

# BIMr2 – User manual

Program version: 2.0

**Last update:** Wednesday, September 24, 2025

## Contents

1.	Overview.....	2
2.	Data files .....	2
2.1.	The default file format for genotypes .....	3
2.2.	The biallelic transposed file format for genotypes (alternative).....	3
2.3.	The file format for environmental variables.....	4
3.	Settings file .....	4
3.1.	Obligatory settings .....	5
3.2.	Optional settings.....	6
3.3.	Remarks (file name and keyword conventions) .....	6
4.	Executing the analysis.....	7
4.1.	Preparing the program to work.....	7
4.2.	Executing the program (Linux/Windows, command-line).....	7
4.3.	Executing the program (Windows GUI for BIMr2).....	8
4.4.	Remarks (for Windows users).....	8
5.	Output files .....	9
5.1.	The *.sum file.....	9
5.1.1.	Summary info .....	9
5.1.2.	Migration rates/log.psi (table) .....	9
5.1.3.	Migration matrix.....	9
5.1.4.	Regression model for environmental factors of migration .....	9
5.1.5.	Regression model parameters under the best model.....	10
5.1.6.	Divergence rates .....	10
5.1.7.	Allele frequencies .....	10
5.1.8.	Allelic drop-out rates .....	10
5.1.9.	Individual inbreeding rates .....	10
5.1.10.	Hyper-parameters of inbreeding model .....	10
5.1.11.	Acceptance rates across MCMC.....	10
5.1.12.	Parameters of proposal distributions for regression slopes.....	11
5.2.	The *.mcmc file .....	11
5.3.	The *.anc file.....	11
5.4.	The *.q file .....	11
5.5.	The *.env file .....	11
5.6.	The *.waic file.....	11
5.7.	The *.tim file .....	11
6.	Example data.....	12
7.	Note on using WAIC .....	12
8.	Final remarks .....	12
	References .....	12

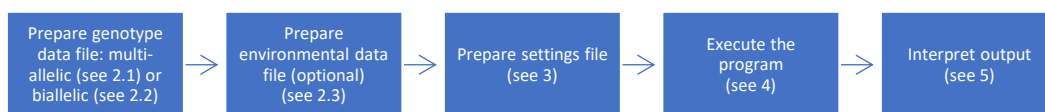
## 1. Overview

As a successor of BIMr<sup>1</sup>, the BIMr2 program re-implements a Bayesian approach to jointly estimate recent immigration rates and their ecological drivers across multiple populations within a closed metapopulation. The method assumes that all populations in the network have been sampled and relies on random samples of individuals from each population, genotyped at multiple codominant, possibly multiallelic marker loci, such as microsatellites or SNPs. To investigate the ecological determinants of migration, each population must be characterized by a set of ecological variables, with a geographic distance matrix typically serving as a baseline. Alternatively, if such covariates are unavailable, BIMr2 can still be used to estimate the recent migration matrix solely based on genetic data. The method accounts for excess homozygosity due to inbreeding, enabling unbiased estimation of individual inbreeding levels while controlling for population structure and recent individual ancestries. Modelling inbreeding at the individual level is a new feature of the method.

Compared to the original program, BIMr2 adopts a revised Markov Chain Monte Carlo (MCMC) approach, borrowed from another software, RSPMi<sup>2</sup>, that improves the estimation of migration rates, especially in scenarios involving a larger number of populations (especially when  $K > 10$ ). Consequently, BIMr2 requires shorter MCMC runs without compromising accuracy and precision.

The program utilizes parallel processing (using the OpenMP library; <https://www.openmp.org/>) to reduce analysis time when multiple CPU cores are available. For small to moderate datasets – particularly those involving microsatellites or substantially thinned SNP arrays – it runs efficiently on a modern personal computer. However, larger datasets (e.g., 1,000 individuals and 10,000 SNPs) require substantial computing resources to complete within a reasonable timeframe. Simulations indicate that as few as 1,000 SNPs can yield high-quality estimates, provided that genetic differentiation between source populations is sufficiently strong (i.e.,  $F_{ST} \geq 0.025$ ). Therefore, it is recommended to pre-filter large SNP datasets to reduce computational burden. In addition to excluding loci under selection (which is essential), filtering can be based on missing data and minor allele frequency criteria to increase the effective information content per locus, as well as on linkage disequilibrium to better align the data with the model's assumptions (i.e., independence between markers).

The analysis workflow is as follows:



## 2. Data files

Data are provided as simple text files, with values separated by tab or white space. For genotypes, two file formats are accepted: the default (2.1) and the biallelic transposed (2.2). The latter has been implemented to manage SNP biallelic genotype data, where the number of markers is typically larger than the number of individuals. It is important to note that the program expects the default file format unless a specific (optional) keyword is added in the settings file (see 3). For environmental variables, a single file format is implemented (2.3).

**Important note for genotype data files:** even if individuals need not be sorted according to sampling locations (population IDs) or individual ID numbers, it is recommended to sort them according to populations if the second data file with environmental variables is also provided. The

reason is that the program identifies populations using individual population IDs. Hence, the order of population IDs managed by the program in a given project will follow the order of unique population IDs in the data file with genotypes. In the examples below (2.1 and 2.2), the list of population IDs managed by the program will be ordered as [1,3,9]. This is important for processing environmental variables, which are provided for population pairs as matrices (see 2.3).

### 2.1. The default file format for genotypes

In the default case, each row contains all the data corresponding to a single individual, beginning with the individual's identity number (as an integer) followed by its population identification number (also as an integer). Subsequently,  $2L$  integers are expected encoding the individual's multilocus genotype (where  $L$  is the number of marker loci), each integer value encoding a single allele. To illustrate the data file structure, an excerpt from the example data file, containing 6 individuals from 3 populations, genotyped at 3 marker loci is shown below.

1	1	286	286	150	154	193	193
2	1	286	286	162	162	193	193
334	3	286	286	152	154	193	193
339	3	286	286	-1	-1	193	193
1601	9	280	286	154	154	181	193
1602	9	280	283	152	154	193	195

Here, the first individual has the id 1, and is sampled in population 1. The individual has a genotype at 3 loci. The genotype at the first locus is 286/286 (a homozygote). Missing genotypes are denoted with double -1 (here, the individual 339 has missing data at the second locus).

### 2.2. The biallelic transposed file format for genotypes (alternative)

In this case, data frame is transposed, i.e. each column contains all the data corresponding to a single individual, beginning with the individual's identity number (as an integer) followed by its population identification number (also as an integer). Subsequently (but optionally), two coordinate values: longitude and latitude, are given, corresponding to the individual's sampling location. Next,  $L$  integers are expected encoding the individual's multilocus genotype (where  $L$  is the number of marker loci), each integer value encoding a copy number of a major allele (i.e. 2, 1, 0 for major allele homozygote, heterozygote, and minor allele homozygote, respectively). To illustrate the data file structure, a toy example, containing 6 individuals from 3 populations, genotyped at 3 marker loci is shown below.

1	2	334	339	1601	1602
1	1	3	3	9	9
2	2	2	2	1	0
1	0	1	-1	2	1
2	2	2	2	1	1

Here, the first individual has the id 1, and is sampled in population 1. The individual has a genotype at 3 loci. The genotype at the first locus is 2 (a major allele homozygote). Missing genotypes are denoted with -1 (here, the individual 339 has missing data at the second locus).

Since the binary transposed data format is optional, an optional keyword `+biallelic_transposed` must be added in the settings file (see 3.2) to inform the program that a data frame is prepared using this format. Otherwise, the program will fail to read genotype data correctly.

### 2.3. The file format for environmental variables

The file format for environmental variables is basically the same as in the first program version. Each variable is provided as a separate  $K \times K$  matrix, where  $K$  is the number of populations. Each matrix is preceded by an identifier  $[G]=g$  placed in a separate row, where  $g$  is an index number pointing at the  $g$ -th variable. Following the identifier, the matrix is presented. There are as many matrices, as there are variables to consider in the regression model. In each matrix, rows represent recipient populations and columns represent source populations. Consequently, the element in the first matrix in the  $i$ -th row and the  $j$ -th column ( $G_{ij}^{(1)}$ ) will be used to model migration from population  $j$  to population  $i$ . The order of populations in rows needs to follow that in the genotype data file (see [2](#)).

Matrices can represent either unsigned distances (metrics), e.g. a geographic separation distance, or signed contrasts, e.g. a difference in forest cover between a source and a recipient population. It is crucial to distinguish between the two types, as they account for different aspects of causality. A distance-based matrix implies symmetric impact of a variable on migration whereas a contrast-based matrix implies asymmetric impact. The example (for  $K = 3$ ) shown below illustrates these two matrix types.

```
[G]=1
0.00    7.39    8.23
7.39    0.00    0.85
8.23    0.85    0.00
[G]=2
0.00    -0.94    0.03
0.94     0.00    0.97
-0.03   -0.97    0.00
```

Here, the first variable (G1) is an Euclidean distance between a source and a recipient, so that the matrix is perfectly symmetric and positive-valued ( $G_{ij}^{(1)} = G_{ji}^{(1)}$ ). The second variable (G2) is formed by contrasts between sources and recipients, so that the above- and below-diagonal elements cancels each other ( $G_{ij}^{(2)} = -G_{ji}^{(2)}$ ).  $G_{ij}^{(2)}$  results from subtraction  $x_{j2} - x_{i2}$ , where  $x_{i2}$  is a measurement for variable 2 taken in population  $i$ .

Importantly, the program computes standardized values in each matrix to get zero mean and unit variance variables before an analysis (see also [5.5](#)).

## 3. Settings file

The program requires a simple text file, a “settings file”, providing a list of user-defined settings. The settings file contains a number of keywords, each starting with either # or +. Obligatory settings, required for any analysis scenario, start with #, whereas optional settings start with +. Some keywords are expected to have an assigned value (after obligatory =). The structure of the settings file, prepared to analyze the example data is as follows:

```

#seed          = 0
#individuals    = 300
#markers       = 1000
#samples       = 20000
#burnin        = 10000
#keepevery     = 10
#threads       = 20
#gfile         = example.biallel_transformed
+efile         = example.factors
+inbreeding
+rjmc
#end

```

### 3.1. Obligatory settings

The keyword `#seed` is the integer number used for initializing a random number generator. If `#seed` equals zero (as in the example), the program uses a system clock-based initialization.

Using `#individuals` and `#markers` (both positive integers), the user declares genetic data dimensions, in terms of the number of individuals and marker loci, corresponding to the data stored in a file that must be named as declared in the `#gfile` setting. The length of data file name (in the above example, `example.biallel_transformed`) accepted by the program is limited to  $L = 255 - P$  characters, with  $P$  being a number of characters preceding the file name, including spaces. In other words, as the program reads up to 255 characters per line, the total number of characters per a keyword plus its value cannot exceed 255.

Using `#samples` and `#burnin`, the user declares the number of MCMC iterations used for the sampling and burn-in stages, respectively. As the burn-in stage is used for pilot tuning of proposal distributions, with proposal parameter adjustments every 1,000 iterations, the minimum recommended value for `#burnin` is 10,000 (as in the example above). As a result of the sampling stage, the program generates a final sequence of parameter values (hopefully representing a good approximation to the posterior distribution). To thin the sequence (to control for autocorrelation), the program keeps every  $k$ -th value which is defined by the user using the `#keepevery` setting.

Using `#threads`, the user sets the number of parallel processes used by the program. The maximum number of processes is limited by the CPU architecture (the number of processor cores/threads). Generally, the program is expected to run faster if more parallel processes are allowed. However, for specific data dimensions (broadly for relatively small numbers of individuals and loci), increasing the number of parallel processes above a certain level may result in no improvement or even some performance deterioration (e.g. due to the high general cost of parallelization relative to speed improvement per process). Usually, finding the optimal `#threads` value requires some experimentations that may be time-consuming for large data sets. Hence, the program has an in-built, simple optimization procedure that can be enabled if `#threads = 0`. Then, the number of parallel threads will be determined based on time consumption in initial iterations. It should be stressed that there is no guarantee that optimization will select the true optimum number of threads because the procedure relies on single-iteration times which can be subjected to some random variance due to temporal fluctuations in the overall CPU load.

The special keyword `#end` is used to inform the program that it reaches the end of settings declarations.

### 3.2. Optional settings

Optional settings are used to build a model that underlies the analysis. The list of possible keywords include: `+efile`, `+envir.factors`, `+interaction`, `+constant`, `+rjmcmm`, `+inbreeding`, `+dropout`, `+convert_data`, `+biallelic_transposed`.

Only `+efile` and `+envir.factors` can (and has to) be followed by a value (after obligatory `=`). In the case of `+efile`, the value after `=` defines a file name for environmental data (in the above example, `example.factors`; see also [2.3](#)). If the environmental file is defined, `+envir.factors` is used to define the number of environmental factors included in the environmental file. Two optional settings for regression analysis may be added: `+interaction` and `+constant`, with the former informing the program that first-order interaction terms between variables need to be added to the model, and the latter informing the program that a constant term needs to be added to the model. Interactions may often help uncover more complex relationships between migration patterns and environmental factors. Hence, they may be important for understanding the movement biology of a focal species. On the other hand, the optional constant term is implemented mostly for backward compatibility of BIMr2 with the predecessor. The `+rjmcmm` option needs to be added, to enable model selection via the posterior probabilities of the competing regression models. If `+rjmcmm` is not added, the full regression model (containing all environmental variables plus optional interaction terms, if selected) is used as a prior for migration rates.

To account for excessive homozygosity due to inbreeding, the `+inbreeding` keyword needs to be used, in which case individual inbreeding coefficients will be estimated rather than being assumed to be zero (as if the keyword is missing). Noteworthy, the program implements a model for genetic data where another source of increased homozygosity can be optionally accounted for, namely allelic drop-out<sup>3</sup>. To enable this functionality, `+dropout` needs to be added. This option may be especially helpful for microsatellite data, where such problems are often experienced. As inbreeding and allelic dropout are competing sources of increased homozygosity, ignoring one of them may lead to the overestimates in the other one.

To change the default data format ([2.1](#), typically used for multiple alleles codominant markers, e.g. microsatellites) into the binary transposed ([2.2](#), used for binary SNP markers), `+biallelic_transposed` keyword needs to be used.

Finally, if `+convert_data` keywords is added, the program outputs genotype input file in BayesAss<sup>4</sup> (\*.BA3) format.

### 3.3. Remarks (file name and keyword conventions)

**Output files** ([5](#)) generated based on the information in a given settings file have the same filename prefix as this file. Consequently, for the settings file 'test1.txt', the program will generate test1.sum, test1.mcmc, test1.anc, etc. Therefore, the name of the settings file serves much like the project's title. This is thus recommended to use the full file name convention, i.e. including a prefix (a file name extension), but avoiding the following extensions: "sum", "mcmc", "env", "tim", "anc", "waic", and "q" (see [5](#)). To run several analyses for the same data file, one needs to use unique names for settings files to distinguish output files for different runs. Using the settings file with the same filename for subsequent analyses will cause silent over-writing any existing output files with the same filename.

It is worth mentioning that the program reads five initial characters per keyword to recognize a specific option in the settings file (3). Hence, e.g. `#samp = 20000` is sufficient for informing the program about the required number of MCMC samples in the sampling stage.

## 4. Executing the analysis

### 4.1. Preparing the program to work

Under Microsoft Windows, the program does not require any specific installation. To use the program, one just needs to save the executable file (`bimr2.exe`) in a directory that will serve for storing all settings, data, and output files. Alternatively, Windows users can use a simple GUI program (`wBIMr2.exe`) (4.3) that assists in executing the analysis without needing to prepare the settings file or perform command-line operations via a terminal. Finally, Windows users may wish to take advantage of Linux optimizations natively via WSL (4.4).

For Linux users, to get the program ready to work one needs to compile the provided Fortran code (`bimr2.f90`) under their own system, because in this way the Linux environment will easily meet all library dependencies. For that purpose, it may be required to install first GNU Fortran (<https://gcc.gnu.org/fortran/>) (if not installed already). Then, since the program is written as a single source file, to make an executable, one can use (but do not do that!)

```
gfortran -fopenmp -o bimr2 bimr2.f90
```

The above command is the minimum to get the program to work. However, it is strongly recommended to enable helpful optimizations, to get the program fully efficient (e.g. to enable parallel computing):

```
gfortran -O3 -ftree-vectorize -funroll-loops -fopenmp -o bimr2 bimr2.f90
```

### 4.2. Executing the program (Linux/Windows, command-line)

Generally (both Windows and Linux), the program is launched from a command prompt. A single command-line argument is required to inform the program about the name of the settings file one wish to use for an analysis. Generally, after launching a terminal (Linux) or PowerShell (Windows), the user needs to navigate to the directory containing the BIMr2 executable file, the settings file, and data files. Then, to execute the BIMr2 program, they type (and execute):

```
./bimr2.exe example.prj
```

or

```
./bimr2 example.prj
```

under Windows and Linux, respectively. Here, the program argument `example.prj` is the settings file (3) name (the settings file is assumed to be located in the same directory as the executable).

If there are no issues related to the content of settings or data file, the program will start running immediately, displaying some information in the runtime, including the progress of MCMC iterations, e.g.

```
[bimr2] burn-in: 0.5% | LogPr: -0.597776E+06 | m: 0.2495 | Fst: 0.0507
```

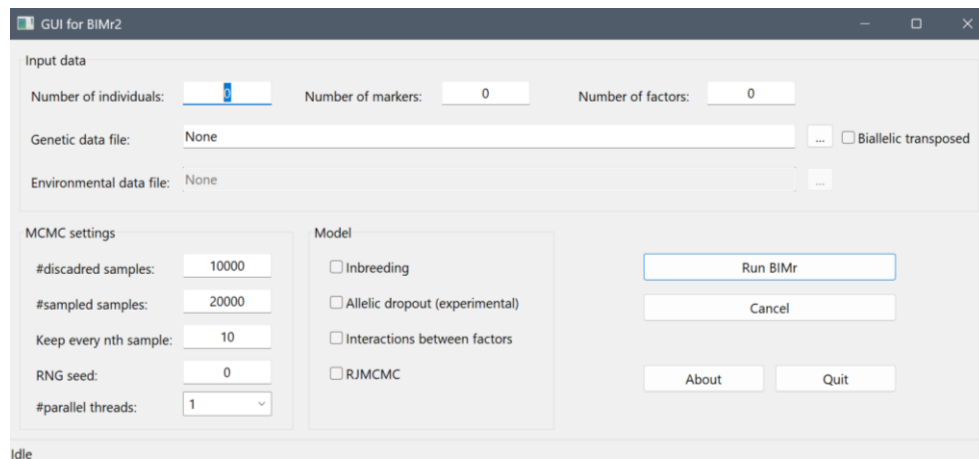


Once the program completes the analysis, the execution is automatically terminated with the information

Analysis completed without errors. Output written to  
(\* .sum, \* .mcmc, \* .anc) files.

#### 4.3. Executing the program (Windows GUI for BIMr2)

In the package, a simple Windows program (wBIMr2.exe) is provided, designed to assist Windows users in executing BIMr2.exe without the need for writing the settings file or using the PowerShell (a Windows terminal):



To use the GUI, the user must know in advance their data dimensions, including the number of individuals, markers, and (optionally) environmental factors. Only the latter can be set to zero, resulting in the environmental factors being not considered in the analysis. Simple rules apply:

1. Once data dimensions are set (only integers are allowed), the GUI enables selecting the genetic data file. Use the [...] button for that purpose.
2. As long as the default file format ([2.1](#)) is used for genetic data, leave '[ ] Biallelic transposed' unchecked. Otherwise ([2.2](#)), assure this checkbox is checked.
3. Once the number of factors is >0, the GUI enables selecting the environmental data file ([2.3](#)) using the [...] button on the right.
4. Before executing the analysis, the user needs to choose the MCMC settings and define the model according to their preferences.
5. Once the settings are chosen, use the [Run BIMr2] to execute the analysis. The GUI first writes an appropriately-formatted settings file ([3](#)) and then launches the BIMr2.exe program in a separate window (a terminal).
6. The on-going analysis can be cancelled any time using the [Cancel] button. A cancelled analysis cannot be continued. However, an already-generated MCMC sequence, in terms of \*.mcmc file ([5.2](#)), will be saved.
7. One advantage of using the GUI is that data files do not need to be in the same directory as the BIMr2.exe program. Importantly, if data files are located in the other directory, all output files ([5](#)) will be saved in that directory.

#### 4.4. Remarks (for Windows users)

Windows users are encouraged to run the program natively under Linux using the Windows Subsystem for Linux (WSL), such as with an Ubuntu distribution. WSL often manages local computing resources (e.g., memory access) more efficiently than standard Windows environments, allowing for shorter analysis time, compared to native Windows systems. For that purpose, first,



WSL needs to be installed. For that purpose, first WSL needs to be installed. Then, the instruction for Linux given above ([4.1](#) and [4.2](#)) needs to be followed.

## 5. Output files

Once the program completes the analysis, several output files are generated in the same directory. All the output files are simple (mostly tab-delimited) text files. As mentioned earlier, for the settings file `example.prj`, the output files have the prefix filename `example`. Taking the perspective of the user, the most important output file is `example.sum` ([5.1](#)). This file contains the posterior parameter estimates, as well as the summary of the settings used in the analysis.

### 5.1. The \*.sum file

This file contains the summary output file, with the post-processed posterior estimates of model parameters. The content is split into several sections. Details are described below. Some sections may be not present, depending on the analysis settings.

#### 5.1.1. Summary info

This section provides information about the program version, the data file and the settings used for the analysis. In addition, `Mean.LogPrAugData` and `Var.LogPrAugData` provide the mean and the variance of the log-probability of the augmented data (i.e. including inferred ancestry identifiers) across MCMC. In addition, a value of Widely-Applicable Information Criterion<sup>5</sup> is provided (WAIC). WAIC can be used to compare the relative performance of different models, e.g. with vs. without inbreeding, or with vs. without dropout (for more information see [7](#)). Finally, `CPU time (hr:min:sec)` provides the analysis time.

#### 5.1.2. Migration rates/log.psi (table)

This section provides posterior estimates of migration parameters  $m_{ij}$  and  $\log \psi_{ij}$ .  $\psi_{ij}$  can be interpreted as the expected number of migrants from population  $j$  to  $i$ , as predicted by the regression model. Each row shows estimates for a pair of population. Estimates of  $m_{ij}$  and  $\log \psi_{ij}$  are given as posterior means (`m.ij` and `log.psi.ij`, respectively), standard errors (SE), and 95% highest posterior density limits (`HPDI.l` and `HPDI.u`).

#### 5.1.3. Migration matrix

This section provides posterior mean vectors of migration rates  $\mathbf{m}_i$  in a matrix form. It is apparently redundant to the previous section but can be useful for showing gene pool sharing across populations due to recent migration, e.g. using a STRUCTURE-like<sup>6</sup> stacked bar plot.

#### 5.1.4. Regression model for environmental factors of migration

This section appears only if `+rjmcmc` option is used, and provides posterior probabilities of regression models visited in the resulting MCMC chain. The table shows `model.structure` using binary switches, with 1 and 0 denoting presence and absence of a given environmental variable in the model. For example, assuming that two environmental variables were provided (`+envir.factors = 2`) and the interaction term was added (`+interaction`), a series of binary switches `{1, 1, 0}` corresponds with G1 and G2 being included and G1x2 (their interaction) being excluded from the regression model.

Models are sorted in ascending order of posterior probability. Models that were not visited during MCMC are not shown - in this case the posterior probability value is about zero (meaning negligible importance).

#### 5.1.5. Regression model parameters under the best model

This section provides posterior estimates of parameters related to the best-performing regression model (the one with the highest posterior probability). One obligatory row is `sig2.psi`, i.e. variance of  $\psi_{ij}$  hyper-parameters, describing dispersion of individual  $\psi_{ij}$  values around the best regression model. If environmental factors are added, this section provides an estimate of the Bayesian  $R^2$  (`R2.psi`), i.e. the proportion of variance in  $\psi_{ij}$  explained by the best model. Subsequently, posterior estimates of regression coefficients are shown, e.g. `G.1`, `G.2`, `G.1x2`, with the letter being the interaction term between G1 and G2. In the case of regression coefficients, `freq.incl` shows the visiting frequency for a given coefficient being non-zero. For example, `freq.incl = 0.981` can be interpreted as the posterior probability of inclusion of a given coefficient in the regression model.

#### 5.1.6. Divergence rates

This section provides posterior estimates of divergence parameters ( $F_{ST}$ ) across populations. Estimates are given as posterior means (`fst`), standard errors (SE), and 95% highest posterior density limits (`HPDI.l` and `HPDI.u`). It should be kept in mind that estimates refer to the pre-migration divergence of local gene pools from the global gene pool.

#### 5.1.7. Allele frequencies

This section provides posterior mean allele frequencies in the global population (`anc`) and across local populations (`pop.k`). These values refer to the pre-migration gene pools. To reduce table complexity, only point estimates are shown.

#### 5.1.8. Allelic drop-out rates

This section appears only if allelic drop-out is chosen (`+inbreeding`). This section provides posterior estimates of allelic dropout rates across marker loci. Estimates are given as posterior means (`dout`), standard errors (SE), and 95% highest posterior density limits (`HPDI.l` and `HPDI.u`).

#### 5.1.9. Individual inbreeding rates

This section appears only if the optional keyword `+inbreeding` is included in the settings file. This section provides posterior estimates of inbreeding values across sampled individuals. Estimates are given as posterior means (`fis`), standard errors (SE), and 95% highest posterior density limits (`HPDI.l` and `HPDI.u`).

#### 5.1.10. Hyper-parameters of inbreeding model

This section appears only if inbreeding is included in the model. This section provides posterior estimates of hyper-parameters related to the inbreeding distribution. Here, each row shows the estimates for a single parameter, starting with the posterior mean (`estim.mean`), followed by the standard error (SE), and 95% highest posterior density limits (`HPDI.l` and `HPDI.u`). `mean.fis` is the mean and `disp.fis` is the dispersion parameter (proportional to variance) of a beta distribution used as a prior for individual inbreeding.

#### 5.1.11. Acceptance rates across MCMC

This section provides information about acceptance rates and proposal parameters for hyper-parameters estimated using the Metropolis algorithm. The program attempts to adjust proposal parameters to get the acceptance rates between 25 and 45%. If the acceptance rate lays outside this range, there may be issues with mixing across MCMC. In such cases, the results should be interpreted with caution.

#### 5.1.12. Parameters of proposal distributions for regression slopes

This section provides means and variances of the proposal distributions used in the RJMCMC analysis to make between-model jumps. These values refer to parameters of a normal distribution, and are estimated during the burn-in/pilot tuning stage of MCMC.

#### 5.2. The \*.mcmc file

This file contains a thinned MCMC sequence of parameter updates that can be used, for example, to plot approximate marginal distributions of selected parameters. The file has a tabular structure, starting with a header row, with columns representing parameters and rows representing subsequent MCMC iterations. It should be noted that numbers are rounded to four decimal places, except for regression slopes for the distance effects. As a result, any estimates (e.g. credible intervals) produced based on the \*.mcmc file can be affected by rounding, unlike the estimates in the \*.sum file.

#### 5.3. The \*.anc file

This file contains ancestry details, in terms of auxiliary variables, summed across the sampling stage (after thinning). It can be understood as an approximation to the posterior distribution of auxiliary variables used to describe individual ancestries of seed and pollen gametes. One row contains information on the frequency of assignments of a given individual's seed and pollen gamete to a given source population. The table omits null frequency cases, so that individuals may be represented by variable numbers of rows.

#### 5.4. The \*.q file

This file contains simplified posterior individual genealogies, and is somewhat redundant to the \*.anc file. In one row, the summary information is given for a single individual: the individual ID, the ID of individual's sampling location, the frequency of non-migrant, half-migrant, and full-migrant assignments across the (thinned) sampling stage. All three genealogies take sampling location as a reference information. Consequently, the half- and full-migrant classes inform what is the probability that a given individual has one or both (multilocus) gametes of non-local origin. In addition, the summary frequency of gamete assignments to candidate source populations is included.

#### 5.5. The \*.env file

This file contains standardized values of environmental factors, shown per population pairs. This file may help understand (e.g. by drawing a plot) the relationship between environmental factors and estimated  $\log \psi_{ij}$  values, shown in the \*.sum file.

#### 5.6. The \*.waic file

This file contains a list of individual WAIC scores (WAIC\_i) together with accompanied individual penalty scores (Penalty\_i). It is worth noting that the sum of individual WAICs equals the (global) WAIC value shown in the \*.sum file.

#### 5.7. The \*.tim file

This file plays a special role in informing about the estimated analysis time. The file is generated after 50 MCMC iterations. It may be helpful, especially when dealing with large data sets.

## 6. Example data

Together with the program, an example data set is provided. Briefly, this is an artificial set of 300 individual genotypes typed at 200 biallelic markers. Individuals were sampled from 15 populations characterized by the global  $F_{ST}$  of 0.05 and the mean individual inbreeding of 0.10. Two environmental factors were simulated from a normal distribution with their expected regression slopes equal to -0.9 and 1.1. The interaction was not simulated, so the expected interaction term is zero. Importantly, the genetic data are provided both in the default file format (“example.default”; 2.1) and in the biallelic transposed file format (“example.biallel\_transposed”; 2.2). However, the settings file is prepared for using the latter.

To run the analysis, the settings file (3) is provided as the “example.prj” file, where the number of burn-in and sampling iterations is set to 10,000 and 20,000, respectively, while a thinning interval is set to 10 iterations. Random number generator’s seed is chosen to be set based on the system clock. The number of parallel threads is set to be optimized by the program. The settings file was prepared to include inbreeding.

## 7. Note on using WAIC

As an information criterion, WAIC is a useful tool to assess the predictive power of a model. If multiple nested models are used to analyze the same data, WAIC can be used to compare their relative performance. For example, if one analyzes their data under several alternative settings, say with and without inbreeding, the model with lower WAIC is characterized by higher predictive power and should be preferred over the other one. WAIC is a deviance-based criterion, and as such, it can also be used to identify outliers among replicated MCMC runs performed under the same settings, as suggested in the study of Faubet et al.<sup>7</sup>.

A more detailed inspection of WAIC differences between models/replicates can be done using point-wise WAIC values in the \*.waic output file (5.6). Individual WAICs can be used to compute the standard error of the global WAIC, or of the difference in WAICs between two competing models. Assuming that the difference in WAIC between two competing models is normally distributed, one can test whether the difference in predictive power between these models is significant<sup>8</sup>. For that purpose, parallel `WAIC_i` values for different models/replicates need to be used to compute point-wise WAIC differences across individuals, from which the standard error is computed as usual.

## 8. Final remarks

The program is written in GNU Fortran (<https://gcc.gnu.org/fortran/>). It is released as a free, open-source software. Consequently, the user can modify the code freely to adjust the program to their needs. The program was designed to help resolve a scientific problem, but no warranty is given that it is free from bugs or fulfills someone’s expectations. In case of questions or problems, in particular to report errors, the user may contact [igorhy@ukw.edu.pl](mailto:igorhy@ukw.edu.pl).

## References

1. Faubet, P. & Gaggiotti, O. E. A new Bayesian method to identify the environmental factors that influence recent migration. *Genetics* **178**, 1491–1504 (2008).
2. Chybicki, I. J. & Robledo-Arnuncio, J. J. Spatially explicit estimation of recent migration rates in plants using genotypic data. *Genetics* **229**, iyae218 (2025).
3. Taberlet, P. et al. Reliable genotyping of samples with very low DNA quantities using PCR. *Nucleic Acids Res.* **24**, 3189–3194 (1996).

4. Wilson, G. A. & Rannala, B. Bayesian inference of recent migration rates using multilocus genotypes. *Genetics* **163**, 1177–1191 (2003).
5. Watanabe, S. Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory. *J. Mach. Learn. Res.* **11**, 3571–3594 (2010).
6. Pritchard, J. K., Stephens, M. & Donnelly, P. Inference of population structure using multilocus genotype data. *Genetics* **155**, 945–59 (2000).
7. Faubet, P., Waples, R. S. & Gaggiotti, O. E. Evaluating the performance of a multilocus Bayesian method for the estimation of migration rates. *Mol. Ecol.* **16**, 1149–1166 (2007).
8. McElreath, R. Statistical Rethinking : A Bayesian Course with Examples in R and STAN. *Stat. Rethink.* (2020) doi:10.1201/9780429029608.