ITO Integrative Transcriptomics
Prof. K. Nieselt,
Institute for Bioinformatics and Medical Informatics Tübingen
Prof. S. Nahnsen,
Institute for Bioinformatics and Medical Informatics Tübingen

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Lecture: Grundlagen der Bioinformatik — SoSe 2022

Assignment 8 (20 points)

Hand out: Thursday, June 23
Hand in due : Thursday, June 30, 18:00
Direct inquiries via the ILIAS forum or to your respective tutor at:
caroline.jachmann@uni-tuebingen.de
meret.haeusler@student.uni-tuebingen.de
simon.hackl@uni-tuebingen.de
carolin.schwitalla@qbic.uni-tuebingen.de

Theoretical Assignments

1. **Sequencing Experiment** (3P)

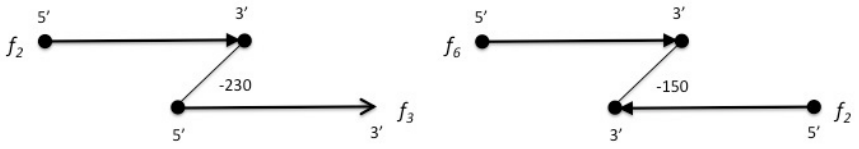
   Assume you discovered a new microorganism that seems to produce a natural product with antibiotic properties. You want to know more about this organism and you plan to generate the genome of it. Design a sequencing experiment that will allow you to assemble the genome (the estimated size of the genome is 6 MB). Which sequencing platform(s) would you suggest to use and why? What could be potential difficulties that come with the generated data? And how would you design your bioinformatics analysis to assemble the genome and to overcome the difficulties.

2. **Assembly with OLC** (5P)

   Given reads $\mathcal{F} = \{f_1, f_2, \ldots, f_8\}$, each of length 500, that overlap as follows:

   - $o(5'f_3 - 3'f_2) = 230$
   - $o(5'f_5 - 3'f_8) = 150$
   - $o(5'f_5 - 5'f_7) = 50$
   - $o(5'f_8 - 5'f_7) = 400$
   - $o(3'f_4 - 5'f_1) = 400$
   - $o(3'f_1 - 3'f_5) = 350$
   - $o(3'f_4 - 3'f_2) = 40$
   - $o(3'f_4 - 3'f_5) = 250$
   - $o(3'f_6 - 3'f_2) = 150$
   - $o(3'f_6 - 3'f_3) = 420$
   - $o(3'f_6 - 3'f_8) = 80$
   - $o(3'f_7 - 3'f_3) = 100$
   - $o(3'f_7 - 5'f_6) = 180$

   where e.g. $o(5'f_3 - 3'f_2) = 230$ means that the 5′-end of f_3 overlaps the 3′-end of f_2 by 230 bases and $o(3'f_6 - 3'f_2) = 150$ means that the 3′-ends of f_6 and f_2 overlap by 150 bases (see below).



(a) **The overlap graph:** Based on the provided overlaps, draw the overlap graph OG, labeling all edges as discussed in the lecture.

(b) **A minimal spanning tree:** Draw a minimal spanning tree in OG, containing all read edges.

(c) **The layout:** Draw the layout of the reads as given by the minimal spanning tree, indicating the coordinates of the start and end of each read. What is the length of the final assembly?

(d) **Consistent overlaps:** Are all overlaps consistent with the computed layout? If not, which overlaps are not consistent with the layout, and why?

Hand in the overlap graph, the resulting layout as well as the consensus sequence, you can hand in a handwritten and **scanned** solution. **Note:** If you choose to hand in a handwritten solution please provide clear and legible solutions. If you do not have the possibility of scanning, you can use mobile apps that return good quality scans, instead of only including a picture of your notes.

3. The Weather in the Land of Oz (2P)

In the lecture the weather in the land of Oz can be modelled using a Markov model. It has the three states (Nice=N, Rainy=R, Snowy=S), and the transition matrix

$$P = [P_{ij}] = \begin{array}{c|ccc} & N & R & S \\ \hline N & 0 & 0.5 & 0.5 \\ R & .25 & 0.5 & .25 \\ S & .25 & .25 & 0.5 \end{array}$$

(a) Assume it is rainy today, what is the probability that it is snowy in two days?

(b) Generalize this to an equation computing the probability that any Markov chain with k states, starting in state s_i, will be in state s_j after n steps.

Practical Assignments

4. Markov Chains (10P)

In this task we ask you to implement methods to simulate and evaluate, i.e., calculate the log probability, of Markov chains with respect to a Markov model. We provide you with a simple implementation of Markov models, so you do not have to implement a full framework on your own: The class `MarkovModel` already comprises methods to read and store Markov models and you will not have to change the code of the `Main` class for this task – still, make yourself familiar with the code. Once you implement the remaining code fragments the resulting `JAR` will be executable with the command

```
java -jar <NAME.jar> -mm <FILEPATH.mm> -s -mc <FILEPATH.seq>
```

- `-mm`: File path to a .mc file in the following format, specifies the markov model.

```
# Number of states:
6
# State labels:
A C G T * +
# Transition matrix:
.2995 .2045 .2845 .2095 0 .002
.3215 .2975 .0775 .3015 0 .002
.2475 .2455 .2975 .2075 0 .002
.1765 .2385 .2915 .2915 0 .002
.2495 .2495 .2495 .2495 0 .002
0     0     0     0     0 1
```

- `-s`: Prints a simulated Markov chain and its log probability wrt. to the stored model to the console (optional flag, no arguments).
- `-mc`: File path to a .seq file. The file content is interpreted as Markov chain. A log probability will be printed to console for its content.

(a) Implement the `simulate` and `getLogProbability` methods in the `MarkovModel` class. The methods should simulate a Markov chain and compute the log probability of a string interpreted as markov chain with respect to the currently loaded model, respectively.

(b) Use your program to simulate one Markov chain once with the `CpG.mm` and the `nonCpG.mm` files (both are located in the resources directory of the code skeleton). Attach the output of the program to your pdf report. How do you evaluate the validity of the two provided Markov models in the light of the computed log probabilites?

(c) Use your program to compute log probabilities of the two provided files `sequence1.seq` and `sequences2.seq` (both files are again provided in the resources directory) once with the `CpG.mm` and the `nonCpG.mm` files specified as models. Attach the output of the program to your pdf report and additionally provide the ratio of the respective probabilities. What do you conclude from the results in the light of classifying the input sequences (wrt. CpG-islands)?

Please read the questions carefully. If there are any questions, you may ask them during the tutorial session or in the forum of ILIAS. You will usually get an answer in time, but late e-mails (e.g. the evening of the hand-in) might not be answered in time. Please upload all your solutions to ILIAS. Don't forget to put your names on every sheet **and** in your source code files. Please pack both your source code as well as the theoretical part into one single archive file and give it a name using this scheme: `<name1>_<name2>_<Assignment>_<#>.zip`. The program should run without any modification needed.