Introduction to the IM and IMa computer programs

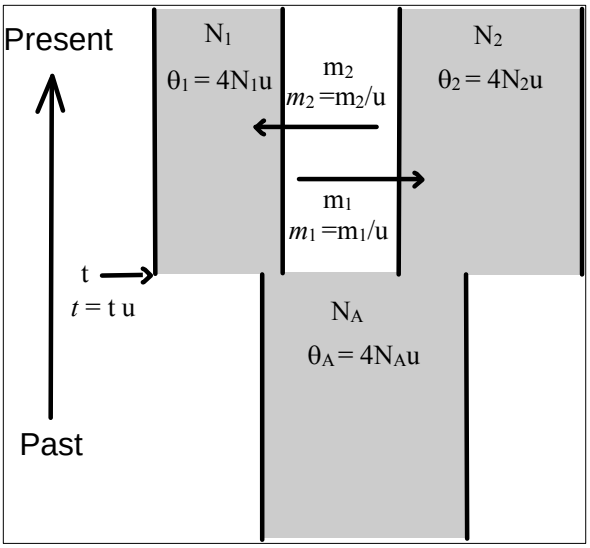
By Jody Hey Department of Genetics, Rutgers University

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

Table of Contents

Overview

This document explains the general principals and workings behind the IM and IMa computer programs, both of which apply the Isolation with Migration model to genetic data drawn from a pair of closely related populations or species. These programs were originally described in Hey and Nielsen (2004) and Hey and Nielsen (2007) respectively. The two programs differ in fundamental ways regarding how model parameters are estimated, however both are based on Markov chain Monte Carlo simulations of gene genealogies. Both IM and IMa use the same input file format. This introductory document provides an overview of the issues that are common to both programs.



There are six demographic parameters in the full two-population Isolation with Migration model, with additional mutation parameters added with multiple loci. Two challenges arise when applying such a full model: (1) obtaining good estimates of the marginal densities; and (2) interpreting the results. With large numbers of parameters the method has vast potential to exceed our intuition and thus to provide estimated distributions that we have not expected.

This is figure 1 from Hey and Nielsen (2004). With six parameters it can capture many of the phenomena that can occur when one population splits into two: the splitting event may have been long ago or recent; the ancestral and the two descendant populations may differ in size; there may have been gene exchange during the time since population splitting; and this gene exchange may have occurred more in one direction than the other. Please note the direction of migration in the figure, in which m_1 pertains to the movement of genes from population 1 to population 2. It is important to understand that this is in the coalescent, meaning back into the past. In other words, in the conventional sense of time moving forwards, m_1 pertains to genes moving from population 2 to population 1, and m_2 pertains to genes moving from population 1 to population 2.

The Isolation with Migration model differs sharply from the general family of models in which populations have been exchanging genes for an indefinitely long period of time. Such 'island models' or 'stepping-stone' models assume that the pattern of variation within and between populations is at equilibrium between the counteracting forces of mutation, genetic drift and gene exchange. In this way the Isolation with Migration model should be more appropriate for the analysis of populations that have recently separated.

The overall approach that is represented in these programs has many complexities and for this reason these programs can not be used to quickly obtain an answer to a particular question. Rather than being 'plug and chug', the programs are tools for analyzing data.

Use of the programs, without understanding the methodology will almost certainly lead to results that are not interpretable or that are very misleading. It is also useful to keep in mind that the programs are not the same thing as the Isolation with Migration model. The model is a theoretical idea, and the program implements one way to fit the model to the data.

Key References

Details and examples are explained in the primary references.

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

Hey, J. 2005. On the number of new world founders: a population genetic portrait of the peopling of the Americas. *PLoS Biol* 3:e193.

Hey, J., Y.-J. Won, A. Sivasundar, R. Nielsen, and J. A. Markert. 2004. Using nuclear haplotypes with microsatellites to study gene flow between recently separated Cichlid species. *Mol Ecol* 13:909-919.

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.

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

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

Assumptions

The major overall assumption is that the history of a sample from two populations can reasonably be described by an Isolation with Migration model. This means that there should not be other populations that are more closely related to the sampled populations than they are to each other, and it means that there should not be other unsampled populations exchanging genes with the sampled populations or their ancestor. The other major population genetic assumptions underlying the method are as follows:

- Selective Neutrality. The method assumes that the variation within the data set is neutral (i.e. not affected by directional or balancing selection).

- No Recombination Within Loci. The method assumes that there has not been recombination within the gene, or genes, that are being studied since the time of common ancestry of the gene copies included in the study. For loci that might have had recombination it is generally a good idea to check the data for obvious signs that recombination has shaped the pattern of variation, such as by using the four-gamete test (Hudson and Kaplan 1985). This test which can be applied by eye or using computer programs such as SITES or DnaSP. See Hey and Nielsen (2004) for detailed discussion of the recombination issue.
- Free Recombination Between Loci. The method treats different loci as having segregated independently over the time since the common ancestry of the gene copies. In practice this does not require that loci be on separate chromosomes, but rather that multiple recombination events would have occurred between loci more recently than the most recent times of common ancestry among samples for each locus. Often this will mean that even loci that have only a low rate of crossing over between them, per generation, are effectively unlinked over longer time frames.
- Mutation has Followed the Model Applied to the Data. The IM program includes implementations of three mutation models. Each model invokes assumptions about the nature of the mutation process.

Mutation Models

Three different mutation models are represented in the program. Also the program includes two types of compound mutation model, in which a locus has multiple portions each with its own mutation rate.

- The Infinite Sites (IS) model (Kimura 1969). Under this model every mutation that has occurred in the history of a sample of sequences occurs at a different place in a DNA sequence. While this may seem unrealistic, it is actually a highly applicable model for many DNA sequence data sets, particularly for nuclear genes. Genes from the nucleus generally experience mutation rates on the order of 10^{-9} per base pair per generation, which means that one does not expect there to be multiple mutations per base pair along a lineage (e.g. a branch on a gene tree), unless that branch is very many generations long. Because the Isolation with Migration model

is intended for relatively recent cases of population splitting, the genealogies of loci that are analyzed will usually have depths of the same order as are found within populations. It is precisely in such contexts, where polymorphic sites tend to have low density along a sequence, that the IS model is ideal. Also, keep in mind that one can check to see if the IS model roughly applies. If it does, there will not be any polymorphic sites with more than two base pairs segregating in the sample. Also, if the IS model applies *and* there has been no recombination, then the data for each locus fall on a branching tree with zero homoplasy.

- The Hasegawa-Kishino-Yano (HKY) model (Hasegawa et al. 1985) was applied to the Isolation with Migration model by Palsbøll et al., (2004). It is a general model that allows for multiple substitutions, with different rates of transitions and transversions as well as unequal frequencies of the four nucleotides. This is a useful model for when you want to allow for multiple substitutions, such as when using mitochondrial DNA sequence data. It is important to recognize that the model does not accommodate different underlying substitution rates at different positions in the sequence.
- The Stepwise Mutation Model (SMM) (Kimura and Ohta 1978). This is a model that can be applied to allelic variation (as opposed to sequence, or haplotype variation) in which each mutation causes an allele to increase or decrease by one step on whatever scale the alleles are being measured. Microsatellite loci (aka Short Tandem Repeat loci, or STR loci) experience high mutation rates to different numbers of repeats in an approximately stepwise manner (Estoup et al. 2002). The SMM model is implemented in a manner that follows Wilson and Balding (Wilson and Balding 1998). An important question, that needs to be investigated, is whether the ways that STR loci do *not* fit the SMM model, lead to important biases when analyzed assuming the SMM model.
- Compound Locus Models. These programs also implement two kinds of models in which a locus has multiple parts. One of these is a model in which a locus includes multiple completely linked STRs, each of which follows the SMM model. An example of this would be data from human Y chromosomes, in which each individual has been genotyped at a series of STRs, all of which are perfectly linked because of the absence of recombination on the Y. A second compound model is

for loci that include a DNA sequence portion (that evolves under the IS model) and one ore more STRs (each of which follow the SMM model). Compound loci with both sequence and STR will have both high- and low- mutation rate components and offer the potential for increased resolution for recent splitting events. Inspired by Joanna Mountain's use of compound loci, that she called SNPSTRs (Mountain et al. 2002), we call these loci HapSTRs (Hey et al. 2004).

Understanding Markov chain Monte Carlo (MCMC)

Each program begins by simulating a genealogy (i.e. a gene tree) for each locus in the data set, and then runs a simulation in which each genealogy is repeatedly updated to a new value following specific criteria (see references for details). Under the method of Nielsen and Wakeley (2001) (as implemented in IM) the state space of the simulation also includes all of the model parameters. Under the method of Hey and Nielsen (2007) (as implemented in IMa) the state space does not include population size and migration rate parameters (though it does include the splitting time parameter, and mutation scalar parameters). At intervals, a record is made of the current state of the simulation, and over the course of a sufficiently long run the distribution of recorded values can be expected to approximate the posterior probability density of those values. This is true for whatever is being recorded from the simulation - whether it be genealogies (as in IMa) or model parameters (as in IM). To put it another way, when the program is up and running it is generating random samples from the desired posterior probability distribution.

The simulation is of a type called Markov chain Monte Carlo (MCMC). There is a large literature on this, and methods like this one have become commonly used for population genetic and phylogenetic problems. It is suggested that users have a good general understanding of the principles involved (better than what is provided here), if they are going to be doing any extensive analysis.

The length of the Markov chain simulation is measured in steps. Each step is one iteration through the routines that pick new values for genealogies and parameters that are included in the state space of the simulation. Each new value that is considered is evaluated in

terms of the appropriate Metropolis-Hastings criteria, and either the new value is accepted or it is rejected (in which case the old value is retained). Each quantity is updated will have yield an acceptance rate, which is the proportion of proposed values that are accepted. If acceptance rates are too low, then the simulation cannot explore the state space.

How Long to Run the Program

The general answer to this question is the same as for any MCMC method, and has two parts, (1) reaching stationarity and (2) convergence of the sample to the stationary distribution. The stationary distribution is just the distribution that we want to estimate with our samples of parameter values that are drawn from the Markov chain. The speed at which the program achieves both stationarity and convergence depend on how well the Markov chain explores the space of genealogies and parameter values. If it moves around fairly well and quickly ('mixes' well) then both processes will usually happen quickly. However if the Markov chain is slow to explore the space, then it can take awhile to reach stationarity and possibly a very long time for the sample to converge on the stationary distribution.

First, before you can get started recording values from the simulation, you must run the program until the parameter values and genealogies in the simulation belong to the actual stationary distribution. The initial running of the program to get the parameters and genealogies to the point where they fall within the stationary distribution (i.e. the posterior distribution) is called the 'burn-in'. When the program starts it begins by using randomly selected parameter values and random genealogies, and it is quite likely that the probabilities of the genealogies, given the data, and the probabilities of the parameters, given those genealogies, are very low - so low that it is not really fair to consider them as being drawn from the posterior distribution that is to be estimated. However you can't simply tell whether or not this is the case just by checking the probability calculations because you don't know the true distribution. Usually one does a preliminary run to see how strongly parameter values are autocorrelated over the course of the run (more on this below), and this can give you an idea of how long the burn-in should be. The goal is to pick a burn-in time that allows the parameter values and genealogies to move to a point such that the values are effectively independent of those at which the program started. There is

no problem in having a longer burn-in than the minimum needed. Also, if the burn-in is too short usually this will be because the chain mixes slowly, and in these cases convergence is going to be a big problem. So for the most part, burn-in is not a big issue, and if it is a big issue it is usually also the case that convergence is an even bigger issue. The most common situation where burn-in is a problematic issue is when the starting genealogy that is simulated is far removed from genealogies with a reasonable fit to both data and to parameter priors. This can happen particularly for SMM model data, especially compound STR data, and in these cases the possible need for a very long burn-in should be anticipated.

Turning to the matter of convergence, by which is meant the approach of the observed distribution of samples, that are saved over the course of the run, to the true posterior probability (also called the 'stationary' distribution – see above). The importance of convergence cannot be underestimated, as a lack of convergence means that the program is giving you the wrong answer. The program will generate some distribution, and you can plot it. It may look nice, but it could be completely wrong.

How do you know if you are getting a good estimate of the true distribution? There are two general kinds of approaches, neither of which are foolproof. One is to see if you get the same distribution from multiple different runs that are identical except for starting random parameter values. The other is to observe over the course of a run how well a chain seems to be mixing. In these programs this is done by plotting recorded values over the course of the run and by measuring how these values are correlated with themselves (i.e. autocorrelated) over the length of the run. If the plotted values show trends over long portions of the run, or if autocorrelations persist for a large number of steps, then this means the state space is being explored slowly, in which case longer runs are required.

Metropolis Coupling of Markov chains

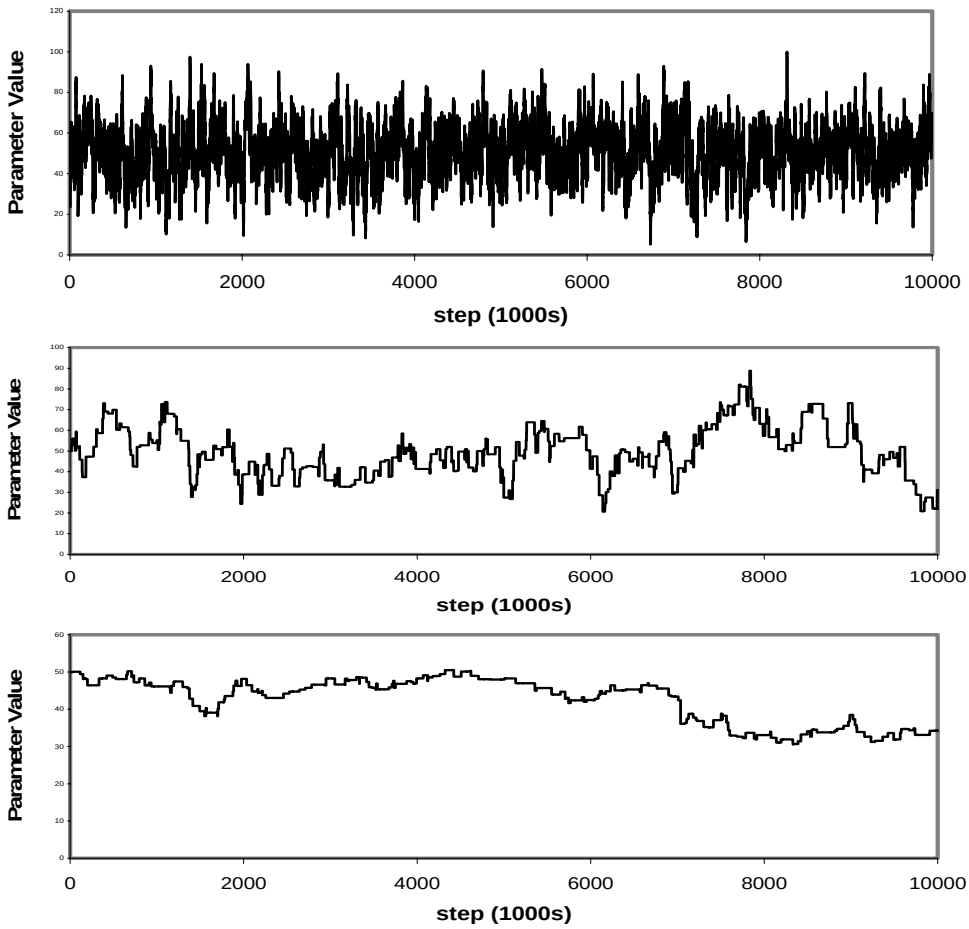
To improve mixing, and thereby convergence, the programs can optionally implement a Metropolis-coupled version of the algorithm in which multiple chains are run simultaneously, with all chains but one having heated stationary distributions (Geyer 1991). These heated chains will not individually return the correct posterior distributions but they

will explore the state space far more quickly than will the non-heated chain. Increased mixing in the non-heated chain is obtained by symmetrically swapping parameter and genealogy states between chains at rates determined by a Metropolis criterion that is a function of the difference in overall probabilities between the chains and the difference in heating values of the chains (Geyer 1991). For a simulation with Metropolis coupling among k chains, each chain will be approximately a fraction $1/k$ as long as would a single chain run for the same length of time. The advantage gained is that the overall rate of mixing on the primary chain may be vastly improved. In practice we have found this method solves the difficulties of inadequate mixing that arise sometimes with datasets that include multiple loci.

Assessing the Autocorrelation of Recorded Values

Although a Markov chain with properly specified Metropolis-Hastings update criteria will provide samples from the desired distribution, these samples will not be independent of each other. This is another way of describing the difficulty of ensuring convergence, as convergence would not be such a problem if we had a method that was assured of providing independent samples from the desired distribution. Therefore one way to keep tabs on the convergence question is to explicitly inquire of just how independent the samples seem to be.

The simplest way to roughly assess independence, over the course of the run, is to plot successive values of recorded quantities and to observe the actual sequence of change. The IM and IMa programs include an option to plot trend lines. Good mixing is suggested by pattern in which the recorded value explores the same range of values in different portions of the plot (see top Trendline figure example). At the other extreme, poor mixing is reflected in movement of the recorded quantity that is so slow that it is not yet clear what the likely range of values is going to be (see bottom trendline figure). If a long run produces patterns like that in the bottom figure, then there is a severe mixing problem. Intermediate patterns (middle figure) may not be cause for concern, but will at least require longer runs. Trendline plots do not offer any guarantees, and it is quite possible (and common) for a run to generate a plot like the top figure over the early course of the



Example figures of parameter trendlines. The plots generated by IM are more approximate and are generated using ASCII text.

run, but to then turn into a plot like the bottom figure, as the simulation proceeds and begins to explore a larger or different portion of the state space.

Another way to assess convergence is to monitor the autocorrelation of recorded values over the course of a run. Each autocorrelation calculation is based on a distance k , called the lag, that is the number of steps between the pairs of values that are included in the calculation. The autocorrelation for a given lag k is ρ_k , and it can be calculated for all possible values of k. One expects very high values for ρ_k (near 1) for small values of k, and one hopes for values near zero for larger values of k. One widely used measure of mixing is based on the idea of Effective Sample Size (ESS). If samples taken at each successive

step were independent then ESS would be the number of steps. When they are not independent then ESS can be estimated as

$ESS = \frac{M}{1 + 2\sum_{k=1}^{\infty} \rho_k}$, where M is the total number of steps.

These programs approximate this sum by measuring ρ_k for a series of values for k up to 1,000,000 steps. If you observe strong autocorrelations for this large a lag value then the *ESS* value is underestimated, and you've got a real mixing problem.

The IM and IMa programs estimate the autocorrelations and ESS values for whatever parameters are included in the MCMC simulation as well as for Log(P) = Log(P(Data|Genealogy)) + Log(P(Genealogy|Parameters)). ESS estimates are always going to be far less than the actual number of steps because values are highly autocorrelated along the chain. ESS estimates will not be stable or reliable for short runs. From experience with diverse data sets, the rule of thumb seems to be that one should run for at least a million steps before beginning to rely upon the ESS estimates.

The program will provide runtime values, in the running window on the computer screen, of estimated ESS values, however these are not reliable for short run times. With some experience, one can get a feel for what kinds of ESS values suggest good mixing, even if a run has only been going for a short while, however one needs to be quite cautious in using estimated ESS values for short runs. ESS estimates are inherently unstable for short runs because only low lag values are available. As the run progresses, larger lag values come online and this can cause the ESS estimates to jump around quite a lot.

If multiple Metropolis-coupled chains are used, then the ESS values will often be greatly inflated when they first begin to appear. This is because the different chains will have started out at widely different points in the parameter space, and if there is much swapping of chains, the initial assessments of autocorrelations will be quite low. Again, one needs to wait and see what the autocorrelations look like after a million or more steps.

Parameter Conversions - How to Obtain Demographic Parameter Estimates from Model Parameter Estimates

The IM and IMa programs can be used to generate estimates of model parameters (θ_1, θ_2, θ_A, m_1 , m_2 and t). Typically the peaks of the estimated distributions are taken as the estimates, just as if one was taking a maximum likelihood estimate. Indeed, because the method uses uniform prior distributions, these estimates are maximum likelihood estimates in those cases when the prior distributions are very wide and the tails of the posterior distributions are clearly contained within the range of the prior distributions (i.e. as if the prior really was uniform over the range of zero to infinity).

Once the model parameter estimates are in hand, many investigators will wish to also generate estimates of the demographic quantities (i.e. N_1, N_2, N_A, t, m_1, and m_2). Most of these conversions can be done automatically by the program, provided that mutation rate estimates are included in the input file (see the section on Input File Format and command line flags for the program you are using). Whether this is done by the program or by the investigator, it is important to understand what is being done here, as the subject can be a bit confusing. The next few paragraphs explain how these calculations are made.

First, note that that all of the parameters in the model include the mutation rate u (which is a value for the gene, not per base pair). If you have multiple loci, then u is the geometric mean of the mutation rates of all the loci. The method for converting parameter scales is explained for diploid autosomal loci for the parameters t, θ_1, and m_1. The same approach is readily applied to θ_2, θ_A, and m_2 in the same way.

First, gather your model parameter estimates, and an estimate of actual mutation rates (using an outgroup or some other relevant data).

- Let ***A*** be your estimate of θ_1 (i.e. $4N_1u$ where N_1 is the effective size of population 1)
- Let ***B*** be your estimate of t, the time parameter (i.e. t u, where t is the time since splitting)

- Let ***C*** be your estimate of m_1 (i.e. m_1/u). It is important to understand that m_1 is the rate per gene per generation from population 1 to population 2, *in the coalescent*. Since the coalescent goes backwards in time, m_1 is more easily thought of as the rate at which genes come into population 1, from population 2, as time moves forward.
- Let ***U*** be an estimate of the mutation rate per year for the gene being studied. (not per base pair, but for the entire gene). This must usually be obtained using some other data, such as distance from an outgroup of known time separation. If you are doing multiple loci, then ***U*** is the geometric mean of the mutation rates (per year) for the loci.
- Let ***V*** be an estimate of the mutation rate per generation for the gene being studied. If ***G*** is the number of years per generation, then ***V*** = ***U*** ***G***.

Now generate estimates of demographic quantities:

- To estimate the effective population size, N_1, calculate ***A***/(4 ***V***). This is because N_1 is defined in the coalescent models as being proportional to the inverse of the coalescent rate per generation. Therefore we need to use ***V*** since it is an estimate of the mutation rate on a scale of generations.
- To estimate the time since splitting, t, in units of years, take ***B***/***U***.
- To estimate the time since splitting in generations, take ***B***/***V***.
- To estimate the migration rate per generation, m_1, take ***C*** × ***V***.
- To estimate a migration rate per year, take ***C*** × ***U***.
- To estimate the population migration rate (the effective rate at which genes come into a population, per generation) for population 1 (i.e. $2 N_1 m_1$) you don't even need the estimate of mutation rate, Since $4N_1u \times m_1/u / 2 = 2N_1 m_1$, all you need to do is take ***A***×***C***/2.

When multiple loci are studied, the program also allows estimation of mutation rate scalars for each locus. If x_i is the estimate of the scalar for locus i, then an estimate of $4N_1u_i$ can be obtained by taking ***A*** × x_i.

If data from multiple loci are used, but per-year mutation rates are only available for a subset of them, then it is still possible to generate estimates of demographic quantities. The trick is to make use of the estimates of the mutation rate scalars for those same loci. Let ***X*** be the geometric mean of the estimates of the mutation rate scalars for just those loci for which per-year mutation rates are also available. Let ***U*** be the geometric mean of the per-year mutation rates *for just those same loci*, and let ***V*** = ***U G***. Then ***A X*** /(4 ***V***) is an estimate of N_1. Similarly an estimate of the number of years since divergence began is obtained with ***B X/U***. An estimate of the migration rate per generation per gene copy is obtained with ***C V/X***.

If you are working with multiple loci that have mixed inheritance models, then you will want to be setting the inheritance scalars in the input file (see Input File Format). In this case the method described above can be applied in exactly the same way without changes.

Similarly if you are working with a single locus, and the inheritance scalar in the input file is not one (e.g. if the data is mtDNA and the scalar is set to 0.25), then the affect of that scalar is to yield results that would be comparable to a diploid autosomal locus, and again the method described above can be applied in exactly the same way without changes.

However if you are working with a single locus that does not have diploid autosomal inheritance, but the inheritance scalar is set to 1 in the input file. Then the estimates of θ_1, θ_2, and θ_A are not of $4N_eu$ but rather of the product of $4N_eu$ and whatever the true inheritance scalar actually is for that locus.

Cautions, Suggestions and Interpretations

One of the greatest challenges is knowing whether the Markov chain is mixing sufficiently. Even when update acceptance rates are high, and ESS rates are high, and even when trend lines are suggestive of good mixing, it is recommended that multiple runs be made using different seeds for the random number generator.

Typically 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 it looks like the posterior distributions are fully contained within the bounds of the prior distribution, and if the observed maxima for any distributions are far to the left of the upper bounds, then the priors can be reduced for subsequent runs. Both the second and third 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.

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 marginal 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.

References

Estoup, A., P. Jarne, and J. M. Cornuet. 2002. Homoplasy and mutation model at microsatellite loci and their consequences for population genetics analysis. Molecular Ecology 11:1591-1604.

Geyer, C. J. 1991. Markov chain Monte Carlo maximum likelihood. Computing Science and Statistics, Proceedings of the 23rd Symposium on the Interface, Seattle, WA 156-163.

Hasegawa, M., H. Kishino, and T. Yano. 1985. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. Journal of Molecular Evolution 22:160-174.

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.

Hey, J., Y.-J. Won, A. Sivasundar, R. Nielsen, and J. A. Markert. 2004. Using nuclear haplotypes with microsatellites to study gene flow between recently separated Cichlid species. Molecular Ecology 13:909-919.

Hudson, R. R., and N. L. Kaplan. 1985. Statistical properties of the number of recombination events in the history of a sample of DNA sequences. Genetics 111:147-164.

Kimura, M. 1969. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. Genetics 61:893-903.

Kimura, M., and T. Ohta. 1978. Stepwise mutation model and distribution of allelic frequencies in a finite population. Proc Natl Acad Sci U S A 75:2868-72.

Mountain, J. L., A. Knight, M. Jobin, C. Gignoux, A. Miller, A. A. Lin, and P. A. Underhill. 2002. SNPSTRs: empirically derived, rapidly typed, autosomal haplotypes for inference of population history and mutational processes. Genome Res 12:1766-72.

Palsbøll, P. J., M. Berube, A. Aguilar, G. Notarbartolo-Di-Sciara, and R. Nielsen. 2004. Discerning between recurrent gene flow and recent divergence under a finite-site mutation model applied to North Atlantic and Mediterranean Sea fin whale (Balaenoptera physalus) populations. Evolution Int J Org Evolution 58:670-5.

Wilson, I. J., and D. J. Balding. 1998. Genealogical inference from microsatellite data. Genetics 150:499-510.