quantiNemo
release 2.0.0

User Manual

March 20, 2018

authors
Samuel Neuenschwander
samuel.neuenschwander@unil.ch

Jérôme Goudet
jerome.goudet@unil.ch

Frédéric Michaud
frederic.michaud@unil.ch

website
[Is temporal] http://www.unil.ch/popgen/softwares/quantinemo

Contents

1 Introduction **1**

1.1 Scope 1

1.2 How to read this Manual 1

1.3 Main features 3

1.4 Input and output 5

1.5 Availability 5

1.6 Technical 5

1.7 License 6

1.8 Citation 6

1.9 Acknowledgments 6

2 Getting started **7**

2.1 Installation 7

2.2 Launching quantiNemo 7

2.3 Minimal settings file 8

2.4 Simulation example 9

3 Input file structure **12**

3.1 Default value 12

3.2 Comments 13

3.3 Line end . . . 13
3.4 Parameter types . . . 13
3.5 Temporal parameters . . . 16
3.6 Keywords . . . 17
3.7 External files . . . 17
3.8 Command line parameters . . . 18
3.9 Macros . . . 18
3.10 Batch mode . . . 25

4 General simulation settings **28**

5 Life Cycle **32**
5.1 Introduction . . . 32
5.2 Mating system . . . 34
5.3 Death . . . 37
5.4 Regulation offspring . . . 37
5.5 Regulation adults . . . 38
5.6 Extinction . . . 38

6 Demography **40**
6.1 Dimensions . . . 40
6.2 Initialization . . . 41
6.3 Population growth . . . 42
6.4 Dispersal . . . 45

7 Genotype configuration **50**
7.1 Introduction . . . 50
7.2 Defining loci . . . 51
7.3 Initial genotypes . . . 53

7.4 Mutation 55
7.5 Multiple traits 59
7.6 Genetic map 60

8 Quantitative traits and selection 65
8.1 Introduction 65
8.2 From genotype to genotypic value 67
8.3 Environmental effect 74
8.4 Selection pressure 78
8.5 Selection level and position 87
8.6 Fitness factor 90

9 Coalescence 92
9.1 Introduction 92
9.2 Output 95
9.3 Summary statistics 100

10 Ouputs and Statistics 102
10.1 Files name 103
10.2 Sampling 104
10.3 Summary statistics 106
10.4 Raw data 119
10.5 print input file 126

Bibliography 128

Appendices 130

A technical details 131
A.1 Allelic value distribution 131

A.2 Multiple traits with varying types of selection on various patch: simple case . . . 135
A.3 Multiple traits with varying types of selection on various patch: matrix expansion . . . 136
A.4 Selection pressure definition . . . 138
A.5 Simulating sexual chromosome . . . 138

B Speeding up simulation **142**
B.1 general consideration . . . 142
B.2 How to improve simulation time . . . 144

C Glossary **146**

Index **149**

Chapter 1

Introduction

1.1 Scope

QuantiNemo is an individual-based, genetically explicit stochastic simulation program. It was developed to investigate the effects of selection, mutation, recombination, and drift on quantitative traits and neutral marker with varying architectures in structured populations connected by migration and located in a heterogeneous habitat. QuantiNemo is highly flexible at various levels: population, selection, trait(s) architecture, genetic map for QTL and/or markers, environment, demography, mating system, etc.

1.2 How to read this Manual

A full understanding of quantiNemo is not necessary to start using it, and most of the reader will find themselves only using a small part of the manual, therefore reading it from the beginning to the end is not recommended.

If you are only interested in simple simulation, you may directly go to getting started 2, which will help you grasp the very basic concept of quantiNemo, and then jump to whatever chapter is of interests for you. Most of the chapters have an introduction which detail the philosophy of the simulation component. If you are planing to run advanced simulation and want to have a deeper understanding of quantiNemo, then reading the chapter about

settings file 3 and understanding the concept of the life cycle's chapter 5 is important. The introduction of chapter 8 and 7 are also important to grasp the philosophy of quantiNemo. All the other part can be skipped or read in details depending on the type of simulation that you want to perform. At the end of the manual, you will find an index with all possible parameters: see C.

The structure of the manual is the following:

1. Chapter Introduction presents the program, giving general information about it (license, citation, etc) and presents keys features. It is attended for people who wants to know more about quantiNemo, but no information is provided about how to run it.

2. Chapter Getting started gives the minimal necessary knowledge to launch a basic quantiNemo simulation and output simple result about the simulation. Newcomers should start with this chapter.

3. Chapter Input file structure gives deeper information about how to write an input file. It is recommended for more advanced users and can be skipped at first reading: simple example given in the chapter Getting started are already sufficient for simple simulation. It's more relevant to advance user.

4. Chapter General Simulation settings gives details information about the parameters used for meta-information about the simulation, *i.e.* relevant but not modifying the scenario itself, like the folder, the number of replicates, the initial seed, etc.

5. Chapter Life cycle explains a bit more about how the different event through which all individual go (birth, migration, etc) and give information about how to change their behaviors.

6. Chapter Demography contains all the information about how to setup a specific demographical scenario, tuning the number of individuals, the population growth model, the dispersal model, etc.

7. Chapter Genotype configuration contains all the information about how to set the genetic information of all individual, like the number of loci, their position on the genome, their mutation rate, etc.

8. Chapter Quantitative traits and selection is about how to define quantitative traits and add selection pressure to them.

9. Chapter Coalescence defines all parameters needed to perform a backward in time simulation. This type of simulation is much more efficient in term of CPU and memory usage (i.e. much faster) but can only simulate neutral marker and no quantitative traits.

10. Chapter Outputs and statistics describes in great detail how to output information about the simulation such as summary statistics or raw data (like population genome).

11. In the annex, a few more technical points are discussed. If you have technical difficulties to implement a scenario, you may want to check if an annex could help.

1.3 Main features

QuantiNemo consists of several simulation components which may be easily extended. The simulation components with their corresponding parameters are described in more detail in the rest of this manual.

Quantitative traits
: QuantiNemo allows the simulation of one to multiple quantitative traits each having its own specifications. Each trait is defined by one to many loci. The allelic effects at each locus can be drawn from a normal distribution or can be set explicitly. Mutations are implemented with several models. The trait determinism can be purely additive or include dominance and/or epistatic interactions among loci. Environmental effects can also be set in different ways. The selection pressure can also be from different type, from stabilizing selection to arbitrary fitness landscape, allowing to simulate a large variety of system including deleterious mutation. Last, selection can be soft or hard (Wallace, 1968).

Neutral markers
: quantiNemo also allows the simulation of neutral markers, such as microsatellites or SNPs with different mutation models (K Allele, Step-

wise). Different types of neutral markers and/or quantitative trait loci can be combined within the same simulation.

Genetic map

QuantiNemo has an underlying genetic map, which may consist of several chromosomes. This allows an explicit positioning of all types of loci on the map (quantitative trait loci (QTL) and neutral markers) to simulate any recombination rate between loci.

Metapopulation

QuantiNemo allows simulating realistic population dynamics. Population sizes may vary in space and in time. The user can choose between several preset migration models (island, 1-D stepping-stone, 2-D stepping-stone), or specify the full migration matrix. The migration pattern can change over time, allowing to investigate scenarios of population fragmentation.

Lifecycle

Each individual undergoes a single life cycle (non-overlapping generations). The life-cycle starts with breeding and reproduction. Several mating systems are available: random mating, selfing or cloning for hermaphrodites; promiscuity, monogamy or polygyny for dioecious (gonochoric) species. Selection acts by default on the reproductive fitness of individuals, however selection at other life cycle stages is also supported. After reproduction, juveniles may disperse to other populations, and then population size is possibly regulated. Environmental stochasticity can also be introduced, where populations may go extinct due to an external factor, independent of population size or genetic constitution of the population.

Coalescence

To simulate larger population size and genome, QuantiNemo offers the possibility to switch from individual to population based simulation, using coalescence. In coalescence, a forward in time simulation in first performed to simulate any complex demographical scenario, followed by a backwards coalescence simulation. This setup allows for much more elaborate scenarios than common coalescence tools.

1.4 Input and output

Input

QuantiNemo is launched using a settings file. The settings file is a text file with flexible and easy to understand structure. The file can be edited with any text editor, and comments may be added for better readability. We provide also a syntax-highlighting definition for better readability.

Output

QuantiNemo provides summary statistics for the different simulation components, including genetic variance estimates, quantitative trait analysis (e.g. Q_{ST}), and F-statistics for all types of loci (neutral and QTL). The summary statistics can be computed for any generation during the simulation. QuantiNemo can also produce files with the raw genetic data. The genotypes at all loci can be dumped to file in the FSTAT (Goudet, 1995) or the Arlequin format (Excoffier, 2010). Phenotypes, as well as the additive, dominance, and epistatic effect values can be written to a file and then analyzed with any population or quantitative genetic software, e.g. to get patterns of differentiation, study linkage disequilibrium, or scan for QTLs.

1.5 Availability

The website http://www.unil.ch/popgen/softwares/quantinemo **[Is temporal]** includes executables for Windows, Linux and Mac, the source code, a detailed user's manual and tutorials. All downloads are freely available under the terms of the GNU General Public License.

1.6 Technical

quantiNemo is a console program and is coded in standard C++ using an object oriented approach. This allows compiling quantiNemo on any computer platform which supports standard C++ compilation. There is no limit

on the number of populations, individuals, genes, etc that quantiNemo can handle, apart from the available hardware capacities (CPU and memory). QuantiNemo was optimized for high computation efficiency in particular for large simulations on clusters.

1.7 License

1.8 Citation

[Is temporal] quantiNemo has been published in the journal *Bioinformatics*. If you are using quantiNemo for your research please cite it as
Neuenschwander S, Hospital F, Guillaume F, Goudet J, 2008. QuantiNemo: an individual-based program to simulate quantitative traits with explicit genetic architecture in a dynamic metapopulation. **Bioinformatics** 24, 1552-1553.

1.9 Acknowledgments

[Is temporal] **[FM: JÃľrÃťme, can you update this?]** We are grateful to Elisa Cavoto, Olivier Blaser, Christine Grossen, Ricardo Kanitz, Sylvain Antoniazza and Elsa Guillot for feedback on the new version of quantiNemo. We are also grateful to Fardo Witzenburg who did most of the tutorials.

Chapter 2

Getting started

This chapter explains how to use quantiNemo starting from the basics.

2.1 Installation

Executables of quantiNemo for several operating systems can be downloaded from the web site http://www.unil.ch/popgen/softwares/quantinemo **[Is temporal]** . The executables are standalones, meaning that quantiNemo does not require an installation. After downloading the compressed file with the executable corresponding to your operating system simply extract it to a folder of your choice. The compressed file includes the executable for your operating system, the user manual, and also an example settings file.**[Is temporal]** .

2.2 Launching quantiNemo

There are different ways to launch a simulation with quantiNemo. Normally a simulation is defined using a settings file and the settings file is then passed to quantiNemo, but it is also possible to specify the simulation parameters directly as arguments passed to the program. See chapter 3 for more details.

no argument

If no argument is passed to quantiNemo during launching (double-clicking on the executable or by launching the executable via a console (e.g. `quantinemo.exe` for Windows)) then quantiNemo will automatically use the settings file named `quantinemo.ini` residing next to the executable if present. If such a settings file (with the given name) is not present, then quantiNemo will prompt you for the name of a settings file until a valid name is entered (relative and absolute paths are accepted) or "exit" is passed as file names (without quotes).

file name

The settings file name may be passed as a parameter to the executable when quantiNemo is launched. This can be done using a console window:

```
> quantinemo.exe settings.ini
```

Depending on the operating system a '`./`' in front of the executable is sometimes needed to specify that the executable is located in the current directory (Linux/Mac OS):

```
> ./quantinemo settings.ini
```

2.3 Minimal settings file

Most of the parameters have default values. This allows making short and clear settings files. This section describes the parameters needed for a minimal settings file and describes the simulated model, respectively how the default values are set.

The following two parameters are needed in every settings file:

```
generations     500
patch_capacity  1000
```

This minimal settings file allows performing a simulation with a single population consisting of 1000 hermaphrodites. The population evolves under neutral random mating for 500 generations keeping the population size constant at carrying capacity. No genetic data are simulated since no quantitative traits or neutral markers are specified. The simulation generates no output

apart from the log file. Although this simulation works it makes no sense, since no output is generated.

2.4 Simulation example

This section describes a more realistic simulation example and describes how the output is stored. The settings file of this example named "quantiNemo2_example.ini" is included in the compressed folders of the downloads of quantiNemo:

```
generations                 1000

# metapopulation
patch_number                10
patch_capacity              1000
dispersal_rate              0.01


# selection
quanti_stab_sel_optima       5
quanti_stab_sel_intensity    0.1

# quantitative trait
quanti_loci                 5
quanti_all                  255
quanti_mutation_model       0
quanti_mutation_rate        1e-4


# statistics
stat                        {q.adlt.ho}
```

A simulation based on this settings file named "quantiNemo2_example.ini" produces the following output to your terminal window: **[Is temporal]**

```
Program:    quantiNemo2 (quantitative genetics simulator)

Version:    2.0.0 [Dec 25 2017; 23:12:33]

Authors: Samuel Neuenschwander (samuel.neuenschwander@unil.ch) &
         Frederic Michaud (frederic.michaud@unil.ch)&
         Jerome Goudet (jerome.goudet@unil.ch)
```

```
          Department of Ecology and Evolution
          University of Lausanne, Switzerland

Reading settings file 'example.ini' ...
Reading settings file 'example.ini' done (16 parameters)

SETTINGS (example.ini)
  Simulation:
    individual-based
    120 generations
    2 replicates
    1. of 1 batch simulations
    1 threads used
    random generator initialized by the time

  Loaded traits:
    Quantitative trait: 5 loci; 255 alleles; neutral trait;

  Life cycle sequence:
    1. breed
    2. save_stats
    3. save_files
    4. aging
    5. dispersal

  Metapopulation:
    10 patches (K_tot=10000)
    Migration model: island
    Mating system: random mating (hermaphrodite)

  Genetic map:
    5 independent loci

  Output:
    (to folder 'example')
    genotype (quanti) computed at 4 generations
    statistics (2 statistics) computed at 120 generations


    replicate 2/2 [00:00:01] 120/120   RAM: 0.000000 MB

------ SIMULATION done (CPU time: 00:00:02s) ------

quantiNemo2 terminated successfully!
```

This output informs you that this simulation was parametrized by the settings file "quantiNemo2_example.ini". The simulation consisted of one quantitative trait and one type of neutral markers. The simulated life cycle is indicated and shows that summary statistics and genotypes or phenotypes are output. Only one replicate of 100 generations was performed. For each replicates the elapsed time (hh:mm:ss) is printed out to the console and at the end of the simulations also the total elapsed time. The output of this simulation was stored in the following structure relative to the executable:

```
simulation.log      #log file to relaunch the same simulation
simulation_stats.txt          # statistics for each replicate
simulation_mean.txt           # statistics across replicates
simulation_var.txt            # variance across replicates
simulation_legend.txt         # statistic legends
simulation_g030_r1.dat        # genotype raw data
simulation_g030_r2.dat        # genotype raw data
simulation_g060_r1.dat          # genotype raw data
simulation_g060_r2.dat        # genotype raw data
simulation_g090_r1.dat          # genotype raw data
simulation_g090_r2.dat        # genotype raw data
simulation_g120_r1.dat        # genotype raw data
simulation_g120_r2.dat        # genotype raw data
```

Chapter 3

Input file structure

This section describes the format of the settings file. The settings file is a text file with in general one parameter per line in a key-value scheme. For example

```
patch_capacity 1000
```

sets the parameter patch_capacity to the value of 1000. The order of appearance of the parameters in the settings file does not matter. However, a particular parameter should appear only once in the settings file. If a parameter appears several times in the file only the last instance is considered.

3.1 Default value

Most of the parameters have default values. The default value of a parameter is taken into account when either the parameter is not listed in the settings file or its argument is missing. The default values are listed behind the parameter name in this manual and are specified by (default:). The default values are the most common setting of the parameter. the default values allow keeping the settings file short and clear.

3.2 Comments

quantiNemo allows adding comments in the settings file (and all other input files). There are two different types of comments:

Single line comments
A simple hash character '#' defines the start of a single line comment. The hash character and the remaining text of the line are ignored by quantiNemo.

Block comments
Block comments may start and end at any place in the file. This allows to comment out multiple lines at once or only a part of a line. A block comment starts with the characters '#/' and ends with the characters '/#'. The starting and ending characters and the text between them are ignored.

3.3 Line end

In general, a parameter (the key and its argument) has to be written on a single line (except for matrices and temporal parameters). However, using a backslash '\' it is possible to bypass the end of a line and to write an argument on several lines. Note, that after the backslash any text on the line is removed.

3.4 Parameter types

There are different types of arguments that a parameter may take. In the following the argument type is specified within square brackets. Example:

```
stat_log_time [integer]
```

Integer
Integers are whole-numbers, *i.e.* a dot-less number. The following forms are equivalent: 1000 or 1e3.

Decimal

Decimals are floating-point numbers. The following forms are equivalent: 0.0001, .0001 or 1e-4.

String

Strings are text arguments. If the string contains spaces the argument has to be enclosed within quotation marks "...". When a string is enclosed by quotation marks it may be written over several lines. Example of a string with a space:

```
folder "first simulation"
```

Matrix

Matrices allow passing several numbers (integer or decimal) to a parameter. This may be necessary for example to specify carrying capacities (see section 1D Matrix), or to pass a dispersal matrix (see section 2D Matrix) to quantiNemo. Matrices are enclosed between curly brackets '{ }', and numbers are separated by at least a space. Matrices may be written on several lines and may also contain comments. There is no *a priori* restriction on the size of the matrix.

1D Matrix

A one dimensional matrix (vector) consists of data in a single dimension. There are three different ways to write a one dimensional matrix, which are all equivalent:

```
patch_number 4
patch_capacity {  20   10   20   10  }
patch_capacity { {20   10   20   10} }
patch_capacity { {20} {10} {20} {10} }
```

2D Matrix

Some parameters need a second dimension for their argument. A second dimension is obtained by enclosing the inner rows of the matrix again within curly brackets '{ }'.

```
patch_number 4
disp_rate { {0.2  0.0  0.0  0.8}
            {0.4  0.2  0.0  0.4}
            {0.4  0.4  0.2  0.0}
            {0.0  0.4  0.4  0.2} }
```

This example shows the pairwise dispersal matrix for 4 patches (4x4). Each row specifies a source patch from which emigrants emigrate. Each column specifies the target patch receiving the immigrants. The diagonal of the matrix specifies the proportion of individuals remaining in the natal patch.

Matrix length adjustment

Usually, matrices are defined in whole, *i.e.* the number of carrying capacities in the 1D matrix example above meets the number of patches (4 patches). However, if there is a repeating pattern in the matrix, it is also possible to define only the repetition. In this case, the matrix will be repeated as needed. For instance, the 1D matrix above could also be written in one of the following ways as there is a repetition in it:

```
patch_number 4
patch_capacity {20  10}
```

Note, that rows and/or columns are repeated as needed. If the number of columns (or rows) is not an entire subset a warning will be returned and the simulation will adjust the matrix as:

```
patch_number 5
patch_capacity {20  10}
```

In this example we have 5 patches, however, the carrying capacities are only specified for 2 patches. As 2 is not an entire subset of 5 a warning will be returned and the patches will have the following carrying capacities: 20, 10, 20, 10, 20. If all values of a matrix are identical one may leave out the brackets. In the following example all three declarations result in the same simulation:

```
patch_number 4
patch_capacity {20  20  20  20}
patch_capacity {20}
patch_capacity  20
```

Unbalanced matrices

Usually, a matrix is balanced, *i.e.* each row has the same number of columns. However, some parameters (such as the parameter `quanti_loci_positions`) allow having unbalanced matrices, *i.e.* rows do not necessarily contain the same number of columns.

Other parameters (such as the parameter `quanti_loci_positions`) allow to skip a row, *i.e.* some rows do not contain any data. A row can be skipped by explicitly indicating the rank of the row just after the beginning of the row followed by a colon ":". The ranking starts with 1. If a row has no explicit rank it is assumed to follow the preceding row. Here is an example for the genetic map of a quantitative trait:

```
quanti_loci 11
quanti_loci_positions { {1: 10}
                        {3: 20  40  60  80  100}
                        {   20  40  60  80  100} }
```

In this example, the quantitative trait is defined by 11 loci located on three out of four chromosomes. The first chromosome contains a single locus at position 10 cM. No locus is located on the second chromosome. The third and fourth chromosome have the same structure: 5 loci are located on each of the two chromosomes, and the distance between adjacent loci is 20 cM.

3.5 Temporal parameters

An important feature of quantiNemo is that some parameters may change over time during a simulation. Such parameters are indicated as "temporal" in this manual. Temporal arguments are enclosed within two parentheses '(...)'. For each change of the argument over time a pair consisting of a generation index and a corresponding argument is needed. The values of a pair are separated by at least one space. Pairs are separated by a comma ',' or by a semi-colon ';'. The first value of a pair specifies the time, *i.e.* before which generation the change happens. The second value is the new argument. A parameter may change as often as any generation. A simulation starts at generation 1. Therefore the first change has to have the time value 1 otherwise the parameter cannot be initialized leading quantiNemo to return an error. A temporal argument may be written over several lines.

```
patch_capacity (1    100,
                50   200,
                100  500)
```

Here, the carrying capacity is 100 for the first 49 generations, 200 from generation 50 on, and 500 from generation 100 on.

3.6 Keywords

quantiNemo supports keywords in the settings file. A keyword may be defined using the word 'set', followed by the keyword and the argument. The keyword can be used in parameter arguments and quantiNemo will replace all keywords by their defined arguments. A keyword supports any argument which makes sense for the parameter where the keyword is used, including macros and temporal parameters.

```
set NUM_GEN runif(1, 100, 1000)
generations NUM_GEN
stat_log_time NUM_GEN
```

In this example the keyword 'NUM_GEN' is defined as a random macro. Consequently both parameters **generations** and **stat_log_time** are set to the same value. In other words, this settigns file allows to easily define randomly the number of generations to simulate between 100 and 1000 generations and to compute the statistics at the last generation.

3.7 External files

In general, arguments are written directly after the parameter name on a single line. However, arguments may be sometimes large (e.g. big matrices, temporal parameters, ...) leading to poorly readable settings file. Using external files for large arguments allows keeping the settings file clear and well readable. An external file may be used for any parameter. Instead of writing the argument directly after the parameter name, the name of an external file is written after the parameter name. In order for quantiNemo to recognize the argument as a file name the prefix '$' has to be added before the file name. The external file must contain the argument of the parameter. Only a single argument per external file is possible, however, the same external file may be used for several parameters. The format of the argument in the external file follows the same rules as in the settings file,

except that line breaks are ignored, i.e the character '\' between lines is not necessary. Here is an example:

settings file:

```
disp_rate $dispersal_file.txt
```

external file named "dispersal_file.txt":

```
# dispersal rates
{ {0.2  0.0  0.0  0.4  0.4}
  {0.4  0.2  0.0  0.0  0.4}
  {0.4  0.4  0.2  0.0  0.0}
  {0.0  0.4  0.4  0.2  0.0}
  {0.0  0.0  0.4  0.4  0.2} }
```

3.8 Command line parameters

Parameters are normally defined in a settings file which is then passed to the command line. All parameters and their arguments may also be passed directly to the command line. Command line options provide a convenient mechanism to override commonly altered parameter values. Parameters and their argument passed to the command line override the arguments defined in the settings file. To do so, the argument must be specified after the name of the input file. The parameter itself should have two – in front, and the value it takes should be right after it, separated by a space:

On Windows:

```
quantiNemo.exe settings.ini --generations 1000
```

On Linux or Mac OS:

```
./quantiNemo settings.ini --generations 1000
```

3.9 Macros

QuantiNemo supports several macros allowing to write more easily the settings file. Several macros allow drawing random deviates from a given dis-

tribution. Note, that the random deviates are only drawn once, *i.e.* all replicates have thus the same numbers. The names and parameters of all macros follow as good as possible the equivalent functions in the statistical package R. To understand or control the outcome of a macro the log file (see parameter **logfile_type**) may be consulted. This log file may either contain the macros as they were specified or the interpreted macros.

option "Time"

Adding this option to most of the macro allows generating temporal arguments. If this option is used with a macro allowing to draw random deviates it is possible to simulate temporal fluctuations. In principle, the macro draws for the given time interval random numbers following the specifications and generates a corresponding temporal argument with *nb* intervals. The option *Time* has to follow directly the macro name, e.g. *rnormTime* for the macro *rnorm* and the arguments of the macro have to be extended at the end by two values which are the start and the end time, e.g. *rnorm(nb, mean, sd, timeFrom, timeTo)*. In this example the number of random deviates (*nb*) specifies the number of regular temporal changes between *timeFrom* and *timeTo*. The time intervals are generated internally as $seq(timeFrom, timeTo, nb)$. An example of such a temporal macro is shown with the macro *seq* below.

seq($from, to, steps$**)**
seqTime($from, to, steps, timeFrom, timeTo$**)**

With this macro it is possible to specify a sequence of data points. *seq* works similar as *seq* in the statistical package R. The macro needs three arguments separated by a comma: $from$ is the first data point of the sequence, *to* is the last data point of the sequence and *steps* specifies the number of data points including the edges (equivalent to *length.out* in R). $from$ and *to* may be single values or matrices while *steps* has to be a number. If the matrices of $from$ and *to* do not have identical dimensions the matrices are adjusted to each other. Example of *seq*:

```
patch_capacity {seq(100, 1000, 10)}
patch_capacity {100 200 300 400 500 600 700 800 900 1000}
```

Both specifications of the carrying capacities are identical, *i.e.* the macro *seq* translates its arguments to the lower specification.
The command *seq* can also be used for temporal sequences:

```
patch_capacity (seqTime(200, 1000, 5, 1, 5))
patch_capacity (1 200, 2 400, 3 600, 4 800, 5 1000)
```

```
patch_stab_sel_optima (seqTime({{1 10}},{{10}{1}}, 10, 1 10))
patch_stab_sel_optima (1 {{1 9}{1 9}},
                       2 {{2 9}{1 8}},
                       3 {{3 9}{1 7}},
                       4 {{4 9}{1 6}},
                       5 {{5 9}{1 5}},
                       6 {{6 9}{1 4}},
                       7 {{7 9}{1 3}},
                       8 {{8 9}{1 2}},
                       9 {{9 9}{1 1}})
```

Both specifications of the linear increase of the carrying capacities and selection optima, respectively, over time are identical. Again the macro *seqTime* translates its arguments into the lower specification.

seq2D($xlim, ylim, patchID1, patchID2, value1, value2, steps$))
This macro allows generating a two-dimensional cline of values. For example, this allows generating a cline of carrying capacities across a two-dimensional landscape. Since the macro does not know the other parameter arguments all necessary information has to be passed to the macro, *i.e.* some information may be redundant in the settings file. In principle, the cline is defined by the values at two patches and the number of steps between these two patches to interpolate. The changes of numbers are vertical to the line between the two patches. $xlim$ and $ylim$ are the matrix dimensions, $patchID1$ and $patchID2$ are the patch-IDs for which the values will be defined and the corresponding values are $value1$ and $value2$, respectively. *step* is the number of changes between these two patches, following the macro *seq*.

seq2Db($xlim, ylim, patchID1, patchID2, value1, value2, steps$))
This macro is similar to the previous one $seq2D$ but the interpolation for the cline follows another rule: The value of any patch is defined by the ratio of the distances from this patch to the two specified patches.

rep($text, number$)
With this macro it is possible to specify a repetition of the text. *rep* works similar to *rep* in the statistical package R. The macro needs

two arguments separated by a comma: the first one is the text to be repeated, and the second argument specifies the number of repetitions. Example:

```
patch_ini_size {1000 rep(0, 9)}
patch_ini_size {1000 0 0 0 0 0 0 0 0 0}
```

Both specifications of the initial population sizes are identical, *i.e.* the macro *rep* translates its arguments to the lower specification. In this example only the first patch is populated at the start of the simulation, allowing to simulate a colonization scenario.

runif(nb)
runif(nb, min, max)
runifTime($nb, timeFrom, timeTo$)
runifTime($nb, min, max, timeFrom, timeTo$)

Random numbers may be drawn from a uniform distribution. nb specifies the number of random deviates to draw. min and max specify the range of the distribution. If min and max are not specified the random deviates are drawn within 0 and 1. The returned values are always decimal numbers. To obtain entire random deviates of an uniform distribution the macro has to be set within the macro *ceil*.

rnorm($nb, mean, sd$)
rnormTime($nb, mean, sd, timrFrom, timeTo$)

Random numbers may be drawn from a normal distribution, where *mean* is the mean of the normal distribution, *sd* the standard deviation of the normal distribution, and nb specifies the number of random deviates to draw. *mean* and *sd* may be single values or matrices, while nb has to be a number.

rnorm($nb, mean, sd, min, max$)
rnormTime($nb, mean, sd, min, max, timrFrom, timeTo$)

Same as above but random numbers are drawn from a truncated normal distribution, with min as lower bound and ma as upper bound. min and max have to be single numbers.

rlnorm($nb, meanlog, sdlog$)
rlnormTime($nb, meanlog, sdlog, timeFrom, timeTo$)

Random numbers may be drawn from a log normal distribution, where

meanlog is the mean of the log normal distribution on the log scale ($log(mean)$), *sdlog* the standard deviation of the log normal distribution on the log scale ($log(sd)$), and *nb* specifies the number of random deviates to draw. *meanlog* and *sdlog* may be single values or matrices, while *nb* has to be a number. Note that the parameters to pass to the function are not the *mean* and *standarddeviation* of the underlying distribution. The statistics of the underlying distribution *mean* and *sd* may be obtained as follows:

$$mean = e^{meanlog + \frac{sdlog^2}{2}}$$

$$sd = \sqrt{(e^{sdlog^2} - 1)e^{2meanlog + sdlog^2}}$$

If these equations are transformed it allows to compute the input parameters *meanlog* and *sdlog* for the macro if the statistics of the underlying distribution are known (i.e. *mean* and *sd*):

$$meanlog = log\left(mean\right) - \frac{1}{2}log\left(\left(\frac{sd}{mean}\right)^2 + 1\right)$$

$$sdlog = \sqrt{log\left(\left(\frac{sd}{mean}\right)^2 + 1\right)}$$

rlnorm($nb, meanlog, sdlog, min, max$)
rlnormTime($nb, meanlog, sdlog, min, max, timeFrom, timeTo$)

Same as above but random numbers are drawn from a truncated log normal distribution, with *min* as lower bound and *ma* as upper bound. *min* and *max* have to be single numbers.

rgamma($nb, shape, scale$)
rgammaTime($nb, shape, scale, timeFrom, timeTo$)

Random numbers may be drawn from a gamma distribution, where *shape* is the shape of the gamma distribution, *scale* the scaling factor, and *nb* specifies the number of random deviates to draw. *shape* and *scale* may be single values or matrices, while *nb* has to be a number.

rbeta($nb, alpha, beta$)
rbeta($nb, alpha, beta, from, to$)
rbetaTime($nb, alpha, beta, timeFrom, timeTo$)

rbetaTime(*nb*, *alpha*, *beta*, *from*, *to*, *timeFrom*, *timeTo*)

Random numbers may be drawn from a beta distribution, where *alpha* and *beta* define the shape of the beta distribution, *min* and *max* specify the range of the beta distribution, and *nb* specifies the number of random deviates to draw. If *min* and *max* are not specified the random deviates are drawn in the standard beta distribution ranging from 0 to 1. *alpha*, *beta*, *min*, and *to* may be single values or matrices, while *nb* has to be a number.

rpois(*nb*, *mean*)
rpoisTime(*nb*, *mean*, *timeFrom*, *timeTo*)

Random numbers may be drawn from a discrete poisson distribution, where *mean* is the mean of the poisson distribution, and *nb* specifies the number of random deviates to draw. Note, since the poisson distribution is discrete the macros *rpois*() and *rpoisI*() are identical. *mean* may be a single value or a matrix, while *nb* has to be a number.

rbinom(*nb*, *size*, *prob*)
rbinomTime(*nb*, *size*, *prob*, *timeFrom*, *timeTo*)

Random numbers may be drawn from a discrete binomial distribution, where *size* is the number of trials, *prob* is the probability of success on each trial, and *nb* specifies the number of random deviates to draw. *size* and *prob* may be single values or matrices, while *nb* has to be a number.

rsample(*nb*, *with_replacement*?, *elem*1, *elem*2, *elem*3, ...)
rsampleTime(*nb*, *with_replacement*?, *elem*1, *elem*2, *elem*3, ..., *timeFrom*, *timeTo*)

This macro allows sampling within the given elements (numbers or text supported). *nb* specifies the number of elements to output. The second parameter specifies if the numbers are randomly drawn with replacement (1) or not (0). Any following element separated by a comma is considered as an element allowed to be sampled. Note, that without replacement the number of possible elements to draw may not exceed the number of input elements.

round(*elements*)

This macro rounds all numbers within the brackets to entire numbers.

ceil(*elements*)
: This macro ceils all numbers within the brackets to entire numbers, *i.e.* to the next higher entire number.

floor(*elements*)
: This macro floors all numbers within the brackets to entire numbers, *i.e.* to the next lower entire number.

trunc(*elements*)
: This macro truncates the numbers at zero, *i.e.* all negative numbers are set to zero.

equation(...)
: This macro allows solving simple equations from left to right. Supported are arithmetics on single values, matrices, or a combination of them. It supports the arithmetics plus, minus, product and division. Values and arithmetic signs have to be separated by any space.

In principle macros may be combined within each other or may be concatenated:

```
patch_capacity {rnorm(1000, 1000, 6)}
# {1032.52 968.315 1006.58 1009.23 974.836 976.087}

patch_capacity {round(rnorm(1000, 1000, 6))}
# {972 1001 998 998 973 1003}

patch_capacity {rep(round(rnorm(1000, 1000, 3), 2))}
# {942 1055 1035  942 1055 1035}
```

In all three examples the mean carrying capacity is 1000, however, the capacity of the individual patches varies following a normal distribution with variance 1000. By default, the random generator draws double deviates as shown in the first example. Note, that this argument will be accepted by quantiNemo, however since this parameter expects entire numbers quantiNemo will round the numbers down to the next smaller entire number. By adding the *I* for *integer* to the distribution name the double values are rounded to the next entire number as shown by the second example. The third example illustrates how the macros may be encapsulated with each other resulting in two patches having always the same capacity.

3.10 Batch mode

QuantiNemo allows performing multiple simulations by executing a single command. There are two ways to perform such batch simulations. Note, that these two types may not be mixed, *i.e.* be used at the same time.

3.10.1 Multiple settings files

A normal simulation can be launched by passing the settings file name as parameter to the executable. It is also possible to pass several settings file names to the executable. In this case a simulation for each settings file is executed consecutively:

```
> quantinemo.exe sim1.ini sim2.ini
```

In this example quantiNemo is launched with two settings files (`sim1.ini` and `sim2.ini`). The simulations will be executed one after the other leading to the following console output:

```
...
Reading settings file 'sim1.ini' ...
Reading settings file 'sim1.ini' done (29 parameters)

Reading settings file 'sim2.ini' ...
Reading settings file 'sim2.ini' done (29 parameters)

------ SIMULATION 1/2 ------

...

------ SIMULATION 1/2 done (CPU time: 00:00:15s) ------


------ SIMULATION 2/2 ------

...

------ SIMULATION 2/2 done (CPU time: 00:00:16s) ------
```

3.10.2 Sequential parameters

A batch simulation may also be launched by a single settings file if so-called sequential parameters are used. Sequential parameters are any parameter with not only one but several arguments.

```
patch_capacity 5 10 20
```

In this example patch_capacity is a sequential parameter with three arguments. If patch_capacity is the only sequential parameter three consecutive simulations will be launched with identical parameter arguments, except for the parameter patch_capacity which will be set to 5 for the first simulation, to 10 for the second simulation, and to 20 for the third simulation.

If several parameters are sequential parameters all combinations of the sequential parameters will be simulated. Example:

```
patch_number    10 50
patch_capacity 5 10 20
```

There are two sequential parameters in this example. This will result in 6 consecutive simulations with the following parameters:

```
                patch_number    patch_capacity
1. simulation   10              5
2. simulation   10              10
3. simulation   10              20
4. simulation   50              5
5. simulation   50              10
6. simulation   50              20
```

To prevent the output from overwriting the preceding simulation a unique base file name is given to each simulation. This unique base file name consists of the base file name (parameter filename) plus a suffix which includes the rank of the simulation. If in the example above the parameter filename was set to "mysim" the base name for each simulation would be as follows:

```
                filename
1. simulation   mysim-1
2. simulation   mysim-2
3. simulation   mysim-3
```

```
4. simulation    mysim-4
5. simulation    mysim-5
6. simulation    mysim-6
```

Notice that if you specify a folder using the parameter folder, all simulation will be saved in the same folder and each result will override the previous one.

Chapter 4

General simulation settings

This section describes general parameters of a simulation:

replicates [integer] (default: 1)
Number of replicates to perform per simulation. Replicates are identical simulations (in terms of parametrization), but their outcome differ due to stochastic events.

working_directory [string/integer] (default: 0)
This parameter allows defining the working directory, the directory where the simulations will be stored in. By default, the working directory is the current working directory specified by the OS. The following options are available:

0 : current working directory. The output is stored in the current working directory specified by the OS (default). Normally this is the directory from where the simulation is run.

1 : settings file directory. The output is stored next to the settings file (e.g. *quantinemo.ini*).

2 : executable directory. The output is stored next to the executable.

string : explicitly defined directory. The working directory can also be explicitly defined. If the specified path is relative it is relative to the current working directory given by the OS, but the path can also be absolutely specified.

folder [string] (default: "simulation_yyyy-mm-dd_hh-mm-ss")
This parameter specifies the folder for the output. The default argument of this parameter differs from the rules, as the default folder name is dynamic, *i.e.* comprises the start time and date of the simulation. The default folder name is "inputfilename_" with the date and time as a suffix in the format "yyyy-mm-dd_hh-mm-ss" (Year-Month-Day_Hour-Minute-Second). This dynamic default folder name allows storing each simulation separately, avoiding that previous outputs are overwritten. If the parameter is set, the passed argument will be used as folder name (i.e. without any addition of the time). In this case, it may happen that the new output wants to overwrite previous outputs. The parameter overwrite (see below) allows specifying the rules for overwriting other outputs. If the output should be stored directly in the working directory this parameter has to be listed in the settings file followed by an empty argument "" (for this special parameter an empty argument is not identical to a missing one).

[FM: this parameter is actually not working.File are always overwritten without asking, i.e. value 1. What should we do?]

overwrite [integer] (default: 0)
This parameter specifies how present output is treated if there is a danger of overwriting:

0 : no. If the output of the running simulation will overwrite present files, quantiNemo will ask the user how to proceed (*Do you want to overwrite all the files that use it ? (y(es)/n(o)/s(kip)):*), and will suspend the simulation until the user responds to the question.

1 : yes. Present output is overwritten without asking.

filename [string] (default: "simulation")
This name will be used as the base filename for all outputs if they are not specified individually. The output file extensions are added to this base filename. If a file is written on a replicate-periodic basis, the replicate number will be added between the base name and the extension, so that the same file is not overwritten periodically.

logfile [string] (default: "quantinemo.log")
This is the file name (including the extension) for the log file in which

the simulation logs are recorded. This log file is stored next to the executable and records the main information of each simulation, such as the elapsed time for the simulation and the replicates and the time of the start and end of a simulation. By default, quantiNemo will save all this information in a file named "quantinemo.log".

logfile_type [0-2] (default: 0)

For each simulation, a log file is created containing the settings of the simulation. The file is created for each simulation and is stored in the simulation folder (parameter folder). The name of the log file is composed of the base name (parameter filename) and the suffix ".log".

- **0 : as input.** The file contains the same parameters as the settings file, plus the parameter seed.
- **1 : minimal.** The file contains a minimal set of parameters still able to perform the same simulation. All parameters of the settings file which were set to the default value are not reported.
- **2 : maximal.** The file contains all parameters, including the ones not set in the settings file. For each parameter the type, the default value, if it is a temporal parameter, and a possible range limit is added as comment.

seed [integer/matrix] (default: "")

This parameter specifies the seed which will be used to initialize the random number generator. It is possible to pass a single number as seed in the range of 0 to 4,294,967,295 depending on the computer system (corresponding to an unsigned long in C++). To increase the number of possible seeds it is possible to pass an array of seeds (up to 624) to quantiNemo if specified as a one-dimensional matrix. If the seed is not set the random generator is initialized by the time, *i.e.* a matrix with two seeds is used. Thereby the first number is the time in seconds since 1.1.1970 (function *time(NULL)* in C++), and the second number is the number of clock ticks elapsed since quantiNemo started (function *clock()* in C++). Note, no macros are allowed in the seed argument.

random_per_replicate [0-1] (default: 0)

This parameter allows specifying how to treat macros with random numbers in the case of multiple replicates:

0 : same. All replicates share the same result for the macros

1 : individual. All replicates have different result for the macros

all_combinations [0-1] (default: 1)
This parameter allows specifying how multiple sequential parameters are treated:

0 : order. Simulations with the different orders will be simulated (Sim 1: all first arguments; sim 2: all second arguments, ...). If the number of sequential arguments differs, then the y are extended as a vector in R.

1 : full. All possible combinations will be simulated.

preexec_script [string] (default: "")
This parameter allows specifying a script which will be executed before the simulation. The settings file name is passed as unique parameter to the script.

postexec_script [string] (default: "")
This parameter allows specifying a script which will be executed after the simulation. The settings file name is passed as unique parameter to the script.

Chapter 5

Life Cycle

5.1 Introduction

QuantiNemo is a discrete generation-based simulator. This means that individuals undergo only once the life cycle and that the generations are not overlapping. Depending on the parameterization in the settings file some events may be skipped. The life cycle has a fixed order of events. A simulation starts with "Breeding", *i.e.* only adults are present at the initialization of the simulation. The life cycle is repeated for each generation. The life cycle in quantiNemo is the following.

1 : Reproduction: Adults mate and may produce offspring. By default, selection acts at this stage.

2 : Death: Adults are removed from the population. Only the juveniles remain in the model.

3 : Regulation offspring: Reduce if necessary the population size to carrying capacity on each patch before dispersal.

4 : Dispersal: Individual can migrate to other patches and become adults.

5 : Regulation adult: Reduce if necessary the population size to carrying capacity on each patch after dispersal.

6 : Extinction: Due to stochastic events populations may go extinct.

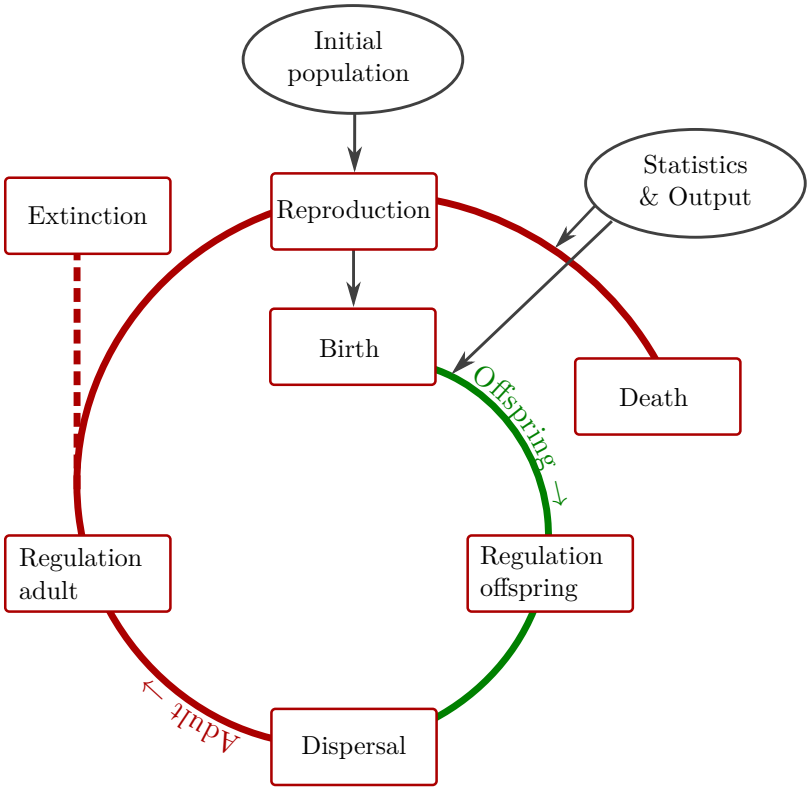


Figure 5.1: Schematic representation of the life cycle.

Moreover, during each cycle, two events are called between breeding and death, allowing to save informations about the simulation. Notice that at this moment, both adults and Juveniles are present, the main difference between both being that the adult might come from another patch while Juvenile not.

Statistics: It is the stage were summary statistics can be computed for adults and juveniles.

Output: Genotypes and/or phenotypes can be dumped to files for adults and/or juveniles.

Most of the life cycle events are described in the following pages. However, the dispersal is included in the chapter about demography, section 6.4. All

the information about statistics and output is also given in a separate chapter 10.

5.2 Mating system

This stage performs mating and reproduction following the selected mating system. The implemented reproduction model in quantiNemo consists of two steps: First, for each patch, the number of offspring to be produced is defined. This number of offspring depends on the parameter mating_nb_offspring_model and if selection acts at this stage also on the level of selection (see parameter selection_level). In a second step, parents are randomly assigned for each offspring (parents are coming form the same patch). This assignment of the parents to the offspring depends on the mating system (parameter mating_system) and on the fitness of the parents if selection acts. Thereby, adults with a higher fitness have on average a higher reproductive success. By default, selection acts at the reproductive success. Adults are not removed at this stage (adults are removed in the event death). The following parameters allow to parametrize this life cycle event:

mating_system [0-6] (default: 0)

Five general mating systems are implemented in quantiNemo. The assignment of the parents to the offspring is random depending on the fitness of the local parents (if no selection acts all individuals have a fitness of 1). Adults with a higher fitness have on average a higher reproductive success):

0 : random mating (hermaphrodite). For each offspring, two hermaphrodite parents are randomly assigned. With probability $1/N$ these two hermaphrodites are identical which leads to selfing. Females are used to simulate hermaphrodites.

1 : selfing (hermaphrodite). For each offspring, a hermaphrodite is randomly assigned to self-fertilize. The parameter mating_proportion allows setting the proportion of outcrosses. quantiNemo controls that the proportion of outcrosses is met, *i.e.* that outcrossing does not result by chance ($1/N$) in selfing. Females are used to simulate hermaphrodites.

2 : cloning (hermaphrodite). This mating system is identical to the model selfing but without any recombination. The genotype of the offspring is identical to the genotype of the mother (only females are simulated) except for changes due to mutations. The parameter mating_proportion allows setting the proportion of sexual reproduction (random mating). Females are used to simulate hermaphrodites.

3 : random mating (promiscuity). This is random mating with two sexes. For each offspring a father and a mother are randomly assigned.

4 : polygyny. Depending on the parameter mating_males only one (default) or several males per patch may reproduce. This fixed number of reproductive males are selected randomly depending on their fitnesses, *i.e.* the reproductive males have on average a higher fitness. Then for each offspring, a mother and one of these reproductive males are randomly assigned depending on their fitnesses. Thus reproductive males with higher fitnesses have a higher reproductive success among the reproductive males. If no selection acts the reproductive males are randomly chosen (all males have the same probability) and each male has the same probability to father an offspring. The parameter mating_proportion allows to set the proportion of random matings between any male and female, *i.e.* also males of the non-reproductive group may get the chance to reproduce.

5 : monogamy. For each female a male is randomly assigned to be its partner for all offspring. If there are fewer females than males present in the patch, not all males will mate. In contrast, if there are more females than males present in the patch, males may belong to several mating pairs. For each offspring, a parent pair is randomly assigned depending on the fitness of the female (if no selection acts the parent pairs have the same probability to be selected). Thus parent pairs, where the females have a higher fitness have on average a higher reproductive success. The parameter mating_proportion allows setting the proportion of random matings.

6 : no mating/reproduction. In this case no mating or reproduction occurs, *i.e.* this life cycle event is skipped. This option allows

using quantiNemo as a statistical package. The input data may be passed as initial genotype files (ntrl or quanti) in the FSTAT format (see sections 7.3). The number of generations has to be set to one since a simulation without any mating and reproduction does not make sense.

Models and their specific parameters

model	additional parameters
0 : random mating (herma.)	
1 : selfing(herma.)	mating_proportion
2 : cloning (herma.)	mating_proportion
3 : random mating (prom.)	sex_ratio
4 : polygyny	sex_ratio / mating_proportion / mating_males
5 : monogamy	sex_ratio / mating_proportion
6 : no mating/reproduction	

mating_proportion [decimal] (temporal/default: 1)

This parameter allows specifying the ratio of a special mating system (selfing, cloning, polygyny, or monogamy) in relation to random mating. A value of 1 (default) means that only the special mating occurs and a value of 0 means that only random mating occurs. For example, if we want to simulate a plant with a selfing rate of 90% we have to set the parameter **mating_system** to 1 (selfing) and this parameter **mating_proportion** to 0.9. These settings will lead to 90% selfing and 10% random mating. Note, that quantiNemo controls that the ratio is met, *i.e.* that selfing does not occur by chance (probability would be $1/N$) when random mating should occur.

mating_males [integer] (default: 1)

This parameter sets the number of males that will be available for mating within each patch. The parameter will only be used if the mating system is polygyny (parameter mating_system set to 4). The range of values is between 1 (a single male mates) and the carrying capacity of the males (all males may mate).

sex_ratio [decimal] (default: 1)

This parameter allows to specify the ratio of males to females of the

offspring in a patch. If hermaphrodites are simulated (parameter mating_system is set to 0, 1 or 2) the sex ratio is not considered, respectively set to 0 (females are used to simulate hermaphrodites).

sex_ratio_threshold [decimal] (default: "")
By default, the sex of an individual is randomly assigned depending on the specified sex ratio (see parameter sex_ratio). If the parameter sex_ratio_threshold is set the sex of an offspring is determined by the phenotype of the first quantitative trait of the offspring. The argument specifies the threshold above which an individual becomes a male. If this parameter is set the parameter sex_ratio is not considered. Note, that when using this parameter the phenotype has to be defined at the offspring stage, thus the parameter quanti_environmental_proportion has to be set to 0. Using this parameter it is possible to simulate sex chromosomes.

5.3 Death

This life cycle event simply removes the adults present.

5.4 Regulation offspring

This event performs population regulation before dispersal, *i.e.* at the offspring stage. This life cycle event allows to regulate the population sizes down to carrying capacity. In fact the regulation should only be used if there is no regulation at the reproduction stage, *i.e.* if the parameter mating_nb_offspring_model is set to 1 (keep number), 2 (fecundity), 3 (fecundity stochastic) or 4 (fecundity binomial). If the parameter selection_position is set to 2 then selection acts at this stage. In this case the following parameter is ignored and offspring regulation happens following the parameter selection_level.

regulation_model_offspring [0,1] (default: 0)

0 : no regulation. No population size regulation takes place at the offspring stage, *i.e.* overcrowding can occur.

1 : random regulation. For each patch quantiNemo regulates the population size down to its carrying capacity if the population size exceeds carrying capacity. Individuals are thereby randomly removed. No regulation takes place if the population size is lower than carrying capacity.

5.5 Regulation adults

This event performs population regulation after dispersal, *i.e.* at the adult stage. Due to asymmetric dispersal, some patches may be overcrowded. This life cycle event allows regulating the population sizes down to carrying capacity. If the parameter selection_position is set to 3 then selection acts at this stage. In this case, the following parameter is ignored and offspring regulation happens following the parameter selection_level.

regulation_model_adults [0,1] (default: 0)

0 : no regulation. No population size regulation takes place at the adults stage, *i.e.* overcrowding can occur.

1 : random regulation. For each patch quantiNemo regulates the population size down to its carrying capacity if the population size exceeds carrying capacity. Individuals are thereby randomly removed. No regulation takes place if the population size is lower than carrying capacity.

5.6 Extinction

This event allows to randomly wipe out populations entirely or partially. It is useful to simulate different types of event like diseases or natural disasters. The probability that a population goes extinct is specified by the parameter extinction_rate and the parameter extinction_rate_survival specifies how much a population is affected. If the extinction rate is zero this event is skipped.

extinction_rate [decimal/matrix] (temporal/default: 0)
Extinction probability of a patch at each generation. Can be specified for each patch separately.

extinction_rate_survival
extinction_rate_survival_fem
extinction_rate_survival_mal [decimal/matrix] (temporal/default: 0)
These parameters specify how a population is affected when it is hit by an extinction event. By default (value 0) the entire population is wiped out. If the parameter is not 0 then at an extinction event individuals will survive. The effect on the population may be defined absolutely if the value is 1 or larger (e.g. 5 individuals survive), relatively if the value is between 0 and 1 (e.g. 10% of the individuals survive), or a combination of relatively and absolutely when specified patch specific using a matrix. The survival rate may be specified for each sex separately or in common. Note that in this later case the values will be used for each sex separately, e.g. if the parameter extinction_rate_survival is set to 0.4, then 40% of the females and 40% of the males will survive if an extinction event happens and if set to 100 and two sexes are simulated 50 females and 50 males will survive.

Chapter 6

Demography

This section describes how to simulate a specific demographical scenario, changing the initial number of individual in each patch, the patch capacity, the migration model and how the population growth with time.

6.1 Dimensions

generations [integer]

Number of generations to perform per replicate. This parameter is mandatory, and has no default.

patch_capacity [integer/matrix] (temporal)

Carrying capacities of the patches. This parameter is mandatory, and has no default.

If all patches have the same carrying capacities a single number as argument is sufficient to specify the carrying capacities. To set the carrying capacities individually for each patch a matrix is needed. Note, using a two-dimensional matrix with two columns allows addressing directly a patch and thus specifying carrying capacity explicitly for this patch (e.g. *{{patchID value}{patchID value}...}*), while all not specified patches will have a carrying capacity of 0.

patch_capacity_fem

patch_capacity_mal [integer/matrix] (temporal)
The carrying capacities may vary among sexes, patches, and time. The carrying capacities have to be specified either for each sex separately (parameters patch_capacity_fem and patch_capacity_mal), or for both sexes together (parameter patch_capacity). In the first case both sex specific parameters have to be set if two sexes are simulated. In the latter case the carrying capacities for females and males are assumed to be identical, *i.e.* the carrying capacity of females and males is *patch_capacity*/2. If hermaphrodites are simulated the parameters patch_capacity and patch_capacity_fem are identical. In case all three parameters are set, only the sex specific parameters will be used as they are more informative.

patch_number [integer] (default: 1)
This parameter specifies the number of patches in the metapopulation.

6.2 Initialization

This section allows specifying how the metapopulation is initialized. The population size of the patches may be set in three different ways. Also, the genotypes of the individuals of the initial populations may be set differently. By default, *i.e.* if no special parameters of this section are defined, the initial population size of each patch corresponds to its carrying capacity (see parameter **patch_capacity** above). If the initial population sizes deviate from the carrying capacities the parameter **patch_ini_size** allows setting the initial population size for each patch separately. Finally, it is possible to define the genotypes of each individual explicitly using an FSTAT file (Goudet, 1995). By doing this also the initial population sizes are defined. For details please have a look at the parameters **quanti_ini_genotypes** and **ntrl_ini_genotypes** in the chapter about genotype configuration. If a genotype file is present the initial population sizes are defined using this file.

patch_ini_size
patch_ini_size_fem
patch_ini_size_mal [integer/matrix]
These parameters allow to set the initial population sizes of the patches,

i.e. the populations sizes present at the beginning of the simulation. These parameters are optional. If none of these parameters is set, *i.e.* the initial population sizes are not set, the simulation will start with all the population sizes set at carrying capacity. The initial population sizes may vary among sexes, and patches. The initial population sizes have to be specified either for each sex separately (parameters patch_ini_size_fem and patch_ini_size_mal), or for both sexes together (parameter patch_ini_size). In the first case both sex specific parameters have to be set if two sexes are simulated. In the latter case the initial population sizes for females and males are assumed to be identical, *i.e.* the initial population size of females and males is $patch\_ini\_size/2$. If hermaphrodites are simulated the parameters patch_ini_size and patch_ini_size_fem are identical. In case all three parameters are set the sex specific parameters will be used as they are more informative. To set the initial population sizes individually for each patch a matrix is needed. The matrix is adjusted to the number of patches if necessary (see section 3.4). If all patches have the same initial population sizes a single number as argument is sufficient to specify the initial population sizes.

Note, using a two dimensional matrix with two columns allows addressing directly a patch and thus specifying the initial population size for this patch (e.g. *{{patchID value}{patchID value}...}*), while all not specified patches will have an initial population size of 0.

6.3 Population growth

mating_nb_offspring_model [0-9] (default: 0)

This parameter specifies how the total number of offspring (N_{Off}) produce at each generation is determined.

The actual number of offspring also depend on the selection level (see parameter selection_level). In case of soft selection, only the parameter mating_nb_offspring_mode is taken into account. However, if the selection is hard, then the total number of offsping is computed taking also into account the mean fitness of the population. Finally, in the case of metapopulation selection, the total number of offspring for the entire population is computed using only mating_nb_offspring_model,

but they are then reassigned to specific patches depending on the mean fitness of each patch.

0 : carrying capacity.

$$N_{Off} = K$$

The total number of offspring (N_{Off}) is set to the carrying capacity (K, parameter patch_capacity) of the patch or the metapopulation, respectively.

1 : keep number.

$$N_{Off} = N$$

The total number of offspring (N_{Off}) corresponds to the number of adults (N), *i.e.* the number of individuals is kept constant. Note that a regulation of the patch densities after dispersal can lead to an unwanted continuing reduction of the entire metapopulation size.

2 : fecundity.

$$N_{Off} = Poisson(N_F f)$$

The number of offspring (N_{Off}) depends on the mean fecundity of the females (f) defined by the parameter mean_fecundity.

3 : fecundity simple.

$$N_{Off} = round(N_F f)$$

A simplified version of point 2 assuming that the fecundity is always the same (no fluctuations). This simplification speeds up computation and is acceptable for a wide range of simulation problems.

4 : fecundity binomial.

$$N_{Off} = floor(N_F f) + Binomial(N_F f - floor(N_F f), 1)$$

Similar to point 3, but the rounding is replaced by random rounding with probability equal to the decimal part of the number, *i.e.* drawing a random number in a binomial distribution with probability equal to the decimal part of the number.

5 : fecundity limited.

$$N_{Off} = Poisson(N_F f)$$

$$if(N_{Off} > K) N_{Off} = K$$

Same as point 2, but if the new population size exceeds carrying capacity the population size is down-regulated to carrying capacity.

6 : fecundity simple & limited.

$$N_{Off} = round(N_F f)$$

$$if(N_{Off} > K) N_{Off} = K$$

Same as point 3, but if the new population size exceeds carrying capacity the population size is down-regulated to carrying capacity.

7 : fecundity binomial & limited.

$$N_{Off} = floor(N_F f) + Binomial(N_F f - floor(N_F f), 1)$$

$$if(N_{Off} > K) N_{Off} = K$$

Same as point 4, but if the new population size exceeds carrying capacity the population size is down-regulated to carrying capacity.

8 : logistic regulation.

$$N_{Off} = \frac{NK(1+r)}{N(1+r) - N + K}$$

The total number of offspring (N_{Off}) is logistically regulated following the discrete-time function of Beverton and Holt (1957) and depends therefore on the carrying capacity (K) and on the parameter growth_rate (r).

9 : stochastic logistic regulation.

$$N_{Off} = Poisson\left(\frac{NK(1+r)}{N(1+r) - N + K}\right)$$

Same as point 8, but the computation of the total number of offspring has a stochastic component.

Models and their specific parameters

model	additional parameters
carrying capacity (0)	
keep number (1)	
fecundity (2-7)	mean_fecundity
logistic regulation (8-9)	growth_rate

mean_fecundity [decimal] (temporal)

This parameter specifies the mean female fecundity. The parameter is mandatory (and only used) if the fecundity of the female specifies the number of offspring (i.e. parameter mating_nb_offspring_model set to 2 or 3).

Note that for population of hermaphrodites (mating_system 0,1 or 2), a mean fecundity of 1 leads to constant population size, while in dioecious systems (mating_system 3,4 or 5), it's a mean fecundity of 2 which leads to a constant population size.

growth_rate [decimal] (temporal)

This parameter is mandatory (and only used) if logistic regulation is used to specify the number of offspring (i.e. parameter mating_nb_offspring_model set to 4 or 5). It specifies the growth rate r of the population using the discrete-time function of Beverton and Holt (1957).

When the population is much smaller than the carrying capacity, $r+1$ can be thought as the mean number of offspring created per individual. So if $r = 0$, the mean number of offspring per individual is 1, and the population size is constant.

6.4 Dispersal

During its life cycle, each individual might migrate from one patch to another. Several dispersal models are available (see parameter dispersal_model). It is also possible to specify a dispersal matrix, which will have precedence over other dispersal parameters. After dispersal, individuals become adults. By default (if none of the following parameters are set) individuals do not disperse among patches.

dispersal_rate
dispersal_rate_fem
dispersal_rate_mal [decimal/matrix] (temporal/default: 0)

These parameters allow setting the emigration rate (if the argument is between 0 and 1 (1 included), or the number of emigrants (if the argument is an integer).

If the argument is a single value the dispersal model used depends on the other parameters of this section. But it is also possible to specify the dispersal rate explicitly between each pair of patches (for both directions) if the argument is a matrix. A dispersal matrix has precedence over all other dispersal settings. The matrix must be `patch_number` x `patch_number` in dimensions. Each d_{ij} element of this matrix is the dispersal probability from patch i to patch j, where i specifies the row and j the column of the matrix. Consequently, the values in a row must sum up to 1 (if the row do not sum up to 1 the proportion of residents (diagonal) will be adjusted and a warning returned). The dispersal rates can either be specified for both sexes in general or for each sex separately. If the dispersal rates are sex specific the dispersal rates for both sexes have to be specified and they have to be in the same format (matrix or a single dispersal rate). Sex specific dispersal rates have precedence over a general dispersal rate. Note, that the dispersal matrix has to be fully specified, *i.e.* the matrix is not adjusted to the number of patches as for other parameters.

dispersal_model [0-4] (default: 0)

The following dispersal models are available:

- **0 : Migrant-pool Island model.** If the dispersal rate is m and the number of patches is n_p, the probability to disperse to any $n_p - 1$ non-natal patch is $\frac{m}{n_p - 1}$ while the probability to stay at home is $1 - m$.
- **1 : Propagule-pool Island model.** In that modified version of the Island model, emigrants tends to move between patches in group. More precisely, for each patch, a fraction of individual of size $m\varphi$ will migrate altogether to another randomly picked patch. Another fraction of emigrant, $m(1 - \varphi)$, will migrate to all others patches. Finally, a fraction of $1 - m$ individuals will stay at home.

2 : 1D Stepping-Stone model. In the one dimensional Stepping Stone model patches are placed on a line and migrants can only move to one of the two adjacent patches. If the dispersal rate is m, the probability to disperse to one of the adjacent patches is $m/2$ while the probability to stay at home is $1 - m$. The parameter `dispersal_border_model` allows to specify how to treat the border patches.

3 : 2D Stepping-Stone model. In the two dimensional Stepping-Stone model patches are placed on a grid (or lattice) and migrants can move to 4 or 8 adjacent patches (set by the dispersal_lattice_range parameter below). If the dispersal rate is m, the probability to disperse to one of the adjacent patches is $m/4$ or $m/8$ depending on the parameter dispersal_lattice_range, while the probability to stay at home is $1 - m$. The parameter dispersal_border_model allows to specify how to treat the border patches and the parameter dispersal_lattice_dims allows specifying the dimensions of the grid.

Models and their specific parameters

model	additional parameters
0 : Migrant-pool Island	
1 : Propagule-pool Island	dispersal_propagule_prob
2 : 1D Stepping-Stone	dispersal_border_model
3 : 2D Stepping-Stone	dispersal_border_model / dispersal_lattice_range / dispersal_lattice_dims

dispersal_lattice_range [0,1] (default: 0)

This parameter sets the number of neighboring patches used for dispersal. The dispersal probabilities to these adjacent patches are $m/4$ in the first case and $m/8$ in the second. This parameter is only used in the 2D Stepping-Stone model (parameter dispersal_model set to 3).

0 : 4 neighbors. 4 adjacent patches (up, down, left and right)

1 : 8 neighbors. 8 adjacent patches (as before plus the diagonals)

dispersal_border_model [0-2] (default: 0)

This parameter specifies how the patches at a border of the Stepping Stone model should be treated:

0 : Circle/Torus. In the 1D Stepping-Stone model the first and last patches are connected to each other by migration leading to a circle. In the 2D Stepping-Stone model individuals of an edge patch may migrate to the other side leading to a torus (donut world). This means that there are no edges, eliminating any such effects.

1 : Reflective boundaries. The borders are reflective. Dispersers from the border patches cannot move beyond the border. Border cells have thus less cells connected to them and their dispersal probabilities to the adjacent patches are higher (e.g. m for the 1D Stepping-Stone model, $m/3$ (corners $m/2$) for the 2D Stepping-Stone model with four adjacent cells, and $m/5$ (corners $m/3$) for the 2D Stepping-Stone model with eight adjacent cells). No dispersers are lost.

2 : Absorbing boundaries. Dispersers of the border patches are lost if they choose to move beyond the border. The dispersal probabilities of a border patch are not modified.

dispersal_lattice_dims [matrix]

This parameter allows to specify the length and width of the 2D Stepping-Stone lattice (only used when dispersal_model is set to 3). The argument is an integer matrix with two values. The first value stands for the number of rows, and the second value for the number of columns. The product of the two values results in the number of patches and thus must match the parameter patch_number. If the parameter is not set quantiNemo assumes that the 2D Stepping-Stone lattice is quadratic. If this is not possible due to the number of patches an error is returned.

dispersal_propagule_prob [decimal] (temporal/default: 1)

This parameter is only used for the Propagule-pool Island model (parameter dispersal_model set to 1). It specifies the probability that a migrant will move to the propagule-assigned patch, *i.e.* this is also the proportion of emigrants of a patch which migrate to the same non-natal patch. A probability of 1 means that all emigrants migrate to the same non-natal patch, while a value of 0 means that all emigrants migrate to any patch, but the natal and the propagule patch.

6.4.1 Density dependent dispersal rate

By default, the dispersal rate is not influenced by the population size/density of the natal patch. The following parameters allow describing a situation where the migration depends on the density of the natal patch:

dispersal_rate_model [decimal] (temporal/default: 0)
this parameter define how the dispersal rate is set.

0 : Flat rate. The rate is constant and does not depend of the density

1 : Dependant rate 1 Rate depend on density following the relation $m_{\text{eff}} = \frac{me}{N} * (N - K(1 - \exp(-N/K)))$, where m is the migration rate, K the patch capacity, N the number of individual on the patch before migration and e the Euler constant. Notice that in this case, the migration rate is exactly m (parameters dispersal_rate) when the density is one.

2 : Dependant rate 2 Rate depend on density following the relation $m_{\text{eff}} = \frac{(N-K(1-\exp(-N/K)))}{N}$. Notice that in this case, the migration rate specified using the parameter dispersal_rate is not used, and instead the migration rate is smoothly adjusted so that the density after dispersal is always smaller than one.

Chapter 7

Genotype configuration

7.1 Introduction

[FM: I'm still not sure about the title of this chapter. It was first "genetics", which is probably too broad, I then though about "genome prototype", but Guillaume doesn't find it very clear. We could also say "reference genome", though it's not exactly that neither.] QuantiNemo allows the simulation of quantitative traits as well as neutral markers, both related by a common genetic map so that processes like linkage disequilibrium and genetic hitchhiking can be simulated easily. More precisely, quantiNemo allows to simulate neutral markers, such as microsatellites or SNPs with different mutation models (K allele and Stepwise). Quantitative trait under selection pressure composed of various loci can also be added to the same simulation. In this chapter, we explain how to define loci and traits, locate them in a map, set an initial genotype and choose a mutation model for them.

Most of the parameter and behaviour are the same for neutral and quantitative traits (except of course that quantitative traits are under selection, but this will be cover in the next chapter). The main difference is that neutral trait's parameters are defined with a ntrl_ in front while quantitative traits are specified with quanti_. In the case where the parameters have the same meaning for neutral and quantitative traits, the (ntrl/quanti)_parameter notation will be used, which should be read as the existence of two param-

eters, one ntrl_parameter and another one quanti_parameter. If differences are present ,as for example in the mutations models, both parameters are explained separately.

7.2 Defining loci

(ntrl/quanti)_loci [integer]
This parameter specifies the number of neutral/quantitative loci per individual per traits (by default, there is only one trait). This parameter is mandatory for the simulation of a neutral/quantitative locus *i.e.* it is necessary as soon as we want to study genetic effect. If this parameter is not present, the other parameters related to genetics make no sense and will not be loaded.

(ntrl/quanti)_all [1-256/matrix] (default: 255)
This parameter specifies the maximal possible number of alleles per locus. Using a matrix it is possible to specify this number for each locus of a trait separately. For quantitative trait, it is recommended to take a odd number of allele in order to have a "central allele", but it's not mandatory.

(ntrl/quanti)_allelic_file [string] (default: "")
This parameter allows to pass the name of a file containing allele informations, such as the initial allele frequencies, and/or the mutation probability to an allele. The number of alleles and loci has to be in line with the parameters (ntrl/quanti)_loci and (ntrl/quanti)_all. The information can be set globally for all loci, if they have the same specifications, or for each locus separately. The allelic files for neutral markers and for quantitative trait have similar format except for two columns:

```
#Allelic file
##################
[FILE_INFO]{
  col_locus 1
  col_allele 2
  col_allelic_value 3
  col_mut_freq 4
  col_ini_freq 5
}
```

```
#locus  allele  value   mut_freq  ini_freq
1       1       -1.     0.2       {0  0.33}
1       2       0       0.4       {0  0.33}
1       3       2.      0.4       {1  0.33}
2       1       -1.     0.33      {0  0.33}
2       2       0.      0.33      {0  0.33}
2       3       1.      0.33      {1  0.33}
```

The file has to start with a file information box [FILE_INFO]{...}. This box contains the information of the structure of the following table allowing a flexible structure of the table. For example the order of the columns in the table may vary, or columns may be ignored. The file information box starts with the keyword [FILE_INFO] and the information is enclosed by brackets "{...}". Line by line the index of the columns to be read have to be declared. Thereby a keyword for the specific setting is followed by the column index (the ordering starts with 1). The following column keywords are available:

col_locus This keyword specifies the column containing the locus index. If this column is not declared in the file information box, quantiNemo will use the same settings for all loci. In this case, the length of the table must meet the number of alleles (ntrl_all). If this keyword is declared the length of the table must meet the number of alleles times the number of loci (ntrl_loci * ntrl_all).

col_allelic_value This keyword specifies the column containing the allelic effects, *i.e.* how an allele contribute to the value of a trait (c.f. 8.2.1). If this column is not set the allelic effects will be equally spaced on a range which depend of the parameter quanti_all_effect_var. Since only quantitative trait have an allelic value, this column is only available for quantitative traits.

col_allele This keyword specifies the column containing the allele index. This column is mandatory. The index of the allele goes from 1 to quanti/ntrl_loci.

col_mut_freq This keyword specifies the column containing the mutation probabilities, *i.e.* the probability to mutate to this allele when a mutation occurs. The behaviour of this mutation probability depends on the mutation model (see parameter (ntrl_mutation_model).This keyword is only available for quantitative traits.

col_ini_freq This keyword specifies the column containing the initial frequencies of the alleles. This column allows to explicitly set the allele frequencies at the start of a simulation. The frequencies can be set for each patch separately using a matrix. In the example above, individuals of the first population are initially fixed for the allele 3 at the first locus as well as at the second locus. In the second population, all alleles have the same initial frequency of 0.2. Note, that the matrix is adjusted in length if the number of populations does not correspond to the length of the matrix. If this column is not given the initial allele frequencies are set globally depending on the parameter ntrl_ini_allele_model.

Note, that as in all input files for quantiNemo it is possible to define comments (also in the file information box) using the hash character: '#' or '#/ ...any text... /#'.

7.3 Initial genotypes

There are several methods to set the allele frequencies or even the genotypes of the individuals at the start of a simulation (initialization). The genotypes of the individuals may be set using an FSTAT file (Goudet, 1995) (parameter **(ntrl/quanti)_ini_genotypes**). If such a FSTAT file is not present the genotypes are randomly drawn following the explicitly set allele frequencies in the allelic file (see parameter **(ntrl/quanti)_allelic_file** and especially column keyword **col_ini_freq**). If the allele frequencies are not set explicitly in the allelic file the initialization is performed following the parameter **(ntrl/quanti)_ini_allele_model**.

(ntrl/quanti)_ini_genotypes [string] (default: "")
This parameter allows to specify a name of an FSTAT file (Goudet, 1995) containing the initial genotypes of the individuals for each population. If such a file is present the initialization of the metapopulation is done solely by this file ignoring the parameters **(ntrl/quanti)_ini_allele_model** and **patch_ini_size**. A single FSTAT file is needed for all quantitative traits together containing the total number of loci of all quantitative traits together. Note, that quantiNemo allows to output an appropriate file for any generation (see parameter **(ntrl/quanti)_save_genotype**).

This allows resuming a simulation, to generate tailored initial conditions, or to continue a simulation with modified settings. If the parameter **(ntrl/quanti)_save_genotype** is set to 2 an extended FSTAT file is generated. Also, this file may be used to initialize the metapopulation. In this case, quantiNemo overtakes the supplement information provided by the file, especially the sex and age of the individual, the index of the individual, its mother and father. Thus the supplement information allows resuming a simulation without the loss of the pedigree. Note, that for an entire resume of a simulation the genotypes of both neutral and quantitative traits have to be set. In this case the number of individuals per patch and the individuals supplement information (sex, age, individuals id, mothers id, fathers id must be in agreement with each other.

ntrl_ini_allele_model [0,1] (default: 0)

If the genotypes or allele frequencies are not already defined in another way, the initialization of the genotypes may be either polymorphic, where the probability of each allele is identically or monomorphic, where all populations are fixed for a single allele.

0 : polymorph. The populations are maximally polymorph with respect to allele frequencies at the start of a simulation.

1 : monomorph. The populations are monomorphic with respect to allele frequencies at the start of a simulation. All individuals are fixed for a single allele, which is the "middle" allele, *i.e.* the allele with the index $\lfloor \mathsf{ntrl\_all}/2 \rfloor$.

quanti_ini_allele_model [0,1] (default: 0)

If the genotypes or allele frequencies are not already defined in another way, the initialization of the genotypes may be either polymorphic, where the probability of each allele follow a normal distribution or monomorphic, where all populations are fixed for a single allele.

0 : polymorph. The initial allele frequency is given by a (discretized) normal distribution with a variance given by the parameter **quanti_allelic_var**.

1 : monomorph. The populations are monomorphic with respect to allele frequencies at the start of a simulation. All individuals are fixed for a single allele, which is the "middle" allele, *i.e.* the allele with the index $\lfloor$ntrl_all/2$\rfloor$.

7.4 Mutation

Mutation rates may be defined for each locus individually by explicitly defining the individual mutation rates (parameter (ntrl/quanti)_mutation_rate) or by defining the gamma distribution from which the individual mutation rates are drawn (parameters (ntrl/quanti)_mutation_rate and (ntrl/quanti)_mutation_var). For quantitative trait, depending on the mutation model (parameter quanti_mutation_model) the mutant effect is the effect of the drawn allele (model 0), or the effect of the drawn allele is added to the current allelic effect to get the mutant effect (model 1). Using the allelic file (see section 7.2) it is possible to specify for each allele its effect and the probability to mutate to this allele given that there is a mutation. A minimal definition for mutations requires the setting of a common mutation rate (parameter (ntrl/quanti)_mutation_rate has a single value). In this case, all loci have the same mutation rate.

Note, that for all mutation models the number of alleles has to be odd if the allelic effects and the probabilities to mutate to the alleles given that there is a mutation are set automatically. In this case, also a warning will be drawn if the number of alleles is below 200, informing that this number of alleles may not well represent the normal distribution of the allelic effects. the probability of each allele is identically or monomorphic, where all populations are fixed for a single allele.

(ntrl/quanti)_mutation_rate [decimal/matrix] (temporal/default: 0)
This parameter specifies the mutation rate per locus and generation. If the argument is a single value the mutation rate for all loci is the same. By passing a matrix of mutation rates it is possible to set the mutation rate for every single locus individually. By default, no mutations occur.

7.4.1 quantitative mutation model

quanti_mutation_model [0-3] (default: 0)

This parameter allows specifying the mutation model, which define toward which allele we mutate when a mutation occurs. For quantitative traits, quantiNemo offers the possibility to simulate four very common models. Notice that the two first one are more common in quantitative traits while the two last ones are more meaningful for neutral markers, and are proposed here only as a "nice to have".

0 : RMM (Random Mutation Model)

The new allele is randomly selected based on its index, following a Gaussian centred on the allele with an allelic value of 0 (the so-called central allele). The standard deviation of the Gaussian is the same as the one from which the allelic effects are drawn randomly (see parameter quanti_allelic_var). This simulates a system where every mutation has an allelic value normaly distributed around 0 with a variance of quanti_allelic_var.

If the probability to mutate to a given allele are given explicitly in the allelic file (see section 7.2), then, the new allele is randomly selected following the given distribution.

1 : IMM (Increment Mutation Model)

The new allele is randomly selected following a Gaussian which is centered around the allelic value of the previous allele, *i.e.* the allele it is mutating from. The standard deviation of the Gaussian is the same as the one from which the allelic effects are drawn randomly (see parameter quanti_allelic_var). This simulates a system where every mutation has an effect which is normally distributed around the value of the previous allele.

Note that with this model, the allelic values are equally space between -20σ and 20σ instead of -6σ and 6σ (where σ is the square root of the parameter quanti_allelic_var) as it is the case for the other mutation model. This lets more freedom to the system to explore value far away from the starting point. It is therefore recommended to select a large number of allele to have a good representation of the possible values.

If the probability to mutate to a given allele are given explicitly in the allelic file (see section 7.2), then, the new allele is randomly

selecting following this given distribution. However, the distribution is re-centred around the current allele. If the new allele does not exist, no mutation is performed.

2 : KAM (K-Allele Model)
At each mutation, the existing allele is randomly exchanged by another allele within the range of alleles (i.e. $[1; quanti\_all]$) not taking into account any allelic effect. The probability to mutate to any allele is the same. In this case, the frequencies that are given in the allelic file (see section 7.2) are not taken into account.

3 : SSM (Single Step Mutation)
In the single step mutation, the new allele depends on the current allele index. When a mutation occurs the current allele is replaced by one of its neighbouring alleles. For example, if the allele with the index 12 mutates, it changes either to the allele with the index 13, or to the allele with the index 11. The boundaries are reflexive, *i.e.* the allele index cannot exceed the range of alleles (i.e. $[1; ntrl\_all]$). Notice also that in quantiNemo the allelic value depend linearly on the allelic index (unless specified otherwise in the allelic file) so that this is equivalent to change incrementally the allelic value.

The different models are schematically described in Fig. 7.1, where the probability to mutate to a given allele is given taking into account the previous allele and the reference allele (which has an allelic value of 0).

7.4.2 neutral mutation model

ntrl_mutation_model [0,1] (default: 0)
In the neutral case, there is only access to two of the model, since the allelic value are not define and some of the model don't make sense.

0 : KAM (K-Allele Model)
At each mutation the existing allele is randomly exchanged by another allele within the range of alleles (i.e. $[1; quanti\_all]$) not taking into account any allelic effect. The probability to mutate to any allele is the same.

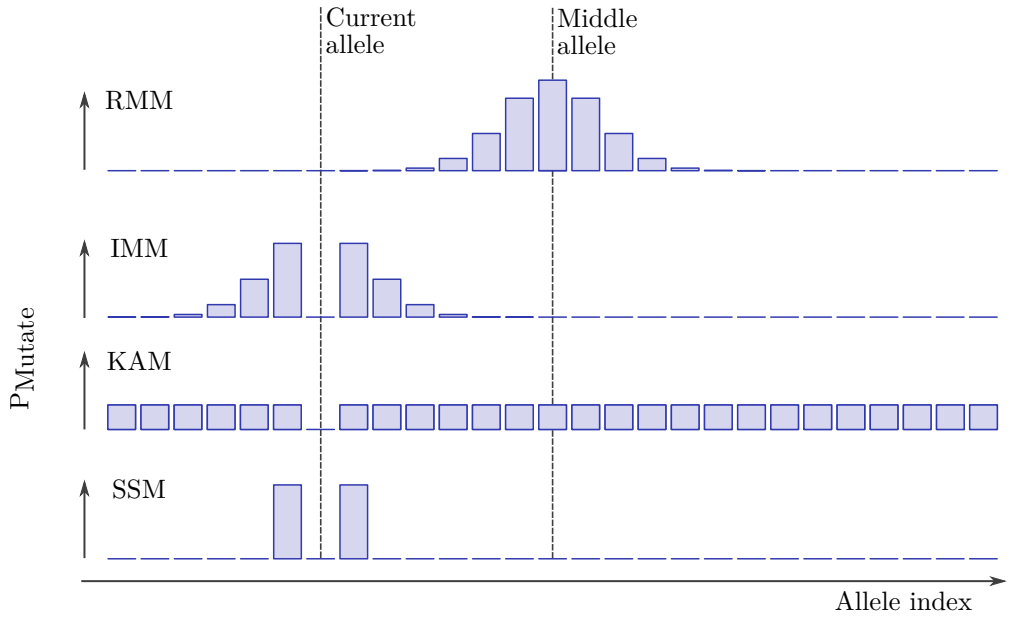


Figure 7.1: Schematic representation of the various mutation model for a system with 27 alleles, where the middle allele is 14 and the allele from which we mutate (present allele) is 7. P_{Mutate} represent the probability to mutate to a given allele. Notice also that the allelic value is almost equivalent as the allele index in quantiNemo

1 : SSM (Single Step Mutation)
In the single step mutation, the new allele depends on the current allele index. When a mutation occurs the current allele is replaced by one of its neighboring alleles. For example, if the allele with the index 12 mutates, it changes either to the allele with the index 13, or to the allele with the index 11. The boundaries are reflexive, *i.e.* the allele index cannot exceed the range of alleles (i.e. $[1; ntrl\_all]$).

7.5 Multiple traits

QuantiNemo allows simulating multiple traits (quantitative and/or neutral) simultaneously. Each trait has its own architecture and quantitative traits may be under different selection pressures.

(quanti/ntrl)_nb_trait [integer] (default: 1)
This parameter defines the number of traits. Notice that it is possible to have ntrl traits. This has to be thought has a set of neutral loci which share common parameters, like mutation model or mutation rate.

Each trait may have its own specifications, but it is also possible to specify parameters for some traits together, named grouping. If several traits are used it is possible to address a certain trait by its number. For instance to specify a parameter for the fifth trait one has to append a `"_5"` to the parameter name. In contrast if for the fifth trait no parameter with the suffix `"_5"` is passed quantiNemo checks if the parameter is passed for the fourth trait (suffix `"_4"`). If this is also not the case quantiNemo checks if the parameter is passed for the third trait (suffix `"_3"`), and so forth until a parameter is found. Note, that a parameter without a suffix is the same as the parameter with the suffix `"_1"`. This behavior of quantiNemo allows specifying parameters for a group of traits. Sometimes this grouping is not desired. It can be suppressed by setting 'NOT_SET' as an argument. In this case, this parameter and all parameters inheriting this parameter will use the default argument. An example for quantitative traits may make it clearer:

```
quanti_nb_trait        12
```

```
quanti_loci              5
quanti_loci_7            10

quanti_all_1             10
quanti_all_4             20
quanti_all_7             10
quanti_all_10            20

quanti_allelic_file_1  file_1.txt
quanti_allelic_file_4  file_2.txt
quanti_allelic_file_7 NOT_SET

quanti_mutation_rate 0.0001
```

In this example, we simulate 12 quantitative traits. Traits 1 to 3 consist of 5 loci with up to 10 alleles, traits 4 to 6 consist of 5 loci with up to 20 alleles, traits 7 to 9 consist of 10 loci with up to 10 alleles, and traits 10 to 12 consist of 10 loci with up to 20 alleles. All traits have the same mutation rate of 0.0001. Allele characteristics for traits 1 to 3 are specified in file "file_1.txt", allele characteristics for the traits 4 to 6 in the file "file_2.txt". Since 'NOT_SET' is set for the parameter quanti_allelic_file_7 all allele characteristics for traits 7 to 12 are set automatically without any file.

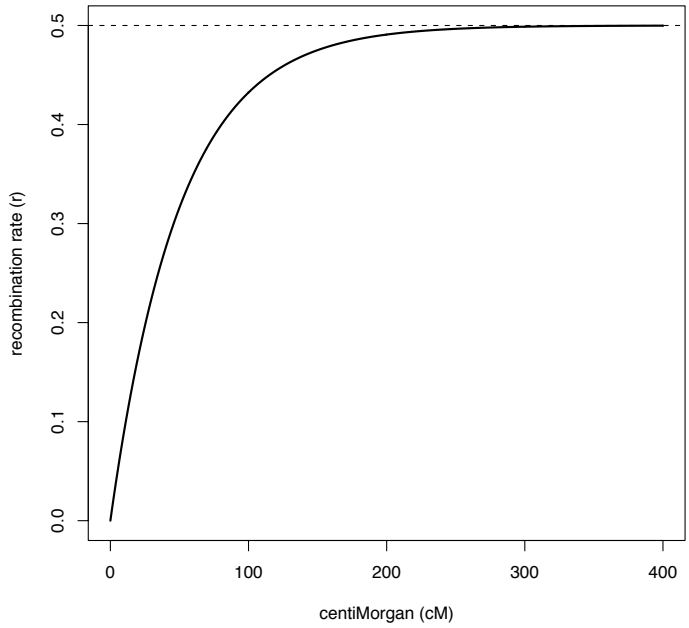
7.6 Genetic map

QuantiNemo has an underlying genetic map, which may consist of several chromosomes. This allows explicitly positioning all types of loci on the map (quantitative trait loci (QTL) and neutral markers). The unit of the genetic map is centi Morgans (Haldane, 1919, cM). This means that between two loci separated by 100 cM on average a single recombination event per meiosis is expected. The relation between centiMorgan (m) and recombination rate (r) is given as

$$r = \frac{1 - e^{-2m/100}}{2}$$

.

To simulate linked loci two types of parameters have to be set: One to specify



the locus positions on the genome (in cM) and one for each trait (QTL or neutral) to attribute the loci of the trait to these locus positions. If a locus of a trait is not attributed to a position then quantiNemo2 assumes that this locus is unlinked to any other locus. A genome is identical for everybody, however, sexes may differ in the rate of recombinations, e.g. hotspots of recombination, or degenerated Y-chromosome (see also parameter **sex_ratio_threshold**). quantiNemo2 accounts for this by allowing specifying different genetic maps (in cM) for each sex using the suffixes **_fem** and **_mal**. Note, that in this case, the order of loci must be identical, only distances between loci may differ.

genome
genome_fem
genome_mal [matrix] (default: "")

These parameters allow specifying the positions of loci in centi Morgans for all traits in common. Brackets separate different chromosomes and within a chromosome, the positions of loci are defined cumulatively, *i.e.* the position of the locus is defined as the length to the start of the

chromosome. Note, that chromosomes may contain different numbers of loci. It is possible to skip chromosomes if they don't contain loci (see section 3.4). If the genetic map is sex specific both parameters have to be set. If hermaphrodites are simulated the parameters **genome** and **genome_fem** are identical. If all three parameters are set, only the sex specific parameters will be used. The genome may only be specified in general (i.e. parameter **genome**) or for each type of trait separately (i.e. parameters **quanti_genome** and **ntrl_genome**).

(quanti/ntrl)_genome
(quanti/ntrl)_genome_fem
(quanti/ntrl)_genome_mal [matrix] (default: "")

Similar to the parameters above these parameters allow specfiying the positions of loci in centi Morgans. In contrast to the parameters above these parameters allow specifiying the positions for each type of trait separately.

Example:

```
genome { {1: 10}
         {3: 10 40 60 80 100}
         {   10 40 60 80 100} }
```

In the example above, the genetic map consists of 11 loci located on three of at least four chromosomes. The first chromosome contains a single locus at position 10 cM. No loci are located on the second chromosome. The third and fourth chromosomes have 5 loci at positions 10, 40, 60, 80, and 100 cM.

(quanti/ntrl)_locus_index [matrix] (default: "")

This parameter allows assigning each locus of a trait (quanti or ntrl) to a locus position on the genome (see above). The parameter has to be set for each trait separately and the assignment of the locus to a locus position on the genome is made based on the index of the locus on the entier genome (starting with 1). Note, that the indexing is ignoring any chromosomal structure. Note also, that the indexing depends on the corresponding genome, *i.e.* if it is specified in general (parameter **genome**) or trait type specific (e.g. parameter **quanti_genome**).

Note further, that this parameter is not sex specific, since the order of the loci is identical for both sexes.

Example 1 (trait type sepcific):

```
quanti_nb_trait       2
quanti_loci           3
quanti_genome         {0.1 0.2 0.3 0.4 0.5 0.6}
quanti_locus_index_1  {1 2 5}
quanti_locus_index_2  {3 4 6}
ntrl_nb_trait         2
ntrl_genome           {0.1 0.6 0.7 0.8 0.9 1.0}
ntrl_locus_index_1    {1 2 5}
ntrl_locus_index_2    {3 4 6}
```

Example 2 (general):

```
genome                {0.1 0.1 0.2 0.3 0.4 0.5 0.6 0.6 0.7 0.8 0.9 1.0}
quanti_nb_trait       2
quanti_loci           3
quanti_locus_index_1  {1 3 6}
quanti_locus_index_2  {4 5 7}
ntrl_nb_trait         2
ntrl_locus_index_1    {2 8 11}
ntrl_locus_index_2    {9 10 12}
```

Both examples are identical, the genome is defined once for each type of trait separately (Example 1), and once in general (Example 2). In the example, we have 5 QTLs and 6 neutral markers. At 0.1 cM we have a QTL and a neutral marker without any recombination between them.

recombination_factor
recombination_factor_fem
recombination_factor_mal [decimal/matrix] (temporal/default: 1)

These parameters allow setting a factor for the recombination rate. This factor is then multiplied with the positions of the loci on the chromosomes allowing easily stretching or shrinking a chromosome, keeping the relation between the loci intact. These parameters allow simulating the evolution of the recombination rate. To achieve this it is possible to use the phenotype (Z) or the genotypic value (G) of a quantitative trait as recombination factor. Thereby the key character G and Z are

followed directly by the quantitative trait index.

Example:

```
quanti_genome          {1 4}{1 10}
recombination_factor_mal {2 0.5}
recombination_factor_fem {Z1 Z2}
```

In this example, the genome consists of two chromosomes containing each 2 loci. A recombination factor is set separately for females and males. The recombination rate for males is fixed and modifies the positions of the first chromosome by a factor of two (increasing the recombination rate among the loci) and by a factor of 0.5 for the second chromosome (reducing the recombination rate among the loci). In contrast, the recombination factor for females is variable and is given by the phenotype of the first and second quantitative trait for the first and second chromosome, respectively.

Chapter 8

Quantitative traits and selection

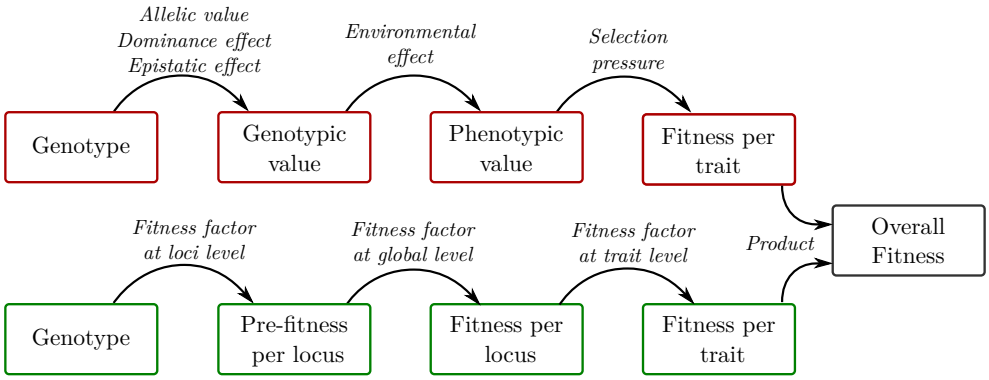
8.1 Introduction

In quantiNemo, selection is based on the fitness of individuals. It generally acts at reproduction level (for efficiency reason) but can also act at the regulation stage. More precisely, in a first step, the fitness of all individuals of the population are computed based on their genetic information (see below). Depending on the type of selection (soft/hard), the fitness can be further rescaled by the mean fitness of the patch/metapopulation. Then, during breeding, the parents of each individual are randomly selected proportionality to their fitness.

Two different procedure can be used to specify the fitness of an individual. One follows the philosophy of quantitative genetic, calculating the phenotypic value of each trait and applying selection pressure to it. The other one is closer to a population genetic approach, where the fitness is specified explicitly for each genotype and locus. The two approaches can be combined, in different traits (some defined using one approach and the other using the other approach) but also in a single trait. In this case, the final fitness of the trait is the product of the fitness obtained by both procedures.

8.1.1 Quantitative genetic approach

[FM: I'm not sure if it's better to speak about phenotypic value or just



about phenotype, which for me is more vague but maybe make more sens to new comer] To compute the fitness of an individual, quantiNemo define a genotypic value, which is the value of a trait would have if no environmental effect plays a role (therefore, if environmental effect are set to zero, the genotypic value is equal to the phenotypic value). So first, the genotypic value for each traits is computed, from the allelic values, the dominance model and if present the epistatic value. Then, the phenotypic value is computed by adding to the genotypic value the environmental value. Overall, the phenotypic value is given by:

$$P = G + E \tag{8.1}$$

$$G = \epsilon + \sum_{i=1}^{nbLoci} a_i + a_{i'} + k_{ii'}|a_{i'} - a_i| \tag{8.2}$$

Where P is the phenotypic value, G the genotypic value, E the environmental effect, ϵ the epistatic effect, a_i and $a_{i'}$ the allelic value and $k_{ii'}$ the dominance factor.

In a second step, the fitness of every trait is computed from its genotypic value using one of the models of selection pressure *e.g.* stabilizing selection where the value of the fitness decrease when the phenotypic value is far from its optimal value. Finally, the overall fitness of each individual, which is the product of the fitness of each trait is computed.

$$W = \prod_t W_t \tag{8.3}$$

where W is the overall fitness and W_t is the fitness of each trait.

8.1.2 Population genetic approach

In this approach, the fitness of an individual is considered to be a product of the fitness of each of its loci. The fitness of its loci is obtained through what is called fitness factor, which directly relates the genotype of a locus to its fitness. This factor can either be specified for each genotype and each locus explicitly using the dominance file (see sub-section: 8.2.2) or globally depending on the heterozygosity (see section 8.6)

In the following sections, we explain in details the parameters to define quantitative traits, the selection pressure acting and other parameters affecting the fitness and the selection.

8.2 From genotype to genotypic value

In this section, we explain how quantiNemo compute the genotypic value from a given genotype taking into account the value of every allele, dominance effect and if it's present, epistasis.

8.2.1 Allelic effects

Each allele has an allelic effect, its contribution to the genotype, which is represented by a number, the allelic value. There are two possibilities to define these effects. Either they are defined explicitly for each allele using a separate file (see parameter quanti_allelic_file in section 7.2), or they can be defined by specifying the variance of the normal distribution of the genotypic value (see parameter quanti_allelic_var). If the allelic effects are defined in both ways the explicitly defined effects are used. Note, that all effects have to be defined in the same way, *i.e.* explicitly or by their distribution.

quanti_allelic_var [decimal/matrix] (default: 1)

If the allelic values are not defined explicitly using the allelic file (params. **(ntrl/quanti)_allelic_file**), quantiNemo will proceed as follow. The allelic values will be equally spaced between -6σ and 6σ, where σ is the square root of the variance given by **quanti_allelic_var**. If the initial frequencies and/or the mutations frequencies toward each allele are not set by the allelic file, these frequencies will be set by a (discretized) normal distribution with a variance given by the parameter **quanti_allelic_var**. If dominance is purely additive, this ensures that through the population, the variance of the genotype is $2 * quanti\_allelic\_var$ (the factor 2 comes from the diploid character of individuals).

Note that if the number of alleles is small (smaller than 6), the allelic values will be spaced between $-(l-1)\sigma$ and $(l-1)\sigma$ instead of -6σ and 6σ where l is the number of alleles. So for example, for a bi-allelic system, we would have two allelic values at $-\sigma$ and σ.

8.2.2 Dominance effects

By default alleles are considered to be purely additive. Dominance effects can be defined in different ways. Either the dominance effect of specific pairs of alleles are defined explicitly using a separate file (parameter quanti_dominance_file), or by defining the normal distribution from where the dominance effects are randomly drawn (see parameters quanti_dominance_mean and quanti_dominance_var). An explicitly defined dominance effect has precedence over the general settings. By default, the parameters quanti_dominance_mean and quanti_dominance_var are set to zero resulting in a purely additive genotypic value of a locus. There are also two methods to define dominance:

quanti_dominance_model [0,1] (default: 0)

This parameter allows to choose among two models to define dominance effects.

0 : method k. $G_i = a_i + a_{i'} + k_{ii'}|a_i - a_{i'}|$

1 : method h. $G_i = 2[(1 - h_{ii'})a_i + h_{ii'}a_{i'}]$

Where G_i is the genotypic value of locus i, a_i and $a_{i'}$ are the effects of the two alleles at locus i, whereas the effect of a_i is smaller than the effect of $a_{i'}$. $k_{ii'}$ and $h_{ii'}$ are the dominance values between allele i and i' for method 1 and method 2, respectively. k and h can have the following effects:

Effect	method k	method h
overdominance	$k < -1$	$h < 0$
smaller allele a_i is dominant	$k = -1$	$h = 0$
purely additive	$k = 0$	$h = 0.5$
larger allele $a_{i'}$ is dominant	$k = 1$	$h = 1$
underdominance	$k > 1$	$h > 1$

quanti_dominance_file [string] (default: "")

This parameter allows to pass the name of a file containing the dominance effects and/or the fitness factor (see section 8.6). The information can be set globally for all loci, if they have the same specifications, or for each locus separately. For all not specified allele pairs other settings or the default values are applied. The dominance file has a similar format as the allelic file:

```
#Dominance file
######################
[FILE_INFO]{
  col_locus          1
  col_allele1        2
  col_allele2        3
  col_dominance      4
  col_fitness_factor 5
}

#locus  allele1  allele2  dominanc fitness
1       1        1        99999    1
1       1        2        0.298    1
1       1        3        0.435    1
1       1        4        0.224    1
1       2        2        99999    1
1       2        3        0.104    1
1       2        4        0.974    1
1       3        3        99999    1
1       3        4        0.808    1
1       4        4        99999    0
```

The file has to start with a file information box [FILE_INFO]{...}. This box contains information about the structure of the following table and thus allowing flexibility in the format of the table. For example, the order of the columns in the table may vary, or some columns may be ignored. The file information box starts with the keyword [FILE_INFO]. This keyword is followed by brackets "{...}" within which the user has to specify the contents of the columns to be considered by quantiNemo. Each column is specified by a pair consisting of a keyword (e.g. col_locus followed by the column number (the ordering starts with 1). Each column definition has to be on a new line. The following column keywords are available:

col_locus This keyword specifies the column containing the locus index. If this column is not declared in the file information box, quantiNemo will use the same settings (dominance effect and/or fitness factor) for all loci.

col_allele1 This keyword specifies the column containing the index of the allele with the smaller allelic effect. This column is mandatory. The index of the allele goes from 1 to quanti_all.

col_allele2 This keyword specifies the column containing the index of the allele with the larger allelic effect. This column is mandatory. The index of the allele goes from 1 to quanti_all.

col_dominance This keyword specifies the column containing the dominance effects. For any pair of alleles for which the dominance effect is not specified a dominance effect will be drawn from the normal distribution defined by the parameters quanti_dominance_mean and quanti_dominance_var.

col_fitness_factor This keyword specifies the column containing the fitness factors (see section 8.6 for more informations). For any pair of alleles for which the fitness factor is not specified explicitly the fitness factor will be zero or given by the parameters quanti_sel_fitness_factor or quanti_fitness_factor_homozygote if they are defined.

Note, that as in all input files for quantiNemo it is possible to add comments (also in the file information box) using the hash character: '#' or '#/ ...any text... /#'. If you define explicitly the dominance

effect and the fitness factor, but for a given pair of alleles you want only to define one of the parameters you may use the number *99999* as placeholders. quantiNemo will treat this number as not set. In the above example, this has been used to show that a dominance effect for a monomorphic locus is not valid (However a fitness factor may be set for a monomorphic locus). Please note that in this special case (monomorphic locus) one could have set any number as placeholders for the dominance effect since the value would never be used.

quanti_dominance_mean [decimal] (default: 0)
This parameter allows specifying the mean of the normal distribution from where the dominance effects are randomly drawn. Note that a_1 has the smaller allelic effect than a_2. This parameter is only taken into account if the dominance effects are not defined explicitly by the dominance file.

quanti_dominance_var [decimal] (default: 0)
This parameter allows specifying the variance of the normal distribution from where the dominance effects are randomly drawn. This parameter is taken into account only if the dominance effects are not defined explicitly by the dominance file.

8.2.3 Epistatic effects

It is possible to simulate epistatic effects between alleles at different loci. The following equation is used to compute the genotype:

$$G_{11'22'...kk'...} = \epsilon_{11'22'...kk'...} + \sum_{i=1}^{nbLoci} G_i$$

Where $G_{11'22'...kk'...}$ is the genotypic value of genotype $11'22'...kk'...$ (1 and $1'$ are the alleles of locus 1, 2 and $2'$ the alleles of locus 2, ...), G_i is the genotypic effect of locus i (additive and dominance effects), and $\epsilon_{11'22'...kk'...}$ is the epistatic effect of genotype $11'22'...kk'...$ (unique for each multilocus genotype). There are two possibilities to define epistatic effects. Either they are defined explicitly for each genotype using a separate file (see parameter

quanti_epistatic_file), or by defining the variance of the normal distribution from where the epistatic effects are randomly drawn (see parameter quanti_epistatic_var). Using the parameter quanti_epistatic_file it is also possible to define the genotypic effects directly. If the epistatic effects are defined in both ways the explicitly defined effects are used. Note, that all effects have to be defined in the same way, *i.e.* explicitly or by their distribution. By default, the parameter quanti_epistatic_var is set to zero resulting in simulations without any epistatic effects.

quanti_epistatic_file [string] (default: "")

This parameter allows to pass the name of a file containing the epistatic effects and/or the fitness factor (see section 8.6). The number of defined genotypes has to be in line with the parameters quanti_loci and quanti_all if epistatic or genotypic values are passed. If only fitness factors are defined it is enough to list the desired genotypes with their fitness factors. In this case all not specified genotypes have a fitness factor of 1. The epistatic file has a similar format as the allelic file:

```
#Epistatic file
######################
[FILE_INFO]{
  col_genotype 1
  col_epistatic_value 2
  # col_genotypic_value 3
  col_fitness_factor 4
}

#genotype        epistaticVal   genotypicVal   fitness
{0101 0101}      0.242984       1.87009        1
{0101 0102}      0.580787       0.834811       99999
...
{5050 5048}      0.264001       1.24981        0
{5050 5049}      0.118982       -0.55701       1
{5050 5050}      0.071359       -2.26644       1
```

The file has to start with a file information box [FILE_INFO]{...}. This box contains informations about the structure of the following table and thus allowing flexibility in the format of the table. For example the order of the columns in the table may vary, or some columns may be ignored. The file information box starts with the key word [FILE_INFO]. This key word is followed by brackets "{...}" within which the user

has to specify the contents of the columns to be considered by quantiNemo. Each column is specified by a pair consisting of a key word (e.g. col_locus followed by the column number (the ordering starts with 1). Each column definition has to be on a new line. The following column keywords are available:

col_genotype This keyword specifies the column containing the genotype. This column is mandatory. The genotype is enclosed by brackets "{...}" and the genotype itself has to be in the FSTAT format (Goudet, 1995). The two alleles of a locus are written consecutively without any space. For all alleles the same number of digits are needed. Two different loci are separated by a space. The above example consists of two loci, each with 50 alleles. Each allele is written using two digits.

col_epistatic_value This keyword specifies the column containing the epistatic effects. If this column is set the epistatic effect is added to the genotypic effect computed as described above.

col_genotypic_value This keyword specifies the column containing directly the genotypic effects. If this column is set, the genotypic effect of each genotype is set directly without taking into account allelic, dominance, and/or epistatic effects. If both keywords col_epistatic_value and col_genotypic_value are specified in the information box, only the column col_genotypic_value is considered.

col_fitness_factor This keyword specifies the column containing the fitness factors. For more informations on the fitness factor see please section 8.6. For each genotype the fitness factor is not specified explicitly the fitness factor will be taken either from dominance file (parameter quanti_dominance_file) or from the global parameters quanti_sel_fitness_factor and quanti_fitness_factor_homozygote. If a genotype is listed to define the genotypic or epistatic value, but you don't want to define explicitly the fitness factor for this genotype you may use the number *99999* as placeholder. quantiNemo will treat this number as not set. In the above example this has been used to show that a fitness factor for a given genotype is not set.

Note, that as in all input files for quantiNemo it is possible to add

comments (also in the file information box) using the hash character: '#' or '#/ ...any text... /#'.

quanti_epistatic_var [decimal] (default: 0)
This parameter allows specifying the variance of the normal distribution from where the epistatic effects are drawn randomly. The normal distribution is centered around 0. This parameter is only taken into account if the epistatic effects are not defined explicitly by the epistatic file.

8.3 Environmental effect

The phenotype is given by the sum of the genotypic value and the environmental contribution. There are several possibilities to set the contribution of the environment to the phenotype globally or for each patch separately. By default (without any specification of the following parameters) the environment has no effect on the phenotype of the quantitative trait. Either the contribution of the environment to the phenotype is defined directly by the variance of the environmental effect (model 0), by the narrow-sense heritability h^2 (model 1 and 2), or by the broad-sense heritability H^2 (model 3 and 4). The heritability is later translated into a corresponding environmental variance V_E:

$$h^2: \qquad V_E = \frac{1-h^2}{h^2} * V_A$$
$$H^2: \qquad V_E = \frac{1-H^2}{H^2} * V_G$$

Where V_A is the additive genetic variance computed following (Lynch and Walsh, 1998, p85-87), and V_G the genetic variance.

quanti_environmental_model [0-4] (default: 0)
This parameter specifies how the environmental variance is defined. The following models are available:

0 : set V_E directly. The variance of the environment is set directly by the parameter quanti_heritability, which is in this case not the heritability, but the environmental variance.

1 : V_E defined by the narrow-sense heritability (V_E constant). The variance of the environment (V_E) is set at the beginning of a simulation (generation 1) and is based on the narrow-sense heritability (h^2, parameter quanti_heritability) and the additive genetic variance (V_A at generation 1). Note, that in this case, the environmental variance remains constant over time, but not the heritability.

2 : V_E defined by the narrow-sense heritability (h^2 constant). This is the same as model 1, but the environmental variance is readjusted at each generation. Thus the narrow-sense heritability remains constant over time, but not the environmental variance.

3 : V_E defined by the broad-sense heritability (V_E constant). The variance of the environment (V_E) is set at the beginning of a simulation (generation 1) and is based on the broad-sense heritability (H^2, parameter quanti_heritability) and the genetic variance (V_G at generation 1). Note, that in this case, the environmental variance remains constant over time, but not the heritability.

4 : V_E defined by the broad-sense heritability (H^2 constant). This is the same as model 3, but the environmental variance is readjusted at each generation. Thus the broad-sense heritability remains constant over time, but not the environmental variance.

quanti_heritability [decimal/matrix] (default: 0)

This parameter has different meaning depending on the environmental model chosen (parameter quanti_environmental_model): If V_E is directly set (model 0) this parameter is the environmental variance.

For the environmental model 1 and 2 this parameter is the narrow-sense heritability (h^2):

$$h^2 = V_A/V_P$$

For the environmental model 3 and 4 this parameter is the broad-sense heritability (H^2):

$$H^2 = V_G/V_P$$

Note, that a heritability (narrow and broad sense) of 0 (no genetic component) makes no sense for models 1 to 4 in this simulation framework. Therefore a value of 0 is not accepted by quantiNemo for models 1 to 4,

resulting in an error message. If the parameter quanti_environmental_model is set to 0 the parameter quanti_heritability is no longer the heritability, but the environmental variance directly, and can be set to 0 (default). Using a matrix as argument it is possible to set the environmental variance, respectively heritability for each patch separately.

quanti_environmental_proportion [decimal] (default: 1)

This parameter specifies which environment affects the phenotype of the quantitative trait: the natal or the current (at the adult stage) environment? The argument specifies the relative weight of the current patch effect on the phenotype: if the value is 1 (default value) only the environmental variance of the current patch affects the phenotype while if the value is 0 only the environmental variance of the natal patch affects the phenotype.

quanti_va_model [0-2] (default: 0)

This parameter specifies how the additive genetic variance (V_A) of a quantitative trait is computed. The additive genetic variance of a trait is used in several statistics. For instance it is used to set the environmental variance of the patches if this variance is specified by the narrow-sense heritability ($h2$, parameter quanti_environmental_model set to 2 or 3, see also parameter quanti_heritability), the additive genetic variance may also be directly output as summary statistic (stat option $q.varA$), or indirectly in the population differentiation measurement Q_{ST} (stat option $q.qst$, $q.qst.f$, $q.qst_paair$, and $q.qst.f_paair$). The additive genetic variance (V_A) is computed following Lynch and Walsh (1998, p85-87). For a quantitative trait determined by a single locus this is:

$$V_A = 2 \sum_{i=1}^{nbAllele} p_i \alpha_i \alpha_i^*$$

Where p_i is the allele frequency of allele i, α_i is the additive effect of allele i, and α_i^* is the average excess of allele i. Note, that we show here the computation for a single locus for simplicity reasons. In quantiNemo, a full version for traits determined by multiple loci is implemented. However, the implementation does not take into account

linkage between loci, thus the additive genetic variance is underestimated when loci are linked.

0 : for any case. If this model is chosen the additive effect (α_i) and the average excess (α_i^*) are computed. While the average excess is simple to compute, the computation of the additive effect requires a time consuming least-square regression. This formula is valid for any mating system, but its computation is rather slow.

Note, that the least-square regression has not always a solution thus the additive effects α, are not always computable. In this case, the locus in question is skipped from the analysis and a warning is returned:

```
***WARNING*** Va could not be correctly estimated (1. time
 at generation 20, see manual parameter 'quanti_va_model')
```

This warning is returned the first 10 times the problem occurs and then every hundredth time. It is up to the user to decide whether the problem occurs too often or if these "missing points" are acceptable. If the environmental variance is set by the narrow-sense heritability (parameter quanti_environmental_model set to 1 or 2) and the additive genetic variance cannot be computed quantiNemo uses the additive genetic variance computed using the algorithm for random matings (this parameter set to 1). Note, that

1 : limited to random mating. In case of random mating, the additive effect (α_i) and the average excess (α_i^*) are identical. Therefore the time consuming computation of the additive effect can be omitted. Note, that even when the mating system is set to random mating (parameter mating_system set to 0 or 2) mating is not randomly if selection acts, as the choice of the parents depends on their fitnesses. It's up to the user to decide whether this quick way to compute the additive genetic variance is appropriate or not.

1 : $V_A = V_G$. This model assumes that the additive genetic variance V_A is identical or may be approximated by the genetic variance V_G.

8.4 Selection pressure

Each quantitative trait is under a specific selection pressure. The type of selection may vary among quantitative traits, but not among patches and sexes. However, the strength, direction, etc, of selection can change among patches, sex and time. This flexibility allows for example to simulate a dynamic environment such as for global warming or variation in the altitude between patches. To specify the selection pressure individually for quantitative traits and patches, matrices may be used.

8.4.1 Selection models

Selection pressures have to be specified either for each sex separately (parameters with the suffix "_fem" for females and "_mal" for males), or for both sexes together (parameters without a suffix). In the first case both sex specific parameters have to be set if two sexes are simulated. In the latter case, the selection pressure of females and males are assumed to be identical.

quanti_selection_model [0-4] (default: 0)

This parameter allows defining the selection model for each quantitative trait. By default no selection acts on a quantitative trait.

0 : neutral quantitative trait.

The phenotype of this trait does not feel any selective pressure. Selection might act only ifs fitness factor is set, otherwise, it behaves as a neutral trait :

$$W_t = 1$$

1 : stabilizing selection

Stabilizing selection acts on the quantitative trait. The fitness W of quantitative trait t is computed using the following standard Gaussian function for stabilizing selection:

$$W_t = e^{\left(-\frac{\left(P-Z_{Opt}\right)^2}{2\omega^2}\right)}$$

Z_{Opt} is the selection optimum of the current trait and patch, P is the phenotypic value of the individual, and ω is the intensity of the selection. See subsection 8.4.2 for more details.

2 : directional selection.
Directional selection acts on the quantitative trait. A generalized logistic curve (Richards, 1959) is implemented in quantiNemo to characterize the directional selection pressure:

$$W_t = \left(1 + s * e^{r(P_{r_{Max}} - P)}\right)^{-1/s}$$

Where r is the growth rate, $P_{r_{Max}}$ is the phenotype with the maximal slope, and s defines the symmetry of the slope. See subsection 8.4.3 for more details.

3 : fitness landscape.
The selection pressure is defined by a fitness landscape. The fitness value of any phenotype may be specified. The fitness value of a phenotype within the specified range of phenotypes is linearly interpolated, Phenotype outside the range of specified phenotypes will result in the fitness value of the smallest phenotype, respectively in the fitness value of the largest phenotype. See subsection 8.4.4 for more details.

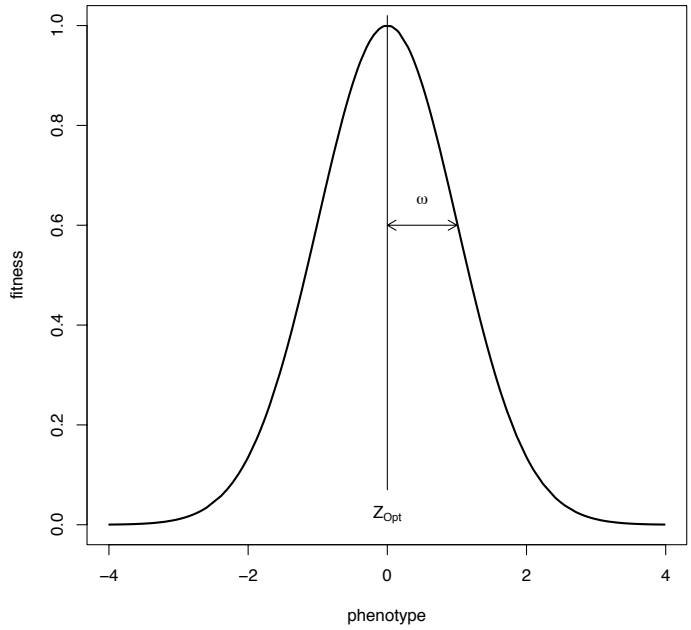
4 : selection coefficient.
The selection pressure of a bi-allelic locus is defined by a selection coefficient s and a dominance factor h. In this simplified, but widespread model, the genotype is directly mapped to a fitness. Note that for this type of selection a trait may only contain a single bi-allelic locus, without explicitly specified allelic effects. See subsection 8.4.5 for more details.

Genotype	Fitness	Fitness extended
AA	1	F
Aa	1 - hs	F - hs
aa	1 - s	F - s

8.4.2 Stabilizing selection

Stabilizing selection act on the phenotype (P) of quantitative traits. The selection pressure is defined by the parameters quanti_stab_sel_optima (Z_{Opt}) and quanti_stab_sel_intensity (ω). The fitness (W) of a quantitative trait is computed using the following standard Gaussian function for stabilizing selection:

$$W = e^{\left(-\frac{\left(P-Z_{Opt}\right)^2}{2\omega^2}\right)}$$



quanti_stab_sel_optima
quanti_stab_sel_optima_fem
quanti_stab_sel_optima_mal [decimal/matrix] (temporal/default: 0)

These parameters allow to set the selection optimum z_{Opt} for each patch for a given quantitative trait.

quanti_stab_sel_intensity
quanti_stab_sel_intensity_fem
quanti_stab_sel_intensity_mal [decimal/matrix] (temporal/default: 1)

These parameters allow to set the selection intensity ω for each patch for a given quantitative trait. Contraintuitively a small value results in a strong selection pressure, whereas a large value results in a weak selection pressure.

patch_stab_sel_optima_var [decimal/matrix] (temporal/default: 0)

This parameter specifies the variance of the normal distribution by which the selection optimum varies between generations (e.g. annual fluctuations of the mean temperature). By default the local selection optimum does not vary.

patch_stab_sel_intensity_var [decimal/matrix] (temporal/default: 0)

This parameter specifies the variance of the normal distribution by which the selection intensity varies at each generation (e.g. annual fluctuations of the mean temperature). By default the local selection intensity does not vary.

Example

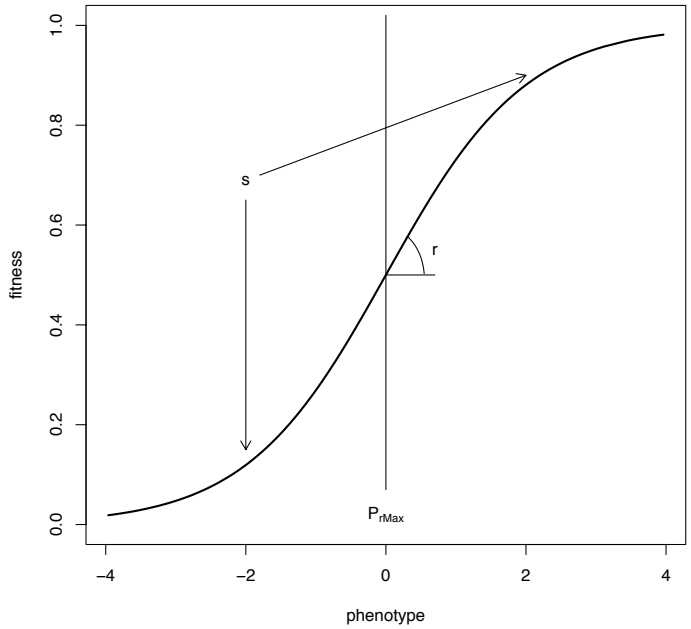
```
patch_number              2
quanti_nb_trait           3
quanti_stab_sel_optima_1 {-0.1   0.1}
quanti_stab_sel_optima_2 { 0.2   0.2}
quanti_stab_sel_optima_3 {-0.3   0.3}
quanti_stab_sel_intensity 1
```

In this example the environment consist of two patches with varying selection pressures. Three quantitative traits are simulated. The first trait has a selection optimum at -0.1 in patch 1 and at 0.1 in patch 2. The selection optimum of the second trait is the same in both patches (0.2). The third trait has an optimum at -0.3 in patch 1 and at 0.3 in patch 2. The intensity of the selection is identical for all three traits and in both patches.

8.4.3 Directional selection

Directional selection may act on quantitative traits. The fitness (W) of a quantitative trait is computed using the following generalized logistic function (Richards, 1959):

$$W = min + \frac{max - min}{(1 + s * e^{r(P_{r_{Max}} - P)})^{1/s}}$$



Where min is the lower asymptote (parameter quanti_dir_sel_min), max is the upper asymptote (parameter quanti_dir_sel_max), r is the growth rate (parameter quanti_dir_sel_growth_rate), $P_{r_{Max}}$ is the phenotype with the maximal slope (parameter quanti_dir_sel_max_growth), and s defines the symmetry of the curve (parameter quanti_dir_sel_symmetry; the curve is symmetric by default (value 1)).

quanti_dir_sel_min
quanti_dir_sel_min_fem

quanti_dir_sel_min_mal [decimal/matrix] (temporal/default: 0)
These parameters allow to set the lower asymptote of the selection curve for each patch for a given quantitative trait.

quanti_dir_sel_max
quanti_dir_sel_max_fem
quanti_dir_sel_max_mal [decimal/matrix] (temporal/default: 1)
These parameters allow to set the upper asymptote of the selection curve for each patch for a given quantitative trait.

quanti_dir_sel_growth_rate
quanti_dir_sel_growth_rate_fem
quanti_dir_sel_growth_rate_mal [decimal/matrix] (temporal/default: 1)
These parameters allow to set the slope of the selection curve for each patch for a given quantitative trait. If the argument is positive larger phenotypes have a higher fitness, while if negative smaller phenotypes have a higher fitness.

quanti_dir_sel_max_growth
quanti_dir_sel_max_growth_fem
quanti_dir_sel_max_growth_mal [decimal/matrix] (temporal/default: 0)
These parameters allow to set the phenotype with the maximal growth.

quanti_dir_sel_symmetry
quanti_dir_sel_symmetry_fem
quanti_dir_sel_symmetry_mal [decimal/matrix] (temporal/default: 1)
These parameters allow to set the symmetry of the curve. The default value of 1 results in a symmetric slope.

patch_dir_sel_min_var [decimal/matrix] (temporal/default: 0)
This parameter specifies the variance of the normal distribution by which the lower asymptote of the selection curve varies at each generation (e.g. annual fluctuations of the mean temperature). By default the lower asymptote of the selection curve does not vary.

patch_dir_sel_max_var [decimal/matrix] (temporal/default: 0)
This parameter specifies the variance of the normal distribution by which the upper asymptote of the selection curve varies at each generation (e.g. annual fluctuations of the mean temperature). By default the upper asymptote of the selection curve does not vary.

patch_dir_sel_growth_rate_var [decimal/matrix] (temporal/default: 0)
This parameter specifies the variance of the normal distribution by which the selection slope varies at each generation (e.g. annual fluctuations of the mean temperature). By default the local selection slope does not vary.

patch_dir_sel_max_growth_var [decimal/matrix] (temporal/default: 0)
This parameter specifies the variance of the normal distribution by which the phenotype with maximal growth varies at each generation (e.g. annual fluctuations of the mean temperature). By default the local phenotype with maximal growth does not vary.

patch_dir_sel_symmetry_var [decimal/matrix] (temporal/default: 0)
This parameter specifies the variance of the normal distribution by which the symmetry of the curve varies at each generation (e.g. annual fluctuations of the mean temperature). By default the symmetry of the slope does not vary.

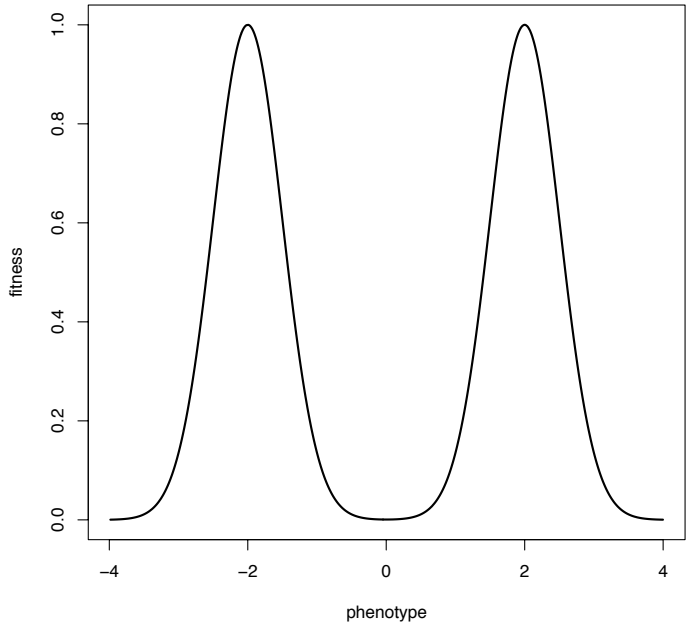
Example

```
quanti_dir_sel_growth_rate 1
quanti_dir_sel_max_growth  0
quanti_dir_sel_symmetry    1
```

In this example the selection pressure for all patches and quantitative traits are identical and set to the default values. The specified directional selection pressure favours larger phenotypes (parameter quanti_dir_sel_growth_rate is positive). This means that individuals with larger phenotypes have on average higher fitnesses and thus higher reproductive successes.

8.4.4 Fitness landscape

Specifying a fitness landscape allows defining any selection pressure for a given quantitative trait. More precisely, it allows to associate to any value of the trait (*i.e.* phenotypic value) a given fitness. The phenotypic values (parameter quanti_phenotype_landscape) and fitness values (parameter quanti_fitness_landscape) are separately passed to quantiNemo as two vectors. The fitness value of any phenotype within the specified range of phenotypes is linearly interpolated and if the phenotype lies outside of the specified range of phenotypes the resulting fitness value will result in the fitness value of the smallest and largest phenotype, respectively. By default, all phenotypes will result in a fitness of 1.



The fitness landscape is defined by two tightly linked parameters:

quanti_phenotype_landscape
quanti_phenotype_landscape_fem

quanti_phenotype_landscape_mal [decimal/matrix] (temporal/default: 0)

These parameters allow to specify an array of phenotypes for which the corresponding fitness is defined using the parameter quanti_fitness_landscape. The phenotype values are either specified by a single value (similar to the default value) specifying a unique phenotype for all patches and quantitative traits together (all phenotypes will result in the same fitness), or by a one dimensional array (1D matrix) specifying the same fitness landscape for all patches and a given quantitative traits together, or by a 2D matrix specifying the fitness landscape separately for each patch for a given quantitative trait. Since the two parameters quanti_fitness_phenotype and quanti_fitness_fitness are tightly linked, *i.e.* specify the phenotypes and their corresponding fitness values, the architecture of the two parameters have to be identical, *i.e.* show up the same number of values for a patch.

quanti_fitness_landscape
quanti_fitness_landscape_fem
quanti_fitness_landscape_mal [decimal/matrix] (temporal/default: 1)

These parameters allow to specify the fitnesses corresponding to the phenotypes defined with the parameter quanti_phenotype_landscape, in the same order as the phenotypes are defined in the parameter quanti_fitness_phenotype.

Example

```
quanti_phenotype_landscape {-4.0 -3.2 -2.4 -1.6 -0.8  0.0
                             0.8  1.6  2.4  3.2  4.0}
quanti_fitness_landscape   {0.00 0.05 0.72 0.72 0.05 0.00
                            0.05 0.72 0.72 0.05 0.00}
```

In this example the selection pressure for all patches and quantitative traits are identical. The specified fitness landscape corresponds to the above shown figure, however, the resolution of the specification is reduced in the shown example. The binomial landscape is the sum of two stabilizing selections with optima set to -2 and 2 and an intensity of 0.5.

8.4.5 Selection coefficient

QuantiNemo supports the simulation of bi-allelic loci where the selection pressure is defined by a selection coefficient. In this simplified, but widespread model, the genotype is directly mapped to a fitness:

Genotype	Fitness	Fitness extended
AA	1	F
Aa	1 - hs	F - hs
aa	1 - s	F - s

Where s is the selection coefficient (parameter quanti_coef_sel), h the dominance factor (parameter quanti_dominance_mean) with the model (parameter quanti_dominance_model) set to 1 (h-model, has to be set explicitly!), and F is the fitness factor for the wild type genotype AA set by parameter quanti_coef_sel_AA. This latter parameter is normally not used therefore it is listed in the column "Fitness extended".

quanti_coef_sel
quanti_coef_sel_fem
quanti_coef_sel_mal [decimal/matrix] (temporal/default: 0)
: These parameters allow to set the selection coefficient s for each patch and quantitative trait/locus. The default value of 0 will result in equivalent fitness for both alleles, and thus no selection acts.

quanti_coef_sel_AA
quanti_coef_sel_AA_fem
quanti_coef_sel_AA_mal [decimal/matrix] (temporal/default: 1)
: These parameters allow to set the fitness for the wild type genotype AA for each patch and quantitative trait/locus. By default it is 1.

8.5 Selection level and position

8.5.1 Selection level

With the following parameters it is possible to define the level of selection.

selection_level [0-3] (default: 0)
This parameter specifies how selection acts at the reproduction stage.

0 : soft selection. $nbOff_p = N_p$
The fitness of an individual is relative to the mean fitness of the population of its patch (soft selection at the patch level). This means that the population size does not depend on the mean fitness of the population and that selection does act locally at the patch level, *i.e.* patches do not interact (Wallace, 1975). No populations may therefore go extinct due to their maladaptation.

1 : metapopulation selection. $nbOff_p = (W_p/W_m)N_m$
The fitness of an individual is relative to the mean fitness of the entire metapopulation (soft selection at the metapopulation level). This means that the size of the entire metapopulation does not depend on the mean fitness. However, the population size of each patch depends on the mean fitness of its population (Ravigne et al., 2004). This implies that a well adapted population grows while the population size of a less adapted population declines. Individual populations may go extinct due to their maladaptation, however not the entire metapopulation.

2 : hard selection . $nbOff_p = \bar{W}_p * N_p$
The fitness of an individual is absolute. If selection acts at reproduction the number of offspring to produce is scaled by the mean fitness of the population (the maximum number of offspring to produce (with a mean fitness of 1) is defined by the parameter mating_nb_offspring). If selection acts during regulation then the fitness of an individual is identical to its survival probability. Only with hard selection, the entire metapopulation may go extinct if the populations are maladapted.

The shown equation are correct for selection during reproduction and deviate slightly if selection acts at another stage. $nbOff_p$ is the number of offspring to be produced in patch p, N_p and N_m are the total number of offspring defined by the parameter mating_nb_offspring_model of patch p and metapopulation m, respectively, W_p and W_m are the sum of fitnesses of patch p and metapopulation m, respectively, and $\bar{W}_p$ is the mean fitness of patch p.

Here is an example of the different selection levels in case of selection at the reproductive success. Both populations have a carrying capacity (K) of 1000 individuals, the parameter mating_nb_model is set to 0, and the mean fitnesses of the populations ($\bar{W}$) are 0.6 and 0.2, respectively:

selection level	population 1 $K = 1000$ $\bar{W} = 0.6$	population 2 $K = 1000$ $\bar{W} = 0.2$	total
soft	1000	1000	2000
metapopulation	1500	500	2000
hard	600	200	800

patch_mean_fitness [0-2] (default: 0)
This parameter specifies how the mean fitness of a patch is defined. By default, all individuals are considered.

0 : all. The fitness of all individuals (females and males) is considered (default).

1 : only females Just the fitness of the females is considered.

2 : only males Just the fitness of the males is considered.

8.5.2 Selection position

By default, the selection appends during breeding. This is a reasonable choice for performance issue (all individual created go until the next reproduction stage, *i.e.* no "useless" individuals are created). However, quantiNemo also offer the possibility for selection to act at other stages of the life cycle. Note, however, that selection may only act at a single stage at once.

selection_position [0-4] (default: 0)
The parameter specifies when and how selection acts. By default, selection acts at reproduction where the fecundity of an individual depends on its fitness.

0 : reproductive success. Selection acts at the reproduction stage (see section 5.2), where the number of offspring of an individual depends on its fitness.

1 : reproductive success special. Currently not working (do not use).

2 : offspring survival. Selection acts at the survival probability of the offspring, *i.e.* before dispersal (see section 5.4).

3 : adult survival. Selection acts at the survival probability of the adults, *i.e.* after dispersal (see section 5.5).

4 : no selection. This option allows performing simulations without any selection.

8.6 Fitness factor

The fitness factor allows translating a genotype directly into a fitness value. A fitness factor may be set either at the quantitative trait genotype level, at the locus level or globally for homozygote and/or heterozygote loci. The resulting fitness factor is determined in the above mentioned order, first quantitative trait genotype specific settings are considered, followed by locus specific settings and finally global parameters. The default fitness factor is 1 for all cases. This feature of quantiNemo allows among others to simulate recessive deleterious alleles, heterozygote or homozygote deficits, or incompatibilities of alleles for example to simulate hybridization or speciation.

8.6.1 Fitness factor at locus level

The fitness factors at the locus level are defined in the dominance file (see section 8.2.2)

8.6.2 Fitness factor at trait level

The fitness factors at the trait level are defined in the epistatic file (see section 8.2.3)

8.6.3 Fitness factor at global level

If the fitness factors are defined globally, the contribution to the fitness will be computed for each locus, and the final fitness will be obtained by mul-

tiplying the local fitnesses To define the fitness factor globally the following parameters may be used:

quanti_fitness_factor_heterozygote [decimal/matrix] (default: 1)
This parameter allows to specify the fitness factor for heterozygote loci, *i.e.* if the two alleles at a given locus are different. Using a matrix it is possible to define different fitness factors for each locus. By default the fitness factor for heterozygote loci is 1, resulting in no effect on the fitness.

quanti_fitness_factor_homozygote [decimal/matrix] (default: 1)
This parameter allows to specify the fitness factor for homozygote loci, *i.e.* if the two alleles at a given locus are identical. Using a matrix it is possible to define different fitness factors for each locus. By default the fitness factor for homozygote loci is 1, resulting in no effect on the fitness.

Chapter 9

Coalescence

9.1 Introduction

QuantiNemo has been developed to perform forward in time simulations at the individual level. This allows simulating realistically highly complex quantitative traits under selection. The drawback of this type of simulation is the huge demand of memory and long computation times. In contrast, population-based backward in time simulations based on the coalescence theory are much more efficient in terms of computation time and memory usage but do not allow to simulate realistic quantitative traits under selection. In order to take advantage of both types of simulations, we have added a layer of coalescence to quantiNemo2. Having both individual and population-based simulations allow for the user to switch easily the simulation mode depending on the type of simulation, keeping all the rest of the simulation/demography identical.

This allows, for example, to explore a demographic history using efficient population-based simulations and then to make the simulations more realistic by switching to individual based simulations. Technically the population-based simulations simulate forward in time the evolution of the entire populations. During this simulation, the population sizes and immigration rates for all patches are stored in an internal database. In a second step, the database is used to simulate backward in time corresponding coalescence trees on which in a third step the mutations are sprinkled to generate the

genetic polymorphism. Most of the parameters of quantiNemo may be used in both simulation modes, but not all. Thus it is not possible to simulate any quantitative traits and thus no selection acts in population-based simulations. It is also important to note that population sizes in coalescence simulations are effective population sizes, whereas in individual-based simulation census population sizes. In addition at the current stage, the coalescence simulations are limited to hermaphrodite individuals and unlinked neutral markers. The genetic maps is not taken into account in the population-based mode. Finally, statistics are only available for adults, since only the last generation of individuals is created.

In this chapter, the coalescence specific parameters are described.

coalescence [0-1] (default: 0)

This parameter allows to switch between forward-in-time simulation of individuals and backward-in-time simulations of populations (coalescence):

0 : individual based. Forward-in-time simulation of individuals. That is the default mode where quantitative traits and selection may be simulated.

1 : population based. Backward-in-time simulations of populations using a coalescence approach. Only able to generate unlinked neutral markers.

coalescence_model_threshold [decimal] (default: 0.1)

The probability p that a single coalescence event per generation and patch occurs depends on the number of traced lineages n and the population size N (in diploid numbers):

$$p = \frac{n(n-1)}{4N}.$$

This approximation is valid for small numbers of traced lineages compared to the population size ($n << N$). If n in relation to N gets bigger the probability that more than one coalescence event occurs increases and thus the method above is not any more accurate. Therefore quantiNemo allows setting a threshold (this parameter) for the probability of a single coalescence even p above which the method used to simulate

the coalescence events is changed. In this latter case the coalescence process is simulated realistically, *i.e.* for each traced lineage a parental gene is drawn randomly. All lineages with an identical parental gene are then coalesced. In this case, at each generation, several coalescence events may happen but also events involving the fusion of three or more lineages are possible.

This method is always valid, however much slower in terms of computation time than the previously mentioned one (Note however that this is still much faster than forward in time simulation). By default the value of this parameter, *i.e.* the threshold, is set to 0.1 which is an acceptable threshold. Note that the probability that a single coalescence event occurs may exceed extensively 1. A value of 0 implies that the coalescent process is always realistically simulated. And ratios above 0 are used to specify the threshold below which a singe coalescence event is assumed and above which the coalescence process is simulated realistically. A parameter value of 1e6 is arbitrarily used to tell quantiNemo that only single coalescence events should be assumed.

divergence_time [integer] (default: 0)

A coalescence simulation lasts until the most recent common ancestor (MRCA) is reached. Depending on the simulated demography the MRCA may be found within the simulated time, but it can well be that the MRCA of all sampled lineages lays before the start of the demographic simulation. In this case, going backward in time the coalescence simulation is continued behind the demographic simulation assuming no migration between the remaining patches. At the divergence time (parameter **divergence_time** all remaining lineages are then merged to a single patch of specified size (parameter **divergence_pop_size** below). By default the value of the parameter **divergence_time** is 0. If the divergence time is set to less than the number of simulated generations (as the default) the divergence time will be readjusted to the number of generations, thus all remaining lineages are merged to a single patch when the onset of the demographic simulation is reached.

divergence_pop_size [integer] (default: 0)

This parameter allows to define the final population size after the divergence time. Check the parameter **divergence_time** for a description of it. If the argument is 0 (default value) the population size is adjusted

to the current total number of individuals. If the specified population size is smaller than the number of traced lineages, then the population size is increased to the smallest possible population size still capable to carry the traced lineages.

9.2 Output

The coalescence simulations allow to output the coalescence trees and also the times to the MRCA.

9.2.1 Trees

The coalescence trees may be dumped to file in the NEXUS format. The branches of the tree may be either scaled by the coalescence times, or by the number of mutations. Several programs allowing to visualize NEXUS files such as for example FigTree developed by Andrew Rambaut www.tree.bio.ed.ac.uk/software/figtree. The NEXUS file has the extension **.tree**.

coalescence_save_tree [0-2] (default: 0)
This parameter specifies the output of the coalescence trees.

0 : None. No output is generated.

1 : scaled by coalescence time. The trees are outputted in the NEXUS format and branches are scaled by the coalescence times.

2 : scaled by number of mutations. The topology of the trees is the same as before, but the branches are now scaled by the number of mutations.

An example of a tree file for 3 loci and 3 sampled diploid individuals:

```
#NEXUS

[Treefile generated by quantiNemo
    quantiNemo v1.5.0[May 20 2010; 09:14:19]
    File created the 20-05-2010 09:14:27
    Branches are scaled by the coalescence time
]
```

```
Begin trees;

Translate
    1 1a,
    2 1b,
    3 2a,
    4 2b,
    5 3a,
    6 3b;

tree LOCUS_1 = ((5: 114, 4: 114): 2, ((1: 59, 6: 59): 2, (3: 13, 2: 13): 48)
tree LOCUS_2 = (1: 72, ((6: 33, 5: 33): 22, ((2: 3, 4: 3): 2, 3: 5): 50): 17
tree LOCUS_3 = (((1: 9, 2: 9): 13, 6: 22): 209, ((4: 0, 5: 0): 180, 3: 180):

End;
```

coalescence_tree_dir [string] (default: "")

This parameter allows to specify the subdirectory where the tree files are stored. This directory has to be specified relative to the simulation folder (parameter folder), and may also contain subdirectories. If not specified (default) the output is stored in the simulation folder (parameter folder).

coalescence_tree_filename [string] (default: "")

This parameter is used to specify an individual base filename for the coalescence trees. If not specified the generic base filename will be used (see parameter filename).

coalescence_tree_script [string] (default: "")

It is possible to launch a script just after the tree file is generated. The argument of the parameter is the file name of the script. The name of the tree file is passed as unique parameter to the script.

9.2.2 MRCA

The times to the MRCA may be dumped to file as a simple list. The file has the extension **.mrca**.

coalescence_save_mrca [0-1] (default: 0)

This parameter specifies whether the times to the MRCA are outputted.

0 : None. No output is generated.

1 : Output. The times to the MRCA are outputted. mutations.

An example of such a file for 3 loci is:

```
112
339
813
```

coalescence_mrca_dir [string] (default: "")
This parameter allows to specify the subdirectory where the MRCA files are stored. This directory has to be specified relative to the simulation folder (parameter folder), and may also contain subdirectories. If not specified (default) the output is stored in the simulation folder (parameter folder).

coalescence_mrca_filename [string] (default: "")
This parameter is used to specify an individual base filename for the coalescence MRCA. If not specified the generic base filename will be used (see parameter filename).

coalescence_mrca_script [string] (default: "")
It is possible to launch a script just after the MRCA file is generated. The argument of the parameter is the file name of the script. The name of the MRCA file is passed as unique parameter to the script.

9.2.3 Lineages

For the coalescence simulation, it is possible to dump the current populated patches, including their population sizes and number of traces lineages to a file for any given generation. The file is generated for each locus separately and has the ending **_lX.lin**, where X is the locus index. The format is as follows: Each line contains a single generation, *i.e.* time slice. The line starts with the generation index relative to the start of the simulation. Thus since the coalescence simulations are proceeded backward in time the generations decrease from top to down of the file. Positive numbers are time slices where

a demographic simulation is available and negative numbers for time slices before (forward in time) the demographic simulation. The generation number is followed by the total number of currently traced lineages. Then each populated patch is listed using three numbers, the patch index, the current population size, and the current number of traced lineages. Since over time some patches may be colonized or freed the number of columns per line may change between generations.

coalescence_save_lineages [0-1] (default: 0)
This parameter specifies whether the patch stages are outputted.

0 : None. No output is generated.

1 : Output. The patch stages are outputted.

coalescence_lineages_logtime [integer] (temporal/default: 1)
This parameter allows to specify at which generations DURING the period of the demographic simulation the patch stages are dumped to file.

coalescence_lineages_logtime2 [integer] (temporal/default: 1)
This parameter allows to specify at which generations BEFORE (forward in time) the demographic simulation the patch stages are dumped to file.

coalescence_save_lineages_dir [string] (default: "")
This parameter allows to specify the subdirectory where the lineages files are stored. This directory has to be specified relative to the simulation folder (parameter folder), and may also contain subdirectories. If not specified (default) the output is stored in the simulation folder (parameter folder).

coalescence_save_lineages_filename [string] (default: "")
This parameter is used to specify an individual base filename for the lineages files. If not specified the generic base filename will be used (see parameter filename).

coalescence_lineages_script [string] (default: "")
It is possible to launch a script just after the patch stage file is generated. The argument of the parameter is the file name of the script.

The name of the patch stage file is passed as unique parameter to the script.

9.2.4 Population sizes

For the coalescence simulation, it is possible to dump the current populated sizes to file. The format of the file is the generation time followed by the population sizes for the patches of choice.

coalescence_save_pop_sizes [0-1] (default: 0)
This parameter specifies whether the population sizes are outputted.

0 : None. No output is generated.

1 : Output. The population sizes are outputted.

coalescence_pop_sizes_logtime [integer] (temporal/default: 1)
This parameter allows to specify at which generations the population sizes are dumped to file.

coalescence_save_pop_sizes_dir [string] (default: "")
This parameter allows to specify the subdirectory where the population sizes files are stored. This directory has to be specified relative to the simulation folder (parameter folder), and may also contain subdirectories. If not specified (default) the output is stored in the simulation folder (parameter folder).

coalescence_save_pop_sizes_filename [string] (default: "")
This parameter is used to specify an individual base filename for the population sizes files. If not specified the generic base filename will be used (see parameter filename).

coalescence_pop_sizes_of_patch [integer/matrix] (default: "")
This parameter allows defining the patches for which the population sizes should be outputted using a matrix with the patch IDs. By default the population sizes of all patches is outputted.

coalescence_pop_sizes_script [string] (default: "")
It is possible to launch a script just after the population sizes file is

generated. The argument of the parameter is the file name of the script. The name of the population size file is passed as unique parameter to the script.

9.3 Summary statistics

The summary statistics listed in the table below are available for coalescence simulations. The column Stat name contains the name of the summary statistic used to specify which summary statistics are computed (parameter stat, for details see section 10.3). These names appear also in the output file. The column Description contains a short description of the summary statistic.

The summary statistic name (column Stat name) may be used to specify the summary statistic to be computed (e.g. mrca.mean). Similar summary statistics (within a thematic group) may be obtained at once using the name within square brackets after the group title (e.g. mrca). Using this group statistic name all summary statistics of the thematic group marked with a star (*) will be computed.

sample_all_or_nothing [0,1] (default: 0)

This parameter allows to specify when statistics should be computed (only available for the coalescence simulations):

0 : when possible. In this case statistics and outputs are generated whenever it is possible, *i.e.* whenever a patch is enough populated that a given statistic may be computed.

1 : all or nothing. In this case statistics and output are generated only if the entire sampling schema may be applied, *i.e.* if the patches are enough populated that the specified sampling may be applied. If this is not the case a NaN or a zero depending on the statistic is outputted for all statistics. Note that in such a case the genetic part of the coalescence simulations are omitted, thus the simulation may be much quicker although useless. A specified sampling may not be applied if (and only if) not all specified patches (parameter sampled_patches defined as a matrix) may be

sampled or the population size is inferior to the sampling size (parameter patch_sample_size defined in entire numbers (absolute)).

Table 9.1: Summary statistics available for coalescence simulations

Stat name	Description
MRCA [mrca]	
mrca.mean	mean time to the MRCA*
mrca.var	variance of the time to the MRCA*

Table 9.1: Summary statistics available for coalescence simulations continued

Chapter 10

Ouputs and Statistics

QuantiNemo allows to output summary statistics as well as raw data about the population, including the complete genome, in order to offer the highest flexibility. QuantiNemo can also output log files to resume a simulation or restart it from the beginning with the same random parameter. More precisely, quantiNemo can output:

summary statistics
: QuantiNemo provides summary statistics for the different simulation components, including for example genetic variance estimates, quantitative trait analysis (e.g. Q_{ST}), and F-statistics. The summary statistics can be computed for any generation during the simulation. If several replicates are performed, the averaged across replicates as well as the value for each replicate can be outputted.

raw data
: QuantiNemo can also produce files with the raw genetic and phenotypic data. The genotypes at all loci can be dumped to file in the FSTAT (Goudet, 1995) or Arlequin format (Excoffier, 2010). Phenotypes, as well as the additive, dominance, and epistatic effect values can be written to a file and then analyzed with any population or quantitative genetic software, e.g. to get patterns of differentiation, study linkage disequilibrium, or scan for QTL. Genotypes and phenotypes can be saved for any generation during the simulation.

log files

QuantiNemo also generates log files allowing to reconstruct performed simulations. There are two types of log files. The first log file records the simulations performed with quantiNemo and stores some general information. This log file is stored in the folder of the executable and allows to reconstruct the chronology of performed simulations and their main features. The other log file contains the used parameters, is generated for each simulation separately and is stored in the simulation folder. This file is in principle a copy of the used settings file and contains the starting time and the duration time of the simulation. It contains also the seed (see parameter seed) used to initialize the simulation. This file can be used as settings file to exactly repeat the performed simulation. Note, that due to the seed in the file the random generator will be initialized in the same way leading to the exact same values in the output.

10.1 Files name

All files of a simulation are stored in a unique folder (see parameter folder). This simulation folder may contain a substructure. The names of the output files are based on the base name given by the parameter filename. Depending on the type of output different extensions are added to the base name. To avoid that recurring outputs overwrite previous outputs a counter is added to the file name between base name and extension. There are two types of counters: the generation counter and the replication counter. A counter is only added if there is a risk of overwriting. For example, the replication counter is only added if several replications are performed. The generation counter starts with "_g" and the replication counter with "_r". These characters are followed by the number of the generation and replication, respectively. Note, that generations and replications start at 1. The number has as many digits as are needed to represent the highest number in the simulation:

```
simulation_g0001_r01.dat
simulation_g0002_r01.dat
...
simulation_g5000_r10.dat
```

10.2 Sampling

By default, the outputs generated by quantiNemo (summary statistics, genotypes, phenotypes, and genotypic values) are computed/outputted considering all individuals and populations. This section describes parameters allowing to constrain the considered populations and/or individuals. The parameter **sampled_patches** allows to make a pre-selection on the patches, whereas the parameter **patch_sample_size** allows defining for each patch and/or sex the sampled number of individuals. It is therefore possible to specify the samples just using the parameter **patch_sample_size**.

sampled_patches [integer/matrix] (default: 0)

The parameter **sampled_patches** allows defining a sampling schema for patches, where ALL individuals are sampled. There are different methods to define the sampled patches:

single number. If the argument is a single number the patches to sample are drawn randomly. In this case, the passed number specifies the total number of patches to sample. Patches are drawn randomly for each replicate but remain the same during a simulation. A special case is 0 as argument (default value). In this case, all patches are sampled.

matrix. Using a one dimensional matrix as argument allows to define explicitly the patches to sample. Please note that for this parameter a matrix has to be defined fully, *i.e.* all patches to sample have to be listed in the matrix and a single number is not expanded to a matrix. Note, that in contrast to the normal matrix behaviour the two arguments *20* and *{20}* are not identical. While the former one is considered as a single number defining the total number of sampled patches, the second one is a matrix and defines the patch to be sampled, *i.e.* patch 20.

patch_sample_size
patch_sample_size_fem
patch_sample_size_mal [decimal/matrix] (temporal/default: NaN)

These parameters allows to define for each patch and/or sex the sample size or sample proportion. Since the sampling schema may change over

time the summary statistics are computed and outputted for all populations sampled at any time during the simulation even if at a given time the sample size is zero. These parameters here overwrite any settings specified by the parameter **sampled_patches**.If the sampling is sex specific both sex specific parameters have to be set. The sampling for a single patch may be defined in the following ways:

proportion. A decimal number (number between 0 and 1, exclusive 1) may be used to define the sample size relatively to the population size.

absolute. The sample size may be defined in an absolute number (1 or larger). If the absolute sample size exceeds the population size the entire population is sampled.

NaN. This is the default argument and allows to sample all individuals of a population.

The sampling schema for all patches may be defined as follows:

single number. All patches have the same sampling schema, *i.e.* sampling proportion or sampling number.

1D matrix. The sample sizes/proportions of each patch may be set using a 1D matrix. The setting of this matrix behaves as the matrix for the parameter **patch_capacity** or **patch_ini_size**, *i.e.* the matrix size is adjusted if needed. Within a matrix, the individual patch sample sizes may vary between relative, absolute, or *NaN* definitions.

2D matrix. A 2D matrix allows specifying directly the sample sizes/proportions for a given number of patches. The number of columns has to be 2, where the first column contains the patch ID and the second column the corresponding sampling number/proportion. Within a matrix, the individual patch sample sizes may vary between relative, absolute, or *NaN* definitions. All not specified patches will not be sampled, *i.e.* have a sample size of 0.

10.3 Summary statistics

It is possible to record summary statistics specified by the parameter stat. At the end of a simulation, the summary statistics are written to a text file. By default, the summary statistics are printed individually per replicate and/or summed up across replicates (mean and variance across replicates). In this latter case an additional statistic named *alive.rpl* will be added which contains the number of alive replicates, *i.e.* the number of simulations where the populations did not get extinct. It is also possible to set the frequency (parameter stat_log_time) of the recording, which reduces the simulation time and the size of the output file.

Some of the summary statistics are available for adults and offspring (indicated by (adlt/off)). To obtain a certain summary statistic for adults the prefix adlt. has to be added to the summary statistic name (e.g. adlt.allnb), respectively the prefix off. to obtain the summary statistic for offspring (e.g. off.allnb).

Relevant statistics can be computed either for the entire Metapopulation, or of every patch separately. In the latter case, the statistics are characterized by a suffix "_p" to the Stat name and by the words *(computed for each patch)* in the description of the statistic. Other summary statistics are computed for pairwise combinations of patches. These statistics are characterized by a suffix "_pair" to the Stat name and by the words *(all pairwise combinations computed)* in the description of the statistic. The names of such summary statistics in the output have the suffix "_pX" and "_pX-Y", respectively. Where X and Y are the index of the patches (starting with 1).

The summary statistics are computed by default for every trait. If several traits are simulated the postfix "_tT" is added to the summary statistic name in the output file, where T is the index of the traits. It is possible to compute statistics just for specified traits. This can be specified by the index of the type inserted just after the *(n)*. For example the stat option *(*n.adlt.fst) computes the Fst for all types of neutral markers while the stat option *(*n2.adlt.fst) computes the Fst just for the second neutral marker type.

stat [string/matrix]

This parameter allows specifying the summary statistics to be computed. A exhaustive list of existing statistics is given in the next sub-

section: 10.3.1, 10.3.2, 10.3.3 (except for statistic specific to coalescence which are treated in the chapter about coalescence, section 9.3). The arguments are keywords standing for one or multiple summary statistics. Keywords have to be written as a matrix within brackets separated by space.

The summary statistic name (column Stat name) may be used to specify the summary statistic to be computed (e.g. adlt.fst). Similar summary statistics (within a thematic group) may be obtained at once using the name within square brackets after the group title (e.g. adlt.fstat). Using this group statistic name all summary statistics of the thematic group marked with a star (*) will be computed.

```
stat {n.fstat
      quanti
      adlt.demo}
```

stat_save [0-6] (default: 0)

This parameter specifies if the summary statistics should be computed and how they should be dumped to file. The summary statistics may be dumped to file for each specified generation and replicate separately (file "generic_name_stats.txt"), or summary statistics may be summed up across replicates by their mean (file "generic_name_stats.txt") and their variance (file "generic_name_var.txt").

The column Stat name contains the name of the summary statistic used to specify which summary statistics are computed. These names appear also in the output file. The column Description contains a short description of the summary statistic.

0 : All. Output includes all types of summary statistic (files "generic_name_stats.txt", "generic_name_mean.txt", and "generic_name_var.txt").

1 : Detailed. Output includes only the file containing the summary statistics for each replicate separately (file "generic_name_stats.txt"). Since in this case (in contrast to all other options) it is not necessary to save the statistics over all replicates, the statistics are written to file when they are computed. This means that the internal database to store the statistics is not used, consequently, the memory used by quantiNemo does not increase with each generation and replicate.

2 : Summed up. Output includes the files containing the summary statistics summed up by their mean and variance across replicates (files "generic_name_mean.txt", and "generic_name_var.txt").

3 : Mean. Output includes only the file containing the summary statistics summed up by their mean across replicates (file "generic_name_mean.txt").

4 : Variance. Output includes only the file containing the summary statistics summed up by their variance across replicates (file "generic_name_var.txt").

5 : Median. Output includes only the file containing the summary statistics summed up by their median across replicates (file "generic_name_median.txt")

6 : None. No summary statistics are written. The life cycle event "Statistics" is skipped.

Whenever the summary statistics are output (parameter stat_save not set to 6) a file named "generic_name_legend.txt" containing a small description of the summary statistics is also generated.

stat_log_time [integer] (temporal/default: 1)

This is the time interval at which summary statistics are recorded. The interval must range between 1 and the number of generations. Since the parameter may change over time (temporal parameter) the summary statistics may be computed for any generation:

```
stat_log_time (1 1, 10 10, 100 100)
```

In this example for the first 9 generations the summary statistics are computed every generation, from the tenth until generation 99 they are computed at every tenth generation, and from the generation 100 every hundred generation.

stat_dir [string] (default: "")

This parameter is used to specify a subdirectory within the simulation folder (parameter folder) where the summary statistic files will be stored. If the parameter is not set, the files will be stored in the simulation folder.

stat_filename [string] (default: "")

This parameter is used to specify an individual base filename for the statistics. If not specified the generic base filename will be used (see parameter filename).

param [string/matrix]
This parameter allows specifying the parameter arguments to output together with the statistics. Note, the output is listed only in the detailed statistic file.

stat_NaN [string] (default: "NaN")
This parameter allows specifying a placeholder used for the output of the summary statistics for statistics which are not computable. By default the placeholder is *NaN*. This default value is correctly read by the statistical package R when such a file is imported.

10.3.1 Demography

Table 10.1: Summary statistics available for the demographic structure

Stat name	Description
Demography [(adlt/off).demo]	
(adlt/off).nbInd	total number of individuals in the metapopulation*
(adlt/off).nbFem	total number of females in the metapopulation*
(adlt/off).nbMal	total number of males in the metapopulation*
(adlt/off).meanInd	mean number of individuals per inhabited patch*
(adlt/off).meanFem	mean number of females per inhabited patch*
(adlt/off).meanMal	mean number of males per inhabited patch*
(adlt/off).sexRatio	sex ratio ($\frac{males}{females}$)*
(adlt/off).nbPops	number of inhabited patches*
(adlt/off).nbInd_p	number of individuals in patch i
(adlt/off).nbFem_p	number of females in patch i
(adlt/off).nbMal_p	number of males in patch i
Demography of all individuals: In contrast to all other stats and outputs the following stats consider all individuals and patches and not just the sampled ones. [(adlt/off).demoTot]	
(adlt/off).nbIndTot	total number of individuals in the metapopulation*
(adlt/off).nbFemTot	total number of females in the metapopulation*
(adlt/off).nbMalTot	total number of males in the metapopulation*
(adlt/off).meanIndTot	mean number of individuals per inhabited patch*
(adlt/off).meanFemTot	mean number of females per inhabited patch*
(adlt/off).meanMalTot	mean number of males per inhabited patch*

Table 10.1 continued on next page

Stat name	Description
(adlt/off).sexRatioTot	sex ratio of individuals ($\frac{males}{females}$)*
(adlt/off).nbPopsTot	number of inhabited patches*
(adlt/off).nbIndTot_p	number of individuals in patch i
(adlt/off).nbFemTot_p	number of females in patch i
(adlt/off).nbMalTot_p	number of males in patch i
Patch extinction [ext.rate]	
ext.rate	proportion of extinct patches in the metapopulation*
Fecundity [fecundity, available only for adults]	
fem.meanFec	mean realized female fecundity*
fem.varFec	mean variance of realized female fecundity*
mal.meanFec	mean realized male fecundity*
mal.varFec	mean variance of realized male fecundity*
Kinship [(adlt/off).kinship]	
(adlt/off).fsib	mean proportion of full-sib*
(adlt/off).phsib	mean proportion of paternal half-sib*
(adlt/off).mhsib	mean proportion of maternal half-sib*
(adlt/off).nsib	mean proportion of non-sib*
(adlt/off).self	mean proportion of selfed offspring*
Migration [migration, available only for adults]	
emigrants	mean number of emigrants per patch*
immigrants	mean number of immigrants per patch*
residents	mean number of residents per patch*
immigrate	mean effective immigration rate per patch* ($\frac{immigrants}{immigrants+residents}$)
colonisers	mean number of colonizers per extinct patch*
colon.rate	mean effective colonization rate of extinct patches*
Fitness [fitness, available only for adults]	
VwW	variance of the fitness of adults within patches*
VwB	variance of the fitness of adults between patches*
meanW_p	mean fitness of adults in patch i (computed for each patch)
varW_p	variance of the fitness of adults in patch i (computed for each patch)

Table 10.1 continued on next page

Stat name	Description
Random number initialization [seed]	
seed	outputs all used seeds (not really a statistic)*

Table 10.1: Summary statistics available for the demographic structure continued

10.3.2 neutral markers

Table 10.2: Summary statistics available for neutral markers

Stat name	Description
Genotype coancestry [n.(adlt/off).coa]	
n.(adlt/off).theta	mean within patch coancestry*
n.(adlt/off).alpha	mean between patch coancestry*
n.(adlt/off).thetaFF	mean within patch, within females coancestry*
n.(adlt/off).thetaMM	mean within patch, within males coancestry*
n.(adlt/off).thetaFM	mean within patch, between sexes coancestry*
n.(adlt/off).coa.fsib	mean coancestry within full-siblings*
n.(adlt/off).coa.phsib	mean coancestry within paternal half-siblings*
n.(adlt/off).coa.mhsib	mean coancestry within maternal half-siblings*
n.(adlt/off).coa.nsib	mean coancestry within non-siblings*
n.(adlt/off).theta_p	mean coancestry within patch i (computed for each patch)
n.(adlt/off).alpha_pair	mean coancestry between patch i and j (all pairwise combinations computed)
Genetic diversity [n.(adlt/off).gendiv]	
number of alleles:	
n.(adlt/off).nbAll	mean across patches and loci*
n.(adlt/off).nbAll_p	mean across loci (computed for each patch)
n.(adlt/off).nbAll_l	mean across patches (computed for each locus)
n.(adlt/off).nbAll_p_l	(computed for each patch and locus)
n.(adlt/off).nbAll.tot	mean total across loci*
n.(adlt/off).nbAll.tot_l	total (computed for each locus)
number of fixed loci:	
n.(adlt/off).nbFixLoc	mean across patches and loci*
n.(adlt/off).nbFixLoc_p	mean across loci (computed for each patch)
n.(adlt/off).nbFixLoc_l	mean across patches (computed for each locus)
n.(adlt/off).nbFixLoc_p_l	(computed for each patch and locus)
n.(adlt/off).nbFixLoc.tot	mean total across loci*
n.(adlt/off).nbFixLoc.tot_l	total (computed for each locus)
allele frequencies (caution: any potential allele is outputted!):	
n.(adlt/off).a.freq	local (computed for each patch, locus and allele)
n.(adlt/off).a.freq.global	global (computed for each locus and allele)

Table 10.2 continued on next page

Stat name	Description
locus gentoype frequencies (caution: any potential allele combination is outputted!):	
n.(adlt/off).l.freq	local (computed for each patch, locus genotype)
n.(adlt/off).l.freq.global	global (computed for each locus genotype)
observed heterozygosity following Nei and Chesser (1983):	
n.(adlt/off).ho	*
n.(adlt/off).ho_p	(computed for each patch)
n.(adlt/off).ho_l	(computed for each locus)
n.(adlt/off).ho_p_l	(computed for each patch and locus)
expected heterozygosity following Nei and Chesser (1983):	
n.(adlt/off).hs	*
n.(adlt/off).hs_p	(computed for each patch)
n.(adlt/off).hs_l	(computed for each locus)
n.(adlt/off).hs_p_l	(computed for each patch and locus)
n.(adlt/off).ht	total*
n.(adlt/off).ht_l	total (computed for each locus)
expected heterozygosity ($H = 1 - \sum p^2$):	
n.(adlt/off).hs.p2	
n.(adlt/off).hs.p2_p	(computed for each patch)
n.(adlt/off).hs.p2_l	(computed for each locus)
n.(adlt/off).hs.p2_p_l	(computed for each patch and locus)
n.(adlt/off).ht.p2	total*
n.(adlt/off).ht.p2_l	total (computed for each locus)
allelic richness following El Mousadik and Petit (1996):	
(rarefaction is based on the smallest sample size)	
n.(adlt/off).rs	
n.(adlt/off).rs_p	(computed for each patch)
n.(adlt/off).rs_l	(computed for each locus)
n.(adlt/off).rs_p_l	(computed for each patch and locus)
n.(adlt/off).rt	total
n.(adlt/off).rt_l	total (computed for each locus)
allelic range (difference between min and max allele):	
n.(adlt/off).r	
n.(adlt/off).r_p	(computed for each patch)

Table 10.2 continued on next page

Stat name	Description
n.(adlt/off).r_l	(computed for each locus)
n.(adlt/off).r_p_l	(computed for each patch and locus)
n.(adlt/off).r.tot	total
n.(adlt/off).r.tot_l	total (computed for each locus)
garza-williamsons statistic following Garza and Williamson (2001):	
(modification: $gw = \sum(nb.allele)/\sum(1 + range)$)	
n.(adlt/off).gw	
n.(adlt/off).gw_p	(computed for each patch)
n.(adlt/off).gw_l	(computed for each locus)
n.(adlt/off).gw_p_l	(computed for each patch and locus)
n.(adlt/off).gw.tot	total
n.(adlt/off).gw.tot_l	total (computed for each locus)
F-statistics following Nei and Chesser (1983) [n.(adlt/off).fstat]	
n.(adlt/off).fst	global F_{ST}*
n.(adlt/off).fst_l	global F_{ST} (computed for each locus)
n.(adlt/off).fst_pair	pairwise F_{ST} between patch i and j (all pairwise combinations computed)
n.(adlt/off).fst_pair_l	pairwise F_{ST} between patch i and j (all pairwise combinations computed for each locus separately)
n.(adlt/off).fis	global F_{IS}*
n.(adlt/off).fis_l	global F_{IS} (computed for each locus)
n.(adlt/off).fit	global F_{IT}*
n.(adlt/off).fit_l	global F_{IT} (computed for each locus)
F-statistics following Weir and Cockerham (1984) [n.(adlt/off).fstat.wc]	
n.(adlt/off).fst.wc	global F_{ST}*
n.(adlt/off).fst.wc_l	global F_{ST} (computed for each locus)
n.(adlt/off).fst.wc_pair	pairwise F_{ST} between patch i and j (all pairwise combinations computed)
n.(adlt/off).fst.wc_pair_l	pairwise F_{ST} between patch i and j (all pairwise combinations computed for each locus separately)
n.(adlt/off).fis.wc	global F_{IS}*
n.(adlt/off).fis.wc_l	global F_{IS} (computed for each locus)
n.(adlt/off).fit.wc	global F_{IT}*

Table 10.2 continued on next page

Stat name	Description
n.(adlt/off).fit.wc_l	global F_{IT} (computed for each locus)
Linkage disequilibirum reviewed in Devlin and Risch (1995)	
n.(adlt/off).Dprime	global *Dprime* (computed globally for each pair of loci)
n.(adlt/off).Dprime_pair	$Dprime_p$ (computed for each patch each pair of loci)
n.(adlt/off).Dstar	global *Dstar* (computed globally for each pair of loci)
n.(adlt/off).Dstar_pair	$Dstar_p$ (computed for each patch each pair of loci)
n.(adlt/off).R2	global $R2$ (computed globally for each pair of loci)
n.(adlt/off).R2_pair	$R2$ (computed for each patch and each pair of loci)
n.(adlt/off).Chi2	global $Chi2$ (computed globally for each pair of loci)
n.(adlt/off).Chi2_pair	$Chi2$ (computed for each patch and each pair of loci)

Table 10.2: Summary statistics available for neutral markers continued

10.3.3 quantitative traits

Table 10.3: Summary statistics available for quantitative traits

Stat name	Description
Quantitative trait statistics [quanti, available only for adults]	
q.VgW	genetic variance within patches*
q.VgB	genetic variance between patches*
q.VpW	phenotypic variance within patches*
q.VpB	phenotypic variance between patches*
q.VaW	additive genetic variance within patches
q.qst	Q_{ST}
q.qst.f	Q_{ST} corrected for inbreeding following Bonnin et al. (1996). Inbreeding coefficient F computed following Nei and Chesser (1983)
q.qst_pair	Q_{ST} between patch i and j (all pairwise combinations computed)
q.qst.f_pair	Q_{ST} between patch i and j (all pairwise combinations computed) corrected for inbreeding
q.varA_p	additive genetic variance of patch i following Lynch and Walsh (1998, p85-87) (computed for each patch)
q.meanG_p	genetic mean of patch i (computed for each patch)
q.varG_p	genetic variance of patch i (computed for each patch)

Table 10.3 continued on next page

Stat name	Description
q.meanP_p	phenotypic mean of patch i (computed for each patch)
q.varP_p	phenotypic variance of patch i (computed for each patch)
Genotype coancestry [q.(adlt/off).coa]	
q.(adlt/off).theta	mean within patch coancestry*
q.(adlt/off).alpha	mean between patch coancestry*
q.(adlt/off).thetaFF	mean within patch, within females coancestry*
q.(adlt/off).thetaMM	mean within patch, within males coancestry*
q.(adlt/off).thetaFM	mean within patch, between sexes coancestry*
q.(adlt/off).coa.fsib	mean coancestry within full-siblings*
q.(adlt/off).coa.phsib	mean coancestry within paternal half-siblings*
q.(adlt/off).coa.mhsib	mean coancestry within maternal half-siblings*
q.(adlt/off).coa.nsib	mean coancestry within non-siblings*
q.(adlt/off).theta_p	mean coancestry within patch i (computed for each patch)
q.(adlt/off).alpha_pair	mean coancestry between patch i and j (all pairwise combinations computed)
Genetic diversity [q.(adlt/off).gendiv]	
number of alleles:	
q.(adlt/off).nbAll	mean across patches and loci*
q.(adlt/off).nbAll_p	mean across loci (computed for each patch)
q.(adlt/off).nbAll_l	mean across patches (computed for each locus)
q.(adlt/off).nbAll_p_l	(computed for each patch and locus)
q.(adlt/off).nbAll.tot	mean total across loci*
q.(adlt/off).nbAll.tot_l	total (computed for each locus)
number of fixed loci:	
q.(adlt/off).nbFixLoc	mean across patches and loci*
q.(adlt/off).nbFixLoc_p	mean across loci (computed for each patch)
q.(adlt/off).nbFixLoc_l	mean across patches (computed for each locus)
q.(adlt/off).nbFixLoc_p_l	(computed for each patch and locus)
q.(adlt/off).nbFixLoc.tot	mean total across loci*
q.(adlt/off).nbFixLoc.tot_l	total (computed for each locus)
allele frequencies (caution: any potential allele is outputted!):	
q.(adlt/off).a.freq	local (computed for each patch, locus and allele)
q.(adlt/off).a.freq.global	global (computed for each locus and allele)

Table 10.3 continued on next page

Stat name	Description
locus gentoype frequencies (caution: any potential allele combination is outputted!):	
q.(adlt/off).l.freq	local (computed for each patch, locus genotype)
q.(adlt/off).l.freq.global	global (computed for each locus genotype)
observed heterozygosity following Nei and Chesser (1983):	
q.(adlt/off).ho	*
q.(adlt/off).ho_p	(computed for each patch)
q.(adlt/off).ho_l	(computed for each locus)
q.(adlt/off).ho_p_l	(computed for each patch and locus)
expected heterozygosity following Nei and Chesser (1983):	
q.(adlt/off).hs	*
q.(adlt/off).hs_p	(computed for each patch)
q.(adlt/off).hs_l	(computed for each locus)
q.(adlt/off).hs_p_l	(computed for each patch and locus)
q.(adlt/off).ht	total*
q.(adlt/off).ht_l	total (computed for each locus)
expected heterozygosity ($H = 1 - \sum p^2$):	
q.(adlt/off).hs.p2	
q.(adlt/off).hs.p2_p	(computed for each patch)
q.(adlt/off).hs.p2_l	(computed for each locus)
q.(adlt/off).hs.p2_p_l	(computed for each patch and locus)
q.(adlt/off).ht.p2	total*
q.(adlt/off).ht.p2_l	total (computed for each locus)
allelic richness following El Mousadik and Petit (1996):	
(rarefaction is based on the smallest sample size)	
q.(adlt/off).rs	
q.(adlt/off).rs_p	(computed for each patch)
q.(adlt/off).rs_l	(computed for each locus)
q.(adlt/off).rs_p_l	(computed for each patch and locus)
q.(adlt/off).rt	total
q.(adlt/off).rt_l	total (computed for each locus)
allelic range (difference between min and max allele):	
q.(adlt/off).r	
q.(adlt/off).r_p	(computed for each patch)

Table 10.3 continued on next page

Stat name	Description
q.(adlt/off).r_l	(computed for each locus)
q.(adlt/off).r_p_l	(computed for each patch and locus)
q.(adlt/off).r.tot	total
q.(adlt/off).r.tot_l	total (computed for each locus)
garza-williamsons statistic following Garza and Williamson (2001):	
(modification: $gw = \sum(nb.allele)/\sum(1 + range)$)	
q.(adlt/off).gw	
q.(adlt/off).gw_p	(computed for each patch)
q.(adlt/off).gw_l	(computed for each locus)
q.(adlt/off).gw_p_l	(computed for each patch and locus)
q.(adlt/off).gw.tot	total
q.(adlt/off).gw.tot_l	total (computed for each locus)
F-statistics following Nei and Chesser (1983) [q.(adlt/off).fstat]	
q.(adlt/off).fst	global F_{ST}*
q.(adlt/off).fst_l	global F_{ST} (computed for each locus)
q.(adlt/off).fst_pair	pairwise F_{ST} between patch i and j (all pairwise combinations computed)
q.(adlt/off).fst_pair_l	pairwise F_{ST} between patch i and j (all pairwise combinations computed for each locus separately)
q.(adlt/off).fis	global F_{IS}*
q.(adlt/off).fis_l	global F_{IS} (computed for each locus)
q.(adlt/off).fit	global F_{IT}*
q.(adlt/off).fit_l	global F_{IT} (computed for each locus)
F-statistics following Weir and Cockerham (1984) [q.(adlt/off).fstat.wc]	
q.(adlt/off).fst.wc	global F_{ST}*
q.(adlt/off).fst.wc_l	global F_{ST} (computed for each locus)
q.(adlt/off).fst.wc_pair	pairwise F_{ST} between patch i and j (all pairwise combinations computed)
q.(adlt/off).fst.wc_pair_l	pairwise F_{ST} between patch i and j (all pairwise combinations computed for each locus separately)
q.(adlt/off).fis.wc	global F_{IS}*
q.(adlt/off).fis.wc_l	global F_{IS} (computed for each locus)
q.(adlt/off).fit.wc	global F_{IT}*

Table 10.3 continued on next page

Stat name	Description
q.(adlt/off).fit.wc_l	global F_{IT} (computed for each locus)
Linkage disequilibirum reviewed in Devlin and Risch (1995)	
q.(adlt/off).Dprime	global *Dprime* (computed globally for each pair of loci)
q.(adlt/off).Dprime_pair	$Dprime_p$ (computed for each patch each pair of loci)
q.(adlt/off).Dstar	global *Dstar* (computed globally for each pair of loci)
q.(adlt/off).Dstar_pair	$Dstar_p$ (computed for each patch each pair of loci)
q.(adlt/off).R2	global $R2$ (computed globally for each pair of loci)
q.(adlt/off).R2_pair	$R2$ (computed for each patch and each pair of loci)
q.(adlt/off).Chi2	global $Chi2$ (computed globally for each pair of loci)
q.(adlt/off).Chi2_pair	$Chi2$ (computed for each patch and each pair of loci)

Table 10.3: Summary statistics available for quantitative traits continued

10.4 Raw data

10.4.1 Genotype

The genotype of the sampled individuals and populations may periodically be dumped to files. The output files will be stored in the folder given by the parameter (ntrl/quanti)_genot_dir and will have the name of the base file name (see parameter filename in section 4). The extension is ".dat". A counter for the generation (e.g. _g05) and the replicate (e.g. _r4) is inserted before the extension. An example of such a file name is "simulation_g05_r4.dat". Note, that such a genotype file may be used to start a new simulation (see parameter **(ntrl/quanti_ini_genotypes**). By default all individuals of all populations are sampled. Section 10.2 describes how to specify a sampling schema.

(ntrl/quanti)_save_genotype [0-2] (default: 0)
This parameter specifies the output of the quantitative genotype at the QTLs.

0 : None. No output is generated.

1 : FSTAT. Genotypes are outputted in the FSTAT format (Goudet, 1995).

2 : FSTAT extended. Same as point 1, but the file contain the following six additional columns: the age class (1 = offspring, 2 =

adult), the sex (0 = male; 1 = female), the ID of the individual, the ID of the mother, the ID of the father, and the fitness of the individual. The ID is a unique identifier for each individual of a simulation in the format "345_23", meaning that this is the 345th individual born in patch 23. The IDs of the individual, the mother and the father allow to extract pedigree informations, if the output is stored for each generation, and also to investigate the migration behavior of the individual and its parents.

3 : Arlequin. Genotypes are outputted in Arlequin format (Excoffier, 2010).

4 : Arlequin extended. Same as point 3, but with additional commented individual information as in point 2.

5 : PLINK. Outputs the **quantitative** genotypes with the phenotypes of the quantitative traits (Purcell et al., 2007). The standard two files .ped and .map are created. The 6th column (phenotype) of the .ped file contains the fitness of the individual. If quantitative traits are simulated, their phenotypes are listed in an alternate phenotype file (.pheno). The alleles of all bi-allelic loci (quanti_nb_allele set to 2) are listed. The .map file is generated for each replicate. The .ped and .pheno files are generated for each specified time point and outputs of successive time points are concatenated to a single file, allowing to obtain entire or parts of pedigrees. The filename of such a concatenated file contains the time stamp of the first entry. Note that only successive individuals list their parentIDs. To generate an entire pedigree the entire populations have to be sampled.

6 : PLINK extended. Same as point 5, but all **quantitative** genotypes are listed, including the multi-allele loci.

An example of such a file (with quanti_save_genotype set to 2):

```
5 4 20 2
t1_l1
t1_l2
t1_l3
t1_l4
1 1415 1019 2002 0820 1 1 10_1 1_1 0_1 0.345
1 0814 0219 2002 2020 1 1 11_1 8_1 2_4 0.334
1 0808 0217 1902 0820 1 1 12_1 5_3 5_1 0.123
```

```
...
5 1004 0917 1404 1007 1 1 16_5 9_5 3_2 0.999
5 2017 1010 2013 1812 1 0 17_5 3_2 9_2 1.000
5 2017 1008 2013 1811 1 1 11_4 8_2 9_2 0.678
```

The first line contains the number of patches (5 patches here), the number of loci (4), the highest possible allele index (20), and the number of digits used to write each allele (2). The next four lines contain the locus names. The following lines contain the individual's info, one individual per line. The first number is the patch number of the individual, followed by the genotype. Each column represents a locus, and the first half of the locus (first 2 digits) represents the first allele index, while the second half of the locus (last 2 digits) the second allele at the given locus. As, in this example, we are using two digits per allele, the first two digits of a locus genotype number are the first allele (e.g. allele 14 for the first allele of the first locus of the first individual) while the two next digits are the second allele (e.g. allele 15 for the second allele of the first locus of the first individual). Each line ends with six columns consisting supplementary information on the individual (see above) if the parameter quanti_save_genotype is set to 2.

(ntrl/quanti)_genot_dir [string] (default: "")
This parameter allows to specify the subdirectory where the genotypes are stored. This directory has to be specified relative to the simulation folder (parameter folder), and may also contain subdirectories. If not specified (default) the output is stored in the simulation folder (parameter folder).

(ntrl/quanti)_genot_filename [string] (default: "")
This parameter is used to specify an individual base filename for the genotypes. If not specified the generic base filename will be used (see parameter filename).

(ntrl/quanti)_genot_logtime [integer] (temporal/default: 1)
This parameter specifies the time interval of the genotype output. Since the parameter may change over time the output may be generated at any generation.

(ntrl/quanti)_genot_script [string] (default: "")
It is possible to launch a script just after the genotype file is generated.

The argument of the parameter is the file name of the script. The name of the genotype file is passed as unique parameter to the script.

(ntrl/quanti)_genot_sex [0-2] (default: 0)
This parameter allows to choose which sex is output.

0 : Both. Output includes both sexes.

1 : Females. Output includes only female genotypes.

2 : Males. Output includes only male genotypes.

(ntrl/quanti)_genot_age [0-2] (default: 0)
This parameter allows to choose which age is output.

0 : Adults. Output includes only adult genotypes.

1 : Juveniles. Output includes only juvenile genotypes.

2 : Both. Output includes juveniles and adults genotypes.

10.4.2 Genotypic value

Similar to the genotypes the genotypic values may be periodically dumped to files. The output files will be stored in the folder given by the parameter quanti_geno_value_dir and will have the name of the base file name (see parameter filename in chapter 4). The extension is ".gen". A counter for the generation (e.g. _g05) and the replicate (e.g. _r4) is inserted before the extension. An example of such a file name is "simulation_g05r4.gen". By default, all individuals of all populations are sampled. The section 10.2 describes how to specify a sampling schema.

quanti_save_geno_value [0-2] (default: 0)
This parameter specifies the output of the phenotype.

0 : None. No output is generated.

1 : Standard. The output contains the phenotypes in the standard FSTAT-like format (Goudet, 1995).

2 : Extended. Same as point 1, but the file contain the following six additional columns: the age class (1 = offspring, 2 = adult), the

sex (0 = male; 1 = female), the ID of the individual, the ID of the mother, the ID of the father, and the fitness of the individual. The ID is a unique identifier for each individual of a simulation in the format "345_23", meaning that this is the 345th individual born in patch 23. The IDs of the individual, the mother, and the father allow extracting pedigree information, if the output is stored for each generation, and also to investigate the migration behavior of the individual and its parents.

An example of such a file (quanti_save_geno_value is set to 2):

```
2 5
genotypic_value_trait-1
genotypic_value_trait-2
genotypic_value_trait-3
genotypic_value_trait-4
genotypic_value_trait-5
1 0.0493 -3.203 -2.441 0.0683 -3.199 2 1 10_1 1_1 0_1 0.345
1 0.4924 -3.803 -0.869 -2.002 -2.594 2 1 11_1 8_1 2_2 0.334
1 2.2342 -2.931 -0.725 -0.750 -0.698 2 1 12_1 5_2 5_1 0.123
...
2 0.8623 0.6525 -0.857 1.7483 -4.194 2 1 16_2 9_2 3_2 0.999
2 1.7752 -2.223 -3.117 0.3409 -2.003 2 1 17_2 3_2 9_2 1.000
2 0.2081 -2.803 -0.146 -0.456 -5.137 2 1 11_1 8_1 9_1 0.678
```

The first line contains the number of patches (2 patches here), and the number of traits (5). The next five lines contain the five trait names. The following lines contain the individual's info, one individual per line. The first number is the patch number of the individual, followed by the genotypic value for each trait. Each line ends with six columns consisting supplementary information on the individual (see above) if the parameter quanti_save_geno_value is set to 2.

quanti_geno_value_dir [string] (default: "")

This parameter allows to specify the subdirectory where genotypic values are stored. This directory has to be specified relative to the simulation folder (parameter folder), and may also contain subdirectories. If not specified (default) the output is stored in the simulation folder (parameter folder).

quanti_geno_value_filename [string] (default: "")

This parameter is used to specify an individual base filename for the

genotypic values. If not specified the generic base filename will be used (see parameter filename).

quanti_geno_value_logtime [integer] (temporal/default: 1)
This parameter specifies the time interval of the genotypic value output. Since the parameter may change over time the output may be generated at any generation.

quanti_geno_value_script [string] (default: "")
It is possible to launch a script just after the genotypic value file is generated. The argument of the parameter is the file name of the script. The name of the genotypic value file is passed as unique parameter to the script.

quanti_geno_value_sex [0-2] (default: 0)
This parameter allows to choose which sex is output.

0 : Both. Output includes both sexes.

1 : Females. Output includes only female genotypic values.

2 : Males. Output includes only male genotypic values.

quanti_geno_value_age [0-2] (default: 0)
This parameter allows to choose which age is output.

0 : Adults. Output includes only adult genotypic values.

1 : Juveniles. Output includes only juvenile genotypic values.

2 : Both. Output includes juveniles and adults genotypic values.

10.4.3 Phenotypic value

Similar to the genotypes the phenotypic values of the adults may be periodically dumped to files. The phenotype of juveniles cannot be output, as the phenotype is only computed when selection acts, and this is at the reproduction stage, *i.e.* when individuals are adults. The output files will be stored in the folder given by the parameter quanti_phenot_dir and will have the name of the base file name (see parameter filename in section 4). The extension is ".phe". A counter for the generation (e.g. _g05) and the replicate (e.g. _r4) is inserted before the extension. An example of such a file name

is "simulation_g05r4.phe". By default all individuals of all populations are sampled. The section 10.2 describes how to specify a sampling schema.

quanti_save_phenotype [0-2] (default: 0)

This parameter specifies the output of the phenotype.

0 : None. No output is generated.

1 : Standard. The output contains the phenotypes in the standard FSTAT-like format (Goudet, 1995).

2 : Extended. Same as point 1, but the file contain the following six additional columns: the age class (1 = offspring, 2 = adult), the sex (0 = male; 1 = female), the ID of the individual, the ID of the mother, the ID of the father, and the fitness of the individual. The ID is a unique identifier for each individual of a simulation in the format "345_23", meaning that this is the 345th individual born in patch 23. The IDs of the individual, the mother, and the father allow extracting pedigree information, if the output is stored for each generation, and also to investigate the migration behavior of the individual and its parents.

An example of such a file (quanti_save_phenotype is set to 2):

```
2 5
phenotypic_value_trait-1
phenotypic_value_trait-2
phenotypic_value_trait-3
phenotypic_value_trait-4
phenotypic_value_trait-5
1 0.0493 -3.203 -2.441 0.0683 -3.199 2 1 10_1 1_1 0_1 0.345
1 0.4924 -3.803 -0.869 -2.002 -2.594 2 1 11_1 8_1 2_2 0.334
1 2.2342 -2.931 -0.725 -0.750 -0.698 2 1 12_1 5_2 5_1 0.123
...
2 0.8623 0.6525 -0.857 1.7483 -4.194 2 1 16_2 9_2 3_2 0.999
2 1.7752 -2.223 -3.117 0.3409 -2.003 2 1 17_2 3_2 9_2 1.000
2 0.2081 -2.803 -0.146 -0.456 -5.137 2 1 11_1 8_1 9_1 0.678
```

The first line contains the number of patches (2 patches here), and the number of traits (5). The next five lines contain the five trait names. The following lines contain the individual's info, one individual per line. The first number is the patch number of the individual, followed

by the phenotype value for each trait. Each line ends with six columns consisting supplementary information on the individual (see above) if the parameter quanti_save_phenotype is set to 2.

quanti_phenot_dir [string] (default: "")
This parameter allows to specify the subdirectory where phenotypes are stored. This directory has to be specified relative to the simulation folder (parameter folder), and may also contain subdirectories. If not specified (default) the output is stored in the simulation folder (parameter folder).

quanti_phenot_filename [string] (default: "")
This parameter is used to specify an individual base filename for the phenotypes. If not specified the generic base filename will be used (see parameter filename).

quanti_phenot_logtime [integer] (temporal/default: 1)
This parameter specifies the time interval of the phenotype output. Since the parameter may change over time the output may be generated at any generation.

quanti_phenot_script [string] (default: "")
It is possible to launch a script just after the phenotype file is generated. The argument of the parameter is the file name of the script. The name of the phenotype file is passed as unique parameter to the script.

quanti_phenot_sex [0-2] (default: 0)
This parameter allows choosing which sex is output.

0 : Both. Output includes both sexes.

1 : Females. Output includes only female phenotypes.

2 : Males. Output includes only male phenotypes.

10.5 print input file

Depending on the definition of the architecture of the quantitative trait it is possible to obtain the allelic file, the dominance file, and/or the epistatic

file. All three files follow the structure of the homonymous input files, except that for the input all possible combinations have to be present, while in the output only the used combinations are output. Therefore the output files of allelic, dominance, and epistatic cannot always be used as input files. The files have the names "allelic_values.txt", "dominance_values.txt, and "epistatic_values.txt" and are stored in the simulation folder (parameter folder). If several replicates are simulated the files are generated for each replicate and a replicate counter (e.g. _r4) is inserted before the extension.

quanti_output [0,1] (default: 0)

- **0 : None.** The files are not generated.
- **1 : Output.** The allelic, the dominance, and/or the epistatic files are stored in the simulation folder (parameter folder), if they were used during the simulation.

Bibliography

Beverton, R. J. H. and Holt, S. J. (1957), On the dynamics of exploited fish populations, Technical report, U.K. Ministry of Agriculture and Fisherie.

Bonnin, I., Prosperi, J. M. and Olivieri, I. (1996), 'Genetic markers and quantitative genetic variation in medicago truncatula (leguminosae): A comparative analysis of population structure', *Genetics* **143**, 1795–1805.

Devlin, B. and Risch, N. (1995), 'A comparison of linkage disequilibrium measures for fine-scale mapping', *Genomics* **29**, 311–322.

El Mousadik, A. and Petit, R. J. (1996), 'High level of genetic differentiation for allelic richness among populations of the argan tree [argania spinosa (l.) skeels] endemic to morocco', *Theoretical and Applied Genetics* **92**, 832–839.

Excoffier, L. L. H. (2010), 'Arlequin suite ver 3.5: A new series of programs to perform population genetics analyses under linux and windows', *Molecular Ecology Resources* **10**(3), 564–567.

Garza, J. C. and Williamson, E. G. (2001), 'Detection of reduction in population size using data from microsatellite loci', *Molecular Ecology* **10**, 305–318.

Goudet, J. (1995), 'Fstat (version 1.2): A computer program to calculate f-statistics', *Journal of Heredity* **86**(6), 485–486.

Haldane, J. B. S. (1919), 'The combination of linkage values, and the calculation of distances between loci of linked factors. journal of genetics', *Journal of Genetics* **8**, 299–309.

Lynch, M. and Walsh, B. (1998), *Genetics and analysis of quantitative traits*, Publisher Sinauer Associates Inc.

Nei, M. and Chesser, R. K. (1983), 'Estimation of fixation indexes and gene diversities', *Annals of Human Genetics* **47**(Jul), 253–259.

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M., Bender, D., Maller, J., Sklar, P., de Bakker, P., Daly, M. and Sham, P. (2007), 'Plink: a toolset for whole-genome association and population-based linkage analysis', *American Journal of Human Genetics* **81**, 559–575.

Ravigne, V., Olivieri, I. and Dieckmann, U. (2004), 'Implications of habitat choice for protected polymorphisms', *Evolutionary Ecology Research* **6**(1), 125–145.

Richards, F. (1959), 'A flexible growth function for empirical use', *J. Exp. Bot* **10**, 290–300.

Wallace, B. (1968), Polymorphism, population size, and genetic load, *in* R. C. Lewontin, ed., 'Population biology and evolution', Syracuse University Press, Syracuse, N. Y., pp. 87–108.

Wallace, B. (1975), 'Hard and soft selection revisited', *Evolution* **29**, 465–473.

Weir, B. S. and Cockerham, C. C. (1984), 'Estimating f-statistics for the analysis of population structure', *Evolution* **38**, 1358 – 1370.

Appendices

Appendix A

technical details

A.1 Allelic value distribution

In this appendix, we will see why quantiNemo simulates allelic value the way it does. The goal of simulation software in general is not to match nature as close as possible, but more to simulate a model which retains the essential ingredient of reality, letting apart unimportant parameters. QuantiNemo uses a lot of such approximations. This section is about such an approximation used in quantiNemo which might seem strange at first sight but is actually quite powerful.

The phenotype of a quantitative trait is determined by the genotype of the individual (and maybe some environmental effects). When one of the underlying genes mutates, leading to a new allele, the phenotype generally also changes. In general, it is rather likely that the mutation will only lead to a small change in the phenotype. This could be easily modeled by drawing the value of a new allele in a normal distribution. In quantiNemo, allelic values are drawn at the beginning of a simulation. Then a simple idea would be to proceed as follow:

- Draw all allelic value in a normal distribution centered around zero at the beginning of the simulation
- Mutate to any allele with the same probability

This leads however to two problems. First, the genotype is somehow stuck

around zero, since most of the alleles have a value close to zero, and only a few are further away. If a locus switch to an allele with a high value, my next mutation is likely to get me back close to 0, which mean that I cannot gradually increase my phenotype. This problem could probably be overcome by various ways but at some cost.

The second problem is somehow more fundamental. If the number of alleles is small, it is likely that it will be drawn with some bias at the beginning of the simulation (i.e. that the mean phenotype will not be exactly zero). Since the values are determined once and for all at the beginning of the simulation, the bias will stay the same.

To illustrate this problem, let's suppose that we have a stabilizing selection around zero but some noise due to mutation. If we average over generations, we expect the mean value of the phenotype to converge towards zero. However, due to the initial bias, this will not be the case. This bias would of course also appear in other statistics adding unnecessary complexity to the data analysis.

To overcome these problems, quantiNemo rather follow the following algorithm:

- The allelic values are equally spaced (like in [-3; -2; -1; 0; 1; 2; 3])
- At the beginning of the simulation, the frequency of the central allele is higher than the frequency of allele with a very high or very low value, following a Gaussian distribution.
- When a mutation occurs, it is more likely to draw an allele close to the current value than far from it (or close or far from zero, depending on the mutation model).

We see that at the beginning of the simulation, both algorithms lead to a phenotype with a normal shape with a variance twice as big as the one of the allelic value. And it allows to overcome the previously mentioned problem:

For the first one, there are as many alleles around zero than around a large value. The new allele can be drawn from a re-centered normal distribution, allowing the mean phenotype to evolve smoothly through time, or always around zero to simulate a "reference allele".

For the second problem, since we choose the allelic values to be equally spaced and centered around zero, we know that if we average over enough generation, the mean phenotype will be zero and not bias will come from here. In Fig. A.1 we sketch the main differences between the two approaches

More precisely, quantiNemo proceed as follow:

- At the beginning of the simulation, the allele are defined with values equally spaced between -6σ and 6σ or between -20σ and 20σ depending of the mutation model (see parameter quanti_mutation_model). σ is defined by the parameter quanti_allelic_var. If the number of allele is smaller than 6, the value will be spaced between $-(l-1)\sigma$ and $(l-1)\sigma$ where l is the number of allele.

- By default, the initial allele frequency will follow a Gaussian law: $f(a_i) = \frac{1}{N}\exp(-a_i^2/2\sigma^2)$, were N allow to normalize the frequency (the sum of frequency should be one) and a_i is the allelic value of allele i.

- When a mutation occurs,the probability to switch to a given allele is given by $f(a_i) = \frac{1}{N}\exp(-a_i^2/2\sigma^2)$ in the RMM model and by $f(a_i) = \frac{1}{N}\exp(-(a_i - a_c)^2/2\sigma^2)$ in the IMM model (where a_c is the allelic value of the current allele). Loosely speaking, this mean that in the RMM model, it is likely to go to an allele with a value close to zero, while in the IMM model, it is likely to mutate to an allele close to the current allelic value a_c.

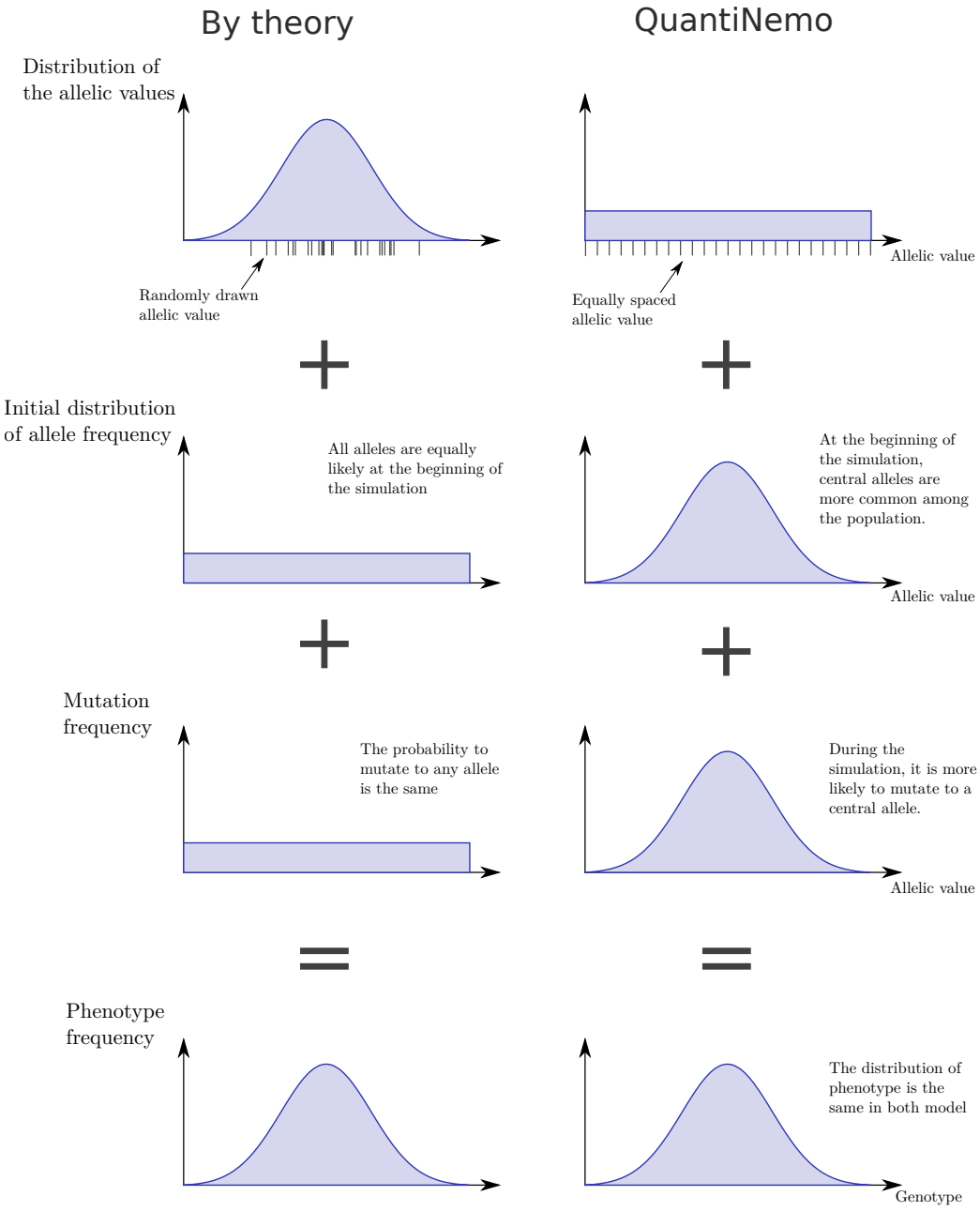


Figure A.1: Schematic comparison between a "natural" algorithm and the one adopted by quantiNemo to set allelic values

A.2 Multiple traits with varying types of selection on various patch: simple case

Phenotypes for quantitative traits may be under selection. Selection pressures may vary among quantitative traits, sexes, patches, and time. To specify the selection pressure individually for quantitative traits and patches, matrices may be used. They are adjusted to the number of quantitative traits and to the number of patches if needed. If a parameter does not change among quantitative traits and patches, a single value may be used as argument. Selection pressures have to be specified either for each sex separately (parameters with the suffix "_fem" for females and "_mal" for males), or for both sexes together (parameters without a suffix). In the first case both sex specific parameters have to be set if two sexes are simulated. In the latter case, the selection pressure of females and males are assumed to be identical. Each row of the matrix corresponds to a quantitative trait, each column to a patch:

```
{ {patch_1 patch_2 . . . patch_n}    # trait 1
  {patch_1 patch_2 . . . patch_n}    # trait 2
  ...
  {patch_1 patch_2 . . . patch_n} }  # trait m
```

Example

```
patch_number              2
quanti_nb_trait           3
patch_stab_sel_optima     {{-0.1   0.1}
                           { 0.2   0.2}
                           {-0.3   0.3}}
patch_stab_sel_intensity  1
```

In this example, the environment consists of two patches with varying selection pressures. Three quantitative traits are simulated. The first trait has a selection optimum at -0.1 in patch 1 and at 0.1 in patch 2. The selection optimum of the second trait is the same in both patches (0.2). The third trait has an optimum at -0.3 in patch 1 and at 0.3 in patch 2. The intensity of the selection is identical for all three traits and in both patches.

A.3 Multiple traits with varying types of selection on various patch: matrix expansion

A somehow complicated behaviour is how quantiNemo extend matrix in the case of multiple quantitative traits with varying types of selection. The problem is how to specify the individual selection pressures. First, quantiNemo investigates which types of selection will be simulated based on the settings file. Then, quantiNemo sets for each simulated type of selection the corresponding parameters assuming one selection type after the other that all quantitative traits have the same type of selection: assuming for example first that all quantitative traits are under stabilizing selection, then in a second step assuming that all quantitative traits are under directional selection. This detail is important to understand since this makes it clear how a matrix is treated, *i.e.* how a matrix is expanded if needed. Of, course finally the selection pressure parameters are only set where needed, *i.e.* if the type of selection for a given quantitative trait requires the parameter. In other words, the matrix of a selection pressure has to have the number of rows of the total number of traits and not only of the traits with the corresponding selection pressure (this is controlled by quantiNemo returning a warning if not met). If multiple quantitative traits are simulated with varying selection pressures it is handy to use the row indicator for the rows of the matrix. Example:

```
patch_number 3
quanti_nb_trait 5
quanti_selection_model 0
quanti_selection_model_3 1
patch_stab_sel_optima      {{1: 1 2 3}{2: 2 3 4}}
patch_dir_sel_growth_rate {{3: 4 5 6}{4: 7 8 7}{5: 2 3 2}}
```

In this example the first two quantitative traits are under stabilizing selection, whereas the three last quantitative traits are under directional selection. Using the row indicator it is possible to set the selection pressure directly for the required quantitative trait. However, the following parameterization is equivalent to the upper one:

```
patch_number 3
quanti_nb_trait 5
quanti_selection_model 0
quanti_selection_model_3 1
```

```
patch_stab_sel_optima      {{1 2 3}{2 3 4}{9 9 9}{9 9 9}{9 9 9}}
patch_dir_sel_growth_rate {{9 9 9}{9 9 9}{4 5 6}{7 8 7}{2 3 2}}
```

In the example above the rows consisting of the number nine are read but then not taken into account, since the rows do not correspond to the correct quantitative traits. Caution if you are using matrix expansions since this may lead to unwanted configurations as shown below:

```
patch_number 3
quanti_nb_trait 5
quanti_selection_model 0
quanti_selection_model_3 1
patch_stab_sel_optima      {{1 2 3}}
patch_dir_sel_growth_rate {{4 5 6}{2 3 7}{2 8 3}}
```

In this example the growth rates are as follows:

```
1. trait: stabilizing selection optima: {1 2 3}
2. trait: stabilizing selection optima: {1 2 3}
3. trait: directional selection growth rate: {2 8 3}
4. trait: directional selection growth rate: {4 5 6}
5. trait: directional selection growth rate: {2 3 7}
```

Maybe this behavior was desired, but it could also well be that one anticipated the following specification which is wrong:

```
1. trait: stabilizing selection optima: {1 2 3}
2. trait: stabilizing selection optima: {1 2 3}
3. trait: directional selection growth rate: {4 5 6}
4. trait: directional selection growth rate: {2 3 7}
5. trait: directional selection growth rate: {2 8 3}
```

Why is this not the case? Assume that all quantitative traits are under directional selection. Thus the matrix of the growth rate has to be repeated leading to the following full matrix: {{4 5 6}{2 3 7}{2 8 3}{4 5 6}{2 3 7}}. This matrix expansion leads to a warning indicating that the number of rows is not an entire subset of the number of quantitative traits. Based on this matrix it is now obvious that the growth rate of the third quantitative trait (thus the first trait under directional selection) has the growth rates {2 8 3} and not {4 5 6}.

A.4 Selection pressure definition

If the selection varies among patches, but not among traits (or a single trait is defined), it might be easier to specify it with the following parameter. In this case, all the parameter related to selection change from **quanti_*** to **patch_***. This is also how QuantiNemo 1 used to work.

selection_pressure_definition [0;1] (default: 1)
This parameter specifies how the selection pressure is defined.

- **0 : patch.** The selection pressure is defined at the patch level. The advantage is that across quantitative traits the matrix expansion may be used.
- **1 : quanti.** The selection pressure is defined a the quantitative trait level. The advantage is that the parameters may be extended by the postix "_1", "_2". This allows to define the selection pressure separately for each quantitative trait or to group the selection pressure among groups.

A.5 Simulating sexual chromosome

Quantinemo does not come with a direct parameter to have a sexual chromosome but gives the possibility for a trait to code for the sex of individuals. As we will see it, this allows to simulate classical sexual chromosome, preserving the freedom to simulate other more complex scenarios.

For example, the XY and WZ model can be simulated easily, but we can also simulate systems where the environment plays a role in the sex determination or with more than two allele coding for the sex. Moreover, one can also simulate situations where the trait is not located on a single locus but on various locus on different chromosomes.

A.5.1 Main idea

To simulate sexual chromosomes, the main idea is the following. The sex of all individual should be determined by one gene, in Qn language a quantitative trait on a single locus. This gene should have two alleles, one making

individuals to be male, and another one female. This locus can be located on the genetic map, making the chromosome which carries it the so called*sexual chromosome*. In the case of the XY model, a chromosome carrying an allele which makes you male would be the Y chromosome, while a chromosome with the other allele would be called the X chromosome.

In order to set one gene to be sex-determining, we need to set the parameter sex_ratio_threshold. If this parameter is set, the sex of individuals is determined by the phenotypic value of the first quantitative trait instead of randomly (as it is usually the case). More precisely, this parameter gives the threshold above which an individual becomes a male. Typically, it will be set to zero, and individuals with a positive genotype will be male while individuals with a negative genotype will be female. For the XY system, we can fine-tune the allelic value so that XX individual have a negative genotype and XY individual a positive one.

A.5.2 Example: XY system

In this section, we explicit and detail the necessary step in order to simulate a minimalistic sexual chromosome of the XY system type.

1. Set a dioecious mating system
2. Set the parameter sex_ratio_threshold to 0
3. Have the first quantitative trait to be determined by a single diallelic locus
4. Locate this locus on a given chromosome
5. Set the value of the allele so that two X leads to a negative genotype and XY leads to a positive genotype

In the following input file, we specify all these values. Notice that to make our example easy to adapt, we also added a second trait coded by a single locus placed on the sexual chromosome but with more possible alleles. This second trait could then be under any type of selection:

input file

```
patch_capacity       100    # number of individuals
generations          100    # number of generation
sex_ratio_threshold  0      # male have positive phenotype
                            # female have negative genotype
quanti_nb_trait      2      # we want two traits
quanti_loci          1      # one loci per trait,
quanti_all_1         2      # two allele for x and y
quanti_all_2         255    # much more allele
                            # to simulate a continuous trait
mating_system        3      # dioecious random mating
genome             {{0 100}}# single chromosom with 2 locus
                            #  at 100 cm from each other
quanti_locus_index_1 {1}    # Sexual gene at the beginning
quanti_locus_index_2 {2}    # other quanti trait at the end
quanti_dominance_file          "quanti_dominance_file.ini"
quanti_allelic_file_1          "quanti_allelic_file.ini"
quanti_allelic_file_2          NOT_SET
recombination_factor_mal {0}# No recombination in Y
quanti_environmental_proportion 0 # phenotype is set at birth
```

quanti_allelic_file.ini

```
[FILE_INFO]{
col_locus          1
col_allele         2
col_allelic_value  3
}
#locus allele value
1       1     -1 # Allele carried by the chromosome X
1       2      1 # Allele carried by the chromosome Y
```

quanti_dominance_file.ini

```
[FILE_INFO]{
col_locus         1
col_allele1         2
col_allele2         3
col_dominance         4
}
#locus allele 1 allele2 dominance
1      1      1      1
1      1      2      1 #allele 2 is dominant, making xy male
1      2      2      1
```

Notice that here, at the first generation, some individuals will be YY, which is not the case in real populations, and we will have 75% of male and only 25% of female . This leads to a smaller effective population size. However, after one generation, the YY individuals will disappear and the sex ratio will be back to one. It is also possible to specify explicitly the genotype of all individual using an FSTAT file so that we don't have any YY individual during the first generation.

A.5.3 More complex scenario

From the example above, setting a WZ model is straightforward. The main difference would be that the dominance should be different so that heterozygote individuals are female. To include a partially determining gene, we need to add an environmental effect. To add it, we can, for example, set the value quanti_environmental_model to 0 and quanti_heritability to 1. In this way, we will have a genotype which is not fully deterministic but only partially.

To include several loci, the parameter quanti_loci should be set to more than one. The corresponding loci can then either be all independent, or we can place same on the genetic map.

Appendix B

Speeding up simulation

Individual-based simulations are time and memory consuming. While quantiNemo is a well-optimized software using C++, simulations are still time-consuming as soon as the number of loci and individuals are both larger than, let's say, 1000. In this chapter, we just want to introduce the user to some general knowledge about how simulation consumption evolves and little details that might speed-up simulation. First part is really more for people with interests and basics knowledge of optimization, while the second part is easier to understand and apply for anyone.

In general, if one wants to understand how to speed up a simulation, it's useful to understand what are the key point that can take time in quantiNemo. A few hints about that is given in the following section. Common sense will, in general, be your first ally, but might be sometimes misleading. It is in general useful play with the parameters to see which one makes a difference, and which one doesn't. For example, one can try to remove migration, mutation, selection or outputting statistics to see what is time-consuming. This can help to grasp what is time-consuming which is the first necessary step in order to improve simulation time.

B.1 general consideration

The memory consumption is almost never a problem in Quantinemo. The largest variable is by far the genotype which is stored as an array of char.

Since a char occupies 8 octets in the memory, the total amount of memory used by the genotype is about $8 \times 2 \times 2 \times n \times l$, where the first 2 comes from the fact that every individual has 2 chromosomes, and the second one because during breeding, both parents and offsprings are present. We see that even with 10'000 individuals with 10'000 loci, the memory is still reasonable, of about 4 Go, which is complete with most modern computers. For such a large number of loci and individual, the simulation would anyway be very slow, so memory is not really a problem. We, however, see that quantiNemo is not designed and could not simulate full genome with, let's say, 10^9 base because each individual would need about 32 Go.

In general, the time-consuming part of quantiNemo is building individual from their parents. In particular, to create an individual, quantiNemo has to duplicate half of the genome of both parents. This step is in general one of the most time-consuming step. Another step which is performed at the same time is to compute whether a mutation occurs at a locus. If the mutation rate is not the same among locus, then quantiNemo has to check independently for each locus and it's time-consuming. Finally, for each parent, quantiNemo has to compute the phenotype from the genotype, and this is also long if the individual has a lot of loci. All this step have in common that quantiNemo has to access and do something with all loci of all individual. It is therefore easy to see that the time needed to do this step grows linearly in terms of individuals and loci, so a simulation with $10'00$ individuals with 1000 loci is about $10 \times 10 = 100$ times longer than one with 1000 individual with 100 loci.

If only a few loci are simulated (of the order of ten or less), then the above part is not so long, and other aspects of quantiNemo might be more time-consuming. In the quantitative case, randomly selecting the parents accordingly with their fitness is a consuming task which scale as $n \log_2 n$ where n is the total number of individuals. While most part of quantiNemo scale linearly with the number of individual, this one scale in a worst way, so for very large population, this might be the bottleneck of the population. A very crude approximation would be to compare $n \log_2 n$ to $n \times l$ where n is the size of the population and l is the number of loci to see what is more time-consuming for your simulation, but the second one actually comes with an important pre-factor so you would expect selection of the parent to be longer iff $n \log_2 n \gg n \times l$.

Most of the time, the breeding is the most costly part of quantiNemo. However, if the number of patches is large (~ 1000) and the number of individuals per patch is low (~ 10), then the migration can become time-consuming. In particular, in the Island model, individuals might migrate from any patch to any patch, so there are 1000×1000 possible migration path. QuantiNemo checks for each of them if migration occurs, so this might be more time-consuming than breeding. Because breeding still involves more step than migration, we expect migration to be longer than breeding only if $N^2 \gg n \times l$ where N is the number of patches.

Finally, statistics can also be time-consuming. In general, they are faster or scale as other events. A lot of them need to access all loci of all individual, so they scale in the same way as breeding, *i.e.* as $n \times l$. It is tempting to think that statistics computed per patch or per allele are more costly than overall statistic since there are more of them, but this is not true since quantiNemo need to access anyways the information of all individual. Nevertheless, some statistics still have a different scaling. This is, in particular, the case when we compute pairwise statistic for the patch like q.adlt.fst.wc_pair. In term of scaling, we expect something like $N \times n \times l$.

B.2 How to improve simulation time

As we have seen before, one of the most important quantity is the total number of loci in the population, *i.e.* the number of loci per individual times the number of individuals. Decreasing this quantity would, therefore, improve computation time. In general, these parameters are fixed by external constrains. However, sometimes tricks can be used to try to cut down one of this quantity. For example, quantiNemo can have up to 256 alleles per locus. When simulating quantitative trait, increasing the number of allele per locus while decreasing the number locus will speed up simulation while keeping a reasonable diversity (not exactly the same though). Here, a compromise should be found between simulating exactly the biology and observing similar effect with different configurations.

As we have seen, writing and reading the genotype from the memory is one of the longest steps in quantiNemo. It is therefore not surprising that writing it on a hard drive (where the access is much slower than on the memory)

is very long. Therefore, outputting raw data is very costly in term of time and should be avoided when possible. A good practice here is to save it only from time to time and not at every generation. When running quantiNemo on a cluster, these data should be written when possible on a local hard drive and not on a shared drive since it can take much more time to send the data through a network.

This recommendation also stands if you compute a large number of statistics. If only a few basic stats are outputted, such as number of individuals, average fitness, etc, it should not be very time-consuming. However, if you start having loci or patch specific statistic (like n.adlt.nbAll_l or adlt.nbInd_p) with a lot of loci or patches, writing them on the hard drive is time-consuming and you should try to output them only every few generations.

Another trick to quicken the statistics event is to sample only a subpart of the Metapopulation. Most of the time, this will make no difference and is not recommended due to the extra complication that it brings. However, for some specific statistics mentioned before, this might make a difference, in particular in the case of coalescence where populations can grow really large.

Another thing that should be taken into account is that often, quantiNemo deals more easily with constant parameters than by specific parameters. For example, if the mutation rate is constant over loci, it's faster than if every locus has a different mutation rate (because in the first cast quantiNemo can randomly draw the number of mutation for the entire genome in a binomial and only pick a few loci to change, while in the latter case it has to go through every locus to check if a mutation occurs). This stand also for example for migration, where the pre-defined models are faster than when the user specifies a dispersal matrix.

At the opposite, the amplitude of these type of parameters makes in general not much of a difference. For example, It's not because the mutation rate is low, so that few mutation occurs, that the program will run more quickly. The actual number of mutation is not important. Same stand for recombination rate, migration rate, etc.

Appendix C

Glossary

In this chapter we define more precisely what is meant by some term in this manual, beyond the common knowledge about them.

- **Adults** is a status given to all individuals after the life cycle migration occurs. Even if no migration occurs, the offspring become adult at that moment of the life cycle. They die after reproducing but live long enough so that we can compute statistics.
- **The Allelic value** is the contribution to a quantitative trait from one allele. In the simple case (no dominance, epistasis, environmental effect, etc), the phenotype is simply the sum of the allelic values.
- **Coalescence simulations** are simulations where we only follow the lineages which were present at the end of the demography (last generation). It allows for very fast simulation since the individuals of the last generation generally come from only a few lineages in the past. If all lineages are exactly followed, we call it *exact coalescence*. If only a few are followed, approximations can be made which speed-up furthermore the simulation. This is what is in general meant by *coalescence*
- **Females** are individual which can only reproduce with males.
- **The fitness** of an individual indicates the likeliness that this individual reproduces. In some special cases (hermaphrodite with a fecundity of one under hard selection, ...), it actually corresponds to the average

number of offspring, but in most cases, the number of offspring also depend on other factors. The number of offspring is however always linearly proportional to the fitness so that an individual twice as fit as another one produce twice as many offsprings (on average).

- **The genotypic value** is a number representing the contribution of the genotype to the phenotypic value, i.e. to the quantitative trait.

- **Hermaphrodites** are individuals which can reproduce between each other. In general, hermaphrodite can reproduce with them self, so a single individual can colonize a new patch. It might not be the case for some special value of the mating system. In quantiNemo, females are used to simulate hermaphrodite, so parameters specific to female are also applied to hermaphrodite.

- **Individual based simulation** is a type of simulation where all individuals are actually created and each of them is described by some specific values (genotype, parents, natal patch, etc). It is much more time and memory expensive than population-based since each individual exists.

- **The life cycle** is the set of events (creation, dispersal, reproduction, death, etc.) that each individual might go through in its life. In quantiNemo, each individual goes only once through its life cycle, which means that it can only reproduce once.

- **A loci** is a position in the genome where we can have various allele. Note that one locus can either be though as one base or as a sequence of various bases. Locus should not be confused with position on the genome. Two loci can have the same position, which means that they don't recombine, but still have a different set of alleles.

- **Male** are individual which can only reproduce with female.

- **The metapopulation** is formed by all the population of various patch. Unlike population, they might be under different selection pressure

- **Offsprings** are individual before the migration was performed. They are present along with the adult just after breeding to compute statistics separately and becomes adult during migration.

- **A patch** is a physical location where a population might be. It is characterized by a capacity and is related to the others patches through the migration model. A patch is also characterized by having a specific value for the selection pressure. So for example, the temperature or the altitude of various patches can be different

- **The patch capacity** is the number of individuals that can leave ideally on a patch. Note however that depending on the population growth model and other parameters, the actual population size might be much smaller or much larger than the actual patch capacity.

- **The phenotypic value** is a number representing a quantitative trait. It can be either the actual value, but can also be a deviation from the average value in the population.

- **The population** is formed by all the individuals of a single patch. All the individuals feel the same selection pressure.

- **Population based simulation** is a type of simulation where only the number of individual on each patch is store

- **A quantitive trait** is a trait with a phenotype. It might, however, be under no selection pressure. Notice also that it can be continuous as well as discrete.

- **A trait** in quantiNemo is not exactly what is meant by a trait in a usual genetics textbook. It is more a set of loci which share common characteristics (number of alleles, mutation rate, etc). In particular, it is possible to define neutral traits which have no allelic value and no related phenotype but mutate all using the same mutation model, for example, to simulate a set of micro-satellite.

Index

(..), 16
#, 13
#/../#, 13
$, 17
\, 13
{..}, 14

all_combinations, 31

ceil, 24
coalescence, 93
coalescence_lineages_logtime, 98
coalescence_lineages_logtime2, 98
coalescence_lineages_script, 98
coalescence_model_threshold, 93
coalescence_mrca_dir, 97
coalescence_mrca_filename, 97
coalescence_mrca_script, 97
coalescence_pop_sizes_logtime, 99
coalescence_pop_sizes_of_patch, 99
coalescence_pop_sizes_script, 99
coalescence_save_lineages, 98
coalescence_save_lineages_dir, 98
coalescence_save_lineages_filename, 98
coalescence_save_mrca, 96
coalescence_save_pop_size, 99
coalescence_save_pop_sizes_dir, 99
coalescence_save_pop_sizes_filename, 99
coalescence_save_tree, 95
coalescence_tree_dir, 96
coalescence_tree_filename, 96
coalescence_tree_script, 96

dispersal_border_model, 47
dispersal_lattice_dims, 48
dispersal_lattice_range, 47
dispersal_model, 46
dispersal_propagule_prob, 48
dispersal_rate, 46
dispersal_rate_fem, 46
dispersal_rate_mal, 46
dispersal_rate_model, 49
divergence_pop_size, 94
divergence_time, 94

equation, 24
extinction_rate, 39
extinction_rate_survival, 39
extinction_rate_survival_fem, 39
extinction_rate_survival_mal, 39

filename, 29
floor, 24
folder, 29

generations, 40
genome, 61
genome_fem, 61
genome_mal, 61
growth_rate, 45

logfile, 29
logfile_type, 30

mating_males, 36
mating_nb_offspring_model, 42
mating_proportion, 36
mating_system, 34
mean_fecundity, 45

ntrl_all, 51
ntrl_allelic_file, 51
ntrl_genome, 62
ntrl_genome_fem, 62
ntrl_genome_mal, 62
ntrl_genot_age, 122
ntrl_genot_dir, 121
ntrl_genot_filename, 121
ntrl_genot_logtime, 121
ntrl_genot_script, 121
ntrl_genot_sex, 122
ntrl_ini_allele_model, 54
ntrl_ini_genotypes, 53
ntrl_loci, 51
ntrl_locus_index, 62
ntrl_mutation_model, 57
ntrl_mutation_rate, 55
ntrl_nb_trait, 59
ntrl_save_genotype, 119

overwrite, 29

param, 109
patch_capacity, 40
patch_capacity_fem, 40
patch_capacity_mal, 41
patch_dir_sel_growth_rate_var, 84
patch_dir_sel_max_growth_var, 84
patch_dir_sel_max_var, 84
patch_dir_sel_min_var, 83
patch_dir_sel_symmetry_var, 84
patch_ini_size, 41
patch_ini_size_fem, 41
patch_ini_size_mal, 41
patch_mean_fitness, 89
patch_number, 41
patch_sample_size, 104
patch_sample_size_fem, 104
patch_sample_size_mal, 104
patch_stab_sel_intensity_var, 81
patch_stab_sel_optima_var, 81
postexec_script, 31
preexec_script, 31

quanti_all, 51
quanti_allelic_file, 51
quanti_allelic_var, 68
quanti_coef_sel, 87
quanti_coef_sel_AA, 87
quanti_coef_sel_AA_fem, 87
quanti_coef_sel_AA_mal, 87
quanti_coef_sel_fem, 87
quanti_coef_sel_mal, 87
quanti_dir_sel_growth_rate, 83
quanti_dir_sel_growth_rate_fem, 83
quanti_dir_sel_growth_rate_mal, 83
quanti_dir_sel_max, 83
quanti_dir_sel_max_fem, 83
quanti_dir_sel_max_growth, 83
quanti_dir_sel_max_growth_fem, 83
quanti_dir_sel_max_growth_mal, 83
quanti_dir_sel_max_mal, 83
quanti_dir_sel_min, 82
quanti_dir_sel_min_fem, 82
quanti_dir_sel_min_mal, 83
quanti_dir_sel_symmetry, 83
quanti_dir_sel_symmetry_fem, 83
quanti_dir_sel_symmetry_mal, 83

quanti_dominance_file, 69
quanti_dominance_mean, 71
quanti_dominance_model, 68
quanti_dominance_var, 71
quanti_environmental_model, 74
quanti_environmental_proportion, 76
quanti_epistatic_file, 72
quanti_epistatic_var, 74
quanti_fitness_factor_heterozygote, 91
quanti_fitness_factor_homozygote, 91
quanti_fitness_landscape, 86
quanti_fitness_landscape_fem, 86
quanti_fitness_landscape_mal, 86
quanti_geno_value_age, 124
quanti_geno_value_dir, 123
quanti_geno_value_filename, 123
quanti_geno_value_logtime, 124
quanti_geno_value_script, 124
quanti_geno_value_sex, 124
quanti_genome, 62
quanti_genome_fem, 62
quanti_genome_mal, 62
quanti_genot_age, 122
quanti_genot_dir, 121
quanti_genot_filename, 121
quanti_genot_logtime, 121
quanti_genot_script, 121
quanti_genot_sex, 122
quanti_heritability, 75
quanti_ini_allele_model, 54
quanti_ini_genotypes, 53
quanti_loci, 51
quanti_locus_index, 62
quanti_mutation_model, 55
quanti_mutation_rate, 55
quanti_nb_trait, 59
quanti_output, 127
quanti_phenot_dir, 126
quanti_phenot_filename, 126
quanti_phenot_logtime, 126
quanti_phenot_script, 126
quanti_phenot_sex, 126
quanti_phenotype_landscape, 85
quanti_phenotype_landscape_fem, 85
quanti_phenotype_landscape_mal, 86
quanti_save_geno_value, 122
quanti_save_genotype, 119
quanti_save_phenotype, 125
quanti_selection_model, 78
quanti_stab_sel_intensity, 81
quanti_stab_sel_intensity_fem, 81
quanti_stab_sel_intensity_mal, 81
quanti_stab_sel_optima, 80
quanti_stab_sel_optima_fem, 80
quanti_stab_sel_optima_mal, 80
quanti_va_model, 76

random_per_replicate, 30
rbeta, 22
rbinom, 23
recombination_factor, 63
recombination_factor_fem, 63
recombination_factor_mal, 63
regulation_model_adults, 38
regulation_model_offspring, 37
rep, 20
replicates, 28
rgamma, 22
rlnorm, 21, 22
rnorm, 21
round, 23
rpois, 23
rsample, 23
runif, 21

sample_all_or_nothing, 100

sampled_patches, 104
seed, 30
selection_level, 88
selection_position, 89
seq, 19
seq2D, 20
seq2Db, 20
set, 17
sex_ratio, 36
sex_ratio_threshold, 37
stat, 106
stat_dir, 108
stat_filename, 108
stat_log_time, 108
stat_NaN, 109
stat_save, 107

trunc, 24

working_directory, 28