P1: Python programming

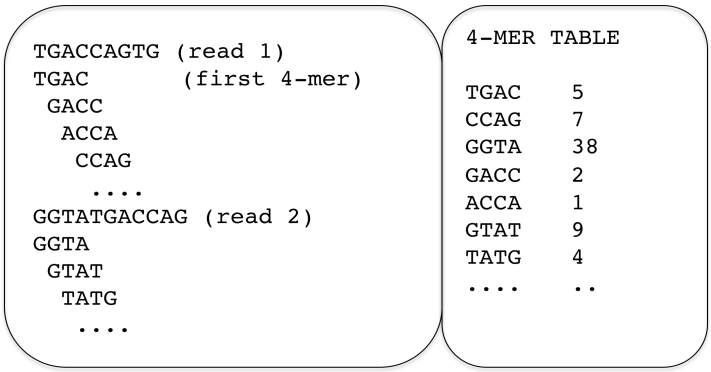
Hi, my name is D.E. Bug. I have created a really cool script that parses a FASTA file and counts the number of 15-mers occurring in sequences. I compare my output to the output of the program jellyfish, which is built to perform this task. Please give it a try on the server (altschul.bioinformatics.nl)! In case you find any bugs, please solve them for me ☺
Script: http://www.bioinformatics.nl/courses/BIF-30806/docs/debug_P1.py
Data: http://www.bioinformatics.nl/courses/BIF-30806/docs/tomato.fasta

Download the provided script and data. Copy the script:
`cp debug_P1.py fixed_P1.py`
Put **your name and student number** in fixed_P1.py. Debug the script until it performs the tasks described below. Compare the fixed version with the original script:
`diff debug_P1.py fixed_P1.py`
Put these differences in the module-level docstring, so we can see what you changed. **Turn in your debugged python script with recorded changes on BlackBoard (under P1)**.

Background

A *k*-mer is a substring (with length *k*) of a read. *K*-mers are for example used in a De Bruijn Graph approach to genome assembly, but a *k*-mer frequency table can also be used to detect and correct base-calling errors. A read of *n* basepairs consists of (*n* – *k* + 1) *k*-mers. For example: a read of 75 bp consists of 45 (overlapping) 31-mers. A *k*-mer table stores the frequency (number of occurrences) of each *k*-mer in a given set of NGS reads.

Creating a *k*-mer table requires the extraction of all overlapping *k*-mers from all the reads in the input, and keeping track of their frequencies. **`jellyfish`** is a program to count *k*-mers in DNA sequences.



Functionality of my script

My script creates a 15-mer table from a set of genomic tomato reads, reports certain statistics, and compares these with the output of the program **`jellyfish`**. It specifically performs the following tasks:

1. Parse a FASTA file with genomic reads from a tomato plant.
2. Create a 15-mer table from the reads by extracting all overlapping 15-mers and storing their counts in the data set.
3. Report the
    - number of unique 15-mers (15-mers that occur only once in the data set)
    - number of different 15-mers in the table (ignoring the frequencies)

   - total number of 15-mers extracted from the reads (so, including multiplicity)
   - max count in the table (highest frequency seen)
   - 15-mer strings that occur with the max count
4. Run the program **`jellyfish`** on the input file to count the *k*-mers and report the statistics. Print the jellyfish output. If my script is correct, the output should match the statistics calculated by my own code. It uses two commands:
   - **`jellyfish count`** (specify –m and –o , set –s to 1000000, of course specify the input) (this command should produce a single output file)
   - **`jellyfish stats`** (give the output from the first command as input)

   The manual of jellyfish:
   http://www.cbcb.umd.edu/software/jellyfish/jellyfish-manual-1.1.pdf

Environment

You should work on the server **`altschul.bioinformatics.nl`**
The program jellyfish is installed there. Try it by typing:

```
jellyfish –h
```

or

```
jellyfish count -h
```

on the command line. You should see information on the usage and options.