UPPSALA
UNIVERSITET

Introduction to Programming in Python and R for Bioscience

Final Python Project: Advanced

This is the advanced version of the complete Final Project. It is more difficult than the Easy version but uses the same data. Besides being more fun, completing this version can lead to Pass with Distinction, the highest grade for the course. Of course, a similarly good result in R will be needed to achieve this grade for the course.

Portions of a gene have been sequenced for a group of patients. These have been compared to a standard reference via BLAST to identify specific mutations (SNPs). The haploids appear as separate hits against the reference to check for differences, which can be hetero- or homozygous (in one or both alleles). We'll work with a simplified version of the analysis which still has enough complexity to act as a realistic problem but doesn't work well with automated tools.

Your data file holds the results for many patients, one after the other. An example of the part of the sequence area for one patient appears below:

```
Query_1   241   ACCGTGCGTGGTTTTTCACAGTTCGGGGAT   270
HAP_A     281   .T........................T...   310
HAP_B     281   ...........C..............T...   310

Query_1   271   TCGTATGTGTATGTCATTTTTGAAGACGGC   300
HAP_A     311   ..............................   340
HAP_B     311   .............................G   340
```

The first line shown here is for the query (reference). It holds the query name, the starting position in the sequence, a portion of the sequence and the ending position for this fragment. The lines immediately following hold the aligned hits (the haploids in this case) with similar fields except that matches to the query are replaced with "." (period) and the differences show the base code for the mutation.

A blank line separates portions of the sequence from each other until the sequence ends. There are other lines before and after the sequence data. These "header" and "footer" lines are not relevant for this study except to identify the division between patients, each of which will follow a similar pattern.

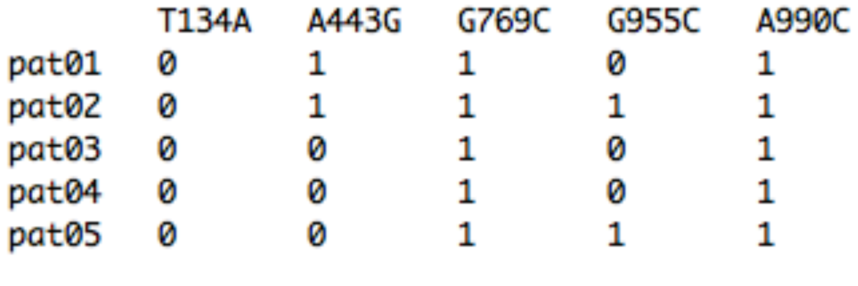
Looking at the data in this area, we see a total of two SNPs (mutations) for HAP_A and three for HAP_B. The positions of the differences can be read from the position in the line relative to the start (or finish) of the query (ignore the position numbers for the haps). For example, HAP_A shows C242T, where C is the base in the reference, T is the "mutation" and 242 is the position in the query (one away from the start, 241). Similarly, HAP_B shows C300G (not 340). For this exercise, we'll pay most attention to the position numbers and use the full forms (with base codes) only in table headers. Note that there are many differences from the reference but only a few of these will be relevant in this study. This is a common situation and the list of interesting positions will be given.

Instructions:

Input data is individual. Input data is individual. Check Studium for your number -- same as given for the Python Servers. If your aren’t listed, contact the teacher. Download and use it.

Write a program with standard modules to process the data file to find if the following mutations are present for the patients: **T134A, A443G, G769C, G955C, A990C, G1051A, G1078T, T1941A, T2138C, G2638T and A3003T**. For this exercise, the actual DNA base changes (ACGT) can and should be ignored for comparisons (e.g., assume any change at position 134 is T to A).

Results should be recorded as follows: no difference from reference is 0, heterozygous (only appearing in one allele/hap) is 1, homozygous (both alleles) is 2. A header row should appear first and the results for each patient ("pat01", etc) should make a tab-delimited table similar to the following (but with more columns and rows):



	T134A	A443G	G769C	G955C	A990C
pat01	0	1	1	0	1
pat02	0	1	1	1	1
pat03	0	0	1	0	1
pat04	0	0	1	0	1
pat05	0	0	1	1	1

Show some of the work leading to your solution. Breaking the task in few functions is recommended but not strictly required. Please comment your code for readability (but not too much). Briefly explain what you did and why – possibly as comments in separate cells. Cite anything taken from someone else (another student or externally). Do not copy too much from other sources. This should primarily be your own work.

External packages that would need to be downloaded are strictly forbidden since they will not be installed in the test environment. Only upload your scripts (preferably as a Jupyter notebook). Do not submit the resulting solution file or the data. To pass, the examiner must be able to execute your code and get a (reasonably) correct solution.

Suggested Procedure:

The following is a list of suggested "subgoals" to help you. None are strictly required but some progression from simpler scripts to the final should be shown.

1. Write a script which loops over the data file. Print the patient number (starting from 1) and the last line for each patient. Patient numbers are simply the order in which they appear in the file. Look through the file to determine how to identify the last line for each patient.

2. Identify lines which hold the query sequences. Add this to the main loop. Mark them as "found" for the next pass of the loop. Also get the number for the current starting position.

3. Recall that blank lines separate groups of sequence lines (query and hits). Look for "blank" lines (allowing for a couple characters like newlines, etc). This is a good place to reset the "found" variable.

4. Identify lines where "found" is true in the main loop, i.e., between the query and the blank line. This is where the you'll gather data from the hits. Identify the mutations, maybe by looping over the characters in these lines to look for letters or "not a period" within the sequence section of the line. Recall that the numbers are based on the starting position of the **query**. Check for matches against the list of interesting mutations or record them all to be processed later.

5. Gather data for each patient. A good place to complete this is in the section made in step 1 to identify the end of the patient record. Could do the processing/printing through a function you define or directly in the main loop.

6. Clean up output with tabs between the fields as shown in the instructions. Note that patient 10 is "pat10", not "pat010". Of course, the sample above is incomplete and probably different than what you'll get for your data file.