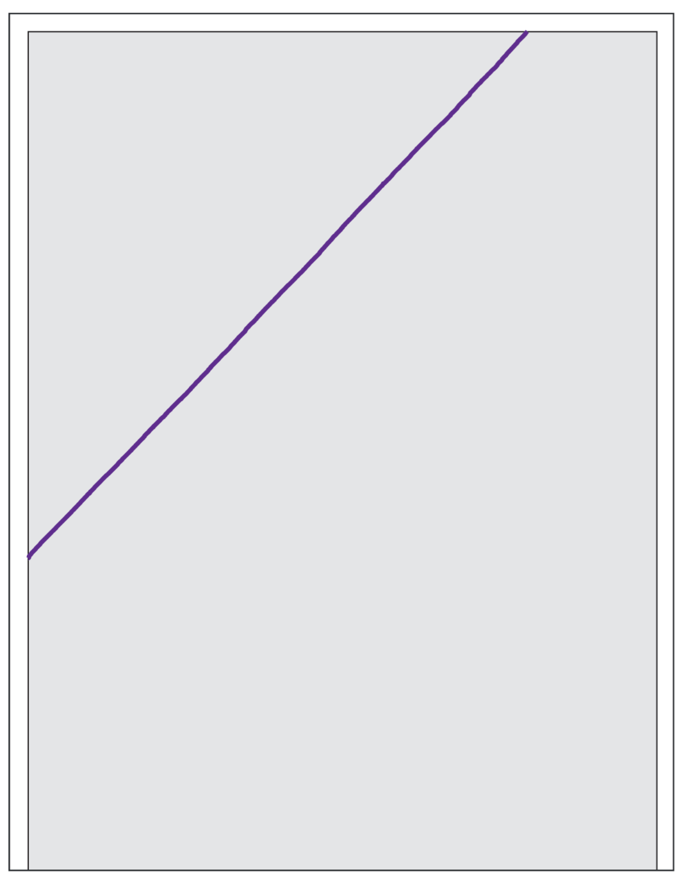
The selected common k-mers are then used as seeds in a banded chain alignment in SeqAn. This process performs dynamic programming alignment in a band around the common k-mers.

This algorithm spans gaps in k-mer seeds, and is therefore not hindered by regions lacking common k-mers (such as the gap from the previous step).

The final alignment is semi-global and extends until one of the sequences ends. Therefore, when the read overlaps the end of a contig (as is the case in this example), the alignment indicates the exact read position at the point of overlap.

Comparison of long-read alignment tools

Unicycler-align



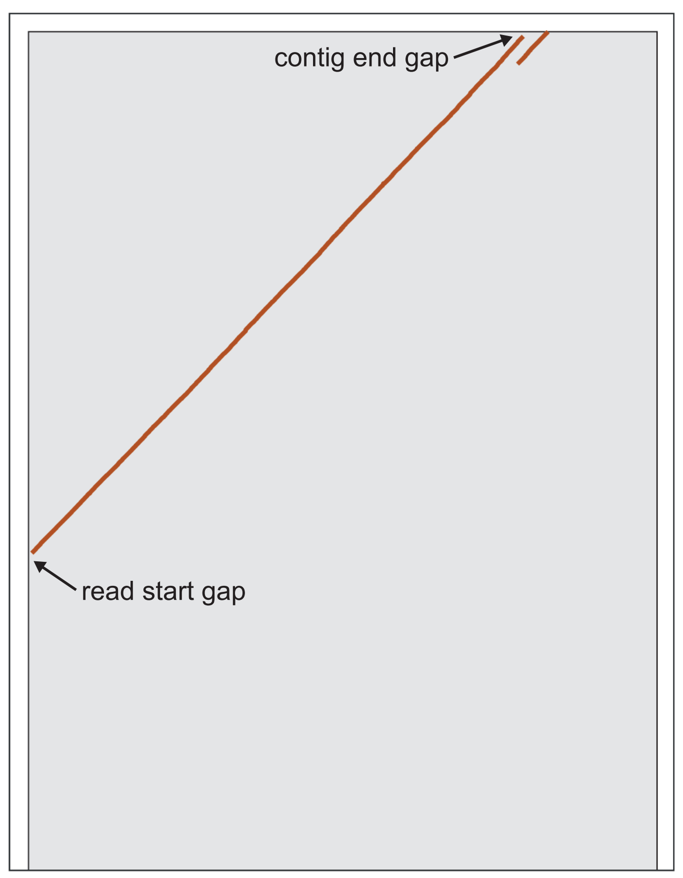
Command:

```
unicycler_align --ref contig.fasta
  --reads read.fastq --sam out.sam
```

End gaps:
- Read start gap: 0 bp
- Contig end gap: 0 bp

Unicycler-align finds a single alignment between the read and contig, ignoring smaller alignments in the repetitive region. Its alignment is semi-global, leaving no end gaps.

BWA-MEM (v0.7.15-r1140)



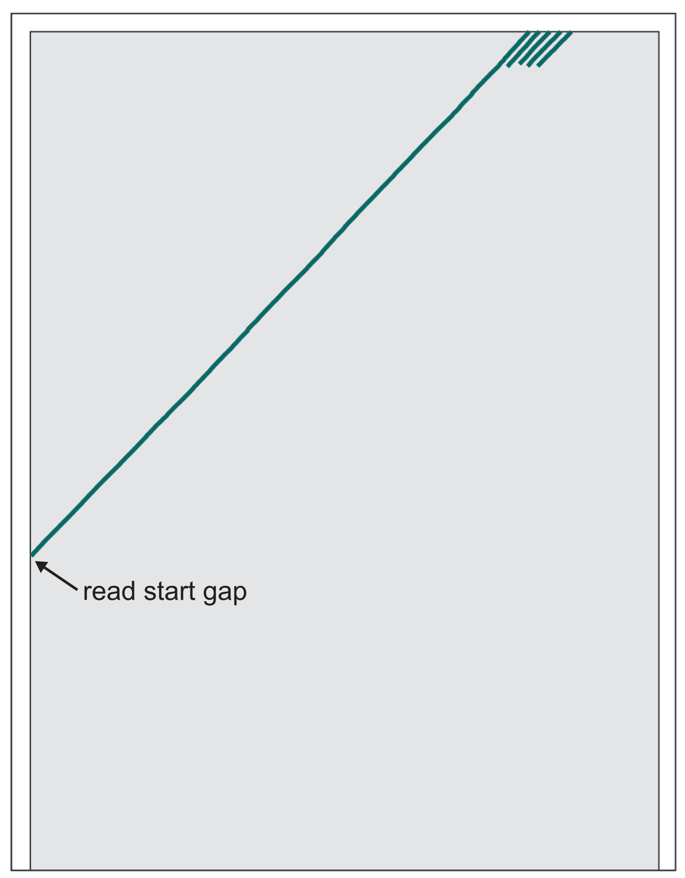
Commands:

```
bwa index contig.fasta
bwa mem contig.fasta read.fastq
```

End gaps (longest alignment):
- Read start gap: 35 bp
- Contig end gap: 49 bp

BWA-MEM finds the same alignment line as Unicycler-align, though there are small gaps on both ends. It also reports a second alignment in the repetitive region. While this alignment is genuine, it is spurious when looking for the read's single best alignment to the contig.

BLAST (megablast, v2.6.0+)



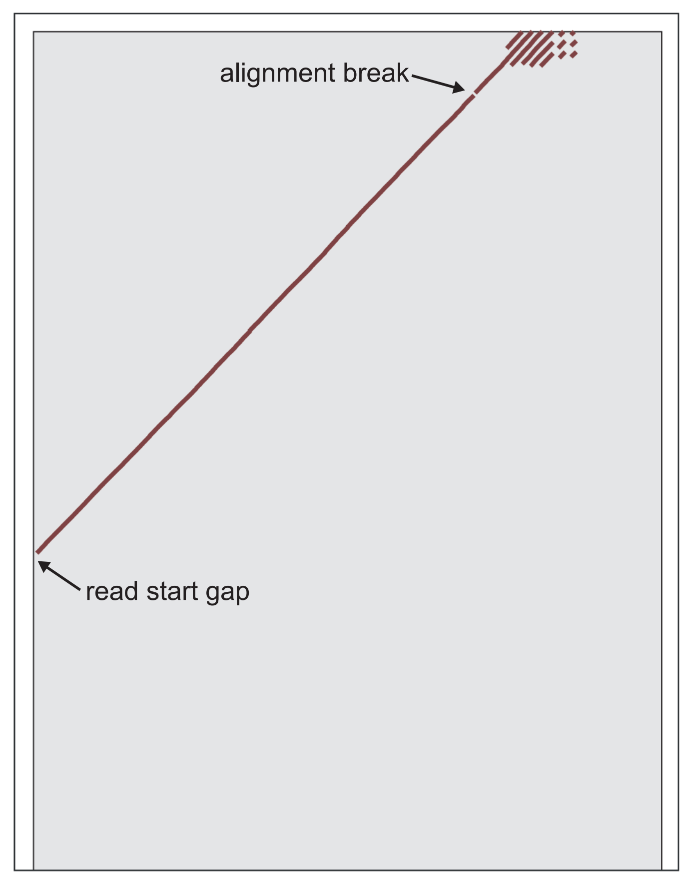
Commands:

```
makeblastdb -dbtype nucl -in contig.fasta
blastn -db contig.fasta -query read.fasta
```

End gaps (longest alignment):
- Read start gap: 10 bp
- Contig end gap: 0 bp

BLAST extends the primary alignment line further than BWA-MEM, leaving only a small gap at the start of the read. It also reports additional alignments in the repetitive region.

LAST (v843)



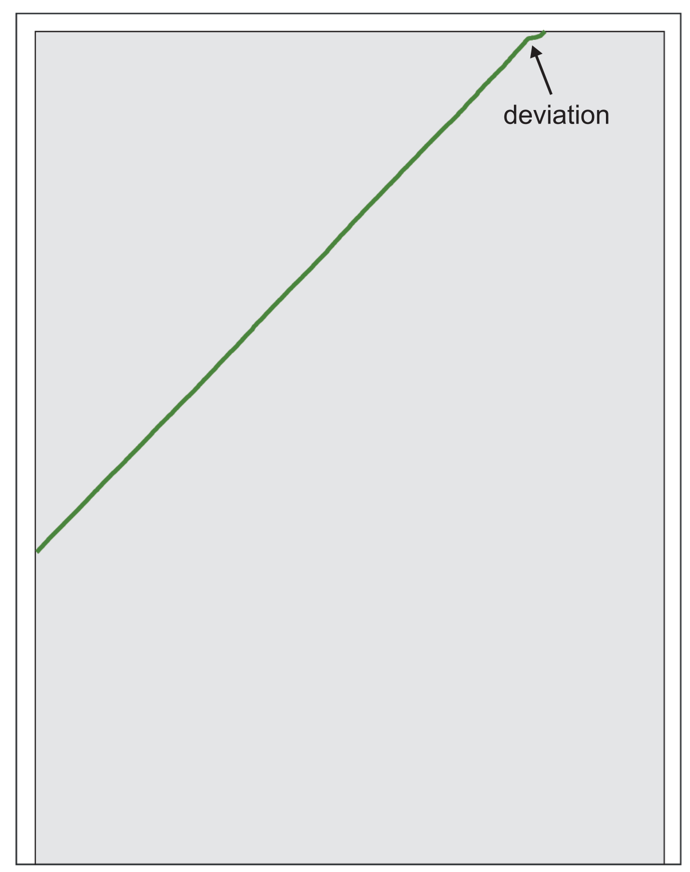
Commands:

```
lastdb -R01 contig_db contig.fasta
lastal -Q1 contig_db read.fastq
```

End gaps (longest alignment):
- Read start gap: 35 bp
- Contig end gap: 650 bp

LAST follows the main alignment line, though it is split into two parts at read position 4490. This results in a large end gap for the main alignment. LAST also reports many smaller alignments in the repetitive region.

BLASR (v5.3.f8bfa9c)



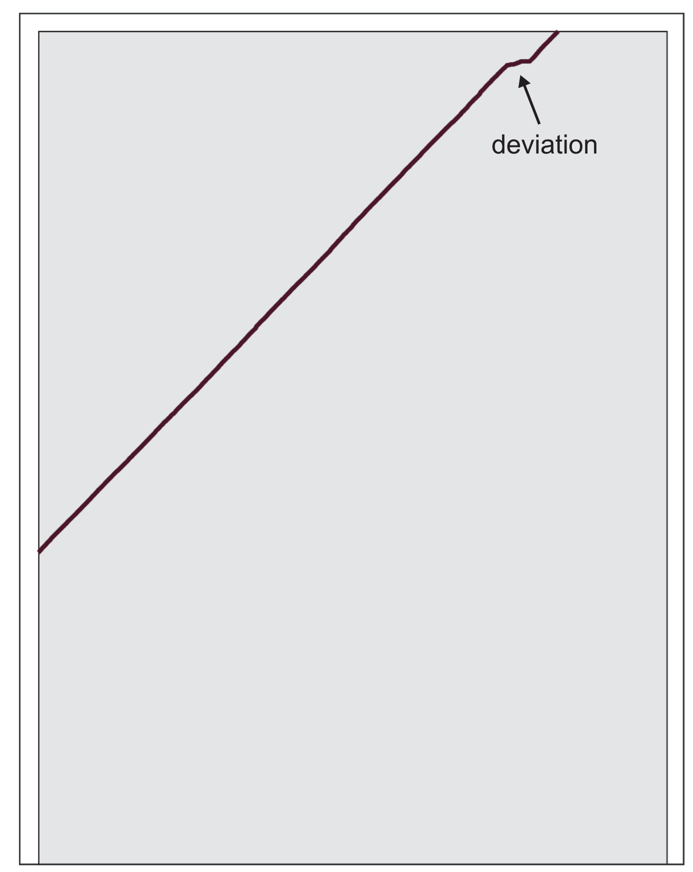
Command:

```
blasr read.fasta contig.fasta
```

End gaps:
- Read start gap: 10 bp
- Contig end gap: 0 bp

BLASR finds a single alignment without an end gap, but the repetitive region causes a major deviation from the true alignment line. This results in an incorrect read position at the point of overlap.

GraphMap (v0.5.1)



Command:

```
graphmap align -d read.fasta
  -r contig.fasta -o out.sam
```

End gaps:
- Read start gap: 0 bp
- Contig end gap: 0 bp

GraphMap performs true semi-global alignment, leaving no end gaps, but its alignment deviates in the repetitive region. Like BLASR, it gives an incorrect read position at the point of overlap