Technical University of Munich

ROSTLAB.

Chair for Bioinformatics and Computational Biology
Advanced Bioinformatics SoSe 2024
Prof. B. Rost
L. Richter

SoSe 2024
Exercise sheet 0
20.04.2024 – 25.04.2024

Python Refresher

(H) Python

Remark: This homework on completing and submitting `exe0_python_intro.py` **will not be graded** and only serves as a refresher or intro to python. Nevertheless, you are still **expected** to **be able** to at least solve the tasks in pseudocode. Do **not change** the name of this file. (This will be explained in the next exercise.)
Remark: You might want to have a look at the types of the input and output of the function to get an idea how the function should behave. Furthermore, you are allowed to add helper functions if necessary but you may **not change** the **signature** of the given functions. Feel free to create a main function to run the implementation of the requested implementations.
The archive contains a file `requirements.txt`. You may use it to adopt your working environment. This is a topic for more advanced python users. If you do not understand this yet, fine as well, we will come back to this later. Just try to complete the tasks. At this point we ask for a personal submission, not for group submissions yet. This is dry-run for you and us to establish the process and make sure things are working as expected and bugs are fixed before the graded exercises.

0.1 (H) Quick Intro Python Basics (-)

You will have to write Python code to do the exercises. If you do not know this language, please check out the Python tutorial.

0.2 (H) AA Comp (-)

Complete the function `get_aa_composition`. The function receives a string that represents a protein sequence. Count the occurrence of each amino acid in the sequence and output an according dictionary.

0.3 (H) K-Mer (-)

Complete the function `k_mers` that outputs all possible k-mers of length k given an alphabet. E.g. *alphabet* $=' ML'$, then all possible 3-mers are
$[' MMM',' MML',' MLM',' MLL',' LMM',' LML',' LLM',' LLL']$.
$\rightarrow$ Input:

- *alphabet*, the underlying alphabet of the words you want to count the k-mers for. Create all possible k-mers from that alphabet
- *k*, an integer indicating how long the k-mers you are interested in should be.

0.4 (H) Compose the K (-)

Complete the function `get_kmer_composition`. Return the composition of k-mers for a protein sequence given k as dictionary. You are asked to output the complete composition, hence not appearing k-mers should also be listed with a 0-occurrence.

0.5 (E) Locally Aligned with the task

- This task will be presented and demonstrated comprehensively in class. Nevertheless, feel free to engage and try your luck. Your are allowed to pull in weight matrices like BLOSUM from other places.
Complete the function `get_alignment`. For that, implement the Smith–Waterman algorithm. Your implementation does not have to be efficient or complete. It is sufficient to output exactly one alignment if possible, starting at the highest score in the score matrix you can find by first searching row- and then column-wise. *Note*, the scoring matrix should represent the query sequence from top to bottom while the second sequence is put from left to right.
$\rightarrow$ Input:

- *protein_seq_1*, the query sequence you want to find an alignment for.
- *protein_seq_2*, the sequence you align the query sequence with. If gaps are introduced, first introduce them into this sequence.
- *gap_penalty*, an integer stating how much of the alignment score should be decreased if a gap is introduced.
- *substitution_matrix*, a data structure that holds the substitution scores of each amino acid x amino acid pair.