Computing GC Content

Jan Emmanuel Samson

2024-07-29

Problem

The GC-content of a DNA string is given by the percentage of symbols in the string that are ‘C’ or ‘G’. For example, the GC-content of “AGCTATAG” is 37.5%. Note that the reverse complement of any DNA string has the same GC-content.

DNA strings must be labeled when they are consolidated into a database. A commonly used method of string labeling is called FASTA format. In this format, the string is introduced by a line that begins with ‘>’, followed by some labeling information. Subsequent lines comntain the string itself; the first line to begin with ‘>’ indicates the label of the next string.

In Rosalind’s implementation, a string in FASTA format will be labeled by the ID “Rosalind_xxxx”, where “xxxx” denotes a four-digit code between 0000 and 9999.

- **Given**: At most 10 DNA strings in FASTA format (of length at most 1 kbo each).
- **Return**: The ID of the string having the highest GC-content, followed by the GC-content of that string. Rosalind allows for a default error of 0.001 in all decimal answers unless otherwise stated.

Sample Dataset

```
>Rosalind_6404
CCTGCGGAAGATCGGCACTAGAATAGCCAGAACCGTTTCTCTGAGGCTTCCGGCCTTCCC
TCCCACTAATAATTCTGAGG
>Rosalind_5959
CCATCGGTAGCGCATCCTTAGTCCAATTAAGTCCCTATCCAGGCGCTCCGCCGAAGGTCT
ATATCCATTTGTCAGCAGACACGC
>Rosalind_0808
CCACCCTCGTGGTATGGCTAGGCATTCAGGAACCGGAGAACGCTTCAGACCAGCCCGGAC
TGGGAACCTGCGGGCAGTAGGTGGAAT
```

Sample Output

```
Rosalind_0808
60.919540
```

Intuition

The first step is to parse the input data, storing headers as keys and sequences as values in a dictionary. The input is read line-by-line and headers are identified by searching for ‘>’ as the starting character in each line. For the sequences, a helper function is needed to calculate the GC-content. We then iterate over the items of the dictionary, computing the GC-content of each sequence and keeping track of the maximum value (along with its id). Upon finishing the iteration, we return the sequence ID with the highest GC-content, along with its sequence.

Solution

```
def compute_gc(seq: str) -> float:
    """Compute the proportion of G's + C's in a DNA string."""
    return (seq.count('G') + seq.count('C')) / len(seq) * 100

def read_fasta(filepath: str) -> dict[str, str]:
    """Parse the headers and sequences in a FASTA file."""
    id, seq = None, []
    for line in filepath:
        line = line.rstrip()
        if line[0] == ">":
            if id:
                yield (id, ''.join(seq))
            id, seq = line.replace(">", ""), []
        else:
            seq.append(line)
    if id:
        yield (id, ''.join(seq))

def filter_max_gc(fasta_path: str) -> tuple[str, float]:
    """Return the sequence and id of the DNA string with the highest GC-content."""
    with open(fasta_path) as fp:
        max_id, max_gc = None, 0
        for id, seq in read_fasta(fp):
            gc = compute_gc(seq)
            if gc > max_gc:
                max_gc = round(gc, 6)
                max_id = id
    return (max_id, max_gc)
```

```
result = filter_max_gc(fasta_path)
print(*result, sep='\n')
```

```
Rosalind_0808
60.91954
```