Short Manual for IMPPS

Ian Wilson

November 8, 2006

1 IMPPS

Inference of Maternal Pedigrees for Partial Selfing (**IMPPS**) is a C++ computer program that enables inferences to be made about the selfing rate within a local deme, the population size of that population and the relationships between sampled individuals. The program uses a Bayesian Markov chain Monte Carlo method to sample from the posterior distribution of selfing pedigrees, selfing rate and local population size N.

1.1 Obtaining IMPPS

The program is available from Ian Wilson's web site at http://www.staff.ncl.ac.uk/i.j.wilson/impps.html. Any comments or problems should be emailed me at I.J.Wilson@ncl.ac.uk

For Microsoft Windows a precompiled binary, `wimpps.exe` is available. For other operating systems the program must be compiled.

1.2 Compiling under UNIX

The Gnu Scientific Library (GSL) is required for the compilation of **IMPPS**. This is available at http://www.gnu.org/software/gsl/ and binaries are available for the major Linux distributions, and cygwin/mingw32.

Once GSL is compiled **IMPPS** can be compiled by typing `make` in the source directory.

Note that using the STLPORT C++ STL libraries seems to increase the speed by a factor of two when combined with with native `g++` implementation of the STL so that it can be useful to compile with support for these if available. To compile with STL support define the STL environmental variable to be `stlport` using (in bash)

```
export STL=stlport
```

before typing `make`. If your STLPORT libraries are not at `/usr/include/stlport` then you will need to edit the `makefile`.

2 Running IMPPS

2.1 Unix

The program is command line driven. Open a shell, move to the directory where **IMPPS** is located and type

```
./impps --help
```

for help. A summary of the options is shown in figure 4 on page 5 in this manual. The program uses a default input file `default.input` if present in the directory where `impps` is run.

All error messages to the screen are echoed to a log file, `res.log` is the default, so that error messages may be examined there.

2.2 Windows

Open a command window using **start→run** and type `cmd` in the box to bring up a shell. Then go to the directory containing `impps.exe` and type `impps.exe --help` for available options.

Alternatively, right click on the desktop and select **new→shortcut**. Type `cmd` in the box and press **next**, change the name and **finish**. If you right click and choose **properties** and change **start in** to the directory in which you have installed **IMPPS**.

Type `impps.exe --help` for available options.

All error messages to the screen are echoed to a log file, `res.log` is the default so that error messages may be examined there.

3 Setting Options

There are a number of options available in the **IMPPS** software. These can be set either on the command line (which takes precedence) or using an input options file.

3.1 The Command Line

All options (except for the name of the input file) are input to the command line in the format `--option<=value>`, where `option` is the name of the option, and `value` is the value taken. Note that *switches* (options that can be either true or false) are `false` by default, and are made true just using `--option`.

Note that options set on the command line override values read from the input file.

The input file for this model is specified in the first position after the program name, and is `default.input` by default.

3.2 The Options File

The options file consists of lines with multiple
`option = value`
pairs, and lines
`option`
for switches. Anything on a line after a # is a comment

An example file `input.a` which works with the example data is in the data directory and is shown in figure 1.

4 Options Available

Figure 4 gives a summary of the options available from `impps`. These can be generally split into three types: options that give input and output files, options that determine the model, and options that change the way that the Markov chain is sampled.

4.1 Input and Output Files

4.1.1 Output Files

The option **stem** gives the first part of the output file. The default output stem is `res`. Several output files are created from this stem.

```
# input file used to get data for
# the trace plot of Wilson, Johnson and Dawson

burnin=1
bet=10
reps=2000
N=normal(100,100)
k=10
T=5
stem=resa
d=a.data
seed=1
freq=population.frequencies
logN                          # use logarithmic updates for N
tuneN=0.5                     # with tuning parameter of 0.5
startN=60
starts=0.2
```

Figure 1: Input file `input.a`

`res`	The standard output file with each line giving the log-posterior, the posterior N, posterior selfing rate, s and the posterior number of matrilineal pedigrees.
`res.log`	A file giving details of the run including all options used.
`res.partition`	A file that can be used to construct the posterior partition into matrilineal trees. The partitionview program of Dawson and Belkhir can be used to get get representations of the posterior partition.
`res.MROA`	A file that can be used to construct the posterior partition into Individuals with the same most recent outcrossed ancestor. Again the partitionview program can be used on this output.

4.2 Input Files

4.2.1 Data file format

The main data file consists of lines of genotypes with the alleles separated by "/" characters. We assume that the allele labels start at A and go through consecutive letters of the alphabet. The first five lines of an example data file file are shown in figure 2.

```
A/A A/A B/B A/A A/A F/F
C/C A/A B/B A/A A/A B/B
A/A B/B B/B A/A B/B F/F
C/C D/D A/A B/B A/A J/J
C/A A/B C/A A/A A/A F/G
D/A D/D D/A B/A B/A J/H
```

Figure 2: Format of a data file with six loci.

4.2.2 Frequency File Format

The frequency file consists of **loci** rows with each line giving the relative frequencies of the alleles at that locus in the overall population. The first allele is allele A, the second B and so on. An example of this format is shown in figure 3. Note that a frequency file is not necessary for analyses. See the option **varf**.

```
0.5 0.2 0.2 0.05 0.05
0.4 0.3 0.1 0.1 0.02 0.03 0.05
0.2 0.2 0.2 0.2 0.15 0.05
0.6 0.3 0.1
0.7 0.3
0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

Figure 3: Format of a frequency file.

The data are split into two parts. The input file, and the parameter file.

4.3 The Model

4.3.1 Local population size N

The program needs prior distributions for the local population size N. These are provided on the command line or in an options file. For example to set a normal(50,10) prior for N one can use `--N="normal(50,10)"` on the command line or a line
`N = normal(50,10)` in an input file.

Note that the distribution must be given in either single or double quotes for options given from the command line

The default prior for N is an improper uniform.

4.3.2 Selfing rate s

In the same way as N, the prior for s can be set on the command line or in an options file. You can provide any distribtution, but if it extends outside $[0, 1]$ then it will be truncated. It is more usual to use a prior restricted to between 0 and 1 such as the Beta distribution with parameters α and β which gives a mean of α/β and a variance of $\alpha\beta/((\alpha+\beta)^2(\alpha+\beta+1))$. To set the prior to a Beta with parameters 2 and 2, with a mean of $0{\cdot}5$ and a variance of $0{\cdot}05$ use `--self="beta(2,2)"` on the command line or a line
`self = beta(2,2)` in an input file.

The default prior for self is `uniform(0,1)`.

4.3.3 Number of Generations

The number of generations in the local population is set by the option **T**.

4.3.4 The Frequency file

If you have information on the background frequencies of the alleles then this can help the analyses. The format oif a frequency file is given on page 3. However, if data are only available on the selfing population then setting the option **varf** allows you to use a dirichlet prior on the allele frequencies at each locus. At the moment this is a dirichlet$(1, 1, \ldots, K+1)$ prior where K is the number of alleles at the locus.

4.4 The Markov chain

Another set of options specify the length of the Markov chain and the types of updates performed. The general things that can be changed are he number of samples to discard before starting to sample values, **burnin**, the number of samples to take from the Markov chain, **reps**, the number of gibbs one step updates of the tree between other steps, **bet**, and the number of leaves to add remove in the Metropolis Hastings add/remove update, given by **k**. Al these are integer values.

```
Usage:
impps <positionOption1> <Other Options>

Options Used in mapafs
 positionOption1 ()        Positional Option 1: Input File
  --help                   produce help message
  --version                version number
  --N (positiveuniform)    Prior for N
  --T (5)                  Time population existed (generations)
  --bet (20)               gibbs steps between samples
  --burnin (100)           Burn in
  --d (out.data)           File name for data file
  --freq ()                File name for frequency data
  --k (5)                  Number to add and remove
  --logN (false)           logarithmic update for N?
  --reps (1000)            Replicates
  --seed (1)               Random number seed (positive integer)
  --self (uniform(0,1))    Prior for s
  --startN (0)             starting value for N (0 for sample from prior)
  --starts (0)             starting value for s (0 for sample from prior)
  --stem (res)             Stem for output files
  --tuneN (10)             Tuning parameter for N
  --tunes (0.2)            Tuning parameter for s
  --varf (false)           Variable Frequency Data?
```

Figure 4: Options returned by `impps --help`

4.4.1 Updating N and s

Additional options are the tuning parameters for update of *self* and *N*, which are given by **tuneN** and **tunes**, and an option to update *N* on a logarithmic scale which is a switch given by **logN**.

5 A Toy Example

```
1  B/B F/F D/C A/A B/B A/A
2  C/C B/B A/A B/B A/A F/F
3  A/A B/B C/C A/A A/A I/I
4  B/B A/A E/E A/A A/B F/F
5  A/A B/B B/B C/C A/A I/I
6  A/A A/A F/E A/A A/A F/F
7  A/A B/B C/C B/B A/B A/A
8  A/A B/B B/B C/C A/A I/I
9  A/A B/B B/B C/C A/A I/I
10 B/B A/B B/B A/A A/A I/I
11 A/A B/B C/C C/C B/B I/I
12 B/B F/F C/C A/A B/B A/A
13 C/C C/B A/A A/A A/B F/D
14 C/C B/B A/A B/B A/A F/F
15 A/A A/A C/C A/A A/B G/G
16 A/A B/B C/C C/C A/B I/A
17 B/B A/B B/B A/A A/A I/I
18 B/B C/B A/F B/C A/A F/C
19 A/A A/A F/F A/A A/A D/F
20 A/A B/B B/C B/B A/B I/A
```

Figure 5: Toy example data

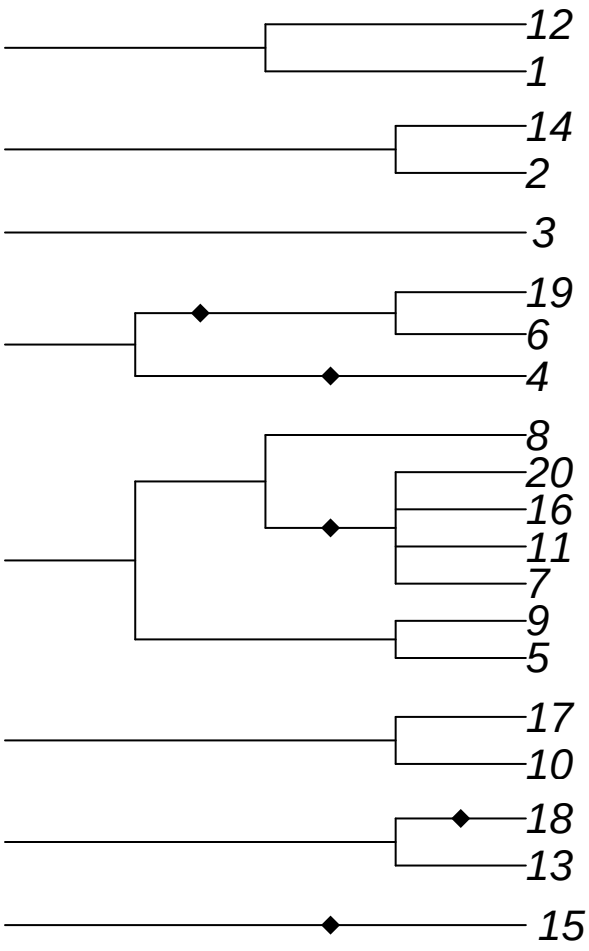

Figure 6: Toy example trees

```
1  1 2 3 4 5 4 5 5 5 6 5 1 7 2 8 5 6 7 4 5
2  1 2 3 4 5 6 7 5 5 8 7 1 9 2 10 7 8 11 6 7
```

Figure 7: matrilineal pedigree partition (line 1) and Most recent outcrossed ancestors for toy example data

To illustrate the use of the pacvkage and to show the output we consider a toy example with a sample of 20 individuals. The data are generated using the `mpsim` program which is in the same directory as the $impps$ file.

The set of commands used is in the file

We shall do a very short analysis on these data, with a burn-in of 100 and taking 200 samples, adding and removing half of the samples in the add/remove step and performing 10 gibbs updates between larger updates. This is an inadequate sample but should be relatively quick (76 seconds on my machine).

Figures 8 and 9 give very good esimates of the partition structure with, if onyl those paritioins induced by the MROA are correct if a cutoff of $0{\cdot}5$ is selected and except 1 of the matrilineal pedigree partitions are correct (14 and 2 are joined to 18 and 13).

The posterior traces seem to be relatively well behaved

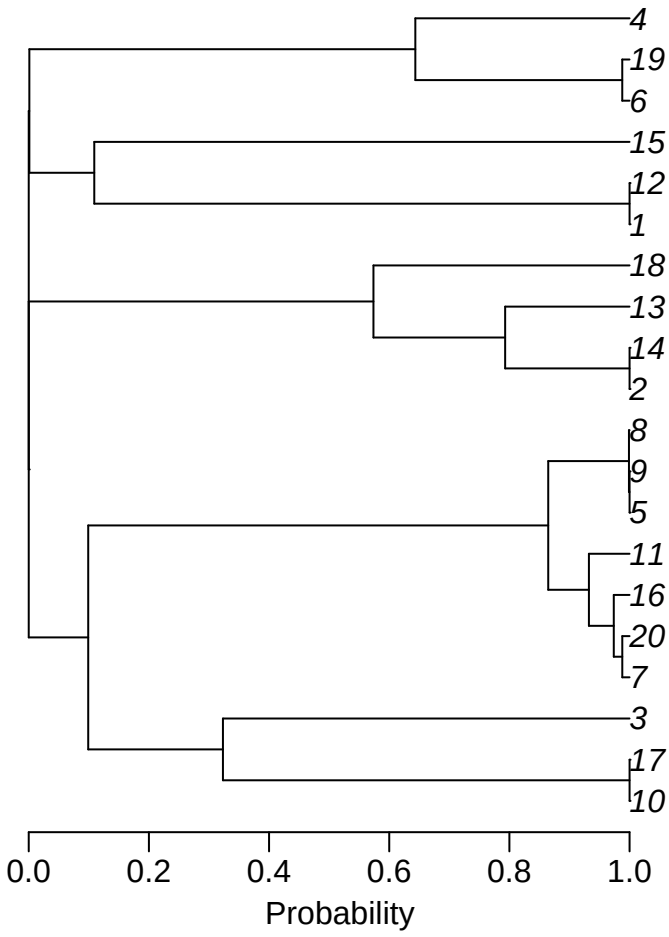

Figure 8: Partitionview output for matrilineal pedigrees from toy example.

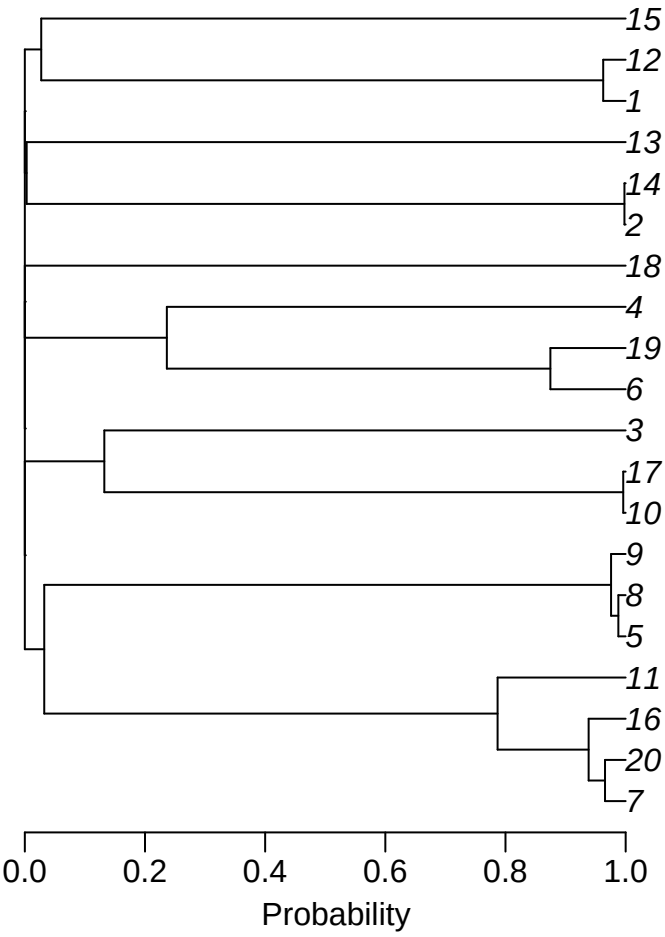

Figure 9: Partitionview output for MROA from toy example.

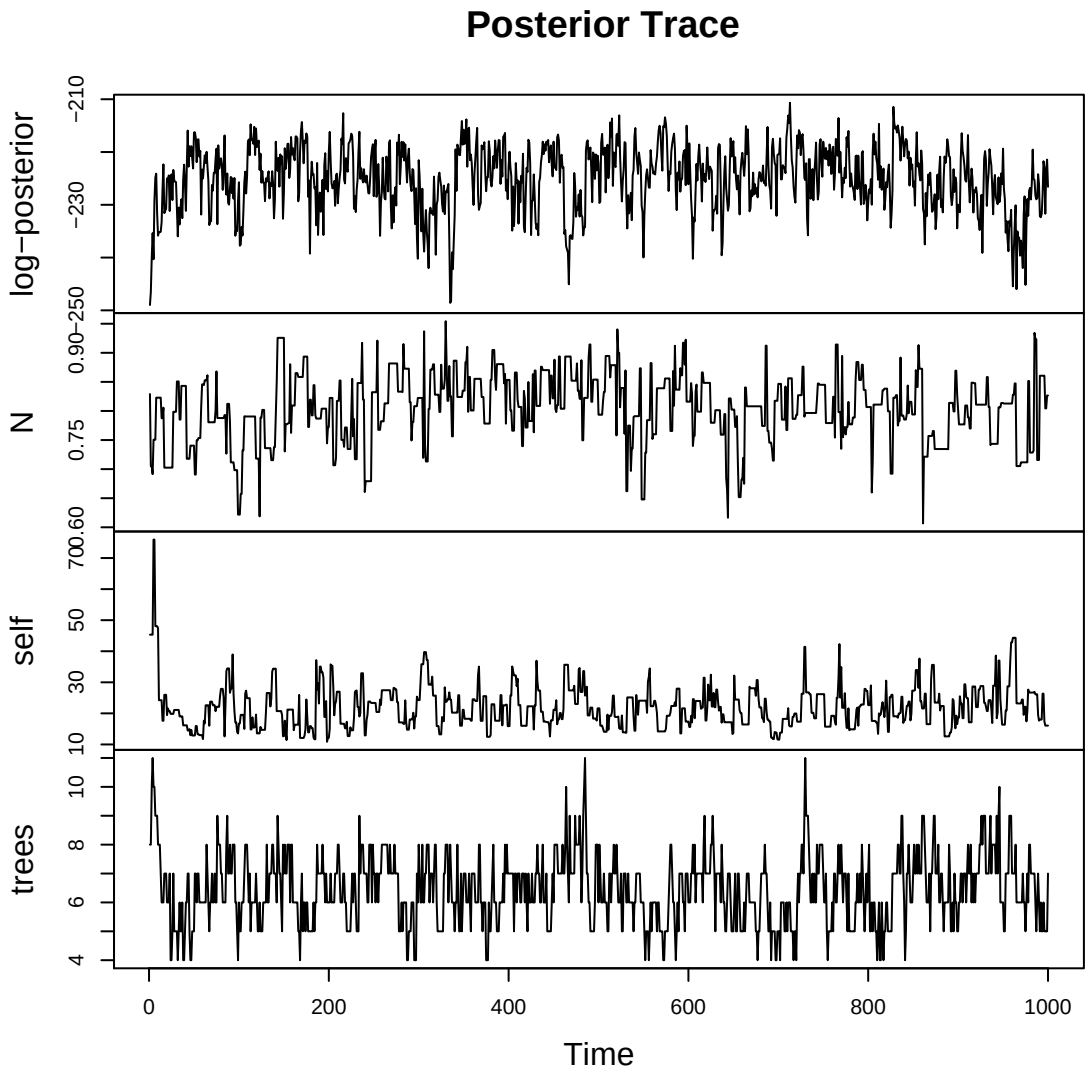

Figure 10: Traces from the posterior of toy example

6 Additional Utitities

6.1 pvconv

An additional C++ program is included that is used to translate from my output format to the partitionview format. This file should be made at the same time as the `impps` program.
It is used as:
`pvconv < infile > outfile`
where `infile` is the name of the file with the posterior partitions (say `res.partition` or `res.MROA`), and `outfile` is the name of the file to be written in the format used by `partitionview`.

6.2 fixtree.py

This is a short python script that alters the trees returned by the `paritionview` program to remove the additional root branch. It is used as:
`partitionconv < infile > outfile`
where `infile` cannot be the same as `outfile`.

6.3 pv

`pv` is a wrapper script that wraps around the console (not windows) `partitionview` program and automatically converts the output file, writes the parameter file, and fixes the outputted tree file.
`pv partitionfile treefile`.