spatpg

Zach Gompert
Utah State University

July 2015 – software v. 1.0, documentation v. 1.0

1 Overview

This manual provides documentation for the `spatpg` software. `spatpg` is a program that can be used to estimate variance effective population sizes and environment-dependent selection coefficients from genetic time-series data (allele frequencies) from multiple populations. The statistical models and methods that are implemented in this software are described and analyzed in the paper "Bayesian inference of selection in a heterogeneous environment from genetic time-series data" that is under review in *Molecular Ecology*. This paper should be studied thoroughly prior to using the `spatpg` software. This document simply provides information on how to compile and run the software. I assume familiarity with the models, which is essential to make proper use of the software.

2 Obtaining the software

The `spatpg` software can be downloaded from `source forge` at `http://sourceforge.net/projects/spatpg/`. The software is distributed 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 `spatpg` 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 `spatpg` 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). The software also depends on free and open-source software called HDF5 (`http://www.hdfgroup.org/`). HDF5 is a data model, library, and file format that `spatpg` uses for storing and managing MCMC samples.

I 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 and HDF5 have been installed, here is an example of how to compile the `spatpg` software using the `h5c++` compiler suite (provided with the HDF5 software).

- For Linux or other UNIX systems:
  `h5c++ -Wall -O3 -o spatpg main.C func.C mcmc.C -lm -lgsl -lgslcblas`

3 Input file formats

The `spatpg` software requires two input text files, one for the genetic data and one for the environmental covariate data. Both are required.

Genetic data file: This file provides sample allele frequencies in a format very similar to that required for `TreeMix`. A header row gives the number of populations, generations and genetic loci separated by spaces. This is followed by one row with sample allele frequencies for each population and generation for each locus. Data on each row should be organized as follows: population 1 generations 1 to T, followed by population 2 generations 1 to T, etc. At present only bi-allelic loci are allowed and the count of each allele should be give separated by a comma, e.g. '7,13' would denote 7 copies of the reference allele and 13 of the non-reference allele. Use '0,0' for missing data. A short example three populations, two generations and four SNPs is shown in below.

```
3 2 4
22,8 23,7 17,13 16,14 22,8 20,10
26,4 26,4 24,6 19,11 17,13 20,10
17,13 22,8 20,10 19,11 22,8 21,9
19,11 19,11 21,9 15,15 23,7 17,13
```

Environmental data file: This file provides the environmental covariate data for each population and generation. There should be one row per population by generation combination with the centered covariate value for that observation. Similar to the genetic data, give the values for population 1 generations 1 to T first, then population 2 generations 1 to T, etc.

4 Command line arguments

File names and MCMC parameters can be specified and adjusted using command line arguments. These command line arguments are defined in Table 1. Many of these arguments have default values, which are used when alternative values are not supplied. Default values are shown in square brackets.

Table 1: Command line arguments that are used by `spatpg`.

`-g`	Infile with allele frequency data in TreeMix format.
`-e`	Infile with environmental covariate data.
`-o`	Outfile for MCMC samples [outfile.hdf5]
`-n`	Number of MCMC steps [1000].
`-b`	Number of MCMC steps to discard as a burn-in [0].
`-t`	Thinning interval integer), record every nth step [1].
`-l`	Lower bound for uniform prior on Ne [20].
`-u`	Upper bound for uniform prior on Ne [4000].
`-s`	Standard deviation for prior on coefficients [0.1]
`-p`	Standard deviation of normal proposal distn. [0.05].

5 Software output

Samples from the posterior distribution of the model parameters are stored in a hdf5 format outfile (these include the variance effective population size (Ne), the population allele frequencies (p), the intercept (alpha) and regression coefficient (beta) from the linear model, and the selection coefficient (s) for each locus, generation and population. You can view these and make very simple plots using the `hdfview` program provided by HDF5 group (`http://www.hdfgroup.org/hdf-java-html/hdfview/`). However, you will want to use the program `estpost` to summarize and analyze the results. This program should be downloaded with the `spatpg` software and can be compiled with the h5cc compiler suite (provided with the HDF5 software).

- For Linux or other UNIX systems:
  ```
  h5cc -Wall -O3 -o estpost estpost_h5_spatgen.c -lm -lgsl -lgslcblas
  ```

The software `estpost` can do four things: (i) generate point estimates and credible intervals for parameters, (ii) generate histograms for posterior samples, (iii) generate ascii text output of individual files, and (iv) generate MCMC diagnostics.

5.1 Command line arguments for `estpost`

File names and options can be specified and adjusted using command line arguments. These command line arguments are defined in Table 2. Some of these arguments have default values, which are used when alternative values are not supplied. Default values are shown in square brackets. The hdf5 file or files from `spatpg` should be listed after all of the other command line options (and without a flag). If more than one file is given, the posterior samples will be combined across files (chains). This is required for some of the MCMC diagnostics.

Table 2: Command line arguments that are used by `estpost`.

`-o`	Outfile [default = postout.txt].
`-p`	Name of parameter to summarize, possibilities include: 'ne' ,' p', 'alpha', 'beta', and 's'.
`-c`	Credible interval to calculate [default = 0.95].
`-b`	Number of additional (beyond the burn-in passed to `spatpg` MCMC samples to discard for burn-in [default = 0]. This burn-in is based on the number of thinned samples.
`-h`	Number of bins for posterior sample histogram [default = 20].
`-s`	Which summary to perform: 0 = posterior estimates and credible intervals, 1 = histogram of posterior samples, 2 = convert to plain text, 3 = MCMC diagnostics.
`-w`	Write parameter identification and headers to file, Boolean [default = 1].
`-v`	Display software version.

5.2 Output from `estpost`

The output from estpost depends on the summary requested.

Point estimates and CI (`-s 0`): This file contains an optional header row and parameter identification column (parameters are ordered and number as they were ordered in the input files but starting with 0). Each line gives the mean, median, and lower and upper bounds of the specified credible interval (this is an equal-tail probability interval).

Posterior histograms (`-s 1`): This file contains an optional parameter identification column (parameters are ordered and number as they were ordered in the input files but starting with 0). Each row contains paired (i.e., x1, y1, x2, y2, x3, y3 ...) x (parameter value for the midpoint of a bin) and y (number of MCMC samples in the bin) coordinates to plot a posterior histogram for a parameter.

Convert to ascii text (`-s 2`): This file contains an optional parameter identification column (parameters are ordered and number as they were ordered in the input files but starting with 0). Each line contains the post-burn-in MCMC samples for the designated parameter.

MCMC diagnostics (`-s 3`): This file contains an optional parameter identification column (parameters are ordered and number as they were ordered in the input files but starting with 0). Each row then gives the effective sample size and gelman-rubin potential scale reduction factor for each parameter.

6 Example

Here I provide a brief example to illustrate the use of `spatpg`. Infiles for this example are included with the software distribution in the examples folder: mod_rsimFluctLG0.txt

(genetic data; 10 populations, 10 generations, 1000 SNPs) and rsimFluctLE0 (environmental data). These data were simulated in R under a Wright-Fisher model (as described for linear fluctuating selection in the manuscript). In the simulations, the first locus affected fitness ($\alpha = 0$, $\beta = 0.1$), while the other 999 loci were neutral.

I used the following commands to analyze these data (see the description of command line arguments and their defaults described above for more information). Two chains were runs. The MCMC analysis took about an hour on my laptop.

```
./spatpg -g examples/mod_rsimFluctLG0.txt -e examples/rsimFluctLE0.txt -o out1.hdf5
-n 20000 -b 10000 -t 10 -l 20 -u 4000 -s 0.1 - p 0.05
```

```
./spatpg -g examples/mod_rsimFluctLG0.txt -e examples/rsimFluctLE0.txt -o out2.hdf5
-n 20000 -b 10000 -t 10 -l 20 -u 4000 -s 0.1 - p 0.05
```

Note: You might need to adjust the tuning parameter to achieve efficient mixing when analyzing your own data. Observe how the chains behave by analyzing the output, and verify good mixing before using the results from these analyses in any way.

Parameter effective sample sizes and the Gelman-Rubin potential scale reduction factor can be calculated with estpost. Here I have done with for α, β, and Ne (results will be in mcmcdiagA.txt, mcmcdiagB.txt and mcmcdiagNe.txt). In general, you want the effective sample size for most (all) parameters to be large (at least a few hundred, preferably a few thousand), and the Gelman-Rubin diagnostic to mostly be below 1.1. You can easily increase the effective sample size by running the MCMC longer, and this will often (but not always) improve the Gelman-Rubin diagnostic (if combined with a longer burn-in).

```
./espost -o mcmcdiagA.txt -p alpha -s 3 out1.hdf5 out2.hdf5
```

```
./espost -o mcmcdiagB.txt -p beta -s 3 out1.hdf5 out2.hdf5
```

```
./espost -o mcmcdiagNe.txt -p ne -s 3 out1.hdf5 out2.hdf5
```

You can then summarize the posterior distribution of the parameters of interest by calculating its mean, median and any quantiles of interest. Here I have calculated the 2.5th and 97.5th quantiles, i.e. the 95% credible intervals for β and the selection coefficient. You should see that the center of the posterior density of β for the first locus is around 0.1 (the true value), and that the 95% CIs exclude 0. Note however that there is nothing special about the 95% CIs and you could chose other intervals to summarize the posterior. You can also drop the headers and row-names from the output by setting `-w 0`, which can be useful when reading the results into `R` for plotting. Finally, you could instead make a histogram of the posterior or convert the MCMC to text using `-s 1` or `-s 2`.

```
./espost -o estB.txt -p beta -c 0.95 -s 0 out1.hdf5 out2.hdf5
```

```
./espost -o estS.txt -p s -c 0.95 -s 0 out1.hdf5 out2.hdf5
```

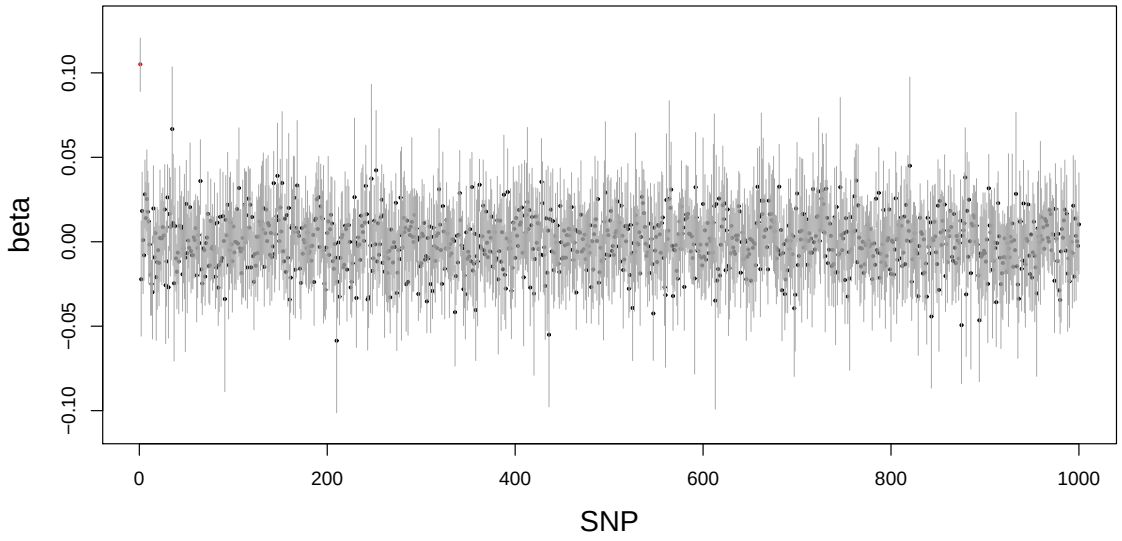

Figure 1: Plots depict point estimates (dots; median of posterior) and 95% credible intervals (vertical gray lines) for the regression coefficient. A single functional variant that affected fitness is shown in red (this is the first locus).

Here is `R` code that I used to make a plot (Fig. 1) of the median and 95% CIs for the β parameter from the essB.txt file.

```
b<-read.table("estB.txt",header=TRUE,sep=",")
postscript("examplePlot.eps",width=10,height=5,paper="special")
par(mar=c(5.5,5.5,0.5,0.5))
plot(b$median,pch=20,ylim=c(-0.11,0.13),xlab="SNP",ylab="beta",
col=c("red",rep("black",999)),cex.lab=1.5,cex=0.5)
segments(1:1000,b$ci_0.950_LB,1:1000,b$ci_0.950_UB,lwd=0.1,col="darkgray")
dev.off()
```

7 Questions and additional information

I will do my best to respond to requests for additional information and features in the software. Should the need arise, I will post answers to Frequently Asked Questions on the `spatpg` web-site: `http://sourceforge.net/p/spatpg/wiki/FAQ/`

7.1 Terms of use

This version of the `spatpg` 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.