msABC: A modification of Hudson’s ms to facilitate multi-locus ABC analysis

User's Guide & Manual

contact details:

Pavlos Pavlidis: pavlidis@bio.lmu.de

Stefan Laurent: laurent@bio.lmu.de

Table of Contents

msABC: A modification of Hudson’s ms to facilitate multi-locus ABC analysis....1
User's Guide & Manual 1
- Complilation and demostration....3
  - Compilation....3
  - Demonstration....3
- Introduction....4
- Methods....4
  - Simulation outline....4
  - Prior distributions....4
    - Implemented distributions....4
    - Further distribution implementation....5
    - Including priors in the command line....5
    - Coalescent parameters....5
      - Scaled mutation rate θ 5
      - Scaled recombination rate ρ 6
      - Past demographic events: 6
      - Migration rate....6
      - Specifying the order of past demographic events....7
  - Simulating multiple loci....9
    - The locfile....9
    - Then, the configuration of the samples is provided in the command line of msABC after the -I flag (see also the manual of ms). In this case it would be -I 2 10 15, which means that there are two populations and the sample sizes are 10 and 15 respectively. Thus, this information is consistent with the locfile. 10
      - How msABC calculates the sample configuration....10
    - The command line related to the simulation of multiple fragments....11
  - Simulating incomplete information (missing data)....12
    - Report file of missing data....13
      - Creation....13
      - Contents....13
  - Specifying which summary statistics to be calculated....14
  - Command line options....16
- Results - Examples....18
- Contact....18
- Acknowledgments....18
- Literature Cited....18

Complilation and demostration

The code has been tested mainly on Linux machines (mainly 32-bit, but also test was succesful in 64-bit machines) with the gcc (version 4.2.4 and 4.1.2). If you are interested in running the software on other systems please contact the authors.

Compilation

We assume that you have the file msABC.tar.gz. Run the following commands in a Linux terminal to obtain the source directory.

1. tar xvfz msABC.tar.gz (this will create the msABC directory)
2. cd msABC (enter the msABC directory).

The software package provides a compiled executable (msABC). This has been compiled in a 32-bit Ubuntu 8.04 machine using the gcc 4.2.4. It should run without problems on other 32-bit Linux machines or 64-bit Linux machines.

However if you need to recompile the code follow the steps:

1. make clean (this will remove the *.o files and the msABC executable)
2. make

Demonstration

There is the script demo.sh within the msABC directory.

This script will run 7 msABC commands demonstrating various features of the program. Each example will create a directory “exampleX”, where X=1,2,...,7. Within the “exampleX” directory you may find the files:

readme.txt: this describes each example. Please read this file first.

out.txt: this is the output of msABC and contains the parameter values and summaries for each replication of the simulation. For example, if you would like to simulate a give demographic scenario 1000 times, then out.txt will contain 1000 lines, where in each line there will be the parameter values used in the simulation and the summary statistics.

header.txt: this contains the header of the out.txt, i.e. it provides information about each column in the out.txt file. Parameters contain the prefix “p_”, whereas summary statistics contain the prefix “s_”.

tempoutput_ms.txt: this contains the polymorphic table, i.e. the standard output of the ms program.

Introduction

msABC is a program for simulating various neutral evolutionary demographic scenarios based on the software ms (Hudson 2002). msABC extends ms, calculating a multitude of summary statistics. Therefore, msABC is suitable for performing the sampling step of an Approximate Bayesian Computation analysis (ABC), under various neutral demographic models. The main advantages of msABC are (i) use of various prior distributions, such as uniform, Gaussian, log-normal, gamma, (ii) implementation of a multitude summary statistics for one or more populations, (iii) efficient implementation, which allows the analysis of hundrends of loci and chromosomes even in a single computer, (iv) extended flexibility, such as such as simulation of loci of variable size and simulation of missing data.

msABC is suitable, for analysing multi-locus data. A major assumption is that the loci are independent. Therefore, it is suitable for cases when a multitude of loci from a single chromosome or a genome are available and the loci are located far enough in order to be considered independent.

Methods

Simulation outline

The simulation process is based on Hudson's ms software. In brief, the simulation process proceeds backward in time, from the present-day sample to the most recent common ancestor of the chromosomes. The process can be described by the ancestral recombination graph and not by a tree, due to the presence of recombination within the loci. The construction of the ancestral recombination graph is simple, but in many cases unnecessarily time consuming to simulate. Two approaches contribute in more efficient algorithms. First, for each site in a aligned set of chromosomes, we follow the ancestral lines of the sample until their most recent common ancestor. Second, we track only the lines that contain ancestral material of the present day sample; these optimisations are already implemented in ms (for more details see (Hein2005) and the manual of ms at https://webshare.uchicago.edu/users/rhudson1/Public/ms.folder/msdoc.pdf?uniq=-w5jxyb.

Prior distributions

Implemented distributions

In ABC analysis, the parameters used to simulate data are drawn from specific priors or they are defined as constants. We have implemented the uniform (U), Gaussian (N), log-normal (L), and gamma (G) distributions. The uniform is directly based on the *drand48* function (http://opengroup.org/onlinepubs/007908799/xsh/drand48.html) which returns a double precision float number in [0.0,1.0]. The remaining distributions are based on the drand48 function as well. The Gaussian prior, with parameters μ and σ is implemented using the polar form of the Box-Muller transform. In detail, for every call of the Box-Muller function, two normal variates are returned, the first one is used and the second is cached and used in the next cycle. The log-normal prior is a trivial extension of the normal prior. Finally, gamma

variates with shape parameter *a* and scale parameter *k* are obtained using the algorithm by Ahrens1974 and Ahrens1982.

Further distribution implementation

We have implemented priors that often are employed in an ABC analysis. Uniform priors are the most frequently used, however they may create problems in the regression step (Beaumont et al 2002), and the posterior may assign non-zero probabilities to values of zero probability according to the prior. A further disadvantage of uniform priors is that the density function is stepwise, therefore very close parameter values may have very different probabilities. We have implemented the Gaussian, log-normal, and gamma distribution. Future implementations will extend this list depending on the feature requests.

Including priors in the command line

The command line of ms sets constant values for the parameters. In msABC, we extend the command line to include the priors. The following variables may be drawn from priors: **θ, ρ, all the time evens, and the migration rates**. The number of populations should be constant. The reason for that is that usually the number of populations (or demes) are not estimated.

When a value of a parameter is drawn from a prior, then it is applied for all loci in the genome, rescaled by the length of each locus (if necessary). For example if two loci *a* and *b* are assumed, and θ is drawn from a prior, then for the replication *i*, θ **per bp** will be the same for *a* and *b*, and it will be a random number drawn from the prior. An independent and identically distributed value will be drawn for the replication *i*+1. The general pattern for specifying a prior **'d'** (with parameters m_1, m_2) for the parameter p (e.g. θ) is the following (assuming that in the command line p is denoted by the flag -p).:

-p -d m_1 m_2

Here, m_1 and m_2 denote the parameters of the distribution 'd', e.g. the lower and the higher values of a uniform distribution.

Coalescent parameters

Scaled mutation rate θ

Using the -t flag
In ms the θ parameter is specified using the -t flag. Importantly, θ refers to a locus with infinite many positions that are allowed to experience a mutation. In msABC, θ refers also to a locus and the infinite site model applies. However, in the case that **multiple loci** are simulated, θ needs to be rescaled for each fragment according to some length measurement. For rescaling, msABC uses the *nsites* (see below about the recombination rate) , which is specified after the recombination rate. Thus if θ is specified with the -t flag then the recombination rate should be specified as well even if this is 0 (see below).

Specifying θ for each locus separately.
It is possible to specify for each fragment the mutation rate. See below (multiple loci) for more details.

Examples:
(i)ms 12 1 -t -U 10 20 : draw θ from the uniform prior U(10,20).

(ii)ms 12 1 -t 10 : θ is constant (=10).

Scaled recombination rate ρ

In ms and msABC ρ is specified by the -r flag. In a recombining locus the number of segments that may recombine are specified by the parameter *nsites.* Thus, generally, the recombination rate is specified as:

-r ρ *nsites*

A trivial value for *nsites* may be the (physical) length of the locus. However, usually a smaller value may be used, because it just denotes the number of units that can recombine. For example if a 10-kb locus is simulated then instead of *nsites* = 10000 one may use 1000, which means that 1000 recombination breakpoints are allowed within the locus.

In the case that multiple loci are simulated and θ and ρ are specified by the -t and -r flags, respectively, then *nsites* is used to rescale these two parameters (see below multiple loci).

Examples:
(i)ms 12 1 -t 10 -r -U 10 20 1000 : draw ρ from the uniform prior U(10,20).
(ii)ms 12 1 -t 10 -r 10 1000 : ρ is constant (=10).
In both cases (i) and (ii) the number of recombining segments *nsites* = 1000 and θ and ρ refer to this “length”.

Past demographic events:

Past demographic events are specified in the command line of ms using a pair of letters as flag, which consists of an ‘e’ and another letter denoting the type of the event

-eX *t x*

where X may be N (i.e. -eN), G, m (see the manual of ms for more details). *t* denotes the time that the event occurred (backwards) and *x* the change (e.g. new population size, growth rate and so on).

In msABC both of the *t* and the *x* may be drawn from priors. For example:

-eN -U 0.1 0.2 -U 0.001 0.002

,denotes that the time that the population size changed is drawn from the uniform U(0.1 0.2), whereas the ratio N_1/N (N_1 is the population size after the change) is drawn from a uniform prior U(0.001, 0.0020).

Migration rate

When multiple populations are simulated a migration matrix illustrates the gene flow between them (see the manual of ms for more details). In msABC the migration rates may be specified similarly to the examples mentioned above. For example, if the migration rate is specified within the -I flag (e.g. -I 2 n_1 n_2 *m*, where n_1 and n_2 denote the sample sizes from two populations and *m* the symmetric migration rate), then the distribution may be specified before *m* (i.e. -I 2 n_1 n_2 -U m_1 m_2) in order to draw *m* from a prior. The remaining ways to denote migration rates (e.g. using the -m, -em and so on) follows similar patterns.

The remaining past demographic events, including the specification of the migration matrix, are defined with the same way. Please see the section 'Population structure for further information about events that occur when multiple populations are simulated.

Specifying the order of past demographic events

In msABC (and in ms) the events are denoted using the -e flag. When the demographic scenario includes multiple population size changes, then (sometimes) it is necessary to assure the correct order of the events. An example is provided in figure 1.

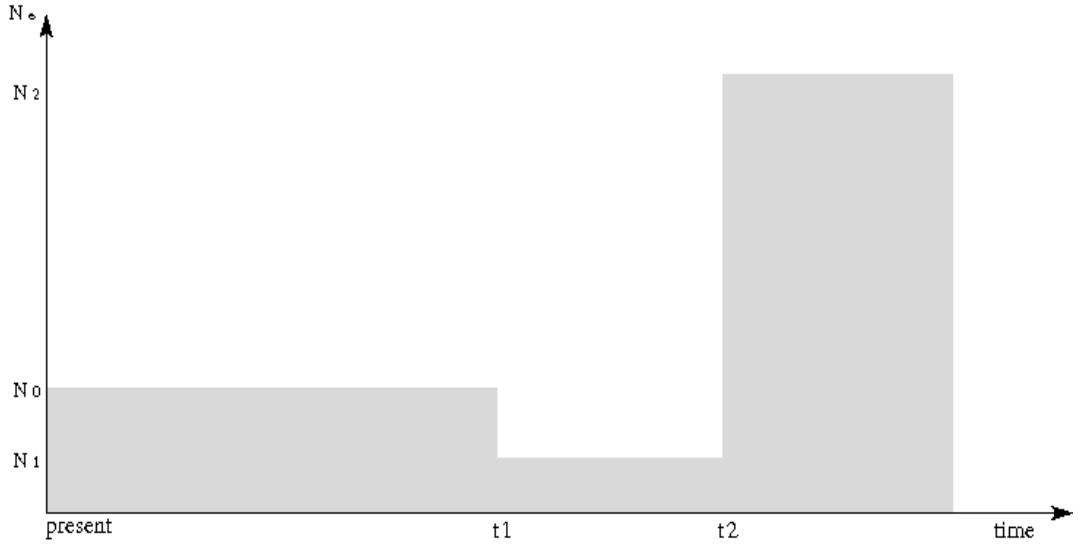

Figure 1: An example of demography where multiple population size changes have occurred. When priors are used for the times t_1 and t_2, it should be assurred that .
In figure 1, two demographic changes occur. First, at time t_1 the population contracts to N_1, and then at time t_2 it expands to N_2.

The resample method
The order of the events in Figure 1 is well-defined and we would like to assure that the order is correct in simulations. However, if t_1 and t_2 are random variables drawn from priors then it is possible that $t_1 > t_2$. For example, if $U_{t1}(0.4, 0.6)$ and $U_{t2}(0.5, 0.7)$, then it is possible that $t_1 > t_2$. A way to consider this would be to resample t_1 and t_2 and until $t_1 < t_2$. This is implemented in the resample mode. If the flag --resample is specified in the command line, then the order of the events is checked, considering that the correct order (backward in time) is the order of the events in the command line.

Alternatively, it may be specified --resample *x* e_1 e_2 … e_x, where *x* denotes the number of events whose order is important and e1 … ex denote the *indexes* (e.g. 1, 2, …) of those events as they are specified in the command line. This second way can be used multiple times to denote parallel constraints in the order of events when more complicated scenarios are simulated.
For example consider a model with 5 time events (1, 2, 3, 4), i.e. the -eX (where X may be N, n, G and so on) exists 4 times in the command line. The order of the events follows the order provided in the command line. If in this example the command line is:

./msABC 12 10 -t 10 -eN -U 0.1 0.5 0.01 -eN -U 0.2 0.7 0.01 -eG -U 0.3 0.6 30 -eG -U 0.4 0.8 60

and we care about the correct order within the -eN events and the -eG events but not between them, i.e. the time of the first -eN should be smaller than the time of the second -eN and the same is valid for the -eG events, then we should write:

./msABC 12 10 -t 10 -eN -U 0.1 0.5 0.01 -eN -U 0.2 0.7 0.01 -eG -U 0.3 0.6 30 -eG -U 0.4 0.8 60 --resample 2 1 2 --resample 2 3 4.

In the flag --resample 2 1 2, the first 2 means the number of events, then 1 is the first event of the command line (i.e. the -eN -U 0.1 0.5) and 2 is the second event of the command line.

The duration mode method
Alternatively, the duration mode is implemented. Here, instead of specifying the absolute time that an event occurs, the user can specify the relative time compared to the previous event using the --dur-mode flag. Thus, it is assured that the order of the events is correct. In this case the duration and not the absolute time is drawn from the prior. Under the duration mode the prior of the absolute time of the events is a sum of prior distributions that are specified in the command line. The figure 2 compares the priors from the resample and duration modes.

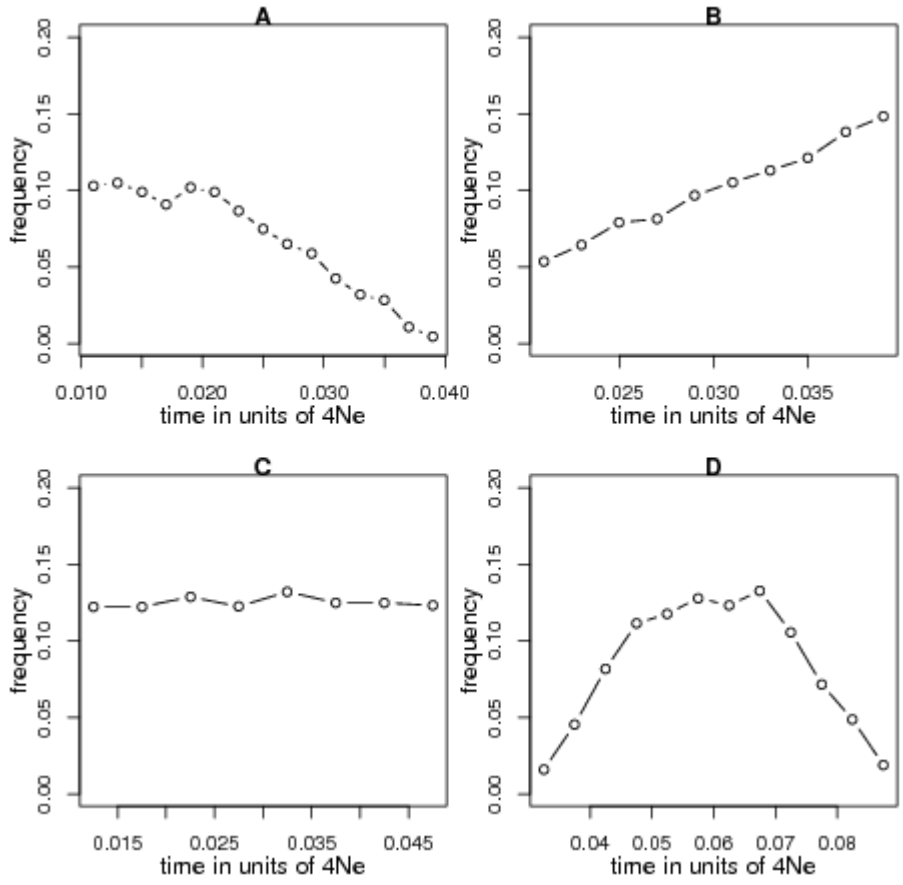

Figure : An example of priors distributions employing the resample and the duration mode. Even if in the command line the priors of the times are uniform distributions, only in C the uniform is retained. In A and B we observe a skew due to the condition $t_2 > t_1$ and in D the

distribution tends to a Gaussian as a sum of uniform distributions.

Generally, using the duration or the resample modes, results in priors that do not equal to the priors specified in the command line. This should be considered in an ABC analysis.

Simulating multiple loci

msABC enables the simulation and calculation of summary statistics from a multitude of loci, that may differ in length, sample size, mutation or recombination rate. The attributes of the loci can be specified in a file. In this manual we assume that is called “locfile”. Any name is possible.

The locfile

The locfile specifies the information about the loci. A header on the file specifies the attributes. This may include (!!! case sensitive !!!):

•**id:** it can be a number or name. It does not play any role in the analysis, just it is comfortable for the user to organize the information.

•**n:** The sample size of the fragment. It is an integer.

•**mu:** The mutation rate. It is not the scaled mutation rate θ, just the mutation probability per base pair. It may be omitted if theta is specified in the command line with the ms flag -t.

•**rec:** The recombination rate: It is not the scaled recombination rate ρ, just the recombination probability per bp. It may be omitted if the recombination rate is specified in the command line with the flag -r.

•**length:** the length of the fragment. The **length has the same meaning as the *nsites* in ms, i.e. it specifies the recombinating units on the locus.** The **length** is used for

rescaling the θ and ρ parameters if they are specified in the command line.
If for example the user specifies -t 10 -r 0 1000 and in locfile the length is 50, then msABC does the following in order to calculate the mutation and recombination rate for the fragment: 10/1000 * 50, or generally θ/nsites * length. This is the reason that nsites and length have the “units”.

Calculate **θ and ρ** if the user speficies **mu** and **rec** in the locfile. This is done by multiplying the mu*length*4N (where N is the effective population size; see below for the effective population size) or rec*length*4N.

•**pop:** It denotes the population to which the information of the “**n**” column refers. This means that for different fragments the sample size of the loci may vary. For example, assume two loci A and B, and two populations p1 and p2. For locus A the sample size from p1 is 10 and from p2 it is 20. For locus B the sample sizes are 15 and 15 respectively. Then, the columns “n” and “pop” will contain the following information in the locfile:

id	n	pop
1	10	1
1	20	2
2	15	1
2	15	2

table 1

If multiple populations are simulated and “n” is constant for all loci then under “pop” we put 0 to denote that the information refers to all the populations. For example, if for

both loci A and B the sample sizes are 10 and 15 for p1 and p2, respectively then we could write:

id	n	pop
1	25	0
2	25	0

table 2

Then, the configuration of the samples is provided in the command line of msABC after the -I flag (see also the manual of ms). In this case it would be -I 2 10 15, which means that there are two populations and the sample sizes are 10 and 15 respectively. Thus, this information is consistent with the locfile.

Alternatively, it would be also possible to write:

id	n	pop
1	10	1
1	15	2
2	10	1
2	15	2

table 3

How msABC calculates the sample configuration

When ms (or msABC) performs simulations from samples of multiple populations, it **needs the configuration of the samples,** i.e. how many individuals each samples contains and **it simulates the coalescent for the total sample (structured coalescent).** Thus, in the scenario of the table:

id	n	pop
1	10	1
1	20	2
2	15	1
2	15	2

when msABC reads the first column after the header:

1.it reads that pop != 0. This means that information for be different for various loci. What is important is the total sample size and the configuration. The total sample size is important because of the memory allocation. Thus, given that pop != 0, msABC does not know the total sample size and the configuration.

2.In order to retrieve this information it reads the lines **until pop == *total number of populations.*** This means that **without performing any simulation,** it reads the next line “1 20 2”. Now, since pop==2, msABC has all the information needed. The sample configuration is 10, 20 and the total sample is 30. Thus it simulates locus A.

Then it reads the next line “2 15 1”: again pop == 1, which means that for this locus there is some “special” information. It stores number “15” and reads the next line until to get the line with pop==2. Thus after reading the line “2 15 2”, msABC knows that the configuration is 15, 15 and the total sample size is 30 and simulates the second fragment.

3.Thus it is important:

when information about configuration varies among loci, the information for a locus should

be given strictly for pop 1, pop 2 and so on until the maximum number of populations.

Loci should not be shuffled. Information should be provided first for locus 1 then for locus 2 and so on.

In the case of table:

id	n	pop
1	25	0
2	25	0

msABC retrieves all the information from the command line, and specifically from the flag -I. In this example, it was “-I 2 10 15”. The information under “n” (25) should agree with the command line (10 + 15).

In the case of the third table

id	n	pop
1	10	1
1	20	2
2	15	1
2	15	2

msABC will act as in the case of the first table.

Finally, in the more complicated scenario:

id	n	pop
1	10	1
1	20	2
2	15	1
2	15	2
3	25	0

table 4

and command line information: -I 2 10 15, msABC will read the first line in locfile, and then immediately the second one. The sample configuration is provided by the locfile since pop != 0. The msABC simulates a sample of size 30 from two populations with configuration 10, 20. Similar, is the information for the second locus (configuration is 15, 15). For the locus 3, msABC reads from locfile pop == 0. That means that the information is provided in the command line i.e. -I 2 10 15. Thus, the configuration 10, 15 is used for the simulation of the third locus.

The command line related to the simulation of multiple fragments

•The attributes of multiple loci are specified in the command line within the flags: -frag-begin, -frag-end. All the flags that refer to the loci should be specified within these two flags.

•--finp <string>: the flag -finp is used to read the locfile.
For example: -finp locfile.txt

•-N <int>: the effective population size. This is used to calculate θ and ρ when mu and rec

are given in the locfile.

Simulating incomplete information (missing data)

Often, in demography inference problems observations contain incomplete information. For example consider the following alignment which is a part of a study on the polymorphism patterns of *Arabidopsis thaliana* by Nordborg et al (2005).

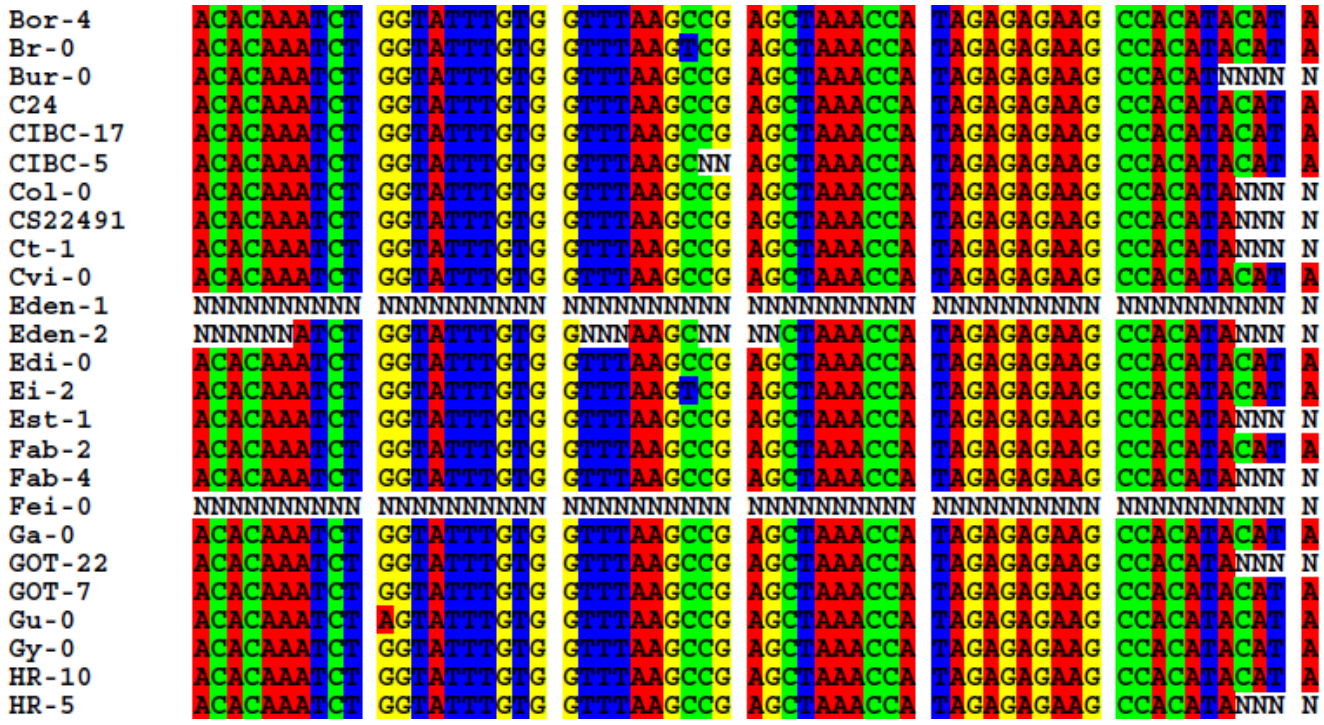

Figure 3: example of a dataset with missing information. The four DNA bases A,C,G,T are represented by red, green, yellow and green color, respectively. Missing information is represented by “N”.

In order to infer the demographic scenario from this dataset it would be necessary to handle missing data. Thus, it would be necessary to remove lines Eden – 1 and Fei – 0, and to discard the columns that contain missing information. There are certain disadvantages with this approach.

1.A lot of information is discarded. For example, analyzing the dataset of the **Drosophila Population Genomics Project (DPGP)**, it would be necessary to discard about half of the total sites of the dataset (1.8 million out of 3.7 million sites), if any site with at least one “N” would have to be discarded.

2.Simulation programs (like ms) cannot generate sites with missing information. Therefore, in an ABC analysis the observed dataset would be different than the simulated dataset.

3.The calculation of summary statistics becomes problematic.

In msABC the following ideas remedy these problems.

1.A perl script (missing_ms_report.pl) reads the observed dataset (e.g. the dataset shown in figure 1) and creates a report for the missing information (see below for details on the report).

2.msABC reads the report file and thus it knows the coordinates (i.e. sequences and positions) of all “N”s.

3.msABC simulates data, but then it replaces any 0 or 1 (which is the output of ms and msABC) which is located in the coordinates of missing information by the missing symbol. This symbol can be anything except 0 or 1, and in msABC it is 2.

4.It adapts the summary statistics calculations in order to take into account missing data. In principle it modifies the sample size of each site. Thus, for the calculation of π it takes into account that the sample size of each site may vary. It should be noted here that the denominators of θ_W, Fay and Wu's H, and Tajima's D use the maximum sample size. Thus, the (absolute) value of those statistics may be meaningless, but since the missing data distribution is equal in observed and simulated data, their comparison is meaningful.

If the report file for the missing information is named “misfile”, then a file should be created e.g. missing.list which contains the name “misfile”. Then, in the command line we write: --missing missing.list (dash dash missing). If multiple loci are simulated, and each locus is associated with its own “misfile” then the missing.list contains the names of all the “missfile”s.

Report file of missing data

Creation

The report file is created by using the perl program missing_ms_report.pl in=<string>, where <string> is a user-defined file name which contains the fasta sequences. The output will be shown in the stdout, so it can be re-directed to a file using “>”, e.g. missing_ms_report.pl in=seq.FA > misfile

Contents

The information within misfile is organized in 3 columns. The first column represents the sequence number, e.g. 0, 1, …. The second and the third column represent the beginning and the end of missing data as float numbers between 0 and 1. For example:

0	0.00161030595813205	0.00483091787439614
0	0.00322061191626409	0.00644122383252818
0	0.00483091787439614	0.00805152979066022
0	0.00644122383252818	0.00966183574879227

This means that when the first sequence (sequence 0) is simulated missing data the positions in the intervals (0.00161030595813205, 0.00483091787439614), (0.00322061191626409, 0.00644122383252818) and so on should be missing data.

The rationale behind this way of organizing the information in misfile is the following:

Observed data are discrete, i.e. the sequence has a finite length and every site is away from its neighboring site by 1. On the other hand, ms generates continuous data (remember the infinite site model), where each site is a decimal number between 0 and 1. Thus, the discrete information should be transformed to continuous in order to be used by msABC. Thus, if in sequence X a site is missing in position i, and the length of the alignment is L, then this information is transformed to: every position between (i-1)/L and (i+1)/L should be missing for sequence X. The extreme values (i-1)/L and (i+1)/L are not included.

Specifying which summary statistics to be calculated

The optionsfile is used to specify which summary statistics should be calculated. Thus using the flag --options <optionsfile> the summary statistics can be specified. If this option is absent all the summary statistics are calculated.

The following is a complete list of the options (statistic 1 or 0 // comment)

tajd 1 // if 1 then Tajima's 1983 D is calculated
segs 1 // if 1 then segregating sites are calculated
thetaw 1 // Theta Watterson
thetapi 1 // Pi estimation
zns 1 // The ZnS statistic (REF)
shared 1 // percentage of shared polymorphisms for subpopulations
private 1 // percentage of private polymorphisms for subpopulations
fixed 1 // percentage of fixed polymorphisms for subpopulations
fst 1 // fst calculation for all the subpopulations

pairfst <n: number of subpopulations that the pairwise fst should be calcuated>
pop_1 pop_2 … pop_n

Attention pairfst is specified in two lines above!!!

For example:

pairfst 3
1 3 5

this means that fst will be calculated for all the pairs between populations 1, 3 and 5 (i.e. 1-3, 1-5, 3-5).

If pairfst 1 is specified then all the possible pairs will be assumed.

fwh 1 // Fay and Wu H
pri-stats 1 // statistics for each subpopulation separately will be calculated. When this is active then global statistics will be calcuated as well
pri-segs 1 // segs for each subpopulation
pri-thetaw // thetaw for each subpopulation
pri-thetapi // thetapi for each subpopulation
pri-tajd 1 // tajd for each subpopulation
pri-ZnS 1 // zns for each subpopulation
pri-fwh 1 // fwh for each subpopulation

Note: if a private statistic is calculated then the global is calculated as well. If, for example,

the pri-tajd is calculated then tajd is calculated as well.

Command line options

msABC <*nsam*> <*repl*> *options,*

where *nsam* denotes the sample size and *repl* the replications (number of independent simulations).

Main options common with ms (see ms manual for more details):

-t <θ>, θ for the locus (also in ms)

-r <ρ> <nsites>, recombination rate ρ for the locus and *nsites* (also in ms)

-s <segregatin sites>, the number of segregating sites that should be simulated (also in ms)

-G <growth rate>, the exponential growth rate, negative value is allowed but then simulation may continue backward for ever (also in ms).

-I <npop> n_1 n_2 … [$4N_0$m], the sample configuration (also in ms)

-m, -ma specify the migration matrix (also in ms)

-eG <time> <growth rate>, change of the growth rate in the past

-eg <time> <pop_i> <growth rate>, change of the growth rate of the population I

-eN <time> <x>, stepwise change of population size from N to x*N

-en <time> <pop_i> <x>, stepwise change of population size of the i^{th} population from N to x*N.

-eM <time> <x>, change all the elements of migration matrix to x at time.

Other options like -ema, -es, -ej also are implemented in msABC. Please consult the manual of ms for more details.

Specifying the priors (examples)

-t -U θ1 θ2, -t -N μ σ and so on for specifying a prior for θ.

-I 2 10 20 -U 0.1 0.2: specifying a prior in the migration rate between two populations

-eg -N μ σ -U g1 g2, specifying a prior (Gaussian, with parameters μ, σ) for the time, and uniform with parameters g1, g2 for the growth rate.

-m 1 2 -U 0.1 0.2, specifying a prior U(0.1, 0.2) for the migration rate between populations 1 and 2.

Options of msABC (not in ms)

Except the priors, the following options are unique in msABC:

--missing <list with misfiles>: gets the information about missing data

--fst <Fst type>: specify the fst type of calculations. Possible values are: hbk, hsm, and slatkin, to denote the calculations of Hudson, Boos, and Kaplan (1992), Hudson, Slatkin and Maddison (1992) and Slatkin (1993), respectively.

--resample: (without any other arguments) it resamples the times of the events assuming that the correct order is the order specified in the command line.

--resample <# of events> ev1 ev2 …: resamples # of events, which are the ev1, ev2, … assuming that the order ev1, ev2, … is the correct one. For example if one writes:

msABC 12 10 -t 10 -eN -U 0.1 0.8 0.1 -eN -U 0.2 0.3 0.5 -eN -U 0.2 0.5 0.8 –resample 2 1 2 then it is necessary the events 1 and 2 to be in this order.

--bm <back mutation rate (0.0)>: it's the back mutation rate for calcuating the Fay and Wu's H. Default is 0.

--options <optionsfile>: optionsfile specifies which summary statistics should be calculated.

--verbose : print the polymorphism table and all the parameters even constant as well.

--filter: filter out singletons

--dur-mode: the duration mode (see above)

--frag-begin: beginning of the locus (fragment) section

--frag-end: end of the locus section

--finp <locusfile>: read the file with the locus information

--nf **<rows in the locusfile excluding the header>

--np **<columns in the header file>

--N <effective population size>

** these two flags are not used: I have modified the fscanf function used to read the the locus file, so that it can automatically calculate its dimensions.

Results - Examples

In the source directory of msABC please run the ./demo.sh file. It will generate 7 examples in the folders example01, ..., example07. In each of the folders there is a readme.txt file, out.txt and tempoutput_ms.txt. The readme contains information about the simulation, the out.txt contains the values of the summary statistics and the tempoutput_ms.txt contains the polymorphic data.

Contact

Please contact us at pavlidis@bio.lmu.de or laurent@bio.lmu.de for further questions, suggestions or requests.

Acknowledgments

We would like to thank Dirk Metzler for useful discussions on the implementation of the msABC. This research was supported by grants from the Volkswagen-Foundation (http://www.volkswagen-stiftung.de/) to P.P. (grant I/824234)

Literature Cited

Beaumont, M.A., Zhang, W. & Balding, D.J., 2002. Approximate Bayesian computation in population genetics. *Genetics*, 162(4), 2025-2035.

Excoffier, L., Estoup, A. & Cornuet, J., 2005. Bayesian analysis of an admixture model with mutations and arbitrarily linked markers. *Genetics*, 169(3), 1727-1738.

Fay, J.C. & Wu, C.I., 2000. Hitchhiking under positive Darwinian selection. *Genetics*, 155(3), 1405-1413.

Hey, J. & Nielsen, R., 2007. Integration within the Felsenstein equation for improved Markov chain Monte Carlo methods in population genetics. *Proceedings of the National Academy of Sciences of the United States of America*, 104(8), 2785-2790.

Hudson, R.R., 2002. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics (Oxford, England)*, 18(2), 337-338.

Hudson, R., Boos, D. & Kaplan, N., 1992. A statistical test for detecting geographic subdivision. *Mol Biol Evol*, 9(1), 138-151.

Hudson, R.R., Slatkin, M. & Maddison, W.P., 1992. Estimation of Levels of Gene Flow From DNA Sequence Data. *Genetics*, 132(2), 583-589.

Hutter, S. et al., 2007. Distinctly different sex ratios in African and European populations of Drosophila melanogaster inferred from chromosome-wide single nucleotide polymorphism data. *Genetics*, 177(1), 469-480.

Kelly, J.K., 1997. A test of neutrality based on interlocus associations. *Genetics*, 146(3), 1197-1206.

Kuhner, M.K., 2006. LAMARC 2.0: maximum likelihood and Bayesian estimation of population parameters. *Bioinformatics (Oxford, England)*, 22(6), 768-770.

Laval, G. & Excoffier, L., 2004. SIMCOAL 2.0: a program to simulate genomic diversity over large recombining regions in a subdivided population with a complex history. *Bioinformatics (Oxford, England)*, 20(15), 2485-2487.

Ross-Ibarra, J. et al. 2008. Patterns of Polymorphism and Demographic History in Natural Populations of Arabidopsis lyrata . PLoS ONE 3(6): e2411

Nordborg, M. et al., 2005. The pattern of polymorphism in Arabidopsis thaliana. *PLoS Biology*, 3(7), e196.

Ometto, L. et al., 2005. Inferring the effects of demography and selection on Drosophila melanogaster populations from a chromosome-wide scan of DNA variation. *Molecular Biology and Evolution*, 22(10), 2119-2130.

Slatkin, M., 1993. Isolation by Distance in Equilibrium and Non-Equilibrium Populations. *Evolution*, 47(1), 264-279.

Tajima, F., 1983. Evolutionary relationship of DNA sequences in finite populations. *Genetics*, 105(2), 437-460.

Tajima, F., 1989. Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics*, 123(3), 585-595.

Thornton, K., 2003. Libsequence: a C++ class library for evolutionary genetic analysis. *Bioinformatics (Oxford, England)*, 19(17), 2325-2327.

Watterson, G.A., 1975. On the number of segregating sites in genetical models without recombination. *Theoretical Population Biology*, 7(2), 256-276.