**Program Documentation – Aver\_each\_gen\_per\_attempt\_and\_sim (2 genes)**

Royal Truman

**Purpose:**  
The program processes simulation results stored in results\_data\_per\_generation.txt and produces two summary files:

1. **aver\_each\_gen\_per\_attempt.txt** – averages per generation **within** each attempt of a simulation.
2. **aver\_each\_gen\_per\_simulation.txt** – averages per generation **across all attempts** of a simulation.

This tool is designed for post-processing population genetics simulations, where multiple replicates (Rep) and attempts are run for each simulation number (SimNr). File results\_data\_per\_generation.txt can be much too large to process with tools like Excel.

**1. Input Data Format**

**Input file:** results\_data\_per\_generation.txt  
Format: semicolon-delimited text file with a header row.  
Key columns used:

* **Identifiers:**
  + SimNr – simulation ID
  + attempt – attempt number within a simulation
  + Rep – replicate number within an attempt
  + generation – generation index
* **Population metrics:**
  + N – population size (integer values)
  + Frequencies: freq\_A, freq\_Aa, freq\_a, freq\_B, freq\_Bb, freq\_b.

where A/a anbd B/b are alleles

* + Diversity metrics: pan\_heteroz, pan\_homoz
* **Simulation parameters:** Ni, r, K, s\_A, h\_A, p\_A\_i, s\_B, h\_B, p\_B\_i, attempts

These parameters are specific in file input\_data.txt.

**2. Processing Steps**

The program runs in **two main stages**:

1. **Average per generation per attempt**
2. **Average per generation per simulation**

**Stage 1 – Per-Attempt Averages**

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

**Logic:**

1. Group all replicate data (Rep) by (SimNr, attempt).
2. For each replicate:
   * Sort entries by generation.
   * Record last generation’s values (to fill missing data later).
3. Determine the maximum generation reached among replicates for the attempt.
4. For each generation:
   * If a replicate has no data for that generation, forward-fill with its last available values.
   * Collect values from all replicates.
5. Compute:
   * Mean frequency values per generation.
   * Mean population size (N) per generation.
6. Append metadata and store results in a list for writing.

**Output file:** aver\_each\_gen\_per\_attempt.txt

**Header fields:**

SimNr;attempt;Ni;r;K;s\_A;h\_A;p\_A\_i;s\_B;h\_B;p\_B\_i;attempts;generation;N;

Ave\_freq\_A;Ave\_freq\_Aa;Ave\_freq\_a;Ave\_freq\_B;Ave\_freq\_Bb;Ave\_freq\_b;Ave\_pan\_heteroz;Ave\_pan\_homoz

**Stage 2 – Per-Simulation Averages**

**Function:** compute\_per\_simulation\_averages(attempt\_rows)

**Logic:**

1. Group attempt-level averages by SimNr.
2. Determine the maximum generation reached across all attempts in each simulation.
3. For each attempt:
   * Forward-fill missing generations with last available values.
4. For each simulation and generation:
   * Combine data from all attempts.
   * Compute mean frequency values and mean N.
5. Append metadata and store results.

**Output file:** aver\_each\_gen\_per\_simulation.txt

**Header fields:**

SimNr;Ni;r;K;s\_A;h\_A;p\_A\_i;s\_B;h\_B;p\_B\_i;attempts;generation;N;

Ave\_freq\_A;Ave\_freq\_Aa;Ave\_freq\_a;Ave\_freq\_B;Ave\_freq\_Bb;Ave\_freq\_b;Ave\_pan\_heteroz;Ave\_pan\_homoz

**3. Functions Overview**

| **Function** | **Purpose** |
| --- | --- |
| **load\_data()** | Reads the input file, extracts identifiers, population metrics, and simulation parameters. Organizes data by (SimNr, attempt, Rep). |
| **complete\_and\_average\_by\_generation(raw\_data, simul\_param)** | Computes per-generation averages across replicates for each attempt. Handles missing generations via forward-fill. |
| **compute\_per\_simulation\_averages(attempt\_rows)** | Aggregates attempt-level averages into simulation-level averages per generation. Handles missing data across attempts. |
| **write\_attempt\_averages(rows)** | Writes per-attempt averages to aver\_each\_gen\_per\_attempt.txt. |
| **write\_simulation\_averages(rows)** | Writes per-simulation averages to aver\_each\_gen\_per\_simulation.txt. |
| **main()** | Orchestrates all steps: load → process per attempt → write → process per simulation → write. |

**4. Forward-Fill Logic**

Missing generations occur when some replicates or attempts end earlier.  
The program **fills missing generations** using:

* The **last available** values from that replicate or attempt.
* This ensures all datasets have equal generation lengths for averaging.

**5. Execution Flow**

1. Start timer.
2. **Load raw data** → load\_data()
3. **Per-attempt averages** → complete\_and\_average\_by\_generation()
4. Write aver\_each\_gen\_per\_attempt.txt
5. **Per-simulation averages** → compute\_per\_simulation\_averages()
6. Write aver\_each\_gen\_per\_simulation.txt
7. Print total execution time.

**6. Output Summary**

* **aver\_each\_gen\_per\_attempt.txt**  
  Averages across replicates **within** each (SimNr, attempt).
* **aver\_each\_gen\_per\_simulation.txt**  
  Averages across all attempts for each SimNr.

Both outputs preserve:

* Simulation metadata. This is necessary to ensure in the future the simulation parameters use to generate the data is known, and also to permit sorting of data for trend analysis.
* Generation index.
* Mean population size. For growing population sizes average vaues from the simulations are calculated.
* Mean frequencies and diversity metrics.