# Documentation for SCAT, version 2

Software code and documentation by Matthew Stephens[1] WWW: http://github.com/steph

September 28 2004; updated September 15 2015

[1]email: mstephens@uchicago.edu

# Contents

# 1 Introduction

The program `SCAT2` (Smoothed and Continuous AssignmenTs) implements a Bayesian statistical method for estimating allele frequencies and assigning samples of unknown (or known) origin across a continuous range of locations, based on genotypes collected at distinct sampling locations. In brief, the idea is to assume that allele frequencies vary smoothly in the study region, so allele frequencies are estimated at any given location using observed genotypes at near-by sampling locations, with data at the nearest sampling locations being given greatest weight. The software can deal with SNP and microsatellite and other multi-allelic loci, in any combination, and missing data are allowed. Details of the method are given in Wasser et al. (2004).

This document describes the use of this software, which comes free of charge, and with no warranty whatsoever. Updates for the software will be made available via

`http://github.com/stephens999/scat`

Please send bug reports and requests for new features by opening an issue on that page.

In publications including results from the use of this program, please *specify the version of the software you used*, and cite Wasser et al. (2004).

# 2 Getting started: installing and running the software

I distribute executables for `SCAT2` to run under MAC OSX. Source code is available on the github page.

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

```
gunzip scat.linux.xx.tar.gz

tar -xvf scat.linux.xx.tar
```

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

The program can be used in two different ways:

- to estimate allele frequencies at specified locations, based on genotypes observed at a number of sampling locations.

- to "assign" (ie estimate the location of origin of) individuals based on allele frequencies estimated using reference samples from a number of sampling locations.

For simplicity, in this section we will illustrate the first of the above uses. See section 3.4 for instructions on using the program for assignment.

To run the program, you must supply two input files: a genotype file, containing the genotype information, and a location file, containing information on the latitude and longitude of sampling locations. To perform continuous assignment tests you must also specify the boundaries of the habitat of the individuals being assigned - see section 3.4.3. Instructions for how to prepare these files are given below. Finally you must create a directory or folder to hold your results.

Once these files are created you can run the program using

```
./SCAT2 <genotype filename> <location filename> <outputdir> L
```

where `L` is the number of loci in the genotype file.

An example genotype file (`test.genotype.toutputdirxt`) and an example location file `test.location.txt` are supplied with the software. You can run the program on the test data supplied by typing

mkdir results`./SCAT2 test.genotype.txt test.location.txt results 2`

(make sure that `SCAT2`, `test.genotype.txt`, and `test.location.txt` are all in the current directory.) Here 2 indicates that there are 2 loci in the genotype file.

The program will input the data, perform a number of iterations, and output results in files in the results directory you created. Section 5 describes the output files in more detail.

# 3 Analysing your own data

To analyse your own data, you must prepare a genotype file and location file in the appropriate format, as described below, and then run the program as above (replacing 2 with appropriate value for number of loci). A number of additional options, which can be used to control certain aspects of how the program runs, are described in subsequent sections and the appendix.

### 3.1 Genotype file format

The genotype file is supplied by the user to specify genotypes of the individuals to be analysed. The file should be a plain text (.txt) file. Its format is similar to that used by the program `structure` (see `http://pritch.bsd.uchicago.edu/`), and can be represented as follows:

```
ID(1) Location(1) Allele(111) Allele(121) Allele(131) ... Allele(1L1)
ID(1) Location(1) Allele(112) Allele(122) Allele(132) ... Allele(1L2)
ID(2) Location(2) Allele(211) Allele(221) Allele(231) ... Allele(2L1)
ID(2) Location(2) Allele(212) Allele(222) Allele(232) ... Allele(2L2)
...
...
ID(N) Location(N) Allele(N11) Allele(N21) Allele(N31) ... Allele(NL1)
ID(N) Location(N) Allele(N12) Allele(N22) Allele(N32) ... Allele(NL2)
```

where

1. `ID(i)` is a string, giving a label for individual $i$.
2. `Location(i)` is an integer specifying the "Location number" of the sampling location of individual $i$ (the location number for each location is specified in the location file; see below). Individuals of unknown origin should be given a location number of -1.
3. `Allele(il1),Allele(il2)` are the two alleles in individual $i$ at locus $l$. **Missing alleles should be represented by `-999`.**

See the example genotype file, `genotype.test.inp`, for an illustration.

Some things to note:

1. There should be no commas separating columns, just white space.
2. ID's for each individual must not contain any spaces.
3. Any non-negative integer can be used to denote the alleles at a locus. For SNP alleles the obvious choice is 0 and 1, but any two numbers can be used, and it need not be the same two numbers in each column (so recoding SNP data using A=1,C=2,G=3 and T=4 is fine). For microsatellites you can use allele length, or number of repeats, or indeed any set of integers.
4. The software assumes independence of loci (no Linkage Disequilibrium), so it is advisable to avoid markers that are very closely linked.

### 3.2 Location file format

The location file is supplied by the user to specify the latitude and longitude of sampling locations, and also of any other locations for which the user wishes to estimate allele frequencies. The file has one row for each location, and its format can be represented as follows:

```
LocationId(1)  LocationNumber(1)  Latitude(1)  Longitude(1)
LocationId(2)  LocationNumber(2)  Latitude(2)  Longitude(2)
...
LocationId(S)  LocationNumber(S)  Latitude(S)  Longitude(S)
```

where

1. `LocationId(s)` is a string, giving a label for location $s$ ($s = 1, \ldots, S$).
2. `LocationNumber(s)` is an integer specifying the "Location number" of sampling location $s$. These locations numbers can be any $S$ distinct integers. Typically they will simply increase from 1 to $S$ (so the location number of sampling location $s$ will be $s$).
3. `Latitude(s) Longitude(s)` are the *decimal* latitude and longitude of location $s$. Note that N and E are entered as positive numbers, and S and W are entered as negative. So, for example, $47°39'N$ and $122°18'W$ become 47.65 and -122.30.

See the example location file `test.location.txt` for illustration.

### 3.3 Estimating allele frequencies

To estimate allele frequencies at all locations in the location file, simply run the program as in section 2 above, using

```
./SCAT2 <genotype filename> <location filename> <outputdir> L
```

### 3.4 Assigning individuals

Individuals can be assigned in one of two ways: either the `-A` option or `-M` option. The `-M` may be more convenient when assigning samples of genuinely unknown origin and `-A` might be more useful when doing cross-validation.

### 3.4.1 -A option

In the `-A` option individuals to be assigned are included in the genotype file, on consecutive lines. Individuals of unknown origin should be given a location number of -1 in the genotype file.

To perform assignments for individuals $i$ to $j$ in the input file, simply add `-A i j` after the `SCAT2` command. For example, to perform assignments on the first, second and third individuals in the genotye file use

```
./SCAT2 -A 1 3 <genotype filename> <location filename> <outputdir> L
```

Using this `-A` option causes `SCAT2` to perform two types of assignments: a smoothing assignment, and a continuous assignment.

The smoothing assignment uses allele frequencies estimated at each location in the location file to compute the probability of each individual's genotype in each location, and outputs the results in the `_probs` file (see section 5.2). Note that these probabilities are also computed even if the `-A` option is not used; the difference is that when the `-A` option is invoked these probabilities are computed using leave-one-out cross-validation (ie the genotypes of individual $i$ are ignored when estimating allele frequencies for assigning individual $i$).

The continuous assignment uses allele frequency estimates to assign individuals, allowing that an individual's location of origin may be anywhere in a continuous region (so, in particular, the location may be somewhere other than any of the locations in the location file). The range of allowable locations should be specified using a boundary file, as described below. The software outputs a number of estimates for the possible location of each individual, one for each iteration – both burn-in and main iterations – of the MCMC scheme. The results for the burn-in iterations (the first 100 values under the default parameter, since Nburn =100 by default - see section 4) should be discarded, and the median latitude and longitude of the remaining samples can be used as a point estimate for the individual's location, and the spread of the points give an indication of the uncertainty. The results for each individual are saved in a file whose name is the same as the individual id. (Note that the third column in these files gives a measure of the log likelihood, which can be used to investigate mixing.)

### 3.4.2 `-M` option

Using this option you provide the individuals you want to assign in a separate file, whose name you specify straight after the `M`: that is you use `-Massignmentfile.txt` (no space betweeen M and filename).

Output details are as in the `-A` option.

### 3.4.3 Specifying the boundary file (`-B`)

When performing continuous assignments it is *highly* recommended that you specify the area within which individuals might be found. Without this individuals may be assigned to unrealistic locations (eg ocean-dwelling individuals may be assigned to the land, or land-dwelling individuals assigned to the ocean). The region can be specified as any polygon, by entering the latitude and longitude of each consecutive vertex into a "boundary" file (see below), and using the `-B` option to tell the software the name of this file. For example, if the boundary file is called `eg.boundary.txt` you should run `SCAT2` using

```
./SCAT2 -Beg.boundary.txt ...
```

Note that there is no space between the B and the file name! The boundary file should contain one row for each vertex in the polygon, with two numbers on each row (decimal latitude and longitude, with N and E being positive, S and W being negative) separated by a space. The vertices should be entered in order (in either direction around the polygon), with coordinates of the last vertex being an *exact* repeat of the coordinates of the first vertex.

For example, to specify a square region, from $5°N$ to $3°S$ and $2°W$ to $1°E$ the file could be

```
5 -2
5  1
-3 1
-3 -2
5 -2
```

Note: the routine that tests for whether a point is inside or outside the specified polygon is based on the 2-D algorithm very helpfully described by Dan Sunday (`http://www.softsurfer.com`). As such it does not take proper account of the fact that the earth is spherical. Most importantly, it will not work for regions that span $180°E$ (the line opposite the prime meridian) unless you enter the Longitudes as having the same sign, eg by using 190 and 170 instead of $-10$ and 170. It also will not work for regions including the North or South Pole. (Possibly it will be badly behaved for other regions too, but I don't have time to fix it just now - let me know if you encounter any problems.)

# 4 Obtaining reliable results: run-length and other parameters

SCAT2 employs an iterative scheme to perform inference. Parameters that control the number of iterations performed can be added to the input line, as follows:

```
./SCAT2 <genotype filename> <location filename> <output filename> N L S Niter Nthin Nburn
```

where Niter is the number of iterations of the MCMC scheme to be performed, Nthin is the steps taken through the Markov chain for each iteration, and Nburn is the number of burn-in iterations.

For example, to set

Niter=100, Nthin=10, Nburn=100

use

```
./SCAT2 <genotype filename> <location filename> <output filename> N L S 100 10 100
```

In fact, the above values are the default values. Each iteration performs Nthin steps through the Markov chain.

The number of iterations required to obtain accurate answers depends on the complexity and size of the data set, and it is helpful to do several initial runs, with different values of the seed for the random-number-generator (see -S option below), to check how many iterations are necessary for the values of $\alpha$ and $\beta$ to be reliably estimated across runs. For computational reasons, these initial runs are best done without the -A option. If there are substantial differences between parameter estimates, or in the range of log-likelihoods acheived, in different runs (output in the _params file) try increasing the lengths of the runs, by increasing either the number of iterations, or the thinning interval.

The way the results were obtained in Wasser et al. (2004) was to first to apply the method without performing assignments (ie without -A) using a relatively large number of iterations (eg Niter=100, Nthin=1000, Nburn=100), and to use results in the _param file to get estimates of $\alpha$ and $\beta$; then to do five independent sets of assignment runs with the parameters $\alpha$ and $\beta$ fixed to their estimated values (see -f below), and the default run-length settings (Niter=100, Nthin=10, Nburn=200). This might be a reasonable general strategy for performing assignments (although if the computer time is not prohibitive it might be helpful to do assignment runs

with a larger Nthin, and not fix $\alpha$ and $\beta$). Note that when fixing $\alpha$ and $\beta$ in this way, it is recommended that you first find estimates for $\alpha$ and $\beta$ from the `_params` file (eg their mean, median or mode after discarding burnin), and then choose the row of estimates in the `_params` file that is closest (in some sense) to these estimates. This will help ensure that the *combination* of parameter estimates used is somewhat sensible.

### 4.1 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` after a space, as in:

```
./SCAT2 -S 3253 ...
```

which will set the seed to be 3253.

### 4.2 Fixing model parameters (the `-f` flag)

As noted above, it may be helpful to first estimate the $\alpha$ and $\beta$ parameters in a preliminary run, and then fix them to given values during assignment runs. You can fix the values of these parameters by using `-f`, followed by a space, and then the values to use for $(\alpha_0, \alpha_1, \alpha_2, \beta)$ (four values, each separated by spaces).

For example,

```
-f 0.43 5300 0.32 2.3
```

will fix $\alpha_0 = 0.43, \alpha_1 = 5300, \alpha_2 = 0.32, \beta = 2.3$.

## 5 Interpreting the output

### 5.1 Output to screen during run

When run, the program initially outputs the data it has read from the input file. Since `SCAT2` 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 (burnin iterations, main iterations etc). It should then tell you that it is producing output files, and exit.

### 5.2 Output files

The program produces a number of output files in the specified output directory, whose names are of the form `Output_xxx`. Their contents are as follows:

`_freqs` Contains estimates (posterior means) of allele frequencies at each location in location file, at each locus. It also contains, for comparison, the empirical allele frequencies at each location.

`_probs` Contains estimates of $\log(\Pr(\text{genotype data for individual } i | i \text{ came from location } j))$, for every $i$ and $j$. Common practice is to "assign" each individual to the location that maximises this probability. Each row of the file contains the data for a single individual. The first two columns give the individual's id, and the number of loci at which it had genotypes given. The next two columns give the location numbers of the true and assigned locations for that individual. Subsequent columns give the estimates of the log probabilities for locations $j = 1, 2, \ldots$. In general you will want to use the output from this file *only* if you used the `-A` option to assign individuals. (The file is still produced even if you don't use the `-A` option, but in that case the results are not based on cross-validation, and so assignment results will be over-optimistic.)

`_params` Contains sampled values of model parameters, and log-likelihood of genotype data, for each iteration of the MCMC (parameters are output during both burnin and main iterations, every Nthin iterations). There should be 5 columns, corresponding to sampled values of $\alpha_0, \alpha_1, \alpha_2$, and $\beta$, and the log-likelihood for the corresponding allele frequency estimates.

`_accept` Contains summary of (cumulative) acceptance rates during MCMC runs. Each line contains data for one iteration of the MCMC (after thinning). Acceptance rates on each line are, in order, for $\alpha_1, \alpha_2$ (if these are updated), $X$ and $\mu$.

`_corr` Empty file, which is intended for use in future releases.

# 6 Changes since to the software since the version used to obtain the published results

A number of small changes have been made to the MCMC update scheme since the software was used to produce the results in Wasser et al. (2004). Most of these changes were made to try to improve mixing for general datasets (rather than the specific one we analysed).

# 7 How to cite this program

In publications including results from the use of this program, please *specify the version of the software you used*, and cite Wasser et al. (2004).

# 8 Acknowledgements

The software makes use of the LAPACK linear algebra routines for finding the Cholesky decomposition of a matrix, and a version of the `wn_PnPoly()` algorithm by Dan Sunday

(`http://www.softsurfer.com/Archive/algorithm_0103/algorithm_0103.htm`)

.

**Appendix**

# A Other Options

This appendix documents a few other options, rather briefly. Maybe I will get around to doing a better job some time.

`-e delta gamma` sets the values of $\delta$ and $\gamma$ in the error model (eq 1 in Wasser et al. (2004)). The defaults are $\delta = 0.05$ and $\gamma = 0$.

`-h sd` sets the proposal standard deviation for $X$ updates to `sd` times $\sqrt{1/\alpha_0}$ (default = 0.5).

`-C extracolumns` tells the program how many extra columns there are in the genotype file, between the location number and the genotype data. (Extra columns must be integers)

`-D` tells the program to use hard-coded boundaries for savanna african elephants

`-d` tells the program to use hard-coded boundaries for forest african elephants

`-j` update $X$ for an allele jointly, rather than one at a time.

`-N` includes a "nugget" effect in the covariance matrix.

`-r` use Langevin update for $X$.

`-R` remove all samples from a region when doing assignment of individuals from that region.

`-w` use smoothing only towards the mean, with no spatial component (ie set $\alpha_2 = 0$).

`-X` "cheat" by initialising location close to true location in assignment runs.

`-Z` specifies that the location file contains "superregion" information in the third column. eg it contains the "broad geographic region" labels from the PNAS paper. (See also the `-H` option below).

`-H` perform the hybrid checking assignment test. There are four options: `-H1` does the hybrid test by broad geographic region ("superregion"); `-H2` does the standard assignment test by broad region ("superregion");

`-H11` does the hybrid test by sampling location; `-H12` does the pure test by sampling location. (Probably `-H1` and `-H2` are most useful.) For `-H1` and `-H2` you would need to specify the super-region of each sampling location in the location file, and use the `-Z` option above. In our initial analyses we would first assign using the `-H2` option to test whether each sample is savannah or forest before performing subsequent analyses. For example

```
../SCAT2 -H2 -Z combined.inp ak.all.region.neworder.txt combined.out
437 16 28 0 0 0 > temp.out
```

Currently the superregions must be labelled 0,1,...5. (The number of superregions is hard-wired to be 6. I think that if you use fewer than 6 then the last few columns of the output will be garbage, but otherwise things will be fine. If you use more than 6 then it will probably crash.)

Note that this will output a bunch of stuff to the file `temp.out`. Near the end of this file there will be the line "Hybrid Probs", and then following that there will be a matrix of numbers, one row per individual, one column per location/super-region. The $(i, j)$th entry will be

$$\log(\Pr(\text{ind } i\text{'s genotype data} \mid i \text{ comes from region } j)).$$

In the case of the "hybrid" options (`-H1` and `-H11`) the columns are

$$\log(\Pr(\text{ind } i\text{'s genotype data} \mid i \text{ has parents from } (j_1, j_2))),$$

for $(j_1, j_2) = (0,0), (0,1), \ldots, (0,5), (1,0), (1,1), \ldots, (1,5), \ldots, (5,5)$.

I pasted these probabilities into a text file and used R to process them to do whatever I wanted. For example:

```
p.nse=read.table("pureprobs.nse.k20.txt")
p.nse.max = apply(p.nse,1,which.max)
p.nse.classm=as.matrix(table(x2$region+1,factor(p.nse.max,levels=1:28)))
sum(diag(p.nse.classm))
```

## References

Wasser, S. K., A. M. Shedlock, K. Comstock, E. A. Ostrander, B. Mutayoba, and M. Stephens (2004). Assigning african elephant DNA to geographic region of origin: applications to the ivory trade. *Proceedings of the National Academy of Sciences 41*, 14844–14852.