# `MIMARgof`, `perlgof` and `perlgofest` - programs to test the goodness of fit of the model estimated by `MIMAR`

C. Becquet

December 17, 2010

This document describes how to use the Perl scripts `perlgofest` and `perlgof`, which call the program `MIMARgof` to generate samples under an "isolation-migration" model with either point estimates of the parameters provided by `MIMAR`, or sets of parameters sampled from the posterior distribution estimated by `MIMAR` (see Figure 1 and section "List of parameters and symbols"). These scripts output the distributions of summary statistics for the simulated samples (see section "The `standard output`" ). These distributions allow one to perform a goodness of fit test. You can represent the distribution of a statistic graphically and consider visually whether the observed value of the statistic (i.e. computed for the data set used to perform the estimation by `MIMAR`) falls within the distribution. If the observed value of the statistic is within the range of the distribution, the estimated model fit the data for this statistic. Alternatively, you can quantify the probability of observing an observed statistic given the simulated distribution of the statistic. If this "p-value" is smaller than 0.05, the goodness of fit test is rejected: the estimated model does not fit the data for this statistic.

Below, I first describe how to run `MIMARgof`, which generates samples under the isolation-migration model. In a second step, I describe how to run `perlgofest`, which allows to test the goodness of fit fof an estimated model using point estimates of the parameters. Finally, I describe how to run `perlgof`, which allows to test the goodness of fit of the estimated model by sampling sets of parameters from the posterior distribution estimated by `MIMAR`. Since this goodness of fit test takes into account the uncertainties associated with the estimates, it is equivalent to the Bayesian posterior predictive p-value (e.g., Meng, 1994).

The program and scripts are intended to run on Unix, or Unix-like operating systems, such as Linux or MacOsX. The next section describes how to download the relevant files and compile the program. The subsequent sections described how to run the scripts and perform a goodness of fit test.

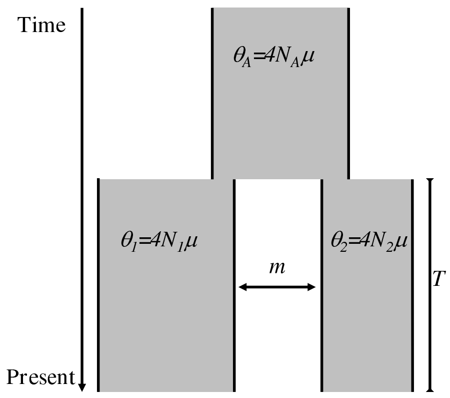

Figure 1: **The "isolation-migration" model**, in which two populations diverged $T$ generations ago from a common ancestral population. The parameters $\theta_1$, $\theta_2$ and $\theta_A$, are the population mutation rates for populations 1, 2 and the ancestral population, respectively. $\mu$ is the mutation rate per bp and $N_1$, $N_2$ and $N_A$ are the diploid effective sizes of the first, second and ancestral populations, respectively. The split time in generations is $T$, $m$ is the symmetrical migration rate between populations per generation such that, $M = 4N_1m$ is the expected number of individuals in population 2 replaced by migrants from population 1 each generation.

## Contents

## What changed since the last version?

- December 17, 2010: I changed the calculation of the p-value for $S_f$ in "TESTgod.R" as there was an error. Thanks to Nick Levsen for finding the error.

- March 18, 2010: I added objects and functions in order to specify the random seeds from the command line. The files `"mimargof.c"`, `"mimargof.h"`, `"params.c"`, `"rand1.c"`, `"rand2.c"`, `"rand1t.c"` and `"rand2t.c"` were changed.

- February 11, 2010: I added the file `"make_gametes_noanc.c"`. The program `MIMARgof_noanc` simulates polymorphism data under the isolation-migration model with parameters specified by the user and outputs the summary statistics of the polymorphism calculated assuming unknown ancestral states. A section about this option was added to `"MIMARgofdoc.pdf"`.

- December 28, 2009: The file `"params.c"` of `MIMARgof` was changed:

  - The locus-specific recombination rates are now calculated by $\rho_y = w_y \rho(Z_y - 1)$ instead of $w_y \rho(Z_y)$.
  - I added several options to obtain the locus-specific population recombination rates.
  - I changed the instances of "calloc(1..." by "malloc(...".

  I updated `"MIMARgofdoc.pdf"` accordingly and tried to clarify many points in the documentation. Note that the scaling of the recombination rate can be confusing (at least for me) so be careful when setting recombination information. Thanks to Peter Andolfatto who helped me through my confusions on getting the recombination rates right.

- November 27, 2009:

  - The file `"mimargof.c"` of `MIMARgof` was changed.: I fixed potential memory leaks.
  - The file `"streec.c"` of `MIMARgof` was changed: I fixed a bug on memory allocation that occurred when $n_{1y} + n_{2y} \gg n_{11} + n_{21}$ for any $y \in [2, Y]$. Thanks to Yongshuai Sun who mentioned the bug to me.

- March 3, 2009:

  - The file `"params.c"` of `MIMARgof` was changed: I corrected a bug on memory allocation. Thanks Susan J. Miller for mentioning this bug to me.
  - The file `"mimargof.c"` of `MIMARgof` was changed so that now when the prior on gene flow is "`1` $a$ $b$", $a$ and $b$ can both be negative and the summary output file show the estimate of the gene flow rate. Thanks to Camille Roux for highlighting this problem to me.
  - The documentation was changed. I added a figure to help calculate the summary statistics.

- May 29, 2008: The file `"params.c"` of `MIMARgof` was changed: I removed several check flags to allow greater freedom to the user. Thanks Armando Geraldes for mentioning the problem to me.

- September 18, 2007: The files `"params.c"` and `"mimargof.h"` of `MIMARgof` were changed. In the previous version, the locus names could not start with a number. Now the locus names can be any string of up to 50 characters.

## Downloading and compiling

All relevant files are included in the tar file "`mimar.tar`" available at `http://mplab.bsd.uchicago.edu/dataNprograms.htm`. Download this tar file to your machine then extract the files from the archive with: "`tar -xvf mimar.tar`". After extracting, type "`cd mimargofdir/`" and compile the program `MIMARgof` by typing:

```
gcc -o mimargof mimargof.c params.c make_gametes.c streec.c randX.c tajd.c -lm
```

($X$ is either `1`, `2`, `1t` or `2t`) or alternatively, by typing `make`, which contains this compilation line with optimization and `rand1.c`.

The choice of compilation depends on which pseudo-random number generator the user has available. "`rand1.c`" and "`rand2.c`" call `drand48()` and `drand()`, respectively. With "`rand1.c`" and "`rand2.c`", `MIMARgof` first looks for the file "`seedmimar`" to find the seed values for initializing the random number generator. If no "`seedmimar`" file is found, the generator is seeded with a default value. When the simulation is finished, the state of the random number generator is output to "`seedmimar`". In this way, each time `MIMARgof` is invoked, a new data set is produced. Note that, unlike in `MIMAR` and `MIMARsim`, the seeds are not printed in the standard output of `MIMARgof`. If you want to perform the same analysis, record the seeds values in "`seedmimar`" *before* executing `MIMARgof` or either of the Perl scripts, then edit "`seedmimar`" to those values. (The program can also be compiled with "`rand1t.c`" and "`rand2t.c`", which use the system clock for seeding the generators and does not use the file "`seedmimar`" at all.)

## Running `MIMARgof`

In this section, I describe how to run `MIMARgof`, a program that simulates polymorphism data under the isolation-migration model with parameters specified by the user and outputs the summary statistics calculated for the simulated data set.

### The input file

The input file for `MIMARgof` has the same format as the input file required by `MIMAR`. Use the switch "`-lf input`", to specify the input file name containing the information on the loci. See "`MIMARdoc.pdf`" for further details.
Below is an example for a four locus data set, in which `locus1_autosom` is autosomal and `locus2_Xlinked`, `locus3_Ylinked` and `locus4_mtDNA` are X- Y- or mtDNA-linked, respectively. The recombination scalar was set accordingly and I assumed that the mutation rate on the mtDNA was $2\times10^{-7}$ (see the file "`inputmimar`"):

```
Name           length x_y  v_y w_y n_1 n_2 S_1 S_2 S_s S_f //
locus1_autosom 1000   1    1   1   10  10  14  1   11  0
locus2_Xlinked 1000   0.75 1   0.5 10  10  2   5   2   0
locus3_Ylinked 1000   0.25 1   0   10  10  2   2   0   0
locus4_mtDNA   1000   0.25 10  0   10  10  9   20  14  0
```

### The basic command line

$$\texttt{mimargof}\ \boldsymbol{Y}\ \textit{-lf input -u}\ \boldsymbol{\mu}\ \textit{-t}\ \widehat{\boldsymbol{\theta_1}}\ \textit{-ej}\ \widehat{\boldsymbol{T}}\ \texttt{[options]}$$

This line shows the simplest usage of `MIMARgof`. There is one argument followed by the parameters (introduced by switches, such as "`-t`"). The argument $Y$, must appear first, while the switches can appear in any order.

| | |
|---|---|
| $Y$ | The number of loci considered. |
| $\mu$ | The generational mutation rate per bp. |
| $\widehat{\theta}_1 = \widehat{4N_1\mu}$ | The estimated population mutation rate per bp for the first population. When the other population mutation rates are not specified, they are equal to $\widehat{\theta}_1$. |
| $\widehat{T}$ | The estimated split time in generations, at which, backward in time, all lineages in population 2 are moved to population 1. |
| `input` | The input file name containing the information on the loci with their $S$ statistics (see section "The input file"). |
| `[options]` | A list of any options/switches described in the next sections. |

The migration rate is zero by default. The user needs to specify $\widehat{\theta}_1$ and $\widehat{T}$ because these two parameters are the minimum information required to built the simplest isolation-migration model, in which the two populations split $\widehat{T}$ generations ago without subsequent gene flow, and in which the ancestral and descendant populations have the same population mutation rate, $\widehat{\theta}_1$.

In the following basic command line `MIMARgof` will simulate data for the loci defined in the file `"inputmimar"` (see example in section "The input file") for a model with: $\widehat{\theta}_1 = \widehat{\theta}_2 = \widehat{\theta_A} = 0.00148198$, $\widehat{T} = 7957.96$ generations and $\widehat{M} = 0$. $\widehat{\theta}_1$ and $\widehat{T}$ are the point estimates (here I chose the modes) found in the file `"exsoutput"` (see section "The summary output file" of `"MIMARdoc.pdf"`). It will output the summary statsistics of the polymorphism in the standard output (see section "The standard output").

```
mimargof 4 -lf inputsim -u 2e-8 -t 0.00148198 -ej 7957.96
```

## Providing the other parameters of the model

To provide information about the of the isolation-migration model, the user needs to use either of the following switches:

$$\textit{-n}\ \widehat{\boldsymbol{\theta_2}}\ \textit{-N}\ \widehat{\boldsymbol{\theta_A}}\ \textit{-M}\ \widehat{\boldsymbol{M}}$$

| | |
|---|---|
| $\widehat{\theta}_2 = \widehat{4N_2\mu}$ | The estimated population mutation rate per bp for the second population. |
| $\widehat{\theta}_A = \widehat{4N_A\mu}$ | The estimated ancestral population mutation rate per bp. |
| $\widehat{M} = \widehat{4N_1m}$ | The estimated expected number of migrants between the two populations each generation, where $\widehat{m} = \widehat{M}\frac{\mu}{\widehat{\theta}_1}$ is the estimated symmetrical migration rate between the two populations. Note that $M$ is defined in term of $N_1$ (see section "Spatial structure and migration:" in `"msdoc.pdf"` for further details). |

In the following example, `MIMARgof` will proceed as before, but with the parameter values: $\widehat{\theta}_1 = \widehat{\theta}_2 = 0.00148198$, $\widehat{\theta}_A = 0.005$, $\widehat{T} = 7957.96$ generations and $\widehat{M} = 0.878097$ (i.e., the point estimates (here I chose the modes) found in the file `"exsoutput"`). The summary statistics of the polymorphism will be recorded in the standard output file `"outgof"`.

```
mimargof 4 -lf inputmimar -u 2e-8 -t 0.00148198 -ej 7957.96 -n 0.00148198 -N .005 -M
                              0.878097 > outgof
```

## The standard output

An example of a standard output from `MIMARgof` is reported below (see the file `"outgof"`):

```
9 13 31 1 0.443244 2.77222 3.86667 -0.817477 0.53465
```

In order the nine summaries of the simulated polymorphism are:

| | |
|---|---|
| 1. $\sum_{y=1}^{Y} S_{1_y}$ | The number of derived polymorphisms unique to the samples from population 1, $S_1$, summed over the $Y$ loci. |
| 2. $\sum_{y=1}^{Y} S_{2_y}$ | The number of derived polymorphisms unique to the samples from population 2, $S_2$, summed over the $Y$ loci. |
| 3. $\sum_{y=1}^{Y} S_{s_y}$ | The number of polymorphisms with shared derived alleles between the two samples, $S_s$, summed over the $Y$ loci. |
| 4. $\sum_{y=1}^{Y} S_{f_y}$ | The number of polymorphisms with fixed alleles in either sample, $S_f$, summed over the $Y$ loci. |
| 5. $\frac{\sum_{y=1}^{Y} F_{ST_y}}{Y}$ | The mean over the $Y$ loci of $F_{st}$, a measure of differentiation between the two population samples (Wright, 1931; Hudson et al., 1992). |
| 6. $\frac{\sum_{y=1}^{Y} \pi_{1_y}}{Y}$ | The mean over the $Y$ loci of the mean pairwise difference for population 1, $\pi_1$ (Nei and Li, 1979). |
| 7. $\frac{\sum_{y=1}^{Y} \pi_{2_y}}{Y}$ | The mean over the $Y$ loci of the mean pairwise difference for population 2, $\pi_2$ (Nei and Li, 1979). |
| 8. $\frac{\sum_{y=1}^{Y} D_{1_y}}{Y}$ | The mean over the $Y$ loci of Tajima's $D$ for population 1, $D_1$ (Tajima, 1989). |
| 9. $\frac{\sum_{y=1}^{Y} D_{2_y}}{Y}$ | The mean over the $Y$ loci of Tajima's $D$ for population 2, $D_2$ (Tajima, 1989). |

## More complex models

All the options and switches listed in this section of `"MIMARdoc.pdf"` are conserved in `MIMARgof` (the only exception is that the migration rates cannot be chosen from prior distributions). Read the section "More complex models" in `"MIMARdoc.pdf"` for details. To test the goodness of fit of a model `MIMAR`, one needs to all the options and switches that were used when running `MIMAR`. For example, if `MIMAR` ran with:

```
mimar 11000 1000 4 -lf inputmimar -u 2e-8 -t u .001 .01 -ej u 0 1e5 -N .005 -M 1 -2 2 -o
        soutput [extra_MCMC_options] -r e 1.667 -en .5 .2 -eM .8 10 >std_output
```

The command line for `MIMARgof` should contain all the extra switched describing the model, :in which the values $\widehat{\theta}_1$, $\widehat{T}$ and $\widehat{M}$ are either the point estimates found in the summary output file from `MIMAR` `"exsoutput"` or the values from a set of parameters sampled from the posterior distribution estimated by `MIMAR` (i.e., from a line of the file `"std_output"`):

mimargof 4 -lf inputmimar -u 2e-8 -t -t $\widehat{\theta}_1$ -ej $\widehat{T}$ -N .005 -M $\widehat{M}$ -r e 1.667 -en .5 .2 -eM .8 10

### Crossing over

**Fixed $\rho$ across loci.** If `MIMAR` ran with the command line:

mimar nsteps bsteps $Y$ -lf input -u $\mu$ -t $\theta_1$ -ej $T$ -r $\rho$ -o soutput

`MIMARgof` needs to run with:

mimargof $Y$ -lf input -u $\mu$ -t $\widehat{\theta}_1$ -ej $\widehat{T}$ -r $\rho$

$\rho = 4N_1c$ is the population cross-over rate per bp per generation and $c$ is the probability of cross-over per generation per bp. `MIMARgof` calculates the locus-specific population recombination rate like `MIMAR` (see section "Crossing over" in `"MIMARdoc.pdf"`).

**Fixed locus-specific $\rho$.** If the locus-specific recombination rates were fixed in the input file:

- If the switch "*-r 1*" was used when running `MIMAR`, use "*-r 1*" when running `MIMARgof`.

  **ATTENTION:** This assumes that in the input file, the recombination scalars for the recombining loci were set to $w_y = \omega_y\widehat{\rho_{\circ_y}}$, the scaled sex-averaged locus-specific population recombination rate per bp, i.e., for an X-linked locus, $c$ is the female recombination rate and $\omega_y = \frac{1}{2}$ so that $\omega_y\widehat{\rho_{\circ_y}} = 2\widehat{N_1 c_y}$ (see the file `"inputmimar_4Nc"`).

- If the switch "*-r 2*" was used when running `MIMAR`, use "*-r 2*" when running `MIMARgof`.

  **ATTENTION:** This assumes that in the input file, the recombination scalars for the recombining loci were set to $w_y = \omega_y\widehat{c_y}$, the scaled sex-averaged locus-specific recombination rate per bp, i.e., for an X-linked locus, $\widehat{c_y}$ is the estimate of the female recombination rate so the scaled sex-averaged rate is $\frac{1}{2}\widehat{c_y} = \omega_y\widehat{c_y}$ (see the file `"inputmimar_c"`).

**Variable locus-specific $\rho$.** The recombination rate can be allowed to vary across loci.

- If the switch "*-r e* $\boldsymbol{\lambda}$" was used when running `MIMAR`, use "*-r e* $\boldsymbol{\lambda}$" when running `MIMARgof`: the ratio $r = \frac{c}{\mu}$ is drawn from an exponential distribution prior with mean $\frac{1}{\lambda}$ for each recombining locus.
- If the switch "*-r n* $\boldsymbol{\nu}$ $\boldsymbol{\sigma}$" was used when running `MIMAR`, use "*-r n* $\boldsymbol{\nu}$ $\boldsymbol{\sigma}$" when running `MIMARgof`: the ratio $r = \frac{c}{\mu}$ is drawn from a normal distribution prior with mean $\upsilon$ and standard deviation $\sigma$ for each recombining locus.

### Asymmetrical migration rates

If `MIMAR` ran with the command line:

```
mimar nsteps bsteps Y -lf input -u μ -t θ1 -ej T -m i j Mij -o soutput
```

`MIMARgof` needs to run with:

```
mimargof Y -lf input -u μ -t θ̂1 -ej T̂ -m i j M̂ij
```

$\widehat{M_{ij}} = 4\widehat{N_1 m_{ij}}$, $i$ and $j \in [1, 2]$, $i \neq j$ is the estimated expected number of individuals that migrate from population $i$ into population $j$ each generation, thinking forward in time, where $\widehat{m_{ij}}$ is the estimated fraction of population $j$ that is made up of migrant from population $i$ every generation. Note that $M_{ij}$ is defined in term of $N_1$ (see section "Spatial structure and migration:" in `"msdoc.pdf"` for further details) .

## Other options conserved from ms. Use at your own peril!

Read the section "Other options conserved from ms. Use at your own peril!" in `"MIMARdoc.pdf"` for details.

## Summary of command line options

### The following options are required

| | |
|---|---|
| `-t` $\theta_1$ | Set the population mutation rate per bp to $4N_1\mu$ for population 1 (the default population). |
| `-u` $\mu$ | Set the mutation rate per bp to $\mu$. |
| `-lf input` | Set the input file name. |
| `-ej` $T$ | Set the time of split to $T$ generations ago. Backward in time, all lineages in population 2 are moved to population 1 at time |

**The following options are not required, but should be used if they were used when running `MIMAR`.**

| | |
|---|---|
| `-n` $\theta_2$ | Set the population 2 mutation rate per bp to $4N_2\mu$. |
| `-N` $\theta_A$ | Set the ancestral population mutation rate to $4N_A\mu$. |
| `-M` $M$ | Set the expected number of migrants between the two populations each generations to $4N_1m$. |
| `-m` $i$ $j$ $M_{ij}$ | Set the expected number of migrants from population $i$ into population $j$ each generation, $i$ and $j \in \{1,2\}$, $i \neq j$, to $4N_1m_{ij}$. |
| `-r` $\rho$ | Set the population recombination rate per bp to $4N_1c$. |
| `-r e` $\lambda$ | Set the prior distribution of $r = \frac{c}{\mu}$ to Exponential with mean $\frac{1}{\lambda}$. |
| `-r n` $\nu$ $\sigma$ | Set the prior distribution of $r = \frac{c}{\mu}$ to Normal$(\nu, \sigma)$. |
| `-r 1` | Set the locus-specific population recombination rates per bp to the value specified with $w$ in the input file. |
| `-r 2` | Set the locus-specific recombination rates per bp to the value specified $w$ in the input file. |

**The following options are conserved from `ms`. Use at your own peril!**

| | |
|---|---|
| `-se seed1 seed2 seed3` | Specify the random seeds from the command line. |
| `-f filename` | Read command line arguments from file `filename`. |
| `-c` $f$ $\lambda$ | Set ratio of gene conversion to recombination to $f$ and the track length to $\lambda$. |
| `-G` $\alpha$ | Set growth parameter of all populations to $\alpha$. |
| `-g` $i$ $\alpha_i$ | Set growth rate of population $i$ to $\alpha_i$. |

**The following options specify events occurring at time $\tau T$ generations. Up to 10 such switches can be used. It is the user's responsibility to specify times that are compatible with the isolation-migration model. Note that the switch "`-ej`" can be used only once.**

| | |
|---|---|
| `-eG` $\tau$ $\alpha$ | Set all growth rates to $\alpha$ at time $\tau T$ generations. |
| `-eg` $\tau$ $i$ $\alpha_i$ | Set growth rate of population $i$ to $\alpha_i$ at time $\tau T$ generations. |
| `-eb` $\tau$ $x$ | Set all population mutation rates to $x\theta_1$ at time $\tau T$ generations. |
| `-en` $\tau$ $i$ $x$ | Set population $i$ mutation rate to $x\theta_1$ at time $\tau T$ generations. |
| `-eM` $\tau$ $x$ | Set the symmetrical migration rate to $x$ at time $\tau T$ generations. |
| `-em` $\tau$ $i$ $j$ $x$ | Set $4N_1m_{ij}$ to $x$ at time $\tau T$ generations. |

## Performing a goodness of fit test using point estimates

In this section, I describe how to run the Perl script `perlgofest`, which executes `MIMARgof` using the point estimates of the parameters provided by `MIMAR`, which can be found in the summary output file (e.g., in `"exsoutput"`). By simulating samples using e.g., the mode of the parameters, you can test whether the model (described by the modes of the posterior distributions) estimated by `MIMAR` fits the data.

### Setting `perlgofest`

`perlgofest` runs `MIMARgof $int` times with the point estimates of the parameters provided by the user. The following parameters need to be changed in the file `perlgofest` depending of the data and the user's preferences:

| | |
|---|---|
| `$nloci` | Sets $Y$, the number of loci in the input file. |
| `$mu` | Sets $\mu$, the generational migration rate per bp. Set $\mu$ to the same value used to run `MIMAR` (i.e., after the switch "`-u`"). |
| `$int` | Sets the number of times the program `MIMARgof` will run with the specified set of parameter estimates. |
| `$inputfile` | Sets the name of the input file (see section "The input file"). |
| `$outfile` | Sets the name of the output file (see "The output file of `perlgofest`"). |
| `@temp` | Sets the parameter values to the point estimates (e.g., modes) found in the summary output file from `MIMAR` (e.g., in `"exsoutput"`): `$temps[1]`$= \widehat{\theta_1}$ `$temps[2]`$= \widehat{\theta_2}$ `$temps[3]`$= \widehat{T}$ `$temps[4]`$= \widehat{\theta_A}$ `$temps[5]`$= \widehat{M_{12}}$ `$temps[6]`$= \widehat{M_{21}}$ |
| `$cmd_line` | The user needs to add the extra switches used to run `MIMAR` in the line: `"./mimargof $nloci -lf $inputfile -u $mu -t $temp[1] -ej $temp[3] -N $temp[4] -n $temp[2] -m 1 2 $temp[5] -m 2 1 $temp[6] >>$outfile";` |

For example, if `MIMAR` ran with recombination, a bottleneck $0.5T$ generations ago and a change of gene flow rate $0.8T$ generations ago:

```
mimar 11000 1000 4 -lf inputmimar -u 2e-8 -t u .001 .01 -ej u 0 1e5 -N .005 -M l -2 2 -o
            soutput [extra_MCMC_options] -r e 1.667 -en .5 .2 -eM .8 10
```

You need to set ($\widehat{\theta_1}$, $\widehat{\theta_2}$, $\widehat{T}$ and $\widehat{M}$ are the point estimates (e.g., modes) found in the summary output file `"soutput"`):

| | |
|---|---|
| `$nloci=` | `4;` |
| `$mu=` | `2e-8;` |
| `$int=` | `1000;` |
| `$inputfile=` | `"inputmimar";` |
| `$outfile=` | `"outputgofest";` |
| `$temps[1]=` | $\widehat{\theta_1}$; |
| `$temps[2]=` | $\widehat{\theta_2}$; |
| `$temps[3]=` | $\widehat{T}$; |
| `$temps[4]==` | `0.005;` |
| `$temps[5]=` | $\widehat{M}$; |
| `$temps[6]=` | $\widehat{M}$; |
| `$cmd_line=` | `./mimargof $nloci -lf $inputfile -u $mu -t $temp[1] -ej $temp[3] -N $temp[4] -n $temp[2] -m 1 2 $temp[5] -m 2 1 $temp[6] -m 2 1 $temp[7] -r e 1.667 -en .5 .2 -eM .8 10 >>$outfile";` |

Save `perlgofest`, in a terminal type "`chmod +x perlgofest`" then finally "`./perlgofest`" to execute the script.

By default, `perlgofest` uses the modes of the estimates found in the file `"exsoutput"`, runs `MIMARgof` 1,000 times and print the 1,000 lines of summary statistics in the file `"outputgofest"`.

## The output file of `perlgofest`

Below I report the three first lines out of 1001 of the file `"outputgofest"`, the output file of `perlgofest` with the default values:

```
Sim# Step# S1 S2 Ss Sf Fst pi1 pi2 D1 D2
1 NA 12 42 38 1 0.451486 2.98889 7.98889 -0.764146 0.361559
2 NA 16 19 26 0 0.215028 2.84444 3.45 -0.482247 0.0649717
```

The first line is the header of the results. For each following lines, eleven values are reported:

1. The run number of `MIMARgof` (here the two first runs).
2. "`NA`", required uninformative column in order to use the script `"testGOF.R"` (see section "Does the estimated model fit the data?").
3.-11. The nine summary statistics for the simulated data set output by `MIMARgof` (see section "The standard output").

## Performing a goodness of fit test by sampling from the estimated posterior distribution

In this section, I describe how to run `perlgof`, which executes `MIMARgof` with sets of parameters sampled from the posterior distribution estimated by `MIMAR`. Simulating samples using parameters chosen from the posterior distribution allows one to test whether the model described by the posterior distribution estimated by `MIMAR` fits the data by performing a Bayesian goodness of fit test (i.e., by calculating the posterior predictive p-values for the summary statistics output by `MIMARgof`, see sections "The standard output" and "Does the estimated model fit the data?").

### Setting `perlgof`

`perlgof` reads each line containing a set of parameters from the posterior distribution estimated by `MIMAR` reported in a standard output file of `MIMAR` (see section "The standard output" in `"MIMARdoc.pdf"`). `perlgof` then executes `MIMARgof` with these sets of parameters. The following parameters need to be changed in the file `perlgofest` depending of the data and the user's preferences:

| | |
|---|---|
| `$nloci` | Sets $Y$, the number of loci in the input file. |
| `$mu` | Sets $\mu$, the generational migration rate per bp. Set $\mu$ to the same value used to run `MIMAR` (i.e., after the switch "`-u`"). |
| `$int` | `$int-1` sets the number of lines in the standard output file of `MIMAR` skipped between runs of `MIMARgof`. **My advice:** I would recommend running `MIMARgof` with about 10,000 sets of parameters sampled from the posterior distribution. This will make the process reasonably fast and provide reasonable estimates of the posterior predictive p-values. If the standard output file of `MIMAR` is larger than 10,000 lines, you should consider running `MIMARgof` only every, say, 10 recorded sets of parameters, by setting, e.g., "`$int=10`". |
| `$addburnin` | Sets the number of lines of extra burnin **My advice:** If you find that `MIMAR` was run with too little burnin (see section "How long to run the program and how to improve the MCMC?" in `"MIMARdoc.pdf"`), you can choose to ignore the first few steps recorded in the file to add some extra burnin by setting "`$addburnin`$>$0". |
| `$inputfile` | Sets the name of the input file (see section "The input file"). |
| `$outfile` | Sets the name of the output file (see "The output file of `perlgof`"). |
| `$post_dist` | Sets the name of the standard output file of `MIMAR` containing the estimate of the posterior distribution. |
| `$extra_switches` | The string of character with the extra switches used to run `MIMAR`. |

For example, if `MIMAR` ran with recombination, a bottleneck $0.5T$ generations ago and a change of gene flow rate $0.8T$ generations ago:

```
mimar 11000 1000 4 -lf inputmimar -u 2e-8 -t u .001 .01 -ej u 0 1e5 -N .005 -M l -2 2 -o
        soutput [extra_MCMC_options] -r e 1.667 -en .5 .2 -eM .8 10 >std_output
```

You need to set:

```
$nloci=           4;
$mu=              2e-8;
$int=             1;
$addburnin=       0;
$inputfile=       "inputmimar";
$outfile=         "outputgof";
$post_dist=       "std_output";
$extra_switches=  "-r e 1.667 -en .5 .2 -eM .8 10";
```

Save `perlgof`, in a terminal type "`chmod +x perlgof`" then finally "`./perlgof`" to execute the script.

Note that, unlike in `perlgofest`, the user does not need to specify the parameter values since they are registered in the file `"std_output"` in this case. For example, when analyzing the standard output file of `MIMAR` `"outputmimar"` with `perlgof`, the first run of `MIMARgof` will be with the set of parameters: `$temp[1]=$temp[2]=0.00512989`, `$temp[4]=7179.1` `$temp[5]=0.005` and `$temp[6]=$temp[7]=5.96188`.

By default, `perlgof` uses every nine sets of parameters from the estimated posterior distribution found in the file `"outputmimar"`, runs `MIMARgof` 1,000 times with those sets of parameters and print the 1,000 lines of summary statistics in the file `"outputgof"`.

### The output file of `perlgof`

Below I report the three first lines out of 1001 of the file `"outputgof"`, the output file of `perlgof` with the default values:

```
Sim# Step# S1 S2 Ss Sf Fst pi1 pi2 D1 D2
1 1001 30 36 16 0 0.0626264 3.37222 4.12778 0.125916 -0.509258
2 1011 18 42 27 0 0.0728636 2.93889 4.47222 0.036836 -1.14938
```

The first line is the header of the results. For each following lines, eleven values are reported:

1. The run number of `MIMARgof` (here the two first runs).
2. The step number associated with the sampled set of parameters (i.e., the first number in a recorded MCMC step in the standard output of `MIMAR`).

3.-11. The nine summary statistics for the simulated data set output by `MIMARgof` (see section "The standard output").

## Does the estimated model fit the data?

I provided the `R` script `"testGOF.R"` to help the user test whether the estimated model fit the data.

### Setting `"testGOF.R"`

The following parameters need to be changed in the file `"testGOF.R"` depending of the data and the user's preferences:

| | |
|---|---|
| `gof_file1` | Sets the name of the output file of `perlgof` (or `perlgofest`) from one seed. |
| `gof_file2` | Sets the name of the output file of `perlgof` (or `perlgofest`) from a different seed. |
| `true` | Sets the values of the nine summary statistics calculated from the original data set used to run `MIMAR`. Write the values in the same order as presented in the standard output of `MIMARgof` separated by ",": `true=c(`$\sum_{y=1}^{Y} S_{1_y}$`,` $\sum_{y=1}^{Y} S_{2_y}$`,` $\sum_{y=1}^{Y} S_{s_y}$`,` $\sum_{y=1}^{Y} S_{f_y}$`,` $\frac{\sum_{y=1}^{Y} F_{ST_y}}{Y}$`,` $\frac{\sum_{y=1}^{Y} \pi_{1_y}}{Y}$`,` $\frac{\sum_{y=1}^{Y} \pi_{1_y}}{Y}$`,` $\frac{\sum_{y=1}^{Y} D_{1_y}}{Y}$`,` $\frac{\sum_{y=1}^{Y} D_{2_y}}{Y}$`)` |
| `nbin` | Sets the number of bins in each histogram. Reduce this number to smooth the plots. |

## The output of `"testGOF.R"`

The execution of the functions in `"testGOF.R"` using `R` leads to four graphs displaying the average over the two `perlgof` (or `perlgofest`) runs of the expected distribution of a statistic under the estimated model and its observed value (calculated from the data set used to run `MIMAR` and shown as a vertical line, see Figure 2):

| | |
|---|---|
| `S statistics` | Displays the four expected distributions of $\sum_{y=1}^{Y} S_{1_y}$, $\sum_{y=1}^{Y} S_{2_y}$, $\sum_{y=1}^{Y} S_{s_y}$ and $\sum_{y=1}^{Y} S_{f_y}$. |
| $\mathsf{F_{ST}}$ | Displays the expected distribution of $\frac{\sum_{y=1}^{Y} F_{ST_y}}{Y}$. |
| $\pi$ | Displays the two expected distributions of $\frac{\sum_{y=1}^{Y} \pi_{1_y}}{Y}$ and $\frac{\sum_{y=1}^{Y} \pi_{2_y}}{Y}$. |
| D | Displays the two expected distributions of $\frac{\sum_{y=1}^{Y} D_{1_y}}{Y}$ and $\frac{\sum_{y=1}^{Y} D_{2_y}}{Y}$. |

`"testGOF.R"` also output the p-values calculated for each statistic. If the p-value is small (i.e., $< .05$), it means that the estimated model does not fit this particular statistic well. Note that if the output files were generated by `perlgof`, the p-values correspond to the posterior predictive p-values.

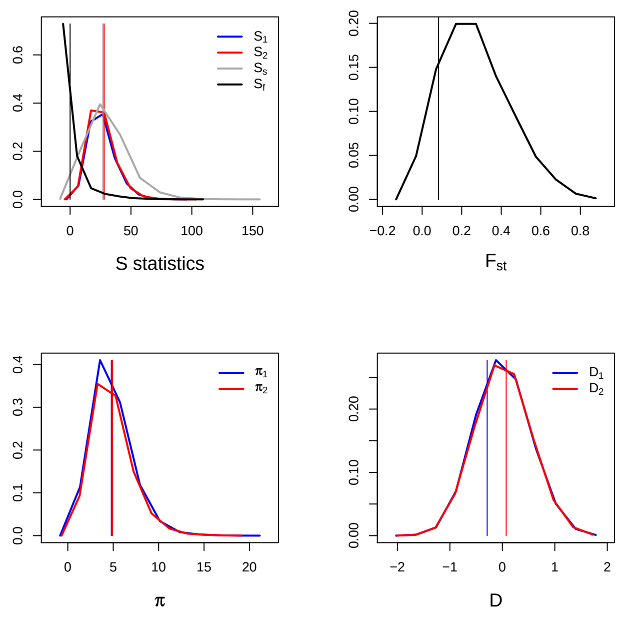

Figure 2: Example of goodness of fit graphs. Shown are the results from the files `"outputgof1"` and `"outputgof2"`, the output files of `perlgof` for the standard output files of `MIMAR` `"outputmimar1"` and `"outputmimar2"`. The observed values (vertical lines) were calculated from the data set summarized in the file `"inputmimar"`.

## If the ancestral alleles are unknown use `MIMARgof_noanc`

I provide the program `MIMARgof_noanc` which calculates the summary statistics of the polymorphism assuming *unknown ancestral alleles*. `MIMARgof` and `MIMARgof_noanc` are identical in all other aspects.

Use `MIMARgof_noanc` only if you used `MIMAR_noanc` to estimate the parameters of the model from data with unknow ancestral states.

### Compiling `MIMARgof_noanc`

To compile the program type:

```
gcc -o mimargof_noanc mimargof.c params.c make_gametes_noanc.c streec.c randX.c tajd.c
                                        -lm
```

See section "Downloading and compiling" for more details.

## List of parameters and symbols

| | |
|---|---|
| $\theta_i = 4N_i\mu$ | The population mutation rate per bp per generation for population $i \in \{1, 2, A\}$. |
| $\mu$ | The generational mutation rate per bp. |
| $N_i$ | The diploid effective size for population $i \in \{1, 2, A\}$. |
| $T$ | The split time in generations, at which, backward in time, all lineages in population 2 are moved to population 1. |
| $M = 4N_1m$ | The expected number of individuals in population 2 replaced by migrants from population 1 each generation (in forward direction). |
| $m$ | The symmetrical fraction of a population that is made up of migrant from the other population each generation. |
| $M_{ij} = 4N_1m_{ij}$ | The expected number of individuals in population $j$ replaced by migrants from population $i$ (in forward direction), $i$ and $j \in \{1, 2\}$, $i \neq j$. |
| $m_{ij}$ | The fraction of population $j$ that is made up of migrant from population $i$ each generation. |
| $\rho = 4N_1c$ | The population recombination rate per bp per generation. |
| $c$ | The generational recombination rate per bp. |
| $r = \frac{c}{\mu}$ | |
| $Y$ | The number of loci considered. |
| $n_i$ | The sample size for the locus in population $i \in \{1, 2\}$. |
| $x$ | The inheritance scalar reflecting copy number differences for a locus. |
| $v$ | The mutation rate variation scalar for a locus. |
| $w$ | The inheritance and rate variation scalar for recombination rate for a locus. |
| $\omega$ | The ratio of the locus-specific population recombination rate per bp over $\rho$. |
| $S_i$ | The number of derived polymorphisms unique to the samples from population $i \in \{1, 2\}$. |
| $S_s$ | The number of polymorphisms with shared derived alleles between the two samples. |
| $S_f$ | The number of polymorphisms with fixed alleles in either sample. |
| $F_{st}$ | A measure of differentiation between the two population samples (Wright, 1931; Hudson et al., 1992). |
| $\pi_i$ | The mean pairwise difference for population $i \in \{1, 2\}$, (Nei and Li, 1979). |
| $D_i$ | Tajima's $D$ for population $i \in \{1, 2\}$(Tajima, 1989). |

## List of files in the directory "mimargofdir"

| **Program files for** MIMAR | **Examples of input files** | **Examples of summary and standard output files from** MIMAR | **Examples of standard output files** | **Scripts to help with goodness of fit tests** | **Other or documentation files** |
|---|---|---|---|---|---|
| make_gametes.c | inputmimar | exsoutput | outgof | perlgof | makefile |
| mimargof.c | inputmimar_4Nc | outputmimar | outputgof | perlgofest | make_gametes_noanc.c |
| mimargof.h | inputmimar_c | outputmimar1 | outputgof1 | TestGOF.R | MIMARgofdoc.pdf |
| params.c | | outputmimar2 | outputgof2 | | seedmimar |
| rand1.c | | | outputgofest | | |
| rand1t.c | | | | | |
| rand2.c | | | | | |
| rand2t.c | | | | | |
| streec.c | | | | | |
| tajd.c | | | | | |

## Downloading other programs and documentations

MIMAR and "MIMARdoc.pdf" are found in "mimar.tar" available at http://przeworski.uchicago.edu/cbecquet/download.html.
ms and "msdoc.pdf" are available at http://home.uchicago.edu/~rhudson1/source/mksamples.html.
R is available at http://www.r-project.org/.

## References

Hudson, R. R., Slatkin, M., and Maddison, W. P., 1992. Estimation of levels of gene flow from DNA sequence data. *Genetics*, **132**:583–589.

Meng, X. L., 1994. Posterior predictive *p*-values. *Ann. Stat.*, **22**:1142–1160.

Nei, M. and Li, W. H., 1979. Mathematical model for studying genetic variation in terms of restriction endonucleases. *Proc. Natl. Acad. Sci.*, **76**:5269–5273.

Tajima, F., 1989. Statistical method for testing the neutral mutation hypothesis by DNA polymorphism. *Genetics*, **123**:585–595.

Wright, S., 1931. Evolution in Mendelian Populations. *Genetics*, **16**:97–159.