Documentation for BayesAss 1.2

Program Description

BayesAss is a program that estimates recent migration rates between populations using MCMC. It also estimates each individual's immigrant ancestry, the generation in which immigration occurred (i.e. the individual can be assigned as an immigrant from a specific population, a non-immigrant, or the offspring of an immigrant and a non-immigrant), the population (as opposed to the sample) allele frequencies, inbreeding (F) values within each population, and genotypes at any locus with missing data. Details of the method can be found in: **Wilson GA, and B Rannala. March 2003. Bayesian inference of recent migration rates using multilocus genotypes. Genetics 163(3), 1177-1191.** BayesAss requires diploid data, and can be used for genetic systems such as microsatellites, RFLPs, SNPs, and allozymes. The method assumes relatively low levels of migration, and the proportion of migrant individuals into a population cannot exceed 1/3 each generation. Loci are assumed to be in linkage equilibrium, but deviations from Hardy-Weinberg equilibrium are allowable. The method does not check for the presence of null alleles, and their occurrence may negatively affect the results. Estimates are most accurate when a large number of variable loci are used, sample size is large, and population differentiation is high, as described in the paper mentioned above. The program is available in Windows, and C++ versions. A Java version will be released shortly.

Input File

The setup for the data file follows the format of IMMANC, and should be:

```
Individual1    population1    locus1  allele1  allele2
Individual2    population1    locus1  allele1  allele2
```

Each line contains the genotype of a single locus. Columns can be separated by any whitespace. Missing genotypes should be represented by '0', '00', or '000'. Blanks are not allowed. Iterate through individuals, then populations, then loci. Spaces within names are not allowed. A sample data file named 'try.txt' is available. The data file should be in UNIX format, with a suffix of '.inp' or '.txt'.

Input Options

The program requires information about a number of user-defined variables. These variables are found in a pop-up menu in the Windows version, and appear after the program is run in the C++ version. Their descriptions are as follows:

1) seed: the initial seed for the random number generator. This value must be a positive integer. The user can test the convergence of the chain by running the program with different initial seeds and comparing the similarity of the results.

2) number of iterations: the number of iterations making up the chain. The recommended setting is 3000000.

3) sampling frequency: also known as the thinning interval, this is the frequency with which data is sampled from the chain. The recommended setting is 2000. A large sampling frequency allows the MCMC to move between states a number of times, thereby ensuring the relative independence of the samples from the chain.

4) burn-in: the length of the burn-in period, or the number of iterations before data collection begins. This value should be increased if the log likelihood values have not reached an apex at the time data collection begins (see output options 11). The total number of iterations where data is sampled from the chain is then

(number of iterations – burn-in) / sampling frequency.

5) delta value for allele frequency: the delta value defines the maximum amount a parameter can be changed by each iteration. They are bound between 0 and 1, but are inputted as integers in the C++ version, so you must multiply your desired delta value by 100 giving a range between 0 and 100. The inputted delta value should ideally be less than 35. As delta values increase, so does the rejected number of proposed changes in the chain. Log likelihood values are usually maximized when the accepted numbers of proposed changes (given at the end of the output file) are between 40 and 60%, and delta values can be changed accordingly.
NOTE: in the Windows version, all delta values are inputted as decimals. Therefore, delta values in this version should be 100 times smaller than those in the C++ version.

6) delta value for migration rate: as described above.

7) delta value for inbreeding (F) value: as described above.

8) name of output file: this follows the normal rules for naming files. The suffixes '.txt'. or '.inp' can be used. Suffixes are not required for the Windows versions.

Output Options

The second batch of choices is used to select what information will appear in the output file. In the C++ version, desired information should be given a value of '1'. Unwanted information should be set to any other integer. The Windows version makes use of check boxes instead of integer assignations.

1) user-defined settings: this gives the user a written record of the input options that were chosen, described above.

2) genotype and sampling information: prints out the information provided by the user in the input file. Note that missing data has been replace by alleles drawn at random.

3) mean and variance of the posterior probabilities for each allele: the mean and variance of the posterior probabilities for each allele, at each locus, in each population.

4) mean and variance of the posterior probabilities for each inbreeding (F) value: as stated.

5) individual assignments: the number of times each individual assigns to the various population and time states. Only non-zero states are printed. If an individual assigns to its source population, it must have a time of 0. If an individual assigns to a different population, it can either be a migrant (time 1), or the offspring of a migrant and a non-migrant (time 2).

6) posterior probabilities for missing data: the number of times each allele is chosen as part of the genotype for an individual that is missing data. This prints out immediately following that individual's assignment information, and is only displayed if the assignment data from choice 5 is also displayed.

7) summary of assignments over populations: the assignment information for each individual in a population is summed and displayed here.

8) mean and variance of the posterior probabilities for each migration rate: information about the migration rates into each population. Note that the migration rate from a population into the same population is defined as the proportion of individuals in each generation that are not migrants.

9) number of times the proposed changes are accepted: the number of times the proposed allele frequency, inbreeding (F) value, and migration rate changes are accepted. Ideally, these values will be between 40 and 60% of the chain length. Modifications in the delta values should change the acceptance rates for these parameters.

10) the number of times each frequency falls within 0.05 intervals: the number of times each allele frequency, inbreeding (F) value and migration rate falls within 0.05 intervals. These are only printed out if the respective posterior probability is also chosen for display. These values can be compared between runs with different random seeds to determine the convergence of the chain. Choosing different delta settings can maximize convergence.

11) changes in log likelihood values: if either log likelihood value changes by a set amount, the iteration and new log likelihoods are printed out. Comparison of these values will show when the log likelihoods have peaked.

12) distribution of log likelihood values when the chain was sampled: the number of times the log likelihoods fall within set intervals. These values can be compared to see if one set of results is more likely than another.

13) likelihood ratio test of the prior and posterior distributions of the migration rates: this uses a _^{2} test to determine if the priors for migration rates differ from the posterior distributions. If they do not, then the data supplies no information about the migration rate of this population.

In the C++ version, the user is then asked for the name of the input file. Remember to type in the entire file name, including the suffix. In the Windows version, the user must use File $\rightarrow$ Open to choose the proper input file.

Limits

Currently, the program can read in a maximum of 6000 individuals, 20 populations, 40 loci, and 40 alleles per locus. Note that, due to excessive memory usage, the maxima for the Windows version had to be set at 3500 individuals, 20 populations, 25 loci, and 35 alleles. If your data exceeds any of these settings, please contact the authors.

About the Results

You may find that the means of the allele frequency posterior probabilities differ from the maximum likelihood estimate obtained from traditional methods. This program takes each individual's immigrant ancestry into account when estimating allele frequencies (i.e. an immigrant is included as part of its original population, and not its sampled population). Consequently, allele frequencies of the populations as calculated here may differ from allele frequencies of the samples. Also, the allele frequencies in a population can be affected by the level of inbreeding within that population, which is also considered in our program.

Some populations may have a uniform distribution of allele frequencies. This usually occurs when no individuals assign to this population. The program then has no information on which to base probability calculations for allele frequencies, so they wander uniformly throughout the sample space. If this occurs, one may want to consider whether two populations are indeed distinct, or increase the population's sample size. You may also want to look at the results for the likelihood ratio test in option 13.

Keep in mind that one assumption of the method is that migration rates are relatively low. The proportion of non-migrants within a population is bound at a minimum of 2/3. As such, non-migration rates of approximately 2/3 may in fact be lower than this, and again suggests that populations may not be distinct.

Program Support

Questions and answers about general issues/problems using this program should be posted to the Genetics Software Forum (http://rannala.org/gsf). While every effort has been made to identify bugs in this program, it is unlikely to be perfect. Comments, suggestions, and program crashes should be reported to gregwils@ualberta.ca. Please send as many details as possible about what the program was doing before it crashed. All feedback is encouraged and will aid in the future development of BayesAss.