WRIGHT-FISHER PROJECT

Alice Albrecht, Noémie Chillier, Loris Constantion, Alexia Dormann, and Aïman Lavallard Fadlane

December 2018

1 Introduction

The aim of this program is to simulate the evolution of a genetic population over time. A population is composed of individuals, each one having a genome. At each generation, a new population of the same size is randomly chosen among individuals of the last population following the Wright-Fisher model. This model works with fixe population size, so that alleles compete only against other alleles and not against an external environment, and use a multinomial distribution to define new frequencies for each alleles (e.g. different genotypes). All this allows the population to evolve through generations. ajouter une phrase sur la migration

2 Implementation

To realize this program, we made several class with several links between them. We made also some tests te ensure that our programme works as we want it to.

2.1 Class

Simulation — The-is is the most important class. It reads the inputs of the user in the TCLAP in **main.cpp**. Form this informations, it creates the **-R** simulations of the same population. It then run the programme **-G** time following the Wright-Fisher model, i.e calling the **multibinomial** function. And finally, it print each step of the programme on a file **-P** or directly on the terminal **-T** and also the genetic code of each population.

Population — This class create -R same populations each time a simulation is create, which contain a **-N** individus each has **-A** alleles. The parameters use to create a population are given by the user in a terminal or by reading a fasta file **-F**.

Multibinomial — The aim of this class is to calculate, at each step, the new frequencies of each population following the Wirth-Fisher model. With this model, we can see that a alleles tends to disappear or to have a frequency equal to 1, i.e be the only last one allele int the genome after a certain generation time. In this case we can say that this allele is a dominant one.

ReadFasta — The aim of this class is to read a fasta file in the case that the user enter a file name **-F**. Reading this file with the corresponding markers **-m** gives the alleles and the frequencies of the population.

2.2 Migration extenstion

À completer... We choose to improve our simulation by adding the ability to move a given number of individuals between different populations. Population exchanges are determined by rates rij (fraction of population i that is moving to population j for each time step. Population sizes are fixed, therefore number of individuals moving out of i must be equal to number moving into i for each population . The rates are implemented in n*n matrix (n= total number of populations). For this, we create a new class, name **migration** which is friend with the class **simulation**. THAT MEANS...

2.3 Tests

je te laisse faire la partie tests comme tu le souhaite! Oui tu peux tout changer là dedans, je suis pas satisfaite de ce que j'ai fait... mais j'ai pas d'inspi pour faire un truc bg. ..
To execute the tests, type the command: make test

A. Test Multibinomial

- sameAverage: tests the stability of alleles frequencies over time. For a large number of populations, frequencies after 1, 2, 3 generations on average are the same has for t=0.
- fixation time: for any simulation, one allele will always reach a frequency equal to 1 (so only one to remain). This test verify it's the case (with 2 and 5 alleles)

B. Test Simulation

- somme1: check if sum of alleles frequencies remains equal to 1 after a given number of calls to multibinomial (10 in this case).
- fixation: controls that if an allele disappear (ie. its frequence is null) it won't reappear in the simulation (corresponds to extinction of the character).

3 Use of the Program

To run the simulation, we can either gives argument from the terminal or read information from a fasta file, provide by the user with -F or -File. To get a overview of all the parameters the user can enter -h or -help.

3.1 From the terminal

-N Specify a positive number that will represent the size of the population, i.e: number of individuals per population. `DEFAULT: 100`

-G Specify a positive number that will represent the duration of the simulation, i.e: the number of generations. `DEFAULT: 10`

-A Specify a positive number that will represent the number of alleles in the population.

-f Specify a floating number between 0 and 1 that will represent the list of the initials frequencies of the alleles of the population. `NOTE:` there must be as many frequencies as there are alleles. The sum of all the frequencies must be equal to zero.

-R Specify a positive number that represent the number of time the simulation is repeat with the same parameters. `DEFAULT: 3`

-P Specify a boolean that will indicate if the output must be in a file.
-T Specify a boolean that will indicate if the output must be in terminal.
`NOTE:` by default the output will be in a terminal. It can be print in terminal, in file and in both of it. But it must be print somewhere.

In the case, the user does not want to work with fasta file, he must enter this parameters in the terminal. We put default values for each parameters, expect for the number of alleles **-A** and the frequencies **-f**, so the user have to enter at least the number of alleles and the frequencies.

3.2 From a Fasta File

-N As in the section ***Fom the terminal*** above.
-G As in the section ***Fom the terminal*** above.
-R As in the section ***Fom the terminal*** above.
-P As in the section ***Fom the terminal*** above.
-T As in the section ***Fom the terminal*** above.

-F Specify a string that represent the name of the fasta file.
`NOTE:` the fasta file must contain nucleotides sequences composed of A, T, C, G or N. In this last case, N will be replace randomly by another nucleotide. The other lines (chromosome number) have to start with "`>`" symbol.
Example:

```
>chr11
CACTGTGTNTGATGATCAACAAAAACCTGGGGGGGGTAGTAGTAGTCC
>chr7
AAATGGTGCGTGATGCCCCCCCCCCCCCNCCTTGTGAAAA
```

-m Specify a positive number that will represent the list of markers, i.e: the position of the nucleotides of each allele in the fasta file.
-M Specify a string that will represent the migration type, i.e: can be star, ring or complete.

We put default values for each parameters, expect for the name of the fasta file **-F** and the markers **-m**, so the user have to provide at least the name of the fasta file and the markers corresponding to read it.

4 Exemples

4.1 Commands

- Execution of the program with the bare minimum of parameters in the terminal (whitout fasta file):
  `./WrightFisher -A 4 -f 0,1 - f 0,2 -f 03, -f 0,4`
  It will run by default a simulation 10 times of 3 same populations with 100 individuals and 4 alleles of frequencies 0.1, 0.2, 0.3, 0.4 (sum equal 1) and print the output in the terminal.

- Execution of the program with all parameters that the user can control in the terminal (without fasta file):
  `./WrightFisher -N 50 -G 150 -A 2 -f 0,882 - f 0,118 -R 2 -P 1 -T 0`
  It will run a simulation 150 times of 2 same populations with 50 indivuduals and 2 alleles of frequencies 0.882 and 0.118 (sum e qual 1) and print only in file name "test.txt", that you can find in the folder build.

- Execution of the program with the bare minimum of parameters when we read a fasta file: `./WrightFisher -F ../test.fa -m 2 -m 4 -m 6 -m 8`
  It will run by default a simulation 10 times of 3 same populations with 100 individuals and will read the file *test.fa*, which is above the folder build.

- Execution of the program with the extension migration: `./WrightFisher -F` a completer

4.2 Read Output

0	0.1|0.2|0.3|0.4	0.1|0.2|0.3|0.4	0.1|0.2|0.3|0.4
1	0.096|0.21|0.274|0.42	0.087|0.206|0.298|0.409	0.098|0.207|0.311|0.384
2	0.075|0.208|0.279|0.438	0.086|0.208|0.307|0.399	0.087|0.214|0.307|0.392
3	0.067|0.2|0.284|0.449	0.074|0.193|0.333|0.4	0.096|0.215|0.288|0.401
4	0.067|0.208|0.287|0.438	0.088|0.196|0.311|0.405	0.106|0.222|0.279|0.393
	ACG|ACA|ATA|ATG	ACT|ACA|AGA|GCA	ACG|AGA|AGT|AAA

Each line corresponds to one generation, the first column is the generation number (i.e. time), each subsequent column represents a replica of the simulation where each is represented by the frequencies of the alleles in the population (separated by '|'). If a fasta file was provided as input, the last line indicates the calculated alleles.