Using the IMa Program

By Jody Hey Department of Genetics, Rutgers University

For questions – check out the *Isolation with Migration* discussion group
http://groups.google.com/group/Isolation-with-Migration

What's new since the last update?

- The option to print histograms of migration counts and mean times has been changed. Instead of recording the mean time of migration for each migration parameter and locus, the times of all migration events are recorded. It turns out that using the mean introduces a number of problems.

Table of Contents

Overview	**1**
Running IMa and Runtime Information	**3**
Input File Format	**5**
Output File	**7**
Command Line Options	**13**
Explanation of Command Line Terms	**14**
Suggested Starting Point	**25**
Cautions, Suggestions and Interpretations	**26**
Compilation	**28**
References	**29**
Appendix - miscellaneous runtime errors	**29**

Overview

This document explains how to use the IMa computer program, which implements a method for generating relative likelihoods/posterior probabilities for complex demographic population genetic models. The method is described in Hey and Nielsen (2007. PNAS 104:2785–2790).

IMa applies the Isolation with Migration model to genetic data drawn from a pair of closely related populations or species. The result is an estimate of the joint-posterior probability density function for the model parameters. The name of the program 'IMa' is a sort of acronym for 'Isolation with Migration – analytic", with the 'a' distinguishing the name from a related program that applies a different method and that is called 'IM'.

That basic Isolation with Migration model has 6 demographic parameters, and is described in the "Introduction_to_IM_and_IMa" document. To understand the principals behind the overall approach, it is necessary to read this introductory document. This document explains the basics of using the IMa program.

The method implemented in IMa is described in the paper by Hey and Nielsen (2007) and is historically related to the methods of Kuhner et al., (1995) and of Nielsen and Wakeley (2001). The IMa program shares a number of features in common with the IM program. They both implement Markov chain Monte Carlo (MCMC) methods under the same Isolation with Migration model, with the same parameters (although IMa does not implement the population size change model). They also both assess the posterior probability densities (proportional to likelihoods) for model parameters. However the nature of the results are different, with IMa generating estimates of a function that can be used for a variety of purposes that go beyond what can be done with the IM program. Also the Markov chain simulation in IMa should have better mixing properties than that in the IM program. This is because with IMa does not include the demographic parameters in the MCMC, and so mixing problems associated with correlations between these parameters and genealogies can no longer occur. IMa does still include the splitting time parameter, *t*, so correlations can (and do) arise with this parameter.

Running IMa and Understanding Runtime Information

The program is run by typing and entering a command at a command line prompt. If your computer is running MS Windows, then this will be done in a 'command prompt' window. It is usually simplest to have the program and data files in the same directory (folder), and for the command prompt window to be open in that directory (folder).

The distributed version of the code and the windows executable will handle up to 200 loci and 200 chains. For larger models, values for MAXLOCI and MAXCHAINS in the IMa.h file need to be increased and the program recompiled (see **Compilation).** Data is loaded into dynamic memory, so the actual size of a memory model during the run can be large and hard to predict (see **Compilation).**

IMa is essentially two different programs put together. One part of IMa runs the MCMC simulation that generates samples of genealogies (i.e. samples from the posterior probability density of genealogies, given the data). The second part of IMa builds a function from these trees, and finds the peaks in the surface described by this function.

Every new analysis must begin with running the program in 'MCMC mode' which generates a set of sampled genealogies. Once these genealogies are saved, then the program can be run in the 'Load-Trees' mode, to do additional analyses using the function generated from saved genealogies. For brevity and clarity, hereafter MCMC mode is identified as **M** mode, and Load-Trees mode is identified as **L** mode.

If the program is run in **L** mode, then the only output to the screen is an occasional indication of where the function optimization is at.

If the program is run in **M** mode, then the runtime output, that appears in the command prompt window includes the following:

- The current step number and the current values for the probability of the data, given the genealogies, 'p(D|G)', and the prior probability of the genealogy 'p(G)'. These probabilities are not particularly useful, but it is good to check that they are actual numbers (non-numerical values indicate a floating point problem and the run should be halted).

- The update rates for genealogies and parameters that are included in the MCMC simulation. These are the percentage of proposed updates that were accepted based on the corresponding Metropolis-Hastings criteria. Values are shown for the splitting time, *t*, if it is included in the model, as well as mutation rate scalar parameters. For HKY model loci, the update rate of the kappa term is shown, and for Stepwise mutation model loci, the update rates are shown for the ancestral allele states (A). For genealogies update rates are given for Genealogy (G), Branch Topology (B), and Common Ancestor Time (T) Update.
- The current value of *t* and mutation rate parameters, as well as locus specific values of p(D|G)
- A table of Effective Sample Size estimates and autocorrelations for a the splitting time parameter, *t*, and the common ancestor time of genealogies. Also shown are values for a quantity called L[P] which is just the sum of P(D|G) and P(G), and is a useful overall summary statistic for the state of the chain. This table begins to appear after 100,000 steps or so. The ESS estimates do not begin to become stable and reliable until after about a million steps or so, depending on the data.
- If Metropolis-coupling has been invoked then there are several series of numbers
  - The heating terms for each chain (beta values)
  - The swap rates between successive chains. If these are lower than 5% then they are not doing much good.
  - Actual swap counts between successive chains.

During a run in **M** mode it is a good idea to keep on eye on things at the beginning of a run to be sure that quantities are being updated. However, even if update rates are nontrivial it is quite possible that the chain is mixing poorly. Note that even if things are not mixing well at the beginning of a run, the mixing properties may change a lot as the system approaches stationarity, so it is usually advisable to watch a run for awhile to before deciding whether or not to restart with different terms.

In general the best approach to ensure mixing is to run the program long enough so that there are no obvious trends in the trendline plots and so the lowest ESS value among the parameters is at least 50, *and* to run the program multiple times (with a different random number seed) to see if results are similar.

Input File Format (format is identical for both the IM and IMa programs)

IM requires one input file that contains the data for all loci to be considered. The input file should be prepared in the same operating system that the program will be running in. This is to ensure that the end-of-line character differences between unix/linux and dos/windows (and older Macs) do not cause problems. The format is as follows:

- line 1 - arbitrary text, usually explaining the content of the file
    - After line 1, but before line 2, comments can be included to provide explanatory information. Each line of comment must begin with a '#'
- line 2 - two population names, for populations 1 and 2 respectively, separated by one or more spaces
- line 3 - an integer, the number of loci in the data set
- line 4 - basic information for locus 1. This line contains at least five items separated by one or more spaces
    1. Locus name (no spaces within the name)
    2. n1, the sample size for population 1
    3. n2, the sample size for population 2
    4. the length of the sequence (if SSM model – a number is needed here, but it is ignored, if HapSTR, the length pertains to the sequence portion of the data)
    5. Letter indicating the mutation model (I- IS, H - HKY, S - SSM, J - joint SSM and IS = HapSTR) - if this letter is not included on this line, the default is IS. If SSM (S) or HapSTR (J), the letter is followed immediately (no spaces) by the number of linked STR markers within the locus.
    6. Inheritance scalar - For example: 1 for autosome, 0.75 for X-linked, 0.25 for Y-linked or mtDNA.

7. The mutation rate per year for the locus (not per base pair). This can be left blank, but is needed for estimating parameters on demographic scales. If there are multiple STRs in the locus then there can be multiple mutation rates on this line separated by spaces. If the locus is a HapSTR, then the first mutation rate given applies to the sequence portion of the locus with subsequent values corresponding to STR markers included in the locus.
8. If the mutation rate is given, it can be followed by a range of mutation rates that can be used (with ranges for other loci in the analysis) to set priors on the ratios of mutation rate scalars. The range is entered with an open parentheses, the lowest value, a comma, the highest value, and a closed parentheses (e.g. '(0.00001, 0.00004)'. The range must bracket the rate. For a locus with multiple mutation rates, and multiple ranges, each range follows its corresponding mutation rate immediately on line. See option '-j7' for more explanation.

- line 5 - data for gene copy # 1 from population 1. For this line and all other data lines, the first 10 spaces are devoted to the sample name. The sequence or allele length (for SSM model) begins in column 11 of the file. The sequence for a given sample is given all on one line without gaps. For SSM or HapSTR data, the allele length assumes a step size of 1. This means that data from STRs that are multiples of lengths greater than 1 must be converted to counts of the number of base repeats (e.g. for a dinucleotide 'CACACACACACA' the length would be 6). If the data is for an SSM model locus and there are multiple STRs, then there will be one integer on each line for each STR, separated by a space. If the locus is HapSTR (joint IS and SSM) then the STR data is given on the line, beginning at column 11, followed by the sequence data. For SSM data, as for other types of data, only one gene copy is represented on each line of the data file. This is true even if the original data consists of diploid genotypes. In other words, diploid genotype data must be broken up and listed, with one data line for each gene copy.
- lines 6 thru line (n1+n2 +4) - the remainder of the data for locus 1. Each line contains the data for one sample. The data for population 1 samples are given in

lines 5 thru line (n1 + 4). The data for population 2 begins on line (n1+5) and proceeds to line (n1+n2+4).

- Additional lines for additional loci - If there is more than one locus, then the data for locus 2 begins on line (n1+n2+5) with a line similar to line 4 presenting the basic information for locus 2. The sample names and sample sizes for locus 2 and the inheritance scalars and mutation model for locus 2 need not be the same as for locus 1.

Finally, the last line of data should end with a newline so that the file ends on a blank line.

Here is an example for a tiny three locus data set. In this made-up example the mutation rate per year is known and specified for locus 1, but not for loci 2 and 3.

```
Example data  for IM
# im test data
population1 population2
3
locus1 1 1 13 I 1 0.0000000008 (0.0000000001, 0.0000000015)
pop1_1    ACTACTGTCATGA
pop2_1    AGTACTATCACGA
hapstrexample 2 1 4 J2 0.75
pop1_1    13 34 GTAC
pop1_2    12 35 GTAT
pop2_1    12 37 GTAT
strexample 2 2 1 S1 1  0.00001 (0.000001, 0.00005)
strpop11a 23
strpop11b 26
strpop21a 25
strpop21b 31
```

Output File

A run in **M** mode contains results on the simulation, and depending on the runtime options, results on posterior density estimates, columns of marginal distributions, and rough plots of trendlines and marginal densities. A run in **M** mode also generates a file with genealogy (tree) information. This file has an extension of ‘*.ti’ and can be read by the program in **L** mode.

All options can be invoked for a run in **M** mode. When the program is run in **L** mode, some of the following information (that associated with the MCMC simulation) is not included in the output file.

INPUT AND STARTING INFORMATION

This section lists the starting information from the input file and the command line settings.

MCMC INFORMATION

This section summarizes information on the run, beginning with the length and time of the run and the number of measurements made. These are followed by the highest values observed for the probability of the data, given the genealogy and the parameters, P(D|G, Params) and the highest values for the prior probability of the genealogy P(G). Note that these values are not of much use, though there are times when it might be useful to compare them between runs.

Next is listed a table of the means and variances of the times of common ancestry of the genealogies. These can be useful, and if desired the program can be set to record the entire posterior distribution of common ancestor times. The units on these values are the same as for *t* .

Next are listed the rates at which updates for genealogies and for those parameter in the MCMC simulation were accepted, over the course of the run. For genealogies these include the total update rate (G), the rate at which the branching (B) pattern was updated, and the rate at which the common ancestor time (T) was updated.

Following the update rates, the output file has a table of parameter autocorrelations and ESS (effective sample size) estimates. These are the same as appear on the computer screen during the course of the run. ESS values are estimates of the number of independent points that have been sampled for each parameter. They can be a valuable guide to how well the Markov Chain is mixing and how long the chain should be run for.

However they should be used with caution (see **Assessing the Autocorrelation of Parameter Values**)

If multiple chains were run with Metropolis-coupling, then the next section summarizes the Beta values and observed swapping rates.

PEAK LOCATIONS AND PROBABILITES

This section provides parameter estimates and associated probability densities based on the locations of the highest values of the posterior density function. In all cases the parameter estimates are found by finding the parameter(s) that are associated with the highest probability. This kind of problem, of finding the value associated with the maximum of a mathematical function, is called 'optimization'. There are lots of computational ways to go about the process of optimization, but without knowing just how bumpy the surface that is described by a function actually is, there is no way to be absolutely sure if any particular peak that is found is really a global optimum or a local optimum. This is a big problem for the case of the joint density function. The method that was finally selected, after a lot of trials, was the simulated annealing method that is implemented in the AMEBSA code of Press et al. (1992).

Estimates from Marginal Distributions

All runs will include parameter estimates based on the marginal distributions for each population size and migration parameter in the model. Also shown are estimated 95% confidence limits, based on the standard approach used with maximum-likelihood estimates, of finding those values associated with the log of the likelihood that is equal to the highest likelihood minus 1.92. These calculations go quite quickly, and these marginal functions are not difficult to optimize.

Estimates from Joint Distributions

If called for in the runtime options, this section will also include a table with the joint parameter estimate, $\hat{\Theta}$, for all of the parameters in the model. This can take a long time to calculate because of the maximization routine that is used. In general, if a function is based on 20,000 trees then it might take up to 24 hours or so to complete this part of the analysis (roughly – depending on lots of things). Longer times are needed for larger sets of saved genealogies.

Three sets of maximizations are done: set 1, based on the first half of genealogies; set 2, based on the second half; and 'All' based on all genealogies being used. If there are enough effectively independent genealogies in the sample then sets 1 and 2 and 'All' should give similar results. However if there are not enough genealogies, or they are not sufficiently independent of each other (i.e. the MCMC simulation was not mixing sufficiently well) then they can vary considerably.

If the splitting time parameter is included in the MCMC simulation, then an estimate of the value of *t* that is associated with the joint parameter estimates is also shown. To get these we calculate the probability of the parameter estimate given a genealogy $p(\hat{\Theta} \mid G_i)$ for each genealogy, where G_i is the i$^\text{th}$ genealogy in the set of genealogies used to generate the posterior density function. Then, because each genealogy has been saved together with the value of *t* at that point in the simulation, we can calculate the weighted mean of *t* associated with any value of Θ, including $\hat{\Theta}$:

$$\hat{t} = \frac{\sum t_i \times p(\hat{\Theta} \mid G_i)}{\sum p(\hat{\Theta} \mid G_i)}$$

Nested Models and Log-Likelihood Ratio Tests

If the optimization of nested models are called for, then a table is output that has the parameter values, posterior probabilities, and log-likelihood ratio statistics (2LLR)for all possible nested models. These calculations can also take a long time. Some nested models can be optimized fairly quickly, but there are a lot of possible nested models.

A partial example of nested model output is provided as Table 1 of Hey and Nielsen (2007). The values of 2LLR will be distributed approximately as a χ^2 (chi-square) distribution with the number of degrees of freedom given under the column headed 'df'. However there are two important caveats.

- First, the χ^2 approximation is achieved only in the limit for large data sets with enough independent observations. This approximation held pretty well for simulated data sets as described in Hey and Nielsen (2007), but this was for data sets with a number of loci (from 6 to 25, depending on the example in Figure 2 of that paper). Small data sets are not going to meet this situation, and until more simulation work is done we cannot know just how good/bad the fit is. Furthermore, the departure from a χ^2 distribution, for small data sets, will be in the direction of having a distribution of 2LLR with more variance than for the corresponding χ^2 distribution. This means that statistical tests using the 2LLR statistic will not be conservative, when small data sets are used.
- The degrees of freedom, as given in the table, is not correct for nested models in which one or more parameters are set at the boundary of that parameter. These are models in which one or both migration parameters are set to zero. These models have distributions of 2LLR that are a mixture (see Hey and Nielsen (2007)). They are identified by an asterisk after the number of degrees of freedom.

MARGINAL DISTRIBUTION VALUES AND HISTOGRAMS

Marginal distribution values are calculated and listed, for 1000 points evenly spaced over the prior range of each parameter. Also, when run in **M** mode, the parameters that are included in the MCMC simulation are also listed in these tables. These are basic histogram tables like those produced by the IM program. Before the large table of, a table of summaries is provided for each parameter.

- Minbin - the midpoint value of the lowest bin.
- Maxbin - the midpoint value of the highest bin.
- HiPt - the value of the bin with the highest count (i.e. highest residence time).

- HiSmth - the value of the bin with the highest count, after the counts have been smoothed by taking a running average of 9 points centered on each bin.
- Mean - the mean value of the parameter.
- 95Lo - the estimated point to which 2.5% of the total area lies to the left.
- 95Hi - the estimated point to which 2.5% of the total areas lies to the right
- HPD90Lo - the lower bound of the estimated 90% highest posterior density (HPD) interval. The 90% HPD interval is the shortest span (on the X axis) that contains 90% of the posterior probability. A question mark, '?', is added if the HPD interval did not appear to be contiguous (in which case the HPD estimates are not reliable) which can happen with multiple peaks or if the surface is rough.
- HPD90Hi - the upper bound of the estimated 90% HPD interval.

Command Line Options

Executing the program with only a command line flag of ‘-h’ will write the following list:

```
Command line usage: - upper or lower case letters can be used
-a  steps between recording parameter values from MCMC (default: 10)
-b  duration of burn  (MCMC mode only)
    - if integer, the number of burnin steps
    - if floating point, the time in hours of burnin period
-c  calculation options:
    0 LOAD-TREE Mode - load trees from previous run(s); also requires -r
    1 find the joint surface peak (default is not to find it)
    2 examine nested models
    3 show details of joint optimization results
-d  number of steps between tree saving -MCMC mode only (default 100)
-f  heat mode: l linear (default); t twostep; a adaptive twostep; g geometric
-g1 first heating parameter, effect depends on heating mode (default 0.05)
-g2 second heating parameter, effect depends on heating mode
-h  comment for output file (no spaces)
-i  input file name (no spaces)
-j  run options:
    0  likelihood() functions return 1 - posterior should equal prior
    1  for 4Nu priors (q1,q2,qa) use command-line values as priors
         default: priors = product of command-line values and data estimates
    2  treat inheritance scalars as parameters.(default= input file value or 1)
    3  include ranges on mutation rates as priors on mutation rate scalars
    4  set t to a very large value, mimics two population island model
    5  set t = 0 to mimic one large panmictic population of size 4NAu
-k  with multiple chains (-n) the number of chain swap attempts per step
-l  duration of run / number of trees
  - if in MCMC mode (not loading trees from a previous run)
     - if integer, the number of trees to save. Use with -d.
         - if floating point, the time in hours between outputs. Use with -d
         run continues until file IMrun is no longer present
         in the directory, or if present, does not begin with 'y'
  - if in load-tree mode (using -c0 to load trees from previous run)
     - integer indicates number of trees to load
-m1 maximum migration rate from population 1 to population 2
-m2 maximum migration rate from population 2 to population 1
-n  number of chains (MCMC mode only)
-o  output file name (no spaces) default is 'outfile.txt'
-p  output options:
    0 do not save genealogies to a file (default saves sampled genealogies)
    1 print TMRCA histogram for each locus (MCMC mode only)
    2 print histogram of parameters on demographic scales
    3 print histograms of # migration events and times (MCMC mode only)
    4 print ASCII-based plots of parameter trends in outfile (MCMC mode only)
    5 print ASCII-based marginal curves in outfile
    6 print file with listplots of 2D marginal distributions
-q1 scalar for 4N1u maximum
-q2 scalar for 4N2u maximum (default =  scalar for 4N1u maximum)
-qA scalar for 4NAu maximum (default =  scalar for 4N1u maximum)
-r  base name (no extension) of files with tree data, requires use of -c0
-s  random number seed (default is taken from current time)
-t  maximum time of population splitting
-w  minimum time of population splitting
-u  generation time in years  (default is 1)
-v  window width adjust for t updating (MCMC mode only)
-y  mutation rate scalar for relevant loci - for use with -p2
-z  number of steps between screen output (default 10000) (MCMC mode only)
```

Explanation of Command Line Terms

`-a  steps between recording parameter values from MCMC (default: 10)`

Because the parameter values over the course of the run are highly autocorrelated, there is no need to record values at every step. There is no harm in recording at every step, but recording at longer intervals saves a little time.

`-b  duration of burn`

The Markov chain begins with parameters and genealogies set to values that are consistent with the data but that may be very far removed from values that have nontrivial probabilities. To avoid having the initial conditions dominate the results, the chain should be run until it has reached a point that is independent of the starting conditions. The minimal length of a burn-in chain depends on the data set and cannot be known prior to looking at the results of some runs. In practice, burn-in lengths of 100,000 steps seem sufficient for most data sets unless the SMM model is used. If the value is an integer (e.g. '`-b100000`') the duration of the burn-in chain is the number of steps. If the value is a floating point number (e.g. '`-b1.0`') then a burn-in chain is run for that length of time, in hours.

`-c  calculation options:`

There are various calculation options. Multiple options can be specified at once (e.g. `–c01` invokes options 0 and 1).

`0 LOAD-TREE Mode - load trees from previous run(s)`

This option invokes the **L** (Load-Tree) mode, meaning that the run begins by reading files that contain information on genealogies that were generated in a previous run under **M** (MCMC) mode. Unless an **M** model run is made using the –p0 option (do not save genealogies) then an **M** mode run will generate a regular output file, as well as a file with the extension '*.ti'. For example an **M** mode run that generated output file 'myresults.out' will also generate a file named 'myresults.out.ti'. It is also necessary to tell the program the base name of the genealogy file, so whenever the '–c0' command is used it is also necessary to use '-r' followed by the base name of the '*.ti' file. For example an **L** mode run might

have an output file named 'myresults_Lmode.out' that would read the genealogies saved in 'myresults.out.ti'. Then the command line would include the following flags (among others): -o myresults_Lmode.out -c0 -r myresults.out
This will cause the program to look for all files that meet the following specification: 'myresults.out*.ti'. If there are multiple files with names that fit this, then they will all be used for the analysis. In this way it is possible to use genealogies sampled in multiple separate **M** mode runs (that use the same data sets and runtime commands).

When running in **L** mode it is also important to tell the program what the priors were for the original run, and so it is necessary to specify all of the '-q' , '-m', '-t', and '-w' flags exactly as they were originally specified in the original **M** model run. Also any run options given by the –j flag in the original **M** mode run should be repeated on the command line of the **L** mode run.

`1 find the joint surface peak (default is not to find it)`

This invokes the optimization for the joint posterior density function. This will take hours, possibly days if many tens of of thousands of trees are used.

`2 examine nested models`

This invokes the fitting and testing of nested models. This is also slow.

`3 show details of joint optimization results`

This causes a fuller output of the joint optimization to be output, which allows a user to get a feel for just how well the optimization might be working.

`-d  number of steps between tree saving (MCMC mode only)`

This is the length of the interval (in steps of the MCMC simulation) between the saving of genealogies. The default is 100. If an **M** mode run is desired to result in a specific number of trees, (i.e. '-L' is used with an integer) then the total length of the run will be the burning length ('-b') plus the product of the integer specified by '-L' and the integer specified by '-d' (or the default for –d). If the Markov chain simulation is mixing slowly then it is ok to raise the value specified with '-d'.

`-f  heating mode`

Small single locus data sets often do not require heating, whereas multiple locus data sets or large data sets often do. STR data almost always require heating and metropolis-coupling. If multiple Markov-coupled chains are run, then it is important to have a heating scheme that leads to sufficient rates of swapping of the chains. When there are n chains, they are numbered from 0 to n-1. For chain i, the Metropolis criteria for parameter updating are raised to a power β_i (beta(i)), where $\beta_i < 1$. For all heating schemes chain 0 is not heated and chains 1 thru n-1 have successively greater values of β. Swapping is done randomly among chains with similar heating values, but most accepted swaps are between chains with the smallest difference in heating values. Chains with low values of β will be strongly heated, but will have lower swapping rates with chains that are much less heated. In general, swapping rates for chain 0 that are less than 5% do not seem to be very helpful. For data sets where good mixing is achieved with a small number of chains, rates between 10% and 90% between chain 0 and chain 1, and generally between chains i and i+1, seem to be ideal. For difficult problems, 20 or 30 or more chains are sometimes needed. In these cases it is best to arrange for very slight heating and very high swap rates (> 90%) in the chains closest to chain 0. In these cases the geometric mode for selecting heating levels is recommended (see below).

Trial and error is required in order to find a useful heating scheme. A user may have to try several values, and wait for the program to run for awhile, to see their effect on chain swapping rates. Also, keep in mind that the rate of swapping among chains cannot always be assessed very well when a chain is just starting out, and sometimes one will not know the swapping rate until a chain has run for a million or more steps.

One difficulty with finding an adequate heating scheme is that the swapping rates between successive chains are often quite low, for low numbered chains (e.g. between chain 0 and chain 1), even for slight heating, whereas higher numbered chains often have higher swapping rates. The trick is to get sufficient swapping with chain 0, and yet also have sufficient heating in high numbered chains, so that the heated chains do really explore the parameter space.

Four different heating schemes have been designed: linear, twostep, twostep-adaptive, and geometric. The values used by these functions are given in the command line using '`-g1`', and '`-g2`'.

The linear scheme ('`-fl`') is the simplest and requires only that the user input a value for '`-g1`'. Under the linear scheme, each successive chain receives an additional heating increment. If the heating term is h, then the heating term β_i, for chain i, is given by

$$\beta_i = 1/(1 + i \times h)$$

For example, consider this simple linear scheme for five chains: `-fl -n 5 -g1 0.05`
For more than just a few chains, it is sometimes better to use the twostep scheme ('`-ft`') , which adds an additional multiplier increment of heating, for each chain i>1. This scheme requires two terms, h1 given by the '`-g1`' flag and h2 by the '`-g2`' flag. The formula for the heating term for chain i is

$$\beta_i = 1/(1 + h1 + h1 \times h2 \times (i-1)$$

For example, this can be useful: `-ft -n 5 -g1 0.05 -g2 2`

The program also includes an adaptive twostep scheme ('`-fa`') which changes the heating parameters for the twostep scheme periodically so that the overall rate of swapping between low numbered chains is between 20% and 80%. Values are updated every 1000 steps during the burn-in period, and every 10000 steps during the main run. Changing the heating value during the run violates the Metropolis criterion for chain swapping, unless each time the heating rates are changed is also followed by an extensive burn-in period without swapping. For this reason the adaptive scheme should not be used for primary analyses, but rather for finding useful heating parameters to use under the two step ('`-ft`') model.

The geometric heating scheme ('`-fg`') is also intended to create successively greater heating levels for successively higher numbered chains. Two terms are required. h1, given by '`-g1`' on the command line, which specifies the degree of non-linearity, where 0<h1 (also usually h1 <1). The case of h1=1 gives a linear decline and h1<1 gives β

values that decline slowly for low numbered chains, and fast for high numbered chains. The second term, h2 given by '`-g2`', is the value of β for the highest numbered chain. The formula is

$$\beta_i = 1 - \frac{(1-h2)\times i \times h1^{n-1-i}}{n-1}$$

To determine h1, on the basis of a particular heating value for any specific chain i, β_i, the following formula can be used

$$h1 = \left(\frac{(1-\beta_i)(n-1)}{(1-h2)i} \right)^{\frac{1}{n-i-1}}$$

For example, with h2, the heating level of the highest numbered chain set to 0.6, and a desired level of heating for chain 1 of 0.999, then with 15 chains, h1 should be set to 0.773. This will create a series in which β drops very slowly among the lowest numbered chains, and more precipitously for the highest numbered chains. A useful starting point is

```
        -f g  -n 6 -g1 0.8 -g2 0.9
```

Note that some data sets require large numbers of chains and very small β values for low numbered chains. In these cases it is best to play around with the geometric scheme.

`-g1 first heating parameter`

See the text for '`-f`'

`-g2 second heating parameter`

See the text for '`-f`'

`-h  comment for output file (no spaces)`

This is if you want to add some explanatory text to appear in the output file. It is optional.

`-h  inheritance scalars as parameters`

The default ('`-h0`') treats inheritance scalars as constants as given in the input file. Setting '`-h1`' causes these terms to be treated as parameters that are sampled on a wide log scale centered at 1, much as are the mutation rate scalar parameters (Hey and Nielsen 2004). This option has no meaning when data includes a single locus, and is not likely to be informative unless there is a large amount of data on polymorphism within populations.

`-i  input file name (no spaces within the name)`

For example, `-i mydatafile`

If no input file name is specified, the default is infile.txt. If no file of this name is found, the program stops.

`-j run options`

There are various model options. Multiple options can be specified at once (e.g. -`j134` invokes options 1,3 and 4).

`0 likelihood() functions return 1`

Causes all likelihood functions to return a value of 1. This makes the analysis not dependent on the data and, if the program is working properly should return the prior distributions in the resulting histograms. This is mostly used for debugging, but it can be useful for checking the priors on things that are not explicitly laid out. For example, when histograms are plotted for time and number of migration events, it is useful to do a run first with the –j0 option to see what the priors are for these distributions. Invoking this option causes the upper bounds on the prior distributions for the theta parameters to be set directly (see the '`-j1`' flag).

`1 for 4Nu priors use command-line values as priors`

Sets the manner for establishing the upper bound on the theta parameters. By default, without invoking this option, the programs sets an upper bound on each population size parameter by roughly estimating 4Nu for each population and then multiplying this times the scalar entered on the command line with the '-q' flags (see -q1, -q2, and -qA'). If –j1 is invoked then the values specified with these –q flags are used directly as the upper bounds of the prior distributions. The default can save a bit of time by picking values that are based on the data and thus avoiding having a prior maximum that is too low, however if you know that you want a particular upper bound then it can make sense to use this option.

`2 treat inheritance scalars as parameters`

NOTE – this has not been tested. The default is to treat inheritance scalars as constants as given in the input file (or as 1, if not specified in the input file. Setting '`-j2`' causes these terms to be treated as parameters that are sampled on a wide log scale centered at 1, much as are the mutation rate scalar parameters (Hey and Nielsen 2004). This option has no meaning when data includes a single locus, and is not likely to be informative unless there is a large amount of data on polymorphism within populations.

3 `include ranges on mutation rates as priors`

If mutation rate range priors are included on in the input file, for multiple mutation rates, then the ratios of these limits are used as limits on the ratios of the mutation rate scalars. If two or more loci in the analysis have a prior range, then this allows the prior information on mutation rates to be included in the analysis and to shape the findings. Great care must be taken in selecting a prior range, as it must include all of the sources of uncertainty in the mutation rate. One possibility is to use a likelihood approach. For example, consider a locus that is going to be used in the analysis, and suppose that there is information on the mutation rate for that locus based on a comparison between other species (not in the analysis). Let x be the amount of observed divergence observed between those species, and let τ be the number of years since those species are thought to have separated (in this example assume that τ is large and does not include a component of coalescent variation from the ancestor). Then develop the math for the probability of x, $p(x\,|\,\tau,r)$. Now decide what the prior range should be on τ (based on whatever information was used to get τ - this will depend very strongly on the source of τ). Then integrate $p(x\,|\,\tau,r)$ over the prior range of τ to get $p(x\,|\,r)$ (you can assume a uniform distribution of τ, or something else – whatever makes the most sense), and define this quantity as the likelihood, $L(r\,|\,x)$. Now solve this expression for the most likely value of r (this will be the value of r you enter into the IM input file) and for the 95% confidence intervals (using standard likelihood assumptions) and use this interval as the prior range on r in the input file.

4 `set t to a very large value, two pop island model`

Set t to a very large value and causes t not to be included in the MCMC in the model. This makes the model a two-island equilibrium model.

5 `set t = 0 to mimic one large panmictic population`

Set $t = 0$. When invoked, the only parameter that is updated is θ_A. This is a single, constant size population model.

`-k  with multiple chains (-n) the number of chain swap attempts per step`

With metropolis-coupling invoked and the number of chains greater than 2, this sets the number of attempts at chain swapping per step. With many chains this can even out the apparent rate of mixing for chain 0 and the autocorrelations. Since chain swapping proceeds fairly quickly, it is easy enough to have multiple swaps. For example for n chains, the number of swaps can also be n or more. The maximum value is n(n-1)/2.

`-l  duration of run`

In MCMC (**M**) mode: If an integer (e.g. '`-l2500000`') this specifies the number of genealogies to save. If a floating point value, this sets the duration of the chain in hours between printings of a results file. In this case, the program will run continuously, as long as the IMrun file is present in the same directory. IMrun is a tiny file that contains the word "yes". Its only purpose is to be present when you want a run to continue, and to be absent when you don't. When using the '`-l`' flag with a floating point value, you control the run length by deciding when to remove the IMrun file from the directory in which the program is running. At each printing of the results file, the previous results file is renamed to *.old. By taking a look at the distributions in the output files, while the program is still running, one can run the program as long as desired. To halt the program at the end of the current interval, rename the IMrun file or edit it so that the first character is not a '`y`'. Be careful not to have the output file open (e.g. in an editor or a spreadsheet program) when it is time for the program to write to the output file, as the program will probably crash if this occurs. To view the output file, while the program is still running, it is usually best to rename it or to copy it to a new file.

In Load-Trees (**L**) mode: An integer given with the –l flag will specify the number of trees to load. If this integer is greater than the total number of genealogies contained in all files to be loaded (as specified by –r), then all trees in those files will be used in the analyses. If this integer is less, then the specified number of genealogies will be sampled from the total (with even spacing among all genealogies available in the files). This option allows a user to sample the results from a long **M** mode run, that generated a large number of saved genealogies. This is useful because the joint parameter density is slow to optimize for large numbers of genealogies.

`-m1 maximum migration rate from population 1 to population 2`

This sets the upper limit of the prior distribution of m_1 . To remove this parameter from the model, specify '`-m1 0`'. If the maximum migration rate is set to 0 for one migration parameter, it must also be set to zero for the other migration parameter.

`-m2 maximum migration rate from population 2 to population 1`

This sets the upper limit of the prior distribution of m_2 . To remove this parameter from the model, specify '`-m2 0`'.

`-n  number of chains (MCMC mode only)`

The number of chains to run under Metropolis-coupling. The default is 1 (i.e. no Metropolis-coupling) and the maximum is 10. For large numbers of chains, only chains with 7 steps of each other are considered for swapping.

`-o  output file name (no spaces within the name)`

The default is 'outfile.txt' however it is usually best to provide a more explanatory name.

`-p  output options:`

There are various output options. All can be given at once (e.g. `–p452` invokes options 2,4 and 5).

`0 do not save genealogies to a file`

This prevents the saving a file with genealogy information.

`1 print TMRCA histogram for each locus`

M mode only. This prints a table in the output file of the distributions of times of the most recent common ancestor (TMRCA) for each locus. The units on these times are the same as for the *t* parameter (i.e. mutation rate times time).

`2 print histogram of parameters on demographic scales`

This prints values and histograms for the theta and time parameters with the scales adjusted by the generation time and mutation rate, as given in the input file and runtime. If one or more of your loci have mutation rate estimates provided in the input file, then this option is very useful as it generates histograms on the relevant scales. For time to be scaled properly it is necessary to specify the number of years per generations (see '-u').

`3 print histograms of # migration events and times`

M mode only. This prints histograms that estimate the posterior distribution for numbers and times of migration events. Histograms are provided in each direction for each locus. The histograms for the time of migration rates have the same units as t and are based on just those genealogies that had at least one migration event. The times of all migration events are recorded (in contrast to earlier releases of this program in which mean migration times were recorded). For each locus the following columns are provided

- m1#, the number of migration events into population 1
- p, the estimate of the posterior density for m1#
- m1time, the time of migration events into population 1 in the genealogy. The times of all migration events are recorded.
- p, the estimate of the posterior density of m1time
- m2#, the number of migration events into population 2
- p, the estimate of the posterior density for m2#
- m2time, the of migration events into population 2 in the genealogy. The times of all migration events are recorded.
- p, the estimate of the posterior density of m1time

`4 print ASCII-based plots of parameter trends`

M mode only. This prints plots of parameter value trends. These plots are ASCII-based and crude, but are still quite useful for assessing how well the parameters are mixing. A plot is also provided for Log(P) = Log(P(Data|Genealogy)) + Log(P(Genealogy)), which is useful for an overall sense of how well the chain is mixing.

`5 print ASCII-based marginal curves in outfile`

Print plots of the marginal density estimates of parameters. These plots are ASCII-based and crude, but are useful for getting a rough idea of how things look and the relation of peaks to the bounds of the prior distribution. For population size and migration parameters, the plots are based on the estimated marginal density functions. If this option is invoked in **M** model then plots are also provided for the recorded values for those parameters in the MCMC simulation.

`6 print file with listplots of 2D marginal distributions`

This option produces a file with an '*.m2d' extension that contains plotting information to make a contour plot for each pair of parameters. Examples are given in Figure 4 of Hey and Nielsen (2007). Each pair of parameters is representing by a table specification with and a ListContourPlot[] plot command. If the contents of this file are cut and pasted into a Mathematica (Wolfram Research, Inc.) notebook and executed, there should be output a set of two dimensional contour plots.

`-q1  scalar for theta1 maximum`

If the '`-qu 1`' option is not used, the program generates a very crude estimate of the population mutation parameters for populations 1 and 2. This command line flag sets a scalar to be multiplied by that value for population 1. This sets the upper bound of the prior distribution for the population mutation parameter for population 1. A useful value for starting with many data sets is 10 (i.e. '`-q1 10`'). If the '`-qu 1`' flag is used, then '`-q1`' sets the actual upper bound for theta1.

`-q2  scalar for theta2 maximum`

As for '`-q1`'. If no value is specified for pop 2, then the value for '`-q1`' is used.

`-qA  scalar for thetaA maximum`

As for '`-q1`'. If no value is specified, then the value for '`-q1`' is used.

`s   random number seed (default is taken from current time)`

An integer (e.g. '`-s 123`'). Useful in case of the need to exactly repeat a run or when starting multiple instances of the program at about the same time and you don't want the program to use the same machine time to seed different runs of the program.

`-t  maximum time of population splitting`

The upper bound on the prior distribution for *t*.

`-w  minimum time of population splitting`

The lower bound on the prior distribution for *t*. Zero is the default.

`-u   generation time in years  (default is 1)`

Specify the generation time. This is only needed if you want the population size parameters rescaled and you have used print flag '`-p4`'. This also requires that at least one locus have a mutation rate given in the input file.

`-v  window width setting for t updating`

Adjust the window width for updating of *t* . Do not confuse this with the upper bound of the prior distribution for *t.* The window width only concerns the way random values are picked for the updating, and this option will only rarely be used. With multiple loci it can be necessary to reduce the window width in order to have reasonable update rates. The formula for the width is (*t*_max - *t*_min)/(5 + v × #loci), v is the value read in using this command line flag (default is 0). Sometimes the default level of mixing for *t* is too low and it can help to reduce the window size.

`-y  mutation rate scalar for relevant loci - for use with -p2`

The calculations of distributions on demographic scales (-p2) requires the geometric mean of mutation rates (on a per year basis). Ordinarily, for data files that specify mutation rates, this value will be automatically calculated during an **M** mode run. However the user can also specify a value using the –y option. Also, if it is desired to use the –p2 option during an **L** mode run, then it will be necessary to use this option (using for example, a geometric mean mutation rate calculated during a previous **M** mode run).

`-z  the number of steps between output to the monitor.`

The default is 10,000, but if the program is quite slow for your data, and you want to look at things immediately, it is useful to set it to a small number (e.g. 100 or 1000).

Suggested Starting Point

To get things started you might try typing the following line at the command prompt and hitting enter:

```
ima -i mydata -o mydata.out  -q1 10 -m1 10 -m2 10 -t 10 -
b 100000 –L10000  -s123  -p45
```

This will cause the program to do a burn-in of 100000 steps followed by a run of 10 million steps (10000 trees, and the default interval of 100 steps between sampled trees). It will generate an output file called mydata.out and a file of genealogy information called mydata.out.ti . It will also generate trend and curve plots (-p45). If good MCMC mixing is observed (based on the trendline, and estimated ESS values), then a subsequent run can be done in **L** model to assess the joint parameter estimate. The corresponding command line for **L** mode for this would be:

```
ima -i mydata -o mydata_Lmode.out –r mydata.out –c012 -q1
10 -m1 10 -m2 10 -t 10 –L10000 -s123 -p5
```

This run will take a number of hours, perhaps a full day, because it invokes the optimization of the joint parameter surface and the fitting of all possible nested models. There is little point in running this unless you have good reason to think that the markov chain has worked well and that the mydata.out.ti file has a good sample of genealogies.

Both of these runs will generate output files that include marginal tables with 1000 parameter values given over the range of each parameter. These allow plotting of the marginal densities better than is afforded using –p5. However to properly look at curves the output file should be opened in a spreadsheet program and the histograms selected and plotted. A spreadsheet (IMchart.xls)is provided in the distribution package.

Cautions, Suggestions and Interpretations

One of the greatest challenges is knowing whether the chain is mixing sufficiently. It is possible to have seemingly high update acceptance rates for the parameters, and yet still have a slow rate of mixing. In cases like these, it is likely that a low rate of genealogy updating is the culprit. Check the rate of genealogy updating, and especially the rate of branching pattern updating (column identified with ‘B%’ in output file). For runs of a couple million or more steps, the ESS estimates can be taken as a rough guide to how many effectively independent observations have been made for each parameter.

A minimum of three runs will be needed to have a rough idea of how well the program is working and of what the marginal distributions look like. The first run is required to assess mixing and to find a range of prior distributions that include all or most of the range over which the posterior density is non-trivial. If things are well behaved and you have a lot of data then a run will probably generate curves that rise from zero, peak, and then fall to zero within the range for each of the basic demographic parameters. If it looks like the posterior distributions are fully contained within the bounds of the prior distributions, and if the observed maxima for any distributions are far to the left of the upper bounds, then the parameter maxima can be reduced for subsequent runs. For the second and third runs the priors can be adjusted as a function of what was observed in the first run. Both of these runs should be long and started using identical settings but different random number seeds. You will have a pretty good idea that the chains are sufficiently long if all ESS values are high and if both runs have generated similar distributions.

In cases where the marginal posterior distributions are sampled well, and where the estimated probabilities at the margins of the prior distribution approach zero (i.e. the histogram drops to zero at the tails), the parameter estimates (based on the locations of the peaks) are also maximum-likelihood estimates. The reasons are that the prior distributions are flat (i.e. all values in the range are equally probable) and because we expect in these cases that extending the prior of the parameters to large values, even to infinity, would not change the posterior distributions.

If the data set is small, and sometimes even if not, it can happen that a large portion of the likelihood surface is very flat over the range of some parameters. In particular it is not uncommon to have high, flat likelihoods for high values of *t* and θ_A. One may find for example in the curve for *t* a sharp peak at a low value, and then a plateau that extends indefinitely to the right at an even higher value. This is awkward because the highest likelihood appears to be associated with an infinitely wide range of parameter values. In these situations, the data does not contain enough information to identify the model, and the results should not be relied upon.

It is tempting in situations with multiple peaks, or where the curve shows a distinct peak at a low value, and a plateau at ever increasing high values, to do more runs with a lower upper bound on the prior for that parameter so as to exclude the area to the right of the distinct peak. An example of this is shown, for the *t* parameter, in Hey (2005). This kind of changing of the prior, to get more readily interpretable plots, is a reasonable thing to do if the altered prior range makes sense given other information (outside of the data). However, in the absence of other information upon which to constrain the prior, then changing it simply because the plots don't give a clear picture, cannot really be justified.

Five things to keep in mind:

1. You may not have enough data to estimate so many parameters (you can try a simpler model with a reduced parameter set).
2. It is critically important that you assess convergence by doing multiple long runs, and by monitoring ESS values and trendlines. If you don't, then your results are probably wrong, possibly very wrong.
3. The prior distributions can indeed be used to reflect your prior expectations. If you think that values of *t* (or any parameter) greater than a certain value should not be open for consideration, then set the maximum of accordingly.
4. Finally - everything depends on the data. Your data may present new challenges, and provide new insights, that have not been anticipated. They may also require runtimes of days, weeks, or quite possibly much worse, in order to achieve convergence.

Compilation

A Windows executable is provided (ima.exe). This has been compiled under Microsoft Visual C++ 6.0. The source code is available if you need to change program constants or need to compile for another computer system. In particular, large data sets or data that requires extensive Metropolis-coupling may require that MAXLOCI and MAXCHAINS be increased in the ima.h header file.

It should be possible to compile for unix/linux. When using a non-Windows compiler and OS it is important to make sure that the input file has the appropriate end-of-line characters for that OS. Failure to do this will probably cause a cryptic crash. On a non Microsoft compiler, the compilation requires inclusion of dirent.h. Source code is provided, and it is worth trying a simple command line compilation, e.g:

```
cc  build_genealogy.c  calc_prob_data.c get_data_initialize.c  ima_main.c
output.c  surface_call_functions.c  surface_search_functions.c
swapchains.c treeprint.c  update_genealogy.c  update_mc_params.c
utilities.c -lm  -o ima
```

This program and the source code files are copyrighted by Jody Hey and Rasmus Nielsen. You may modify them as needed to recompile for different computers, or with different runtime constants, as needed to analyze your data. The source code may not be incorporated into other programs without permission from the authors .

References

Hey, J. 2005. On the Number of New World Founders: A Population Genetic Portrait of the Peopling of the Americas. PLoS Biol 3:0965-0975.

Hey, J., and R. Nielsen. 2004. Multilocus methods for estimating population sizes, migration rates and divergence time, with applications to the divergence of *Drosophila pseudoobscura* and *D. persimilis*. Genetics 167:747-760.

Hey, J., and R. Nielsen. 2007. Integration within the Felsenstein equation for improved Markov chain Monte Carlo methods in population genetics. PNAS 104:2785–2790.

Kuhner, M. K., J. Yamato, and J. Felsenstein. 1995. Estimating effective population size and mutation rate from sequence data using Metropolis-Hastings sampling. Genetics 140:1421-30.

Nielsen, R., and J. Wakeley. 2001. Distinguishing migration from isolation. A Markov chain Monte Carlo approach. Genetics 158:885-96.

Press, W. H. 1992. Numerical recipes in C : the art of scientific computing. Cambridge University Press, Cambridge [England] ; New York.

Won, Y. J., and J. Hey. 2005. Divergence population genetics of chimpanzees. Mol Biol Evol 22:297-307.

Appendix

```
miscellaneous runtime errors
1  cannot open data file
2  cannot create file to hold parameter values, printed at intervals
3  no matching .ti file found as called for using -c0 and -r  flags
5  cannot create file
6  cannot create treefile
7 cannot create output file
8  input and output file names are identical
11 equilibrium migration model, but migration set to 0
```

```
12  too few chains for heating model
21 problem w/ number of loci indicated in data file
22 cannot open data file
30 - to much migration called for - reduce maximum value of
     migration parameter
41 error reading data
42 error in data
51 model not recognized in changeq
71 - problem in updating tree
72 - problem with root
73 - problem in updating migration
74  - problem in treeweight,  n0 vals don't add up
75  - problem in treeweight,  n1 vals don't add up
79 - no updating of tmrca
80 - formatting of input file causes wrong lines to be read as data
81  data not compatible with infinite sites model
82  data not compatible with infinite sites model
83  data not compatible with infinite sites model
84 - sample sizes do not have add up for one locus
91  likelihoods do not add up for stepwise model
92 - no trees loaded
101 - error in gamma functions "a too large, ITMAX too small in gcf"
102 - error in gamma functions "x less than 0 in routine gser"
103 - error in gamma functions "a too large, ITMAX too small in
     routine gser"
104 - error in gammaq functions "Invalid arguments in routine gammp"
105 "bad arguments in expint"
106 "series failed in expint"
107 "continued fraction failed in expint"
108 - error in gammap functions "Invalid arguments in routine gammp"
111 - error in NR functions
```