Metapop user documentation

Soularue J.P., Arnoux L., Thöni A. and Kremer A.

February 26, 2018

Contents

1 Introduction

1.1 Overview

Metapop simulates the evolution of meta-populations of monoecious plants accross heterogeneous landscapes. Based on quantitative genetics theory, Metapop is individual-based, genetically and spatially explicit, its structure embodies several interacting modules (Figure 1). Each simulation is parameterized through a configuration environment composed of textual files. The evolutionary scenario defined is simulated by the model which produces raw data. Finally, output modules process the raw data and enable the monitoring and the analysis of evolutionary changes at the level of quantitative traits, their underlying genes or neutral markers. Fstat and Genepop files of individual values can be generated, which allows the user to further process the data with other packages. A comprehensive description of the use of Metapop is given in the following parts. A synthesis text file *MTP_cheatsheet.txt* is available in the main directory for a quick help on basic commands.

1.2 License

1.3 Contact

Questions can be sent at jean-paul.soularue@inra.fr. Please mention the word "Metapop" in the subject of the email.

2 Getting started

2.1 Prerequisites for installation

Metapop has been developed in C++ and Lua. It was designed and tested for Linux-like platforms and particularly for clusters. In most cases, installation and execution on Microsoft Windows operating systems remain possible through Cygwin libraries. Before installation, be sure that (*i*) you have the permission to install the program on your system, (*ii*) the packages required to compile and execute C++ are already installed, (*iii*) R with *MCMCpack* module is installed.

2.2 Installation

First download the archive on the treepeace.fr website, section Tools. Unzip it and go into the resulting folder. The corresponding console commands are given below:

```
http://www.treepeace.fr/wp-content/uploads/MTP_2_2_1.zip
```

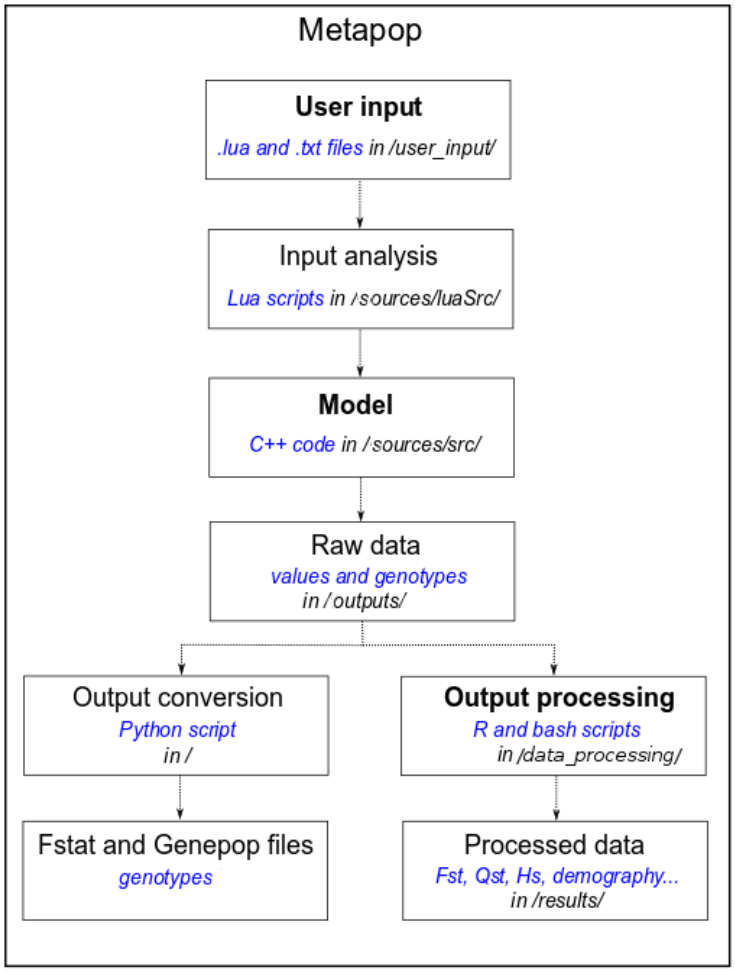

Figure 1: Structure of Metapop.

```
unzip DEMO_MTP_1_beta.zip
cd DEMO_MTP_1_beta
```

Compilation of the main source code can be done with the following commands. On Linux:

```
./installMP.sh linux
```

On Windows, with Cygwin installed:

```
./installMP.sh posix
```

2.3 Running a simulation

Simulations can be run from command line:

```
python launchMTP.py
```

A standard launch includes automatic management of simulation replicates and output post-processing. Settings related to the management of simulations (*e.g.* number of replicates, seeds...) can be defined in the file *simuParams.py*. The complete parameterization of the simulation is documented in section **Input**. It is also possible to run the program without handling replicates nor post-processing outputs with the following command:

```
sh scripts/runMTP.sh
```

2.4 Compilation

When Metapop is already installed but the code base has changed and needs to be compiled, run *compileMTP.sh* to update the binary.

2.5 Input

2.5.1 General structure

The model can be parameterized from the files located in *user_input* directory. Each *.lua* file is composed of sets of variables that are initialized by the user. *simuConf.lua* and *seed.lua* define general parameters of the simulations. */species/Si.lua* describes the attributes related to the species S_i. */traits/Ti.lua* defines the trait i composing the phenotypes of the individuals. Secondary *.txt* files can be used to describe the heterogeneity of the landscape simulated and the dispersal of pollen and seeds (see below).

2.5.2 Initialization modes

In most of the cases, the user simply assigns a single value to the variable of interest. Nonetheless, the composite variables which describe the spatial structure of the landscape can be set according to three modes. These composite variables come along with another variable called "source", which defines the initialization mode. Assuming a variable x, assigning "value" to the variable *source* means that the user is expected to affect a single value to x.

The value entered will be automatically replicated in all the cells of the landscape. Affecting "param" to the variable *source* automatically creates patterns of variation of the variable x throughout the landscape, for instance a gradient. This initialization mode often requires to set additional variables which define the direction or the magnitude of the change across the landscape. Lastly, specifying "userdata" requires the user to manually fill a matrix in an external file. For instance, taking as example the selection pressure exerted on a trait (detailed in the next section), the phenotypic optima assigned throughout the landscape can be set by modifying the variable *zoptParam* in the file *Ti.lua*, where i is an integer used as identifier: *traits/T1.lua*

```
zoptParam =
{
    -- Source of data, allowed values: userdata, value, param
    -- If source == "userdata": matrix zopt.txt directly filled
    by user
    -- If source == "value": a unique value is assigned to all
   cells, arg: zopt = number
    -- If source == "param": generation of a gradient according
    to one dimension, args: opposed = boolean, Kzopt = number
    source = "value",
    opposed = False,
    Kzopt = 10
}
```

Here, the "value" initialization mode is used, the keyword "opposed" and "Kzopt", useless under this initialization mode, are ignored. Assuming the landscape simulated is a 3x3 square, the file *user_input/zopt.txt* in this example is *zopt.txt*

```
10 10 10
10 10 10
10 10 10
```

2.5.3 Secondary files

The variables related to spatial heterogeneity, pollen and seed dispersal are associated with secondary *.txt* and *.lua* files (see section 4.1, figure 3) which essentially contain initialization matrices describing the spatial distribution of the values assigned across the landscape. These *.txt* files can be generated automatically or manually depending on the mode selected in the primary *.lua* files. For instance *e.txt* specifies the macro-environmental values associated with each cell of the landscape simulated, for each trait, at different points of the simulations (see below sections 3 and 4). *zopt.txt* and *sel_int.txt* parameterize the selection experienced by the individuals throughout the landscape, *patterns.txt* configures the dispersal of seeds and pollen. Besides, the generation of random seeds associated with each replicate of a scenario also generate a secondary file.

2.5.4 Syntax

Here are some basic notions useful for editing input *.lua* files:

- lists are delimited by { and }. Elements separated by ,.

```
traitsId = {"T1", "T2"}
```

- comments are defined after double dashes.

```
-- Genomesize is the number of locus times ploidy
```

- booleans are *true* and *false* without capital. *nil* is used for undefined values.

```
counterGrad = false
```

- strings can be defined using simple or double quotes.

```
fileType = "species"
```

More information about the parameterization of the model is available below in section Input

2.6 Output

2.6.1 Raw and processed output

Each simulation replicate generates raw and processed output data in */outputs/Ri* directory, where i designates the replicate ID. Each replicate directory contains 4 sub-directories:

- *MTP*: input files used to simulate the scenario.
- *genotypes*: genotypes of all individuals at the different steps of the simulation process. The genotypes are recurrently saved in *genes_Si_g_.csv*, where i is the species ID and g the generation number. By default, the genotypes are stored according to the following format :

```
nPop nIndiv alleleLoc1 [alleleLoc1] alleleLoc2...
```

- *pRocessed*: output files generated by the R scripts used for the automatic treatment of the raw output.

– *quantitative*: quantitative information (additive values, differentiation, demography, seed and pollen flow...) generated by the post-treatment of raw-data.

In addition, summary files are generated by the automatic treatment of the raw output in */outputs* directory. These files average the quantitative information produced in the replicates. An exhaustive description of the output files is given in section 5.

2.6.2 Genotypes conversion

Metapop regularly saves the genotypes of the individuals according to its own specific format in the directory *outputs/genotypes*. The script (*convertGenFiles.py*) can be used to convert these files into the Fstat or Genepop format. The user can use the files produced to further analyze the data simulated with other programs. The command required to run the script is *python convertGenFiles.py inputFile outputFile outputFormat*, with *outputFormat* = *G or F* depending whether the required format is Genepop or Fstat.

```
python convertGenFiles.py ./GENO.csv user_input/ind.txt
   filename g
```

2.7 Starting meta-population

Two modes of initialization of the starting meta-population are available. In the main case the starting meta-population is a meta-population at the mutation-migration-drift equilibrium generated automatically from the initial demography (N, population size) and gene flow (m, migration rate) specified in */species/Si.lua*. Under these assumptions, the stationary allelic frequencies in each population follow a multinomial Dirichlet distribution with parameters 4Nmq (where q is the allelic frequency vector in the overall metapopulation). Hence the Dirichlet distribution is used to generate the initial genotypic arrays in each population. This mode of initialization assumes that seeds and pollen are dispersed according to the Wright's Island migration model.
Alternatively the user can directly specify the genotypes of the individuals composing the initial meta-population in the file (/ind.txt). This can be done by initializing the variable *initDemo* of *species/Si.lua* as follows:

```
    initDemo = {
        source = "injection"
    }
```

Under this mode of initialization the genotypes should be in the Metapop format (each individual line have the following pattern: *popNumber indNumber loc1all1 loc1all2 loc2all1...*). It is however possible to convert existing Genepop or Fstat files into the Metapop format with the script *convertGenFiles.py* provided with Metapop. The script automatically detects the format of the input file (Genepop or Fstat) according to their header. Assuming a

genotype file named *GENO.csv* structured according to Fstat format in the main directory, the conversion can be done via the following command:

```
python convertGenFiles.py ./GENO.csv user_input/ind.txt I
```

Here the third parameter "I" is mandatory to specify the conversion from Genepop or Fstat format to Metapop format.

3 Model

Several initialization examples are proposed in this part, nonetheless exhaustive information about the parameterization of the model is provided in section 4.

3.1 Frame of the model

The model is composed of three parts. The first part is related to the definition of the trait(s) composing the phenotypes of the individuals simulated, the associated genetic architectures, the underlying genes and genome. The second part is related to the landscape simulated. Each cell composing the landscape is defined by the conditions of natural selection faced by the individuals (phenotypic optima Z_{opt} and intensity of selection ω), a macro-environmental effect (E) which possibly affects the expression of a plastic trait, and the demography of the initial meta-population of the species simulated (starting number of individual N and carrying capacity K). Metapop allows the user to define any pattern of variation in these values throughout the landscape. The third part of the model is the life cycle which defines the sequence of the evolutionary steps and processes undergone each generation by the individuals.

3.2 Phenotypic traits

The phenotypes of the individuals are composed of one or multiple quantitative traits. Following the quantitative genetics theory [1], different genetic architectures can underly each trait.

3.2.1 Genomes

Within Metapop, a genome is designed as a linear continuous succession of loci. The total length of the genome is indicated by the user as the total number of loci. The genome is subdivided in chromosomes, indicated by the position of the first locus within the linear succession. Genetic distance between successive loci is generated by the recombination rate indicated by the user. Recombination rates can be assigned to each pair of successive loci along the linear succession of loci. Recombination rates between successive loci at the limit between two chromosomes (chromosome split) are set to 0.5. See next subsection, paragraph 3.4.3 for more information about the configuration of recombination.

At this stage, all loci are considered neutral by default. The next step consists in identifying loci that contribute to the genetic value of traits undergoing selection, eg the quantitative trait loci (QTL). This is first done by assigning QTLs to traits, and then identifying the position of the QTLs in the genome (number of the locus along the linear succession of loci). Pleiotropic effects can be generated by assigning more than one trait to a given QTLs. The following example gives some of the numerous possible genome representations associated to a trait determined by three QTLs:

```
Position  0 . . . . 5 . . . . 10
Z1        |Q n n n|Q n n n|Q n n n
Z2        |Q n Q Q
```

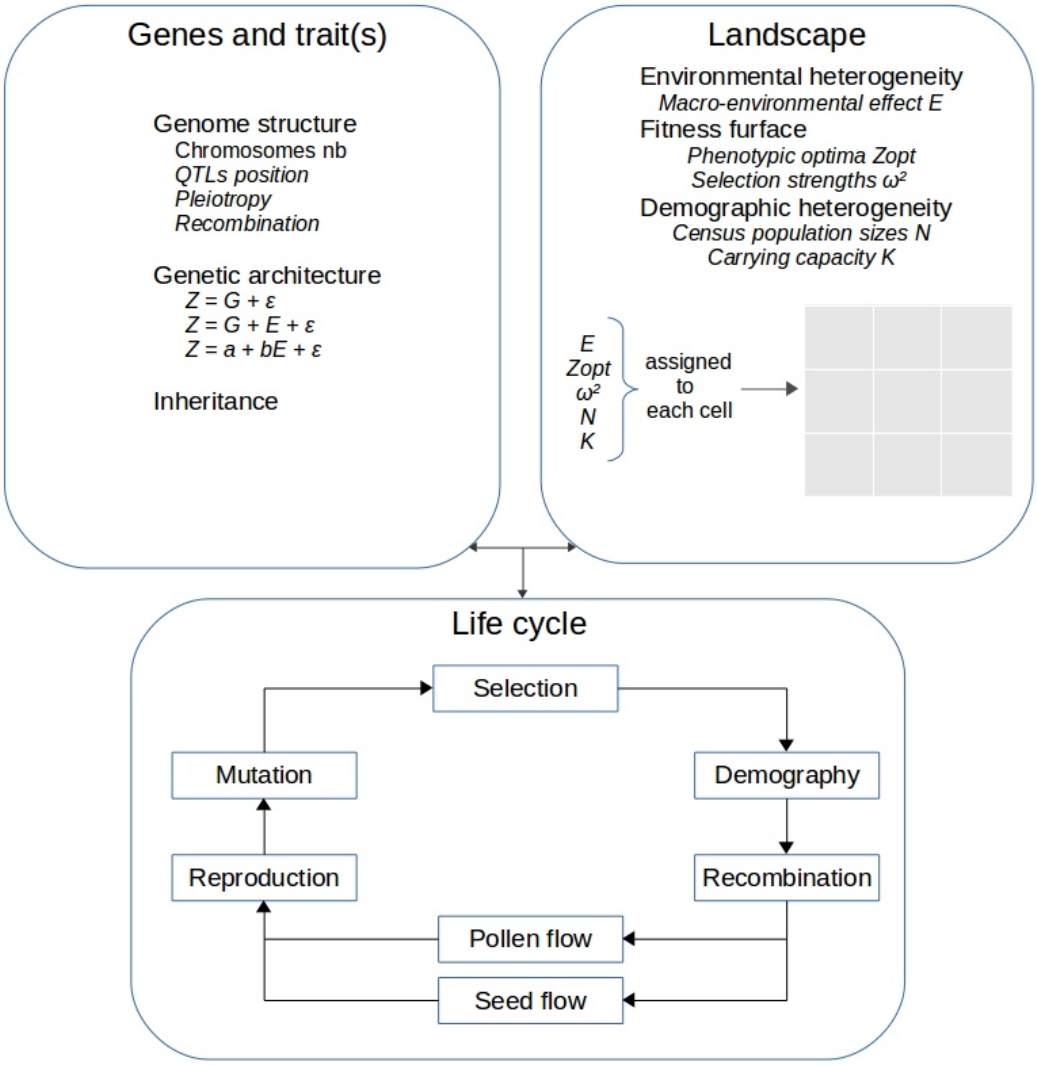

Figure 2: Model structure. Z: quantitative value of the trait, G genetic value, corresponds to the sum of the allelic effects at the loci involved in the variability of the trait, ϵ micro-environmnental value randomly drawn for each individual, E: macro-environmental value assigned at the cell/population scale, a sum of the allelic effects observed at the loci determining the intercept of the reaction norm, b sum of the allelic effects at the loci determining the slope of the reaction norm, Z_{opt} phenotypic optimum of a cell, a component of stabilizing selection, assigned independently to each cell, ω intensity of the stabilizing selection exerted in a cell, N starting number of individuals, K carrying capacity of a cell.

```
Z3          |n Q Q n|n n n n Q n
```

In this example, Q stands for QTL, n stands for neutral loci, the symbol — indicates a chromosome split, namely the beginning of a new chromosome. In the case of Z_1, the QTLs are equidistant and positioned on separate chromosomes, in the case of Z_2 all the QTLs are on the same chromosome but separated by distinct distances, while Z_3 is based on QTLs located on two chromosomes of distinct lengths. The trait Z_3 can be initialized as follows:
species/S1.lua

```
nLocus = 10
genomeSize = 20
chromoSplit = {0, 4}
```

traits/T1.lua

```
ltypes = {
  {
      id = "a",
      lociList = { 1, 2, 8 },
  },
}
```

3.2.2 From genomes to traits: phenotypic and genetic subdivisions

In the simplest case, the phenotypic value of trait Z is the the sum of the additive effects of the alleles at the n QTLs l contributing to the trait and a random micro-environmental contribution ϵ following the centered-reduced distribution $\mathcal{N}(0, \sigma_\epsilon^2)$. Assuming the species under consideration is diploid, this corresponds to:

$$Z = \sum_{l=1}^{n} (\alpha_i + \alpha_j)_l + \epsilon$$

where α_i, α_j refer to the additive effects of the alleles observed at each of the n QTLs for a diploid species.

In what follows we call genetic value G the sum of the additive contribution of the alleles at the QTLs determining the value of the trait. When a species is diploid:

$$G = \sum_{l=1}^{n} (\alpha_i + \alpha_j)_l$$

Thus, in the simplest case a trait value can be expressed as:

$$Z = G + \epsilon$$

In the current version of Metapop, we do not consider dominance nor epistatic interactions effects between alleles.

Multiple populations subdivided in heterogeneous landscapes are exposed to different macro-environmental conditions such as temperature, rainfall or any other abiotic or biotic factor. This macro-environmental effect (E) can contribute to the phenotypic value of the traits under investigation. For instance, in broadleaved trees, the timing of bud burst is affected by local temperatures. Overall, all individuals of a population are exposed to the same macro-environmental conditions and thus share the same E value. In Metapop, accounting for the contribution of plasticity to a trait value is done in two situations: when the plasticity of the trait is only generated by environmental effects (environmental plasticity) or when the plastic response depends also on genotype-environment interactions GxE (heritable plasticity). In the former case, when phenotypic plasticity is not heritable, the environmental conditions defined in a cell affect uniformly all the individuals whatever their genotypes:

$$Z = G + E + \epsilon$$

where E stands for the macro-environmental effect. Alternatively, when phenotypic plasticity includes also genotype-environment interaction GxE, a linear norm characterized by slope b and intercept a can be defined according to:

$$Z = a + bE + \epsilon$$

Here a corresponds to the additive contribution of the QTLs which determines specifically the intercept of the reaction norm, and b corresponds to the additive contribution of the QTLs which determine the slope of the reaction norm. Hence, when plasticity is heritable, two subsets of loci corresponding to a and b have to be initialized by the user. Importantly, there is no constraint nor cost associated with the evolution of plasticity in Metapop.

Here are three examples of initialization of a trait $T1$ determined by 7 QTLs. The 3 examples of *T1.lua* file provided below illustrate each of the possible phenotypic subdivisions.
$Z = G + \epsilon$: *species/T1.lua*

```
nhpp = false
ltypes = {
      {
          id = "a",
          lociList = { 1, 2, 3, 4, 5, 6, 7 },
        },
    },
```

$Z = G + E + \epsilon$: *species/T1.lua*

```
-- Phenotypic plasticity
-- If non heritable phenotypic plasticity, Z = G + E + epsilon
-- no GxE interaction, nhpp should be true
nhpp = true
ltypes = {
```

```
    {
        id = "a",
        lociList = { 1, 2, 3, 4, 5, 6, 7 },
    },
}
```

$Z = a + bE + \epsilon$: *species/T1.lua*

```
-- Phenotypic plasticity
-- If non heritable phenotypic plasticity, Z = G + E + epsilon
-- no GxE interaction, nhpp should be true
nhpp = false
-- If GxE interaction, E reference is needed
Eref = 0.1
ltypes = {
    {
        id = "a",
        lociList = { 1, 2, 7 },
        phi = 0.5
    },
    {
        id = "b",
        lociList = { 3, 4, 5, 6 },
        phi = 0.5
    },
}
```

A full initializing example including two traits is proposed in section 6 (scenario B).

3.2.3 Heritability and allelic effects

The allelic effects assigned to each QTL l are randomly drawn from a Gaussian distribution $N(0, W_l \times \sigma^2)$ where W_l designates the weight of the locus and σ^2 is the additive variance of the allelic effects drawn, respectively. By default, all the loci weights equal 1, alternatively they can follow a pseudo-Gamma distribution parameterized by the user. The specification of the loci weights can be done in */traits/Ti.lua* file:

```
varLociWeights = 0 --same weight of 1 for all loci
```

or

```
varLociWeights = 2 --loci weights follow a gamma_like
   distribution
```

For technical reasons, the Gamma distribution is approximated by a Normal law: $W_l \sim |\mathcal{N}(1/N_L, V_{lw})|$, with N_L is the number of QTLs and V_{lw} the user-defined variance of the loci weights (varLociWeights). In the example above, V_{lw} equals 2. The corresponding gamma distribution defined by m and p, the respective parameters for shape and scale, is: $m \approx 1.6$, $p \approx \frac{0.36}{{N_L}^2 * \sqrt{V_{lw}}} + 0.57 * \sqrt{V_{lw}}$

In the simplest case, *i.e.* when the decomposition of a trait does not include any GxE interactions ($Z = G + \epsilon$ or $Z = G + E + \epsilon$), the value of σ^2 depends on the heritability of the trait which is indicated by the user. In this case, the heritability of a trait is:

$$h^2 = \frac{\sigma_G^2}{\sigma_G^2 + \sigma_\epsilon^2}$$

Without dominance nor epistasis, the genetic variance σ_G^2 is obtained from:

$$\sigma_G^2 = \frac{h^2}{\sigma_\epsilon^2 - h^2}$$

Assuming that the QTLs are independent, the variance of the allelic effects σ^2 drawn at each locus is:

$$\sigma^2 = \frac{h^2}{n_L \times p \times (\sigma_\epsilon^2 - h^2)}$$

where n_L and p designate the number of QTLs and the ploidy of the species, respectively. When a trait is plastic and includes G x E interactions ($Z = a + bE + \epsilon$), the heritability within a given population where the macro-environmental effect is E, is:

$$h^2 = \frac{\sigma_a^2 + \sigma_b^2 E^2 + 2Ecov(a,b)}{\sigma_a^2 + \sigma_b^2 E^2 + 2Ecov(a,b) + \sigma_\epsilon^2}$$

Assuming the covariance between a and b loci is initially null we have:

$$h^2 = \frac{\sigma_a^2 + \sigma_b^2 E^2}{\sigma_a^2 + \sigma_b^2 E^2 + \sigma_\epsilon^2}$$

Unlike the simplest cases, the heritability expression cannot be used to determine σ^2 when GxE interactions are simulated. In this case the definition of σ^2 is based on two variables, ϕ_a and ϕ_b. These variables indicate the proportion of the phenotypic variance that depends on variability at a and b loci respectively [2]. The user chooses values for ϕ_a and ϕ_b at the initialisation step. For a loci:

$$\phi_a = \frac{\sigma_a^2}{(\sigma_a^2 + \sigma_{bE}^2 + \sigma_\epsilon^2)}$$

for b loci:

$$\phi_{bE} = \frac{\sigma_{bE}^2}{(\sigma_a^2 + \sigma_{bE}^2 + \sigma_\epsilon^2)}$$

Hence, σ^2 depends on the type of the loci and the macro-environmental value E of the reference environment used, i.e the environment in which the additive components of the

traits are estimated. Assuming a null reference environment ($E = 0$), the additive variance associated with the a loci is:

$$\sigma_a^2 = \frac{\phi_a \times \sigma_\epsilon^2}{(1 - \phi_a)}$$

At each a locus:

$$\sigma^2 = \frac{\phi_a \times \sigma_\epsilon^2}{n_a * p * (1 - \phi_a)}$$

where n_a and p designate the number of a QTLs and the ploidy of the species, respectively. The calculation of σ_b^2, the additive variance at b loci, requires a non-null macro-environmental value E:

$$\sigma_b^2 = \frac{\phi_{bE} * (\sigma_a^2 + \sigma_\epsilon^2)}{(1 - \phi_b) * E}$$

At each b locus:

$$\sigma^2 = \frac{\phi_{bE} * (\sigma_a^2 + \sigma_\epsilon^2)}{n_b * p * (1 - \phi_b) * E}$$

where n_b and p designate the number of b QTLs and the ploidy of the species, respectively. To sum up, in order to obtain initial values for σ^2, the user is asked to provide initial values of h^2, ϕ_a, ϕ_{bE}, depending on the genetic architecture.

3.3 Inheritance

By default, in the diploid case, all the loci are inherited both from the female and male parents. It is however possible to simulate alternative patterns of inheritance by modifying the *inheritance* variable in the species file (*Si.lua*). This variable specifies from which parent each allele is inherited. The inheritance variable can be initialized either globally or for each locus. A single value indicates that a unique value is used for all the loci, alternatively, multiple values (i.e. the same number than the number of loci) can be specified. In any case, the sum of the integer should equal the ploidy of the species considered. In the following example, the species is diploid and all the loci are inherited both from the female and the male parent in equal proportion:
S1.lua

```
-- Inheritance, sum must equal ploidy
inheritance = {
            female = { 1 },
            male =   { 1 },
      }
```

Let us consider a genome comprising 5 loci. Different inheritance patterns can be defined at the different loci:

```
-- Inheritance, sum must equal ploidy
inheritance = {
            female = { 1, 2, 0, 0, 0},
```

```
            male =    { 1, 0, 2, 2, 2},
        }
```

In this example locus 2 is exclusively inherited from the female parent while locus 3, 4 and 5 are exclusively inherited from the male parent. Finally another example of an haploid species could be:

```
-- Inheritance, sum must equal ploidy
inheritance = {
            female = { 1, 1, 0, 0, 0},
            male =   { 0, 0, 1, 1, 1},
        }
```

3.4 Life cycle

In Metapop generations do not overlap: all the individuals undergo simultaneously each step of the life cycle (Figure 2), moreover the parents are removed from the landscape once offspring have been generated. To sum up, each individual undergoes a single life cycle.

3.4.1 Natural selection

The life cycle starts with natural selection (Figure 2). Each cell composing the landscape (see next section) imposes a stabilizing selection pressure [3] on the phenotypes. Fitness values W are computed for each individual and determine the probability to reproduce and contribute to offspring production. In the case of a phenotype composed of a single trait Z, the fitness value $W(Z)$ is computed from a local optimal trait value Z_{opt} and an intensity of selection $1/\omega^2$:

$$W(Z) = \exp\left(\frac{-(Z - Z_{opt})^2}{2\omega^2}\right)$$

The phenotypic optimal values and the intensity of selection are set for each trait Z in the file *traits/ty.lua*. There is a separate file for each trait. Within each file, the keywords *selIntParam* and *zoptParam* designate the intensity of selection $1/\omega^2$ imposed on the trait and the local optimal values Z_{opt} assigned throughout the landscape for the trait, respectively.

Here is an example of divergent selection set throughout one dimension of a landscape structured according to a 4 x 3 grid. Phenotypes are composed of a single trait. Change in selection pressure and environmental values start at generation 10. For more information about the initialization modes of composite variables, see paragraph 2.5.2. For detailed information of input files structures, see section 4.1.

conf.lua, landscape dimensions:

```
-- Map dimensions
mapLength = 3
mapWidth = 4
```

In this example, both the intensities of selection (selIntParam) and the optimal values (zoptParam) are indicated manually, which has to be specified in the file */traits/T1.lua*:

```
-- Selection Intensity
selIntParam =
{
    source = "userdata",
}

-- Optimal trait value Zopt
zoptParam =
{
    source = "userdata",

}
```

zopts.txt, optimal trait values assigned to each cell, for trait T1. At the beginning selection is uniform, divergent selection starts at generation 8:

```
# zopt T1 1
0 0 0 0
0 0 0 0
0 0 0 0

# zopt T1 8
2 2 2 2
0 0 0 0
-2 -2 -2 -2
```

sel_int.txt is the intensity of stabilizing selection assigned to each cell. Here a moderate intensity of selection is specified for the whole simulation process.

```
# selection_intensity T1 1
50 50 50 50
50 50 50 50
50 50 50 50
```

When a phenotype is composed of multiple traits, the fitness of each individual is calculated according to [4]:

$$W = \exp\left(-0.5 \times (\boldsymbol{Z} - \boldsymbol{\theta})^T \times \boldsymbol{\Omega}^{-1} \times (\boldsymbol{Z} - \boldsymbol{\theta})\right)$$

where $\boldsymbol{Z}$ is a vector of the trait values composing the phenotype, $\boldsymbol{\theta}$ is a vector of the optimal trait values in the cell, and $\boldsymbol{\Omega}$ is a symmetrical matrix, the diagonal being the intensities of stabilizing selection acting on each trait, and off-diagonal elements are measures of intensities of correlational selection. Matrix $\boldsymbol{\Omega}$ illustrates the curvature of the fitness surface of selection intensity. When multiple traits are simulated, the parameter *selIntInteraction* in the file *Si.lua* defines the $\boldsymbol{\Omega}$ matrix. Default values of off- diagonal terms are set to 0. More information about the selection matrix can be found in section 4. Distinct optimal phenotypes and intensities of stabilizing selection can be assigned to the cells composing the landscape, which can result in multiple patterns of divergent selection among cells. For instance, assuming a landscape can be summarized by latitudes and longitudes, divergent selection can be simulated throughout one dimension, by assigning different Z_{opt_k} and ω^2 values to different latitudes (see paragraph 3.5). Likewise, the phenotypic optima and the intensities of selection can be changed over successive generations. No selection can be defined by setting ω^2 to large values, as for example 10^9.
The list of studied traits is indicated in *S1.lua*. For instance, if two traits are considered: *S1.lua*:

```
traitsId = {"T1", "T2"}
```

Note that, by default, Metapop can simulate up to three traits. Nonetheless, this limit can be extended by adding new trait IDs to the file *.struct/traits/index.txt*.

3.4.2 Demography

At each generation t, the census number N_t of individuals is updated for each cell, according to the following demographic growth model. In each cell, N_t depends on (i) the size of the population of parents N_{t-1}, (ii) a growth rate g strictly positive, (iii) the number of seeds N_s reaching the cell, (iv) the carrying capacity K of the cell and (v), random ϵ values which follows the uniform distribution U(0,1). Hence, at each generation the number of individuals to generate is calculated from:

$$N_t = N_{t-1} + g \times N_s \times (1 - \frac{N_{t-1}}{K}) + \epsilon$$

The grow rate g can be positive or negative. The number of offspring N_t is always bounded by 0 and the carrying capacity K of the cell. Note that, because the demographic model is grounded on a bounded sigmoid function, the values of g can have no effect on the size of a population, depending on its size N_t and the parameter K specified.

The demography of the species simulated has to be initialized in the species files *species/Sx.txt*. For instance, for a species S1, the parameters can be initialized according to:

```
-- Demography
cap = 1000
growth = 1
-- Initial number of individuals per cell
```

```
initDemo =
{
    source = "value",
    npop = 1000
}
```

Here, *cap*, *growth* and *initDemo* refer to K, g and the initial number of individuals N_0 respectively. The number of initial individuals can be initialized according to three ways, depending on the *source* variable. When *source* = "*value*", the initial number of individuals is assigned uniformly throughout the landscape (source = "value"). When *source* = "*param*" the user can rapidly enter a pattern of variation of demography according to one axis: latitude or longitude. In this case, all the cells of the same level (latitude or longitude) have the same initial number of individuals (see next chapter for further details). When *source* = "*userdata*", the user has to fill a matrix corresponding to the size of the landscape in the file *demo.txt*. The initial demographic values generated by this example can be:

```
1000 1000 1000 1000
1000 1000 1000 1000
1000 1000 1000 1000
```

At last, as presented in the first part, there is also the possibility to directly import preexisting genotypes through the mode *source* = "injection". In this case the individuals are imported from *ind.txt*. Hence, the number of individuals corresponds to the number of genotypes.

3.4.3 Recombination

When diploid species are simulated, crossing-overs can occur during gametogenesis. The probability of crossing over per locus can be set in */species/Sx.lua* files: */species/S1.lua*

```
crossingRate = { 0.5 }
```

3.4.4 Gene flow

Seeds and pollen are dispersed within and between cells according to the Wright's Island migration model, the stepping stone migration model or any other user-defined pattern. The selection of the island or stepping stone migration models requires the user to specify in *species/S1.lua* the proportion of seeds m_s and pollen m_p produced by the individuals of a population that reach the other connected populations. In the case of a user-defined dispersal pattern, dispersal matrix for pollen and seeds have to be filled in the file *patterns.txt*. Each matrix specifies the proportions of the pollen and seeds produced which remain either local or disperse towards other populations. This matrix should have the same dimensions

than the landscape defined. The values specified at the beginning of the simulation regarding dispersal cannot be changed during the simulation process. The following two examples illustrate the use of the predefined and the user-defined dispersal models.
Example 1: pollen is dispersed according to the island migration model, seeds are dispersed to the stepping stone migration model.

```
fluxSource = { pollen = "island", seed = "steppingStone" }
ms = 0.0004
mp = 0.04
```

Example 2: pollen is dispersed according to a user-defined dispersal model, seeds are dispersed according to the stepping stone migration model.

```
-- Migration
fluxSource = { pollen = "pattern", seed = "steppingStone" }
ms = 0.0004
mp = 0.04  --ignored here
```

patterns.txt

```
# flux_pattern S1 pollen 1
4 5 2
10 50 5
4 5 2
```

3.4.5 Reproduction

The reproduction step produces the required number of offspring, each mating producing one offspring. Different reproduction modes can be simulated: selfing, random, assortative mating or mixed (random + selfing or assortative mating + selfing). Individuals are considered as monoecious. Selfing rate can be set by the user through the $selfingRat$ variable in */species/Sx.lua*. We assume here that each individual both have female and male flowers, and that self-fertilization can eventually occur according to a rate specified by the user. In the case of random mating, the individuals mate freely according to their fitness values and the pollen dispersal matrix. For each mating, the female parent is first randomly drawn, the draw being weighted by the relative fitness values assigned during the selection step. Then a male parent is then also drawn randomly among the individuals which contribute to the pollen cloud that reaches the population of the female parent. Similarly to the female parent, the draw of the male parent is also weighted by the fitness values.
In the case of assortative mating, the random draws of the female and male parents are

also weighted by the fitness values. Moreover for each assortative mating within population p, a female parent is first randomly drawn from the individuals of p. The draw of the male parent occurs next is however only made among the individuals fulfilling two requirements: (1) they contribute to the pollen cloud that reaches population p, (2) their phenotypic value Z_M is sufficiently close to the phenotypic value of the female parent Z_F such that $Z_M \in [Z_F - \delta, Z_F + \delta]$. The strength of assortative mating is scaled by ρ, the correlation between the male and female phenotypic values at generation 0:

$$\rho = \frac{cov(Z_{M0}, Z_{M0} + \delta)}{\sigma_{Z_{M0}} \times \sigma_{Z_{M0}+\delta}}$$

Here Z_{M0} and Z_{F0} designate the male and female phenotypic values at generation 0, respectively. $\sigma_{Z_{M0}}$ and $\sigma_{Z_{F0}}$ refer to the standard deviation of the male and female phenotypic values at generation 0 respectively. Because in our simulations all the individuals can be male or female individuals, $\sigma^2_{Z_{M0}} = \sigma^2_{Z_{F0}} = \sigma^2_{Z_0}$. Assuming that *delta* follows the gaussian distribution $\mathcal{N}(0, \sigma^2_\delta)$, and depends on ρ and $\sigma^2_{Z_0}$, each generation a δ value was randomly drawn from the truncated gaussian distribution:

$$\mathcal{N}(0, \sqrt{\frac{\sigma^2_{Z_0}}{\rho^2} - \sigma^2_{Z_0}})$$

Each δ value was used for all the matings occurring within a generation.

In the example below, the individuals of species S1 mate assortatively according to the character T1. Selfertilization rate is 0.001

S1.lua.txt

```
-- Reproduction regime
assortMating = "T1"
rho = 0.8
selfingRate = 0.001
```

3.4.6 Mutation

Mutation is the last genetic process considered in the life cycle (Figure 2).

When a mutation occurs at a given locus, a new allele is randomly drawn among a predefined number of possible alleles that are generated at the beginning of the simulation. The mutation rate of a species *Sx* can be initialized as:

/species/Sx.lua

```
mutationRate = { 1e-05 }
```

3.5 Landscapes

A landscape is sketched as a collection of cells (Figure 2), also referred as patches, distributed over two dimensions. The individuals of the same species co-existing in a cell form a population. A landscape is characterized by (*i*) its shape and dimensions,(*ii*) the distribution of macro-environmental values across the cells, (*iii*) the distribution of the selection pressures (Z_{opt}, $1/\omega^2$) among the cells.

3.5.1 Shape and dimensions

A landscape simulated by Metapop is always a rectangle (or square) defined by a longitude (called *mapWidth* in the parameter file) and a latitude (called *mapLength* in the parameter file) expressed as a number of cells along the two dimensions. The dimensions of the landscape should be defined in *simuConf.lua*:

```
mapLength = 4
mapWidth = 3
```

3.5.2 Macro-environmental effect

In Metapop, macro-environmental values E can influence the expression of plastic traits (see section 3.2). The E values are assigned at the cell level for each plastic trait composing the phenotype. Hence, all individuals of a cell are exposed to the same macro-environmental conditions. When phenotypes are composed of multiple plastic traits, multiple environmental values are assigned to a cell. It is important to note that the macro-environmental values E differ from the micro-environmental influence ϵ aforementioned (see 3.2.2) which is related to other less determinant factors such as the position of an individual in its cell. Users can set any spatial patterns of variation of E values, such as geographic gradients at the landscape scale. The initialization of E values can be done according to three modes: "userdata", "value" and "param" (see "input section" 4.2 for more information). The "userdata" mode requires the user to fill matrix (e.txt file) assigning an E value to each cell of the landscape. Under this initialization mode, at least one matrix has to be defined per trait. Moreover, under this initialization mode exclusively, the user can set temporal variations of E values by defining multiple matrix (see example below). Each matrix defined in the file e.txt must be entitled by a line "# E *trait name generation*", where *trait name* and *generation* indicate the trait targeted and the point of the simulation at which the matrix of E values is taken into account, respectively. The "value" mode requires a single value uniformly assigned to all the cells of the landscape. One value per trait composing the phenotype is required. Finally the "param" mode automatically generates unidimensional gradients of environmental values across the length of the landscape, *i.e.* the south-north direction. The gradient is scaled from the slope k_e entered by the user.

In the example below, we manually set the macro environmental values influencing the expression of a single plastic trait $T1$. All the E values are set to 0 at the beginning of the simulation and modified at generation 8.

in */user_input/traits/T1.lua*:

```
-- Macro-environmental effect E
envEffParam =
{
    -- Source of data, allowed values: userdata, value, param
    -- If source == "userdata": matrix e.txt directly filled by
    user
    -- If source == "value": a unique value is assigned to all
   cells, arg: E = number
    -- If source == "param": generation of a gradient according
    to one dimension, arg: Ke = number,
    source = "userdata",
}
nhpp = true
```

The last line initializing *nhpp* variable to *true*, indicates that phenotypic plasticity is here not heritable, which corresponds to the phenotypic decomposition formalized by equation (2). in */user_input/e.txt*:

```
# E T1 1
0 0 0 0
0 0 0 0
0 0 0 0

# E T1 8
 4  4  4  4
 0  0  0  0
-4 -4 -4 -4
```

In this example, the first matrix used at generation 1 for trait T1 is replaced by the second matrix at generation 8.

3.5.3 Natural selection

Like macro-environmental values, the optimal phenotypic values (Z_{opt}) and intensity of stabilizing selection ($1/\omega^2$) (subsection 3.4) exerted on the traits can be independently assigned to the cells composing the landscape. This can result in multiple patterns of divergent selection among cells. For instance, divergent selection can be simulated by assigning different Z_{opt} and ω^2 values to latitudes. Note that the intensities of selection set by the users in the configuration files are ω^2 values, inversely proportional to the strength of stabilizing selection $1/\omega^2$.
The initialization of Z_{opt} and ω^2 values can also be done according to three modes: "userdata", "value" and "param" (see "input section" 4.2 for more information). The "userdata"

mode requires the user to fill matrix in zopt.txt and sel_int.txt files. Each matrix respectively assigns a Z_{opt} and an ω^2 value to each cell of the landscape. Under this initialization mode, at least one Z_{opt} matrix and one ω^2 matrix have to be defined per trait. Each matrix defined in the zopt.txt must be entitled by a line "# zopt *trait name generation*", where *trait name* and *generation* indicate the trait targeted and the point of the simulation at which the matrix is taken into account, respectively. Each matrix defined in sel_int.txt must be entitled by a line "# selection_intensity *trait name generation*". Hence, under the "userdata" mode, the multiple matrix successively used by the program a different points of the simulation can describe spatio-temporal variations of both Z_{opt} and ω^2 values. The "value" mode requires single Z_{opt} and ω^2 values uniformly assigned to all the cells of the landscape. One Z_{opt} and ω^2 values per trait composing the phenotype is required. Finally the "param" mode is only available for the Z_{opt} values. This mode automatically generates unidimensional gradients of phenotypic optima across the length of the landscape. The gradient is scaled from the slope k_zopt entered by the user and a boolean variable "inversed" which determines the direction of the gradient generated. "inversed" set to "False" indicates that the direction of the Z_{opt} gradient is south-north, as any macro-environmental gradient defined. "inversed" set to "True" indicates that the direction of the Z_{opt} gradient is north-south.

In the example below, we manually set the the phenotypic optima through the "userdata" mode. No temporal variation of Z_{opt} values is considered. The selection intensity is uniformly set to 1/50 trough the "value" mode.

in */user_input/traits/T1.lua*:

```
-- Selection Intensity
selIntParam =
{
    -- Allowed values: userdata, value
    -- If source == "userdata": matrix sel_int.txt directly
   filled by user
    -- If source == "value": a unique value is assigned to all
   cells
    source = "value",
    omega = 50,
}
-- Optimal trait value Zopt
zoptParam =
{
    -- Source of data, allowed values: userdata, value, param
    -- If source == "userdata": matrix zopt.txt directly filled
    by user
    -- If source == "value": a unique value is assigned to all
   cells, arg: zopt = number
    -- If source == "param": generation of a gradient according
    to one dimension, args: opposed = boolean, Kzopt = number
    source = "userdata",
```

```
    opposed = false,
    Kzopt = 2,
}
```

in */user_input/zopt.txt*:

```
# zopt T1 1
2 2 2
0 0 0
-2 -2 -2
```

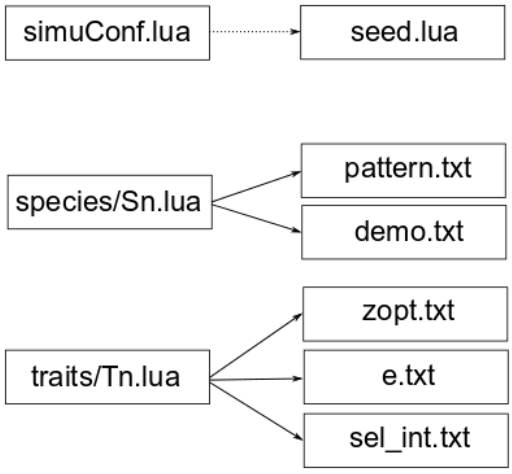

Figure 3: Associations between *primary* and *secondary* configuration files. Dotted arrow: automatic production, plain arrow: automatic or manual production.

4 Input

As mentioned in "Getting started" section, the simulations can be configured by modifying the *.lua* files from *user_input/* directory. This section presents in detail the variables that have to be initialized in each input file.

4.1 Overall structure

The primary files of interest are :

- simuConf.lua. Used for simulation management - number of replicates, seeds, simulation length, reference cell - and description of the dimensions of the landscape.
- species/Sn.lua, where n refers to the id of the species. Used for the specification of information related to the biology of the species considered, for instance: ploidy, reproduction system, gene flow. Although several species profiles can be easily created, there is no possibility of simulating simultaneously multiple species with the current version of the model.
- traits/Tn.lua, where n is the id of the trait. Used for information related to a trait contributing to the phenotype of the individuals of a given species.

Some of the variables in those files trigger the creation of additional *.lua* or *.txt* files exploited by the model. In some cases, when specified in the primary *.lua* files, the user has the possibility to manually create the additional file for some variables, by selecting the relevant keyword during the initialization of the corresponding variable. The distinct initialization modes are introduced in paragraph 4.1. Figure 3 summarizes the links between the primary and secondary configuration files.

Here is some information about the secondary files generated:

- *seed.lua*. Random seeds used for the generation of random variables during the simulations.
- *pattern.txt*. Patterns of pollen and seed flow dispersal.
- *demo.txt*. Initial population size at the starting meta-population.
- *zopt.txt*. Optimal phenotypic values assigned tor each trait throughout the landscape at different time points of the simulation.
- *e.txt*. Macro-environmental values affecting the expression of plastic trait. Also assigned across the landscape and can be changed at different time points of the simulations.
- *sel_int.txt*. Intensity of the selection exerted on each trait.

4.2 Files and variables

We now detail the structure of each primary configuration file. Several complete examples of configuration files are also provided in the section 6 (Configuration examples).

simuConf.lua

- *REPLICATIONS:* positive integers, number of simulation replicates to run.
- *IS_RANDOM_SEED, IS_RANDOM_SEED_INIT:* booleans, determine if the seeds (here random generators number, nothing to do with trees) have to be renewed between simulation replicates. *IS_RANDOM_SEED* determines whether the seeds used during a simulation have to randomly generated, *IS_RANDOM_SEED_INIT* determines whether the seeds used for the initialization of the simulations are randomly generated.
- *mapLength, mapWidth:* positive integers, dimensions of the landscape.
- *simLength:* positive integer, number of generations simulated.
- *saveStep:* positive integer, periodicity of saving raw output data.
- *refCell:* positive integer, cell used as reference for gene flow study, to know where the parents of the individuals of this cell come from.

```
-- REPLICATIONS PARAMS
REPLICATIONS = 3
IS\_RANDOM\_SEED = true
IS\_RANDOM\_SEED\_INIT = true


-- SIMULATION PARAMS
-- Map dimensions
```

```
mapLength = 5 -- MUST be even if loctypes b or c is present,
   otherwise some EE == 0 and AE == -Inf
mapWidth = 1

-- Simulation length
simLength = 10
saveStep = 5

outputDir="../outputs/"
refCell = 23  -- cell used as reference for gene flow study
```

species/S*n*.lua One file per defined species. Simulation of multiple species is not available in the current version of Metapop. The first file to be modified is *species/S1.lua*:

- *traitsId:* list, traits related to the phenotype of the individuals of the species.

```
fileType = "species"
id = "S1"
traitsId = {"T1", "T2"}
```

- *nLocus, genomeSize, ploidy:* positive integers, *nLocus* is the number of loci, $genomeSize = nLocus * ploidy$. The *ploidy* of the species should equal 1 or 2.

```
nLocus = 45
-- Genomesize equals number of locus times ploidy
genomeSize = 90
ploidy = { 2 }
```

- *nAllpLoci, nAllInit:* positive integers. *nAllpLoci* indicates the number of alleles per locus, it is used to randomly generate a set of alleles per locus at the beginning of each simulation. The number specified is assigned globally when a single value is given, or independently to each locus (neutral and QTL) when multiple values are specified. *nAllInit* is the number of alleles initially present in the starting meta-population at each locus. Only global initialization are possible, *i.e.* all the loci are initialized simultaneously.

```
-- Number of allele per locus
nAllpLoci =  { 100 }
 -- Number of alleles initially present in the meta-
   population for each trait. Cannot exceed nAllpLoci
nAllInit = { 6 }
```

Another possible configuration for genomes composed of 5 loci:

```
nLocus = 5
...
-- Number of allele per locus
nAllpLoci =  { 100, 100, 200, 100, 100 }
 -- Number of alleles initially present in the meta-
   population for each trait. Cannot exceed nAllpLoci
nAllInit = { 6 }
```

– *crossingRate, chromoSplit, mutationRate:* *crossingRate* is a list of floats included in $[0, 0.5]$, which corresponds to the recombination rate, *i.e.* the probability of having a crossing over at each locus. The list may contain one value, in which case the recombination rate will be the same at each locus, or it may contain n_{locus} values when the rate is defined for each locus.

chromoSplit is a list of positive integers, which specifies the position of the beginning of each chromosome across the genome, in number of loci. *mutationRate* is a float included in $[0, 1]$, defines the probability of occurrence of mutation at each locus. When a mutation occurs, the allele at the corresponding locus is replaced by an other allele drawn randomly from the set of alleles initially constructed from *nAllpLoci*.

```
crossingRate = { 0.5 }
chromoSplit = {0, 20}
```

– *inheritance:* $inheritance.female$ or $inheritance.male$ are integers included in [0, 1, 2] indicating whether the genes are inherited from the female and/or the male parent. For each locus the sum of the inheritance specified should equal the ploidy of the species. Hence, for a diploid species, the integer 2 assigned to $inheritance.female$ indicate that a locus is exclusively inherited from the female parent. In this case, the corresponding integer assigned to *inheritance.male* is 0. The inheritance can be initialized globally, i.e. simultaneously to all the loci, or separately for each locus. Assuming a genome of 4 loci:

```
-- Global standard initialization
inheritance = {
female = { 1 },
male =   { 1 },
}
```

```
-- Alternative inheritance pattern initialized for each
   locus
```

```
inheritance = {
female = { 1, 2, 1, 0},
male =   { 1, 0, 1, 2},
}
```

– *assortMating, rho, selfingRate:* Under assortative mating, matings can only occur between parents showing similar phenotypic values at a given trait. *assortMating* is a string indicating the name of the trait which triggers the selection of mating parents. No assortative mating, which is equivalent to random mating, is simulated when *assortMating* is initialized with *none*. *rho* is a float between 0 and 1 which scales the required correlation of phenotypic values between mating parents. *selfingRate* indicate the proportion of mating consisting in self-fertilization.

```
assortMating = nil
```

```
assortMating = "T1"
rho = 0.8
selfingRate = 0.001
```

– *fluxSource, ms, mp, Nm:* *fluxSource.pollen* and *fluxSource.seed* take each a value from *steppingStone*, *island*, *pattern* and define the dispersal of pollen and seeds respectively. When the *pattern* value is assigned, the secondary file *pattern.txt* has to be manually created, it is automatically created otherwise. Using the manual initialization allows the user to change the dispersal pattern at different points of the simulations. *ms* and *mp* define respectively the ratio of seeds and pollen dispersed from a population to other connected populations. These two variables are effectively used by the model only when *fluxSource.pollen* and *fluxSource.seed* are initialized with *steppingStone* or *island*. *Nm* quantifies the overall number of migrants in the initial meta-population. It is only used to generate, prior to any simulation, the initial allelic frequencies describing a meta-population at the mutation-migration-drift equilibrium, assuming an Island migration model (Dirichlet distribution). For species S1:

```
fluxSource = { pollen = "pattern", seed = "pattern" }
ms = 0.0004  -- unused here
mp = 0.04  -- unused here
-- Used to generate initial allelic frequencies
Nm = 10.2
```

In the required *patterns.txt* file, the user has to indicate the proportion of the pollen and/or seed produced that reaches the connected cells or populations. Here 8/13 of the

pollen produced by the male individuals of a population is spread equitably towards the surrounding populations. At generation 10 the dispersion curve changes and the proportion of pollen going to the outside of the cell becomes 4/7.

Seeds here disperse according to a stepping stone model[1] with 4/12 of the produced seeds reaching the adjacent cells.

```
# flux_pattern S1 pollen 1
1 1 1
1 5 1
1 1 1

# flux_pattern S1 pollen 10
0 1 0
1 3 1
0 1 0

# flux_pattern S1 seeds 1
0 1 0
1 8 1
0 1 0
```

– *initDemo, growth, cap: initDemo* specifies the census population size of the starting meta-population, namely the number of individuals initially present in each cell of the landscape at the beginning of the simulation process. The three modes of initialization introduced in paragraph 2.5.2 are available for the initialization of the census population sizes. When the variable *source* is initialized with "userdata" the user is required to manually fill a matrix in the secondary file *demo.txt*. Initializing *source* with "value" assigns a unique value, set in *npop* variable, globally at the landscape scale. Using "param" allows the user to specify a variation of the initial number of individuals according to latitudes or longitudes. The two last initialization modes automatically produce the secondary file *demo.txt* from the values assigned to the variable *npop*.

Based on these two cases, the variable *npop* can be initialized according to two ways. When the *value* initialization mode is selected, *npop* should contain a positive integer. When the *param* initialization mode is selected, *npop* contains a list of integers, which describes a variation of the initial number of individuals across the landscape. In this latter case, the *vertical* variable specifies the direction of this variation throughout the landscape, latitudinal if $vertical = true$ or horizontal if $vertical = false$. In this initialization mode, the number of integers assigned to *npop* has to equal the number of latitudinal or longitudinal levels of the landscape.

Lastly, the additional "injection" mode activates the generation of the initial population from an external file *ind.txt*.

[1] manually initiallized, the same seed dispersion pattern could have been obtained by changing the seed fluxSource to "steppingStone", and setting ms = 0.33 in the species configuration file S1.

Some setting examples:

```
initDemo =
{
    source = "value",
    npop = 500 -- global initialization
}
```

Across a 4x4 landscape, this global initialization will generate the following demography structure:

```
500 500 500 500
500 500 500 500
500 500 500 500
```

The initial demography can also be parameterized differently, without defining the whole map, through the *param* initialization mode:

```
initDemo =
{
    source = "param",
    vertical = false,
    npop = {0, 500, 100, 0}
}
```

The *param* initialization mode will generate the following demographic structure:

```
0 500 100 0
0 500 100 0
0 500 100 0
```

The last initialization for population description through the *injection* mode is written:

```
initDemo = {source = "injection"}
```

– *selIntInteractions:* list of non-null off-diagonal values of the selection intensity matrix.

```
selIntInteraction = {T1_T2 = 5}
```

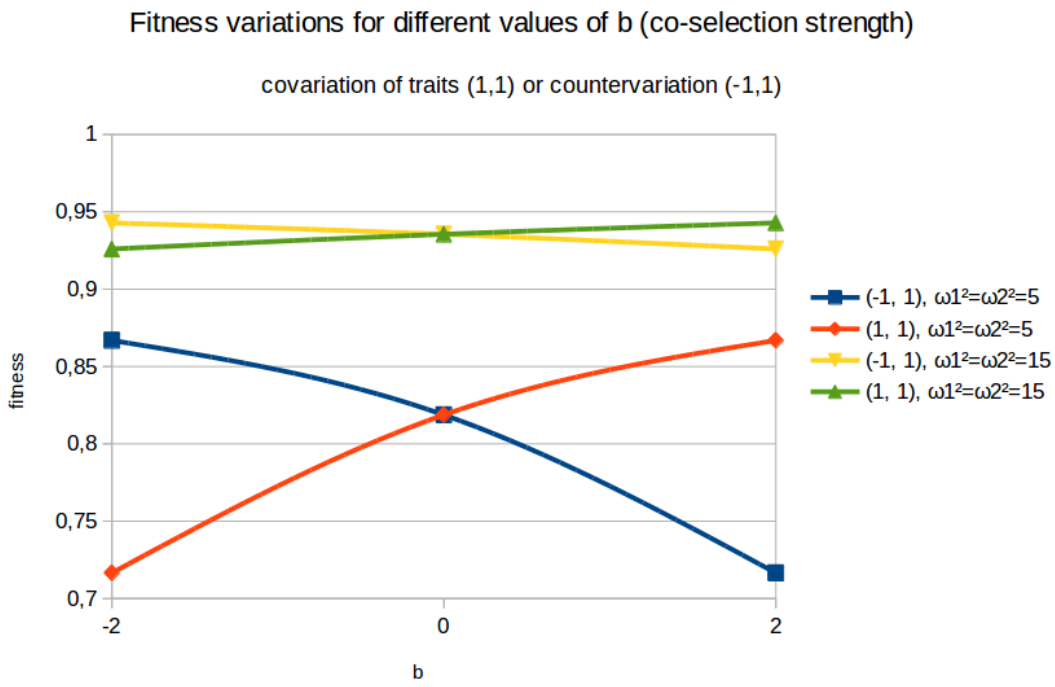

Figure 4: Example of fitness variations in the multi-trait case. ${\omega_1}^2$, ${\omega_2}^2$ refer to the diagonal values of the selection matrix $\mathbf{\Omega}$ used for the fitness computation. The x axis of the graph represents the off-diagonal values (symmetrical). For this graph we took ${\omega_1}^2 = {\omega_2}^2$, with 2 examples: ${\omega_1}^2 = {\omega_2}^2 = 5$ for a strong selection pressure and ${\omega_1}^2 = {\omega_2}^2 = 15$ for a weaker selection pressure.

– The selection intensity matrix Ω is a $n * n$ matrix, where n is the number of traits. For instance, for two traits:

$$\Omega = \begin{matrix} \omega_1^2 & \omega_{1,2}^2 \\ \omega_{2,1}^2 & \omega_2^2 \end{matrix}$$

, where ω_i is related to the strength of selection exerted on trait i and $\omega_{i,j}^2$ to the co-selection strength for trait i given trait j. $\omega_{i,j}^2$ are the values that are defined in the *selIntInteraction* variable.

For the following development, a simple matrix is build with a an identical value for both diagonal values and b as identical value for both off-diagonal values:

$$\Omega = \begin{matrix} a & b \\ b & a \end{matrix}$$

We illustrate here with an example how the user may choose off diagonal terms of the Ω matrix in the case on natural selection acting on two traits 1 and 2. Off diagonal terms are measures of intensities of correlational selection called here b terms for the sake of simplicity.

Figure 4 displays the evolution of fitness in case of two traits T_1 and T_2. The focus is put on the direction of variation of phenotypic values compared to the expected

optimum for the given trait. -1 and 1 in the pair (-1, 1) is the difference $Z - Z_{opt}$ for each trait, and could be written: $(Z_{T_1} - Z_{opt_{T_1}}, Z_{T_2} - Z_{opt_{T_2}})$

For example, the point of the blue curve at $b = 2$ gives the fitness value for

$$(\boldsymbol{z} - \boldsymbol{\theta}) = \begin{pmatrix} Z_{T_1} - Z_{opt_{T_1}} \\ Z_{T_2} - Z_{opt_{T_2}} \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \boldsymbol{\Omega} = \begin{pmatrix} 5 & 2 \\ 2 & 5 \end{pmatrix}$$

In the current case, for the second trait, Z is always the same: its value is greater than Z_{opt} and their difference is 1. The difference of $Z_{opt_{T_1}}$ and Z_{T_1} is also always 1, but the value can be lower than the optimum (-1) or greater (1). If the differences have the same sign, (1, 1) or (-1, -1), we have a covariation of these values. If their signs are opposed (1, -1) or (-1, 1), there is a countervariation.

This figure shows the similarity of the fitness variation profile for ${\omega_1}^2 = {\omega_2}^2 = 5$ and ${\omega_1}^2 = {\omega_2}^2 = 15$, there is a vertical translation and flattening for a=15 but the shape is the same.

The value is the same for countervariation and covariation when the non-diagonal values are null. But the covarying values have a greater value than the null case when b (the non-diagonal value w12 of the selection matrix) is positive, and lower than the null case when b is negative.

When the interaction between traits is positive, the fitness of covarying values becomes greater than without interaction, and lower when the interaction is negative.

traits/T*n*.lua Depending on the number of the traits, multiple trait files can be specified. Each trait is identified by an *id* and associated with a *species*:

```
fileType = "trait"
id = "T1"
species = "S1"
```

The main parameters are:

- *h2:*, float, varies between 0 and 1, gives the initial heritability of the trait. Initial heritability value is needed at the beginning to assign allelic effects at the QTLs (see Model section).
- *varLociWeights:*, float, indicates the variance of the weights of each locus, which affects the variance of the allelic effects generated: the larger the weight of a locus, the larger the variance of the additive effects of the allele drawn at this locus. When $varLociWeights$ is set to 0, all the loci have the same weight of 1. When $varLociWeights > 0$, the loci weight are distributed according to a gamma-like distribution.

```
varLociWeights = 2
```

– *selIntParam:* integers, are the ω^2 used to estimate the intensity of selection exerted on the trait. The intensity of selection is equal to $1/\omega^2$, in each cell of the landscape.

This variable is associated with the secondary input file *sel_int.txt*. When *selIntParam.source* is set to "value", a unique value is assigned to all cells and the corresponding secondary *sel_int.txt* file is automatically generated. When *selIntParam.source* is set to "user-data", the user has to provide its own *sel_int.txt* file. See paragraphs 2.5.2 and 3.5.3 for more information about the initialization modes. Here are two examples, assuming a 3x3 landscape: First case, "value" initialization mode:

```
selIntParam =
{
    source = "value",
    value = 50,
}
```

Second case, "userdata" initialization mode:

```
selIntParam =
{
    source = "userdata",
    value = 50,  --ignored here
}
```

A possible *sel_int.txt* file:

```
# selection_intensity T1 1
100 100 100
 50  50  50
100 100 100
```

– *zoptParam*: floats, optimal phenotypic values assigned throughout the landscape for the trait. These values can be initialized according to three modes of initialization. "userdata" requires the user to directly modify the files *sel_int.txt* or *zopt.txt*. In these two files, the user has to fill the matrix of cells of the landscape. This mode allows the user to indicate multiple matrices that will be successively used at different points in the simulations. Each matrix is identified by a title line "# zopt *trait generation*" where *trait generation* respectively designate the trait targeted by selection and the

generation number at which the matrix is applied during the simulation (see paragraph 3.5.3 for an example). The "value" mode assigns a unique value specified in the variable *zopt* to all cells. The "param" initialization mode generates automatically a gradient of values throughout the landscape, according to one dimension, centered around 0. In this case, the variable *Kzopt* defines the slope of the phenotypic optimal values, and the variable *opposed* specifies the direction of the gradient relatively to the gradient of environmental values, which has a south-north direction. To be activated, this option requires the existence of an environmental gradient activated through the variable *envEffParam* (see below).

```
zoptParam =
{
    source = "param",
    opposed = false,
    Kzopt = 0.2,
}
```

– *envEffParam*: floats, macro-environmental values affecting the expression of plastic traits. The macro-environmental value E is always associated to a trait and to a cell. The initialization modes available are the same as *zoptParam*: "userdata", "value" and "param". Each matrix defined in the "userdata" mode has to be identified by a title line "# e *trait generation*" where *trait generation* respectively designate the trait affected by the macro-environmental effect and the generation number at which the matrix is applied during the simulation (see paragraph 3.5.2 for an example) When *source* equals "param", the slope of the macro-environmental gradient generated is indicated by the variable named K_e. By default, the gradient generated follows a south-north direction and is centered around 0. Other spatio-temporal patterns of variations can be generated through the "userdata" initialization mode.

– *nhpp*: boolean, defined whether plasticity is heritable (GxE interactions) or not (no GxE interaction), when a trait is plastic (see paragraph 3.2.2).

– *Eref*: positive value, defines the reference environmental value used to generate the allelic effects at the a and b loci when a trait is plastic and plasticity is heritable (GxE interactions).

– *ltypes*, object composed of two parts. When the trait is not plastic, *ltypes* specifies the loci under selection (QTLs). When a trait is plastic (GxE interactions), *ltypes* determines which QTL(s) are associated with the intercept of the reaction norm (a loci) and which QTL(s) affect the slope of the reaction norm (b loci). Moreover, when GxE interactions are simulated, *phi* values have to be indicated, for each type of locus. A *phi* value corresponds to the proportion of phenotypic variance due to the additive variance at a type of locus (see formula in section 3.2.3 at page 15). *phi* values are used to draw the allelic effects at each locus.

```
  -- Phenotypic plasticity
-- If non heritable phenotypic plasticity, Z = G + E +
   epsilon
-- no GxE interaction, nhpp should be true
nhpp = false
-- If plasticity, need E reference (not necessary otherwise
   )
Eref = 0.1

-- Otherwise Z = a + bE + epsilon, two kinds of locus a and
    b
ltypes = {
    {
        id = "a",
        -- Locus of this ltype. Must be included in the
   species loci
        lociList = { 1, },
        -- Only used for allelic effects generation at
   locus a
        phi = 0.1,
        -- Power of E
        pow = 0,
    },
    {
        id = "b",
        lociList = {2},
        phi = 0.9,
        pow = 1,
    }
```

5 Output

5.1 Raw output

Each simulation replicate generates raw output in */output/Ri* directory, where *i* designates the replicate ID. This directory contains 3 sub-directories:

- *settings* summarizes the settings specified. Consists in a simple copy of the files modified by the user in *user_input* before the launch of the simulations
- *genotypes* contains the genotypic arrays of all individuals at different steps of the simulation process, according to the following format in case of haploidy:

  ```
  nPop nIndiv alleleLoc1 alleleLoc2...
  ```

  or the following one in case of diploidy:

  ```
  nPop nIndiv alleleLoc1 alleleLoc1 alleleLoc2 alleleLoc2...
  ```

- *quantitative*, contains quantitative information about traits, fitness, loci and gene flow. This directory includes the following files:
  - *g_TR-add.csv*: additive values, i.e. sum of the additive contribution of the alleles observed at all QTLs involved in the variability of the trait TR (see paragraph 3.2.2), at generation g.
  - *g_TRaddbylocus.csv*: additive values at each locus associated to trait TR at generation g.
  - *g_TR-pheno.csv*: trait values, i.e. sum of the additive, the macro- and the micro-environmental values (see paragraph 3.2.2), for trait TR at generation g.
  - *g_TR-add-a.csv*: additive values at the *a* loci associated with the plastic trait TR at generation g.
  - *g_TR-add-b.csv*: additive values at the *b* loci associated with the plastic trait TR at generation g.
  - *g_TR-epsi.csv*: micro-environmental values associated with the trait TR at generation g.

  The directory also contains information about migration flux: for a cell given as reference in *simuConf.lua*, the "geneflow" files contain a map with every cell. Each value corresponds to the quantity of pollen (geneflowP) or seed (geneflowS) coming from each cell that generates the individuals of the population in the reference cell:

- *g_geneflowP.csv*: quantity of pollen contributing to offspring generation in the population of the reference cell[2] at generation g.
- *g_geneflowS.csv*: quantity of seed contributing to offspring generation in the population of the reference cell at generation g.

5.2 Processed output

The automatic processing of outputs generates multiple files. Depending on the user settings, some of the files mentioned may not be generated in all cases. For instance the correlation files between traits *M*_Gcorr* and *M*_Pcorr* will not be generated if only one trait is simulated. Within the processed files listed below, the values given between brackets correspond to the variances among the simulated replicates.

Quantitative genetics

- *Summary_Demography*: number of individuals per cell or populations for the simulation steps recorded.

```
step  Pop 00  Pop 01  Pop 02  Pop 03
1 500.0 [0.0] 500.0 [0.0] 500.0 [0.0]
50  500.0 [0.0] 500.0 [0.0] 500.0 [0.0]
```

- *Summary_Gcorr*: coefficients of genetic correlation between traits. *CCG_b* corresponds to the correlation matrix computed at the between-populations level, *CCG_w* corresponds to the correlation matrix calculated at the within-population level averaged over all populations. Numbers between brackets indicate the variance over replicates.

```
step  CCG_b[1 , 1]  CCG_b[1 , 2]  CCG_b[2 , 2]...
1 1.0 [0.0] 0.014 [0.0079] [0.66] 1.0 [0.0]...
50  1.0 [0.0] 0.08 [0.0071] 1.0 [0.0]...
```

- *Summary_popGen_glob*: Diversity and differentiation parameters for all types of loci: *Gst*, *Hs* and *Ht*.

```
step  Gst Hs  Ht
1 0.022 [0.0009]  0.701 [0.0089]  0.718 [0.0088]
50  0.039 [0.0132]  0.675 [0.0215]  0.704 [0.0133]
```

[2]The ID of the cell corresponds to the parameter *refCell* of the input file *simuConf.lua*. The populations are numbered from 0 (top left of the map) to the number of cells minus one (bottom right). Cells are numbered from left to right, then from top to bottom.

– *Summary_QTLs*: quantitative values for the loci under selection (QTLs): *Gst_q*, *Hs_q*, *Ht_q*, *Qst*, *Qst_a*, *Qst_b*, *theta_b*, *theta_w*.

```
step  [T1] Gst_q  [T1] Hs_q [T1] Ht_q [T1] Qst...
1 0.02 [0.001]  0.74 [0.03] 0.763 [0.03]  0.25 [0.17]...
50  0.02 [0.005]  0.73 [0.02] 0.76 [0.02] 0.26 [0.21]...
```

These genetic index are calculated for a and b loci separately, when the model with genetic plasticity is used.
Gst: differentiation at QTLs.
Hs_q: maean population diversity at QTLs.
Ht_q: total diversity at QTLs.
Qst: differentiation at the quantitative trait.
theta_b: component of the between population variance due to covariance of allelic effects at the underlying QTLs.
theta_w: component of the within population variance due to covariance of allelic effects at the underlying QTLs.

– *Summary_Pcorr*: same as Gcorr, but for phenotypic correlation coefficient between traits.

– *Summary_TABLE_ADDMEAN-Tx-a_byPop*: additive values at a QTLs for each population (lines) and each step (columns) recorded.

– *Summary_TABLE_ADDMEAN-Tx-b_byPop*: additive values at b QTLs for each population and each step of the simulation saved.

5.3 Output conversion

The genotypes (genotypic arrays for each individual) produced by each simulation are recurrently saved in *genes_Si_g_.csv* where i is the species ID and g the generation number. These files can be converted to *Fstat* and *Genepop* files with the script */convertGenFiles.py*. For more information about these two formats, see the reference papers [6] and [5]. For instance, the conversion of the genotypes of species S1 saved at generation 20 can be used to generate a corresponding *Genepop* files using the option *G*:

```
python convertGenFiles.py user_input/R1/genes_S1_20.csv new_filename G
```

This command lines allow to construct a new file "new_filename". Conversion to the *Fstat* format requires to specify the option *F*

The conversion can be done in the other direction as described in the population import section with the option *I*, to get the ind.txt file from a Fstat or Genepop file.

6 Appendix: configuration examples

This appendix provides the complete configuration of two scenarios A and B. While scenario A is a simplified scenario, scenario B is more sophisticated, and includes changes in demography, different locus types, and multiple traits.

These scenarios are available in the folder *./EXAMPLES*, and the corresponding input files can be easily copied into the *./user_input* folder. To do so on Linux systems, use the command:

```
./scenarInputSet.sh A
```

Replace A by B to launch the scenario B. The detail of each file is given below.

Scenario A: neutral

simuConf.lua

```
-- REPLICATIONS PARAMS
REPLICATIONS = 5
IS_RANDOM_SEED = true
IS_RANDOM_SEED_INIT = true

-- SIMULATION PARAMS
-- Map dimensions
mapLength = 11
mapWidth = 5

-- Simulation length
simLength = 1000
saveStep = 20

outputDir = "../outputs/"
refCell = 1
```

species/S1.lua

```
-- Id of the trait, must be unique, avoid spaces
id = "S1"
-- List of traits for this species
traitsId = {"T1"}
-- Genetic architecture, species level
-- Total number of locus for all traits defined. Must be
   consistent with ploidy and genomeSize
nLocus = 20
-- Ploidy, 1 or 2
ploidy = { 2 }
-- Size of the genome should equal nLlocus * ploidy
genomeSize = 40
-- Number of allele per locus
nAllpLoci =  { 10 }
 -- Number of alleles initially present in the meta-population
   for each trait. Cannot exceed nAllpLoci
nAllInit = { 6 }
--  Crossing rate, >= 0 and <= 0.5
crossingRate = { 0.5 }
-- Beginning of chromosomes
chromoSplit = {0, 20}
-- Mutation rate
mutationRate = { 0.0001 }
-- Inheritance, sum must equal ploidy
inheritance = {
            female = { 1 },
            male =   { 1 },
       }
-- Reproduction regime
-- Assortative or random mating: requires a trait id or nil
assortMating = nil --"T1"
-- Correlation required between the trait values of the parents
    under assortative mating
rho = 0.8
-- Selfing rate
selfingRate = 0.02
-- Migration
-- Accepted values: steppingStone, island, matrix, pattern
-- matrix and pattern require a data file
fluxSource = { pollen = "island", seed = "island" }
-- ms (seed dispersal) and mp (pollen dispersal) are only used
   when fluxSource = steppingStone or island
ms = 0.0002  -- 0.0002 par d faut
```

```
mp = 0.02  -- 0.02 pard faut
-- For initial allelic frequencies generation only
Nm = 10
-- Demography
-- Carrying capacity = Maximal number of individuals per cell.
   Single value for all cells.
cap = 500
-- Growth for demography function
growth = 0
-- Initial number of individuals per cell
initDemo =
{
    -- If source == "userdata": matrix demo.txt directly filled
    by user
    -- If source == "injection": individuals are taken as
   described in external file ind.txt
    -- If source == "value": a unique value is assigned to all
   cells
    -- If source == "param": generation of a variation of
   demography according to one dimension
    --     vertical = true for profile following map length,
   false if follows map width
    --     if param, npop has to be a vector of population
   demography for each line or column of pop
    source = "injection",
    --vertical = false,
    npop = 500
}
-- Selection interaction for matricial fitness computation:
-- the value of Ti_Tj (when different from 0) describes the
   correlation (+ ou -) between traits
selIntInteractions = {T1_T2 = 0}
```

traits/T1.lua

```
id = "T1"
species = "S1"
--S7
-- Heritability of the trait
h2 = 0.833

--Locus weight for allelic effects
varLociWeights = 0  -- Changed to have uniform locus weight (
   all at 1), before was at 2 (gamma distrib)

-- Selection Intensity
selIntParam =
{
    -- Allowed values: userdata, value
    -- If source == "userdata": matrix sel_int.txt directly
   filled by user
    -- If source == "value": a unique value is assigned to all
   cells
    source = "value",
    omega = 1000000000, -- neutral selection
}

-- Macro-environmental effect E
envEffParam =
{
    -- Source of data, allowed values: userdata, value, param
    -- If source == "userdata": matrix e.txt directly filled by
    user
    -- If source == "value": a unique value is assigned to all
   cells, arg: E = number
    -- If source == "param": generation of a gradient according
    to one dimension, arg: Ke = number
    source = "value",
    Ke = 2 ,
}

-- Optimal trait value Zopt
zoptParam =
{
    -- Source of data, allowed values: userdata, value, param
    -- If source == "userdata": matrix zopt.txt directly filled
    by user
```

```
    -- If source == "value": a unique value is assigned to all
   cells, arg: zopt = number
    -- If source == "param": generation of a gradient according
    to one dimension, args: opposed = boolean, Kzopt = number
    source = "userdata",
    opposed = false,
    Kzopt = 1.7,
}

-- Phenotypic plasticity
-- If non heritable phenotypic plasticity, Z = G + E + epsilon
   (only loci of type a)
-- no GxE interaction, nhpp should be true
nhpp = false
-- Otherwise Z = a + bE + epsilon, two kinds of locus: a (
   intercept of reaction norm) and b (slope)
-- reference environmental value Eref required
Eref = 0.5

ltypes = {
{
    id = "a",
    lociList = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
   15, 16, 17, 18, 19, 20},
    phi = 0.45,
},
{
    id = "b",
    lociList = {},
    phi = 0.45,
}
}
```

Scenario B: multi-traits

conf.lua

```
-- REPLICATIONS PARAMS
REPLICATIONS = 5
IS_RANDOM_SEED = true
IS_RANDOM_SEED_INIT = true

-- SIMULATION PARAMS
```

```
-- Map dimensions
mapLength = 11
mapWidth = 5

-- Simulation length
simLength = 1000
saveStep = 20

outputDir = "../outputs/"
refCell = 1
```

species/S1.lua

```
-- Id of the trait, must be unique, avoid spaces
id = "S1"
-- List of traits for this species
traitsId = {"T1", "T2"}
-- Genetic architecture, species level
-- Total number of locus for all traits defined. Must be
   consistent with ploidy and genomeSize
nLocus = 30
-- Ploidy, 1 or 2
ploidy = { 2 }
-- Size of the genome should equal nLlocus * ploidy
genomeSize = 60
-- Number of allele per locus
nAllpLoci =  { 10 }
 -- Number of alleles initially present in the meta-population
   for each trait. Cannot exceed nAllpLoci
nAllInit = { 6 }
--  Crossing rate, >= 0 and <= 0.5
crossingRate = { 0.01 }
-- Beginning of chromosomes
chromoSplit = {0, 10}
-- Mutation rate
mutationRate = { 0.0001 }
-- Inheritance, sum must equal ploidy
inheritance = {
            female = { 1 },
            male =   { 1 },
       }
-- Reproduction regime
-- Assortative or random mating: requires a trait id or nil
```

```
assortMating = "T1"
-- Correlation required between the trait values of the parents
    under assortative mating
rho = 0.8
-- Selfing rate
selfingRate = 0.02
-- Migration
-- Accepted values: steppingStone, island, matrix, pattern
-- matrix and pattern require a data file
fluxSource = { pollen = "island", seed = "island" }
-- ms (seed dispersal) and mp (pollen dispersal) are only used
   when fluxSource = steppingStone or island
ms = 0.0002  -- 0.0002 par d faut
mp = 0.02  -- 0.02 pard faut
-- For initial allelic frequencies generation only
Nm = 10
-- Demography
-- Carrying capacity = Maximal number of individuals per cell.
   Single value for all cells.
cap = 500
-- Growth for demography function
growth = 1
-- Initial number of individuals per cell
initDemo =
{
    -- If source == "userdata": matrix demo.txt directly filled
    by user
    -- If source == "injection": individuals are taken as
   described in external file ind.txt
    -- If source == "value": a unique value is assigned to all
   cells
    -- If source == "param": generation of a variation of
   demography according to one dimension
    --     vertical = true for profile following map length,
   false if follows map width
    --     if param, npop has to be a vector of population
   demography for each line or column of pop
    source = "param",
    vertical = true,
    npop = {10, 10, 10, 10, 10, 50, 50, 40, 40, 30, 20, 10}
}
-- Selection interaction for matricial fitness computation:
-- the value of Ti_Tj (when different from 0) describes the
   correlation (+ ou -) between traits
selIntInteractions = {T1_T2 = 0}
```

traits/T1.lua

```
id = "T1"
species = "S1"
--S7
-- Heritability of the trait
h2 = 0.833

--Locus weight for allelic effects
varLociWeights = 2

-- Selection Intensity
selIntParam =
{
    -- Allowed values: userdata, value
    -- If source == "userdata": matrix sel_int.txt directly
   filled by user
    -- If source == "value": a unique value is assigned to all
   cells
    source = "value",
    omega = 5, -- neutral selection
}

-- Macro-environmental effect E
envEffParam =
{
    -- Source of data, allowed values: userdata, value, param
    -- If source == "userdata": matrix e.txt directly filled by
    user
    -- If source == "value": a unique value is assigned to all
   cells, arg: E = number
    -- If source == "param": generation of a gradient according
    to one dimension, arg: Ke = number
    source = "value",
    Ke = 2 ,
}

-- Optimal trait value Zopt
zoptParam =
{
    -- Source of data, allowed values: userdata, value, param
```

```
    -- If source == "userdata": matrix zopt.txt directly filled
    by user
    -- If source == "value": a unique value is assigned to all
   cells, arg: zopt = number
    -- If source == "param": generation of a gradient according
    to one dimension, args: opposed = boolean, Kzopt = number
    source = "userdata",
    opposed = false,
    Kzopt = 0.2,
}

-- Phenotypic plasticity
-- If non heritable phenotypic plasticity, Z = G + E + epsilon
   (only loci of type a)
-- no GxE interaction, nhpp should be true
nhpp = false
-- Otherwise Z = a + bE + epsilon, two kinds of locus: a (
   intercept of reaction norm) and b (slope)
-- reference environmental value Eref required
Eref = 0.1

ltypes = {
{
    id = "a",
    lociList = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
   15},
    phi = 0.45,
},
{
    id = "b",
    lociList = {},
    phi = 0.45,
}
}
```

traits/T2.lua

```
id = "T2"
species = "S1"
--S7
-- Heritability of the trait
h2 = 0.833
```

```
--Locus weight for allelic effects
varLociWeights = 2

-- Selection Intensity
selIntParam =
{
    -- Allowed values: userdata, value
    -- If source == "userdata": matrix sel_int.txt directly
   filled by user
    -- If source == "value": a unique value is assigned to all
   cells
    source = "value",
    omega = 50, -- neutral selection
}

-- Macro-environmental effect E
envEffParam =
{
    -- Source of data, allowed values: userdata, value, param
    -- If source == "userdata": matrix e.txt directly filled by
    user
    -- If source == "value": a unique value is assigned to all
   cells, arg: E = number
    -- If source == "param": generation of a gradient according
    to one dimension, arg: Ke = number
    source = "value",
    Ke = 2 ,
}

-- Optimal trait value Zopt
zoptParam =
{
    -- Source of data, allowed values: userdata, value, param
    -- If source == "userdata": matrix zopt.txt directly filled
    by user
    -- If source == "value": a unique value is assigned to all
   cells, arg: zopt = number
    -- If source == "param": generation of a gradient according
    to one dimension, args: opposed = boolean, Kzopt = number
    source = "userdata",
    opposed = false,
    Kzopt = 0.2,
}

-- Phenotypic plasticity
```

```
-- If non heritable phenotypic plasticity, Z = G + E + epsilon
   (only loci of type a)
-- no GxE interaction, nhpp should be true
nhpp = false
-- Otherwise Z = a + bE + epsilon, two kinds of locus: a (
   intercept of reaction norm) and b (slope)
-- reference environmental value Eref required
Eref = 0.1

ltypes = {
{
    id = "a",
    lociList = {16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27,
    28, 29, 30},
    phi = 0.45,
},
{
    id = "b",
    lociList = {},
    phi = 0.45,
}
}
```

References

[1] Lynch, M.; Walsh, B. Genetics and analysis of quantitative traits. Sunderland, MA: Sinauer, 1998, Vol. 1, p. 4.

[2] Lande, R. 2009. Adaptation to an extraodinary environment by the evolution of phenotypic plasticity and genetic assimilation. *Journal of evolutionary biology*, 22(7), pp.1435-1446.

[3] Turelli, M. (1984). Heritable genetic variation via mutation-selection balance: Lerch's zeta meets the abdominal bristle. Theoretical population biology, 25(2), 138-193.

[4] Reeve J.P. 2000. Predicting long term response to selection. Genet. Res. Gamb. 75: 83-94.

[5] Rousset, F. 2008. Genepop007: a complete reimplementation of the genepop software for Windows and Linux. Molecular ecology resources, 8(1), 103-106.

[6] Goudet, J. 1995. FSTAT (version 1.2): a computer program to calculate F-statistics. Journal of heredity, 86(6), 485-486.