Consensus and Profile

Jan Emmanuel Samson

2024-07-30

Problem

A **matrix** is a rectangular table of values divided into rows and columns. An $m \times n$ matrix has m rows and n columns. Given a matrix A, we write $A_{i,j}$ to indicate the value found at the intersection of row i and column j.

Say that we have a collection of DNA strings all having the same length n. Their **profile matrix** is a $4 \times n$ matrix P in which $P_{1,j}$ represents the number of times that 'A' occurs in the j^{th} position of one of the strings, $P_{2,j}$ represents the number of times 'C' occurs in the j^{th} position, and so on.

A **consensus string** c is a string of length **n** formed from our collection by taking the most common symbol at each position; the j^{th} symbol of c therefore corresponds to the symbol having the maximum value in the j^{th} column of the profile matrix. Of course, there may be more than one most common symbol, leading to multiple possible consensus strings.

- **Given**: A collection of at most 10 DNA strings of equal length (at most 1 kbp) in FASTA format.
- **Return**: A consensus string and profile matrix for the collection. (If several possible consensus strings exist, then you may return any one of them.)

Sample Dataset

```
>Rosalind_1
ATCCAGCT
>Rosalind_2
GGGCAACT
>Rosalind_3
ATGGATCT
>Rosalind_4
AAGCAACC
>Rosalind_5
TTGGAACT
>Rosalind_6
ATGCCATT
>Rosalind_7
ATGGCACT
```

Sample Output

```
ATGCAACT
A: 5 1 0 0 5 5 0 0
C: 0 0 1 4 2 0 6 1
G: 1 1 6 3 0 1 0 0
T: 1 5 0 0 0 1 1 6
```

Intuition

We could reuse the `read_fasta` function from the previous problem entitled *Computing GC Content*. After reading in the FASTA file, we extract the sequences into a list of strings. A dictionary of length 4 is initialized with an empty list; keys are one of the four nucleotides 'A', 'C', 'G', and 'T'. The list can be transposed so that counting the nucleotides per column would be easier. We iterate from 0 up to the length of the first string in the list, keeping track of the base count at specific indices across rows.

Solution

Import the `read_fasta` function from a previous exercise and write a helper function for tranposing the 2D array of sequences:

```
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 transpose(seq_list: list[str]):
    """Convert columns into rows."""
    T = []
    n = len(seq_list[0])
    for i in range(n):
        col = []
        for seq in seq_list:
            col.append(seq[i])
        T.append(col)
    return T
```

Implement a function for computing the profile matrix, and generating the consensus sequence from the profile.

```
def profile(seq_list: list[str]) -> dict[str, list[int]]:
    """Generate the nucleotide counts for a list of string of equal length."""
    profile = {n: [] for n in 'ACGT'}
    T = transpose(seq_list)
    for seq in T:
        for base in 'ACGT':
            profile[base].append(seq.count(base))
    return profile

def consensus(pmatrix: dict[str, list[int]]) -> str:
    """Determine the most probable string from a profile matrix."""
    c = ""
    n = len(pmatrix['A'])
    for i in range(n):
        max_count = -1
        nuc = None
```

```
        for base in 'ACGT':
            if pmatrix[base][i] > max_count:
                max_count = pmatrix[base][i]
                nuc = base
        c += nuc
    return c
```

```
seq_data = []
with open(fasta_path, "r") as file:
    for id, seq in read_fasta(file):
        seq_data.append((id, seq))

seqs = [s for i, s in seq_data]
prof = profile(seqs)
cons = consensus(prof)
print(cons)
for base, counts in prof.items():
    print(f"{base}: {' '.join(map(str, counts))}")
```

```
ATGCAACT
A: 5 1 0 0 5 5 0 0
C: 0 0 1 4 2 0 6 1
G: 1 1 6 3 0 1 0 0
T: 1 5 0 0 0 1 1 6
```