Explanation

Danielo Zamor

2025-05-08

# K-mer Analysis: Design and Implementation Notes

## 1. Data Structures Used to Store K-mers and Their Context

The core data structure used in this project is a **nested dictionary** implemented via collections.defaultdict(Counter):

```python from collections import defaultdict, Counter kmer\_dict = defaultdict(Counter) In this assessment I use the defaultdict and the counter function to map each uniwue k-mer string to a counter object, at the same time the counter function tracks the frequency of each nucleotide that immediately follows the k-mer.

## 2. Handling Edge cases

I try to consider several edge cases. For the first k-mer, I did not think special handling was needed because it begins at sequence [0:k] and it has a valid following character. The last k-mer I exluded it because there to no nucleotide that follows it to count to avoid errors. Invalid characters, such as ‘N’ or ‘X’, result in the entire k-mer and context being skipped to avoid contaminating the counts with unknown bases. For short sequences, if the input sequence is shorter than k + 1, no k-mers are counted, and a warning or zero-result output is expected.

## 3. Overcounting and Missing context

To avoid over counting I inserted a loop to iterate through all possible starting indices i. This way all valid k-mers with a follow up base were counted once. Before counting, the combined string kmer + next\_char is validated to contain only ‘A’, ‘C’, ‘G’, or ‘T’. This validation step prevents counting k-mers that would result in invalid or ambiguous context due to non-standard bases.