**Programming process steps in aver\_per\_sim\_1\_haplo.py**

Royal Truman 22.10.2025

Contents

[Files linked to *aver\_per\_sim\_1\_haplo.py* 1](#_Toc212062128)

[Overview of processing steps 3](#_Toc212062129)

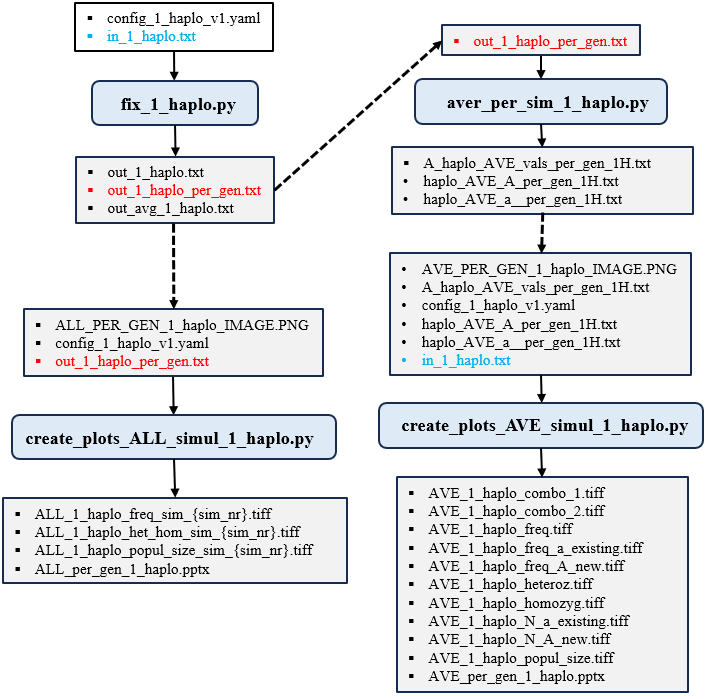
[Content of file haplo\_AVE\_A\_per\_gen\_1H.txt 5](#_Toc212062130)

[Content of file haplo\_AVE\_a\_\_per\_gen\_1H.txt 6](#_Toc212062131)

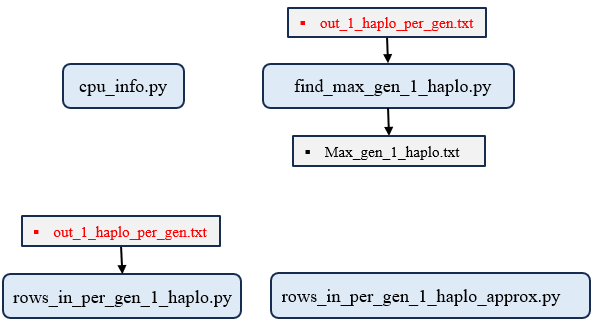
[Content of file A\_haplo\_AVE\_vals\_per\_gen\_1H.txt 7](#_Toc212062132)

# Files linked to *aver\_per\_sim\_1\_haplo.py*

Figure 1 shows the file dependencies of python program *aver\_per\_sim\_1\_haplo.py* and figure 2 the dependencies of various tools associated with this program.



**Figure 1**. Input and output file dependencies for python program *aver\_per\_sim\_1\_haplo.py*.



**Figure 2**. Input and output file dependencies for additional tools.

# Overview of processing steps

1. Import necessary Python modules including csv, os, time, concurrent.futures, multiprocessing, and tempfile.

2. Prevent system sleep on Windows using Windows API calls.

3. Define a function process\_roll\_forward\_and\_average\_streaming that computes time-averaged population statistics across simulation runs up to the fixation or last generation.

4. Define a specialized function process\_roll\_forward\_and\_average\_streaming\_fix\_A that processes only runs where allele A fixed (freq\_A ≈ 1.0), always including generation 0.

5. Define a specialized function process\_roll\_forward\_and\_average\_streaming\_fix\_a that processes only runs where allele a fixed (freq\_a ≈ 1.0), always including generation 0.

6. Define a worker function process\_one\_sim\_nr that reads per-generation data for a single SimNr from the input file.

7. In process\_one\_sim\_nr, parse and filter input rows belonging to the target SimNr using early exit for sorted data.

8. Convert field values to appropriate types (int or float) and skip malformed rows.

9. If valid data exists, compute averaged time-series for heterozygosity loss using process\_roll\_forward\_and\_average\_streaming.

10. Write the averaged results for heterozygosity to a temporary file named by SimNr.

11. Compute and write separate averaged time-series for runs where allele A fixed to a second temporary file.

12. Compute and write separate averaged time-series for runs where allele a fixed to a third temporary file.

13. Return the SimNr and success status to the main process.

14. Define a helper function get\_sim\_nrs that scans the input file to extract all unique SimNr values while checking for proper sort order.

15. In the main function, record the start time and verify that the input file "out\_1\_haplo\_per\_gen.txt" exists.

16. Initialize the primary output file "A\_haplo\_AVE\_vals\_per\_gen\_1H.txt" with a header row.

17. Scan the input file to collect and sort all SimNr values.

18. Create a temporary directory to store intermediate per-SimNr result files.

19. Prepare a list of arguments for parallel processing, one per SimNr.

20. Set the multiprocessing start method to 'spawn' for compatibility.

21. Launch parallel workers using ProcessPoolExecutor to process each SimNr independently.

22. As each worker completes, report its success or failure and track progress.

23. After all workers finish, concatenate all temporary heterozygosity result files into the primary output file.

24. Initialize and write the header for the allele-A fixation output file "haplo\_AVE\_A\_per\_gen\_1H.txt".

25. Append all temporary A-fixation result files into this output file.

26. Initialize and write the header for the allele-a fixation output file "haplo\_AVE\_a\_\_per\_gen\_1H.txt".

27. Append all temporary a-fixation result files into this output file.

28. Print the names of the three output files.

29. Print the total program execution time.

The program aver\_per\_sim\_1\_haplo.py generates 3 output data files .

# Content of file haplo\_AVE\_A\_per\_gen\_1H.txt

The input file *out\_1\_haplo\_per\_gen.txt* (Table 1) generated by program *fix\_1\_haplo.py* contains time-series data from multiple stochastic simulation runs. The number of runs is specified by *attempts* and *reps*.

**Table 1**. File out\_1\_haplo\_per\_gen.txt generated by fix\_1\_haplo.py.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sim Nr | Rep | att | … | gen | N | freq\_A | freq\_Aa | freq\_a | homoz |  | hetero (Excel) | homoz (Excel) |
| 1 | 1 | 1 |  | 0 | 6 | 0.5 | 0.50000 | 0.50000 | 0.50000 |  | 0.50000 | 0.50000 |
| 1 | 1 | 1 |  | 1 | 7 | 0.285714 | 0.40816 | 0.71429 | 0.59184 |  | 0.40816 | 0.59184 |
| 1 | 1 | 1 |  | 2 | 8 | 0.5 | 0.50000 | 0.50000 | 0.50000 |  | 0.50000 | 0.50000 |
| 1 | 1 | 1 |  | 3 | 9 | 0.388889 | 0.47531 | 0.61111 | 0.52469 |  | 0.47531 | 0.52469 |
| 1 | 1 | 1 |  | 4 | 10 | 0.4 | 0.48000 | 0.60000 | 0.52000 |  | 0.48000 | 0.52000 |
| 1 | 1 | 1 |  | 5 | 11 | 0.545455 | 0.49587 | 0.45455 | 0.50413 |  | 0.49587 | 0.50413 |
| 1 | 1 | 1 |  | 6 | 12 | 0.583333 | 0.48611 | 0.41667 | 0.51389 |  | 0.48611 | 0.51389 |
| 1 | 1 | 1 |  | 7 | 13 | 0.653846 | 0.45266 | 0.34615 | 0.54734 |  | 0.45266 | 0.54734 |
| 1 | 1 | 1 |  | 8 | 14 | 0.714286 | 0.40816 | 0.28571 | 0.59184 |  | 0.40816 | 0.59184 |
| 1 | 1 | 1 |  | 9 | 16 | 0.875 | 0.21875 | 0.12500 | 0.78125 |  | 0.21875 | 0.78125 |
| 1 | 1 | 1 |  | 10 | 18 | 0.861111 | 0.23920 | 0.13889 | 0.76080 |  | 0.23920 | 0.76080 |

The content of *out\_1\_haplo\_per\_gen.txt* is used by aver\_per\_sim\_1\_haplo.py to produce output file *haplo\_AVE\_A\_per\_gen\_1H.txt* (Table 2).

**Table 2**. File *haplo\_AVE\_A\_per\_gen\_1H.txt*, computed from the content of *out\_1\_haplo\_per\_gen.txt* shown in table 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sim Nr** | **Eff Gen** | **N** | **freq\_A** |
| 1 | 0 | 6 | 0.5 |
| 1 | 1 | 6.604 | 0.530979 |
| 1 | 2 | 7.266 | 0.559064 |
| 1 | 2.994 | 7.989 | 0.583485 |
| 1 | 3.972 | 8.766 | 0.606233 |
| 1 | 4.928 | 9.601 | 0.626904 |
| 1 | 5.857 | 10.492 | 0.645241 |

Each run in *out\_1\_haplo\_per\_gen.txt* is uniquely identified by (Rep, attempt), with only gen changing.

For a given SimNr, the program:

1. Selects only those runs (i.e., (Rep, attempt) pairs) in which **allele A fixes**, defined as freq\_A ≈ 1.0 (within 1e⁻¹⁰) in the last recorded generation.

2. Always includes generation.

3. For each generation *g* from 0 to the maximum fixation generation among the selected runs:

- It retrieves the value at *g* if available; if *g* exceeds the run’s length, it holds the final (fixed) values.

- It then computes column-wise averages across all selected runs:

- EffGen: mean of the effective generation index per run (min(g, fixation\_gen)).

- N: mean population size.

- freq\_A: mean allele A frequency.

Thus, the output reflects generation-wise averages **only over runs where A fixed**.

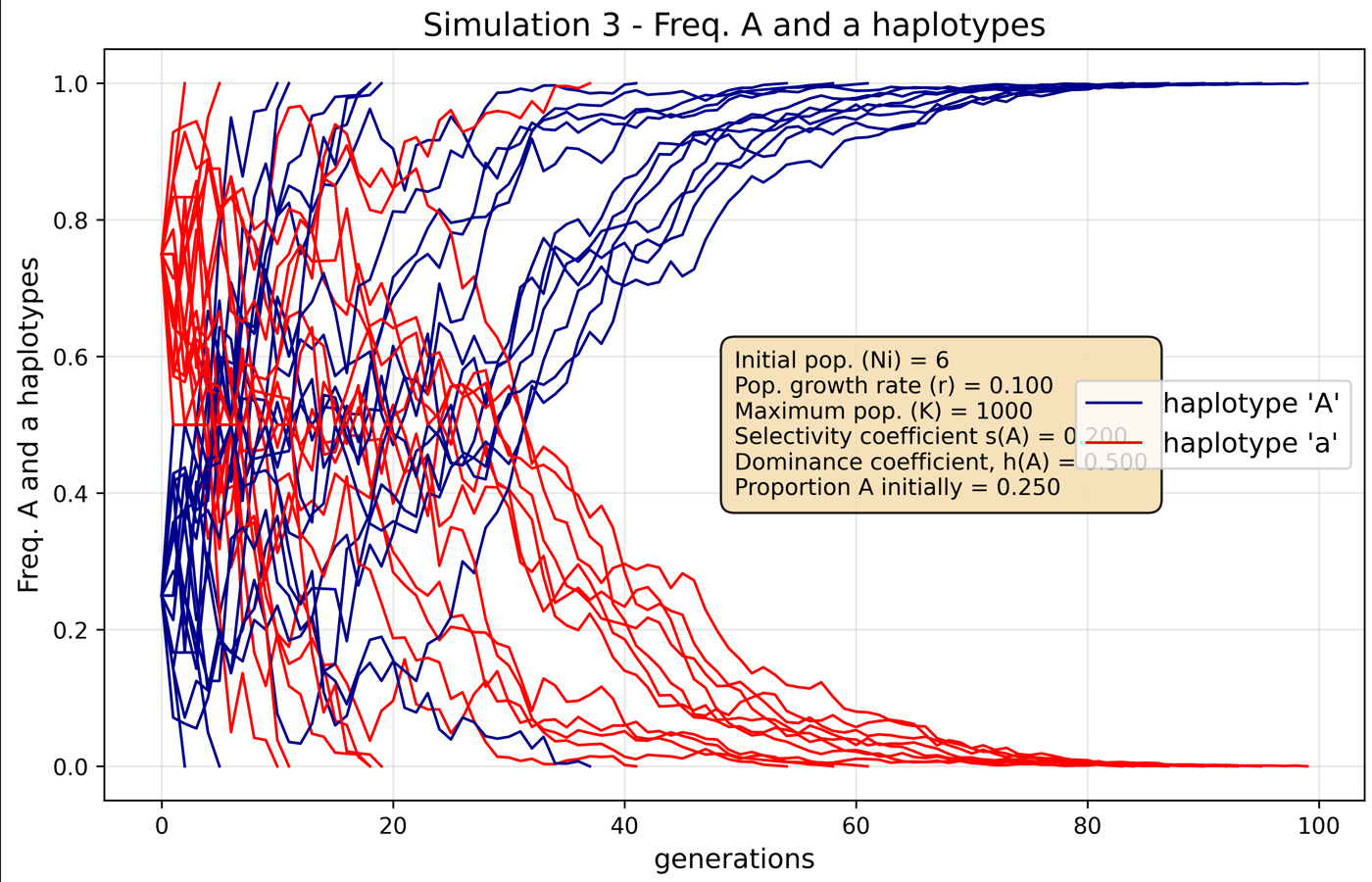
# Content of file haplo\_AVE\_a\_\_per\_gen\_1H.txt

Column freq\_a is used in an analogous way as just described for freq\_A above to create the content of file *haplo AVE\_a\_\_per\_gen\_1H.txt.*

**Table 3**. File *haplo\_AVE\_a\_\_per\_gen\_1H.txt*, computed from the content of *out\_1\_haplo\_per\_gen.txt* shown in table 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sim Nr** | **Eff Gen** | **N** | **freq\_a** |
| 1 | 0 | 6 | 0.5 |
| 1 | 1 | 6.499 | 0.605407 |
| 1 | 1.999 | 7.064 | 0.689671 |
| 1 | 2.974 | 7.703 | 0.759608 |
| 1 | 3.88 | 8.367 | 0.814656 |
| 1 | 4.692 | 9.025 | 0.856559 |

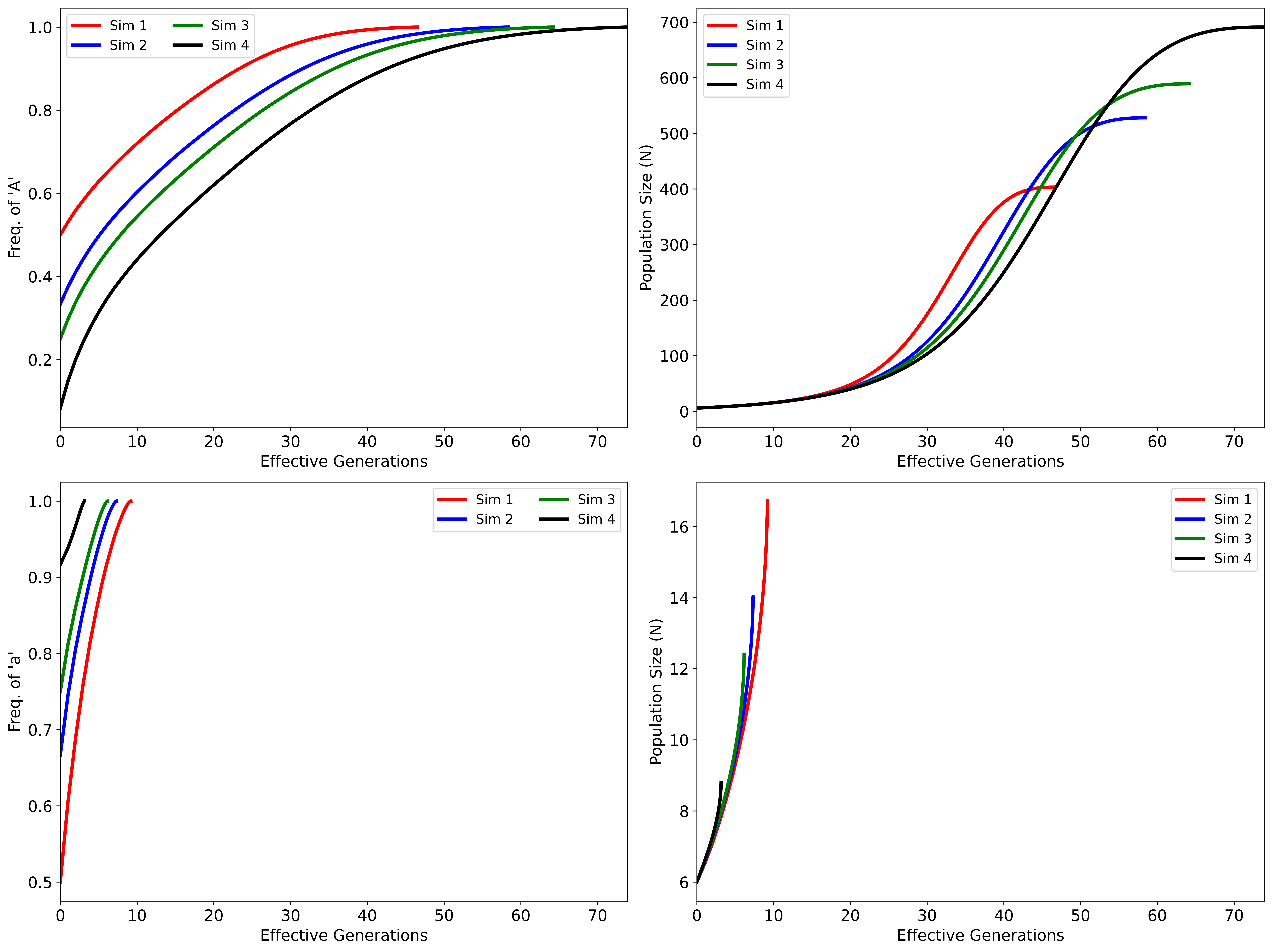
What is the biological purpose of Table 2 and 3? Despite *A* having a positive selectivity coefficient, there is a small probability that *a* has fixed anyway, especially when present in a small founder group, due to genetic drift. If a deleterious allele is nevertheless found fixed in a population, the data in Table 3 permits the *freq\_a* and *N* to be displayed over the averaged (effective) generations retroactively. Figure 3 illustrates how one of the deleterious runs for allele *a* (red lines) ended up fixing. Therefore, only those runs that ended up fixing should be taken into account retroactively to estimate the beharior of *freq\_a* and *N* over time until fixed. An alternative view would be to take all the runs into account, but if under strong selection it is extremely unlikely the deleterious allele would have fixed, the retroactive assumption is more plausible, which implies that random drift played a decisive role in determining which allele fixed.



**Figure 3.** Plot generated by create\_plots\_ALL\_simul\_1\_haplo.py to illustrate how a minority of runs show the deleterious allele *a* fixing (due to random drift dominating over selection).

# Content of file A\_haplo\_AVE\_vals\_per\_gen\_1H.txt

The content of file *A\_haplo\_AVE\_vals\_per\_gen\_1H.txt* shown in Table 4 is also computed from the input file *out\_1\_haplo\_per\_gen.txt*.



**Figure 4**. Average frequencies of 'A', and 'a' haplotypes, and average population size vs effective generations. (Based on only alleles *A* or *a* destined to fix). Data for images from files haplo\_AVE\_A\_per\_gen\_1H.txt and haplo\_AVE\_a\_\_per\_gen\_1H.txt.

**Table 4**. File *A\_haplo\_AVE\_vals\_per\_gen\_1H.txt*, computed from the content of *out\_1\_haplo\_per\_gen.txt* shown in table 1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sim Nr** | **Eff Gen** | **N** | **freq A** | **freq a** | **hetero** | **homoz** |
| 1 | 0 | 6 | 0.50000 | 0.50000 | 0.50000 | 0.50000 |
| 1 | 1 | 6.598 | 0.52262 | 0.47738 | 0.46083 | 0.53917 |
| 1 | 2 | 7.253 | 0.54381 | 0.45619 | 0.42658 | 0.57342 |
| 1 | 2.992 | 7.972 | 0.56245 | 0.43755 | 0.39700 | 0.60300 |
| 1 | 3.967 | 8.742 | 0.58043 | 0.41957 | 0.37094 | 0.62906 |
| 1 | 4.913 | 9.566 | 0.59726 | 0.40274 | 0.34848 | 0.65152 |
| 1 | 5.83 | 10.442 | 0.61238 | 0.38762 | 0.32809 | 0.67191 |
| 1 | 6.715 | 11.373 | 0.62710 | 0.37290 | 0.31029 | 0.68971 |
| 1 | 7.568 | 12.358 | 0.64121 | 0.35879 | 0.29442 | 0.70558 |
| 1 | 8.391 | 13.402 | 0.65451 | 0.34549 | 0.27992 | 0.72008 |

For each SimNr, the program:

1. Includes all runs (i.e., all `(Rep, attempt)` pairs), regardless of fixation outcome.

2. Defines a fixation generation for each run as the first generation where heterozygosity (`freq\_Aa`) drops to zero (within 1e⁻¹⁰); if this never occurs, the last recorded generation is used.

3. For each generation g from 0 to the maximum fixation generation across all runs:

- For each run, it uses the actual values at *g* if *g* ≤ fixation generation; otherwise, it holds the final values constant (i.e., “rolls forward” the state at fixation).

- It then averages the values across all runs:

- *EffGen*: mean of the effective generation index per run (min(*g*, run’s fixation\_gen)).

- *N*: mean population size.

- *freq\_A*, *freq\_a*, *freq\_Aa*: mean allele and genotype frequencies.

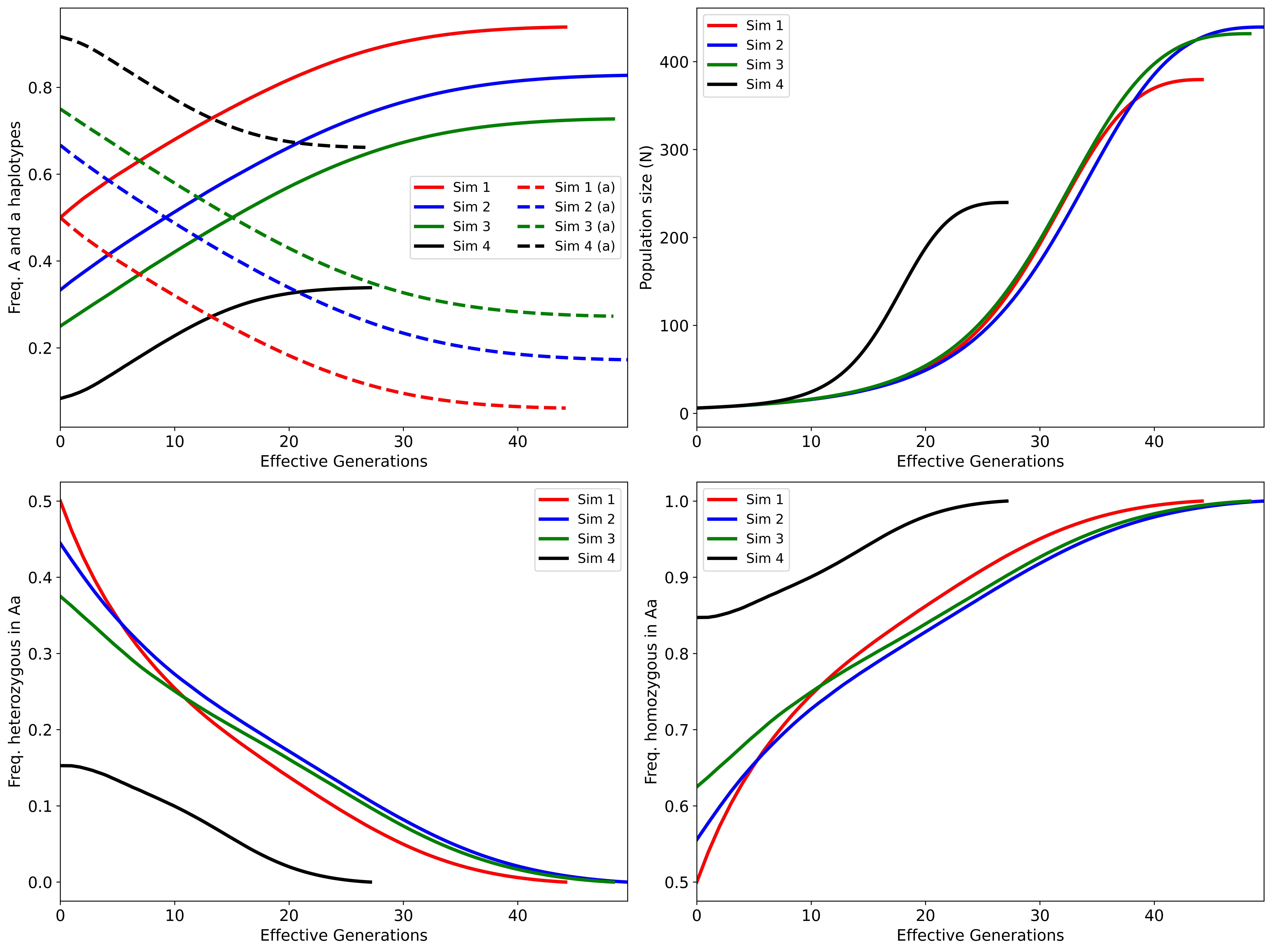
- *hetero* = mean of freq\_Aa (directly averaged, not computed from freq\_A and freq\_a for that row as 2(freq\_A)(freq\_a).

- *homoz* = 1.0 − hetero. This is only true for the case of 1 allele, *A/a*.

Thus, this file provides unconditional generation-wise averages over all simulation runs for each SimNr, found in file out\_1\_haplo\_per\_gen.txt.

The results in Table 4 correspond to expected or predicted outcomes based on the parameters used generate the simulated results per generation.

It is import to reiterate that the column hetero is the average hetero for all the runs (Rep,attempt combination) at a generation. If heterozygosity occurred for a run it must be recorded as such. Using average values of freq\_A and freq\_a to calculate hetero = 2(freq\_A)(freq\_a) would be wrong.

****

**Figure 5**. Average frequencies of '*A*', and '*a*' haplotypes, and average population size vs effective generations. (Based on all allele *A* and *a* fates). Data for images from file *A\_haplo\_AVE\_vals\_per\_gen\_1H.txt*.