

Project 2

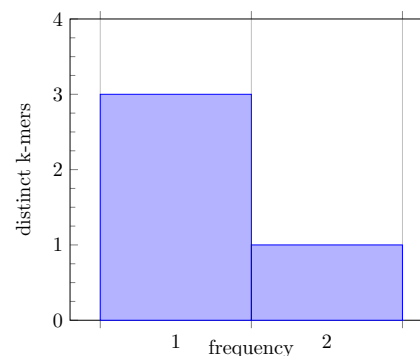
Deadline: Sunday, 10 April 2022

Background information

In this project you will work with DNA sequences. Some definitions:

- A DNA sequence S of length n is an n -letter string made up of the letters "A", "C", "G", "T", representing a sequence of nucleotides.
- A k -mer K within S is a k -letter substring of S , i.e. a sequence of k consecutive nucleotides in S .
- The frequency of a k -mer K within S is the number of times the substring K can be found in S .
- A k -mer spectrum is a graphical representation of a DNA sequence S in the form of a histogram, showing how many unique k -mers appear a certain number of times in S . The frequency of occurrence is plotted on the x -axis and the number of unique k -mers with that frequency on the y -axis.

For example, the DNA sequence "AACGACG" contains the four unique 3-mers "AAC", "ACG", "CGA" and "GAC", with all 3-mers appearing once, apart from "ACG", which appears twice. Hence in this trivial example, the histogram representing the 3-mer spectrum would look like shown on the right.



Throughout this project you may assume that $n \leq 10^8$ and $k \leq 10^2$.

Task 1

Write a Python function that, given a DNA sequence S of length n and the number $k < n$ computes some characteristics of S , including the histogram data needed to produce the k -mer spectrum for S .

In particular:

- Your function should be defined as `kmer_hist(S, k)` and return two lists: `h`, `mfkmers`.
- The list `h` should contain at position ℓ the number of unique k -mers in S with frequency ℓ in S . The final entry in `h` should correspond to the frequency of the most frequent k -mer(s) in S .
- The list `mfkmers` should contain the most frequent k -mers in S . Hence every element of `mfkmers` should have frequency equal to `len(h)-1` in S . (Note that the length of `mfkmers` will often be 1.)

Your function `kmer_hist` may use all available standard types and methods in Python 3, but must work without importing any packages or modules. In particular, your function *should not* produce any graphical plots.

Together with your code you should submit a report with a very short description of your algorithm and an analysis of your algorithm's asymptotic complexity. Here you may for simplicity assume that *any* reading or writing access to a **dictionary** in Python is $\mathcal{O}(k)$, when the key is a string of length k .

Task 2

Use your function `kmer_hist` from the previous task to compute and visualize some k -spectra for real-life DNA sequencing data. To this end, download the publicly available DNA sequencing data in FASTA format for

- (a) the Escherichia coli bacteria:
`https://www.ncbi.nlm.nih.gov/nuccore/U00096.3?report=fasta`
- (b) the human chromosome 21:
`https://www.ncbi.nlm.nih.gov/nuccore/CM000683.2?report=fasta`

You can find more details on the FASTA data format, and on the used notation at the links

- `https://en.wikipedia.org/wiki/FASTA_format`
- `https://en.wikipedia.org/wiki/Nucleic_acid_notation`

Note that before passing the data to your function `kmer_hist`, it needs to be “cleaned”. For example, the FASTA data may contain comments, i.e. lines starting with the character “>”, and the sequencing data may contain letters other than “A”, “C”, “G”, “T”. For the purpose of this task, you should remove any such letters before your analysis.

In your report, include the k -mer spectrum for the above e-coli and human chromosome 21 genomes for $k = 8$. In addition, for $k = 3, 8, 15, 25, 35$, report in a table on the most frequent k -mers in the two sets of sequencing data, stating their frequency and the k -mers themselves.

Task 3

Write a Python function that, given a DNA sequence S of length n and a list of k -mers L of length m , with $k < n$, finds the most frequent k -mer from the list L in S . You may assume that $m \leq 10^2$.

In particular:

- Your function should be defined as `kmer_search(S, L)` and return two integers: `pos`, `freq`.
- `pos` should be the first location in S of the most frequent k -mer listed in L . If there is more than one k -mer with the highest frequency, `pos` should refer to the k -mer that appears first (left-most) in S .
- `freq` should be the number of times that `S[pos:pos+k]` appears within S .
- If none of the k -mers listed in L can be found in S , simply return `None`, `0`.

Your function `kmer_search` may use almost all available standard types and methods in Python 3, and must work without importing any packages or modules. However, in contrast to Task 1, your implementation should **not** use the Python types `dict`, `set`, or the Python function `hash()`.

Similarly to Task 1, in your report include a concise description of your algorithm and an analysis of your algorithm's asymptotic complexity.

Submission via Moodle

Prepare a single Python source file called `project2.py` that contains your functions `kmer_hist` and `kmer_search`, together with any routines or classes that you have written and that your functions depend on, but nothing else. Importing your source code via `import project2` should **not** execute any code. The docstring of the file `project2.py` must contain your full name. In addition, prepare a report in PDF format, called `project2.pdf`, that includes your answers for Tasks 1, 2 and 3.