1. Which program and task(s) you have implemented

I use read_mapping.py to solve the Read mapping task.

2. How the program is started and used.

2.1 Usage of read_mapping.py

Take reads_small.fa and refs_small.fa as an example, you enter the belowing code on the command line:

python read_mapping.py --k 3 refs_small.fa reads_small.fa output/

The "k" represents the length of the cutting window. For short and repetitive sequences and reference sequences with a large number of repetitions, it is easy to make a wrong match, so it is recommended to adjust the size of the “k” value and use a higher matching threshold.

The higher matching threshold can be adjusted by the parameter "match_threshold", which represents the base matching threshold in each cutting window, with default equaling to 0.7, meaning no less than 70% matching rate. This parameter can be adjusted according to different situations. (Note: match_threshold is not a necessary parameter, not entering it on the command line means to use default match_threshold)

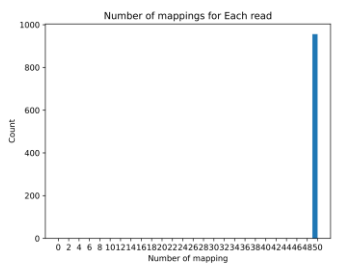
"refs_small.fa" and "reads_small.fa" are the fasta file for reads and the fasta file for references, respectively; and "output/" means the output folder where the output file will be saved.

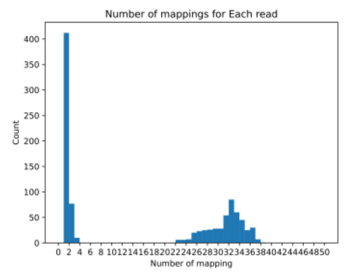
2.2 Routine test data and results

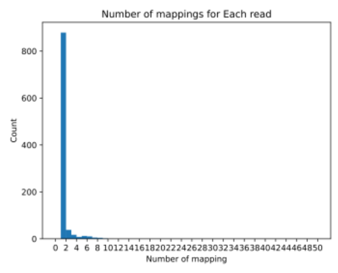
Regarding to the testing, different kinds of data sets can be used for validation. I used 5 sets of data (such as reads1.fa and ref1.fa) provided on the course website. All test data is saved in the “test_data“ folder, and all corresponding test results are saved in the ”test_results” folder. Please note that when testing data, you need to place the data and python file in the same directory, or use equivalent methods to run it.

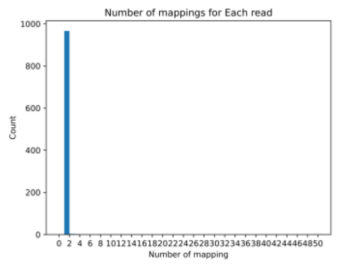
The reference sequence with the highest coverage in each alignment can be found based on the reference_coverage.txt file. In the combination of refs0_reads0, the ratio of all reference sequences is the same, 0.956. In the combination of refs1_reads1, the proportion of ref17 reference sequence is the highest 3.164. In the combination of refs5_reads5, the proportion of reference sequence ref19 is the highest 3.336. In the combination of refs10_reads10, the proportion of reference sequence ref16 is the highest, which is 4.867

The test results can be seen in the figure below. From left to right and from top to bottom are the test results of reads0.fa and refs0.fa; reads1.fa and refs1.fa; reads5.fa and refs5.fa; reads10.fa and refs10.fa.









2.3 Abnormal data and results

2.3.1 Parameter Input Errors

Argparse is able to recognise and report errors in parameter input, such as missing parameters. In addition, I also use try- except FileNotFoundError statement in the read_fasta_to_dict() function for error handling, in order to cope with the case of file name error. Tests can be found below (e.g. not entering the last folder location parameter or entering the wrong file name):

> python read_mapping.py --k 3 refs_small.fa reads_small.fa

usage: read_mapping.py [-h] [--k K] [--match_threshold MATCH_THRESHOLD] infile_ref infile_read outfile
read_mapping.py: error: the following arguments are required: outfile

> python read_mapping.py --k 3 refs_small.fa reads_smell.fa output/

The file reads_smell.fa you entered does not exist.Please check it again.

2.3.2 File error

2.3.2.1 The file format is not in fasta format

If the input contains non-ATGC words, use the following code (error file: reads_small_wrong_characters.fa) to test:

> python read_mapping.py --k 3 refs_small.fa reads_small_wrong_characters.fa output/

ValueError: DNA sequnece AAACCCCCK contains at least one invalid character

2.3.2.2 Reads compare to the end of reference

In this case, the remaining length of read sequence is longer than the remaining length of reference sequence, resulting in an error. Note: There will be many end reference mapping under the seed-extension method, so when running a large data set, there will be a lot of such errors. If you do not need this function, you can just turn off the error reporting function end_ref_error(), which will not affect the running of the program. Use the following code to test (error file: reads_small_ref_end.fa):

> python read_mapping.py --k 3 refs_small.fa reads_small_ref_end.fa output/

Error: Mappng outside the border of the reference

2.3.2.3 Read is too long

In this case, the read length is longer than the reference, and comparison cannot be performed. Use the following code to test (for errors, see reads_small_too_long.fa):

> python read_mapping.py --k 3 refs_small.fa reads_small_too_long.fa output/

ValueError: The length of read sequnce read1 is longer than the length of reference sequnce ref1.

2.3.2.4 Read is too short

In this case, the read length is shorter than k, so the comparison cannot be performed. Use the following code to test (error file: reads_small_too_short.fa):

> python read_mapping.py --k 3 refs_small.fa reads_small_too_short.fa output/

ValueError: Sequence read1 is shorter than k (3)

3. Which libraries/modules are used and how these are downloaded and installed if they are not part of Python's standard distribution.

I used argparse and matplotlib. The argparse module is used for parsing command-line arguments in a convenient way. It's a standard library that comes pre-installed with Python.

Matplotlib is used for creating visualizations, such as plots and charts. I installed it by running the following command "pip install matplotlib" in the terminal, otherwise it can be downloaded from the command line using conda's command at https://anaconda.org/conda-forge /matplotlib, the code is "conda install conda-forge::matplotlib"

4. A description of how you structured your program (which files contain what, etc.).

The entire file includes the report document, read mapping pthon file, and two folders test_data and test_result. The test_result folder stores the results of five test data sets, and each subfolder covers three required output files. The test_data contains different test fasta files.

This program can compare reads to reference sequences, compute the coverage of the reference sequences and output the corresponding image and documents. It uses a k-length window to cut reads and reference sequences, and builds a library containing k-length strings. Based on the first k-string of the reads as seeds, it extends the comparison if it meets the matching threshold, and then filters it to get the read-reference pairs with the smallest mismatch and outputs the results. The first output best_mapping.txt contains all the best locations for read-reference pairs with the smallest mismatch rate, the second output reference_ coverage.txt contains the reference coverage, and the third output reference_coverage.pdf is a histogram plot displaying the number of mapping locations (x- axis) per read.

5. Notes on citations and references in this article

1) Regarding SystemExit in the read_fasta_to_dict() function, I referenced the following URL：
https://www.codenong.com/13992662/

2) Regarding the check_dna(seq) function, I referenced the detection of non-ATGC sequences provided by the “lab material” in this course(DA7066).

3) For the while and else statements in extend_seeds, I referenced the following URL：
https://zhuanlan.zhihu.com/p/565981882

6.Reflection on code design

For a detailed introduction to each function, please refer to the documentation in the python file. I would like to talk about my reflection on one of the most important functions in my code: extend_seeds() ,which uses the method of moving the reading frame of length k by k distances each time for comparison.

This function can also be implemented in another way, i.e., by moving k-mers one by one for comparison. In this case, the length of the function would be much smaller than the current version because it does not need to consider the issue that the read not being able to be divided into a complete set of k-mers. Therefore, there won't be the “else” statement corresponding to the “while” loop present in the current version. However, the downside of this approach is that the reading frame moves very slowly, significantly increasing the computational workload (at least by around k times).

Therefore, due to the complex logical relationships and numerous involved parameters, it is challenging to ensure that all functions consist of only a few lines. Besides, forcibly splitting them may lead to unnecessary parameter passing. I have tried my best to compress or split all functions including extend_seeds() into cases that meet the conditions. However, for this function in my data structure, if the functionality can reduce the computational workload, the length of the function may be acceptable.

7. Time efficiency of the code

This function takes less than 5 minutes to complete the mapping of files as large as reads1.fa and refs1.fa.