SIMPLE: a tool to simulate mutations in yeast populations

**SHORT DESCRIPTION:**

These R scripts (SIMPLE\_Neutral.R and SIMPLE\_Transient.R) simulates the appearance of 2 types of mutations in an exponentially growing yeast population starting from a single cell. The simulation can be performed either in the absence (null model with SIMPLE\_Neutral.R) or in the presence (refined model with SIMPLE\_Transient.R) of a subpopulation of transient mutator cells. For both models, the number of generations and the values of the two single mutation rates are chosen by the user. For the refined model, the size of the mutator subpopulation, the fold-change of the mutation rates and the duration of the mutator episode (in generations) are also defined by the user. The output provides the number of both single and double mutant cells in the final colony and the number of mutations that occurred during the colony development.

**INSTALLATION:**

No installation of additional library is needed. There are two scripts, one for the neural model called “SIMPLE\_Neutral.R” and one for the refined model called “SIMPLE\_Transient.R”. Place both script in the current working directory. If needed, you can set the working directory in R using he command:

* Setwd()

**LAUNCH THE SCRIPT (for both version):**

Prior any simulation, you have to manually enter the parameter values of the simulation in the 1st section the Rscript files entitled: “Simulation parameters”.

For both versions:

* Output: name used for the result files (output.verbose.txt and output.res.txt)
* r1: rate of the mutation A
* r2: rate of the mutation B
* g: number of generations starting from 1 cell (note that more than 27 generations will generate huge files that will require more than 16Gb of RAM)
* n: number of realizations (correspond to the number of colonies to simulate)

Only for the refined version:

* hyp: number of hypermutator cells in the population (for simulation without hypermutator cells, set this parameter to 1)
* mult: the fold change increase of mutation rates in the hypermutator subpopulation (for simulation without hypermutator cells, set this parameter to 1)
* ngen: number of consecutive hypermutator generations

**About system requirement**

Note that the simulation process requires mainly memory resources. The script was tested on a current generation computer (Core i7-9700 - 16Go RAM). With that configuration, each simulation uses up to 30% of one thread, but also generates up to 10Gb of temporary files. Even if they are deleted at the end of each simulation, this has to be considered if you want to launch several simulations in parallel.

**Windows environment**

If you want to simulate a lot of generation (e.g. 27), it might happen that R return a memory error. Then memory can be adjusted using the following command line:

* memory.limit(size = 10000)

**- Using R Console**

**-** Open R

**-** Use the File / Source menu to select the file “SIMPLE\_Neutral.R”

**- Using RStudio**

- Open Rstudio

- Using the File/Open File, open the file “SIMPLE\_Neutral.R”

**Linux environment (e.g. ubuntu)**

**-** Go into the folder containing the file “SIMPLE\_Neutral.R”

**-** The command to launch the script is:

> source(“SIMPLE\_Neutral.R”)

**DURING THE SIMULATION:**

Once you have entered all these parameters; the simulation will start. First there will be a summary of your parameters and then the results of each generation of each simulation, starting by indicating the current realization (“simulation cycle: X on Y”). Then the generation at which the transient mutators will appear is indicated ("transient mutators appears at generation: x"). Finally, the results for the sampling for each generation will be indicated, as follows: "generation: 1 res: 0 0 0 0 0 0". The res gives in order from the left to the right:

* The cumulative number of mutant A
* The cumulative number of mutant B
* The cumulative number of mutant A and B
* The cumulative number of mutation A
* The cumulative number of mutation B
* The cumulative number of mutation AB

**SAVING THE RESULTS:**

The results are saved automatically in the current working directory. If you are under windows, you have chosen it at the beginning of the process. Under Ubuntu, they will be stored in the same folder than the one containing the script. Note that 2 output files will be generated:

<output>.verbose.txt and <output>.res.txt.

The verbose.txt file is generated using the sink() function and thus will contain all the information that have been printed during the simulation. The res.txt file contains the number of mutants and mutations A, B and AB for each realization.