Documentation for `PHASE`, version 2.1

Algorithm by Matthew Stephens[1], Nicholas J. Smith, and Peter Donnelly.

Code by Matthew Stephens, with contributions from Na Li.

June 2004

[1]Address for correspondance: Department of Statistics, University of Washington, Seattle. email: stephens@stat.washington.edu; WWW: http://www.stat.washington.edu/stephens/

Contents

1 Introduction **3**
- 1.1 What is new in version 2.1? . . . 3
- 1.2 What was new in version 2.0? . . . 4
- 1.3 Quick tour for existing users . . . 4

2 Getting started : Installing and running the software **5**

3 Input file format **6**
- 3.1 Missing alleles . . . 9

4 Interpreting the output **9**
- 4.1 Output to screen during run . . . 9
- 4.2 Output files . . . 10
  - 4.2.1 The summary output file . . . 10
  - 4.2.2 Additional Output files . . . 11

5 Obtaining reliable results: run-length and other parameters **13**

6 Modelling options (the `-M` flag) **14**
- 6.1 Different assumptions about recombination rate variation . . . 15
- 6.2 Notes on priors for recombination parameters . . . 16
  - 6.2.1 Altering the priors: `-r` . . . 17

7 Additional Options **18**
- 7.1 Alternative formats for the input file: `-f` and `-n` . . . 18
- 7.2 Specifying known phases: `-k` . . . 19
- 7.3 Phasing trio data: the `-P1` option . . . 20
- 7.4 Initialising estimates for the recombination parameter: `-R` . . . 21
- 7.5 Setting the seed: `-S` . . . 21
- 7.6 Performing a case-control permutation test: `-c` . . . 21
- 7.7 Multi-allelic loci without stepwise mutation: `-d` . . . 22
- 7.8 Options affecting run times and accuracy . . . 23
  - 7.8.1 Increasing the number of iterations of the final run of the algorithm: `-X` . . . 23
  - 7.8.2 Increasing the number of individuals used to estimate recombination rates: `-N` . . . 24
  - 7.8.3 Controlling the frequency threshold for a haplotype to make the list: `-F` . . . 24

7.8.4 Controlling the frequency threshold for a haplotype to be output in the _pairs file: -0 . . . 24
7.8.5 Size of segments: -l . . . 25
7.8.6 Running the algorithm multiple times automatically: -x . . . 25
7.9 Options controlling output . . . 25
7.9.1 Setting the output probability threshold: -p and -q . 25
7.9.2 Outputting a sample from the posterior distribution: -s . . . 25

8 How to cite this program 26

9 Acknowledgements 26

A Quick Tour for existing users 27
A.1 Numbers of iterations . . . 27
A.2 Performing a case-control permutation test: -c . . . 27
A.3 Modelling options (the -M flag) . . . 28
A.4 Specifying physical locations of loci . . . 29
A.5 New output files . . . 29

B Brief description of test for differences among cases and controls 31

C Benchmark Results 31

D Options not intended for general use 32
D.1 Using the Naive Gibbs sampler to estimate haplotypes (-G) . 32
D.2 Simplifying the output (-T) . . . 33
D.3 Running several data sets from the same input file (-D) . . . 33
D.4 Initialising phase guesses (-i) . . . 33
D.5 Controlling the prior annealing . . . 34

1 Introduction

The program `PHASE` implements a Bayesian statistical method for reconstructing haplotypes from population genotype data. The software can deal with SNP, microsatellite, and other multi-allelic loci (eg tri-allelic SNPs, and HLA alleles), in any combination, and missing data are allowed. The current version (v2.1) implements extensions to the original method that was described in Stephens et al. (2001). Some of the extensions are described in Stephens and Donnelly (2003), Crawford et al. (2004), and Stephens and Scheet (2005). One unpublished extension is described briefly in the appendix. Experiments with the software on both real and simulated data indicate that it can provide an improvement over other existing methods for reconstructing haplotypes - for example, in simulation experiments in Stephens et al. (2001) the mean error rate using `PHASE` was about half that obtained by the EM algorithm.

This document describes the use of this software, which comes free for non-commercial use, and with no warranty whatsoever. Commercial users require a license. Full details of the conditions of use are available via

`http://www.stat.washington.edu/stephens/software.html`,

where updates for the software will also be made available. Please send bug reports, and requests for new features, to Matthew Stephens at the email address on the title page.

In publications including results from the use of this program, please *specify the version of the software you used*. If you use haplotype estimates from the program please cite Stephens et al. (2001) and Stephens and Donnelly (2003). If you use estimates of the recombination rate from the `_recom` or `_hotspot` output files, please cite Li and Stephens (2003) and Crawford et al. (2004).

1.1 What is new in version 2.1?

The main changes from version 2.0.x are:

- the incorporation of more flexible models for recombination, including a recombination hotspot model.
- more user control over the priors used for recombination rates.
- a few minor bug-fixes, the most important being: i) a bug with the `-k` option for specifying known haplotypes; ii) a bug in the way the PAC-B correction from appendix B of Li and Stephens (2003) was being

computed, which may have created small biases in recombination rate estimates in previous versions.

- reduced memory usage requirements, which may make the program more stable for larger data sets with lots of SNPs or individuals.

- introduced the `-O` flag to control the frequency threshold at which haplotype pairs are output in the `_pairs` file (this was previously controlled by the `-F` flag).

The details of the methods used to incorporate the recombination hotspot model are described in Crawford et al. (2004).

1.2 What was new in version 2.0?

The main changes from version 1.x were:

- the introduction of a new computational approach, resulting in much faster haplotype resolution.

- the introduction of a new model that allows for recombination and decay of Linkage Disequilibrium (LD) with distance, which results in more accurate haplotype estimates.

- the facility to perform a test for haplotype frequency differences between cases and controls.

- more extensive output summarising the results, including:

  - Population haplotype frequency estimates.
  - Lists of the most probable haplotype pairs for each individual.
  - The facility to output a sample from the posterior distribution of haplotype reconstructions.

1.3 Quick tour for existing users

Users who are familiar with version 2.0.x of `PHASE` will probably only want to read the section on different recombination models (Section 6.1).

Users who are familiar with version 1.0 of `PHASE` , and want to use the new version without reading the whole of the instruction manual, might skip to the "Quick tour for existing users" in appendix A, and then read Section 6.1.

2 Getting started : Installing and running the software

I distribute executables for `PHASE` to run under Linux and Windows. Please note: I only perform extensive testing under Linux, and you are highly recommended to use that version if at all possible. The Windows version has been reported to be unstable for large data sets (for unknown reasons), resulting in the program terminating prematurely without producing results.

I also distribute source code, which you are welcome to attempt to compile on your favourite platform, but which (due both to pressures of time, and lack of expertise) I can provide only limited help with. If you do manage to get an executable working on another platform, and wish to share it with others, please email it to me, and I will make it available on the web site.

The executable comes in a gzipped tar file. Extract the files from the appropriate `.tar.gz` file, by typing (for example)

```
gunzip phase.linux.xx.tar.gz

tar -xvf phase.linux.xx.tar
```

at the command line, where `xx` is the version number. This will create a new directory called something like `phase.linux.xx`. Change to this directory before attempting to run the program.

To run the program, you must supply an input file containing the genotype information. Instructions for how to prepare an input file are given below. An example input file (`test.inp`) is supplied with the software. You can run the program on the test data supplied by typing

```
./PHASE test.inp test.out
```

(make sure that both `PHASE` and `test.inp` are in the current directory.) The program will input the genotype data from `test.inp`, and output a summary of the results to the file `test.out`. Additional output files (with names of the form `test.out_xxx`) containing more details for some of the results will also be created, and information (including a summary of progress) will be printed to the screen. Section 4 describes how to interpret the program output.

To analyse your own data, you must prepare an input file in the appropriate format. as described below, and then run the program as above. A number of additional options, which can be used to control certain aspects of how the program runs, are described in section 7. **Note: the**

most common mistake made when preparing the input file is to incorrectly input missing alleles, so pay particular attention to the instructions for this below.

3 Input file format

The input file is supplied by the user to specify how many individuals there are to be analysed, how many loci/sites each individual has been typed at, what sort of loci/sites these are (SNP or microsatellite), and the genotypes for each individual. Optionally, the file may also specify a group label for each individual (eg case/control status), and the relative physical positions of the markers.

There are three possible formats for the input file: this section describes the default format, as illustrated in the accompanying file `test.inp`. The alternative formats are described later (section 7.1).

The default structure for the input file can be represented as follows:

```
NumberOfIndividuals
NumberOfLoci
P Position(1) Position(2)      Position(NumberOfLoci)
LocusType(1) LocusType(2) ... LocusType(NumberOfLoci)
ID(1)
Genotype(1)

ID(2)
Genotype(2)

.
.
.
ID(NumberOfIndividuals)]
Genotype(NumberOfIndividuals)
```

where the quantities above are as follows:

1. `NumberOfIndividuals` An integer specifying the number of individuals who have been genotyped.
2. `NumberOfLoci` An integer specifying the number of loci or sites at which each individual has been typed.

3. `P` The character 'P' (upper case, without quotation marks).

4. `Position(i)` A number indicating the position of locus i, relative to some arbitray reference point. It is highly recommended that you use units of base pairs for this (any units can be used, but the priors are all set up to make sense in terms of base-pairs. If you use a unit other than base-pairs see the documentation on the `-R` option). The loci must be in their physical order along the chromosome (ie these Positions must be increasing). Unless you have very good reasons to do otherwise, all loci should be on the same chromosome!

5. `LocusType(i)` A letter indicating the type of locus i. The options are

   (a) `S` for a biallelic (SNP) locus, or biallelic site in sequence data.

   (b) `M` for microsatellite, or other multi-allelic locus (eg tri-allelic SNP, or HLA allele). The default assumption is that this denotes a microsatellite locus with stepwise mutation mechanism, but this can be changed using the `-d` option described later.

   These characters can be separated by spaces, if desired.

6. `ID(i)` A string, giving a label for individual i.

7. `Genotype(i)` The genotypes for the ith individual. This is given on two consecutive rows. At each locus, one allele is entered on the first row, and one on the second row. It does not matter which allele is entered on each row. For biallelic loci, any two characters (e.g. `A/C, G/T, 0/1`) can be used to represent the two alleles, and they do not need to be separated by a space. **Missing alleles at SNP loci should be entered as `?`**. For multiallelic loci a positive integer must be used for each allele (representing the number of repeats at microsatellite loci), and data for each locus should be separated by a space. **Missing alleles at multiallelic loci should be represented by `-1`.**

Some of the above items are optional, as follows:

- If you do not know, or do not wish to specify, the positions of the markers, omit the P, and the list of positions. (In this case, the loci will be assumed to be evenly-spaced if the recombination method is used.)

- If you do not wish to include individual IDs, then omit them and use the `-n` option described later.

For example, consider the example input file, `test.inp`, which is as follows:

```
3
5
P 300 1313 1500 2023 5635
MSSSM
#1
12 1 0 1 3
11 0 1 0 3
#2
12 1 1 1 2
12 0 0 0 3
#3
-1 ? 0 0 2
-1 ? 1 1 13
```

The first two numbers in the file say that there are 3 individuals typed at 5 loci. The next line indicates their relative positions along the chromosome, and the line `MSSSM` indicates that the first and last locus are microsatellites/multi-allelic, and the other loci are bi-allelic. The genotype information then follows, with three lines for each individual. The first line gives an ID for the individual, and the second and third lines give the genotypes. The third individual is missing genotype data at the first microsatellite locus, and the first SNP locus.

Notes:

1. In our example file the alleles for each locus are separated by spaces. These spaces are necessary between alleles for microsatellites and bi-allelic loci, but are optional between bi-allelic loci. In particular, for data that consist of bi-allelic loci only, no spaces need separate the alleles; e.g. 10010010 would be an acceptable input for the alleles at eight successive bi-allelic loci.

2. There should be no commas in the input file.

3. While we have used 0 and 1 to denote the two alleles at the bi-allelic loci, any two characters can be used at each locus (except for the `?`

character, used to indicate missing data). If more than two characters (plus `?`) are used at a particular bi-allelic locus then the program will behave unpredictably. If you have a tri-allelic SNP that you would like to include, then specify it as a "multi-allelic" locus (M), and use the `-d` option described below to bypass the stepwise mutation model.

3.1 Missing alleles

Missing alleles at SNP loci should be entered as `?`. Missing alleles at multiallelic loci should be represented by `-1`.

4 Interpreting the output

4.1 Output to screen during run

When run, the program initially outputs the data it has read from the input file; however, the phase at unknown positions is randomised, and any missing genotypes are imputed with random guesses, so the data will not exactly match the ones you input. Since `PHASE` currently does not check that your input file conforms to its expected format, it is highly advisable to check that the output here is consistent with what you thought you had input.

As the program continues to run, it keeps you informed of progress. Current versions of `PHASE` combine the approximate coalescent prior from Stephens et al. (2001), with a divide-and-conquer strategy known as Partition Ligation (Niu et al., 2002), as described in Stephens and Donnelly (2003). This improves the speed and accuracy of haplotype estimates over previous versions of `PHASE` , particularly for large data sets. The algorithm starts by dividing the data into segments of consecutive loci. It then computes a list of plausible haplotypes within each segment, and then iteratively combines segments to obtain a list of plausible haplotypes, and a best guess for each pair of haplotypes, across the whole region. The division into segments is random, but you can get a rough approximation of how many segments it will use by dividing the number of loci by 7 (see the `-l` option for how to change the default length of a segment). The program runs a number of "burn-in" and "main" iterations on each segment, and informs you of its progress in terms of how many "segment operations" it has completed (the total number of segment operations that it will perform is approximately twice the number of segments).

4.2 Output files

The program produces a number of output files. The first, which has the user-specified name, and a similar format to previous versions of `PHASE` , contains a summary of the individual haplotype estimates for each individual. The others, described below, have names of the form `outputfilename_xxx`, and contain more details of the results, and estimates for the sample haplotype frequencies.

4.2.1 The summary output file

The format of the summary output file (`test1.out` in the example above) is as follows.

- A header containing the version number of the software used, and credits.
- A copy of the command line used to run the program.
- A list, and brief description of the output files produced.
- A brief summary of the input file.
- A list of haplotypes in the "best" reconstruction, with a summary of the frequency with which each haplotype occured in this "best" reconstruction (note that these are not supposed to be population frequency estimates; frequency estimates are given in the `_freqs` file described below).
- A list of the best haplotype guess for each individual, with parentheses () at positions where the phase was difficult to infer, and square brackets [] around alleles that were difficult to infer. Specifically, the bracketed positions indicate those positions where phase certainty (respectively genotype certainty) was $< p$ (respectively $< q$), where the thresholds p and q can be set by the user at runtime with the `-p` and `-q` options. (e.g. use `-p0.8` to set the phase threshold to 80%.) The default thresholds are $p = q = 90\%$.) Hint: if you don't want any brackets displayed, set p and q to 0 using `-p0 -q0`.
- As above, but with only the ambigous positions listed.
- A list of the confidence probabilities associated with each phase call (with only the ambiguous positions listed).

It is worth noting that the test file supplied is very small, and there is little information in the data to allow the haplotypes to be inferred. This is reflected in a relatively low confidence being assigned to many phase calls. Realistic data sets will typically be much more informative about a large proportion of phase calls, and the results in the test file should not be viewed as typical. Nevertheless, this demonstrates the ability of the software to provide an honest assessment of the confidence it has in its calls.

4.2.2 Additional Output files

The additional output files are given names of the form `outfilename_xxx`, where the suffix `xxx` indicates the contents of the file, as follows:

_freqs Estimates of the sample haplotype frequencies (which can also be used as estimates of the population haplotype frequencies). The first two columns are self-explanatory. The next 2 columns give i) estimates (posterior means) of haplotype frequencies for the whole sample, and ii) estimated standard deviations (square root of the variance of the posterior distribution) for these frequencies. If the `-c` option is used (see below) then additional pairs of columns give estimated haplotype frequencies and standard deviations for each group specified in the input file.

_pairs List of the most likely pairs of haplotypes for each individual, together with their probability. Only pairs whose probability exceeds the threshold given by the `-O` flag are listed.

_recom Contains estimates of recombination parameters across the region, using the general model for varying recombination rate from Li and Stephens (2003). (Note: these estimates are produced only if the recombination model is used: see the `-M` option. Also, the estimates have a straightforward interpretion only if the positions of the loci are specified in the input file.) The first line of the file gives the positions of the loci in the input file (to facilitate subsequent analyses using results in this file). Each subsequent line of the file gives a sample from the posterior distribution of the recombination parameters. The first column gives estimates of the background recombination rate (more precisely, the value of the population genetics parameter ρ, here denoted $\bar{\rho}$). Column 2 gives the factor by which the recombination rate between locus 1 and 2 exceeds the background

rate, and, similarly, column i gives the factor by which the rate between locus $i-1$ and locus i exceeds the background rate. To get point estimates of these quantities I suggest taking the median of the results for each column. If you are planning to make use of the results from this file, then it is advisable to use the `-X` option described later (eg `-X10` or `-X100`), to obtain more accurate estimates.

_hotspot Contains estimates of recombination parameters if one of the hotspot options is used: see the `-M` option. The estimates have a straightforward interpretion only if the positions of the loci are specified in the input file. Each line of the file gives a sample from the posterior distribution of the recombination parameters. The first column gives estimates of the background recombination rate (more precisely, the value of the population genetics parameter $\bar{\rho}$). Column 2 gives the estimated left hand edge of the first hotspot, column 3 gives the right-hand edge, and column 4 gives the estimated intensity (the factor by which the recombination rate in the hotspot exceeds the background rate.) If this last column is 1 then this corresponds to no increase in recombination rate in the "hotspot". To get point estimates of these quantities I suggest examining both the mean and the median of the results for each column. (Histograms of each column can also be helpful in giving you an idea of the uncertainty and skewness in the posterior.) If you are planning to make use of the results from this file, then it is advisable to use the `-X` option described later (eg `-X10` or `-X100`) to obtain more accurate estimates.

_signif Gives the p-value for a permutation test of the null hypothesis that the cases and controls are random draws from a common set of haplotype frequencies (only if the case-control status is specified for each individual; see the `-c` option below)).

_monitor The monitor file for the goodness of fit of the estimated haplotypes to the underlying model. It has two columns: the first is the pseudo-likelihood from Stephens and Donnelly (2003); the second is the PAC-B likelihood from Li and Stephens (2003). When comparing behaviour of different runs from different starting points I recommend using the first column, as the second column can be very dependent on the order of the individu-

als, which can vary a lot across runs. This file replaces the ‘‘temp.monitor’’ file produced by previous versions of PHASE .

PHASE will also produce an empty file with suffix _hbg. You can ignore this.

Sharp-eyed users may note that for some data sets the "best guess" for an individual in the _out does not exactly match the most likely pair in the _pairs file. These two point estimates are different attempts to summarise the posterior distribution, using two different loss functions: the "best guess" attempts to make the haplotypes as similar to the truth as possible, while the most likely pair in the _pairs file attempts to maximise the probability of getting the haplotype complete correct. In many cases the two guesses will be exactly the same, and in comparisons the average performance of both methods was almost identical (with perhaps a slight edge for the most likely pair in the _pairs file).

5 Obtaining reliable results: run-length and other parameters

In common with many other existing haplotype inference procedures, PHASE employs an iterative scheme to perform inference. Parameters that control the number of iterations performed can be added to the input line, as follows:

```
./PHASE <filename.inp> <filename.out> <number of iterations>
                <thinning interval> <burn-in>
```

For example, to set

```
<number of iterations>=100
<thinning interval>=1
<burn-in>=100
```

use

```
./PHASE <filename.inp> <filename.out> 100 1 100
```

In fact, the above values are the default values. Each iteration consists of performing <thinning interval> steps through the Markov chain, and each step updates each individual once.

The number of iterations required to obtain accurate answers depends on the complexity and size of the data set. In order to obtain reliable results it is helpful to apply the algorithm multiple times (I suggest 5 times or more), with different seeds for the random number generator (see the `-S` option), and check for consistency across results. I suggest two ways for checking for consistency:

- Examine the haplotype frequency estimates in the `_freqs` output file, and check that the estimates across runs are consistent (within a tolerance that will depend on your particular application).
- Examine the goodness-of-fit measure, output in the `_monitor` file. This measures the goodness of fit to the estimated haplotypes to an approximate coalescent model (with recombination if the recombination model is used).

If there are substantial differences between runs by these criteria try increasing the lengths of the runs, by increasing either the number of iterations, or the thinning interval. If independent runs continue to give different results within the computer time that you are prepared to expend, I suggest using the results from the run with the highest average value for the goodness of fit.

If all this seems like too much effort, you can get the program to automatically run several independent runs, and output the answer corresponding to the run with the best average goodness-of-fit, using the `-x` option. However, in this case you will not be able to examine the results of the independent runs to see how variable they are.

6 Modelling options (the `-M` flag)

`PHASE` provides two main models for the user to choose between for haplotype reconstruction. The first is the original model in Stephens et al. (2001), and Stephens and Donnelly (2003) and ignores recombination (or more precisely, the decay of LD with distance). The second is a new model that takes makes explicit allowance for recombination. The method to be employed is specified by the `-M` flag: `-MS` specifies the old model, and `-MR` specifies the new "recombination" model (see below for details of different possible assumptions for the underlying recombination rate in the recombination model). For example, to run the test example under the old (non-recombination) model, use

```
./PHASE -MS test.inp test.out
```

If the `-M` flag is not used, the program uses the new recombination model as the default, which seems generally to be slightly more accurate, but is more computationally-intensive.

In addition to these two models, a hybrid model is also available, which uses the faster original model for the preliminary computations, and then the recombination model for the final computations. This hybrid method can be specified by using `-MQ` flag. In the benchmark test results given in an appendix, this hybrid method produces haplotype estimates that are roughly as accurate as the `-MR` option, but runs more quickly. However, other (limited) comparisons suggest that the `-MR` option performs better in general, and this is thus the default method. For another way to decrease the time taken for computations see the `-F` option.

Notes:

1. in general the recombination (`-MR`) method will be less confident in its calls than the (`-MS`) method. This is not a defect of the method, but a reflection that it makes less stringent assumptions.

2. if you have only a small data set (say $<$ about 20 individuals and $<$ about 10 loci), the `-MR` method may struggle to get reliable estimates for the decay of LD (ie recombination rate parameters). In such cases you should consider either using the `-MS` option, or altering the prior on the background recombination rate to be more informative (see `-r` below).

6.1 Different assumptions about recombination rate variation

New in v2.1, `PHASE` allows the user to specify different models for variation in the underlying recombination process. Of particular note, the user can now use `PHASE` to assess the support for the existence of a hotspot in the region being analysed, either specifying a location for the putative hotspot, or allowing `PHASE` to estimate a putative location. Some of these options were used in Crawford et al. (2004) and Ptak et al. (2004). The available models are as follows:

`-MR0`: the default model, which is the general model for recombination rate variation in Li and Stephens (2003).

`-MR1 n`: n hotspots in unknown positions to be estimated. Currently only $n = 1$ is permitted, so the model allows for at most one hotspot.

`-MR2 n a1 b1 ... an bn`: n hotspot in specified positions, with left and right hand edges of ith hotspot being given by ai and bi. Currently only $n = 1$ or $n = 2$ are permitted (ie at most 2 hotspots). The prior is that each hotspot exists (independently if there are 2) with probability 0.5; otherwise recombination rate in the hotspot region is equal to the background rate.

`-MR3`: constant recombination rate (estimated by `PHASE` from data).

`-MR4`: constant recombination rate (fixed at value specified by user, using `-R` option).

For example, to run `PHASE` with a single putative hotspot between positions 4200 and 5400, use

```
./PHASE -MR2 1 4200 5400 infilename outfilename
```

For the hotspot options (`-MR1` and `-MR2`) the `_hotspot` file contains samples from the posterior for estimated hotspot location (left and right boundaries) and intensity. Note that the estimated intensity is allowed to be 1. If we define a hotspot to be "intensity greater than x times the background rate" then the proportion of times the estimated intensity is $> x$ gives an estimate of the posterior probability for a hotspot.

Note: when estimating the recombination parameters, PHASE uses only 100 of the entered individuals (randomly chosen) because this is typically enough to get good estimates, and because the computation increases quadratically with the number of individuals. To change this, use the `-N` option, documented later.

6.2 Notes on priors for recombination parameters

Advanced users who are mostly interested in estimates of the recombination parameters may wish to play around with some of the priors (either to check for sensitivity to priors, or because they don't like the default priors.)

The form for the prior on the background recombination parameter, $\bar{\rho}$, is that $\log(\bar{\rho})$ is normal with mean $\log(\mu)$ and standard deviation σ (truncated so that $\bar{\rho}$ is forced to lie between 10^{-8} and 10^{3}). The value of σ is $0.5 * \log(f)$, where f is chosen so that you would typically (95% of the time) expect $\bar{\rho}$ to be within a factor f of μ. The default value of f is large (10^6) which makes the prior approximately uniform. If your data set is small then there may be little information about $\bar{\rho}$ in the data, and you may find the resulting estimates of $\bar{\rho}$ vary widely to include values that you would find *a priori*

implausible. If this is the case then you can make the prior more informative by reducing f (to 100 say).

The default value for μ is 0.0004, which is a typical value for humans based on $\bar{\rho} = 4N_e c$, where N_e is roughly 10 000 for humans, and c (the recombination probability per base pair) is roughly 10^{-8} (1 cM/Mb). You can adjust this up or down, according to what you know about the recombination rate or effective population size for your organism compared with humans.

For the `-MR1` option the default prior is that "hotspots" (defined here, for convenience, to be *any* increase over the background rate) occur as a Poisson process of one every 50 000 bp, and the intensity of the hotspot is allowed to vary from 1 to 1000 (uniform on the log scale). The width of the hotspot has a truncated normal distribution, truncated to lie above 200 bp, with standard deviation 2000 bp (reflecting a prior belief that most hotspots will be less than 4kb wide). The prior distribution for the hotspot center is uniform between the first and last marker. (In Crawford et al. (2004) the whole hotspot was constrained to be entirely within the two most distal markers, but we have now dropped this constraint, so that either edge of the hotspot may fall outside the two most distal markers.)

6.2.1 Altering the priors: `-r`

You can specify new values for some aspects of the priors discussed above, by creating a file containing those values and then specifying this file name straight after the `-r` flag, with no intervening space. For example, if the file `eg.prior` contains the prior information then use `-reg.prior`.

The structure of the file is that it should be plain text (a `.txt` file in Windows) and contain one number on each line, in the following order:

```
Prior guess for rho (mu above)
Factor by which rho might deviate from the prior guess (f above)
Maximum hotspot intensity
Standard deviation of normal prior for hotspot width
Minimum size of hotspot
Expected distance between hotspot centers
```

For example, the following file would correspond to the default prior:

```
0.0004
1000000
1000
```

```
2000
200
50000
```

For the `-MR2` option, only the first three numbers are used. If you are using an option without hotspots then only the first two numbers are used.

7 Additional Options

Note that in what follows, all options are case-sensitive, so for example `-s` is different from `-S`.

7.1 Alternative formats for the input file: `-f` and `-n`

The options `-f` and `-n` allow the user to specify alternative formats for the input file.

`-f` : Used to specify format of genotypes in the input file. Options are:

- `-f0` : The default format described above, with two lines per genotype.
- `-f1` : Alternative format in which the genotypes are listed on a single line, locus by locus.
- `-f2` : (SNP data only). Format in which each genotype is indicated by a single character, with heterozygotes denoted by 'H', and homozygotes denoted by the allele they are homozygous for. Note that if there is a site in the data at which there are no minor-allele homozygotes, then `PHASE` will automatically use the character that follows the major allele alphabetically to denote the minor allele.

`-n` : Used to indicate that there are no IDs in the input file.

These options may be used alone, or in conjunction. For example, the file `test2.inp` supplied with the software, contains the same data as `test.inp`, with no IDs, and with the genotypes given on a single line, locus by locus, and the program can be run using this input file by entering

```
./PHASE -n -f1 test2.inp test2.out
```

at the command line.

7.2 Specifying known phases: `-k`

The `-k` option allows the user to specify that some of the phases are known (e.g. from allele-specific PCR or family studies). Note that this option works in exactly the same way as the `-s2` option in `PHASE` 0.2x, which it replaces. Including some known haplotypes in the sample can considerably improve performance.

There are two ways to use this option. The first is for the user must create a file which specifies the known phases. This file should contain one line for each individual, with a single character (either a `*, 0` or `1`) for each locus.

`*` indicates that the phase for that individual at that locus is unknown.

`0` indicates that the phase for that individual at that locus is as in the input genotypes.

`1` indicates that the phase for that individual at that locus is the reverse of the phase in the input genotypes.

The example file `eg.known` supplied with the software shows how you might specify that, for the genotype data in `test.inp`, the phases for individual 2 are known to be as in the input file (i.e. the genotypes are `12 111 2` and `12 000 3`, as in the input file), and the phases for individuals 1 and 3 are unknown. (In fact individual 1 is homozygous at some loci, so its phase is unambiguous at these positions, but the program detects this automatically.)

The name of the file specifying the known phases must follow the `-k` without using a space. For example, to use the supplied file `eg.known` with the data in `test.inp`, use

`./PHASE -keg.known test.inp test.out`.

The second usage of the `-k` option is that if *every* haplotype in the sample is known then use `-k999`. This option is intended to make it easy to use `PHASE` to estimate recombination rates from haploid data. If you use this option, by default `PHASE` will *not* perform Partition Ligation, but will do all the loci in one go. This reduces run times, but may not be a good idea if you have a lot of missing data or a lot of loci. For example, you might be concerned if multiple runs from different random seeds give very different results. To over-ride this behaviour, use the `-l` flag after the `-k` flag on the command line (eg `-k999 -l8` to perform standard Partition Ligation).

Caution: unlike previous versions, version 2.1 of `PHASE` *may* now perform properly if you specify "known phase" for a genotype that does not have both alleles present (ie has one allele missing). Thus if you are in the happy position of having known haplotypes, but with missing data, you can pair them randomly to form diploid individuals, and specify `-k999` to treat the haplotypes as known. However, this has not been carefully tested: if you observe odd behaviour please let me know.

7.3 Phasing trio data: the `-P1` option

We modified `PHASE` to allow it to take account of trio data (see Marchini et al. (2006)). The aim was to allow `PHASE` to be used to phase the HapMap trios. Due to limits of time the interface is not especially friendly, and the computational efficiency could be improved. In particular data sets with larger numbers of trios may take a long time to run.

Before I describe the interface, I want to mention an alternative approach that may be helpful for larger trio data sets, or data sets, with (slightly) larger pedigrees: i) identify the unrelated (founder) individuals. ii) use the family information to identify which phases are unambiguous in the founders. iii) specify these as known in the "known" file using the `-k` option. This approach does not take account of all the information available in the family data, but it may work in situations where the `-P1` option I describe here does not.

To use the `-P1` option you need to:

1. format your input file so that the parents are first, and then the children come after. The order is such that the first child is the child of the first 2 parents; the second child is the child of the next 2 parents etc. eg: for two trios you might use the order

   ```
   Mother1
   Father1
   Mother2
   Father2
   Child1
   Child2
   ```

   (Note that the format of the input file is unchanged; the above is intended to illustrate the *order* in which individuals must appear in the input file.)

2. 2. Set the "number of individuals" in the input file (the first line) to be equal to the number of *parents*. eg if you have two trios, that is 4 individuals.

3. Call `PHASE` with the `-P1` option; eg:

```
PHASE -P1 inputfile outputfile
```

7.4 Initialising estimates for the recombination parameter: `-R`

The `-R` option can be used to set the initial estimate of the background recombination parameter (often denoted ρ in the population genetics literature, and here denoted $\bar{\rho}$ to emphasise that it is the background rate). You probably want to use this option only if you input the positions of your markers in units of something other than base-pairs.

The units of $\bar{\rho}$ are "per physical distance unit used to input the positions". Thus, if you use base-pairs as your physical distance unit in the input file then the units of $\bar{\rho}$ are "per base pair". The default value for $\bar{\rho}$ is 0.0004, which is appropriate for humans if the locus positions are input in base pairs. If you use another unit to enter your positions then you probably want to change this. For example, if you input your positions in kilobases, use `-R0.4` to set $\bar{\rho}$ to be 0.4 per kb (again, generally appropriate for humans).

Note: If you are testing `PHASE` on simulated data produced by a coalescent simulator that outputs locus positions so that the total region is scaled to be of length 1.0, then I suggest using `-R1` as a default setting.

7.5 Setting the seed: `-S`

The `-S` option can be used to set the value of the seed of the pseudo-random number generator. The value of the seed should be a positive integer, and must follow the `-S` without a space, as in:

```
./PHASE -S3253 test.inp test.out
```

which will set the seed to be 3253.

7.6 Performing a case-control permutation test: `-c`

Version 2 of `PHASE` can perform a permutation test for significant differences in haplotype frequencies in case and control groups. More precisely, `PHASE`

tests the null hypothesis that the case and control haplotypes are a random sample from a single set of haplotype frequencies, versus the alternative that cases are more similar to other cases than to controls. By taking the similarity of haplotypes into account - rather than just whether two haplotypes are the same or different - the test can have power even when every haplotype in the sample is unique (which other similar tests, based on a more traditional likelihood ratio for example, do not). To perform the permutation test you must do two things:

1. specify the case-control status of each individual in the input file by putting a "0" or a "1", followed by a space, *just before the individual's identifier* (or, if you are not including identifiers in the input file, before the genotype data for the individual). See the file `test.casecontrol.inp` for an example.

2. use the `-c` flag when running the program to tell `PHASE` that the input file contains case-control status data.

The number of permutations performed can, optionally, be specified after the `-c` flag. Eg `-c100` will perform 100 permutations. If no number is specified, the default is 100 permutations. Note that the permutation test is quite computer-intensive, so you might start with a small number of permutations to decide whether it is worth doing a larger number. Note that specifying `-c0` will produce the default behaviour of 100 permutations. If you really want to do 0 permutations (ie you do not want to do the test), but you are using an input file with case-control status information, then you should use `-c-1` (i.e. "-1 permutations"). *If your input file contains case-control information then you MUST use the* `-c` *option.*

Note:`PHASE` actually allows more than 2 groups. Simply use labels $0, 1, 2, ..., (k-1)$, to indicate which of the k groups each individual is in.

7.7 Multi-allelic loci without stepwise mutation: `-d`

The `-d` option allows the user to relax the assumption of stepwise mutation mechanism for multiallelic loci. This allows the method to be applied to multi-allelic loci other than microsatellites, or to microsatellite loci where the stepwise mutation mechanism is inappropriate. (Note that where the stepwise model is a good assumption, using it will provide more power to reconstruct haplotypes.) To do this we have implemented a more general mutation mechanism, where each mutation is a step-wise mutation with probability $1 - \delta$, but with probability δ the mutant type is uniform from

all possible alleles. The step-wise mutation mechanism thus corresponds to $\delta = 0$, while $\delta = 1$ specifies parent-independent mutation (PIM). The PIM model might be appropriate for tri-allelic SNPs, or, absent a better alternative, for HLA allele designations, for example.

There are two ways to use this option. First, if every multi-allelic locus in your dataset is better modelled by PIM than by a stepwise model then you can simply specify `-d1`. Otherwise, you must create a file that contains the values for δ for every locus or site in your input file, with the values separated by spaces. Note that in this case you must supply values for δ even for bi-allelic markers/sites in your input file, although they will be ignored.

For example, to specify that the first locus in `test.inp` is a microsatellite with stepwise mutation mechanism, and the final locus is a multiallelic marker for which parent-independent mutation is appropriate, the user would supply an extra file consisting of the single line

```
0 0 0 0 1
```

to specify that the first locus has a step-wise mutation mechanism, while the last locus has a parent-independent mutation mechanism (which might be considered appropriate for multi-allelic SNPs for example). See the file `eg.delta` provided with the software. The name of this file should immediately follow the `-d` flag with no intervening space. For example,

```
./PHASE -deg.delta test.inp test.out.
```

Note that for microsatellite loci where the stepwise model is a good assumption, using that model ($\delta = 0$) will provide more power to reconstruct haplotypes than other models ($\delta > 0$). However, using a small value for δ (e.g. $\delta = 0.05 - 0.1$) will provide more robustness to occasional deviations from the stepwise model.

7.8 Options affecting run times and accuracy

7.8.1 Increasing the number of iterations of the final run of the algorithm: `-X`

The `-X` flag allows the user to make the final run for the whole dataset (all loci) longer than for the other runs. For example, `-X10` will make the final runs 10 times longer than other runs (i.e. 10 times the burn-in, and 10 times the number of main iterations specified). May be useful for obtaining better estimates of uncertainty without greatly increasing run-time. (I particularly

suggest using this option when you are making use of the results in the _recom or _hotspot files.)

7.8.2 Increasing the number of individuals used to estimate recombination rates: -N

To save computer time, if more than 100 individuals are entered then PHASE uses only 100 of the input individuals (randomly chosen) to estimate recombination rates (and hotspot location and intensity etc). Based on the results of Li and Stephens (2003), where only 100 chromosomes (ie 50 individuals) seemed to give pretty good estimates, this seems unlikely to affect accuracy greatly. If you want to decrease run times, and are happy with less accurate recombination rate estimates you could use -N50 to base recombination rate estimates on only 50 individuals (for example). I don't really recommend using this option to *increase* the number of individuals used (unless perhaps you are using the -k999 option) as computing times increase considerably, but if you really want to....eg -N1000 will use 1000 individuals in estimating the recombination rates.

7.8.3 Controlling the frequency threshold for a haplotype to make the list: -F

The -F flag controls the value that a haplotype's frequency estimate must surpass in order for that haplotype to be included in the "list" used when merging sections. The default (corresponding to -F0.01) is that the haplotype will be included in the list if its estimated frequency exceeds 0.01 times the inverse of the number of chromosomes in the sample (so a haplotype will be included if even a single individual is adjudged to have at least a 1 percent chance of carrying that haplotype). Smaller values (e.g. -F0.001) will produce longer lists, which means longer run times, but potentially more reliable results. Conversely, larger values (e.g. -F0.1) will produce quicker, potentially less-reliable, results. My intuition is that this may be, in some cases, a better way to reduce running times than, say, reducing the number of iterations, but I have not done a careful study (and have found examples where using this option definitely decreases accuracy).

7.8.4 Controlling the frequency threshold for a haplotype to be output in the _pairs file: -O

In the _pairs output file, only haplotype pairs whose probability exceeds the value specified after the -O flag are listed. By default, only pairs with

a probability of at least 0.01 are listed. Note: in versions before 2.1 the `-F` flag controlled both the frequency threshold for making the list (as it still does, see above), and this output threshold. The `-O` flag was introduced so that you can make the output prettier without foresaking accuracy.

7.8.5 Size of segments: `-l`

`PHASE` uses a divide-and-conquer strategy, Partition Ligation (Niu et al., 2002; Stephens and Donnelly, 2003), to improve the speed and accuracy of haplotype estimates. The algorithm starts by dividing the data into segments of consecutive loci. It then computes a list of plausible haplotypes within each segment, and then iteratively combines segments to obtain a list of plausible haplotypes, and a best guess for each pair of haplotypes, across the whole region. By default the segments are of size 6-8 loci. You can change the maximum size of a segment using the `-l` option. Eg using `-l12` will set the segments to be of size 10-12 loci.

7.8.6 Running the algorithm multiple times automatically: `-x`

The `-x` option runs the algorithm several times, starting from different starting points, and outputs the results from the run with the best average value for the "goodness of fit". Example: `-x5` runs the algorithm 5 times.

7.9 Options controlling output

7.9.1 Setting the output probability threshold: `-p` and `-q`

The `-p` and `-q` options can be used to set a thresholds at which phases and genotypes are called as "correct". The default for each is 90%. For example,

```
./PHASE -p0.7 test.inp test.out
```

would set the phase call threshold to 70%. See section 4.2 for more details.

7.9.2 Outputting a sample from the posterior distribution: `-s`

You can output a sample from the posterior distribution on haplotypes using the `-s` option. The estimated haplotypes are then output for every main iteration of the final merge operation (so, with the default run-lengths, a sample of 100 draws from the posterior distribution of the haplotypes is output). You will get a warning that this can create large files. The samples

can be helpful in taking into account uncertainty in haplotype reconstructions in future analyses (see Stephens et al. (2001) for brief discussion), or in examining features of the posterior distribution not included in other output files.

8 How to cite this program

In publications including results from the use of this program, please *specify the version of the software you used*. If you use haplotype estimates from the program please cite Stephens et al. (2001) and Stephens and Scheet (2005). If you use estimates of the recombination rate from the `_recom` or `_hotspot` output files, please cite Li and Stephens (2003) and Crawford et al. (2004).

9 Acknowledgements

The software makes use of the Mersenne twister random number generator Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura (`http://www.math.keio.ac.jp/~matumoto/MT2002/emt19937ar.html`).

The software makes use of the TNT matrix library (`http://math.nis.gov/tnt/`).

Appendices

A Quick Tour for existing users

This appendix is for existing users, who are familiar with version 1.0 of `PHASE` , and want to use the new version without reading the whole of the instruction manual.

A.1 Numbers of iterations

For several reasons, the new version of the algorithm requires many fewer iterations than the old version to produce reliable results. This is both because i) the new version updates every individual in an "iteration", whereas the old version updated a single individual in each iteration; ii) the new version makes use of partition ligation (Niu et al., 2002; Stephens and Donnelly, 2003) to break the problem up into smaller groups ("segments"), and then paste the results back together. The program performs the number of iterations specified for each of the segments, and for each pasting together operation. Thus, depending on the number of segments (which in turn depends on the number of loci), the number of iterations performed in total may be many times more than the number specified. The default number of iterations is now 100, with 100 burn-in, and a thinning interval of 1. As before, the user is advised to run the program several times, with different seeds for the random-number generator, and check for consistency of results across runs.

A.2 Performing a case-control permutation test: `-c`

Version 2.1 of `PHASE` can perform a permutation test for significant differences in haplotype frequencies in case and control groups. More precisely, `PHASE` tests the null hypothesis that the case and control haplotypes are a random sample from a single set of haplotype frequencies, versus the alternative that cases are more similar to other cases than to controls. By taking the similarity of haplotypes into account - rather than just whether two haplotypes are the same or different - the test can have power even when every haplotype in the sample is unique (which other similar tests, based on a more traditional likelihood ratio for example, do not). To perform the permutation test you must do two things:

1. specify the case-control status of each individual in the input file by putting a "0" or a "1", followed by a space, *just before the indi-*

vidual's identifier (or, if you are not including identifiers in the input file, before the genotype data for the individual). See the file `test.casecontrol.inp` for an example.

2. use the `-c` flag when running the program to tell `PHASE` that the input file contains case-control status data.

The number of permutations performed can, optionally, be specified after the `-c` flag. Eg `-c100` will perform 100 permutations. If no number is specified, the default is 100 permutations. Note that the permutation test is quite computer-intensive, so you might start with a small number of permutations to decide whether it is worth doing a larger number. Note that specifying `-c0` will produce the default behaviour of 100 permutations. If you really want to do 0 permutations (ie you do not want to do the test), but you are using an input file with case-control status information, then you should use `-c-1` (i.e. "-1 permutations"). *If your input file contains case-control information then you MUST use the* `-c` *option.*

Note:`PHASE` actually allows more than 2 groups. Simply use labels $0, 1, 2, ..., (k-1)$, to indicate which of the k groups each individual is in.

A.3 Modelling options (the `-M` flag)

`PHASE` 2.1 provides two main models for the user to choose between for haplotype reconstruction. The first is the original model in Stephens et al. (2001), and ignores recombination (or equivalently, the decay of LD with distance). The second is a new model that takes makes explicit allowance for recombination. The method to be employed is specified by the `-M` flag: `-MS` specifies the old model, and `-MR` specifies the new "recombination" model. For example, to run the test example under the old (non-recombination) model, use

```
./PHASE -MS test.inp test.out
```

If the `-M` flag is not used, the program uses the new recombination model as the default, which seems generally to be slightly more accurate, but is more computationally-intensive.

In addition to these two models, a hybrid model is also available, which uses the faster original model for the preliminary computations, and then the recombination model for the final computations. This hybrid method can be specified by using `-MQ` flag. In the benchmark test results given in an appendix, this hybrid method produces haplotype estimates that are roughly as accurate as the `-MR` option, but runs more quickly. However,

other (limited) comparisons suggest that the `-MR` option performs better in general, and this is thus the default method. For another way to decrease the time taken for computations see the `-F` option.

Notes:

1. in general the recombination (`-MR`) method will be less confident in its calls than the (`-MS`) method. This is not a defect of the method, but a reflection that it makes less stringent assumptions.

2. if you have only a small data set (say $<$ about 20 individuals and $<$ about 10 loci), but have reason to expect that there will be moderately strong LD between markers, then the `-MS` method might be more appropriate, as the `-MR` method may struggle to get reliable estimates for the decay of LD (ie recombination rate parameters) from such small data sets.

A.4 Specifying physical locations of loci

To assist `PHASE` in taking into account the expected decay of LD with distance into account in inferring haplotypes, you can optionally specify the physical positions of the loci in the input file. To do this, include a line that begins with a "P" (for position), and then a space-delimited list of the positions of the sites (eg in base-pairs, although any units are fine). This line must go after the specification of the number of loci, and before the specification of the loci types. See the file `test.inp` for an example. Note: in the current version, the loci must appear in the order that they occur along the chromosome (ie the positions must be increasing). If no positions are input then the software assumes that the loci are equally-spaced.

A.5 New output files

In addition to the output file produced previously (whose format has changed very slightly, so any programs that parse the results in this file may need minor alteration), the program now outputs several other output files, whose names are your output file followed by a suffix that consist of an underscore character (_), and a description. These are as follows:

_freqs Estimates of the sample haplotype frequencies (which can also be used as estimates of the population haplotype frequencies). The first two columns are self-explanatory. The next 2 columns give i) estimates (posterior means) of haplotype frequencies for

the whole sample, and ii) estimated standard deviations (square root of the variance of the posterior distribution) for these frequencies. If the `-c` option is used (see below) then additional pairs of columns give estimated haplotype frequencies and standard deviations for each group specified in the input file.

_pairs List of the most likely pairs of haplotypes for each individual, together with their probability. Only pairs whose probability exceeds the threshold given by the `-F` flag are listed.

_recom Contains estimates of recombination parameters across the region, using the general model for varying recombination rate from Li and Stephens (2003). (Note: these estimates are produced only if the recombination model is used: see the `-M` option. Also, the estimates have a straightforward interpretion only if the positions of the loci are specified in the input file.) The first line of the file gives the positions of the loci in the input file (to facilitate subsequent analyses using results in this file). Each subsequent line of the file gives a sample from the posterior distribution of the recombination parameters. The first column gives estimates of the average background recombination rate (more precisely, the value of the population genetics parameter $\bar{\rho}$). Column 2 gives the factor by which the recombination rate between locus 1 and 2 exceeds the background rate, and, similarly, column i gives the factor by which the rate between locus $i-1$ and locus i exceeds the background rate. To get point estimates of these quantities I suggest taking the median of the results for each column. If you are planning to make use of the results from this file, then it is advisable to use the `-X` option (eg `-X10`), to obtain more accurate estimates.

_signif Gives the p-value for a permutation test of the null hypothesis that the cases and controls are random draws from a common set of haplotype frequencies (only if the case-control status is specified for each individual; see the `-c` option)).

_monitor The monitor file for the goodness of fit of the estimated haplotypes to the underlying model. (The goodness of fit measure is the pseudo-likelihood from Stephens and Donnelly (2003).) This file replaces the ``temp.monitor'' file produced by previous versions of PHASE .

For more details of additional program options, see the main text of these instructions.

B Brief description of test for differences among cases and controls

Here we describe briefly this test, which is currently unpublished. The descriptions is brief, and not intended to be comprehensitve, but I hope that it is better than nothing.

The test statistic we use is the PAC-likelihood (Li and Stephens (2003)) computed for estimated haplotypes in case individuals, multiplied by the PAC-likelihood for estimated haplotypes in control individuals, averaged over the estimated haplotypes in the main iterations of the final merge of the last two segements. By averaging over the estimated haplotypes in this way, we take account of uncertainty in haplotype estimates. Significance (p-value) is assessed by computing the same statistic under permutations of the group (case/control) labels, and counting how many of the permutations give larger values for the test statistic. This approach is essentially a likelihood ratio test using the PAC likelihood as a likelihood. Unlike the more conventional likelihood, which is based on Hardy-Weinberg Equilibrium and the unknown haplotype frequencies, the PAC likelihood takes into account the similarity of haplotypes, and the decay of LD with distance. As a result this test has power to detect a situation where the cases are more similar to one another than one would expect by chance, even if every haplotype in the sample is unique.

C Benchmark Results

I ran PHASE v2.1, with each of the model options (`-MR, -MQ and -MS`) on datasets simulated under conditions corresponding to the right-hand panel of Figure 2 in Stephens et al. (2001). We also ran both programs on the same data sets with 5% of genotypes randomly deleted (i.e. set as missing). Note that for the data sets with missing data, the program had to correctly impute all missing alleles, in addition to correctly calling phase, in order to score a haplotype as "correct". The error rates (as defined in Stephens et al. (2001)) are given in the table below. For these data sets the `-MQ` and `-MR` options gave very similar results, which are consistently slightly more accurate than the `-MS` option. In further unpublished comparisons the `-MQ` option also gives results similar to the `-MR` option, and is somewhat quicker.

The -MR option is set to be the default method, partly as it seems more principled than -MQ and partly because it might be expected to give more accurate estimates of the underlying recombination parameters (although I have not tested this).

	no missing data		
No. inds.	PHASE -MR	PHASE -MQ	PHASE -MS
10	0.45	0.45	0.46
20	0.29	0.29	0.33
30	0.24	0.24	0.28
40	0.21	0.21	0.23
50	0.16	0.17	0.19

Table 1: Results with no missing data

	missing data		
No. inds.	PHASE -MR	PHASE -MQ	PHASE -MS
10	0.46	0.46	0.47
20	0.30	0.30	0.35
30	0.26	0.26	0.29
40	0.23	0.24	0.25
50	0.18	0.18	0.22

Table 2: Results with 5% missing data

D Options not intended for general use

This appendix documents some options that are not intended for general use. Mostly they are documented for completeness, but they may be of interest to some, particularly researchers into haplotyping methods.

D.1 Using the Naive Gibbs sampler to estimate haplotypes (-G)

The -G flag forces the program to use the "Naive Gibbs" sampler (Algorithm 2 in Appendix A of Stephens et al. (2001)) to estimate haplotypes. Since this algorithm uses less information than the standard PHASE algorithm (the

prior for the Naive Gibbs sampler ignores relationships among haplotypes) it will in general be less accurate. However, it is very fast, and may be of interest for those wishing to compare the results of the two algorithms. The current implementation essentially uses $\theta\nu_h = \epsilon$ for all h (in the notation of Appendix A of Stephens et al. (2001)), and lets $\epsilon \to 0$. This captures the idea that the number of haplotypes in the population is likely very much smaller than the (typically) huge number of theoretically-possible haplotypes. Although we have not done extensive investigation, this seems to perform better than, say, a uniform prior on allele frequencies. (That said, this algorithm does not give an irreducible Markov Chain, so probably some small value of ϵ would be more sensible.)

D.2 Simplifying the output (`-T`)

The `-T` option causes the program to output, in the standard output file, only the best guess of each haplotype, with uncertainty represented by () and [], but with no other information. This can be helpful in comparing the estimated haplotypes with the truth in simulation studies.

D.3 Running several data sets from the same input file (`-D`)

If you want to perform simulation studies on many data sets, all input in a single input file, you can use the `-D` option to specify how many data sets are in the input file. eg `-D100` will do 100 data sets in the input file, one after another. I use this flag in conjunction with the `-T` option when performing simulation studies.

D.4 Initialising phase guesses (`-i`)

This option allows you to specify how `PHASE` initialises the phase estimates. Although this option is still available in `PHASE` version 2.1, the changes in the computational algorithm used means that it is less relevant than for previous versions (since the results from 2.1 should be less dependant on the starting point used). It is documented here for comleteness. There are three ways of initializing `PHASE` using the `-i` option:

`-i0` will initialize phases at each position randomly (this is the default behaviour).

`-i1filename` will initialize phases using the phases in the file `filename`. Note that the filename must immediately follow `-i1` with no intervening spaces.

This file should consist of only 0s (denoting that phase is initialised as input in the input file) and 1s (meaning the opposite) for each locus of each individual in turn.

`-i2` will initialize the phases so that the initial haplotype guesses are determined by the way the genotypes are entered in the input file.

Note that the companion option, `-z`, from previous versions of `PHASE` is not currently available in v 2.1.

D.5 Controlling the prior annealing

The initial and final values of the parameter β, which controls the probability of adding rather than multiplying haplotype probabilities during burn-in (see Stephens and Donnelly (2003)) can be set using `-b` (for initial) and `-B` (for final) options. The default values are that β starts at 1 at decays to 0 (i.e. equivalent to `-b1 -B0`). Note: you cannot change the linear decay of beta. Also, during the actual (rather than burn-in) iterations, β is set to 0 (which is not necessarily its final value over the burn-in).

References

Crawford, D., T. Bhangale, N. Li, G. Hellenthal, M. Rieder, D. Nickerson, and M. Stephens (2004). Evidence for substantial fine-scale variation in recombination rates across the human genome. *Nature Genetics 36*, 700–706.

Li, N. and M. Stephens (2003). Modelling linkage disequilibrium, and identifying recombination hotspots using snp data. *Genetics 165*, 2213–2233.

Marchini, J., D. Cutler, N. Patterson, M. Stephens, E. Eskin, E. Halperin, S. Lin, Z. Qin, H. Munro, G. Abecasis, and P. Donnelly (2006). A comparison of phasing algorithms for trios and unrelated individuals. *American Journal of Human Genetics*. To appear.

Niu, T., Z. S. Qin, X. Xu, and J. S. Liu (2002). Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *American Journal of Human Genetics 70*, 157–169.

Ptak, S. E., A. D. Roeder, M. Stephens, Y. Gilad, S. Pääbo, and M. Przeworski (2004). Absence of the TAP2 human recombination hotspot in chimpanzees. *Public Library of Science 2*, 849–855.

Stephens, M. and P. Donnelly (2003). A comparison of bayesian methods for haplotype reconstruction. *American Journal of Human Genetics 73*, 1162–1169.

Stephens, M. and P. Scheet (2005). Accounting for decay of linkage disequilibrium in haplotype inference and missing data imputation. *American Journal of Human Genetics 76*, 449–462.

Stephens, M., N. J. Smith, and P. Donnelly (2001). A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics 68*, 978–989.