bamova

Zach Gompert and Alex Buerkle
University of Wyoming

February 2011 – software v. 1.0, documentation v. 1.0

1 Overview

This manual provides documentation for the `bamova` software. The models that are implemented in this software are described and analyzed in Gompert and Buerkle (2011) and we expect that this paper will be studied thoroughly prior to using the `bamova` software. This document simply provides information on how to compile and run the software and we assume familiarity with the models, which is essential to make proper use of the software. Depending on the scale of the analysis, this software will be used to estimate thousands of parameters based on large sets of data. This is a reasonably large computational problem that requires some computational skills on the part of the user, including use of UNIX and the ability to write code to produce input files and to analyze and summarize output.

2 Obtaining the software

The `bamova` software can be downloaded from `http://www.uwyo.edu/buerkle/software`. The software is distributed as a compiled binary for Intel-based computers running Mac OS X, and as C++ source code that can be compiled by the user on any linux or UNIX platform. The software consists of a program that is run from the UNIX command-line and does not have a graphical interface.

The software depends on free and open-source software called the GNU Scientific Library (GSL), so the GSL needs to be installed on the user's system (`http://www.gnu.org/software/gsl/`). This means that the compiled binary of `bamova` requires that the GSL is installed in `/usr/local`, which is the default location. Compiled binaries for the GSL are available as part of many standard linux distributions. Alternatively, the GSL can be compiled by the user. Users who are compiling `bamova` from the source code should install GSL in the standard location (`/usr/local`) or modify their compilation command to point to the proper location of the library (see below).

We assume that users are familiar with moving the compiled binary into a directory in their UNIX `$PATH`, setting permissions for execution, etc., or can obtain assistance from other local users, books or the web.

2.1 Compiling the software

Assuming that the GSL has been installed, here are examples of how to compile the `bamova` software using the gcc compiler suite.

- For Mac OS X:
  ```
  g++ -Wall -O2 -o bamova bamova.C func.C -lgsl -lm -framework Accelerate
  ```
- For linux or other UNIX systems:
  ```
  g++ -Wall -O2 -o bamova bamova.C func.C -lgsl -lm -lgslcblas
  ```
- For linux/UNIX systems with non-default install location for GSL and ATLAS CBLAS
  ```
  g++ -Wall -O2 -L/usr/local/gsl-1.14/lib -L/usr/local/atlas-3.9.28/lib/ \
  -I/usr/local/gsl-1.14/include -I/usr/local/atlas-3.9.28/include/ \
  -o bamova bamova.C func.C -lgsl -lm -lcblas -latlas
  ```

3 Input file formats

3.1 Haplotype counts

All analyses require an input text file that gives the observed counts of each of the haplotypes for each population and locus. The format of this file is identical for the *known haplotypes model* and *NGS-population model*. Data for each genetic region begins with a line that gives the genetic region's number "`Marker0`" for the first genetic region, "`Marker1`" for the second and so for each genetic region. The numbers should be consecutive and begin with 0. The genetic region identification line should be followed by one line of data for each populations. These lines should begin with "`Population`", then give the population number (start with 0 and number populations consecutively, and then the counts of each haplotype, which should come in the same order for each population. A short example with two genetic regions each with five haplotypes sequenced in four populations is given in Table 1.

The haplotype count file for the *NGS-individual model* is a bit different from the other models (Table 2). Genetic regions are denoted as described above, but are followed by population identifiers and then a single line per individual giving the number of reads of each haplotype observed for that individual, as shown below for three populations and one genetic region.

3.2 Number of pooled individuals

The *NGS-population model* requires a second input text file that provides the number of diploid gene copies that were sampled for each population and genetic regions. This file should be in the form of a white-space separated matrix with rows corresponding to genetic regions and columns corresponding to populations.

Table 1: Sample input data file format for the counts of haplotypes in each population and for each genetic region.

```
Marker0
Population  0  0  30  0  5  5
Population  1  0  10  0  5  5
Population  2  5  0   5  0  0
Population  3  5  10  5  0  0
Marker1
Population  0  2  12  1  1  5
Population  1  4  8   1  5  3
Population  2  7  7   7  1  1
Population  3  5  17  3  4  12
```

Table 2: Sample input data file format for the *NGS-individual model* and the number of reads for each genomic region, per individual and for each population.

```
Marker0
Population  0
10  0  0    0  5
10  0  0    0  5
⋮   ⋮  ⋮    ⋮  ⋮
10  0  0    0  5
5   0  0    0  1
Population  1
10  0  0    0  0
10  0  0    0  5
⋮   ⋮  ⋮    ⋮  ⋮
5   0  5    0  0
5   0  5    0  0
Population  2
0   0  10   5  0
0   4  10   0  0
⋮   ⋮  ⋮    ⋮  ⋮
0   0  10   5  0
0   0  10   0  0
```

Table 3: Sample input data file format for the distances between haplotypes for each genetic region (two loci in this case).

```
Marker0
0  1  3  2  3  3
1  0  2  1  2  2
3  2  0  1  2  2
2  1  1  0  1  1
3  2  2  1  0  2
3  2  2  1  2  0
Marker1
0  1  2  3  4
1  0  1  2  3
2  1  0  3  4
3  2  3  0  1
4  3  4  1  0
```

3.3 Distances between haplotypes

An input text file giving distances between pairs of haplotypes will often be used in the analysis (`-d` argument). If this file is not provided by the user, all distances will be set to 1. Genetic regions are specified in the distance file just as they are in the haplotype count file. The identifier for each genetic region is followed on the next lines by a pairwise distance matrix with haplotypes in the same order as the haplotype count file. The diagonal of this matrix should be 0 and the matrix should be symmetrical. An example for two genetic regions is given in Table 3.

3.4 Groups of populations

Estimating ϕ_{SC} and ϕ_{CT} requires specifying groups of populations, which can be done by including a population group file (`-g` argument). This file should have one row per group that gives the group number and the numbers of the populations in each group. For example to specify two groups, one with populations 0 and 3 and the other with populations 1, 2, and 4 you would use the following simple file format:

```
Group0  0  3
Group1  1  2  4
```

4 Command line arguments

Filenames and MCMC parameters can be specified and adjusted using command line arguments. These command line arguments are defined in Table 4. Many of these arguments have default values, which are used when alternative values are not supplied. Default values are shown in parentheses.

Table 4: Command line arguments that are used by `bamova`.

`-f`	Filename for haplotype frequency input file (“undefined”)
`-d`	Filename for genetic distance input file (“undefined”)
`-g`	Filename for population groups input file (“undefined”)
`-t`	Filename for MCMC samples of haplotype frequencies (“hapfreq_output.txt”)
`-m`	Filename for MCMC samples of genome-level parameters (“ab_output.txt”)
`-p`	Filename for MCMC samples of ϕ statistics (“phi_output.txt”)
`-q`	Filename for MCMC samples of ϕ statistic quantiles (“quantile_output.txt”)
`-l`	Likelihood model: *known haplotypes model* = 0, *NGS-population model* = 1, *NGS-individual model* = 2 (0)
`-n`	File containing the number of gene copies (2N for diploid organisms) sampled for each locus and population. Only used with the *NGS-population model* (“samn.txt”)
`-x`	Number of MCMC iterations for which the model will run (NA)
`-v`	Adjustment for the variance of proposal distribution for α and β. This value multiplied by the mean expected value of α and β is used for the standard deviation of the proposal distribution (0.2)
`-a`	MCMC algorithm for genome-level parameters α and β: 0 = random walk, 1 = independence chain. Both algorithms use a bi-variate normal distribution to propose new values of α and β (0)
`-D`	MCMC algorithm for proposing haplotype frequency vectors: 0 = independence chain, 1 = random walk. Both algorithms use a Dirichlet distribution to propose new values for haplotype frequencies (0)
`-w`	Parameter to adjust the variance of the proposed values for the haplotype frequency vectors. Increasing this parameter will decrease the variance of the proposal distribution (1)
`-W`	Parameter to adjust the variance of the initial values for the haplotype frequency vectors, relative to the observed haplotype frequencies. Increasing this parameter will decrease the variance of the values used for initialization (1)
`-u`	Minimum of the uniform hyper-prior for the genome-level parameters α and β (0.5)
`-U`	Maximum of the uniform hyper-prior for the genome-level parameters α and β (10^5)
`-c`	Correlation of proposal values of α and β from a bi-variate Normal distribution (0.8)
`-i`	Parameter to adjust the proposal distribution for haplotype frequencies. This value is added to the current haplotype frequencies (random-walk) or observed frequencies (independence chain) after they are multiplied by the variance parameter given by `-w` to specify the parameters of the Dirichlet proposal distribution (0.00001)
`-I`	The same as `-i`, but used for initialization (1)
`-T`	Thinning parameter. Specifies how often MCMC samples are printed to files (10)
`-o`	Specifies which set of MCMC output files to print: 0, genome-level parameters output only; 1, genome-level parameters and ϕ statistics; 2, same as 1 plus quantile MCMC results; 3, same as 2 plus haplotype frequencies. Warning, the haplotype frequency MCMC results can be very large and can fill a small disk drive (1)

5 Software output

Several output files are produced by the program (the specific files generated depend on the `-o` argument). All of these files simply contain the MCMC samples. These can be used to estimate various aspects of the posterior probability distribution for each parameter of interest (using for example `R`, R Development Core Team 2010). Be aware that many of these files will be quite large, and parsing these files using `Perl` or a similar computer language often will often be necessary prior to summarizing the results in a statistical package.

Each row in the MCMC haplotype frequency output file (default "hapfreq_output.txt"; which is often very large) begins with the genetic region number, followed by the population number, the MCMC iteration, the number of unique haplotypes for the genetic region, and then the haplotype frequencies for each haplotype for that MCMC iteration. Rows are padded with trailing `NA` entries to ensure that all rows are the same length.

The genome-level parameters outfile (default "ab_output.txt") has the following format: each row begins with the MCMC iteration, followed by the values for α_{ST}, β_{ST}, mean genome-level ϕ_{ST}, the standard deviation of the genome-level ϕ_{ST} distribution, and the probability of the parameter values given the data and priors for that MCMC iteration. If the group option is used this file will instead have the MCMC iteration, followed by the values for α_{ST}, β_{ST}, α_{CT}, β_{CT}, α_{SC}, β_{SC}, mean genome-level ϕ_{ST}, the standard deviation of the genome-level ϕ_{ST} distribution, mean genome-level ϕ_{CT}, the standard deviation of the genome-level ϕ_{CT} distribution, mean genome-level ϕ_{SC}, the standard deviation of the genome-level ϕ_{ST} distribution, and the probability of the parameter values given the data and priors for that MCMC iteration.

The ϕ statistic outfile (default "phi_output.txt") begins with the MCMC iteration, followed by the genetic region and ϕ_{ST} at that MCMC step for that genetic region. If the group structure option is used ϕ_{ST} is followed by ϕ_{CT} and ϕ_{SC}.

The final potential outfile gives the quantiles associated with each locus-specific ϕ statistic in the genome-level distribution. Each row in this file begins with the MCMC iteration, then gives the genetic region number followed by either the quantiles for the locus-specific ϕ_{ST} (the population structure only model) or locus-specific ϕ_{ST}, ϕ_{CT} and ϕ_{SC} (the group structure model).

6 Example

We provide a brief example to illustrate the use of `bamova`. This example includes two input files: hapcountexample.txt (a file of haplotype counts) and distfileexample.txt (a file of distances among haplotypes). These files are included with the software distribution and include five populations each made up of 20 diploid individuals with data from 50 genetic regions. The number of haplotypes per genetic region varies. These data files were generated using coalescent simulation (with the `ms` software, Hudson 2002) using a splitting time parameter of 0.7 and population mutation rate $\theta = 0.2$.

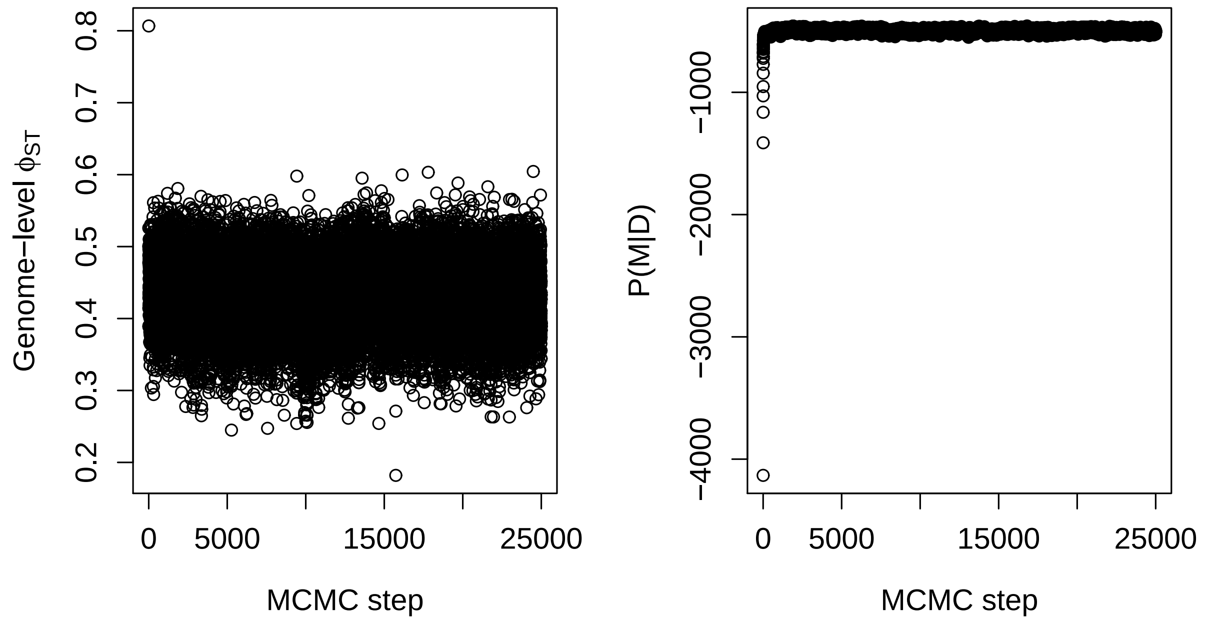

Figure 1: Scatterplots showing mean genome-level ϕ_{ST} and the log probability of the model parameters given the data as a function of MCMC iteration.

We used the following command to analyze this data set (see the description of command line arguments and their defaults in Table 4 for more information). The mixing parameters we use here were selected after a number of short runs with different tuning parameters.

```
bamova -f ./hapcountexample.txt -d ./distfileexample.txt -l 0 -x 250000 \
-v 0.25 -a 0 -D 0 -w 1 -W 1 -i 0.0 -I 0.0 -T 10
```

Important note: Efficient mixing can be very difficult to achieve, particularly for large data set with many haplotypes per locus. You cannot assume that the default mixing parameters will work well for your data set. Adjust these parameters, observe how the chains behave by analyzing the output, and verify good mixing before using the results from these analyses in any way. There is no way around this and this is an inherent feature of MCMC estimation procedures.

Figure 1 contains plots of mean genome-level ϕ_{ST} as a function of MCMC step and the probability of the model parameters given the data, both of which suggest good mixing and the need for only a short burnin (these plots were generated in R). We discard the first 5000 samples as a burn-in, and use the remainder to calculate a point estimate (the median of the posterior distribution) and 95% credible intervals (CI) for the mean genome-level ϕ_{ST}, which gives the following results: 0.435 (0.339–0.517). This was done using the `quantile` function in `R`.

Next we calculate posterior probability distributions for each of the locus-specific ϕ_{ST}.

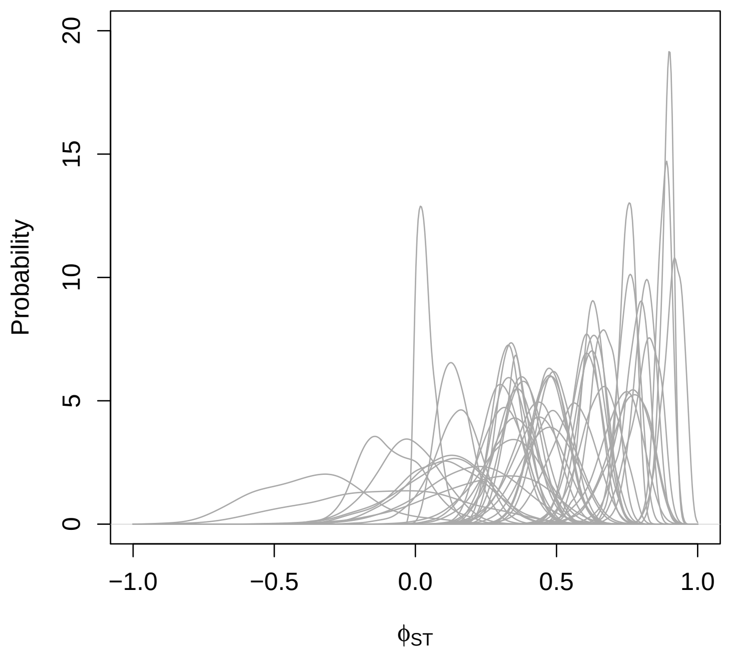

Figure 2: Posterior probability distribution for each locus-specific ϕ_{ST}.

We do this by reading the MCMC samples into `R`, and using the `density` function. The `R` code we used is below and the plot of the posterior probability distributions is in Figure 2.

```
phidata<-read.csv("phi\_output.txt", header=F)
postscript(file="phidensity.eps", width=6, height=6)
nloci<-50
burnin<-5000
mcmcl<-25000
onephi<-phidata[phidata[,2]==0,3]
plot(density(onephi[-c(1:burnin)],adj=2,from=-1,to=1), xlim=c(-1,1), ylim=c(0,20),
     xlab=expression(paste(phi[ST])), ylab="Probability", main="",cex.axis=1.2,
     cex.lab=1.2, col="darkgray")
for(i in 2:nloci){
    onephi<-phidata[phidata[,2]==i -1,3]
    lines(density(onephi[-c(1:burnin)],adj=2,from=-1,to=1), col="darkgray")
}
dev.off()
```

Point estimates and credible intervals for each parameter can be obtained using `quantile` in `R`. For example, an estimate of ϕ_{ST} for genetic region 18 (which is the 19^{th} locus) with a 5000 sample burnin would be obtained using the following (this gives the median and 95% CI),

```
locus<-18
onephi<-phidata[phidata[,2]==locus,3]
quantile(onephi[-c(1:burnin)],prob=c(0.025,0.5,0.975)
```

which gives the following result for our MCMC sample: 0.374 (0.249–0.498).

7 Questions and additional information

We will do our best to respond to requests for additional information and features in the software. Should the need arise, we will post answers to Frequently Asked Questions on the `bamova` website: `http://www.uwyo.edu/buerkle/software/bamova`

7.1 Terms of use

The `bamova` software is available for use under the GNU Public License (GPL). This means it is free to use. You are encouraged to download the source code, review it, and improve it. If you use parts of the code in software of your own, you are required to give your users the same rights that were granted to you under the GPL.

Note that under the GPL, the software is distributed without warranty.

We request that you cite Gompert and Buerkle (2011) if you use the software in analyses that appear in print.

References

Gompert, Z. and C. A. Buerkle, 2011 A hierarchical Bayesian model for next-generation population genomics. Genetics p. doi: 10.1534/genetics.110.124693.

Hudson, R. R., 2002 Generating samples under a Wright-Fisher neutral model of genetic variation. Bioinformatics **18**: 337–338.

R Development Core Team, 2010 *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.