**README for Sim generations population genetics toolkit**

Royal Truman

Download programs from: <https://github.com/RoyalTr/Sim_generations>

Contents

[Introduction 1](#_Toc211507273)

[Simulation Process 1](#_Toc211507274)

[Content of input file in\_2\_haplos.txt 1](#_Toc211507275)

[Output files generated using fix\_2\_haplos.py as an example 2](#_Toc211507276)

[Technical details 2](#_Toc211507277)

# Introduction

This collection of python program are built a round Wright-Fischer forward simulations of population genetics. Diploid organism are assumed along with Mendelian segregation of chromosomes under random mating throughout the population.

The most common genotype fitness model is assumed:

*wAA* = 1 + *s*, *wAa* = 1 + *hs*, *waa* = 1,

where *A* and *a* are alternative haplotypes at the same locus.

The usual goal of the simulations is to calculate the fixation probability of A/a and B/b alleles, and the number of generations required. The programs can simulate assumptions used in evolutionary and young earth creationist models, specifically regarding initial allele proportions and selection coefficients.

**Population Growth**: Population size is fixed (*r* = 0) or increases / decreases according to the \*\*discrete Beverton-Holt model\*\*.

**Fitness Calculation**: The average fitness of the population is recalculated every generation, based on the relative fitness values of the three genotypes (AA, Aa, and aa) and their current proportions within the population.

# Simulation Process

The programs can run independent simulations (controlled by variable Repetitions in config file) for various user-defined scenarios specified in simulation input files. Each scenario is configured using parameters such as the following for fix\_2\_haplos.py.

## Content of input file in\_2\_haplos.txt

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ni | r | K | s\_A | s\_B | p\_A\_i | p\_B\_i | h\_A | h\_B | attempts |
| 10 | 0.04 | 10,000 | **0** | 0.001 | 0.01 | 0.01 | 0.5 | 0 | 1,000 |
| 10 | 0.04 | 10,000 | **0.02** | 0.001 | 0.01 | 0.01 | 0.5 | 0 | 1,000 |
| 10 | 0.04 | 10,000 | **0.04** | 0.001 | 0.01 | 0.01 | 0.5 | 0 | 1,000 |
| 10 | 0.04 | 10,000 | **0.06** | 0.001 | 0.01 | 0.01 | 0.5 | 0 | 1,000 |
| 10 | 0.04 | 10,000 | **0.08** | 0.001 | 0.01 | 0.01 | 0.5 | 0 | 1,000 |
| 10 | 0.04 | 10,000 | **0.1** | 0.001 | 0.01 | 0.01 | 0.5 | 0 | 1,000 |

*Ni*: Initial population size - r: Population growth rate

*r*: Growth rate

*K*: Carrying capacity, the maximum population inthat environment

*s\_A*: Selection coefficient for haplotype *A*

*s\_B*: Selection coefficient for haplotype *B*

*p\_A\_i*: Initial frequency of haplotype *A*

*p\_B\_i*: Initial frequency of haplotype *B*

*h\_A*: Dominance coefficient for haplotype *A*

*h\_B*: Dominance coefficient for haplotype *B*

Attempts: Number of simulations attempts to run x Repetitions

For each simulation attempt, the program iteratively calculates the allele frequencies, genotype frequencies, and mean population fitness. It then uses a binomial distribution to model random genetic drift and determine the allele frequencies in the next generation. The simulation continues until one of the alleles fixes (reaches a frequency of 0 or effectively 1).

## Output files generated using fix\_2\_haplos.py as an example

out\_2\_haplos.txt and out\_2\_haplos\_avg.txt contain key statistics from the simulations. Values stored include the probability of fixation for both alleles, the average number of generations to fixation, and the standard deviations for these metrics.

In addition, if the parameter document\_results\_every\_generation is set to true in config\_2\_haplos\_v1.yaml, then key values for each generation are stored in out\_2\_haplos\_per\_gen.txt. These provide the source input data for create\_plots\_ALL\_simul\_2\_haplos.py and aver\_per\_sim\_2\_haplos.py.

# Technical details

The program leverages multiprocessing to run simulations in parallel, optimizing execution time. The value of Repetitions in config\_2\_haplos\_v1.yaml specifies the maximum number of workers that will be running concurrently.