Documentation for SCAT, version 2.3

Software code and documentation by Matthew Stephens[1]
WWW: `http://github.com/stephens999/scat`

September 28 2004; updated September 15 2015; revised by Mary Kuhner[2] December 30 2020

[1]email: `mstephens@uchicago.edu`
[2]email: `mkkuhner@uw.edu`

Contents

1 Introduction 2

2 Getting started: installing and running the software 2

3 Analysing your own data 3

3.1 Genotype file format 4
3.2 Location file format 5
3.3 Estimating allele frequencies 5
3.4 Assigning individuals 5
3.4.1 `-A` option 6
3.4.2 `-M` option 7
3.5 Specifying the habitat boundary 7
3.5.1 Boundary file: `-B` option 7
3.5.2 Grid file: `-g` option 8
3.5.3 Pre-loaded Afrian elephant location data 8

4 Cautions for use of the software 8

5 Run-length and other search parameters 9

5.1 Setting the seed: `-S` 11
5.2 Fixing model parameters (the `-f` flag) 11
5.3 Setting error parameters (the `-e` flag) 11

6 Options changing the mathematics or search strategy 11

7 Interpreting the output 13

7.1 Output to screen during run 13
7.2 Output files 13

8 Subregions and hybrids 14

9 Other software useful in conjunction with `SCAT2` 15

10 Changes to the software 16

10.1 Changes between 2.0 and 2.1 16
10.2 Changes between 2.1 and 2.2 16

11 How to cite this program 17

12 Acknowledgements 17

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 nearby 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).

The program is fairly slow: data sets attempting to locate a thousand samples can take 3-4 days to run on a moderately fast machine. We recommend trial runs with fewer samples to get an idea of the time and space requirements.

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

SCAT2 was previously distributed as executables for MAC OSX and Linux, but currently only for Linux. Users of other operating systems will need to compile from source.

The Linux executable comes in a gzipped tar file. Extract the files from the `.tar.gz` file by typing

```
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 running the program.

Source code is available on the github page. To build the program from source, unpack the source code into a directory and type:

```
make
```

The Makefile as distributed makes an optimized version of the program for speed. If you need to run the program in a debugger such as `gdb`, turn off optimization and turn on debugging support by locating the following line in the Makefile:

```
CFLAGS = -O3 -DNDEBUG $(INCLUDE)
```

and inserting a hash (#) at the beginning, and locating the following line:

```
#CFLAGS = -g $(INCLUDE)
```

and removing its hash. Then type `make clean` and `make`. The resulting code will be much slower and we recommend this only if you are using a debugger. It will also do extensive checking for internal consistency; if the program runs in optimized mode but aborts in debug mode, we would very much appreciate a bug report.

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

In this section we will illustrate allele frequency estimation. 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 supply a file giving the boundaries of the habitat of the individuals being assigned - see section 3.5. 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 file> <location file> <outputdir> L
```

where `L` is the number of loci in the genotype file and ¡outputdir¿ is the path to your output directory.

An example genotype file `test.genotype.txt` and an example location file `test.location.txt` are supplied with the software. These have $L = 2$. You can run the program on the test data supplied by placing the `SCAT2` executable and the two input files in the current directory and typing:

```
mkdir results
./SCAT2_1 test.genotype.txt test.location.txt results 2
```

The program will input the data, perform a number of iterations, and output results in files in the results directory you created. Section 7 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 the the number of loci in your genotype file). 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 section 9) 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, `test.genotype.txt`, for an illustration.

Some things to note:

1. There should be no commas separating columns, just white space.
2. Individual IDs 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. No use is made of the relative values of these numbers; they are arbitrary labels. However, negative numbers other than `-999` (which will be interpreted as missing data) should not be used.

4. The software assumes independence of loci (no linkage disequilibrium), so it is advisable to avoid markers that are very closely linked. Use of closely linked markers will overstate the amount of independent information available and lead to too-tight estimates of confidence.

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:

```
LocId(1)  LocNo(1)  Latitude(1)  Longitude(1)
LocId(2)  LocNo(2)  Latitude(2)  Longitude(2)
...
LocId(S)  LocNo(S)  Latitude(S)  Longitude(S)
```

where

1. `LocId(s)` is a string, giving a label for location s ($s = 1, \ldots, S$).

2. `LocNo(s)` is an integer specifying the "Location number" of sampling location s. These location 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 an illustration.

If the -Z (subregions) option is in effect, an additional column must be added to the location file:

```
LocId(1)  LocNo(1)  SubregionNo(1)  Latitude(1)  Longitude(1)
```

The `-C` option can be used to skip the subregion data (or other extraneous columns in the location file) if subregion analysis is not desired. It takes an argument of the number of columns to skip; columns will be skipped starting with the first column after the location number.

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 file> <location file> <outputdir> L
```

3.4 Assigning individuals

Individuals can be assigned in one of two ways: either the `-A` option or `-M` option (the "assignment options"). The only difference is that with `-A` the individuals of unknown origin are embedded in the main genotype file, and with `-M` they are placed in a separate file. One or the other may be more convenient for your pipeline.

WARNING: If you set neither assignment option, individual assignments will still be produced; but no cross-validation will be done, and the certainty of the assignments will be inflated as a result.

Both assignment options perform two types of assignments: a smoothing assignment which assigns individuals to existing locations, and a continuous assignment which assigns them to geographical coordinates which may not coincide with any existing location. Note that it is possible, and sometimes desirable, to assign the location of individuals whose location is already known (for example, to test the accuracy of the algorithm).

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 7.2). Note that these probabilities are also computed even if no assignment option is used; the difference is that when an assignment 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 an individual's location of origin to be anywhere in a continuous region (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 5) should generally be discarded. 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 gives an indication of the uncertainty. The results for each individual are saved in a file whose name is the same as the individual ID. (The third column in this file gives a measure of the log likelihood, which can be used to investigate mixing.)

3.4.1 -A option

The `-A` option is used when 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. If an individual is marked for assignment but given a location other than -1, it will participate in allele frequency estimation for that location except for the purposes of its own assignment, where its frequency contribution will be dropped.

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_1 -A 1 3 <genotype file> <location file> <outputdir> L
```

3.4.2 `-M` option

This option behaves exactly like `-A`, except that rather than being included in the input file, the individuals to be assigned are in a separate file, whose name you specify straight after the `M`, with no space between `M` and the filename: `-Massignmentfile.txt`

3.5 Specifying the habitat boundary

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 land-dwelling organisms assigned to ocean).

Two methods of specifying the habitat boundary are provided: as a polygon bounding the area, or as a grid indicating all 1 degree squares within the area. The latter is recommended.

Note that if you intend to use the VORONOI program to post-process your SCAT2 results, both programs *must* use the exact same species boundaries, and should therefore use the same boundary-setting option and the same boundary data.

3.5.1 Boundary file: `-B` option

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 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 - let us know if you encounter any problems.)

3.5.2 Grid file: `-g` option

Alternatively, the habitat can be specified by dividing the area of interest into grid squares of 1 degree latitude and longitude, and listing all grid squares which could contain the organism. An advantage of this method is that it allows a discontinuous habitat, for example an island organism found on several islands but not in the ocean. Like the polygon method, it will likely not work if the habitat crosses the line opposite the Prime Meridian or includes either pole, and if the organism has a very large north-south range the distortion of the 1 degree grid squares may be problematic.

To specify the habitat in this way, prepare a file which contains one line per square included in the habitat, giving decimal latitude and decimal longitude with a space between them. The latitude and longitude describe the lower left corner of the grid square, so that an entry:

```
-12 17
```

indicates the grid square whose lower left corner is 12 S and 17 E, and whose upper right corner is 11 S and 18 E.

The grid file is then specified to the program using the `-g` option followed by the filename of the grid file:

```
-g ../savannah_grid.txt
```

Note that there is a space between the option and the filename.

3.5.3 Pre-loaded Afrian elephant location data

The program contains pre-loaded boundaries for savannah and forest African elephants. These boundaries are very loose (in particular the savannah boundary contains a good deal of ocean) and we do not recommend their use for any purpose other than replicating previous results.

`-d` use hard-coded boundaries for savannah African elephants

`-D` use hard-coded boundaries for forest African elephants

4 Cautions for use of the software

`SCAT2` carries out an MCMC search. If the search is too short or mixes poorly, the results may be inaccurate and their certainty will be overstated. The program `TRACER` (see section 9) may be useful in diagnosing poor searches.

The code which determines the species boundaries will not work correctly if the species range crosses the line opposite the Prime Meridian or includes either pole. Nothing is done by the code to diagnose such problems. The user could, with great care, rescale the latitudes and longitudes to avoid this problem (be sure to reverse the rescaling before interpreting the results).

If the species occurs as widely separated patches, using a grid file to specify correct species boundaries may cause the program to become stuck on the first solution it finds because other solutions are "too far away" in the patchy map. You can diagnose this by making multiple runs with different random number seeds: if each run places a given individual with great certainty, but different runs place this individual in different patches, this problem is likely occuring. Combining estimates across multiple runs will give a better estimate than any single run.

`SCAT2` assumes that markers are unlinked. Tightly linked markers will cause the accuracy of the results to be overstated. Sequencing individual genes and then using all variable sites within them as markers is likely to cause this problem and cannot be recommended as a way to obtain `SCAT2` input data.

`SCAT2` assumes that allele frequencies change smoothly with geographic distance. If there are strong discontinuities in gene flow, it will inappropriately use populations which are nearby but separated by the discontinuity to inform each others' allele frequencies. It may be helpful to separate the discontinuous areas into separate `SCAT2` runs, as was done for forest and savannah elephants in Wasser et al. (2004).

Hybrids between distant populations are problematic for `SCAT2` both in the reference data and, especially, in the unknown data. Hybrids between forest and savannah elephants, in pratice, yield a very wide spread of assignment locations and are sometimes pushed to the far boundaries of the species range: they do not fit anywhere, so they are put as far away from everything else as possible. It is best to pre-identify hybrids, such as with the EBhybrids program, and remove them from both the reference data set and the unknowns. This problem can sometimes be spotted in a heatmap of the `SCAT2` locations.

There are hard-coded constants for the maximum number of loci (20) and maximum number of alleles per locus (120) at the top of the program. If the user's data exceeds these limits, the program will crash or malfunction. These limits were set for microsatellite data and are unlikely to work for SNP data, which generally has far more loci and far fewer alleles. We do not set the defaults to accomodate both microsatellite and SNP data as having high values for both is memory intensive and unlikely to be necessary (except for a mixed data set with both microsatellites and SNP loci). To change these limits, edit the source code (they are found very near the top of file `scat2.hpp`) and recompile (`make clean` followed by `make`).

The practice of summarizing `SCAT2` results as median latitude and median longitude can be misleading if the individual's inferred distribution is patchy or irregular in shape. Displaying a heatmap of all `SCAT2` results for that individual may be preferable, and/or reporting the bounding region containing a given percentage of the estimates.

In our hands, `SCAT2` does a poor job localizing individuals which are hybrids between distant populations or species. We recommend the use of `EBhybrids` (see section 9) to remove putative hybrids. Our preliminary studies suggest that hybrids in the reference data do relatively little harm, but the inferred locations of hybrid individuals are unreliable.

5 Run-length and other search 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 file> <location file> <output filename> L Niter Nthin Nburn
```

where `Niter` is the number of sampled iterations of the MCMC scheme to be performed, `Nthin` is the number of steps taken through the Markov chain between samplings (the "thinning interval") and `Nburn` is the number of burn-in iterations.

For example, to set

`Niter=100, Nthin=10, Nburn=100`

use

```
./SCAT2 <genotype file> <location file> <output filename> L 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; preliminary runs are useful to establish a good run length.

`SCAT2` uses a model of isolation by distance for the relationship of different regional populations' allele frequences: there is expected to be more similarity between populations that are closer together. The behavior of this isolation-by-distance model is controlled by three α parameters: α_0 controls the degree to which a regional population varies from the expectation established by other populations, α_1 controls the scaling for the distances, and α_2 controls how quickly correlations between populations drop to 0 with distance. These parameters are estimated by `SCAT2` during the run. It also estimates a parameter β which governs the variability of marker loci. For a more detailed explanation, see Wasser et al. (2004).

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 α and β to be reliably estimated across runs. For computational reasons, these initial runs are best done without setting an assignment option (`-A` or `-M`). If there are substantial differences between parameter estimates, or in the range of log-likelihoods achieved, in different runs (output in the `_params` file) try increasing the lengths of the runs by increasing either the number of iterations (`Niter`) or the thinning interval (`Nthin`). The program `TRACER` (see section 9) can be helpful here. If traces of the parameters appear to move sluggishly, increase `Nthin`; if there are directional trends visible in the trace over time, increase either `Niter` or `Nthin`; if the estimated Effective Sample Size is below 200, increase `Niter`.

The results in Wasser et al. (2004) were obtained by first applying the method (without performing assignments) using a relatively large number of iterations (eg Niter=100, Nthin=1000, Nburn=100), and using the results in the `_param` file to get estimates of α and β. Next, five independent sets of assignment runs were done with the parameters α and β 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 α and β). Note that when fixing α and β in this way, it is recommended that you first find estimates for α and β from the `_params` file (e. g. 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.

5.1 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. This option can be used to deliberately duplicate a previous run, or to make sure that you do *not* duplicate a previous run. If the seed is not set, a random number seed based on the computer clock will be used; this can lead to multiple runs getting the same seed if they are started at almost exactly the same time, for example by a batch program.

5.2 Fixing model parameters (the `-f` flag)

As noted above, it may be helpful to first estimate the α and β 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 $\alpha_0 = 0.43, \alpha_1 = 5300, \alpha_2 = 0.32, \beta = 2.3$.

You can also modify the aggressiveness of the search across α values using the following option:

`-a sd` set the proposal standard deviation for alpha updates to `sd` (default 0.4). A larger `sd` will correspond to a more wide-ranging search of alpha values.

5.3 Setting error parameters (the `-e` flag)

The assignment analysis machinery allows for the probabilities of an erroneous genotype call at one allele (error term δ) and of amplification of only one allele (error term γ). The defaults are set to the values used in Wasser et al. (2004) of $\delta = 0.05$ and $\gamma = 0$. These values can be changed using `-e delta gamma`.

6 Options changing the mathematics or search strategy

These options change the behavior of the MCMC sampler. Little information is available about their usefulness, but they are documented here for completeness. In the descriptions below, X refers to the vector of rescaled allele frequencies described in the Online Supplement of Wasser et al. 2004. The `SCAT2` algorithm normally proposes new values of X one by one, with a variance of 0.5, but this can be changed using the options below.

`-a sd` set the proposal standard deviation for alpha updates to `sd` (default 0.4). A larger `sd` will correspond to a more wide-ranging search of alpha values.

`-h sd` sets the proposal standard deviation for X updates to `sd` times $\sqrt{1/\alpha_0}$ (default = 0.5). A larger `sd` will correspond to a more wide-ranging search of X values. The higher the value, the further the sampler will attempt to move in one step. Higher values may help in preventing the sampler from becoming stuck (diagnosed by persistent differences between the outcome of different runs on the same data), while lower values may improve acceptance rate (diagnosed by examining the Output_accept file).

`-j` update X for an allele jointly, rather than one at a time. No information is currently available on whether this is helpful.

`-N` includes a "nugget effect" in the covariance matrix. A nugget effect is variation among samples putatively taken at the same point, either because of spatial structure too fine for the sampling scheme to capture, or because of measurement error. No information is currently available on whether this is helpful, but it may be worth considering particularly if your sample locations are rather broadly defined and you think within-location population structure is likely.

`-r` use Langevin update for X. This update attempts to improve sampling performance by using the estimated gradient of the local probability density to choose the direction in which to move. The Wikipedia article "Metropolis-adjusted Langevin algorithm" describes this algorithm. No information is currently available on whether this is helpful, but it may be worth trying if mixing is poor.

`-R` remove all samples from a region when doing assignment of individuals from that region. This is a more aggressive form of cross-validation, but relies on getting enough allele frequency estimates from nearby regions to fill in for the disregarded frequencies for this region, and seems unlikely to work unless the data are quite rich and the number of regions is fairly large. It might be particularly appropriate if the data consist of a large number of regions, each with only a few individuals, so that region-specific empirical allele frequencies rely heavily on the few available samples in that region.

`-w` use smoothing only towards the mean, with no spatial component (ie set $\alpha_2 = 0$). This asserts that the species has no spatial structure, so that all regions other than the one whose frequencies are currently being calculated contribute equally to that calculation. This is unlikely for real biological populations, though could possibly serve as a comparison point.

`-X` "cheats" by initialising location close to true location in assignment runs. This can be used when assigning locations to individuals whose location is already known or suspected. The search will converge more rapidly if it starts from a mostly-correct solution. However, if the search is run for too short a time or mixes poorly, it will return its starting location with excessive confidence. We do not recommend the use of this option to generate publishable results. It could be used diagnostically to test whether the sampler is simply stuck at a poor solution, or is actively moving towards a poor solution.

7 Interpreting the output

7.1 Output to screen during run

When run, the program initially outputs the data it has read from the input file. Since `SCAT2` currently performs very limited error checking of the input data, 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).

7.2 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 the location file, for each locus. It also contains, for comparison, the empirical allele frequencies at that location.

`_probs` Contains estimates of the posterior probability:

$$\log(\Pr(\text{genotype data for individual } i | i \text{ came from location } j))$$

for every i and j. These correspond to assignment to discrete regions, and the 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 for which it had genotype data. The next two columns give the location numbers of the true and assigned locations for that individual (the true location will be `-1` if not known). Subsequent columns give the estimates of the log probabilities for locations $j = 1, 2, \ldots$. In version 2.1 these probabilities were in hexadecimal; they are now in decimal. In general you will want to use the output from this file *only* if you used an assignment option (`-A` or `-M`). The file is still produced even if you don't use an assignment option, but in that case the results are not based on cross-validation, and 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 are 5 columns, corresponding to sampled values of $\alpha_0, \alpha_1, \alpha_2$, β, and the log-likelihood for the corresponding allele frequency estimates. This file works well with the Tracer utility (see section 9) which can display plots of the parameter values over time. This is helpful in diagnosing poor mixing.

`_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 α_1, α_2 (if these are updated), X and μ.

`_corr` Contains estimated and fitted correlations and covariances between each pair of locations. The first two columns give the pair of locations, and the third gives the distance between them in kilometers inferred from their latitude and longitude. The following four columns are empirical and fitted covariance and empirical and fitted correlation. "Empirical" results are estimated from the inferred allele frequences of the two locations. "Fitted" results are estimated using only the distance between the locations and the inferred model parameters (α) which control the rate at which correlation decays with distance. This file may be useful for model checking, specifically spotting pairs of populations that are much more or less correlated than expected based on their geographical distance.

8 Subregions and hybrids

The documentation, code comments, and code for these options do not agree and their correctness in the current version is not known. We do not recommend their use.

This functionality has been superseded by the `EBhybrids` program (see software list), and we strongly recommend use of that program instead. It is documented here only for completeness.

Subregions are broad geographic areas, larger than the regions to which `SCAT2` is assigning individuals, which are considered likely to be more genetically differentiated than their geographic distance would suggest: in African elephants these could represent the forest and savannah species.

`SCAT2` can attempt to identify hybrids either between regions or between subregions. Unless the regions are genetically distinct, hybrid identification will be very difficult; hence the use of subregions. `SCAT2` can also attempt to assign locations to identified hybrids, but this is also problematic. One may instead remove identified hybrids from the analysis.

Hybrid testing is turned on with the `-H` option where the H is followed immediately (no space) by a numeric option. There are three meaningful options:

- -H1 Hybrid analysis with subregions. The program will estimate, for each specimen, the probability that it is solely from each of the subregions, and the probability that it is a hybrid with ancestry from each pair of subregions. (Combinations of more than 2 subregions are not considered.)
- -H2 Normal analysis with subregions. This estimates the subregion membership of each elephant, as in the normal region-based assignment but using subregions instead of regions.
- -H11 Hybrid analysis with regions. The program will estimate hybrid probabilities as in H1, but using regions instead of subregions. This is unlikely to work unless the regions are genetically distinct.

For `-H1` and `-H2` you will need to specify the subregion of each sampling location in the location file, and use the `-Z` option above.

Subregion names must be 0,1,...NSUBREGIONS-1 where NSUBREGIONS is a constant in the file scat2.hpp, defaulting to 6. The program *may* work correctly with fewer than NSUBREGIONS subregions, except that rows and columns corresponding to those subregions will be meaningless; it is likely to crash with more than NSUBREGIONS subregions. To adjust this number, edit scat2.hpp and recompile.

The results of this analysis are written to a file `Output_hybrid` in the output director. This file contains one row per individual.

For `-H2` there is one column per subregion, and the (i, j) entry is

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

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

The maximum entry in a row can be taken as an estimate of the individual's origin. For the hybrid cases, the overall probability that an individual is a hybrid can be taken as the sum of all entries in the row except those indicating both parent from the same region or subregion.

9 Other software useful in conjunction with `SCAT2`

These programs are listed as a convenience to the user; we do not guarantee their availability, compatibility, or performance.

- `STRUCTURE` (`http://pritch.bsd.uchicago.edu/`) – estimate population admixture and assign individuals to populations.
- `EBhybrids` (`https://github.com/stephenslab/EBhybrids`) – postprocess `STRUCTURE` output to infer hybrid probability for each individual. In our experience, hybrid individuals are difficult for `SCAT2` to assign and it may be best to remove them or at least treat their inferred locations with skepticism.
- `TRACER` (`https://beast.community/tracer`) – display behavior of an MCMC run over time. With minimal processing, the `Output_params` file can be read into `TRACER` to visualize the progress of the run, which is useful in diagnosing poor mixing or too-short runs.
- `VORONOI` (`https://github.com/stephens999/voronoi`) – postprocess `SCAT2` estimates to refine continuous assignment on the assumption that the location-unknown samples may be spatially clustered relative to the reference samples.

10 Changes to the software

I presume that the changes listed in the documentation for version 2.1 (September 8, 2004) were from 2.0 to 2.1; this is not entirely clear in the documentation.

10.1 Changes between 2.0 and 2.1

According to the previous maintainer, "A number of small changes were made to the MCMC update scheme. Most of these changes were made to try to improve mixing for general datasets (rather than the specific one we analysed)."

A bug in computation of the likelihood for subregions was fixed.

At some point a change was introduced (probably accidentally) which caused the log probabilities in file `Output_probs` to be printed in hexadecimal; it is not known if this feature was already in 2.0 or was introduced in 2.1.

10.2 Changes between 2.1 and 2.2

From this version on, maintenance is being performed by Mary Kuhner (mkkuhner@uw.edu).

Corrections: The probabilities in file Output_probs were changed to decimal; this may disrupt scripts written for the previous version.

Improvements: Version number was added. The Makefile was revised so that "make" makes the `SCAT2` executable. Function templates were moved to a file `scat2.hpp`. The hard-coded limit on number of regions was removed; the program can now adapt to any reasonable number of regions. The documentation was extensively rewritten.

Increased error checking of input options and data was added. Specifically, each individual is now required to have exactly two haplotypes, and the number of loci must be the same for all haplotypes of all individuals.

Removals: Option `-z` which caused certain individuals to be deleted from the data was removed; it had been added to test a specific hypothesis on a specific data set and was not suitable for distribution.

10.3 Changes between 2.2 and 2.3

Corrections: An initialization bug was corrected. This only disrupted the first burning step, and likely had little impact on correctness; however, runs after this fix will not be numerically identical to those before, as the MCMC search will take a randomly different path.

Internal variable handling was extensively revised to improve maintainability. This should not impact correctness but may cause the MCMC search to take a different path due to rounding differences.

11 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 SK, Shedlock AM, Comstock K, Ostrander EA, Mutayoba B, Stephens M (2004) Assigning African elephant DNA to geographic region of origin: applications to the ivory trade. PNAS 101: 14847-14852.

12 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`)

.