popSim User Manual

Melchior Zimmermann

March 8, 2017

Contents

1 Introduction

popSim is a population simulation tool based on Lotka-Volterra-like models ([1]), and implemented in C++. This user mannual was written to provide a simple guide to using this software, and getting to know its capabilites. This manual was not, however, designed to be an exhaustive documentation of its code (for that, see the documentation on github), nor a scientific analysis of its capabilities and/or application domains (for that, you can refer to my master's thesis). Please note that popSim is still currently being developed and maintained, and I hope to add some more functionalities, in regards to both the simulations it is capable of running as well as usability. If you haven't used it in a while, it might be a good idead to quickly review this guide, to see what (if any) changes/additions were made since.
popSim is open source software (`https://github.com/melchior-zimmermann/popSim`, GPL v3.0), and comes with no warranty whatsoever.

2 Technical notes

The resolution of the Lotka-Volterra-like equations used in popSim is done using the explicit Euler scheme. Calculatory complexity of the equations goes as

the square of the number of species ($O(n^2)$, for details of the equations used in popSim, see section 4), but other factors (such as multiple environments or evolution) can contribute signifcantly to the time needed to run a simulation.
Saves are made directly to disk, which means that interrupting simulations will not lead to loss of results obtained up to that point. However, this also means that the number of saves done per simulation is a big factor in execution speed (especially when saving results on a HDD).
Running 500 simulations with the parameters of the template file (27 species in total, three environments, evolution, environmental constant taken into account) will take approximately 30 hours on a i6700HQ Skylake processor using a single thread, whereas the same number of runs from a simple simulation (same number of species, one environment, no evolution, environmental constant not taken into account) takes about thirty minutes.
Thread safety of this program is not guaranteed, and parallelism is not natively supported.
popSim was written and tested on Arch Linux (www.archlinux.org).

3 Basic use

popSim takes at least one, and at most five arguments. The first argument (herafter called 'execution code') tells the program what we want it to do. Below is a list of all possible execution codes (as of 24.12.2016), as well as the arguments that are needed for each one.

- 0 : The program will create a template file containing simulation parameters. If no path is specified as the second argument, the file will be created in the same directory as the executable.
- 1 : The program will guide the user through the simulation parameters wizard, and launch a simulation with the given parameters. The second argument must be the path to the folder where simulation parameters/results are saved, and the third argument is the number of times the program will run the simulation (not to be confused with the number of steps for which a single simulation will run).
- 2 : The program will load an existing parameter file and run the simulation it defines. The second argument is the path to the save folder, the third argument is the number of runs it will do for that simulation, and the fourth argument is the path to the parameter file it will load.
- 3 : The program will load a simulation from a complete save, and re-run this simulation. The second argument is the path to the save folder, the third argument is the number of runs it will do for that simulation, and the fourth argument is the path to the complete save it will load. The fifth argument is the number of species it will add to the simulation (0 if you do not wish to add any species). Additional species will be created based on the values of the parameter file found in the same folder as the complete save. All simulation attributes not present in the complete save (such as the number of steps for which to run the simulation) will also

be loaded from this parameter file. If there are more than one simulation saved in the complete save file, the first one will be used.

Example usage: ./popSim 2 my/save/folder 100 my/simulation/parameters

The parameter file found in the folder ‘my/simulation/parameters’ will be used to generate 100 simulations, the results of which will be save in a newly created file in the folder ‘my/save/folder’. If there are files present in ‘my/save/folder’ with the same name as those used by popSim, they will be truncated before the start of the simulations (their content will be lost).
When more than one run of a particular simulation is made, all results are appended in the same file.
When required, paths should be to the folders containing the files, and should not include the filename itself (for a more complete description of files and their content, see section 6). Both absolute and relative paths are supported.
Be aware that argument parsing is rudimentary and not guaranteed to catch all possible errors in supplied arguments.

3.1 Generating simulations from parameter files

Parameter files do not detail each attribute of the simulation, but rather, are used to stochastically generate values from given ranges. This means that two simulation generated using the same parameter file can (and very probably will) have different values in regards to e.g. species interactions (for a complete list of all simulation parameters and their use, see section 8). If you want to control all values of a simulation in a precise manner (e.g. to reproduce an existing food web), it is necessary to manually edit the save files (see section 5).
When multiple runs are made from the same parameter file, simulation parameters are generated independently for each run.

4 Equations

popSim uses a generalized version of the Lotka-Volterra equations ([1], but see [2]) with a logistic factor to avoid potentially explosive species growth. Change in density is calculated as:

$$\dot{x_i} = \delta \cdot x_i \cdot (\alpha_i \cdot (1 - (x_i/cc_i)^2) + \beta_i + \sum_{i \neq i} I_{ij} \cdot x_j) \qquad (1)$$

where

- x_i is the density of species i
- $\dot{x_i}$ is the variation in density of species i during a given time-step
- δ is the step size
- α_i is the growth rate of species i

- cc_i is the carrying capacity of species i
- β_i is the mortality rate of species i
- I_{ij} is the interaction between species i and j as seen by species i ($I_{ij} \neq I_{ji}$)
- x_j is the density of species j

When the environmental constant is taken into account, a positive value of $\dot{x_i}$ is affected as follows:

$$\dot{x'} = \dot{x} \cdot e^{-(ec-o)^2} \tag{2}$$

Where:

- $\dot{x}$ is the change in density before environmental effects are taken into account
- $\dot{x'}$ is the change in density after environmental effects are taken into account
- ec is the environmental constant
- o is the optimum of the species

When evolution is active, species undergo an evolutionary event in which they are discretized as individuals:

$$n_i = \text{round}(x_i \cdot \text{resolution}_i) \tag{3}$$

where:

- x_i is the density of species i
- resolution_i is the value of the resolution parameter for species i
- round() is a function that rounds to the nearest integer (from math.h)
- n_i is the number of individuals resulting from species i

Next, we calculate interaction values for each individual that follow a normal distribution centered around the values for the species. Thus, interaction-values for individuals are calculated as follows:

$$II_{ij} = \text{gaussian}(I_{ij}, \sigma^2) \tag{4}$$

where II_{ij} is the interaction value for the individual, I_{ij} is the interaction value for the species, and σ^2 is the evolution range of this interaction.

Of course, since we are now talking about individuals, we cannot simply calculate a change in density. Thus, we use a slight variation on the Lotka-Volterra equations to calculate the survival probability of each individual:

$$q_{i,t} = \delta \cdot \sum_{i \neq j} II_{ij} \cdot (x_j) \tag{5}$$

and:

$$P(0) = 1/2 - q_i/2 \tag{6}$$
$$P(2) = 1/2 + q_i/2 \tag{7}$$
$$P(1) = 0 \tag{8}$$

where:

- P(0) is the probability that a given individual dies without leaving any offspring
- P(2) is the probability that a given individual survives and leaves one offspring (with interaction values identical to that of the parent)
- P(1) is the probability that a given individual survives and leaves no offspring (always 0)

When there are multiple environments, migration is calculated as follows:

1. Each species is given a complete set of migration routes to and from each environment
2. Each species has a certain probability to migrate along a given route (mp_k, where k identifies the given route).
3. Species carrying capacity and species optimum, as well as the environmental constant of the current environment are used to calculate the effective carrying capacity as:

   $$\text{effectiveCc}_i = cc_i \cdot e^{-(ec-o_i)^2} \tag{9}$$

   where:

   - cc_i is the carrying capacity of species i
   - o_i is the optimum of species i (optimum value of the environmental constant for this species)
   - effectiveCc$_i$ is the effective carrying capacity of species i
4. The effective carrying capacity and species density are used to calculate the effect migration probability as:

   $$\text{effectiveMP}_k i = mp_k i \cdot x_i / \text{effectiveCc}_i \tag{10}$$

   where:

- x_i is the density of species i
- effectiveCc$_i$ is the effective carrying capacity of species i
- $mp_k i$ is the probability of species i migrating along route k
- effectiveMP$_k i$ is the effective probability of species i migrating along route k

5. The size (in density) of the part of the population migrating along route k (md_i) is calculated as:

$$md_i k = x_i \cdot x_i / \text{effectiveCc}_i \cdot ps_k i \tag{11}$$

where

- x_i is the density of species i
- effectiveCc$_i$ is the effective carrying capacity of species i
- $ps_k i$ is the size coefficient of a migration of species i along path k
- $md_i k$ is the density value of the part of species i migrating along path k

All characteristics (except density) of the migrating population are exactly the same as those of the parent population.

6. The density of the original population is updated to reflect density loss due to migration.

7. The migrating part of the species is integrated into its new environment in two possible ways:

(a) The species is not yet present in the new environment, in which case the migrant population will simply be added to the list of existing species.

(b) The species is already present in the environment. In this case, the incoming population will be absorbeb by the one already present, and all parameters will be modified by taking the average between the two populations weighted by their density.

For a more detailed analysis of these equations, please refer to my master's thesis.

5 Advanced use

The commands shown in section 3 allow you to determine parameters from which simulations are generated, but, as pointed out in section 3.1, this is done stochastically from the parameter values. Of course, such stochastic generation is not always desired. It is possible to define simulation parameters (or, more accurately, species and environment parameters) in more detail by editing complete saves by hand, and then reloading them. However, parsing of files is rudimentary, and changes in file layout may cause the program to crash while loading the file, or worse, run simulations with parameters other than those intended. The same holds true when loading simulation parameter files, where additional care has to be taken in how values are entered (the wizard implements some basic input validation, but the method that loads parameter files does not). Therefore, if your are editing said files by hand, it is highly recommended that you perform one (or several) short test runs, to make sure that the simulation parameters are as intended. For an overview of the different files generated by this program, see section 6.

Limitations you should be aware of include (but are not limited to):

- When defining a range, the lower bound must always be given first.
- Do NOT modify file layout (an extra line break is enough to throw the parser off).
- interSaveDiv and each how many steps density-values are saved must NOT be set to 0 (to avoid intermediary saves, set it to a number greater than the number of steps for which you run the simulation).
- For obvious reasons, values such as the number of species (or of environments) should never be negative.

6 Files

popSim generates a number of files when running a simulation. The list hereafter gives a short description of each of these files and its contents. Please note that filenames must be exact matches, or the pogram will not recognize them (filenames cannot be specified by the user).

- simParams : This file contains the simulation parameters (as described in section 8).
- rawSave : This files contains the results of simulations, as well as intermediary values for density and species interactions (if requested).
- completeSaveS: A complete save of all parameters made at the beginning of a simulation (used to reload simulations).
- CompleteSaveE: Same as above, but for the end of a simulation.

When a given simulation is run multiple times (runs ≥ 1), saves are appended into the same file (e.g. for 100 runs of a given simulation, the results of all 100 runs will be in the same file).

I believe the contents of completeSave (E and S) and rawSave to be pretty self-explanatory, but if there are people having trouble interpreting them, let me know, and I'll add a section detailing them here.

7 basicStats.py

The python file basicStats.py uses the results in 'rawSave' to generate basic statistics about a number of runs of a simulation (mean final density, mean number of species that died per simulaiton, etc.). This program is mainly useful to get a quick overview of results, and does not pretend to replace detailed statistical analysis by more complete software. basicStats.py takes three arguments:

- The path to the folder where the results of the simulations are saved.
- The number of runs that were done (if the number entered here does not match the actual number of runs in the save file, the program will either segfault, or create empty species).
- The number of added species (if no species where added, set to 0).

Example usage: python basicStats.py my/popSim/results 100 0

The above command will make statistics of the 100 simulations found in the file rawSave present in the folder 'my/popSim/results'.

8 Simulation parameters

This section details all simulation parameters (as of 24.12.2016) in the same order as they appear in the parameter file. Note that depending on the simulation, not all parameters are needed.

- Eco: Wether or not the environmental constant is taken into account ('y' if it is, 'n' otherwise).
- Species optimum Range: The uniform range from which species optimums are generated.
- Environment optimum Range: The uniform range from which the environmental constant value(s) are generated.
- Perturbation probability: The probability (at each step) of their being a change in the value of the environmental constant.
- Perturbation intensity range: The uniform range from which the change in the value of the environmental constant is drawn if there is a perturbation event.
- Evo: Wether or not there is evolution in this simulation ('y' if there is, 'n' otherwise).
- Species evo range Range: The uniform range from which the σ^2 parameter for interaction values of individuals in regards to those of the species are drawn.
- Species opt evo Range: Same as above for optimum.

- Gen time Range: The uniform range from which species generation time (each how many steps an eovlution event occurs) is drawn.
- Evo resolution range: The uniform range from which species resolution is drawn.
- Inter save Div: Each how many timesteps species interactions are saved (do NOT set to 0, set it to a number higher than the number of steps if you only want to save interaction values at the start/end of the simulation).
- Number species: Total number of species in the simulation.
- Multi: Wether or not this simulation contains multiple environments.
- Number of environments: The number of environments in this simulation.
- Number of species in env#n: How many species there are in environment n.
- Mig prob Range: The uniform range from which species migration probabilities are drawn.
- Mig size Range: The uniform range from which species migration size are drawn.
- Interaction range: The uniform range from which species interactions are drawn.
- Initial density range: The uniform range from which initial densities of species are drawn.
- Carryign capacity range: The uniform range from which species carrying capacities are drawn.
- Alpha range: The uniform range from which species alphas are drawn.
- Beta range: The uniform range from which species betas are drawn.
- Number of steps: The number of steps for which to run the simulation.
- Delta (stepsize): Delta parameter for the explicit Euler Scheme.
- Death threshold: Density threshold below wich species are considered extinct (species is set to 0).
- Each how many steps density-values are saved: self-explanatory.
- CompleteStart: Wether or not to make a complete save at the start of the simulation ('y' to make the save, 'n' otherwise).
- CompleteEnd: Wether or not to make a complete save at the end of the simulation ('y' to make the save, 'n' otherwise).

References

[1] AJ Lotka. Fluctuations in the abundance of species considered mathematically (with comment by v. volterra). *Nature*, 119:12–13, 1927.

[2] JD Murray. *Mathematical Biology*. Springer, 2002.