**Program Documentation – Aver\_each\_gen\_per\_attempt\_and\_sim (1 gene)**

Royal Truman

This program is designed to process raw simulation data, calculate averages at two different levels of granularity, and save the results to new, structured files. The raw simulation data file can easily be much too large to be processed directly with tools like Excel, so aggregated results are calculated here.

Program Overview

The primary function of this script is to analyze genetic simulation data from a specified input file. It performs two main tasks:

1. Averaging per Attempt: It calculates the average values for key population metrics (e.g., population size, allele frequencies) for each generation within a specific attempt. An attempt is a set of replicate runs (Rep) for the same simulation parameters.
2. Averaging per Simulation: It then takes the attempt-level averages and computes a final set of averages for each generation within a full simulation. A simulation is a complete set of attempts with identical initial parameters.

The program handles potential inconsistencies in the data, such as runs that do not reach the final generation, by carrying forward the last recorded values. The use of the numpy library allows for efficient numerical calculations.

Function Breakdown

parse\_header(header\_line)

This function takes the first line of the input file and creates a dictionary that maps each column name to its corresponding index. This makes the program more robust, as it doesn't rely on hard-coded column positions.

load\_data()

This is the data ingestion function. It performs the following steps:

* Reads the input file and uses parse\_header to get column indices.
* Initializes two dictionaries:
  + raw\_data: Stores the raw data for each unique run, identified by the tuple (SimNr, attempt, Rep). Each entry is a list of tuples, where each tuple contains the generation number and a list of the corresponding frequency data.
  + simul\_param: Stores the simulation parameters, identified by the tuple (SimNr, attempt). This dictionary is used to retrieve metadata for each simulation run.
* It iterates through each line of the input file, parsing the relevant numerical values and populating the raw\_data and simul\_param dictionaries.

complete\_and\_average\_by\_generation(raw\_data, simul\_param)

This is the first of two core averaging functions. It takes the raw data and simulation parameters and calculates averages for each attempt.

* It first groups the data from raw\_data by (SimNr, attempt).
* For each group, it determines the maximum number of generations recorded across all replicates (Rep).
* It then iterates through each generation from 0 up to this maximum. For any replicate run that ended early, it fills in the missing generations with the values from the last recorded generation of that run.
* Finally, it uses numpy.mean to calculate the average values for N, freq\_A, freq\_Aa, freq\_a, and pan\_homoz (pan homozygosity) across all replicates for that generation. Note that for one gene having two alleles A/a, freq\_Aa is the pan heterozytosity.
* It returns a list of rows, each containing the metadata for an attempt, the generation number, and the calculated averages. The metadata are simulation parameters specific in input\_data.txt.

compute\_per\_simulation\_averages(attempt\_rows)

This function performs the second level of averaging, consolidating the attempt-level results into simulation-level results.

* It takes the output from complete\_and\_average\_by\_generation as its input.
* It groups the data by SimNr, the simulation number.
* Similar to the previous function, it finds the maximum generation for each simulation and fills in missing generations with the last known average values from the attempts.
* It calculates the final averages for each generation by averaging across all the attempts within a given simulation.
* It returns a list of rows, each representing a single simulation and a single generation, with the final averaged values.

write\_attempt\_averages(rows) & write\_simulation\_averages(rows)

These two functions handle the output. They take a list of data rows, format a header string, and write the data to the specified output files (aver\_each\_gen\_per\_attempt.txt and aver\_each\_gen\_per\_simulation.txt), separating each value with a semicolon. They also handle the specific formatting of decimal places for certain columns.

main()

This is the main control flow of the program. It orchestrates the entire process:

1. Calls load\_data() to read the raw input file.
2. Calls complete\_and\_average\_by\_generation() to compute attempt-level averages.
3. Calls write\_attempt\_averages() to save the attempt-level results.
4. Calls compute\_per\_simulation\_averages() to compute simulation-level averages.
5. Calls write\_simulation\_averages() to save the final simulation-level results.