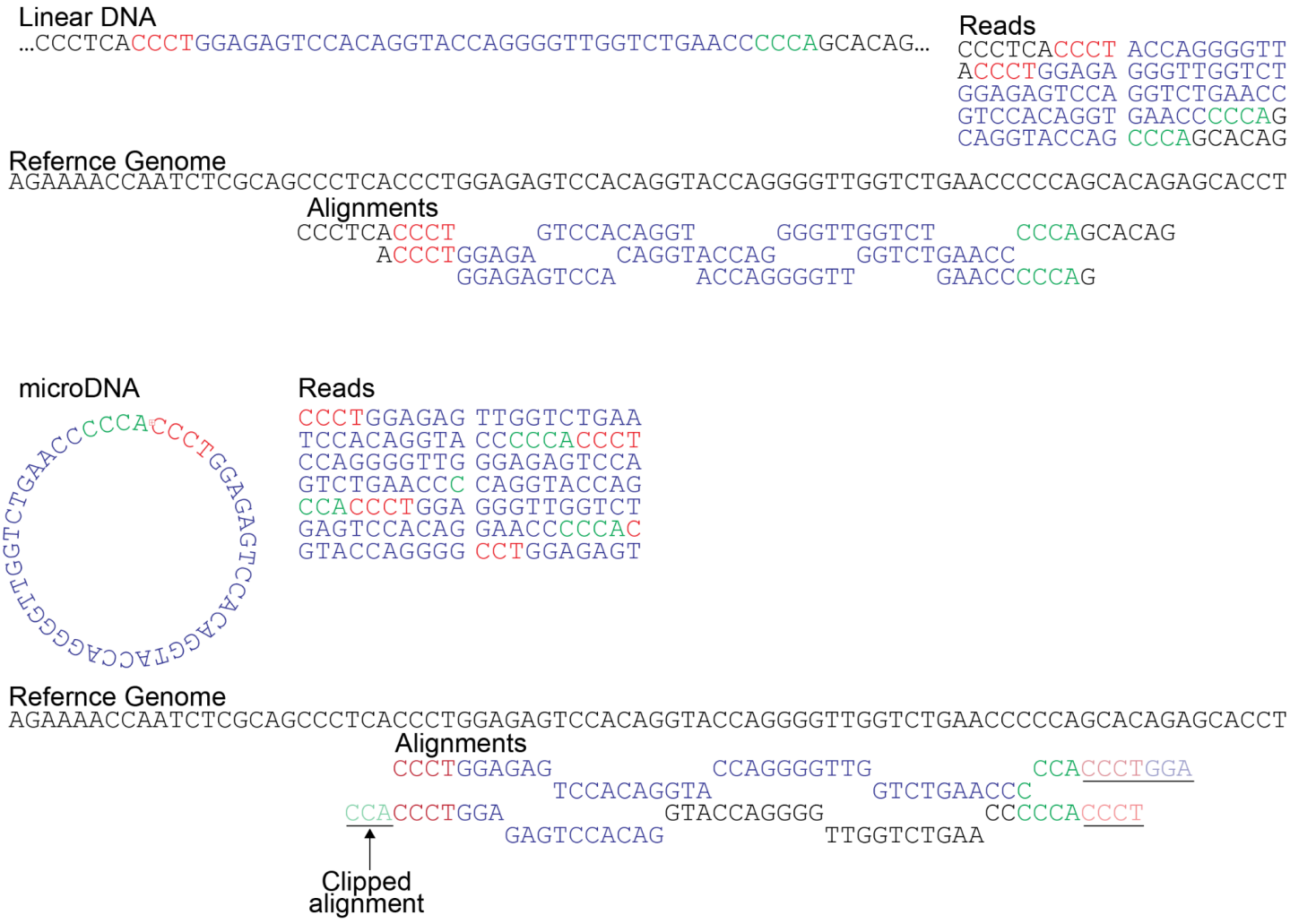
Project: Discover and Characterize MicroDNA in a Cancer Cell Line
Computational Genomics

Objectives: Apply your computational genomics knowledge to solve a real-world problem.

Due Date: May 7, 11:59PM

Introduction: MicroDNA are small, circular, non-coding DNA molecules that were first described in 2012 (https://www.science.org/doi/10.1126/science.1213307), and their role remains not fully understood. It is currently believed to influence cellular homeostasis by binding to transcription factors and has been used as a biomarker for cancer. The detection of microDNA from sequencing data is essential for understanding their role in cancer biology.

What MicroDNA looks like: When standard linear DNA is sequenced and those reads are aligned to the genome, all of the reads fully align to the genome and are uniformly distributed. When we sequence a molecule of circularized linear DNA, some of the reads will span the circle *junction*, or the point where the two ends of the linear DNA joined. When these reads are aligned to the genome, only a portion will match and the remaining sequence will be *clipped*.



The standard format for read alignments is SAM (https://samtools.github.io/hts-specs/SAMv1.pdf). Which has a number of fields, including read name, the chromosome, and start position of the alignment, and crucially for our purposes the CIGAR (Concise Idiosyncratic Gapped Alignment Report) field. The CIGAR field is a string that describes how a read is aligned to a reference genome, and consists of operations that indicate the length of matches, insertions, deletions, and other alignment features. The operations we are most indeed in include:

- M - Match or mismatch (aligned to the reference)
- S - Soft clipping (clipped sequence not included in alignment)

If our reads are 42bp long, a fully aligned read will look like this:

Chromosome	Start	Score	CIGAR	Sequence
NC_000001.10	2383746	60	**42M**	CCTGCCTGGCAGGTAGCAGCCCCGTGGAAGTATTTCATCTTG

And reads that are clipped will look like this (some SAM fields are omitted for brevity):

NC_000001.10	2383556	60	**12S30M**	GCCTGGCAGGTAGCAGCCCCCACTGCATTGCTGAGCCTGGAA
NC_000001.10	2383739	60	**30M12S**	CCCCGGCCCTGCCTGGCAGGTAGCAGCCCCCACTGCATTGCT

The CIGAR string (in bold) of the first read tells us that all 42 bases are aligned. The next two indicate that the first 12 bases of the first read (in blue) and the final 12 bases of the second (in red) did not align. If we compare these reads to the reference they would be

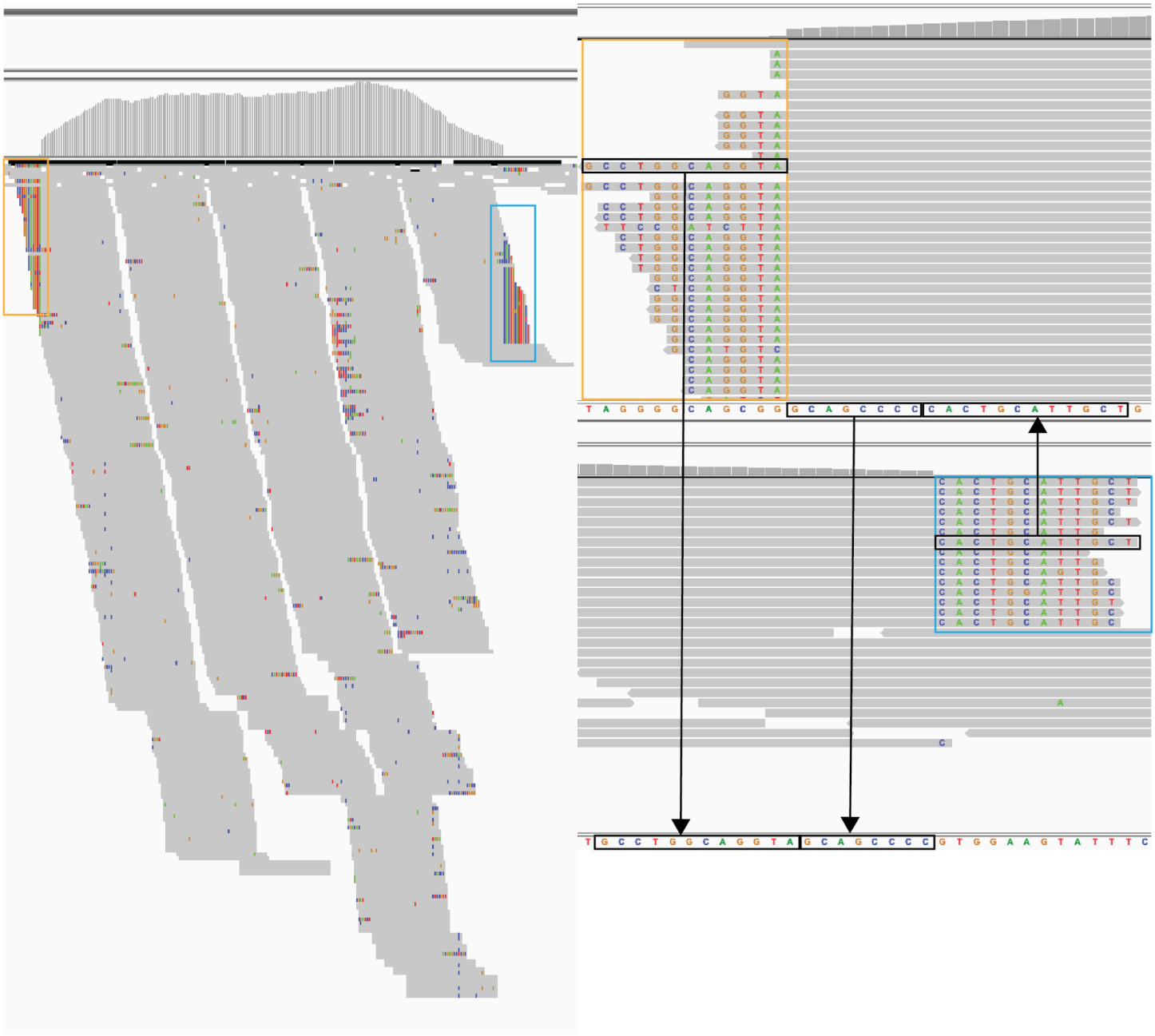
```
Ref:  TAGGGGCAGCGGGCAGCCCCCACTGCATTGCTGAGCCTGGAA
      xxxx|||||xxx||||||||||||||||||||||||||||
Read: GCCTGGCAGGTAGCAGCCCCCACTGCATTGCTGAGCCTGGAA

Ref:  CCCCGGCCCTGCCTGGCAGGTAGCAGCCCCGTGGAAGTATTT
      |||||||||||||||||||||||||||||xxxxxxx|xxx|
Read: CCCCGGCCCTGCCTGGCAGGTAGCAGCCCCCACTGCATTGCT
```

These two clipped reads originated from the junction of the circle, which we can visualize by aligning the two reads. The unaligned bases in the first read (which we will call the *starting junction tag)* (blue), match aligned bases in the second, and unaligned reads in the second (the *ending junction tag*) (red) match aligned bases in the first. Interestingly, it appears that circles contain some homology (in bold), which may indicate the mechanism by which the circles are created.

```
          GCCTGGCAGGTAGCAGCCCCCACTGCATTGCTGAGCCTGGAA
          ||||||||||||||||||||||||||||||||
CCCCGGCCCTGCCTGGCAGGTAGCAGCCCCCACTGCATTGCT
```

Using IGV (https://igv.org/), we can visualize all of the evidence for a circle:



Assignment: Develop an algorithm to detect and quantify microDNAs from alignment data. In addition to your code (in a GitHub repo that follows the standard for course homeworks), write a short (~2 pages) report that details your algorithm and results. Your report should include how you validated a subset of your circles, and a link to your GitHub repository.

Your algorithm should take a as in put a BAM file (provided). Reading BAM files in Python is best accomplished with the pysam library (https://github.com/pysam-developers/pysam) which has extensive documentation (https://pysam.readthedocs.io/en/latest/api.html). For example, you can find alignments with softclips in the CIGAR string with:

```
import pysam

# Path to your BAM file
bam_file = "your_file.bam"

# Open the BAM file
with pysam.AlignmentFile(bam_file, "rb") as bam:
    for read in bam:
        # Check if the read has soft clips in the CIGAR string
        if any(cigar_op[0] == 4 for cigar_op in read.cigartuples):
            print(f"Read {read.query_name} has soft clips: {read.cigarstring}")
```

Your algorithm will scan the bam file for alignments that indicate the existence of a circle and aggregate the alignments that seem to be evidence for the same circle. For each suspected circle, define a metric that indicates how likely the circle is real and not an unrelated alignment artifact (e.g. how well the start and end tags agree, the number of junction tags, the total depth of circle DNA, etc), and report all circles that meet some predefined threshold. This report should give the position and score of each circle.