RSPMi – User manual

Program version: 1.0

Igor J. Chybicki

Last update: Monday, August 26, 2024

Contents

1. About the program

RSPMi stands for recent seed and pollen migration inference, declaring the program's main purpose which is jointly assessing seed and pollen migration, treated as independent processes. The program implements a Bayesian method (Chybicki and Robledo-Arnuncio, submitted) to estimate recent seed and pollen immigration rates for a number of populations assumed to form a closed population network (a meta-population) without migration from external sources. All populations are assumed to be sampled. The estimation relies on random samples of individuals collected from every population, genotyped at a number of codominant marker loci, typically microsatellites or SNPs. In addition, provided that individual or population geographic coordinates are known, the program can be used to test if migration rates are spatially dependent (i.e. if they are significantly associated with inter-population distance), separately for seeds and pollen. The method accounts for excessive homozygosity due to inbreeding and allelic dropout, and therefore it can also be used to get unbiased estimates of individual inbreeding, controlling for the effect of population structure (the Wahlund effect).

Despite the program's (expanded) name, RSPMi can be used for both plants and animals, depending on the specific dispersal biology. For example, sperm cells released by certain animals to the environment in order to fertilize retained eggs are migration vectors effectively equivalent to pollen in plants. Therefore, the program can be suitable for analyzing recent migration in, for instance, sponges and some corals and Mollusca species to distinguish larvae dispersal vs. sex cell dispersal as two different factors of population connectivity. Moreover, as gametic (haploid) migration estimation can be turned off, the program can be used to infer zygotic (diploid) migration only, as expected for animals.

The program was designed to use parallel processing aiming to decrease the time required for a single analysis if multiple cores are available. For small data sets, especially for microsatellites, the program runs very quickly on a modern personal computer. However, for large data sets (e.g. 1,000 individuals and 10,000 SNPs), the program would require substantial computing capacity to complete the run in a reasonable time. For example, the analysis of simulated data for 1,000 individuals, 10 populations, and 10,000 SNPs (under settings: `#samples = 20000, #burnin = 10000`, see the "Settings file" section), takes approximately 85 h using a server computer with 32 GB RAM and AMD Epyc 7601 (32 cores/64 threads). However, on a server computer with 3 TB RAM and 4 × Intel(R) Xeon(R) Platinum 8276 (4 × 28 cores/224 threads), the analysis completed after 17 h, demonstrating that the runtime scales about linearly with the overall computing capacity (i.e. computing capacity per core × number of cores). It should be stressed that parallel processing is achieved via the OpenMP library (https://www.openmp.org/). Consequently, parallelization requires a shared memory system architecture.

2. Data file

The program accepts numerical data entries only. A data file is a simple text file, with values separated by tab, space, or comma. 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 (but optionally, see the next section), two coordinate values: longitude and latitude, are given, corresponding to the individual's sampling location. Next, 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    21.6455   49.5101   286  286  150  154  193  193
2    1    21.6451   49.5097   286  286  162  162  193  193
334  3    21.5705   49.5682   286  286  152  154  193  193
339  3    21.5705   49.5680   286  286  -1   -1   193  193
1601 9    21.4426   49.5845   280  286  154  154  181  193
1602 9    21.4425   49.5845   280  283  152  154  193  195
```

Here, the first individual has the id `1`, and is sampled in population `1`. The individual has longitude and latitude values of `21.6455` and `49.5101`, which are followed by its 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).

If coordinates are unavailable, columns 3 and 4 used for coordinates are simply omitted, so that a genotype immediately follows population ID. It should be noted that individual coordinates need not be unique. If they are, as in the example data, the program computes median coordinates for each population based on individual coordinates. Since the program expects that all individuals have coordinates (missing coordinates are not allowed), missing coordinates need to be replaced with interpolated values. Individuals need not be sorted according to sampling locations (population IDs) or individual ID numbers.

3. Settings file

The program requires a simple text file providing a list of user-defined settings. This file will be hereafter called the "settings file". 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 with all the optional settings included, is as follows:

```
#seed        = 0
#individuals = 1167
#markers     = 20
#coordinates
#samples     = 20000
#burnin      = 10000
#keepevery   = 10
#threads     = 20
#dataframe   = Taxus_Trees_Beskids.txt
+gametic_migration = 0.25
+zygotic_migration
+inbreeding
+dropout
+spatial_zygotic
+spatial_gametic
```

```
+rjmcmc
#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 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 #dataframe setting. The length of data file name (in the above example, Taxus_Trees_Beskids.txt) 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.

If a data file contains geographic coordinates (see above), the program must be informed about it using either #coordinates (as in the example above) or #plane keyword. The difference between the two keywords is subtle but very important. The keyword #coordinates is reserved for longitude/latitude coordinates given in degrees. Whereas, #plane is reserved for the Cartesian plane coordinates (e.g. UTM), typically given in meters. Consequently, the two keywords are mutually exclusive options.

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. By default, the model represents a specific null model, where populations are characterized by divergent gene pools (much like the F-model[1]) but are not connected by recent migration. To include seed and/or pollen recent migration, specific model components need to be added to the model. Similarly, the model may involve inbreeding, dispersal kernels, etc. The list of possible keywords include: `+gametic_migration` (pollen migration), `+zygotic_migration` (seed migration), `+inbreeding`, `+dropout`, `+spatial_zygotic`, `+spatial_gametic`, `+rjmcmc`, `+convert_data`.

Only `+zygotic_migration` and `+gametic_migration` can (optionally) be followed by a value (after obligatory =). The optional value defines a prior assumption about the level of local (within-population) dispersal (λ). The default parameter value is $2/3$, but can vary between 0 and 1 (in the example above, λ is set to 0.25 for zygotic migration; for gametic migration, since the value is not defined, the default parameter value of $2/3$ is used). λ intervenes in the prior distribution used for migration vectors. Let π_{ij} be the expected value of zygotic migration α_{ij} from the j-th to the i-th population, such that

$$\pi_{ij} = \begin{cases} \lambda_\alpha + (1-\lambda_\alpha)\tau_\alpha & \text{if } i = j \\ (1-\lambda_\alpha)(1-\tau_\alpha)\rho_{ij} & \text{if } i \neq j \end{cases}$$

where τ_α is the isolation parameter subjected to estimation, ρ_{ij} is the fraction of seed migration that occurs from population j to population i. For the default value $\lambda_\alpha = 2/3$, the prior probability for zygotic (seed) non-migration is $\pi_{ii} = (2+\tau_\alpha)/3$, so the probability of non-immigration α_{ii} is assumed to be within ($2/3$ and 1). It should be stressed that even if the minimum non-immigration prior π_{ii} is set to $2/3$ (or any other positive value), α_{ii} is treated as a probability vector element of $\boldsymbol{\alpha}_i$, where $\alpha_{ij} \in (0,1)$ for any j and $\sum_{j=1}^{K} \alpha_{ij} = 1$. Consequently, if there is a strong data evidence about high immigration, the posterior estimate of α_{ii} can be lower than the minimum non-immigration prior value. On the other hand, if data evidence is weak (due to low genetic discrimination power, expressed as low genetic differentiation among populations), the prior parameter $\lambda_\alpha > 0$ may help extract the information about migration patterns. In such a case, however, the posterior migration estimates are strongly dependent on the choice of λ_α value. Analogous comments apply to gametic (pollen) migration parameters.

To account for excessive homozygosity due to either inbreeding or allelic drop-out[2], `+inbreeding` and `+dropout` keywords can be used, in which case individual inbreeding coefficients and perlocus allelic drop-out rates will be estimated, respectively, rather than being assumed to be null. Finally, if geographic coordinates are available, the user can include spatially-explicit modeling of migration rates using `+spatial_zygotic` and `+spatial_gametic`. In addition, if `+rjmcmc` is added, the program estimates the posterior probabilities for spatial migration models (vs. non-spatial migration models). However, spatial model-related settings have any effect only if respective `+zygotic_migration` and `+gametic_migration` keywords are added.

Finally, if `+convert_data` keywords is added, the program outputs genotype input files in BayesAss[3] (*.BA3) and BIMr[4] (*.gen) formats.

3.3. Remarks

Output files 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. To run several analyses for the same data file, likely under different settings, one needs to use unique names for settings files to distinguish output files for different runs. Using the settings file with same filename prefix for subsequent analyses will cause over-writing any existing output files with the same prefix.

It is worth mentioning that the program reads five initial characters per keyword to recognize a specific option. Hence, `#samp = 20000` is sufficient for informing the program about the required number of MCMC samples in the sampling stage. The only exceptions are the compound keywords `+spatial_zygotic` and `+spatial_gametic`. In those cases, `+spat_zygot` and `+spat_gamet` represent proper abbreviated forms.

4. Executing the program

Under Microsoft Windows, the program does not require any specific installation. To use the program, one needs to save the executable file (rspmi.exe) in a directory that will serve for storing all settings, data, and output files.

For Linux users, the best way to get the program ready to work is to compile the provided Fortran code (rspmi.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 simply use

```
gfortran -fopenmp -o rspmi rspmi.f90
```

The above command is the minimum to get the program to work. To enable some optimizations, one may wish to use

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

Generally (in 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. Under Windows, assuming that the current directory contains the rspmi executable file, the command is

```
rspmi.exe test.txt
```

where `test.txt` is the settings file name (the settings file is assumed to be located in the same directory as the executable). Under Linux, the command is

```
./rspmi test.txt
```

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.

```
[rspmi] sampling: 67%  -0.349434E+05 0.9383 0.9083 0.2599
0.0085
```

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.
```

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 `test_run.txt`, the output files have the prefix filename `test_run`. Taking the perspective of the user, the most important output file is `test_run.sum`. 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 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 identificators) across MCMC. In addition, a value of Widely-Applicable Information Criterion[5] 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. Finally, `CPU time (hr:min:sec)` provides the analysis time.

5.1.2. Migration rates (table)

This section provides posterior estimates of α_{ij} and β_{ij} migration parameters. Each row shows estimates for a pair of population. Estimates of α_{ij} and β_{ij} are given as posterior means (`alpha` and `beta`, respectively), standard errors (`SE`), and 95% highest posterior density limits (`HPDI_l` and `HPDI_u`). If geographic coordinates are provided, the table includes separation distances between populations.

5.1.3. Seed/pollen pool composition

This section provides posterior mean vectors of $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_j$ in a matrix form. It is apparently redundant to the previous section but can be useful for showing propagule compositions across populations, e.g. akin to a STRUCTURE bar plot.

5.1.4. 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.

5.1.5. 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.

5.1.6. Allelic drop-out rates

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`).

5.1.7. Individual inbreeding rates

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`).

5.1.8. Hyper-parameters

This section provides posterior estimates of hyper-parameters. 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`). The complete list of rows is shown below.

Parameter	Comment
`mean_fst`	Mean of a (truncated at 1) exponential distribution used as a prior for divergence
`mean_fis`	Mean of a beta distribution used as a prior for individual inbreeding
`disp_fis`	Dispersion parameter (proportional to variance) of a beta distribution used as a prior for individual inbreeding
`disp_alpha`	Dispersion parameter (proportional to variance) of a Dirichlet distribution used as a prior for zygotic migration
`disp_beta`	Dispersion parameter (proportional to variance) of a Dirichlet distribution used as a prior for gametic migration
`isol_alpha`	Isolation parameter (tau) used to adjust mean vector of a Dirichlet distribution used as a prior for zygotic migration
`isol_beta`	Isolation parameter (tau) used to adjust mean vector of a Dirichlet distribution used as a prior for zygotic migration
`Pr(spat_alpha)`	The posterior probability of the spatial model for zygotic migration
`Pr(spat_beta)`	The posterior probability of the spatial model for gametic migration
`spat_alpha`	The effect of spatial separation on zygotic migration (scale of power-law dispersal kernel)

spat_beta	The effect of spatial separation on gametic migration (scale of power-law dispersal kernel)

Depending on declarations in the settings file, this section may provide a truncated list of hyper-parameters. It is important to note that, if `+rjmcmc` is declared in the settings file, then the spatial effect parameters' (i.e. `spat_alpha` and `spat_beta`) posterior means, standard errors, and HPDI limits are determined based on a subsample of the MCMC chain with non-zero parameter updates generated under the spatial model only. As a result, for the same data, spatial effect estimates can differ (and usually do) between runs with RJMCMC turned on and off.

5.1.9. 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.

5.1.10. Parameters of proposal distributions for spatial effects

This section provides means and variances of the distribution 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 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 *.mig file

This file contains posterior estimates of *gene* migration rates between populations and can be helpful in understanding how seed and pollen migration shapes total gene migration among populations. Gene migration rates can be directly compared against results generated by other programs (e.g. BIMr[4]).

5.6. The *.geo file

This file contains median population coordinates, computed based on coordinates of individuals. This file may help understand how distances between populations, shown in the `*.sum` file, were computed.

5.7. The *.plt file

This file contains a minimal gnuplot (http://www.gnuplot.info/) script that can be used to plot output information in a form of vector graphics (*.svg). The script can be modified easily to adjust plots according to specific requirements.

5.8. The *.waic file

This file contains a list of individual WAIC scores together with accompanied individual penalty scores. It is worth noting that the sum of individual WAICs equals the (global) WAIC value shown in the `*.sum` file. Individual WAICs can be used to compute the standard error of the global WAIC, or of the difference in WAICs between two competing models, e.g. with vs. without inbreeding.

6. Example data

Together with the program, an example data set on English yew (*Taxus baccata* L.) is provided to replicate the results shown in a paper introducing the method (Chybicki and Robledo-Arnuncio, submitted). Briefly, this is a set of 1167 individual genotypes at 20 microsatellite markers. Individuals (trees) were sampled from 9 remnant populations of the species. Trees were individually georeferenced (lon/lat coordinates in degrees).

To run the analysis with and without spatially-explicit migration model, two settings files are provided: “Taxus_spatial.txt” and “Taxus_nonspatial.txt”. In both cases, 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. Both settings files were prepared to include inbreeding and allelic dropout. Both options can be turned off by removing appropriate keywords (see the “Settings file” section).

7. Final remarks

The program is written in Fortran and compiled under 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. Unlike commercial software companies, the author cannot guarantee regular help. However, in case of questions or problems, especially for bug reporting, the user is welcome to contact igorchy@ukw.edu.pl.

References

1. Gaggiotti, O. E. & Foll, M. Quantifying population structure using the F-model. *Mol. Ecol. Resour.* **10**, 821–30 (2010).

2. Taberlet, P. *et al.* Reliable genotyping of samples with very low DNA quantities using PCR. *Nucleic Acids Res.* **24**, 3189–3194 (1996).

3. Wilson, G. A. & Rannala, B. Bayesian inference of recent migration rates using multilocus genotypes. *Genetics* **163**, 1177–1191 (2003).

4. Faubet, P. & Gaggiotti, O. E. A new Bayesian method to identify the environmental factors that influence recent migration. *Genetics* **178**, 1491–1504 (2008).

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).