SINS - Simulating INdividuals in Space

A population genetics program to simulate realistic genetic and demographic scenarios

T. Maié*, R. Rasteiro*, D. Curbelo, G.M. Sgarlata, T. Zoeten, P-A. Bouttier, D. Monier, V. Sousa, C. Thébaud, L. Chikhi

Population and Conservation Genetics group
Instituto Gulbenkian de Ciência

October 2024

Abstract

In this document we describe the main functionalities of SINS (which stands for Simulating INdividuals in Space). We first present the demographic and genetic models underlying (and implemented in) the simulation program. Next, we describe the different input and output files required and produced by the program. Finally, we explain how to run the program with examples provided along with SINS. It is important to note that SINS can produce huge files and generate large amounts of data. We therefore strongly suggest to run SINS with toy examples before going for the "‘real-life"’ scenarios the user may want to study/simulate.

Table of contents

Chapter 1

General Introduction

SINS is a new forward and individual-based simulation framework that incorporates both geographical and demographic data, as well as several types of genetic markers. The general principle is very similar to that followed by the SPLATCHE and the more recent SPLATCHE2 software [3, 10]. While the principle and many specific details are nearly identical there are also important differences and, as we will show, SINS can simulate scenarios that, to our knowledge, cannot be simulated under other frameworks [9]. There are also significant differences in the general principle. For instance, SINS is not based on the (gene-centered) coalescent, but uses an individual-based (rather than gene-based) forward simulation framework, where the demographic and genetic simulations are carried simultaneously. While this makes our program computationally slower, it has several advantages. In particular it is easier to:

- track the full ancestral information of sampled individuals (i.e., who their parents are and where they came from);
- follow the evolutionary processes through time and space, whereas the coalescent usually (but not always) focuses on the final outcome (present-day sampled data);
- follow *multilocus* genotypes;
- model more complex realistic biological demographic events (such as different mating systems or the variation in male and female migration rates) than in a coalescent framework.

1.1 QUICK-START GUIDES FOR SINS

EXPERIENCED LINUX/UNIX USERS - TODO

Make sure you have Java 8 or later already installed.

1. Download the program from https://github.com/PopConGen/SINS/releases/download/v0.0.2/sins.zip.
2. Open a terminal in the folder to where you downloaded the program, everything will be done from here.
3. Decompress it by typing "tar -zxvf SINS_Distribution.tar.gz"
4. Change directory to the folder you just decompressed by typing "cd SINS_Distribution"
5. Type "make", this will run a program to create a pre-built sample input. Two warning messages will be print into the console, don't worry about these.
6. Type "./runSINS.sh", this will run two simulations in parallel for the pre-built input

Congratulations, your first SINS simulation has probably already finished running. The output created by the simulation you ran is ready to be loaded into R. Now explore the input (and output) folder inside the "SINS" folder and read the manual in order to build your own custom simulations.

EXPERIENCED WINDOWS USERS - TODO

Make sure you have Java 8 or later already installed.

1. Download the program from https://github.com/PopConGen/SINS/releases/download/v0.0.2/sins.zip.
2. Use the program 7-zip (or other) to decompress the file you downloaded
3. Open a terminal in the decompressed folder, everything will be done from here.
4. Install Cygwin (or start the program with batch file if you know how)
5. Open Cygwin console and in there type "java -jar SINS_BuildInput.jar"
6. In Cygwin console type "./runSINS.sh", this will run a couple of simulations for the pre-built input

Congratulations, your first SINS simulation has probably already finished running. The output created by the simulation you ran is ready to be loaded into R. Now explore the input (and output) folder inside the "SINS" folder and read the manual in order to build your own custom simulations.

QUICKSTART GUIDE FOR NEW USERS - TODO

1. Check if Java is already installed in your system. One of the ways you can do this is by opening a console/terminal and typing "`java -version`" and clicking the "Return" key on your keyboard. Hopefully you will see something that resembles "`java version "1.8.X_XXX"`". If you don't, proceed to the next point, else, skip it.

2. If you don't have it installed already, please install Java 8 or later from this link or from a later release. Make sure you choose the correct operating system option. If you are using Windows, the option should probably be the one under "Windows x64 Offline". If you are using Linux, choose the last "Linux x64" option. Make sure you also visit this link to read the "JRE" installation guide (JRE, Java Run-time Environment, is the type of Java distribution that you will install). After installing Java, open a console/terminal and type "`java -version`". Hopefully you see something that resembles "`java version "1.8.X_XXX"`". If you do not see this please make sure you installed the correct java version.

3. Download SINS from https://github.com/PopConGen/SINS/releases/download/v0.0.2/sins.zip to your desired folder.

4. Decompress the downloaded file by using your preferred decompression software. For example in Windows you should be able to do this using the software "7zip". For most Linux distributions, you should be able to do this in the terminal by moving to the directory to where you downloaded SINS (e.g. `cd /home/User/Downloads/`) and inserting the following command:

   `tar -xvzf SINS_package.tar.gz` (the actual name of the package, in this case "SINS_package" might vary with future versions).

5. Move to the directory where SINS_package was decompressed (e.g. `cd ~/Downloads/SINS_package`) and type the following command: `make`[1]. You might see some warning messages, don't worry about those for now. The command you typed will create an example input for SINS. You will use this input to run your first SINS simulation. In the future you can use this example input as a basis to create your own custom inputs or you can create a new one from scratch. If at any time you want to reproduce this example input just repeat the procedure in this step.

6. If you want to change any input before running your first simulation, you can do so by changing directories to the input folder and modifying the files therein. If you are already inside the SINS_package folder, then you can go to the input directory by typing the command:

   `cd ./SINS/input/Premade_SINS_Input`[2]

   You can see a comprehensive list of all the inputs set in this particular example input in the tables 1.1,1.2,1.3 and 1.4. You can go to the previous folder by typing:

   `cd ..`

   If you are on the "`Premade_SINS_Input`" folder you can go back to the "`SINS_package`" folder by entering the command "`cd ..`" three consecutive times.

[1]Make is a program that is usually used to compile programs, here we use it simply to run another program in a very simple manner.

[2]In this example input, "Premade_SINS_Input" is the name of your simulation project.

7. You will now run your first SINS simulation by typing the command:

   `"./runSINS.sh"`.

   Remember that to run this command you must be inside the `"SINS_package"` folder. You will see some information being output to the screen. When the simulation is done you will see the following sentence: `"Simulation Premade_SINS_Input has finished, time elapsed: ..."` (hopefully in the time elapsed you have just a couple of seconds). When you see this press "Ctrl + C" to leave this "output viewer".

8. If everything went according to plan (and hopefully according to this guide), you just ran a SINS simulation with 2 replicates (simulated in parallel) for a single layer on a world of 5 by 5 demes. You can now look at the output by changing directories to the output folder. Assuming you are on the `"SINS_package"` folder, you can do this by entering the command:

   ```
   cd ./SINS/output/Premade_SINS_Input
   ```

   To have a better understanding of the output, please read the rest of the manual.

Table 1.1: Quick description of the parameters used in the CMDLINE (command line or terminal), WORLD (world.txt input file) and OUTPREF (output_preferences.txt input file.). For more detailed information please read the rest of the manual

Input			
Group	Name	Value	Brief Description
CMDLINE	Project name	Premade_SINS_Input	Name of your simulation project.
	Number of simulations	2	Number of simulation replicates you will run.
	Show information	true	If simulation information will be output to the screen.
	Output format	adegenet	Format of the SINS output.
	Run in parallel	yes	If the simulations replicates are to be ran in parallel.
	Output folder	./output/	The path to the output folder.
WORLD	Number of generations	100	Number of timesteps or generations in your simulation.
	Number of layers	1	Number of layers or populations in your simulation.
	Layer name	layer0	Name of your layer or population.
	Arrival time	0	Settlers time of arrival.
	Admixture sex ratio	1.0	Read manual for more info.
	Admixture	1.0	Value of admixture between different layers.
	Competition	1.0	Value of competition between different layers.
	Num. environment changes	0	Number of times you will change the environment.
OUTPREF	Rec. demographic output	true	If demographic data will be recorded.
	Rec. summary stats	true	If summary statistics will be recorded.
	Rec. start at	0	Time at which the recording will start.
	Save interval dem.	5	Recording interval for demography. In this example, save demography every 5 generations.
	Save interval gen.	5	Recording interval for genetics. In this example, save genetic information every 5 generations.
	Save interval stats	5	Recording interval for summary stats. In this example, save stats every 5 generations.

Table 1.2: Quick description of the parameters used in the SAMPREF (files found in the sampling_preferences input folder) and ENV (files found in the environment input folder). For more detailed information please read the rest of the manual

Input			
Group	Name	Value	Brief Description
SAMPREF	Num. of configs	3	Number of sampling configurations defined.
	Config. 0	none	Type of configuration. There are 3 types, none, all and subset.
	Config. 1	all	Type of configuration. There are 3 types, none, all and subset.
	Config. 2	all	Type of configuration. There are 3 types, none, all and subset.
	Config. starts at	0 Config. 0	Time at which the configuration will be applied. In this example at t=0, Config. 0 will be applied.
	Config. starts at	10 Config. 1	Time at which the configuration will be applied. In this example at t=10, Config. 1 will be applied.
	Config. starts at	50 Config. 2	Time at which the configuration will be applied. In this example at t=50, Config. 2 will be applied.
	Subset map	Map, see description	Map with number of individuals we want to sample per deme. In this example the subset map is not used. Read manual for more info.
ENV	Initial K	100 (Map)	Map with initial carrying capacity. Read manual for more info.
	Initial F	0.1 (Map)	Map with initial friction. Read manual for more info.

Table 1.3: Quick description of the parameters used in the LAYERPAR (files found in the layer_parameters input folder). For more detailed information please read the rest of the manual

		Input	
Group	Name	Value	Brief Description
LAYERPAR	Initial Layer	Map, see description	Map with number of individuals per deme at t=0. Read manual for more info.
	Growth rate	0.5	Growth rate of the layer or population.
	Migration rate	0.1	Migration rate of the layer or population.
	Migration sex ratio	0.5	Migrants sex ratio, at 0.5 the same number of males and females will migrate.
	Do LDD	lddkernel	LDD = Long Distance Dispersal. Which kind of LDD should be performed. Valid options are "no", "lddkernel" and "method_1". Other options might need to be defined depending on the option selected. In this example lddkernel is selected.
	LDD Lambda	0.1	Proportion of short distance migrants that will do LDD instead. Value taken from a binomial (lddkernel related option).
	Use mean as par.	true	For a gamma distribution use mean as a parameter or use shape instead? In this example "true" is selected and so "mean" will be used (lddkernel related option).
	Mean (or shape)	5.0	Gamme distribution mean. This parameter will define how far individuals migrate (lddkernel related option)
	Variance (or scale)	1.0	Gamma distribution variance (lddkernel related option).
	Min angle	0.0	Minimum angle to define ldd direction (lddkernel related option).
	Max angle	360.0	Maximum angle to define ldd direction (lddkernel related option).
	LDD sex ratio	0.5	LDD migrants sex ratio, at 0.5 the same number of males and females will migrate (lddkernel related option)
	Reproductive Males	1.0	Proportion of the males in the population that will be able to reproduce.
	Reproductive Females	1.0	Proportion of the females in the population that will be able to reproduce.
	Mating system	random	Type of mating system this layer or population will assume. Valid options are "random", "monogamy", "soft_monogamy", "polygyny" or "polyandry".
	Ratio settlers	0.0	Ratio of individuals from another layer or population that will be used to found this population.
	Settlers	layer0	Layer or population that will be used as settlers. In this example we only have one layer and so we input its name here.

Table 1.4: Quick description of the parameters used in the GENETICS (files found in the genetics input folder). For more detailed information please read the rest of the manual

Input			
Group	Name	Value	Brief Description
GENETICS	X length	1	Length of the X chromosome marker. Read manual for more info.
	X type	microsat	Type of the X chromosome marker. Read manual for more info.
	Y length	1	Length of the Y chromosome marker. Read manual for more info.
	Y type	microsat	Type of the Y chromosome marker. Read manual for more info.
	MtDNA length	1	Length of the mitochondrial DNA marker. Read manual for more info.
	MtDNA type	microsat	Type of the mitochondrial DNA marker. Read manual for more info.
	Num. autosomes	1	Number of non-sexual chromosome markers. Read manual for more info.
	A1 length	1	Length of the A1 chromosome marker. Read manual for more info.
	A1 type	microsat	Type of the A1 chromosome marker. Read manual for more info.
	X mut. rate	0.001	X chromosome marker mutation rate. Read manual for more info.
	Y mut. rate	0.001	Y chromosome marker mutation rate. Read manual for more info.
	MtDNA mut. rate	0.001	MtDNA marker mutation rate. Read manual for more info.
	A1 mut. rate	0.001	A1 chromosome marker mutation rate. Read manual for more info.

Chapter 2

Demographic model

Space is assumed to be divided in demes according to a typical 2D stepping-stone structure [6]. The main difference is that in SINS, not all demes are necessarily of the same size and peopled, whereas the original stepping-stone models typically assume that all demes are of the same constant size. SINS allows to simulate different "layers" in the same geographical space, each "layer" allows the simulation of different populations/species within the same physical space, as in Currat and Excoffier [1, 2]. Each deme can exchange migrants, at a certain rate m, with up to four neighbours depending on its geographical location relative to the edges. This means that gene flow is not allowed along the diagonals. Each deme is characterized by carrying capacity (K) and friction (F) parameters, K represents the number of individuals that can (optimally) survive in a given deme with its current resources, while F represents the difficulty to migrate to a given deme, where 0 denotes no barrier to migration and 1 ($0 \leq F \leq 1$) a complete barrier to migration (impossible to migrate to that given deme). Migration is thus constrained by the F value of neighboring demes. Carrying capacity and friction values can be different among demes and layers and can change with time. This means that for a particular layer the user can define areas where the carrying capacity (or friction) is high whereas in others it is low. Importantly, the K and F values can be defined independently within a layer (for different demes/regions) or between layers (for the same region). Density is logistically regulated within each deme, with intrinsic K and growth rate (r), the latter being equal for all demes of the same layer as it is assumed to be species specific (if different species are simulated) or population specific (if different populations of the same species are simulated) [9]. Interaction between layers can be done either by competition and/or admixture. Of course both interactions can be set to zero so that the two layers actually evolve independently[1].

SINS runs the simulation by going through the same steps each generation, with the different demographic events occurring in the following order:

1. Logistic growth with/without competition [2]
2. Migration (i.e. between demes from the same layer)
3. Admixture (i.e. gene flow across layers)

[1]The interaction values between the same layer should be set to 1 for both competition and admixture.

[2]For each generation that the user chooses to save data from, the printing of the results occurs after this step.

2.1 Logistic growth

SINS computes a corrected version of the Maynard-Smith and Slatkin [8] equation for logistic growth (equation 2.1), using the females as limiting factor for population growth [9]:

$$N_{t+1} = 2N_{f,t} \frac{1+r}{1+r\frac{2N_{f,t}}{K}}, \tag{2.1}$$

where N_{t+1} is the total population size at generation $t+1$, $N_{f,t}$ is the number of reproductive females in the deme, at generation t, r is the growth rate and K is the carrying capacity. In addition, foundation events are only allowed if they involve at least one male and one female. Note that this means that if there are no females among the migrants the founding population will go extinct. Growth is not deterministic, as the number of individuals in generation $t+1$ is drawn from a Poisson distribution, with mean $= N_{t+1}$, as given by equation 2.1.

2.2 Migration

2.2.1 Number of migrants

Migration can only take place in four different directions at most. For each deme, the number of individuals that will emigrate is drawn from a Poisson distribution with mean M, where M is given by the following equation,

$$M = N_t m \frac{n_d}{4} \tag{2.2}$$

where N_t is the number of individuals that occupy the deme at time t, m is the migration rate and n_d is the number of neighbouring demes to which it is possible to send migrants. Note that n_d usually varies from a minimum of two, for a deme in one of the corners of the lattice, to a maximum of four. However n_d can also take the minimum value of one, if a deme is surrounded from all but one side, of non viable habitat.

The migrants are distributed stochastically among the neighbouring demes using binomial distributions ($B(P,n)$) based on the following probability:

$$P_{dir} = \frac{1-F_{dir}}{n_d - F_t} \tag{2.3}$$

where dir represents the direction, F_{dir} is the friction of the deme located in the direction dir considered and F_t is the sum of the frictions of the n_d receiving demes. To simulate isolated demes, i.e. all neighbouring demes correspond to non viable habitat, the F parameter in the four cardinal directions should be set to one. To avoid any statistical bias, the order of the migration directions is chosen randomly, for each deme and each generation.

2.2.2 Sex-biased migration

Once the number of migrants is calculated for each direction, a sex-ratio parameter is applied to determine how many males and females will migrate in the different directions. The m_{SR} parameter (equation 2.4) allows the user to vary male (m_m) and female (m_f) migration rates (where ($m_m + m_f = 2m$) and simulate sex-biased migration [9].

$$mSR = \frac{m_f}{m_m + m_f} \tag{2.4}$$

Thus, if m_{SR} values are:

- $= 0.5$, the probability to migrate is the same for males and females;
- > 0.5, females have a higher probability than males to migrate;
- < 0.5, males have a higher probability than females to migrate.

2.3 Interaction between layers

Interaction between layers (i.e. populations/species) can be of two types: competitions and/or admixture. Note that SINS allows both admixture and competition to be either uni- or bi-directional. Note also that they can only take place between demes that correspond to the same geographical region in the different layers.

2.3.1 Competition

The Lotka-Volterra model [7, 11] is used to incorporate competition on logistic growth (equation 2.5), such that for each layer i, the user can define several terms α_{ij} that give the pressure exerted by each deme of layer j over the corresponding deme of layer i. Thus, the size of deme i at time $t+1$ is calculated as:

$$N_{i,t+1} = 2N_{f,i,t} \frac{1 + r_i}{1 + \sum_{j=1}^{n_{layer}} r_i \alpha_{ij} \frac{2N_{f,j,t}}{K_i}}, \tag{2.5}$$

where n_{layer} is the number of layers and $N_{f,i,t}$ and $N_{f,j,t}$ are the number of reproductive females in the deme, at time t, in population i and j, respectively. Note that if there is only one layer or no competition, the α_{ij} terms are either zero ($\alpha_{ij}, i \neq j$) or one (α_{ii}).

2.3.2 Admixture

The number of individuals that can migrate between layers, is calculated using the formula developed in Currat and Excoffier [2]:

$$N_{ij,t} = N_{i,t}\gamma_{ij} \times \frac{2N_{i,t}N_{j,t}}{(N_{i,t} + N_{j,t})^2}. \tag{2.6}$$

where $N_{ij,t}$ is the number of individuals that migrate from layer i to layer j, $N_{i,t}$ and $N_{j,t}$ are the number of individuals in layers i and j, respectively and γ_{ij} is an admixture or interbreeding parameter. These migrants integrate the new deme and take part in the reproduction phase in their new layer. The user needs to provide the admixture parameters (γ_{ij} and γ_{ji}) from layers i to j, from layer j to i, respectively. A value of zero indicates that the two layers do not admix, whereas a value of one means that the two layers exchange migrants as if the were in random mating [2].

Chapter 3

Genetic model

SINS's genetic model is built in a forward framework and can simulate several types of molecular markers (sequences, SNPs and microssatellites). This is done by defining genetic objects and assigning them to each individual. An individual is characterized by different genetic objects:

- Sex chromosomes that define the sex of the individual (XX for female, XY for male);
- Mitochondrial DNA;
- *n* independent non recombining *loci* (two sequences with the same length), to which we also refer to as "autosomes";

At the moment, the main assumptions of the genetic model of SINS are based on the Wright-Fisher model [12, 4]:

- non-overlapping generations (once the new generation is created, the parental one is "eliminated");
- no selection;
- random mating (among reproductive individuals, see Reproduction, the user can choose to have different mating systems).

3.1 Reproduction

SINS is able to simulate different mating systems. Random mating is assumed by default and individuals are generated as described in algorithm 1. Adding to the mating systems, it is possible to simulate the variance in the reproductive success of male and female individuals by giving as input the percentage of reproductive females and males (see chapter SINS organization and Settings). In SINS the mating systems are loosely defined as:

- Random mating - one reproductive individual from each sex is randomly chosen and a reproduction event occurs;
- Monogamy - all the reproductive individuals from each sex are randomly chosen and coupled, reproduction events will only occur within each couple, individuals that were not coupled, will not reproduce;

- Soft monogamy - as Monogamy, but reproductive individuals that were not coupled, will be coupled with an individual of the opposite sex that had already been coupled;
- Polygyny - all reproductive females will be randomly coupled with a reproductive male. Reproduction events occur only within each "couple" or reproduction group. By chance more than one female can be assigned to the same male. By chance reproductive males can be left with no females to reproduce;
- Polyandry - all reproductive males will be randomly coupled with a reproductive female. Reproduction events occur only within each "couple" or reproduction group. By chance more than one male can be assigned to the same female. By chance reproductive females can be left with no males to reproduce.

Algorithm 1 Generating individuals

```
 1: Calculate new number of individuals per deme (Logistic growth)
 2: while the number of individuals is not reached do
 3:     if $matingSystem == RandomMating$ then
 4:         take randomly one reproductive male and one reproductive female of previous a generation;
 5:     else
 6:         take a random previously assembled couple of a previous generation
 7:     end if
 8:     CreateChild
 9:         give child one sex chromosome from each parent
10:         give child mtDNA from the mother
11:         for each "autosome" do
12:             give child one sequence from each parent
13:         end for
14:     EndCreateChild
15: end while
```

3.2 Mutation model

Once the new generation is created, equation 3.1 is applied to males and females independently to calculate the total number of mutations, per deme in the current generation ($N_{mutation}$).

$$N_{mutation} \sim Poisson(\sum_{k}^{marker} \mu_k \times L_k \times \underbrace{(N_{individual} \times P_k)}_{N_{genes}}) \tag{3.1}$$

where μ_k, P_k and L_k are the mutation rate, ploidy and length of the marker k respectively. L_k is the length of the sequence for DNA sequences and 1 for SNPs (SNPs are coded by a string of length one) and microsatellites (which are coded as a single figure representing the number of repeats). $N_{individual}$ represents the number of males or females in the current deme. Note that we could represent $N_{individual} \times P_k$ simply as N_{genes}, the number of genes in the deme. Then, mutations are randomly assigned to individuals. Since sequences and SNPs are coded by strings of 0s and 1s they are assumed to simply mutate from 0 to 1 or vice-versa. For microsatellites, the mutation process is assumed to follow a simple stepwise mutation model (SMM) [5].

Chapter 4

SINS organization and Settings

Due to the individual-based and forward nature of the simulation framework, both demographic and genetic parameters are simulated at the same time. To run a simulation, SINS needs several user-defined settings [1]. They are basically contained in six major classes: *world, environment, genetics, layer parameters, sampling and output* and finally *command line options*. Once a simulation is finished the output is divided in two main sets of files corresponding to the demographic and genetic data. In the following sections, we describe in detail the structure of the input and output files.

4.1 SINS Inputs

The SINS folder structure [2] for a simulation named "mySimulationProject" with two populations or layers named "myLayer_0" and "myLayer_1", and with 3 simulation replicates is shown in figure 4.1.

```
SINS_Package
 ├── (Other files here)
 ├── SINS
 │    ├── SINS.jar
 │    ├── input
 │    │    └── mySimulationProject
 │    │         ├── environment
 │    │         ├── genetics
 │    │         │    ├── myLayer_0
 │    │         │    └── myLayer_1
 │    │         ├── layer_parameters
 │    │         └── sampling_preferences
 │    └── output
 │         └── mySimulationProject
 │              ├── simulation_1
 │              ├── simulation_2
 │              └── simulation_3
 └── SINS_sampler
```

Figure 4.1: SINS folder structure and organization

[1]The input and output files are all in text format

[2]When you download SINS, a sample project can be found in the input directory with the correct folder/file structure.

Figure 4.2 summarizes the file structure for each of SINS input directories. [t] is a 1-based index that identifies an environmental change, [n] is a 1-based index that identifies a given autosome and [p] could be here a number to distinguish between different layers. In practice you can name the layers whatever you want (within reason, no spaces or special symbols) as long as the naming is consistent between input files/folders. Finally [s] could also be a number to distinguish between different sampling/subsetting maps, as with the name of the layers, you can also name these to your own volition.

```
SINS
|__ (Other files here)
|__ SINS.jar
|__ input
|   |__ mySimulationProject
|       |__ world.txt
|       |__ output_preferences.txt
|       |__ environment
|       |   |__ myLayer_[p]CCInit.txt
|       |   |__ myLayer_[p]CC[t].txt
|       |   |__ myLayer_[p]FInit.txt
|       |   |__ myLayer_[p]F[t].txt
|       |__ genetics
|       |   |__ myLayer_[p]
|       |       |__ genotype.txt
|       |       |__ allelesX.txt ............................................ Optional
|       |       |__ allelesY.txt ............................................ Optional
|       |       |__ alllelesMT.txt .......................................... Optional
|       |       |__ alllelesA[n].txt ........................................ Optional
|       |__ layer_parameters
|       |   |__ myLayer_[p]_init.txt
|       |   |__ myLayer_[p].txt
|       |__ sampling_preferences
|           |__ sampling_conf.txt
|           |__ subset_map[s].txt
|__ output
```

Figure 4.2: SINS file structure and organization

Although this is usually not necessary, if you choose to also use SINS_sampler, the SINS program that allows for more complex sampling, then its folder and file structure is shown in figure 4.3.

For more information regarding SINS_Sampler, see chapter [INSERT FUTURE SAMPLER CHAPTER]

Let us now describe each input file independently per folder. Before we begin, note that **all input files for SINS have a specific layout and both the order and number of parameters are important**.

```
SINS_main_directory
├── SINS
└── SINS_sampler.......................................Don't worry about the other files here
    ├── SINS_sampler.jar
    ├── input
    │   └── myProjectFolder
    │       ├── config_myProjectFolder.txt
    │       ├── generations.txt
    │       ├── scriptCreateSamplingFiles.py
    │       └── SamplingGenFiles
    │           └── sampling[i].txt
    └── output
        └── mySamplingProcess
            ├── myProjectFolder
            └── simulation_[z]
                ├── [i]_myLayer_[p]_Y
                ├── [i]_myLayer_[p]_MT
                └── [i]_myLayer_[p]_A[n]
```

Figure 4.3: SINS_sampler folder structure and organization

4.1.1 The world.txt file

Looking at figure 4.2 we can see that inside the input folder and inside our simulation project folder we have two files, world.txt and output_preferences.txt. The world.txt file has some of the main parameters defining the world to be simulated (number of generations, number of layers, environmental changes). The output_preferences.txt file is used by SINS to determine the type of output the user wishes. For instance, for a 10 by 10 world, $K = 200$ and choosing to save every individual in the population, means saving around 20000 individuals per generation for every simulated genetic marker. This means that files can get huge. Of course a smart sampling/saving scheme can greatly improve this, more on that below.

For now lets look at two examples of the world.txt file, one for a simulation with one layer and no environmental changes and one for a simulation with two layers, environmental changes and different expansion times. In the example in listing 1, SINS will simulate one layer (named myLayer_0), for a total of 2000 generations. The time at which the initial or founding population starts to expand is generation 0. No environmental events (i.e. changes in K or F values) were simulated. The simulations will therefore only use the initial K and F maps provided by the user in the environment folder (see subsection 4.1.2). Since only one layer is simulated no interaction between layers is possible and the admixture and competitions parameters have to be set to one, i.e. SINS uses a simple logistic growth model.

Listing 1 Example of a world.txt file, for a one-layer scenario.

```
numberOfGenerations 2000
numberOfLayers 1
layerName0 myLayer_0
expansionTime0 0 // Start expansion from the beginning (T=0).
admixture00 1.
competition00 1.
numberOfEnvironmentalChanges 0 // No environmental changes.
```

Note: Comments in this example appear after the "//" sign and **would not appear in the actual file**.

In the example in listing 2, SINS will simulate two layers named alien and predator for a total of 100 generations. The populations (or species) living in each layer will start to colonize their layers at generation times 0 and 30, respectively. In other words, the second layer, predator, is empty during the first 30 generations of the simulation. In addition, two environmental events are simulated. The K and F maps change at generations 50 and 60. To do that the user must define three different sets of K and F maps, that must be provided in the environment folder (subsection 4.1.2). In this example , admixture is unidirectional and the admixture parameter (from alien to predator) is set to 0.5, whereas admixture in the other direction is set to zero. In this example, aliens and predators do not compete, so the competition in both directions is set to zero. Within layers, the admixture and competition parameters are set to one (i.e. no competition and full admixture).

Algorithm 2 shows how you can create a world.txt file. Note that the number of parameters depends on the number of layers and on the number of environmental fluctuations/changes, i.e. how many times the user wants to change the K and F maps. **The layer names specified in the world.txt file are important as they are used to define the environmental, genetic and layer parameters.**

Listing 2 Example of a world.txt file, for a two-layer scenario.

```
numberOfGenerations 100
numberOfLayers 2
layerName0 alien
expansionTime0 0
layerName1 predator
expansionTime1 30
admixture00 1.
admixture01 0.5
admixture10 0.
admixture11 1.
competition00 1.
competition01 0.
competition10 0.
competition11 1.
numberOfEnvironmentalChanges 2
envChangeTime1 50
envChangeTime1 60 // Two changes, one at T=50 and another at T=60
```

Note: Comments in this example appear after the "//" sign and **must not appear in the actual file**.

Algorithm 2 How to create a world.txt file. Where a red # symbol is present, your own value should be inputed.

```
numberOfGenerations # // Total number of generations
numberOfLayers # // Total number of layers
for p = 0 to p = (numberOfLayer-1):
  layerName<p> # // Name of layer p
  expansionTime<p> # // Start of expansion time for layer p (in generations)
for i = 0 to i = (numberOfLayers-1):
  for j = 0 to j = (numberOfLayers-1):
    admixture<ij> # // admixture parameter $\gamma_{ij}$, from layer i to layer j
for i = 0 to i = (numberOfLayers-1):
  for j = 0 to j = (numberOfLayers-1):
    competition<ij> # // competition parameter $\alpha_{ij}$, pressure exerted by layer i on j
numberOfEnvironmentalChanges #
for e = 1 to e = numberOfEnvironmentalChanges:
  envChangeTime<e> # // Environmental change time (in generations)
```

Note: Comments in this example appear after the "//" sign.
Possible parameter values: numberOfSimulations $\in [1,\infty[$; numberOfGenerations $\in [1,\infty[$; numberOfLayers $\in [1,3]$; layerName<p> $\subset alphanumerical-string$; expansionTime<p> $\in [0, numberofGenerations]$; admixture<ij> $\in [0,1]$; competition<ij> $\in [0,1]$; numberOfEnvironmentalChanges $\in [0, numberofGenerations]$; envChangeTime<e> 1 $\in [0, numberofGenerations]$

4.1.2 The environment folder

This is where the user specifies the regions with high or low population densities, the areas that are easy or difficult to cross. The environment folder contains K (carrying capacity) and F (friction) maps (figure 4.2), for each layer and each environmental event. The maps are in a rectangular matrix format, but inside these the environment can actually take any shape.

At the start of the simulation SINS will use the files named <nameOfMyLayer>CCInit.txt and <nameOfMyLayer>FInit.txt to defined K and F maps, respectively ("CC" stands for carrying capacity, "F" stands for friction). If environmental change events are defined in the world.txt file (see above), additional sets of K and F maps are required for each event. These maps are in files named <nameOfMyLayer>CC[e].txt and <nameOfMyLayer>F[e].txt, where e = [1 to total number of environmental changes], see algorithm 2. Note that even if the user decides to only change, say the K map while keeping the F values constant, s/he will still need to have the same number of K and F map files with the appropriate names.

Thus, for the example in listing 2, 12 files would be required in the environmental folder as can be seen in figure 4.4.

```
input
  └─mySimulationProject
      ├─environment
      │   ├─predatorCCInit.txt
      │   ├─predatorCC1.txt
      │   ├─predatorCC2.txt
      │   ├─predatorFInit.txt
      │   ├─predatorF1.txt
      │   ├─predatorF2.txt
      │   ├─alienCCInit.txt
      │   ├─alienCC1.txt
      │   ├─alienCC2.txt
      │   ├─alienFInit.txt
      │   ├─alienF1.txt
      │   └─alienF2.txt
      └─...
```

Figure 4.4: Environment folder structure for the example in listing 2

If there are two layers, but no events there would only be 4 files (the "(...)Init.txt" files). Also, note that some of these maps may be identical, for instance if one keeps friction constant while changing only carrying capacities, or vice-versa. If we have a simple 5 by 5 (-demes) layer, then an example of possible K and F maps is shown in listing 3. This should be trivial but note that all the maps have to be of the same size, that is carrying capacity maps, friction maps, the demography maps denoting the number of initial individuals in a layer as well as the sampling maps. A common mistake is for a user to create a given map, and then by mistake create a similar but off by 1-column/row map. This will likely give you an error or incorrect results.

Note that for example if a deme A is surrounded by demes $B_{1,2,3,4}$ all with the same friction values F_B, then the number of migrants sent in each direction (B_1, B_2, B_3 and B_4) is independent of the friction value of the neighboring demes (F_B). This means that F_B could have any value $[0,1[$ and the number of migrants sent in each direction would still be the same. Also note that the friction value, F_A, for the current deme where the individuals are, A, does not influence where the individuals are sent (see subsection 2.2.1).

Listing 3 Example of carrying capacity **(a)** and friction map **(b)** files.

(a) A small 5 × 5 carrying capacity map that could be used as a <nameOfMyLayer>CCInit.txt or <nameOfMyLayer>CC[*e*].txt file. Demes that have a carring capacity of zero will not support any life in them, if an individual migrates there, s/he will die. In this example there is a small 2 × 2 island on the bottom right of the map that is isolated from the rest of the world since it is surrounded by demes that have a carrying capacity of zero.

```
100  100  100  100  100
100  100  100  100  100
100  100    0    0    0
100  100    0  100  100
100  100    0  100  100
```

(b) A small 5 × 5 friction map that could be used as a <nameOfMyLayer>FInit.txt or <nameOfMyLayer>F[*e*].txt file. Note that although the possible Friction values vary between 0 (easy to migrate to this deme) and 1 (impossible to migrate to this deme), we usually use negative numbers to denote demes where individuals can neither live in nor migrate to. This is just for illustrative purposes but you can see the -99 value below as the representation of a body of water or a road for a small terrestrial species, and the value of 1 on the top left as a large geological elevation where individuals can come from but never go into. While both values technically do the same in SINS, when working on large maps it makes it easier to spot mistakes.

```
1.0  0.1  0.1  0.1  0.1
0.1  0.1  0.1  0.1  0.1
0.1  0.1  -99  -99  -99
0.1  0.1  -99  0.1  0.1
0.1  0.1  -99  0.1  0.1
```

Note: Although in this example we use different amounts of spaces to align the map (so that we can visualize it better), you should really use just one space (or tab, but don't mix both in the same file, that is usually a recipe for disaster).
Possible parameter values: Carrying capacity $\in [0,\infty[$; Friction $\in [0,1]$

4.1.3 The genetics folder

This is where the user defines the genetic markers to be simulated and the initial genetic diversity. This folder contains subfolders named according to the <nameOfMyLayer> defined in the world.txt file, each with at the very least, a genotype.txt file (see algorithm 3). The genotype.txt file contains information about the genetics markers to be simulated (listing 4). When no initial allele frequency files are given (figure 4.2) the simulation will start with no genetic diversity (all loci will be monomorphic, and all individuals identical across all markers). Optionally, files can be provided by the user (figure 4.2) specifying the initial allele frequencies of the markers simulated (listing 5). If the genetic make-up of founding populations is taken from pre-specified allele frequencies, the user can obtain it from observed or simulated data. When several layers are simulated, it is also possible to found a new population by sampling the corresponding deme from another layer (see subsection 4.1.4 below).

Listing 4 Example of a genotype.txt file. This is where the user defines the type of markers ("microsat" for microsatellites, "sequence" for sequences and "SNP" for SNPs), their length (usually 1 for independent SNPs or microsatellites) and mutation rates. The mutation rate is *per site* for sequences, while it is *per locus* for SNPs and microsatellites. In this example, SINS will simulate two independent loci (on independent "autosomes"), one being a microsatellite and the other a SNP. Moreover, the X and Y-chromosomes will also harbor one microsatellite each. Finally a 10 bp sequence will be simulated for mtDNA. **Note that the X and Y-chromosomes must always be simulated because they define the sex of individuals, but they can be of any type. MtDNA is also always simulated. If the user is not interested in these markers for genetic analysis, a length and mutation rate of zero will be best to reduce computation costs. If a length of zero is defined for these markers they will also be excluded from the genetic output**.

```
lengthX 1
typeX microsat
lengthY 1
typeY microsat
lengthMtDNA 10
typeMtDNA sequence
nbAutosomes 2
lengthA1 1
typeA1 microsat
lengthA2 1
typeA2 SNP
mutRateX 0.01
mutRateY 0.01
mutRateMtDNA 0.00005
mutRateA1 0.05
mutRateA2 0.0001
```

In SINS, sequences and SNPs are represented (listing 5) using binary notation (i.e. sequences of 0s and 1s). Only in the case where the populations starts with no diversity do we use the 0 as the ancestral allele (because all markers are set to 0) but otherwise, 0 and 1 do not represent the ancestral and derived status, respectively. Microsatellite alleles are represented by a single number corresponding to the number of repetitions. Note that this number can be negative without any effect on the simulations, however, do not do this, since (i) there is no biological meaning in this and (ii) the programs that you might use to analyze the data might not allow for negative numbers. Therefore we advise users to use large positive integers, for

Algorithm 3 How to create a genotype.txt file. Where a red # symbol is present, your own value should be inputed.

```
lengthX # // length of the X chromosome marker
typeX # // type of the X chromosome marker
lengthY # // length of the Y chromosome marker
typeY # // type of the Y chromosome marker
lengthMtDNA # // length of the MtDNA chromosome marker
typeMtDNA # // type of the MtDNA chromosome marker
nbAutosomes # // total number of autosomes
for a = 1 to a = nbAutosomes + 1:
  lengthA<a> # // length of the autosome a marker
  typeA<a> # // type of the autosome a marker
mutRateX # // X chromosome mutation rate
mutRateY # // Y chromosome mutation rate
mutRateMtDNA # // mtDNA chromosome mutation rate
for a = 1 to a = nbAutosomes + 1:
  mutRateA<a> # // autosome a mutation rate
```

Note: Comments in this example appear after the "//" sign.
Possible parameter values: length[X,Y,MtDNA] $\in [0,\infty[$; type[X,Y,MtDNA] $\in$ [”*microsat*”,”*sequence*”,”*SNP*”]; nbAutosomes $\in [0,\infty[$; lengthA<a> $\in [1,\infty[$; typeA<a> $\in$ [”*microsat*”,”*sequence*”,”*SNP*”]; mutRate[X,Y,MtDNA,A<a>] $\in [0,1]$

example 500, to limit the risk of ever generating negative length alleles. The mutation model assumed is the SMM (stepwise mutation model) in which one repeat is added or subtracted with equal probability.

We should mention that currently, in SINS, the difference between a SNP and a sequence is more conceptual than technical. These two types of markers are effectively "coded" the same way, however depending on what the user wants to simulate, the interpretation is different. A "sequence" represents a sequence of nucleotides. A SNP, represents a single nucleotide polymorphism. However the user might want to represent, for example, three completely linked SNPs. These look exactly how we represent sequences, for example:

$$”0 \quad 1 \quad 1”$$

however these could probably be better represented as:

$$”0------1--------------------------------1”$$

where for a long sequence of base pairs, only the polymorphic sites are represented. The user can also choose to simulate fully linked microsatellites on the same marker by defining a "sequence of microsatellites" as such

$$”500 \quad 500 \quad 500”$$

however remember that currently the mutation rate is the same for all linked loci, we will likely change this in a future version.

If we follow the example in listing 2, then a possible genetics folder would look like the one in figure 4.5. Note that for predator we do not specify any allele file, therefore, all the individuals from the predator population/species will start the simulation with no genetic diversity for any of the markers. For the alien population/species, we do not specify the alleleX.txt and the alleleY.txt so, for these markers, these

Listing 5 Example of different allele files for **(a)** sequences, **(b)** microsatellites and **(c)** SNPs. If we take the genotype.txt in listing 4 as an example, then **(a)** could represent the allelesMT.txt file. It provides the allele (haplotype) frequencies for mtDNA in the case where three different alleles were present at the start of the simulations, with frequencies 0.3, 0.6 and 0.1, respectively. Note that the length of the sequence is 10, as it must be equal to the value defined in the genotype.txt file. Following the same example, **(b)** can represent any of the X, Y or alleleA1.txt files and finally **(c)** can represent the alleleA2.txt file.

(a) an allele file for a marker of the type "sequence" that represents a sequence.

```
nbAlleles 3
0.3 0 0 0 0 0 1 0 0 1 0
0.6 0 0 0 0 1 1 0 0 0 0
0.1 1 1 1 0 0 1 0 0 1 0
```

(b) an allele file for a marker of the type "microsat" that represents a microsatellite.

```
nbAlleles 5
0.8 500
0.05 501
0.05 502
0.05 499
0.05 498
```

(c) an allele file for a marker of the type "SNP" that represents a SNP.

```
nbAlleles 2
0.5 1
0.5 0
```

individuals will also start the simulation with no genetic diversity. Finally, also note that you can specify an allele file and still have no genetic diversity if in this file you specify that there is only one allele, therefore its frequency is one.

Algorithm 4 How to create an allele[m].txt file. Where a red # symbol is present, your own value should be inputed.

```
nbAlleles # // number of alleles for the current marker
for i = 0 to i = nbAlleles:
  <# i frequency> <# i haplotype> // the allele frequency should add up to 1.0
```

Note: Comments in this example appear after the "//" sign.

```
input
└── mySimulationProject
    ├── genetics
    │   ├── alien
    │   │   ├── genotype.txt
    │   │   ├── allelesMT.txt
    │   │   ├── allelesA1.txt
    │   │   └── allelesA2.txt
    │   └── predator
    │       └── genotype.txt
    └── ...
```

Figure 4.5: Genetics folder structure for the example in listing 2

4.1.4 The layer_parameters folder

This is where the characteristics of the layers and the populations/species inhabiting them are defined. In the <nameOfMyLayer>.txt file the user can set parameters such as growth and migration rates, long distance dispersal/migration (LDD) rates, sex-biased migration, mating systems and percentage of reproductive males and females. Listings 6 and 7 show examples of <nameOfMyLayer>.txt files without and with LDD respectively.

Growth rate ("growthRate") is the parameter that regulates the logistic growth that will determine the population size. Migration rate ("migrationRate") defines the portion of a population in a deme that will migrate, for example for a deme with population size $K = 100$ and migration rate $m = 0.1$, we will have 10 migrants which will be distributed with equal probability for all neighboring demes (if the friction values for these neighbours are all the same). The migration sex ratio ("migrationSexRatio") parameter defines the ratio of female to male migrants, if it has a value of 0.5 then both sexes migrate with the same ratio, if it has a value > 0.5, then females will migrate more than males and finally if < 0.5, then males will migrate more than females.

In SINS there are two distinct Long Distance Migration (LDD) implementations, this is likely to change in a future version. The user can choose not to have any kind of LDD or to use one of the two implementations. One of the implementations ("lddkernel") is based on a paper by N. Ray and L. Excoffier[3], while the other implementation is based on our own interpretation of what LDD is. Note that this is a very experimental feature and as such it is very likely to change in the future. The two implementations differ in the following way.

The first implementation ("lddkernel"), defines that:

- There will be a portion of the short distance migrants of a deme that will actually perform a long distance migration instead. This means that, potentially, if every deme has more than one male and female, all demes might have long distance migrants.
- The distance that a long distance migrant will move is defined by a Γ distribution for which the mean and variance (or shape and scale) are defined by the user.

The second implementation ("method_1") defines that:

- There will be a pool of long distance migrants that will be independent of the usual pool of (short distance) migrants.
- There will be LDD events with a given rate per generation, that is, per generation, a number of demes (that have more than one male and female) will be randomly selected given a certain rate to have LDD events.
- The distance that a long distance migrant will move is defined by a *Poisson* distribution for which the mean is defined by the user.

There are many idealogical problems with these implementations. For example, one issue with long distance dispersal is that a movement "event" can fall out of our "universe" or layer, or it can fall in a place to where individuals cannot migrate to in any possible way. When this happens we need to decide how the

[3] doi: 10.1111/j.1755-0998.2010.02881.x

LDD event will be resolved. In this particular case, we fixed it in both implementations by trying to complete this LDD event 100 times, if that fails, then the LDD event will not happen even though it was chosen to. This fix is problematic because it will skew the number of times LDD events will actually happen. As we've said above, this feature is very experimental so use it with caution.

Listing 6 Example of a <nameOfMyLayer>.txt file in the layer_parameters folder without LDD. In this file, the parameters of the layer are chosen by the user and by definition assumed to be equal for all demes. Note that the migration is defined for the layer but is modified locally by the F values which determine where the migrants will preferential move. Moreover remember that since we are defining a migration rate here, the actual number of migrants will vary with the number of individuals currently present in a deme. The proportion of males and females reproducing is set to 1 in this example, but the user can modify one of these values to create a differential in reproductive success between males and females. In principle one can modify the two values but this would be inefficient. Furthermore these values can be coupled with the mating system parameter in order to replicate more complex mating systems.

```
growthRate 0.8
migrationRate 0.2
migrationSexRatio 0.5
doLongDistanceDispersal no
reproductionSuccessMale 1.0
reproductionSuccessFemale 1.0
matingSystem monogamy
ratiosSettlers 0.
settlers alien
```

Listing 7 Example of a <nameOfMyLayer>.txt file in the layer_parameters folder with LDD.

(a) lddkernel

```
growthRate 0.8
migrationRate 0.2
migrationSexRatio 0.5
doLongDistanceDispersal lddkernel
lddLambda 0.1
useMeanAsParameter true
mean_shape 4
variance_scale 1
minAngle 0
maxAngle 360
lddSexRatio 0.5
reproductionSuccessMale 1.0
reproductionSuccessFemale 1.0
matingSystem monogamy
ratiosSettlers 0.3
settlers predator
```

(b) method_1

```
growthRate 0.8
migrationRate 0.2
migrationSexRatio 0.5
doLongDistanceDispersal method_1
ldMigrationEventRate 0.1
ldMigrationRate 0.1
meanPoissonDistance 4
minAngle 0
maxAngle 360
lddSexRatio 0.5
reproductionSuccessMale 1.0
reproductionSuccessFemale 1.0
matingSystem monogamy
ratiosSettlers 0.3
settlers predator
```

Algorithm 5 How to create a <nameOfMyLayer>.txt file. Where a red # symbol is present, your own value should be inputed.

```
growthRate # // Growth rate for this layer/population
migrationRate # // Migration rate for this layer/population
mSR # // Migration sex-ratio for this layer/population
doLongDistanceDispersal # // Option that defines if/which LDD implementation will happen
if(doLongDistanceDispersal == "lddkernel"){
  lddLambda # // Proportion of short distance migrants that will do LDD
  useMeanAsParameter # // Which pair of parameters to use in the Γ distribution
  mean_shape # // Mean or shape value for the Γ distribution
  variance_scale # // Variance or scale value for the Γ distribution
  angleSide1 # // Side of the angle, zero points to the right/east
  angleSide2 # // Side of the angle, zero points to the right/east
  lddSexRatio # // LDD sex-ratio for this layer/population
}else if(doLongDistanceDispersal == "method_1"){
  ldMigrationEventRatio # // Ratio of demes that should have LDD events
  ldMigrationRate # // Long distance migration rate for this layer/population
  meanPoissonDistance # // Mean of the Poisson distribution
  angleSide1 # // Side of the angle, zero points to the right/east
  angleSide2 # // Side of the angle, zero points to the right/east
  lddSexRatio # // LDD sex-ratio for this layer/population
}
reprodMale # // Proportion of males that will be able to reproduce in this layer/population
reprodFemale # // Proportion of females that will be able to reproduce in this layer/population
matingSystem # // Mating system that will be in effect in this population
ratiosSettlers # // Ratio of a population sampled to create this population
settlers # // Which population we will be sampling from
```

Note: Comments in this example appear after the "//" sign.
Possible parameter values: growthRate $\in [0,1]$; migrationRate $\in [0,1]$; mSR $\in [0,1]$; doLongDistanceDispersal $\in$ [$"no", "lddkernel", "method\_1"$]; lddLambda $\in]0,1]$; useMeanAsParameter $\in [true, false]$; mean_shape $\in [0,\infty[$; variance_scale $\in [0,\infty[$; ldMigrationEventRatio $\in [0,1]$; ldMigrationRate $\in [0,1]$; meanPoissonDistance $\in [0,\infty[$; angleSide1 $\in [-360,360]$; angleSide2 $\in [-360,360]$; lddSexRatio $\in [0,1]$; reprodMale $\in [0,1]$; reprodFemale $\in [0,1]$; matingSystem $\in$ [$"random", "monogamy", "soft\_monogamy", "polygyny", "polyandry"$]; ratioSettlers $\in [0,1]$; settlers $\subset alphanumerical - string$

4.1.5 The output_preferences.txt file and the sampling_preferences folder

The output_preferences.txt file and the sampling_preferences folder and files therein are coupled in a subsection because they do things that are related and that actually interact with each other. The output_preferences.txt file is where the user defines the time intervals at which the genetic, demographic data and summary statistics are saved. The user can choose to save different kinds of data in different time intervals. In listing 8, we start recording at $T = 0$ generations, saving data for all three categories (demographic data, genetic data and summary statistics) every 10 generations, then at generation $T = 100$ we start recording every 100 generations until the end of the simulation. Remember that the more often you save data, the more disk space you will need and the longer the program will run.

Algorithm 6 shows how you can create an output_preferences.txt file. The number of times the "while" loop is repeated is completely up to the user.

In the sampling_preferences folder the user can, as the name implies, define options that will help on the sampling of the (genetic) data from the simulations. These options will be mostly defined in the

Listing 8 Example of an output_preferences.txt file.

```
RecordDemographicData true
RecordSummaryStats true
recStarts 0
demographyTimeInterval 10
geneticsTimeInterval 10
summaryStatsTimeInterval 10
recStarts 100
demographyTimeInterval 100
geneticsTimeInterval 100
summaryStatsTimeInterval 100
```

Algorithm 6 How to create an output_preferences.txt file. Where a red # symbol is present, your own value should be inputed.

```
RecordDemograpicData # // true/false
RecordSummaryStats # // true/false
while userDefinedRecordChanges:
  recStarts # // At which generation the recording starts/changes
  demographyTimeInterval # // interval at which demographic data should be saved
  geneticsTimeInterval # // interval at which genetic data should be saved
  summaryStatsTimeInterval # // interval at which summary statistics should be saved
```

Note: Comments in this example appear after the "//" sign.
Possible parameter values: RecordDemographicData $\in [true, false]$; RecordSummaryStats $\in [true, false]$; recStarts $\in [0, numberOfGenerations]$; demographyTimeInterval $\in [1, numberOfGenerations]$; geneticsTimeInterval $\in [1, numberOfGenerations]$; summaryStatsTimeInterval $\in [1, numberOfGenerations]$

sampling_conf.txt file. However these options interact with the ones defined in the output_preferences.txt, limiting them. What we mean by this is that if you define that you want to save data every 10 generations, starting from $T = 0$, then in the sampling_conf.txt file you cannot change how often you will be able to sample your individuals, however you can choose **not** to save them here. This means that if in the output_preferences.txt file you define that you want to save the genetic data every generation for a simulation with $T = 100$ generations, in the sampling_conf.txt file you can decide that, for (a purely illustrative) example, from $T = 0$ to $T = 50$ you don't want to save nor sample any individual, from $T = 50$ to $T = 75$ you can choose to save all the individuals, and then from $T = 75$ to $T = 100$ you can choose to save only a subset, a sample of individuals, from your whole simulation. You do this subsetting by creating a map (like the K and F maps) with the number of individuals and the respective demes that you want to save. Listing 9 shows an example of both a sampling_conf.txt file and possible subset maps. Remember that the number of individuals saved will always be divided by 2, half males, half females, and that the subset maps have to be of the same size as all the other maps for a given simulation/scenario.

Note that this system also supports having different sampling schemes for different layers. In fact if you have more than one layer, you will have to specify what are the different schemes for the different layers. This is simply done by adding the desired subset configuration for a given layer after the previous one. That is, for example, if we have two layers, a possible set of subset configurations would be:

```
subset_config3 subset subset_all.txt all
subset_config4 all none
```

where `subset subset_all.txt` of `subset_config3` corresponds to the first layer and `all` of `subset_config3` corresponds to the second layer, whereas `all` of `subset_config4` corresponds to the first layer and `none` of `subset_config4` corresponds to the second layer.

4.1.6 The command line parameters

The command line parameters that you can provide to SINS define to a great extent how your simulation project will be computed and processed. Here you will define for example, how many replicates your simulation will have and if these are going to be computed in parallel or linearly, this is one of the options that greatly improves SINS run-time. In the command line you can type:

```
java -jar SINS2.jar -help
```

to get a list of the command line options. Depending on the version of SINS that you are working with these can be different. In tables 4.1 and 4.2 you can see the current relevant options and their descriptions. For simplicity and to better streamline your work, we advise you create a small script (like the runSINS.sh script we provide with the SINS_package) to launch your simulation projects. In fact you can (and perhaps should) modify the runSINS.sh script and customize it to your workflow. If you are running SINS in a server or cluster, we also recommend that you use and get comfortable with any kind of software that allows you to run a simulation for long periods of time without having to care about turning your console/terminal off. Some of these softwares are "tmux", "screen" and "nohup". We like using "tmux", but for a beginner "nohup" is probably easier to use from the start.

Listing 9 Example of the files you could have in the sampling_preferences folder. The sampling_conf.txt file **(a)** must exist. The subset maps, like **(b)** and **(c)**, are optional according to what you want to do. When sampling the individuals note that the number that you input in the map is the total number of individuals that the program will try to save/sample. The program always assumes that the user wants to save half of the inputed number as males and half of the inputed number as females.

(a) Example of a sampling_conf.txt file. This file allows you to define how and partially when you want to save your genetic data. The subset configurations allowed are "none", if you don't want to save any genetic data, "all" if you want to save all your genetic data (all your individuals) and "subset" which requires a follow-up argument, the name of a map file where the number and position of the individuals that you want to save is defined.

```
NumberOfConfigs 4
subset_config1 none
subset_config2 all
subset_config3 subset subset_all.txt
subset_config4 subset subset_map.txt
timeline
startAt 0 subset_config1
startAt 10 subset_config2
startAt 50 subset_config1
startAt 60 subset_config3
startAt 70 subset_config4
```

(b) Example of a subset map where we try to save all the individuals of a few selected demes. We do this by setting the number of individuals that we want to save as a much higher number than K. If there are no individuals in a selected deme, no individual will be saved.

```
10000    0     10000    0     10000
0        0     0        0     0
10000    0     10000    0     10000
0        0     0        0     0
10000    0     10000    0     10000
```

(c) Example of a subset map where we try to save and sample only 10 individuals of a few selected demes. If there are more than 10 individuals in the selected demes, then the program will randomly select 10 of them (5 males and 5 females). If there are less than 10, then as in the previous example the program will save them all (if we have exactly 5 males and 5 females). In the case where we have an imbalanced sex-ratio in the deme, for example 4 males and 10 females, the program will save 9 individuals (4 males and 5 females).

```
10  0  10  0  10
0   0  0   0  0
10  0  10  0  10
0   0  0   0  0
10  0  10  0  10
```

Table 4.1: All the command line arguments that the user should be using when running SINS.

Command line arguments				
Name	Tags	Valid value	Example	Description
Number of simulations	-numberOfSimulation -numberOfSimulations -nSim	1-Inf	20	Number of simulation replicates you want to run. Theoretically there is no limit to this number, in practice it really depends on how powerful your machine is and in how many replicates you need to be able to answer you biological question.
Project name	-projectName -pjName	string	mySimulation	The name of your simulation project. This name will also define the name of your target folders inside the input and output folders (the output folder is defined by the user too, see below). This means that when you try to run a simulations, this parameter needs to have the same name as the folder inside your input folder where you've built your input. After the simulation has ran, a folder with this name will be created inside the output folder.
Output folder	-of -outfile -outDir	string	./output/	The general output folder for your simulations. This can be a full or a relative path. The output folder for a given simulation will be inside this folder with the name defined in the "-projectName" option. If you do not specify this folder, then SINS will automatically create it inside the folder where it is running and name it "results"
Compression format	-compFormat -outputCompFormat -compress	noComp fZip rZip bCZip	noComp	Specification of the compression format. The "noComp" option, specifies no compression, we usually use this option.
Simulation replicate parallelization	-parallel	yes no	yes	Specifying if the simulation replicates will be ran in parallel or not. If your machine has more than 1 core (which most machines do) and if its architecture allows for thread parallelization (which most machines do), turn this option on, it greatly improves the speed of computation.
Number of cores	-parallelCores -parCores	number	4	Specifying the number of cores the simulation will use if the "-parallel" option was set to "yes". If this option is not set or if the number of cores specified is superior to the number of cores in the machine, the program will use all the cores that are available. Depending on where you are running this (personal computer vs server) we advise you to carefully define this option, so that it doesn't monopolize all the resources available in the machine. If the "-parallel" option was set to "no", then this option has no effect and the program will run single-threaded.

Table 4.2: Continuation of table 4.1

Command line arguments				
Name	Tags	Valid value	Example	Description
Verbosity	-v -verbose	boolean	false	Specifying if information on the status of the simulation is output to screen. We recommend using this option while testing or doing small scale simulations but turning it off on large scale simulations improves the speed of the simulation.
Output format	-outputFormat	sins adegenet	adegenet	Specifying the output format of your simulations. Both formats are easily loadable into R. The "sins" option shows a bit more of information but the "adegenet" format shows the information that the "adegenet" software needs to build its data structures. The SINS_package also comes with an R script that is capable of loading the SINS output data directly into "adegenet" data structures so that the user can easily analyze their own data.

Chapter 5

SINS outputs and data analysis

5.1 SINS outputs

SINS produces demographic and genetic outputs which can be tailored to suit the user's wishes and needs. Simple summary statistics can also be produced. All outputs are recorded inside the user defined output folder, inside yet another folder with the name of the simulation project. If they don't yet exist, these folders are created when the user starts a simulation. Each simulation project has its own folder called "simulation_<i>" where <i> denotes the number of the simulation replicate. Figure **??** shows the usual structure of a SINS output folder (take another look at figure 4.1 to check the overall structure if you need). Alternatively, a single file (with the termination ".db") per replicate can be created if the output compression format chosen is "SQLdb". These files consist of independent and standalone SQLite databases with all (and more) information than its text counterparts. The advantage of using this format is that not only is it smaller than outputting to text, it also makes SINS run slightly faster and it makes the process of working with large datasets on R/RStudio much more manageable since you can choose which parts of the database/table you want to import instead of having to load all the text data into R.

5.1.1 Demographic output

The demographic output is saved in a single text file, named "demography.txt". This file contains, for the generations specified by the user in the appropriate input file, the number of individuals recorded for each deme and layer (listing 10). Thus, depending on the user's input files options, these number can be saved every 10, 50, 500 generations or for all generations or for whatever interval the user sees fit. Beware that depending on the options specified this file can be very large. The information in this file can for example be used to generate plots of how the population sizes are varying over time.

5.1.2 Genetic output

For each layer, the genetic outputs are divided by chromosome/*locus*. For each *locus* one file is created with the genotypes of all individuals for all demes and time steps pre-specified by the user. Depending on the format chosen by the user (as a command line argument option with the tag "-outputFormat"), these files will have mostly the same information but formated in slightly different ways. In figure 5.2 you can see how the "sins" format and "adegenet" format columns differ. The files containing the genetic output from SINS are the main output that the program produces. It is from these files that you will be able to produce the genetic

```
SINS_main_directory
└─ SINS
   └─ output
      └─ <name of project>
         ├─ simulation_1
         ├─ simulation_2
         ├─ simulation_3
         ├─ ...
         └─ simulation_<j>
            ├─ demography.txt
            ├─ SummaryStatistics.txt
            ├─ <name of layer>_X.txt
            ├─ <name of layer>_Y.txt
            ├─ <name of layer>_MT.txt
            ├─ <name of layer>_A1.txt
            ├─ <name of layer>_A2.txt
            ├─ ...
            └─ <name of layer>_A<n>.txt
```

Figure 5.1: SINS output folder structure and organization. <j> denotes the total number of simulation replicates. <n> denotes the total number of "autosomes" defined.

analysis of the populations in your simulations. There are many ways to conduct such analysis, we do it by loading the data from the simulations into R, and then conducting the analysis there.

(a) SINS format

generation	X coordinate	Y coordinate	individual ID	mother ID	father ID	genotype
100	1	2	F_41_0_1_2	F_0_0_1_2	M_35_0_2_2	500/500
100	4	3	F_101_0_4_3	F_100_0_4_3	M_71_0_4_3	501/500
100	4	3	F_13_0_4_3	F_4_0_3_3	M_1_0_4_4	500/502
100	4	4	M_113_0_4_4	F_127_0_4_4	M_102_0_4_4	500/499

(b) Adegenet format

individual ID	genotype	X coordinate	Y coordinate	population name	generation
F_41_0_1_2	500/500	1	2	pop_0.1-2	100
F_101_0_4_3	501/500	4	3	pop_0.4-3	100
F_13_0_4_3	500/502	4	3	pop_0.4-3	100
M_113_0_4_4	500/499	4	4	pop_0.4-4	100

Figure 5.2: Example of SINS output formats. Both formats output the same information except for the mother and father columns which are only output by the "sins" format, and the population name column which is only output by the "adegenet" format. *(a)* and *(b)* show the information for the same individuals in the two different formats. The X and Y coordinate columns show the X and Y coordinates of the individuals' deme of birth.

In both formats each individual is identified by a code containing information about its sex (S), identity (I), layer (L) deme and time step. Both the individuals' and parents' labels are built with a S_I_L_R_C structure:

- S: sex of the individual (M for male, F for female);
- I: individual index given by the program to each individual in a given deme;
- L: layer of birth. From 0 to the [total number of layers -1];
- R and C: coordinates of the deme of birth (row and column indexes respectively).

Note that in the Adegenet format, the population name is simply constructed based on the layer of origin and deme coordinates for a given individual. Therefore individuals from the same demes will be grouped on the same population (this is relevant when analyzing data on adegenet). These population names are constructed in a way where you can easily tinker/change them in R.

Something important to note is that **the numbering of both layers and deme coordinates starts at zero**

5.1.3 Summary statistics

For each simulation replicate, for each generation specified by the user, a small number of summary statistics will be produced per genetic marker. These are produced in real time so you can consult the file at any time while the simulation is running (just make sure you don't make any changes to it or it might lead to unexpected behaviour). These statistics are computed taking **ALL the individuals from all the demes** into account[1], it can be seen as a way to monitor the overall genetic status of your simulations over time. Currently the summary statistics being produced are shown in table 5.1. An example of a SummaryStatistics.txt file can be seen in listing 11

Table 5.1: Name of the columns and corresponding description of each of the summary statistics present in the SummaryStatistics.txt file.

SummaryStatistics.txt - (statistics) column description	
Name of column	Description
Heterozygosity	Expected Heterozygosity calculated per marker/locus.
OverallHetAutozomes	Overall Expected Heterozygosity calculated for all markers, excluding the ones in the X/Y chromosomes and mtDNA.
MeanAlleleFrequency	Mean allele frequency calculated per marker/locus.
NumAlleles	Total number of alleles calculated per marker/locus.

[1]To compute these statistics, we currently group all individuals as if they were in a single panmitic population. These statistics might be misleading depending on what you want to study, so please, do not rely solely on these statistics.

5.2 SINS database

Although this is just another format in which you can output data from SINS, it merits its own section because it brings together a lot of practical advantages when running simulations. The only disadvantage being that since it's not a text file, you can't just double click to open and see what's inside (on the output of a big simulation you wouldn't want to do this anyway). The advantages are the following:

- Faster (IO) output to file, making SINS run slightly faster;
- Smaller output file, making your outputs easier to manage as well as avoiding much of the redundancy provided in the text files;
- Easier to import to R/RStudio (1), since you can choose exactly what you want to import, avoiding the problem of loading large text files into R/RStudio;
- Easier to import to R/RStudio (2), since a simple package manages all the communication between R and the database, you only have to spend 5 minutes to learn a very simple syntax to import the chunks of data that you want;
- Less file clutter; since everything is condensed in a single file per simulation replicate, it is less likely for any of the files to be altered or lost by mistake when trying to, for example, analyze an old simulation.

Table **??** describes the different tables and columns of the database format. Listing 12 shows how easy it can be to directly import all the data into R/RStudio (you don't necessarily want to do this, but it is to show how simple this process is). In section 5.3 you can find alternative ways to load your simulation data into R/RStudio.

5.3 SINS data analysis

Due to the simplicity of the format that SINS outputs, the user can choose many different ways to analyze his/her own data. We usually load the genetic data into R and then we have 1 of 2 scenarios:

- we perform an exploratory analysis, tinkering the methods, parameters and the data as we go and getting the results at the same pace
- we already have a pipeline of analysis ready for the data that we are loading, this pipeline will run unsupervised, and we will only consult the results in the end.

Here we will show a small exploratory analysis, where we will load the data into R with a specific script that is included in the SINS package, compute some statistics and finally produce some plots.

5.3.1 Loading data into R

When choosing to not output the data into the database format, the user will have to find other ways to get the data into R/RStudio. We use an inhouse script "readSinsData.R" to load SINS text data into R. This script is included in the SINS package. Note that this script requires a number of different R libraries/packages to be installed. These are:

- adegenet
- doParallel
- foreach
- data.table

You can and should change this script to fit the way you want to load the data. Currently this script is tunned to load all the data that was saved in a simulation (in all its replicates). While in an R environment (like the R console or RStudio), loading your simulations data can be done as show in listing 13.

Listing 10 Example of a single generation in a demography.txt file for a 25 by 25 deme simulation. The information in this file can be easily loaded and parsed in R or python for example.

```
Simulation: 1
Layer name: layer0
Generation: 50

[map]
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 8 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 24 31 41
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 7 36 75 79 98 85
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 14 54 78 77 100 63 77
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 90 74 104 94 92 104 108
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 19 60 94 90 75 62 105 98 107
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 28 62 74 103 119 119 84 102 100 106
0 0 0 0 0 0 0 0 0 0 0 0 0 1 31 89 115 94 86 98 107 110 99 113 94
0 0 0 0 0 0 0 0 0 0 0 0 0 0 35 84 80 82 109 113 129 92 105 64 118
0 0 0 0 0 0 0 0 0 0 0 0 0 15 70 88 86 102 99 95 78 107 103 85 72
0 0 0 0 0 0 0 0 0 0 0 0 0 21 69 77 77 98 102 108 115 83 98 99 99
0 0 0 0 0 0 0 0 0 0 0 0 4 18 80 96 111 108 109 124 103 108 104 94 86
0 0 0 0 0 0 0 0 0 0 0 0 0 27 59 84 86 117 102 89 95 114 111 90 106
0 0 0 0 0 0 0 0 0 0 0 0 0 1 36 85 88 93 107 104 112 98 114 95 90
[/map]


Simulation: 1
Layer name: layer0
Generation: 51

[map]
...
```

Listing 11 Example of a SummaryStatistics.txt file for a single generation, in a simulation with 5 non-sexual markers (autosomes).

```
Layer Generation Marker Heterozygosity OverallHetAutozomes MeanAlleleFrequency NumAlleles
0 100 X 0.48389217619986846 NA 0.5 2.0
0 100 Y 0.46955641760836564 NA 0.5 2.0
0 100 MT 0.618676492073928 NA 0.19999999999999998 5.0
0 100 A1 0.8880134140550807 0.8084463346482578 0.028571428571428567 35.0
0 100 A2 0.4378492767915845 0.8084463346482578 0.5 2.0
0 100 A3 0.7246015276864636 0.8084463346482578 0.14285714285714285 7.0
0 100 A4 0.9960888989334502 0.8084463346482578 0.0019960079840319308 501.0
0 100 A5 0.9956785557747097 0.8084463346482578 0.0020833333333333277 480.0
```

Listing 12 Example of how to import data from a single simulation replicate into R/RStudio from the database format.

```r
library(DBI)
library(RSQLite)

db_file <- "/home/SINS/output/My_project/SINS_Data1.db"
con <- dbConnect(RSQLite::SQLite(), dbname=db_file)
my_query <- "SELECT * from sinsdata"
sim_data <- dbGetQuery(con, my_query)
```

Listing 13 Example usage of the readSinsData script/function. In this example we are loading simulations 1 through 3 and only selecting markers "A1", "A2" and "A3".

```r
source("/home/user/SINS_Package/R_scripts/readSinsData.R")

dataToAnalyze = readSinsData(
             dataDirectory = "/home/user/SINS_Package/SINS/output/",
             simulationName = "mySimulationProject",
             nSim=c(1:3),
             markers = c("A1","A2","A3"),
             layerName = "layer0",
             dataFormat = "adegenet",
             n_cores = 2
)
```

References

[1] Currat, M. and Excoffier, L. (2004). Modern humans did not admix with Neanderthals during their range expansion into Europe. *PLoS Biol*, 2(12):e421.

[2] Currat, M. and Excoffier, L. (2005). The effect of the Neolithic expansion on European molecular diversity. *Proc R Soc B*, 272(1564):679–688.

[3] Currat, M., Ray, N., and Excofier, L. (2004). SPLATCHE: a program to simulate genetic diversity taking into account environmental heterogeneity. *Mol Ecol Notes*, 4:139–142.

[4] Fisher, R. A. (1922). On the dominance ratio. *Proc R Soc Edin*, 42:321–341.

[5] Kimura, M. and Ohta, T. (1978). Stepwise mutation model and distribution of allelic frequencies in a finite population. *Proc Natl Acad Sci U S A*, 75(6):2868–2872.

[6] Kimura, M. and Weiss, G. H. (1964). The stepping stone model of population structure and the decrease of genetic correlation with distance. *Genetics*, 49(4):561–76.

[7] Lotka, A. J. (1932). The growth of mixed populations : two species competing for a common food supply. *Journal of Washington Academy of Sciences*, 22:461–469.

[8] Maynard-Smith, J. and Slatkin, M. (1973). The stability of predator-prey systems. *Ecology*, 54:384–391.

[9] Rasteiro, R., Bouttier, P.-A., Sousa, V. C., and Chikhi, L. (2012). Investigating sex-biased migration during the neolithic transition in europe, using an explicit spatial simulation framework. *Proc Biol Sci*, 279(1737):2409–2416.

[10] Ray, N., Currat, M., Foll, M., and Excoffier, L. (2010). SPLATCHE2: a spatially explicit simulation framework for complex demography, genetic admixture and recombination. *Bioinformatics*, 26(23):2993–2994.

[11] Volterra, V. (1931). *Variations and fluctuations of the numbers of individuals in animal species living together*, pages 409–448. McGraw-Hill, New York.

[12] Wright, S. (1931). Evolution in Mendelian Populations. *Genetics*, 16(2):97–159.