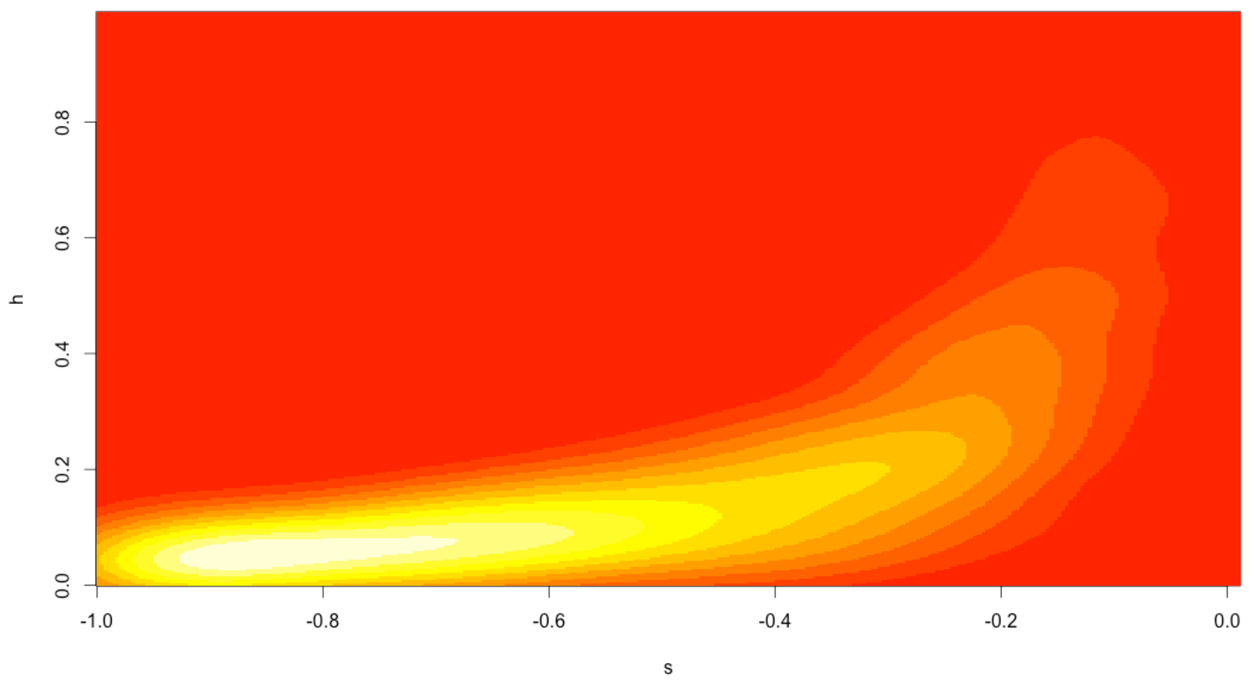
WFABC manual

Wright
Fisher
Approximate
Bayesian
Computation

Matthieu Foll
School of Life Sciences
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
matthieu.foll@epfl.ch

Last update: 15.01.2014



Warning: before using the software, read the two papers describing the underlying model in details: Foll *et al.* (2014) and Foll *et al.* (submitted).

1. Introduction

This program, WFABC estimates parameters of a Wright-Fisher model (http://en.wikipedia.org/wiki/Genetic_drift#Wright.E2.80.93Fisher_model) with selection using allele frequencies measured at different time points and possibly different loci. We consider a gene with two alleles A and B respectively at frequency p and 1-p. For haploids, the fitness of A and B are respectively w_A=1+s and w_B=1. For diploids, the relative fitness of the genotypes are given by:

AA	AB	BB
w	w	w
1+s	1+sh	1

For s>0, the A allele is advantageous, while for s<0 the A allele is deleterious. h defines the dominance effect. For h=1, the A allele is dominant, while for h=0 it is recessive. By default WFABC assumes h=0.5 (codominant), but see below.

We assume a constant effective population size N_e counted as the number of chromosomes (to be more easily constant with the haploid case). This means that for diploids the effective number of individuals is N_e/2.

In this model, the frequency p at generation t+1 is drawn randomly from a binomial distribution with parameters N_e, and x, where x is defined as:

$$x = \frac{w_{AA}p^2 + w_{AB}p(1-p)}{w_{AA}p^2 + w_{AB}2p(1-p) + w_{BB}(1-p)^2} \text{ for diploids}$$

$$x = \frac{w_A p}{w_A p + w_B(1-p)} \text{ for haploids}$$

The idea behind WFABC is that simulating this model is very easy and efficient, while computing a likelihood is quite hard, requires some approximations that may be unrealistic, and is computationally intensive. Being an Approximate Bayesian Computation method, WFABC replaces the likelihood calculation with simulations.

The aim of WFABC is to estimate N_e and the selection coefficients s (in some cases also h for diploids) at every locus.

2. Input file

Here is an example of the first lines of the input file “multiple_loci.txt” (this is a simulated dataset, see below).

```
1000 12
1,10,19,28,37,46,55,64,73,82,91,100,
100,100,100,100,100,100,100,100,100,100,100,100,
48,55,59,59,66,52,52,57,54,49,58,69,
100,100,100,100,100,100,100,100,100,100,100,100,
53,50,45,39,38,42,39,50,45,60,55,59,
100,100,100,100,100,100,100,100,100,100,100,100,
46,51,53,41,35,38,36,34,43,38,39,55,
...
```

In the first line you find the number of loci (1000) and the number of time points (12).

The second line indicates the time of the 12 points. The important thing here is that the difference between the values corresponds to the number of generations between the time points. In this example the first time point is labeled as generation “1”, and the second time point is 9 generations later (“10”). Adding or subtracting a constant to all the values won’t change the results. They can also be negative.

The rest of the file has to contain 1000 pairs of lines corresponding to the 1000 loci. For each locus the first line corresponds to the sample size at each time point (in number of chromosomes, so twice the number of individuals for diploids), and the second line to the number of A alleles at each time point. In this example all sample sizes are 100, and we only show the first 3 loci.

In this example we use a comma (“,”) as a separator for time points, but note that it can be any non-numeric character (but a single character), and that the final comma at the end of each line is optional.

3. WFABC basic usage

WFABC is a command line software actually made of two executables corresponding to the two steps of the method (see the associated references):

1. wfabc_1 estimates N_e and calculates the summary statistics for the data
2. wfabc_2 uses the estimated N_e from step 1 to estimate s (and eventually h) at every locus

We provide executables for Mac OS X, Windows and Linux. The C++ source code along with a “makefile” are provided under a GNU General Public License as published by the Free Software Foundation. If you don’t need to compile WFABC yourself you can jump to the next point (3.1).

WFABC depends on the following libraries:

- OMPTL (tech.unige.ch/omptl/): OpenMP Multi-Threaded Template Library. We use the parallel “sort” function.
- Anyoption (www.hackorama.com/anyoption/): a C/C++ Command line and resource file option parsing.
- Agner Fog’s (www.agner.org/random/) random number generator libraries.

Fortunately, all these library are simply C++ headers and sources. They are provided in the source directory and you won’t need to compile or install anything. Simply typing “make” should work on most configurations.
WFABC uses OpenMP to perform parallel calculation on multicore CPUs, and you need to use a compiler supporting OpenMP. You can also remove the “-fopenmp” option in the “Makefile” and the compiler will ignore all the OpenMP directives (but the program will be much slower…). The gcc compiler provided in Mac OS X 10.7 and later with Xcode (gcc-4.2) does not fully support OpenMP, and will lead to an error at runtime when using more than one thread ("Abort trap: 6"). You will need to install an alternative gcc compiler as the one provided at http://hpc.sourceforge.net, and point to it in the “Makefile” at line “CC=“. For Windows, I recommend using the gcc version available at http://www.equation.com.

3.1 Worked example 1: simulated multi-locus data

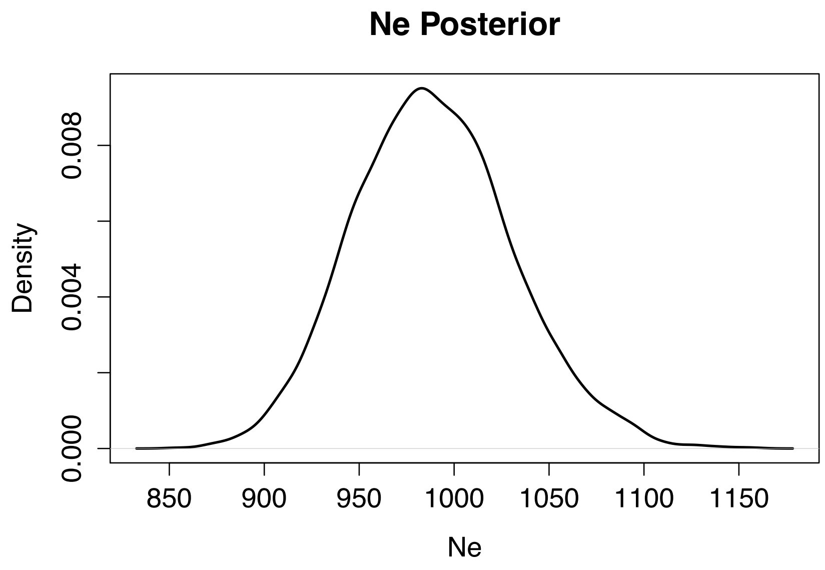
Using the example input file “multiple_loci.txt” presented above, the first step is to launch:

```
./wfabc_1 multiple_loci.txt
```

This creates two output files:

- “multiple_loci_obs_stats.txt” containing the observed statistics for each locus (this will be used in the second step).
- “multiple_loci_Ne_bootstrap.txt” containing samples from the posterior distribution of N_e (one value per line). It will also be used in the second step, but you can already plot the posterior. Using R for example:

```
> post_N=read.table("multiple_loci_Ne_bootstrap.txt")
> plot(density(post_N[,1]),lwd=2,main="Ne Posterior",xlab="Ne")
```



In this simulated data the true value is N_e=1000. The program also writes out the mean and standard deviation of the posterior for N_e (in this example you should see something similar to "987.805 41.6943").

This first step is quite fast and it takes around one second for this example containing 1000 loci with 12 time points. In this simulated data, loci from 1 to 990 are neutral (s=0) and loci from 991 to 1000 are under selection with s=0.1. This is important as the first step assumes that most of the loci are neutral. If if assumption is not met, one can still use WFABC without estimating N_e in the first step (see the second worked example below).

To run the second step and estimate selection coefficients on the same example one can use:

```
./wfabc_2 -min_s -0.5 -max_s 0.5 multiple_loci.txt
```

The option "-min_s -0.5 -max_s 0.5" sets the prior distribution for s as a uniform distribution between -0.5 and 0.5. The program will look for the two files created in the first step, so you necessarily need to run it first. By default it assumes a diploid model with h=0.5.

This step takes a bit longer (around two minutes for this example with 1000 loci), and the progress is indicated as:

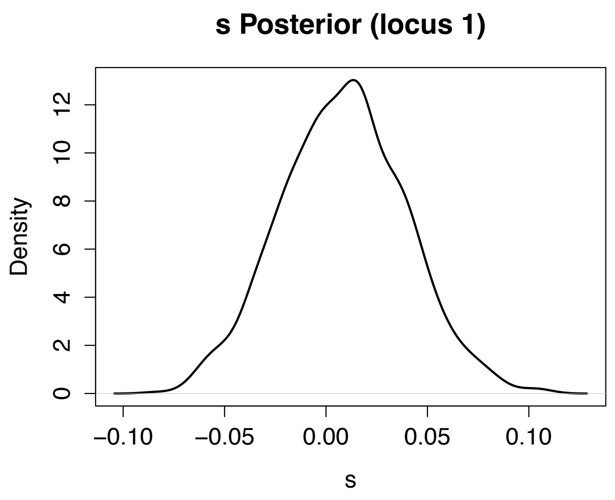
```
Locus 1
Locus 2
Locus 3
…
Locus 1000
```

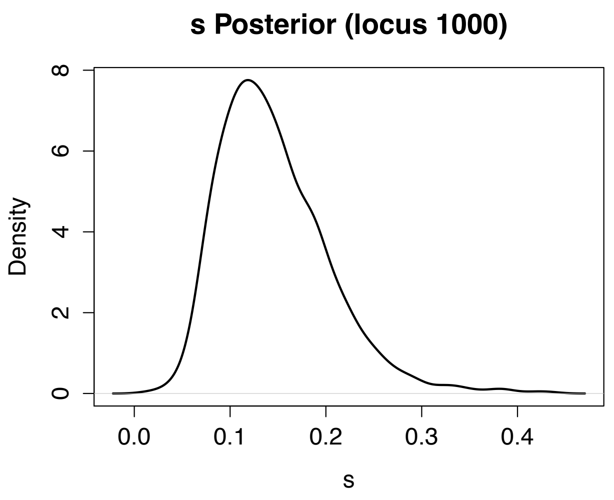
This creates two output files with the same structure:

- "multiple_loci_posterior_N.txt": one line per locus (1000) here. Each line contains a sample from the posterior distribution of N_e.
- "multiple_loci_posterior_s.txt": same for s.

Note that the order of the samples match between the two files, so one can plot a joint posterior density distribution (see below).
From the file "multiple_loci_posterior_s.txt" one can easily plot posterior distributions or obtain point estimate and confidence intervals using R. To plot the posterior for s for the first locus and the 1000th locus:

```
> post_s=read.table("multiple_loci_posterior_s.txt")
> plot(density(t(post_s[1,])),lwd=2,main="s Posterior (locus 1)",xlab="s")
> plot(density(t(post_s[1000,])),lwd=2,main="s Posterior (locus 1000)",xlab="s")
```



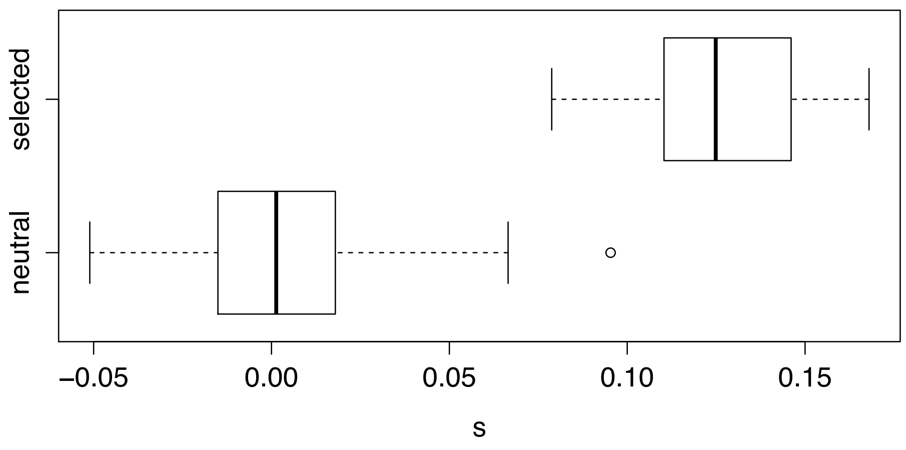


To obtain the posterior mean at all loci (we cut the output to the first three here):

```
> rowMeans(post_s)
   [1]  8.588754e-03  2.603725e-02  1.333430e-02...
```

In this simulated dataset the first 990 loci are neutral (s=0) and the last 10 are selected (s=0.1). To visualize the posterior mean using two boxplots (note that 10 values is too low for a really meaningful boxplot):

```
> boxplot(rowMeans(post_s[1:990,]),rowMeans(post_s[991:1000,]),
            names=c(« neutral”,”selected”),horizontal=T,xlab="s")
```

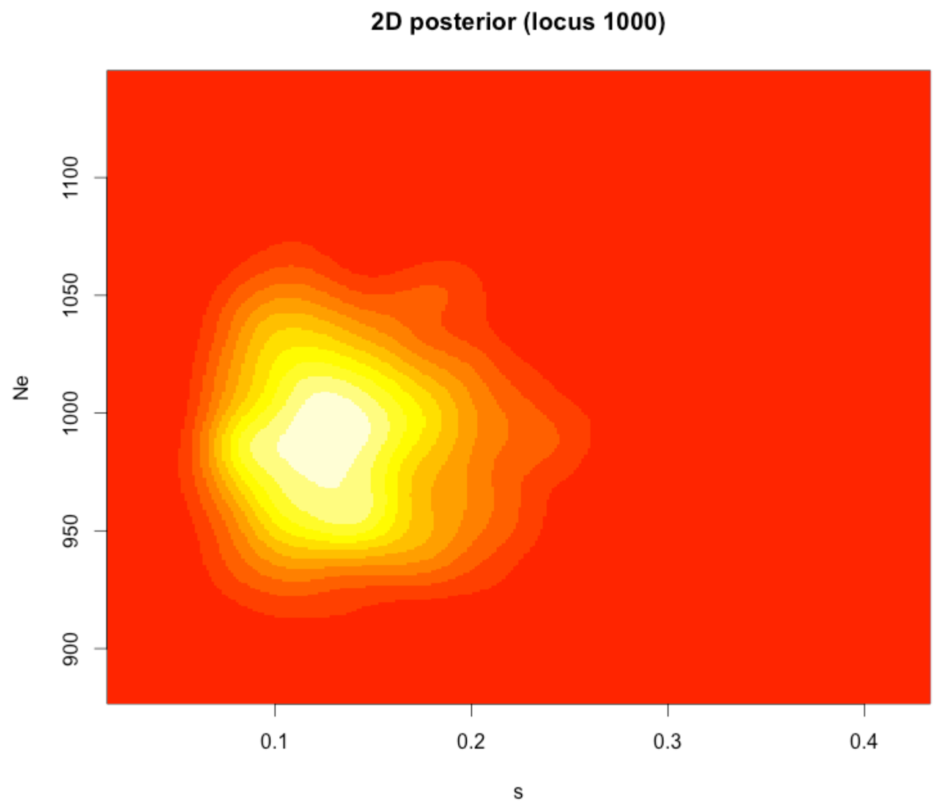


To obtain 95% highest posterior density intervals using the boa package (cran.r-project.org/web/packages/boa/) (we cut the output to the first three here):

```
> require(boa)
> apply(post_s,1,boa.hpd,1-0.95)
                   [,1]       [,2]       [,3]...
Lower Bound -0.0551124 -0.0222231 -0.0439243...
Upper Bound  0.0650899  0.0744987  0.0730273...
```

Finally, if one wants to plot a 2D posterior for s and N_e in R using the MASS package (cran.r-project.org/web/packages/MASS/):

```
> require(MASS)
> post_s=read.table("multiple_loci_posterior_s.txt")
> post_N=read.table("multiple_loci_posterior_N.txt")
> z <- kde2d(t(post_s[1000,]),t(post_N[1000,]),n=300)
> image(z,xlab="s",ylab="Ne")
```



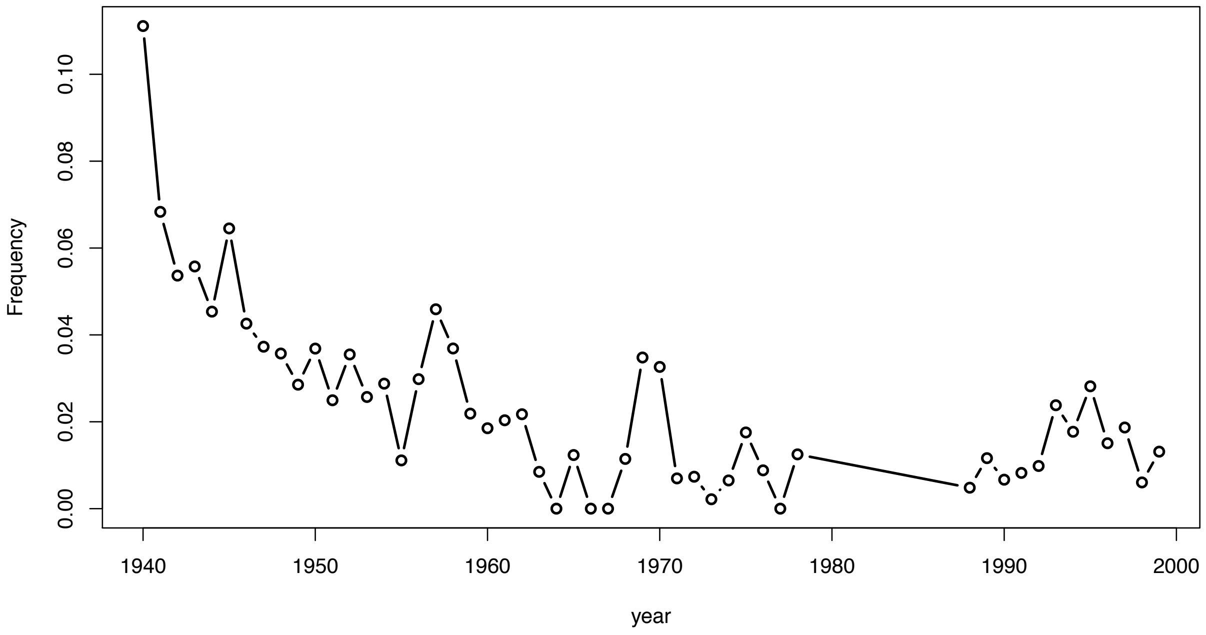
To find the 2D mode (the true simulated values are s=0.1 and N_e =1000):

```
> maxindex=which(z$z==max(z$z), arr.ind=T)
> z$x[maxindex[1]]
[1] 0.1270117
> z$y[maxindex[2]]
[1] 990.8328
```

This is generally not very useful as the correlation between s and N_e is quite low (the two parameters are more or less independents). This will be useful below for s and h.

3.2 Worked example 2: *Panaxia dominula* data

We now use the classical time-serial dataset of the *medionigra* allele in the moth *Panaxia dominula*. The data can be found in Cook and Jones (1996) and Jones (2000), and has been reanalyzed using WFABC in Foll *et al.* (submitted). It consists of a single locus and 51 time points (input file called “data_medionigra.txt”). The *medionigra* allele (denoted as A using the notations from section 1) has been previously shown to be deleterious (s<0) and the frequency trajectory looks like this:



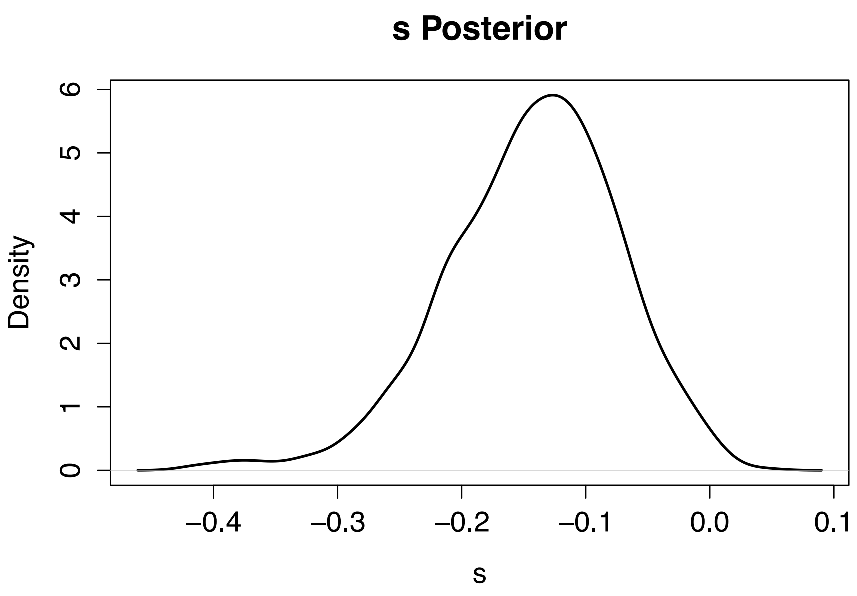
The first step of WFABC:
```
./wfabc_1 data_medionigra.txt
```

Like above, this step creates two files: “data_medionigra_obs_stats.txt” containing the observed statistics, and “data_medionigra_Ne_bootstrap.txt” containing samples from the posterior distribution for N_e. Note that the standard deviation for N_e written in this case is “nan”, and if you open the “data_medionigra_Ne_bootstrap.txt” file, you will see that all lines contain the exact same value. This is because this dataset consists of a single locus, so bootstrapping over loci does’t make sense here and you can actually disable it:
```
./wfabc_1 -nboots 0 data_medionigra.txt
```

This command only produces the “data_medionigra_obs_stats.txt” file which is needed in all cases. Note also that estimating N_e from a single locus violates the assumption that most loci are neutral in the first step, and is anyway very inaccurate. In the following step, we won’t use this estimate and manually fix N_e based on prior knowledge.

Now let’s estimate s using a uniform prior between -1 and 0.1. As we didn’t estimate N_e in the first step we need to specify it here (and the file “data_medionigra_Ne_bootstrap.txt” is not necessary in this case):
```
./wfabc_2 -fixed_N 1000 -min_s -1 -max_s 0.1 data_medionigra.txt
```

To plot the posterior in R:
```
> post_s=read.table(“data_medionigra_posterior_s.txt")
> plot(density(t(post_s[1,])),lwd=2,main="s Posterior",xlab="s")
```



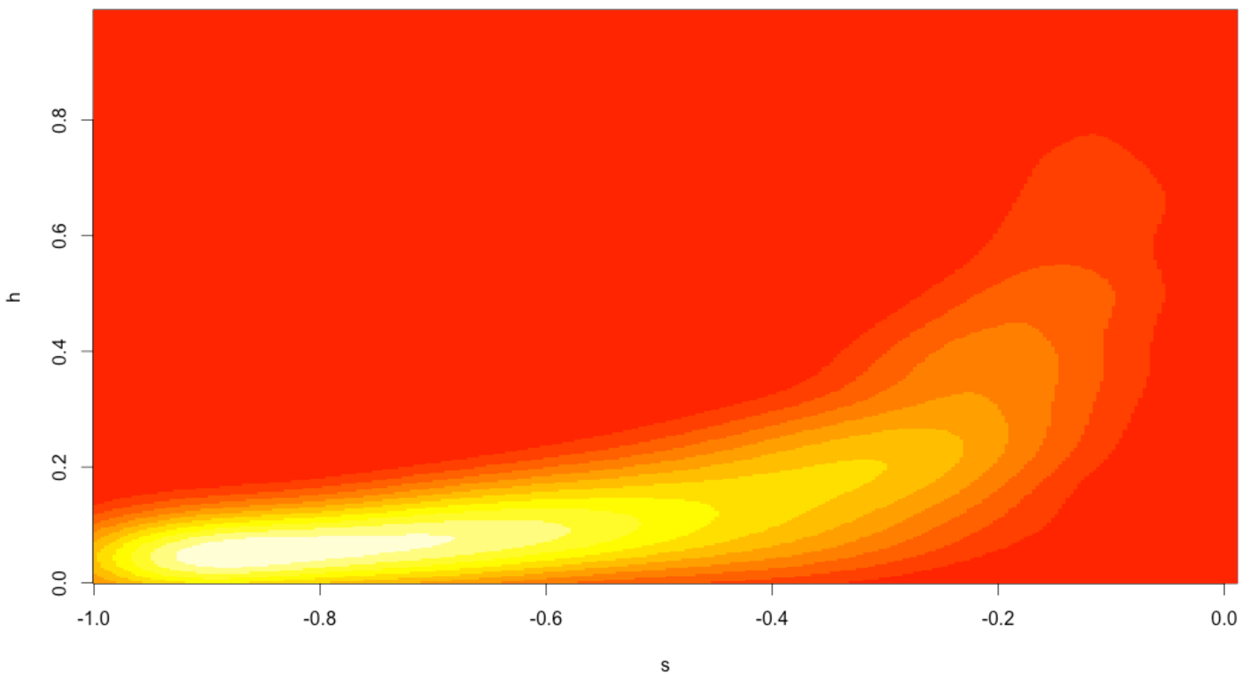
This posterior is constant with previous estimates.

As mentioned above, for diploids, WFABC by default does not attempt to estimate h and fixes it to 0.5 (codominant). You can either fix it to a different value or even try to estimate it by specifying a uniform prior for h:

```
./wfabc_2 -fixed_N 1000 -min_s -1 -max_s 0.1 -min_h 0.0 -max_h 1
data_medionigra.txt
```

Now plotting the 2D posterior in R is very interesting as h and s are strongly correlated in this case:

```
> require(MASS)
> post_s=read.table("data_medionigra_posterior_s.txt")
> post_h=read.table("data_medionigra_posterior_h.txt")
> z <- kde2d(t(post_s[1,]),t(post_h[1,]),n=300)
> image(z,xlab="s",ylab="h")
```



This graph shows that the best model to explain this data is a lethal recessive *medionigra* allele. In particular it explains better the long persistence of the *medionigra* allele over time at a low frequency (see Foll *et al.* submitted for more details). Note that the mode is close to the boundary here which is known to create problems in the kernel density estimation, and one should not directly use it to obtain the posterior mode.

We warn users that estimating h may not be possible in all cases, and in particular:

- It requires a large number of time points (note that we have 51 points in our example).
- Most of the information comes from the time the allele survives at low (<5%) and/or high (>95%) frequency.
- One should never report a point estimate without plotting the posterior or giving a confidence interval. When h can not be accurately estimated, it can easily be detected by doing so (the posterior is very flat).
- The posterior for h is generally very correlated with the posterior for s, and looking at the 2D posterior is very important. The marginal posterior in 1D can be misleading (in our example above it is almost completely flat for s over the whole prior range).

4. Full list of options

To have the full list of options, just launch the programs without argument.

4.1 wfabc_1 options

Option	Description
-nboots 10000	Number of bootstrap replicates.
-nthreads n	Number of threads (automatically detected by default).

Advanced options (change with caution):

Option	Description
-min_freq_fs 0.01	Minimum allele frequency to calculate the Fs statistics.
-iqr 3	Inter-Quartile Region to consider for a robust estimate of N exclude outliers which may bias N
-seed n	Manually set the seed (used for debugging).

4.2 wfabc_2 options

Option	Description
-nthreads n	Number of threads (automatically detected by default).
-ploidy 2	Ploidy of individuals (use 1 or 2).

ABC options:

Option	Description
-nreps 100000	Total number of simulations. Use a larger value for more accurate (but slower) results, and/or if trying to also estimate h. For each simulation a value for N taken from the file “input_name_Ne_bootstrap.txt”. Values are recycled when reaching the end of the file.
-acc_rate 0.01	Proportion of accepted simulations. The number of accepted simulations if the product of nreps and acc_rate. Try to keep at least nreps*acc_rate=1000 to have an accurate posterior estimate (and 10000 if trying to also estimate h). Using a lower acc_rate and a larger nreps will provide more accurate (but slower) results when keeping the product constant.
-fixed_N n	Set N_e=n to a fixed value (does not estimate it). The file “input_name_Ne_bootstrap.txt” is simply ignored.
-min_s x	Lower bound of a uniform prior for s.
-max_s x	Higher bound of a uniform prior for s.
-mean_s x	Mean of a Normal prior for s.
-sd_s x	Standard deviation of a Normal prior for s.
-s_list file_name	User defined s prior (one random sample per line). For each simulation a value for s is taken from file_name. Values are recycled when reaching the end of the file.
-min_h x	Lower bound of a uniform prior for h.
-max_h x	Higher bound of a uniform prior for h.

Conditioning:

The same ascertainment conditions applied to the real data have to be applied to the simulations. We propose three different non-exclusive schemes commonly used. On top of those, the value defined in by “-min_freq_fs 0.01” is also applied (at least one time point must show a minor allele frequency larger than 0.01).

Be careful as some ascertainment conditions might be so unrealistic that performing simulations can become very (infinitely) long. One example would be to require the allele frequency to reach at least a frequency of 0.9, but set the prior to only consider deleterious mutation.

Option	Description
-min_freq 0.01	At least one time point must show an allele frequency (A) larger than 0.01.
-min_freq_last x	The last time point must show an allele frequency (A) larger than 0.01.
-poly_condition	The allele is polymorphic at the last time point.
-max_sims 1	Maximum number of trials to match conditions before changing s, h and N real data, this should in general be set to one: if a site in the genome does not match your criteria, you discard it and move to the next one. This can be different for simulated data: if you simulate several replicates with a fixed value of s, you generate trajectories with always the same parameters until they match your criteria. Then you should in principle keep trying indefinitely in the ABC procedure. However, to avoid infinite loop for unrealistic combination of parameters, you should use -max_sims 100000 or something similar. A warning will indicate you when you reach the maximum.

Advanced ABC options (please use these features only if you know what you are doing):

These options can be used to create simulated data or to manually perform the ABC rejection.

Option	Description
-fixed_N_sample n	Fixed sample size (ignore those in the input file). This will apply to all loci and all time points.
-fixed_j n	Fixed initial number of mutants (ignore those in the input file).
-N_list file_name	List of values for N used to study the influence of non-constant population size. Note that you need to provide one value (and line) per generation and not per time point.
-sims file_name	Use simulations given in file_name for all loci (see wls).
-wls	Write out simulations for the last locus.

Advanced options (change with caution):

Option	Description
-min_freq_fs 0.01	Number of bootstrap replicates
-seed n	Manually set the seed (used for debbuging)

5. R scripts provided

- “multiple_loci_simulations.r”: contains the R commands presented above to create (simulation) and analyze the “multiple_loci” data.
- “WF_simulation.r”: contains the R function used in “multiple_loci_simulations.r” to simulate Wright-Fisher trajectories.
- “medionigra.r”: contains the R commands presented above to analyze the medionigra data.

6. References

Cook LM, DA Jones (1996) The medionigra Gene in the Moth Panaxia dominula: The Case for Selection. Philosophical Transactions: Biological Sciences, 351, 1623-1634.

Foll M, Y-P Poh, N Renzette, A Ferrer-Admetlla, et al. Influenza virus drug resistance: a time-sampled population genetics perspective. PLOS Genetics (in press).

Foll M, H Shim and JD Jensen WFABC: a Wright-Fisher ABC-based approach for inferring effective population sizes and selection coefficients from time-sampled data. Submitted to Molecular Ecology Resources.

Jones DA (2000) Temperatures in the cothill habitat of Panaxia (Callimorpha) dominula L. (the scarlet tiger moth). Heredity (Edinb), 84 (Pt 5), 578-586.