CM124 Spring 2018 - Programming Assignment (due June 6th)

May 18, 2018

Haplotype phasing in recently admixed populations

In this assignment you are going to develop and implement an algorithm that takes as an input the genotypes of individuals coming from admixed populations and infers the haplotypes of the individuals.

Genotyping technologies provide us with strings over $\{0, 1, 2\}$, representing the number of copies of the reference allele at each SNP. However, these technologies do not allow us to measure complete haplotypes (i.e. what is exactly the sequence of alleles in each of the two chromosomes of an individual). The problem gets further complicated when phasing haplotypes of individuals coming from recently admixed populations.

Recall that the creation of an admixed population is a process that initially starts with two or more homogeneous populations (i.e. each individual is strictly coming from one particular population). Then, at each new generation, following a process of random mating, each individual gets two haplotypes, one per parent, from the previous populations (the two haplotypes may come from parents who are part of different populations). Importantly, each of the two haplotypes is going through some recombinations with an unknown recombination rate r. In our case, we consider admixed individuals from two recently admixed populations (i.e. not many generations have passed since the homogeneous populations), under the assumption of random mating.

Develop and implement an algorithm for phasing haplotypes of individuals coming from recently admixed populations. You are free to use any of the methodologies that we have seen in class or that you can find online, and you can use existing software packages as long as they were not designed for haplotype phasing.

Assignment files and compiling your solutions

This assignment is accompanied by the following files:

1. EXAMPLE GENOTYPES: "example_data_1.txt", "example_data_2.txt", "example_data_3.txt" - three example datasets to develop and test your method on. Each file contains the genotypes for a set of

individuals. Format of files: each row in the file is a genomic position (SNP), each column in the file is an individual. Values are separated by spaces.

2. EXAMPLE HAPLOTYPES: “example_data_1_sol.txt”, “example_data_2_sol.txt”, “example_data_3_sol.txt” - each file contains the true haplotypes for ones of the three sets of example genotypes. Format of files: each row in the file is a genomic position (SNP), each pair of consecutive columns in the file is an individual (i.e. for an individual’s genotype column i, the corresponding haplotypes are located at column $2i$ and $2i + 1$). Values are separated by spaces.

3. TEST GENOTYPES: “test_data_1.txt”, “test_data_2.txt” - each file contains the genotypes for a set of individuals. Once you have developed your method, run your method on these test genotypes and output your estimated haplotypes. These are the haplotypes you will submit for grading. Format of files: each row in the file is a genomic position (SNP), each column in the file is an individual. Values are separated by spaces.

4. GENOTYPES POSITIONS: “example_data_1_geno_positions.txt“, “example_data_2_geno_positions.txt“, “example_data_3_geno_positions.txt“, “test_data_1_geno_positions.txt“, and “test_data_2_geno_positions.txt“ - for each of the example and test dataset, you get the physical positions of the genotypes on the chromosome. This additional information can be incorporated into your algorithm (however, you are not required to use it).

In your final submission you are required to submit your estimated haplotypes for all the individuals in the files “test_data_1.txt” and “test_data_2.txt”. Specifically, submit two files “test_data_1_sol.txt” and “test_data_2_sol.txt”, each with the inferred haplotypes of the genotypes in the corresponding test file using the same format as the example haplotype files (i.e. “example_data_1_sol.txt”, see above).

Importantly, the example datasets are merely provided to you as examples of possible input/output. Note that the input of these datasets do not necessarily represent the input of the test data.

Evaluation of performance

The metric we will be using to measure the performance is the switch accuracy, which is formally defined as $\frac{(n-1-sw)}{(n-1)}$, where n denotes the number of heterozygous sites and sw is the number of switches between neighboring heterozygous sites needed in the computer-phased haplotype to recover the original haplotype sequence (see Lin et al. *AJHG* 2002 for a pictorial explanation).

We are providing an evaluation script you can use to evaluate your algorithm. The same script will be used to grade your final homework submissions. This script requires the *R* software environment, and can be run using the following command:

```
Rscript calculate_switch_accuracy.R [estimated haplotypes file] [true haplotypes file]
```

One of the test files will be randomly selected for use in preliminary testing (see below), and the other test file will be used for the final assignment grade.

General instructions:

- Submissions are accepted for groups of 1 to 3 people
- Your solution will be graded as follows:
  - Report and Code (30 points): Submit your code and a brief summary with a description of your algorithm and implementation (one page maximum). Please submit your report as a PDF document. In addition, prepare a file "readme.txt" which includes the command line commands which are to be executed in order to run your code on the three test datasets.
  - Performance (70 points): algorithm performance will be graded in 3 ways (cumulative):
    - (20 points): the switch accuracy of your algorithm is better than an algorithm that choses randomly (i.e. better than 0.5).
    - (20 points): the switch accuracy of your algorithm is better than 0.7.
    - (30 points): your algorithm's performance compared to other groups. This will be calculated as follows: $30 \times \left(1 - \frac{\text{your rank}-1}{\#\text{groups}}\right)$. For example, if your performance is ranked number 5 in the class and there are 40 groups then your score will be 27. If there are ties in the ranking between two groups, the average rank will be used.
  - Submit your solutions electronically via CCLE by submitting a single zip file. Your zip file should include:
    - Your solutions for the two test datasets (two files)
    - Summary/report file
    - Code file(s)
    - "readme.txt" with the execution instructions (see above)

    The title of the zip file should be the UID of the student (e.g., "123456789.zip"). If you submit together with other students, separate your UID numbers by a dot (e.g., "123456789.987654321.zip").
  - For this assignment you are free to use whichever script programming language you prefer (i.e. a language which does not require compilation, such as R, Python, etc.). Note that your code will be tested on a Linux/Unix machine.
  - If you would like to submit preliminary results using the test genotypes, we will randomly select one of the test sets and compute the accuracy of your predictions. This will be done using all submitted preliminary results on May 23rd and May 30th using the same test set both times. The other test set will be used for the final grading. The accuracy of all submissions will be posted to

CCLE so that you can see your rank with respect to other groups. To submit preliminary results, compress your prediction files into a zip file and submit the file to CCLE under “Programming Assignment Preliminary”. The name of this zip file will be used to identify your predictions on CCLE, so if you would like your submission to remain anonymous, name the zip file something identifiable only by you (ex. “foobar123.zip”). Please do not use vulgar language.