Documentation for *structure* software: Version 2.2

Jonathan K. Pritchard[a]
Xiaoquan Wen[a]
Daniel Falush[b] [1] [2] [3]

[a]Department of Human Genetics
University of Chicago

[b]Department of Statistics
University of Oxford

Software from
`http://pritch.bsd.uchicago.edu/software`

April 3, 2007

[1]Our other colleagues in the *structure* project are Peter Donnelly and Matthew Stephens.

[2]The first version of this program was developed while the authors (JP, MS, PD) were in the Department of Statistics, University of Oxford.

[3]Comments and queries can be emailed to us at structure_help@bsd.uchicago.edu. In view of the considerable volume of email that we receive, please check this document carefully first.

Contents

1 Introduction

The program *structure* implements a model-based clustering method for inferring population structure using genotype data consisting of unlinked markers. The method was introduced in a paper by Pritchard, Stephens and Donnelly (2000a) and extended in sequels by Falush, Stephens and Pritchard (2003a, 2007). Applications of our method include demonstrating the presence of population structure, identifying distinct genetic populations, assigning individuals to populations, and identifying migrants and admixed individuals.

Briefly, we assume a model in which there are K populations (where K may be unknown), each of which is characterized by a set of allele frequencies at each locus. Individuals in the sample are assigned (probabilistically) to populations, or jointly to two or more populations if their genotypes indicate that they are admixed. It is assumed that within populations, the loci are at Hardy-Weinberg equilibrium, and linkage equilibrium. Loosely speaking, individuals are assigned to populations in such a way as to achieve this.

Our model does not assume a particular mutation process, and it can be applied to most of the commonly used genetic markers including microsatellites, SNPs and RFLPs. The model assumes that markers are not in linkage disequilibrium (LD) *within* subpopulations, so we can't handle markers that are extremely close together. Starting with version 2.0, we can now deal with weakly linked markers (Section 1.2).

While the computational approaches implemented here are fairly powerful, some care is needed in running the program in order to ensure sensible answers. For example, it is not possible to determine suitable run-lengths theoretically, and this requires some experimentation on the part of the user. This document describes the use and interpretation of the software and supplements the published papers, which provide more formal descriptions and evaluations of the methods.

1.1 Overview

The software package *structure* consists of several parts. The computational part of the program was written in C, and executables are distributed for various platforms (currently Windows, Linux, Sun). The C executable reads a data file supplied by the user. There is also a Java front end that provides various nifty plots and other helpful features for the user; however, you can also invoke *structure* from the command line instead of using the front end.

This document includes information about how to format the data file, how to choose appropriate models, and how to interpret the results. It also has details on using the two interfaces (command line and front end) and a summary of the various user-defined parameters.

1.2 What's new in Version 2.2?

The 2.2 release (April 2007) allows the use of dominant markers such as AFLPs (commonly used in non-model organisms) and can accommodate null alleles and other data ambiguities (Falush et al., 2007). There are a few bug fixes including fixing a problem that occurred occasionally on Windows machines, causing *structure* to crash at seemingly arbitrary values of K. We have tidied up some aspects of the front end and C kernel. We now print out an average pairwise nucleotide distance within and between clusters (Falush et al., 2003b), instead of the Kullback-Leibler distance which was less interpretable and less numerically stable. The output format for the linkage model has been changed. We hope to release further improvements in the coming months.

		loc_a	loc_b	loc_c	loc_d	loc_e
George	1	-9	145	66	0	92
George	1	-9	-9	64	0	94
Paula	1	106	142	68	1	92
Paula	1	106	148	64	0	94
Matthew	2	110	145	-9	0	92
Matthew	2	110	148	66	1	-9
Bob	2	108	142	64	1	94
Bob	2	-9	142	-9	0	94
Anja	1	112	142	-9	1	-9
Anja	1	114	142	66	1	94
Peter	1	-9	145	66	0	-9
Peter	1	110	145	-9	1	-9
Carsten	2	108	145	62	0	-9
Carsten	2	110	145	64	1	92

Table 1: Sample data file. Here MARKERNAMES=1, LABEL=1, POPDATA=1, NUMINDS=7, NUMLOCI=5, and MISSING=-9. Also, POPFLAG=0, PHENOTYPE=0, EXTRACOLS=0. The second column shows the geographic sampling location of individuals. We can also store the data with one row per individual, in which case the first row would read "George 1 -9 -9 145 -9 66 64 0 0 92 94".

1.3 Registering.

We strongly encourage users to complete the brief registration form at the website (`http://-pritch.bsd.uchicago.edu/software`). This will help us to assess where to spend future effort in maintaining or developing the software. Estimates of the number of users will also be helpful when we apply for renewed funding for this and related projects.

2 Format for the data file

The format for the genotype data is shown in Table 2 (and Table 1 shows an example). Essentially, the entire data set is arranged as a matrix in a single file, in which the data for individuals are in rows, and the loci are in columns. The user can make several choices about format, and most of these data (apart from the genotypes!) are optional.

For a diploid organism, data for each individual can be stored either as 2 consecutive rows, where each locus is in one column, or in one row, where each locus is in two consecutive columns. Unless you plan to use the linkage model (see below) the order of the alleles for a single individual does not matter. The pre-genotype data columns (see below) are recorded twice for each individual. (More generally, for n-ploid organisms, data for each individual are stored in n consecutive rows unless the ONEROWPERIND option is used.)

2.1 Components of the data file:

The elements of the input file are as listed below. *If present, they must be in the following order, however most are optional (as indicated) and may be deleted completely.* The user specifies which

data are present, either in the front end, or (when running *structure* from the command line), in a separate file, *mainparams*. At the same time, the user also specifies the number of individuals and the number of loci.

2.2 Rows

1. **Marker Names** (Optional; string) The first row in the file can contain a list of identifiers for each of the markers in the data set. This row contains L strings of integers or characters, where L is the number of loci.

2. **Recessive Alleles** (Data with dominant markers only; integer) Data sets of SNPs or microsatellites would generally not include this line. However if the option RECESSIVEALLELES is set to 1, then the program requires this row to indicate which allele (if any) is recessive at each marker. See Section 5.1 for more information. The option is used for data such as AFLPs and for polyploids where genotypes may be ambiguous.

3. **Inter-Marker Distances** (Optional; real) the next row in the file is a set of inter-marker distances, for use with linked loci. These should be genetic distances (e.g., centiMorgans), or some proxy for this based, for example, on physical distances. The actual units of distance do not matter too much, provided that the marker distances are (roughly) proportional to recombination rate. The front end estimates an appropriate scaling from the data, but users of the command line version must set LOG10RMIN, LOG10RMAX and LOG10RSTART in the file extraparams.

   The markers must be in map order within linkage groups. When consecutive markers are from different linkage groups (e.g., different chromosomes), this should be indicated by the value -1. The first marker is also assigned the value -1. All other distances are non-negative. This row contains L real numbers.

4. **Phase Information** (Optional; diploid data only; real number in the range [0,1]). *This is for use with the linkage model only.* This is a single row of L probabilities that appears after the genotype data for each individual. If phase is known completely, or no phase information is available, these rows are unnecessary. They may be useful when there is partial phase information from family data or when haploid X chromosome data from males and diploid autosomal data are input together. There are two alternative representations for the phase information: (1) the two rows of data for an individual are assumed to correspond to the paternal and maternal contributions, respectively. The phase line indicates the probability that the ordering is correct at the current marker (set MARKOVPHASE=0); (2) the phase line indicates the probability that the phase of one allele relative to the previous allele is correct (set MARKOVPHASE=1). The first entry should be filled in with 0.5 to fill out the line to L entries. For example the following data input would represent the information from an male with 5 unphased autosomal microsatellite loci followed by three X chromosome loci, using the maternal/paternal phase model:

102	156	165	101	143	105	104	101
100	148	163	101	143	-9	-9	-9
0.5	0.5	0.5	0.5	0.5	1.0	1.0	1.0

where -9 indicates "missing data", here missing due to the absence of a second X chromosome, the 0.5 indicates that the autosomal loci are unphased, and the 1.0s indicate that the X chromosome loci are have been maternally inherited with probability 1.0, and hence are phased. The same information can be represented with the markovphase model. In this case the input file would read:

102	156	165	101	143	105	104	101
100	148	163	101	143	-9	-9	-9
0.5	0.5	0.5	0.5	0.5	0.5	1.0	1.0

Here, the two 1.0s indicate that the first and second, and second and third X chromosome loci are perfectly in phase with each other. Note that the site by site output under these two models will be different. In the first case, structure would output the assignment probabilities for maternal and paternal chromosomes. In the second case, it would output the probabilities for each allele listed in the input file.

5. **Individual/Genotype data** (Required) Data for each sampled individual are arranged into one or more rows as described below.

2.3 Individual/genotype data

Each row of individual data contains the following elements. These form columns in the data file.

1. **Label** (Optional; string) A string of integers or characters used to designate each individual in the sample.

2. **PopData** (Optional; integer) An integer designating a user-defined population from which the individual was obtained (for instance these might designate the geographic sampling locations of individuals).

3. **PopFlag** (Optional; 0 or 1) A Boolean flag which indicates whether to use the PopData when using learning samples (see USEPOPINFO, below). *(Note: A Boolean variable (flag) is a variable which takes the values TRUE or FALSE, which are designated here by the integers 1 (use PopData) and 0 (don't use PopData), respectively.)*

4. **Phenotype** (Optional; integer) An integer designating the value of a phenotype of interest, for each individual. ($\phi^{(i)}$ in table.) (The phenotype information is not actually used in this program. It is here to permit a smooth interface with the program STRAT which is used for association mapping.)

5. **Extra Columns** (Optional; string) It may be convenient for the user to include additional data in the input file which are ignored by the program. These go here, and may be strings of integers or characters.

6. **Genotype Data** (Required; integer) Each allele at a given locus should be coded by a unique integer (eg microsatellite repeat score).

2.4 Missing genotype data

Missing data should be indicated by a number that doesn't occur elsewhere in the data (often -9 by convention). This number can also be used where there is a mixture of haploid and diploid data (eg X and autosomal loci in males). The missing-data value is set along with the other parameters describing the characteristics of the data set.

2.5 Formatting errors.

We have implemented reasonably careful error checking to make sure that the data set is in the correct format, and the program will attempt to provide some indication about the nature of any problems that exist. The front end requires returns at the ends of each row, and does not allow returns within rows; the command-line version of *structure* treats returns in the same way as spaces or tabs.

One problem that can arise is that editing programs used to assemble the data prior to importing them into *structure* can introduce hidden formatting characters, often at the ends of lines, or at the end of the file. The front end can remove many of these automatically, but this type of problem may be responsible for errors when the data file seems to be in the right format. If you are importing data to a UNIX system, the dos2unix function can be helpful for cleaning these up.

3 Modelling decisions for the user

3.1 How long to run the program

The program is started from a random configuration, and from there takes a series of steps through the parameter space, each of which depends (only) on the parameter values at the previous step. This procedure induces correlations between the state of the Markov chain at different points during the run. The hope is that by running the simulation for long enough, the correlations will be negligible.

There are two issues to worry about: (1) burnin length: how long to run the simulation before collecting data to minimize the effect of the starting configuration, and (2) how long to run the simulation after the burnin to get accurate parameter estimates.

To choose an appropriate burnin length, it is really helpful to look at the values of summary statistics that are printed out by the program (eg α, F, the divergence distances among populations $D_{i,j}$, and the likelihood) to see whether they appear to have converged. Typically a burnin of 10,000—100,000 is more than adequate.

To choose an appropriate run length, you will need to do several runs at each K, possibly of different lengths, and see whether you get consistent answers. Typically, you can get good estimates of the parameter values (P and Q) with runs of 10,000–100,000 steps, but accurate estimation of $\Pr(X|K)$ may require longer runs. In practice your run length may be determined by your computer speed and patience as much as anything else. If you are dealing with extremely large data sets and are frustrated with the run times, you might try trimming both the length of the runs, and the number of markers/individuals, at least for exploratory analyses.

The front end provides time series plots of several key parameters. You should look to see whether these appear to reach equilibrium before the end of the burnin phase. If the values are still increasing or decreasing at the end of the burnin phase, you need to increase the burnin length.

Label	*Pop*	*Flag*	*Phen*	*ExtraCols*	*Loc 1*	*Loc 2*	*Loc 3*	....	*Loc L*
					M_1	M_2	M_3	....	M_L
					r_1	r_2	r_3	....	r_L
					-1	$D_{1,2}$	$D_{2,3}$	....	$D_{L-1,L}$
$ID^{(1)}$	$g^{(1)}$	$f^{(1)}$	$\phi^{(1)}$	$y_1^{(1)}, ..., y_n^{(1)}$	$x_1^{(1,1)}$	$x_2^{(1,1)}$	$x_3^{(1,1)}$	....	$x_L^{(1,1)}$
$ID^{(1)}$	$g^{(1)}$	$f^{(1)}$	$\phi^{(1)}$	$y_1^{(1)}, ..., y_n^{(1)}$	$x_1^{(1,2)}$	$x_2^{(1,2)}$	$x_3^{(1,2)}$	....	$x_L^{(1,2)}$
					$p_1^{(1)}$	$p_2^{(1)}$	$p_3^{(1)}$	....	$p_L^{(1)}$
$ID^{(2)}$	$g^{(2)}$	$f^{(2)}$	$\phi^{(2)}$	$y_1^{(2)}, ..., y_n^{(2)}$	$x_1^{(2,1)}$	$x_2^{(2,1)}$	$x_3^{(2,1)}$	....	$x_L^{(2,1)}$
$ID^{(2)}$	$g^{(2)}$	$f^{(2)}$	$\phi^{(2)}$	$y_1^{(2)}, ..., y_n^{(2)}$	$x_1^{(2,2)}$	$x_2^{(2,2)}$	$x_3^{(2,2)}$	....	$x_L^{(2,2)}$
					$p_1^{(2)}$	$p_2^{(2)}$	$p_3^{(2)}$	....	$p_L^{(2)}$
....									
$ID^{(i)}$	$g^{(i)}$	$f^{(i)}$	$\phi^{(i)}$	$y_1^{(i)}, ..., y_n^{(i)}$	$x_1^{(i,1)}$	$x_2^{(i,1)}$	$x_3^{(i,1)}$	....	$x_L^{(i,1)}$
$ID^{(i)}$	$g^{(i)}$	$f^{(i)}$	$\phi^{(i)}$	$y_1^{(i)}, ..., y_n^{(i)}$	$x_1^{(i,2)}$	$x_2^{(i,2)}$	$x_3^{(i,2)}$	....	$x_L^{(i,2)}$
					$p_1^{(3)}$	$p_2^{(3)}$	$p_3^{(3)}$	....	$p_L^{(3)}$
....									
$ID^{(N)}$	$g^{(N)}$	$f^{(N)}$	$\phi^{(N)}$	$y_1^{(N)}, ..., y_n^{(N)}$	$x_1^{(N,1)}$	$x_2^{(N,1)}$	$x_3^{(N,1)}$	....	$x_L^{(N,1)}$
$ID^{(N)}$	$g^{(N)}$	$f^{(N)}$	$\phi^{(N)}$	$y_1^{(N)}, ..., y_n^{(N)}$	$x_1^{(N,2)}$	$x_2^{(N,2)}$	$x_3^{(N,2)}$	....	$x_L^{(N,2)}$
					$p_1^{(L)}$	$p_2^{(L)}$	$p_3^{(L)}$	....	$p_L^{(1)}$

Table 2: Format of the data file, in two-row format. Most of these components are optional (see text for details). M_l is an identifier for marker l. r_l indicates which allele, if any, is recessive at each marker (dominant genotype data only). $D_{i,i+1}$ is the distance between markers i and $i+1$. $ID^{(i)}$ is the label for individual i, $g^{(i)}$ is the geographic origin of individual i (PopData); $f^{(i)}$ is a flag used to incorporate learning samples (PopFlag); $\phi^{(i)}$ can store a phenotype for individual i; $y_1^{(i)}, ..., y_n^{(i)}$ are for storing extra data (ignored by the program); $(x_l^{i,1}, x_l^{i,2})$ stores the genotype of individual i at locus l. $p_i^{(l)}$ is the phase information for marker l in individual i.

If the estimate of α varies greatly throughout the run (i.e., not just during the burnin), you may get more accurate estimates of $\Pr(X|K)$ by increasing ALPHAPROPSD, which improves mixing in that situation. (See a related issue in section 4).

3.2 Ancestry Models

There are four main models for the ancestry of individuals: **(1)** no admixture model (individuals are discretely from one population or another); **(2)** the admixture model (each individual draws some fraction of his/her genome from each of the K populations; **(3)** the linkage model (like the admixture model, but linked loci are more likely to come from the same population); **(4)** the prior information model (allows the user to assign some or all individuals to pre-defined populations). See Pritchard et al. (2000a) for more on models 1,2, and 4 and Falush et al. (2003a) for model 3.

1. No admixture model. Each individual comes purely from one of the K populations. The output reports the *posterior probability* that individual i is from population k. The prior probability for each population is $1/K$. This model is appropriate for studying fully discrete populations and is often more powerful than the admixture model at detecting subtle structure.

2. Admixture model. Individuals may have mixed ancestry. This is modelled by saying that individual i has inherited some fraction of his/her genome from ancestors in population k. The output records the *posterior mean estimates* of these proportions. Conditional on the ancestry vector, $q^{(i)}$, the origin of each allele is independent.

We recommend this model as a starting point for most analyses. It is a reasonably flexible model for dealing with many of the complexities of real populations. Admixture is a common feature of real data, and you probably won't find it if you use the no-admixture model. The admixture model can also deal with hybrid zones in a natural way.

3. Linkage model. This is essentially a generalization of the admixture model to deal with "admixture linkage disequilibrium"–i.e., the correlations that arise between linked markers in recently admixed populations. Falush et al. (2003a) describes the model, and computations in more detail.

The basic model is that, t generations in the past, there was an admixture event that mixed the K populations. If you consider an individual chromosome, it is composed of a series of "chunks" that are inherited as discrete units from ancestors at the time of the admixture. Admixture LD arises because linked alleles are often on the same chunk, and therefore come from the same ancestral population.

The sizes of the chunks are assumed to be independent exponential random variables with mean length $1/t$ (in Morgans). In practice we estimate a "recombination rate" r from the data that corresponds to the rate of switching from the present chunk to a new chunk.[1] Each chunk in individual i is derived *independently* from population k with probability $q_k^{(i)}$, where $q_k^{(i)}$ is the proportion of that individual's ancestry from population k.

Overall, the new model retains the main elements of the admixture model, but all the alleles that are on a single chunk have to come from the same population. The new MCMC algorithm integrates over the possible chunk sizes and break points. It reports the overall ancestry for each individual, taking account of the linkage, and can also report the probability of origin of each bit of chromosome, if desired by the user.

This new model performs better than the original admixture model when using linked loci to study admixed populations. It achieves more accurate estimates of the ancestry vector, and can extract more information from the data. It should be useful for admixture mapping.

Clearly, this model is a big simplification of the complex realities of most real admixed populations. However, the major effect of admixture is to create long-range correlation among linked markers, and so our aim here is to encapsulate that feature within a fairly simple model.

The computations are a bit slower than for the admixture model, especially with large K and unphased data. Nonetheless, they are practical for thousands of sites and individuals and multiple

[1]Because of the way that this is parameterized, the map distances in the input file can be in arbitrary units–e.g., genetic distances, or physical distances (under the assumption that these are roughly proportional to genetic distances). Then the estimated value of r represents the rate of switching from one chunks to the next, per unit of whatever distance was assumed in the input file. E.g., if an admixture event took place ten generations ago, then r should be estimated as 0.1 when the map distances are measured in cM (this is $10 * 0.01$, where 0.01 is the probability of recombination per centiMorgan), or as $10^{-4} = 10 * 10^{-5}$ when the map distances are measured in KB (assuming a constant crossing-over rate of 1cM/MB). The prior for r is log-uniform. The front end tries to make some guesses about sensible upper and lower bounds for r, but the user should adjust these to match the biology of the situation.

populations. The model can only be used if there is information about the relative positions of the markers (usually a genetic map).

4. Using prior population information. The default mode for *structure* uses only genetic information to learn about population structure. However, there is often other information that might be relevant to the clustering (e.g., physical characteristics of sampled individuals or geographic sampling location).

At present, *structure* can use this information in two ways. First, the user might find that the pre-defined groups (eg sampling locations) correspond almost exactly to *structure* clusters, except for a handful of individuals who seem to be misclassified. Pritchard et al. (2000a) developed a formal Bayesian test for evaluating whether any individuals in the sample are immigrants to their supposed populations, or have recent immigrant ancestors.

A second way that prior information information may be introduced through the use of learning samples: ie., some individuals are of known origin, and are used to classify individuals of unknown origin. For example Beaumont et al. (2001) wanted to learn about the ancestry of Scottish wildcats (many of which are hybridized with feral domestic cats). They had genetic data from a bunch of pet house cats which were defined as being in one population, and they inferred Q for the wildcats (with $K = 2$). Use of this sort of prior information will normally improve the accuracy of the inference.

Note that in both settings the user should also run the program without population information to ensure that the pre-defined populations are in rough agreement with the genetic information.

To use these options you need to set USEPOPINFO to 1, and choose a value of MIGRPRIOR (which is ν in Pritchard et al. (2000a)). You might choose something in the range 0.001 to 0.1 for ν. Even when using learning samples, it may be sensible to allow for some misclassification by setting MIGRPRIOR larger than 0. Note that **this model assumes that the predefined populations are** *usually* **correct**. It takes quite strong data to overcome the prior against misclassification.

The pre-defined population for each individual is set in the input data file (see PopData) and should be an integer between 1 and MAXPOPS, inclusive. If PopData for any individual is outside this range, their q will be updated in the normal way (ie without prior population information, according to the model that would be used if USEPOPINFO was turned off.[2]). Learning samples are implemented through the use of the PopFlag column in the data file. The pre-defined population is used for those individuals for whom PopFlag=1, and it is ignored for individuals for whom PopFlag=0. If there is no PopFlag column in the data file, then when USEPOPINFO is turned on, PopFlag is set to 1 for all individuals.

Ordinarily, *structure* analyses update the allele frequency estimates using all available individuals. However there are some settings where you might want to estimate ancestry for some individuals, without those individuals affecting the allele frequency estimates. For example you may have a standard collection of learning samples, and then periodically you want to estimate ancestry for new batches of genotyped individuals. Using default options, the ancestry estimates for individuals would depend (somewhat) on which batch they are in. By using UPDATEPFROMPOPFLAGONLY you can ensure that the allele frequency estimates depend only on samples for which PopFlag=1. In a different setting, Murgia et al. (2006) wanted to determine the origin of a set of clonal dog tumours. The tumours were so closely related that using default settings they formed a cluster of their own.

[2]If the admixture model is used to estimate q for those individuals without prior population information, α is updated on the basis of those individuals only. If there are very few such individuals, you may need to fix α at a sensible value.

By using UPDATEPFROMPOPFLAGONLY, they forced the tumours to group with other canid clusters.

3.3 Allele frequency models

There are two basic models for the allele frequencies. One model assumes that the allele frequencies in each population are independent draws from a distribution that is specified by a parameter called λ. That is the original model that used in Pritchard et al. (2000a). Usually we set $\lambda = 1$; this is the default setting.

Falush et al. (2003a) implemented a model with correlated allele frequencies. This says that frequencies in the different populations are likely to be similar (probably due to migration or shared ancestry). Further details are given below.[3]

The independent model works well for many data sets. Roughly speaking, this prior says that we expect allele frequencies in different populations to be reasonably different from each other. The correlated frequencies model says that they may actually be quite similar. This often improves clustering for closely related populations, but may increase the risk of over-estimating K (see below). If one population is quite divergent from the others, the correlated model can sometimes achieve better inference if that population is removed.

Estimating λ: Fixing $\lambda = 1$ is a good idea for most data, but in some situations–e.g., SNP data where most minor alleles are rare–smaller values may work better. For this reason, you can get the program to estimate λ for your data. You may want to do this once, perhaps for $K = 1$, and then fix λ at the estimated value thereafter, because there seem to be some problems with non-identifiability when trying to estimate too many of the hyperparameters (λ, α, F) at the same time.

Correlations model: As described by Falush et al. (2003a) the correlated frequencies model uses a (multidimensional) vector, P_A, which records the allele frequencies in a hypothetical "ancestral" population. It is assumed that the K populations represented in our sample have each undergone independent drift away from these ancestral frequencies, at a rate that is parameterized by $F_1, F_2, F_3, ..., F_K$, respectively. The estimated F_k values should be numerically similar to F_{ST} values, apart from differences that stem from the slightly different model, and differences in estimation. Also, it is difficult to estimate F_k accurately for data with lots of admixture.

P_A is assumed to have a Dirichlet prior of the same form as that used above for the population frequencies:

$$p_{Al\cdot} \sim \mathcal{D}(\lambda_1, \lambda_2, \ldots, \lambda_{J_l}), \tag{1}$$

independently for each l. Then the prior for the frequencies in population k is

$$p_{kl\cdot} \sim \mathcal{D}(P_{Al1}\frac{1-F_k}{F_k}, P_{Al2}\frac{1-F_k}{F_k}, \ldots, P_{AlJ_l}\frac{1-F_k}{F_k}), \tag{2}$$

independently for each k and l. In this model, the Fs have a close relationship to the standard measure of genetic distance, F_{ST}. In the standard parametrization of F_{ST}, the expected frequency in each population is given by overall mean frequency, and the variance in frequency across subpopulations of an allele at overall frequency p is $p(1-p)F_{ST}$. The model here is much the same, except that we generalize the model slightly by allowing each population to drift away from the ancestral

[3]Note that Pritchard et al. (2000a) also outlined a model of correlated allele frequencies; this was last available in Version 1.x

population at a different rate (F_k), as might be expected if populations have different sizes. We also try to estimate "ancestral frequencies", rather than using the mean frequencies.

We have placed independent priors on the F_k, proportional to a gamma distribution with means of 0.01 and standard deviation 0.05 (but with $\Pr[F_k \geq 1] = 0$). The parameters of the gamma prior can be modified by the user. Some experimentation suggests that the prior mean of 0.01, which corresponds to very low levels of subdivision, often leads to good performance for data that are difficult for the independent frequencies model. In other problems, where the differences among populations are more marked, it seems that the data usually overwhelm this prior on F_k.

Estimating admixture proportions when most individuals are admixed. Estimating admixture proportions can be particularly challenging if there are very few representatives of the parental populations. There is an example of this for simulated data in Pritchard et al. (2000b). The data were supposed to approximate a sample from an African American population in which most individuals had some degree of European ancestry. For those data, the estimated ancestry proportions were highly correlated with the true (simulated) values, but the actual proportions of ancestry were biased. That example is also representative of our more recent experience with real data.

This occurs because in the absence of any non-admixed individuals, there may be some non-identifiability where it is possible to push the allele frequencies further apart, and squeeze the admixture proportions together (or vice-versa), and obtain much the same degree of model fit. Using POPALPHAS=1 (separate α for each population) can help a bit when there is strongly asymmetric admixture but does not solve the basic problem. Therefore, the admixture estimates in these situations should be treated with caution.

4 Estimation of K (the number of populations)

In our paper describing this program, we pointed out that this issue should be treated with care for two reasons: (1) it is computationally difficult to obtain accurate estimates of $\Pr(X|K)$, and our method merely provides an *ad hoc* approximation, and (2) the biological interpretation of K may not be straightforward.

In our experience we find that the real difficulty lies with the second issue. Our procedure for estimating K generally works well in data sets with a small number of discrete populations. However, many real-world data sets do not conform precisely to the *structure* model (e.g., due to isolation by distance or inbreeding). In those cases there may not be a natural answer to what is the "correct" value of K.

Perhaps for this kind of reason, it is not infrequent that in real data the value of our model choice criterion continues to increase with increasing K. Then it usually makes sense to focus on values of K that capture most of the structure in the data and that seem biologically sensible.

4.1 Steps in estimating K

1. (Command-line version) Set COMPUTEPROBS and INFERALPHA to 1 in the file *extraparams*. (Front End version) Make sure that α is allowed to vary.

2. Run the MCMC scheme for different values of MAXPOPS (K). At the end it will output a line "Estimated Ln Prob of Data". This is the estimate of $\ln \Pr(X|K)$. You should run several independent runs for each K, in order to verify that the estimates are consistent across

runs. If the variability across runs for a given K is substantial compared to the variability of estimates obtained for different K, you may need to use longer runs or a longer burnin period. If $\ln \Pr(X|K)$ appears to be bimodal or multimodal, the MCMC scheme may be finding different solutions. You can check for this by comparing the Q for different runs at a single K. (cf Data Set 2A in Pritchard et al. (2000a), and see the section on Multimodality, below).

3. Compute posterior probabilities of K. For example, for Data Set 2A in the paper (where K was 2), we got

K	$\ln \Pr(X\|K)$
1	-4356
2	-3983
3	-3982
4	-3983
5	-4006

We can start by assuming a uniform prior on $K = \{1, ..., 5\}$. Then from Bayes' Rule, $\Pr(K = 2)$ is given by

$$\frac{e^{-3983}}{e^{-4356} + e^{-3983} + e^{-3982} + e^{-3983} + e^{-4006}} \tag{3}$$

It's easier to compute this if we simplify the expression to

$$\frac{e^{-1}}{e^{-374} + e^{-1} + e^{0} + e^{-1} + e^{-24}} = 0.21 \tag{4}$$

4.2 Mild departures from the model can lead to overestimating K

When there is real population structure, this leads to LD among unlinked loci and departures from Hardy-Weinberg proportions. Roughly speaking, this is the signal that is used by the *structure* algorithm. But some departures from the model can also lead to Hardy-Weinberg or linkage disequilibrium. These include inbreeding, and genotyping errors such as occasional, undetected, null alleles. Even in the absence of population structure, these types of factors can lead to a weak statistical signal for $K > 1$.

Beginning in Version 2, we have suggested that the correlated allele frequency model should be used as a default because it often achieves better performance on difficult problems, but the user should be aware that this may make it easier to overestimate K in such settings than under the independent frequencies model Falush et al. (2003a).

The next subsection discusses how to decide whether inferred structure is real.

4.3 Informal pointers for choosing K; is the structure real?

There are a couple of informal pointers which might be helpful in selecting K. The first is that it's often the situation that $\Pr(K)$ is very small for K less than the appropiate value (effectively zero), and then more-or-less plateaus for larger K, as in the example of Data Set 2A shown above. In this sort of situation where several values of K give similar estimates of $\log \Pr(X|K)$, it seems that the smallest of these is often "correct".

It is a bit difficult to provide a firm rule for what we mean by a "more-or-less plateaus". For small data sets, this might mean that the values of log $\Pr(X|K)$ are within 5-10, but Daniel Falush writes that "in very big datasets, the difference between $K = 3$ and $K = 4$ may be 50, but if the difference between $K = 3$ and $K = 2$ is 5,000, then I would definitely choose $K = 3$." Readers who want to use a more formal criterion that takes this into account may be interested in the method of Evanno et al. (2005).

We think that a sensible way to think about this is in terms of model choice. That is, we may not always be able to know the TRUE value of K, but we should aim for the smallest value of K that captures the major structure in the data.

The second pointer is that if there really are separate populations, there is typically a lot of information about the value of α, and once the Markov chain converges, α will normally settle down to be relatively constant (often with a range of perhaps 0.2 or less). However, if there isn't any real structure, α will usually vary greatly during the course of the run.

A corrollary of this is that when there is no population structure, you will typically see that the proportion of the sample assigned to each population is roughly symmetric ($\sim 1/K$ in each population), and most individuals will be fairly admixed. If some individuals are strongly assigned to one population or another, and if the proportions assigned to each group are asymmetric, then this is a strong indication that you have real population structure.

Suppose that you have a situation with two clear populations, but you are trying to decide whether one of these is further subdivided (ie, the value of $\Pr(X|K = 3)$ is similar to, or perhaps a little larger than $P(X|K = 2)$). Then one thing you could try is to run *structure* using only the individuals in the population that you suspect *might* be subdivided, and see whether there is a strong signal as described above.

In summary, you should be skeptical about population structure inferred on the basis of small differences in $\Pr(K)$ if (1) there is no clear biological interpretation for the assignments, and (2) the assignments are roughly symmetric to all populations and no individuals are strongly assigned.

4.4 Isolation by distance data

Isolation by distance refers to the idea that individuals may be spatially distributed across some region, with local dispersal. In this situation, allele frequencies vary gradually across the region. The underlying *structure* model is not well suited to data from this kind of scenario. When this occurs, the inferred value of K, and the corresponding allele frequencies in each group can be rather arbitrary. Depending on the sampling scheme, most individuals may have mixed membership in multiple groups. That is, the algorithm will attempt to model the allele frequencies across the region using weighted averages of K distinct components. In such situations, interpreting the results may be challenging.

4.5 When F_{ST} is significant, but *structure* finds no structure

We occasionally get the following sort of question: "I have genotype data for individuals sampled from n locations. Tests of allele frequency differences indicate small but significant F_{ST} between at least some locations. However *structure* does not find any differences. How do I interpret these results?"

When the predefined populations correspond closely to genetic populations, testing for frequency differences between predefined groups can be more powerful than applying *structure*. This is because the basic *structure* models aim to solve a much harder statistical problem, i.e., identifying population

clusters without being told the likely subgroups in advance. For this reason there is a part of parameter space where there is not quite enough data for *structure* to get the "right" answer, even though a test of F_{ST} using the predefined labels detects population differentiation.

5 Missing data, null alleles... and other problems

The program ignores missing genotype data when updating Q and P. This approach is correct when the probability of having missing data at a particular locus is independent of what allele the individual has there. While estimates of Q for individuals with missing data are less accurate, there is no particular reason to exclude such individuals from the analysis, unless they have very little data at all.

A serious problem arises when data are missing in a systematic manner, as with null alleles. These do not fit the assumed model, and can lead to apparent departures from Hardy-Weinberg even without population structure. One would not expect the assumed model to be robust to this sort of violation. However the dominant markers model (below) can be used if null alleles might be an important problem.

Having multiple family members in the sample also violates the model assumptions. This can sometimes lead to overestimation of K, especially for the correlated frequencies model (Falush et al., 2003a), but there is little effect on the assignment of individuals to populations for fixed K.

5.1 Dominant markers, null alleles, and polyploid genotypes

For some types of genetic markers, such as AFLPs, it is not possible to distinguish all the genotypes. Other types of markers may result in ambigous genotypes if some fraction of the alleles are 'null' because the PCR product fails to amplify due to nearby sequence variation. Starting with Version 2.2 we implement a model that deals with the genotypic ambiguity associated with dominant markers.

In brief, we assume that at any particular locus there may be a single allele that is recessive to all other alleles (eg A), while all other markers are codominant. Hence AB and BB would appear in the raw genotype data as a "phenotype" B, AC and CC would be recorded as C, while BC would be recorded as BC. When there is ambiguity, the model sums over the possible genotypes. Full details are given in Falush et al. (2007).

In order to perform these computations the algorithm must be told which allele (if any) is recessive at each locus. This is done by setting RECESSIVEALLELES=1, and including a single row of L integers at the top of the input file, between the (optional) lines for marker names and map distances, that indicate the recessive allele at each of the L loci in the data set. If at a given locus all the markers are codominant then the recessive value at that locus must be set to the MISSING data value. Conversely if the recessive allele is never observed in homozygous state but you think it might be present (e.g. there might be null alleles) then set the recessive value to an allele that is not observed at that locus (but not MISSING!).

Coding the genotype data: If the phenotype is unambiguous, then it is coded in the *structure* input file as it is. If it is ambiguous then it is coded as homozygous for the dominant allele(s). For example, phenotype A is coded AA, B is coded BB, BC is coded BC, etc. If the marker is haploid in an otherwise diploid individual (eg for the X chromosome in a male), then the second allele is coded as MISSING as before. The genotypes AB, AC, etc are illegal in the input file when A is recessive.

When RECESSIVEALLELES is used to deal with null alleles, genotypes that appear to be homozygote null should be entered as homozygotes for the recessive allele and not as missing data. In practice it may be uncertain whether a failed genotype is really due to homozygous null alleles. *structure* should be robust to these being coded as missing unless null alleles are at high frequency at a locus.

In polyploids (PLOIDY>2) the situation is more complicated because there may be genotypic ambiguity even for codominant markers. It is often difficult to call the exact genotypes in heterozygotes. For example the phenotype AB in a triploid might be AAB or ABB. If *structure* is run with RECESSIVEALLELES=0 then it is assumed that there is no ambiguity.

For polyploids, when RECESSIVEALLELES=1, *structure* allows the data to consist of a mixture of loci for which there is, and isn't genotypic ambiguity. If some loci are not ambiguous then set the code NOTAMBIGUOUS to an integer that does not match any of the alleles in the data, and that does not equal MISSING. Then in the recessive alleles line at the top of the input file put the NOTAMBIGUOUS code for the unambiguous loci. If instead, at a particular locus the alleles are all codominant, but there is ambiguity about the number of each (eg for microsatellites in a tetraploid) then set the recessive allele code to MISSING. Finally, if there is a recessive allele, and there is also ambiguity about the number of each allele, then set the recessive allele code to indicate which allele is recessive.

Estimation of $\Pr(K)$: When RECESSIVEALLELES is used for diploids, the likelihood at each step of the Markov chain is calculated by summing over the possible genotypes. For ease of coding, when either PLOIDY>2 or the linkage model is used, we condition on the current imputed genotype. This decreases the likelihood and seems to greatly inflate the variance of the likelihood. Limited experience indicates that this leads to poor performance for estimating K in the latter cases, and you should consider such estimates of K to be unreliable.

5.2 Sequence data, tightly linked SNPs and haplotype data

The *structure* model assumes that loci are independent within populations (i.e., not in LD within populations). This assumption is likely to be violated for sequence data, or data from non-recombining regions such as Y chromosome or mtDNA.

If you have sequence data or dense SNP data from multiple independent regions, then *structure* may actually perform reasonably well despite the data not completely fitting the model. Roughly speaking, this will happen provided that there is enough independence across regions that LD within regions does not dominate the data. When there are enough independent regions, the main cost of the dependence within regions will be that *structure* underestimates the uncertainty in the assignment of particular individuals.

For example, Falush et al. (2003b) applied *structure* to MLST (multi locus sequence) data from *H. pylori* to learn about the population structure and migration history of *H. pylori*. In that case, there is enough recombination within regions that the signal of population structure dominates background LD. (For more on MLST data, see also Section 10.) In an application to humans, Conrad et al. (2006) found that 3000 SNPs from 36 linked regions produced sensible (but noisy) answers in a worldwide sample of humans that largely agreed with previous results based on microsatellites [see their Supplementary Methods Figure SM2].

However, if the data are dominated by one or a few non- or low-recombining regions, then *structure* could be seriously misled. For example, if the data consisted of Y chromosome data only, then the estimated structure would presumably reflect something about the Y chromosome tree, and not population structure *per se*. The impact of using such data is likely to be that (1) the

algorithm underestimates the degree of uncertainty in ancestry estimates, and in the worst case, may be biased or inaccurate; (2) estimation of K is unlikely to perform well. If you have Y or mtDNA data plus a number of nuclear markers, a safe and valid solution is to recode the haplotypes from each linked region so that haplotypes are represented as a single locus with n alleles. If there are very many haplotypes, one could group related haplotypes together.

Note that the linkage model is not necessarily any better than the (no)-admixture models for dealing with these problems. The linkage model is not designed to deal with background LD within populations, and is likely to be similarly confused.

5.3 Multimodality

The *structure* algorithm starts at a random place in parameter space, and then converges towards a mode of the parameter space. (In this context, a mode can be thought of, loosely speaking, as a clustering solution that has high posterior probability.) When prior labels are not used, there is no inherent meaning in the numbering of the K clusters, and so there are $K!$ symmetric modes that correspond to permuting the cluster labels. In theory, *structure* might switch among these modes, but this does not normally occur for real data sets (Pritchard et al., 2000a). For preparing plots for publication, Noah Rosenberg's lab has a helpful program, CLUMPP, that lines up the cluster labels across different runs prior to data plotting (Section 10).

In addition to these symmetric modes, some data sets may have additional non-symmetric modes. The current implementation of *structure* does not normally cross between these in runs of realistic length. This means that different runs can produce substantially different answers, and longer runs will probably not fix this.

This is mainly an issue for very complex data sets, with large values of K, $K > 5$ or $K > 10$, say (but see the example of Data Set 2A in Pritchard et al. (2000a)). You can examine the results for Q to get an idea of whether this seems to be happening. A careful analysis of this type of situation was presented by Rosenberg et al. (2001), for a data set where the estimated K was around 19.

6 Running *structure* from the command line

There are a number of program parameters that are set by the user. These are in two files (*mainparams* and *extraparams*), which are read every time the program executes. *mainparams* specifies the input format for the data file and the most basic run parameters. *extraparams* specifies a wider variety of program options. You will need to set all the values in *mainparams*, while the default values in *extraparams* are probably ok to begin with. Note that the default model assumes admixture, and does not make use of the user-defined PopData.

Each parameter is printed in all-caps in one of these two files, preceded by the word "#define". (They are also printed in all-caps throughout this document.) The value is set immediately following the name of the parameter (eg "#define NUMREPS 1000" sets the number of MCMC repetitions to 1000).

Following each parameter definition, there is a brief comment (marked "//"), describing the parameter. This includes an indication of what sort of value is expected. These include: "(str)", for string (used for the names of the input and output files); "(int)", for integer; "(d)", for double (i.e., a real number such as 3.14); and "(B)", for Boolean (i.e., the parameter takes values TRUE or FALSE by setting this to 1 or 0, respectively).

The program is insensitive to the order of the parameters, so you can re-arrange them or add comments, etc. The values of all parameters used for a given run are printed at the end of the

output file.

6.1 Command-line changes to parameter values

In order to simplify batch runs and make it easier to run simulations involving *structure*, we have added command-line flags that update the values of certain parameters, over-riding the values set in *mainparams*. These are as follows:

-m (mainparams) Read a different parameter input file instead of *mainparams*.

-e (extraparams) Read a different parameter input file instead of *extraparams*.

-s (stratparams) Read a different parameter input file instead of *stratparams*. (For use with the accompanying program, *STRAT*, for association mapping.)

-K (MAXPOPS) Change the number of populations.

-L (NUMLOCI) Change the number of loci.

-N (NUMINDS) Change the number of individuals.

-i (input file) Read data from a different input file.

-o (output file) Print results to a different output file.

Thus, to over-ride one of the preset parameter values, we invoke *structure* and then use the relevant flag, followed by the new parameter value. The flag and new value are separated by a space. The flags can be used in any order.

For example, to change the number of assumed populations to 5, and direct the output to a file called *output5*, we could call *structure* as follows:

```
./structure -K 5 -o output5
```

7 Program parameters

In this section we list all of the parameters that can be set by the user. These are ordered according to the parameter files that are used in the command-line version of *structure*.

7.1 Parameters in file *mainparams*.

The user will need to set all of these parameters before running the program. Several of these parameters (LABEL, POPDATA, POPFLAG, PHENOTYPE, EXTRACOLS) indicate whether particular types of data are present in the input file; these are described in Section 2.

Basic Program Parameters.

MAXPOPS (int) Number of populations assumed for a particular run of the program. Pritchard et al. (2000a) call this K. Sometimes (depending on the nature of the data) there is a natural value of K that can be used, otherwise K can be estimated by checking the fit of the model at different values of K (see Section 4).

BURNIN (int) Length of burnin period before the start of data collection. (See Section 3.1.)

NUMREPS (int) Number of MCMC reps after burnin. (See Section 3.1.)

Input/Output files.

INFILE (string) Name of input data file. Max length 30 characters (or possibly less depending on operating system).

OUTFILE (string) Name for program output files (the suffixes "_1", "_2", ...,"_*m*" (for intermediate results) and "_f" (final results) are added to this name). Existing files with these names will be overwritten. Max length of name 30 characters (or possibly less depending on operating system).

Data file format.

NUMINDS (int) Number of individuals in data file.

NUMLOCI (int) Number of loci in data file.

PLOIDY (int) Ploidy of the organism. Default is 2 (diploid).

MISSING (int) Value given to missing genotype data. Must be an integer, and must not appear elsewhere in the data set. Default is -9.

ONEROWPERIND (Boolean) The data for each individual are arranged in a single row. E.g., for diploid data, this would mean that the two alleles for each locus are in consecutive order in the same row, rather than being arranged in the same column, in two consecutive rows. See section 2 for details about input formats.

LABEL (Boolean) Input file contains labels (names) for each individual. 1 = Yes; 0 = No.

POPDATA (Boolean) Input file contains a user-defined population-of-origin for each individual. 1 = Yes; 0 = No.

POPFLAG (Boolean) Input file contains an indicator variable which says whether to use popinfo when USEPOPINFO==1 (see below). 1 = Yes; 0 = No.

PHENOTYPE (Boolean) Input file contains a column of phenotype information. 1 = Yes; 0 = No.

EXTRACOLS (int) Number of additional columns of data after the Phenotype before the genotype data start. These are ignored by the program. 0 = no extra columns.

MARKERNAMES (Boolean) The top row of the data file contains a list of L names corresponding to the markers used.

RECESSIVEALLELES (Boolean) Next row of data file contains a list of L integers indicating which alleles are recessive at each locus. Setting this to 1 implies that the dominant marker model is in use.

MAPDISTANCES (Boolean) The next row of the data file (or the first row if GENENAMES==0) contains a list of mapdistances between neighboring loci.

Advanced data file options.

PHASED (Boolean) For use with linkage model. Indicates that data are in correct phase. If (LINKAGE=1, PHASED=0), then PHASEINFO can be used–this is an extra line in the input file that gives phase probabilities. When PHASEINFO =0 each value is set to 0.5, implying no phase information. When the linkage model is used with polyploids, PHASED=1 is required.

PHASEINFO (Boolean) The row(s) of genotype data for each individual are followed by a row of information about haplotype phase. This is for use with the linkage model only. See sections 2 and 3.2 for further details.

MARKOVPHASE (Boolean) The phase information follows a Markov model. See sections 2.2 and 9.6 for details.

NOTAMBIGUOUS (int) For use with polyploids when RECESSIVEALLELES=1. Defines the code indicating that genotype data at a marker are unambiguous. Must not match MISSING or any allele value in the data.

7.2 Parameters in file *extraparams*.

These options allow the user to refine the model in various ways, and do more involved analyses. The default values are probably fine to begin with. For Boolean options, type 1 for "Yes", or "Use this option"; 0 for "No" or "Don't use this option".

Program options.

NOADMIX (Boolean) Assume the model without admixture (Pritchard et al., 2000a). (Each individual is assumed to be completely from one of the K populations.) In the output, instead of printing the average value of Q as in the admixture case, the program prints the posterior probability that each individual is from each population. 1 = no admixture; 0 = model with admixture.

LINKAGE (Boolean) Use the linkage model. See section 3.2. **RLOG10START** sets the initial value of recombination rate r per unit distance. **RLOG10MIN** and **RLOG10MAX** set the minimum and maximum allowed values for log10r. RLOG10PROPSD sets the size of the proposed changes to log10r in each update. The front end makes some guesses about these, but some care on the part of the user in required to be sure that the values are sensible for the particular application.

USEPOPINFO (Boolean) Use prior population information to assist clustering. See also MIGRPRIOR and GENSBACK. Must have POPDATA=1.

FREQSCORR (double) Use the "F model", in which the allele frequencies are correlated across populations (Falush et al., 2003a). More specifically, rather than assuming a prior in which the allele frequencies in each population are independent draws from a uniform Dirichlet distribution, we start with a distribution which is centered around the mean allele frequencies in the sample. This model is more realistic for very closely related populations (where we expect the allele frequencies to be similar across populations), and can produce better clustering (section 3.3). The prior of F_k is set using FPRIORMEAN, and FPRIORSD. There may be a tendency to overestimate K when FREQSCORR is turned on.

ONEFST (Boolean) Assume the same value of F_k for all populations (analogous to Wright's traditional F_{ST}). This is not recommended for most data, because in practice you probably expect different levels of divergence in each population. When $K = 2$ it may sometimes be difficult to estimate two values of F_{ST} separately (but see Harter et al. (2004)). When you're trying to estimate K, you should use the same model for all K (we suggest ONEFST=0).

INFERALPHA (Boolean) Infer the value of the model parameter α from the data; otherwise α is fixed at the value ALPHA which is chosen by the user. This option is ignored under the NOADMIX model. (The prior for the ancestry vector Q is Dirichlet with parameters $(\alpha, \alpha, ..., \alpha)$. Small α implies that most individuals are essentially from one population or another, while $alpha > 1$ implies that most individuals are admixed.)

POPALPHAS (Boolean) Infer a separate α for each population. Not recommended in most cases but may be useful for situations with asymmetric admixture.

ALPHA (double) Dirichlet parameter (α) for degree of admixture (this is the initial value if INFERALPHA==1).

INFERLAMBDA (Boolean) Infer a suitable value for λ. Not recommended for most analyses.

POPSPECIFICLAMBDA (Boolean) Infer a separate λ for each population.

LAMBDA (double) parameterizes the allele frequency prior, and for most data the default value of 1 seems to work pretty well. If the frequencies at most markers are very skewed towards low/high frequencies, a smaller value of λ may potentially lead to better performance. It doesn't seem to work very well to estimate λ at the same time as the other hyperparameters, α and F.

Priors.

These values are used to parametrize the assumed probability models. In most cases the default settings should be fairly sensible and you may not want to worry about these.

FPRIORMEAN, FPRIORSD (double) See FREQSCORR. The prior for F_k is taken to be Gamma with mean FPRIORMEAN, and standard deviation FPRIORSD. Our default settings place a lot of weight on small values of F. We find that this makes the algorithm sensitive to subtle structure, but at some increased risk of overestimating K (Falush et al., 2003a).

UNIFPRIORALPHA (Boolean), **ALPHAMAX** (double) Assume a uniform prior for α which runs between 0 and ALPHAMAX. This model seems to work fine; the alternative model (when UNIFPRIORALPHA=0) is to take α as having a Gamma prior, with mean **ALPHAPRIORA** $\times$ **ALPHAPRIORB**, and variance ALPHAPRIORA $\times$ ALPHAPRIORB^2.

LOG10RMIN, LOG10RMAX, LOG10PROPSD, LOG10RSTART (double) When the linkage model is used, the switch rate r is taken to have a uniform prior on a log scale, between LOG10RMIN and LOG10RMAX. These values need to be set by the user to make sense in terms of the scale of map units being used.

Using prior population information.

GENSBACK (int) This corresponds to G (Pritchard et al., 2000a). When using prior population information for individuals (USEPOPINFO=1), the program tests whether each individual has an immigrant ancestor in the last G generations, where $G = 0$ corresponds to the individual being an immigrant itself. In order to have decent power, G should be set fairly small (2, say) unless the data are highly informative.

MIGRPRIOR (double) Must be in [0,1]. This is ν in Pritchard et al. (2000a). Sensible values might be in the range 0.001—0.1.

PFROMPOPFLAGONLY (Boolean) This option, new with version 2.0, makes it possible to update the allele frequencies, P, using only a prespecified subset of the individuals. To use this, include a POPFLAG column, and set POPFLAG=1 for individuals who should be used to update P, and POPFLAG=0 for individuals who should not be used to update P. This can be used both with, or without USEPOPINFO turned on.

This option will be useful, for example, if you have a standard reference set of individuals from known populations, and then you want to estimate the ancestry of some unknown individuals. Using this option, the q estimate for each unknown individual depends only on the reference set, and not on the other unknown individuals in the sample. This property is sometimes desirable.

Output options

PRINTNET (Boolean) Print the "net nucleotide distance" between clusters. This distance between populations A and B, $\mathrm{D_{AB}}$, is calculated as

$$\mathrm{D}_{A,B} = \frac{1}{L}\sum_{l=1}^{L}\left[1 - \sum_{j=1}^{J_l}\hat{p}_{A,j}^{(l)}\ \hat{p}_{B,j}^{(l)}\right] - \frac{(\mathrm{H}_A - \mathrm{H}_B)}{2}, \tag{5}$$

where $\hat{p}_{x,j}^{(l)}$ is the estimated allele frequency of allele j at locus l in population x, L is the number of loci, J_l the number of alleles at locus l and where

$$\mathrm{H}_x = \frac{1}{L}\sum_{l=1}^{L}\left[1 - \sum_{j=1}^{J_l}\hat{p}_{x,j}^{(l)\ 2}\right]. \tag{6}$$

In words, the net nucleotide distance is the average probability that a pair of alleles, one each from populations A and B are different, less the average within-population heterozygosities. Perhaps more intuitively, this can be thought of as being the average amount of pairwise difference between alleles from *different* populations, beyond the amount of variation found within each population. The distance has the appropriate property that similar populations have distances near 0, and in particular, $\mathrm{D_{AA}} = 0$. Notice that the distance is symmetric, so that $\mathrm{D_{AB}} = \mathrm{D_{BA}}$. This distance is suitable for drawing trees of populations to help visualize the levels of difference among the clusters (Falush et al., 2003b).

PRINTKLD (Boolean) [**Deprecated**] Print the "Kullback-Leibler" divergence "D" between clusters. These estimates can be unstable due to alleles that are rare in some populations. **This option is being phased out** and it currently prints net nucleotide distances when this option is called.

PRINTLAMBDA (Boolean) Print current value of λ to screen.

PRINTQSUM (Boolean) Print summary of current Q estimates to screen; this prints an average for each value of PopData.

SITEBYSITE (Boolean) (Linkage model) Print a complete summary of assignment probabilities for every genotype in the data. This is printed to a separate file with the suffix "ss". This file can be big!

PRINTQHAT (Boolean) When this is turned on, the point estimate for Q is not only printed into the main results file, but also into a separate file with suffix "q". This file is required in order to run the companion program STRAT.

UPDATEFREQ (int) Frequency of printing updates to the screen. Set automatically if this =0.

PRINTLIKES (Boolean) Print the current value of the likelihood to the screen in every iteration.

INTERMEDSAVE (int) If you're impatient to see preliminary results before the end of the run, you can have results printed to file at intervals during the MCMC run. A total of INTERMEDSAVE such files are printed, at equal intervals following the completion of the BURNIN. Turn this off by setting to 0. Names of these files created using OUTFILE name.

ECHODATA (Boolean) Print a brief summary of the data set to the screen and output file. (Prints the beginnings and ends of the top and bottom lines of the input file to allow the user to check that it has been read correctly.)

ANCESTDIST (Boolean) Collect information about the distribution of Q for each individual, as well as just estimating the mean. When this is turned on, the output file includes the left- and right-hand ends of the probability intervals for each $q(i)$. (A probability interval is the Bayesian analog of a confidence interval.) The values printed show the middle $100p\%$ of the probability interval, where p is a number in the range 0.0 to 1.0 and is set using **ANCESTPINT**. The distribution of Q is estimated by recording the number of hits in each of a number of boxes between 0 and 1, to form a sort of histogram. The width of these boxes, which are of equal size, is set using **NUMBOXES**.

Miscellaneous

COMPUTEPROB (Boolean) Print the log-likelihood of the data at each update, and estimate the probability of the data given K and the model (see section 4). This is used in estimating K, and is also a useful diagnostic for whether the burnin is long enough. The main reason for turning this off would be to speed up the program ($\sim$ 10–15%).

ADMBURNIN (int) (For use when RECOMBINE=1.) When using the linkage model, a short burnin with the admixture model (say 500 iterations) is strongly recommended in most circumstances. Without such a burnin, the linkage model often produces peculiar results.

Set ADMBURNIN < BURNIN. We have dropped a related parameter (NOADMBURNIN) that was in Version 1.

ALPHAPROPSD (double) The Metropolis-Hastings update step for α involves picking a value α' from a Normal with mean α and standard deviation ALPHAPROPSD> 0. The value of ALPHAPROPSD does not affect the asymptotic behaviour of the Markov chain, but may have a substantial impact on the rate of convergence. If there is a lot of information about α, small values of ALPHAPROPSD are preferable to obtain a reasonable acceptance rate. If there's not much information about α, larger values produce better mixing.

STARTATPOPINFO (Boolean) Use given populations as the initial condition for population origins. (Need POPDATA==1). This option provides a check that the Markov chain is converging properly in cases where you expected the inferred structure to match the input labels, and it did not. This option assumes that the PopData in the input file are between 1 and k where $k \leq$MAXPOPS. Individuals for whom the PopData are not in this range are initialized at random.

RANDOMIZE (Boolean) Use a different random number seed for each run (this is taken from the system clock).

METROFREQ (int) Frequency of using a Metropolis-Hastings step to update Q under the admixture model. When this is used, a new proposal $q^{(i)\prime}$ is chosen for each $q^{(i)}$. This proposal is sampled from the prior (ie $q^{(i)\prime} \sim \mathcal{D}(\alpha, \alpha, ..., \alpha)$). The rationale for having this update is that it may improve mixing when alpha is quite small, by making it easier for individuals to jump between populations. The Metropolis-Hastings move is used once every METROFREQ iterations. If METROFREQ is set to 0, it is never used.

REPORTHITRATE (Boolean) Report acceptance rate of Metropolis update for $q^{(i)}$ (see METROFREQ).

8 Front End

This section provides some general instructions, and a bit of advice about using the front end. General topics are discussed above, and you can get some more detailed information about some of the various parameter options by looking in section 6.

8.1 Download and installation.

First, download the appropriate program file from the web page. There are separate versions for different platforms (at present Windows, Sun, Linux and Mac OS X).

The Windows file is an executable installation file. Double click on the icon to start the installation. You will be guided through the installation. Run the program by double clicking on the *structure* icon.

On a Unix or Mac system, put the file into a temporary directory. Then, unzip the file ("gzip -dc <*filename*> | tar xvf - "), where <*filename*> is the name of the downloaded file. Run the installation script by typing "./install". Upon successful installation, a structure startup script will be created, this script can also be moved into a standard directory for programs, e.g., /usr/local/bin/. To start the front end, simply execute this startup script.

Except for Windows OS, we no longer distribute the Java Virtual Machine with the *structure* **package** (starting from *structure* Version 2.2). A Java Runtime Environment (JRE Version $>$ 1.5.0) by Sun Microsystem is required before *structure* installation. The compatible

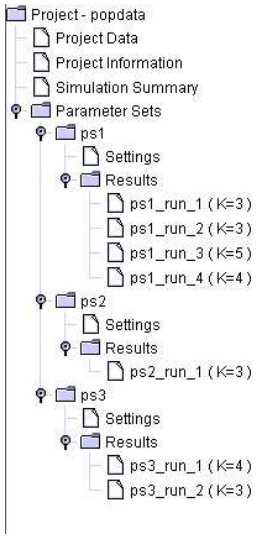

Figure 1: Example showing the components of a project. *Project Data* is the data file; *Project Information* specifies data file format. *Simulation Summary* provides a summary of all MCMC simulations run as part of this project. *Parameter Sets* consists of three groups of MCMC runs that used different parameter settings: ps1, ps2, and ps3; each of these shows the Settings, as well as a list of results for completed MCMC runs with these parameter values. The user can click on any of these to see details.

JRE for various operating systems can be downloaded free from `http://www.java.com/download`. Installation instructions for the JRE can be found on that website.

8.2 Overview.

The front end organizes data analysis into "projects". Each project is connected to a single data file. When creating a project, the user also provides information that specify how to read the data file (number of loci, number of individuals, etc). These are characteristics of the data file, and are always the same within this project.

Each project also contains one or more "parameter sets". These allow the user to specify the details of the MCMC runs, including the number of repetitions, burnin length, etc, as well as specifying the model of analysis (e.g., whether to allow admixture, models of allele frequencies, etc). The user can then run the Markov chain at chosen values of K, for a given parameter set. Figure 1 shows an example of the components of a project called "popdata".

The program can then be run, using these parameter values. The front end stores various summaries of the results, including a number of graphical plots, described below.

8.3 Building a project.

First you need to construct an input file. This is described in Section 2.

Now, click on File→New Project. This opens up a wizard to import the data (Figure 2). The data are copied from the specified input file into the work directory chosen for the project.

The wizard consists of four frames:

1. Specify the project directory, project name, and input data file. (Figure 2.)

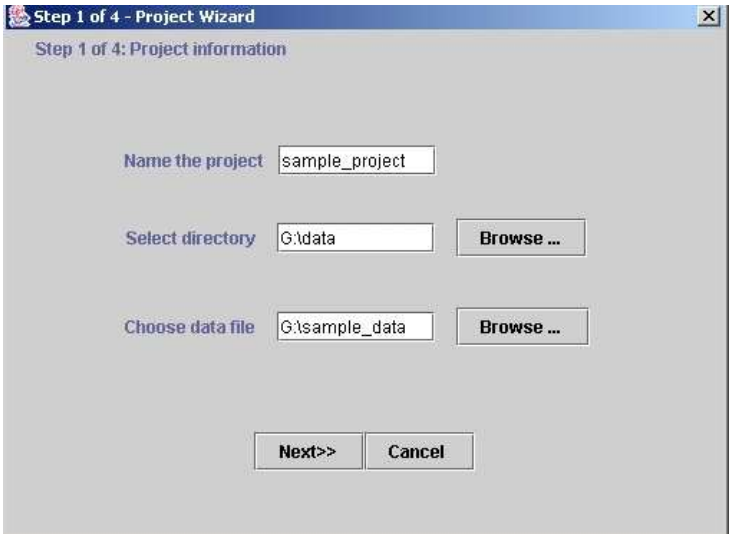

Figure 2: Importing the data (step 1). The user specifies the directory for the project (*data*, here), the name of the project directory (*sample_project*; this is a directory within *data*), and the data file to be read by the program *sample_data*.

2. Specify the basic characteristics of the data file (number of individuals, ploidy of the data (enter '2' for diploid organisms), number of loci, and the value that is used to indicate missing data. Click on "Show data file format" to get a summary of the lengths and number of lines in the data file. (Figure 3.)

3. (Rows) Specify which, if any, of the optional extra row data are present: row of marker names; row of inter-marker distances; and a row of phase data after each individual. Also tick the "single line" box if data for each individual are stored in a single row, instead of in the standard format of two rows per individual.

4. (Columns) Specify which of the optional column data are there: Individual ID (LABEL); Population of origin (POPDATA); USEPOPINFO flag– flag that says to use the POPDATA information for certain individuals when using the prior population information model; phenotype data (for use in association mapping (Pritchard et al., 2000b)); other extra columns of data prior to the genotype data that should be ignored by *structure*.

When you've finished these steps, you'll get a summary of the data format; if this looks correct, click on 'proceed'. The program will now attempt to load the data file and create the new project (Figure 4).

8.4 Configuring a parameter set.

Once you've successfully loaded a data file, you are ready to start running *structure*. You will create one or more "parameter sets"; these represent a whole list of choices that you make about how to analyze the data. We have entered a series of default settings, and these are good place to start. You will probably want to run *structure* multiple times for each parameter set, at different values of K, and the front end is set up to facilitate this.

Go to the pull-down menu under *Parameter Set*. You can create a new parameter set, modify an existing one, or delete one. Click on "New". You now see a dialogue box with four tabs (Figure 5). Click on each of these:

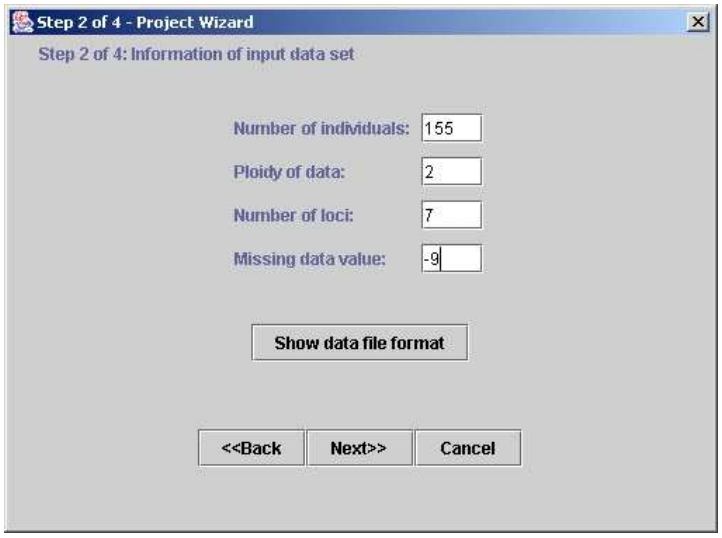

Figure 3: Importing the data (step 2)—Specifying the characteristics of the data file. In this, and the next two frames of the wizard (not shown), the user specifies the characteristics of the data file (number of loci, number of individuals, type of data, etc).

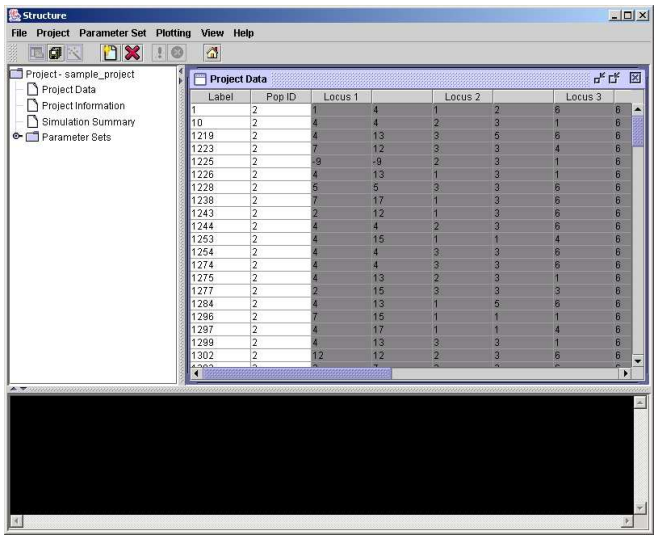

Figure 4: Upon successfully importing the data, the front end loads the data file and creates the new project.

Run Length. See section 3.1 for discussion of this. Note that the front end provides time series plots of some key parameters to help you assess whether the run length seems to be sufficient.

Ancestry Model. See section AncestryModels. The admixture model is a good place to start for most data sets. The linkage model will be disabled unless you entered linkage information about the markers. Section 6 provides some extra details about some of the detailed options, including GENSBACK and MIGRPRIOR under the "Use Population Information" option. Note that the linkage model is relatively computationally intensive.

Allele Frequency Model. See section 3.3. We recommend applying both the correlated frequencies model and the independent frequencies model. The correlated frequencies model has

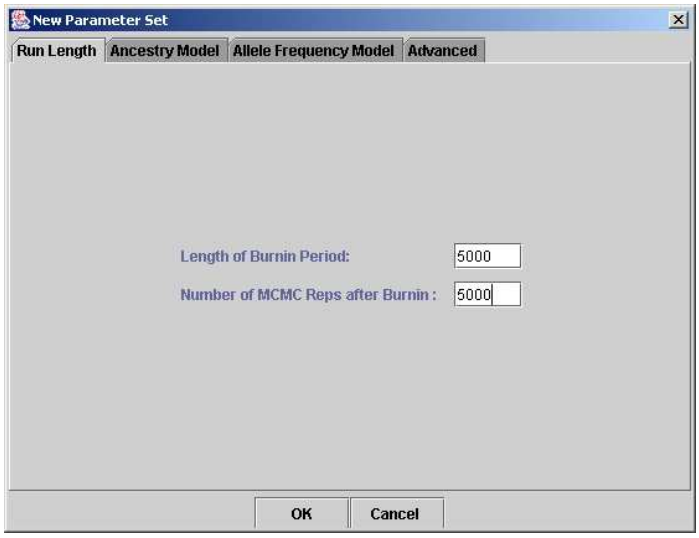

Figure 5: Specifying a new parameter set: setting the run length.

better power to detect subtle population structure, but the posterior probabilities for K may be biased upwards somewhat. The correlated frequencies model is parameterized in terms of F_{ST}, with a separate parameter for each population (details in section 7.2). Inferring λ is probably not necessary during initial investigation.

Advanced. Turning off the function that computes the posterior probabilities (for estimating K) speeds the program up significantly. You can also get the program to output posterior credible regions for the ancestry of each individual (see ANCESTDIST, section 7.2). "Initialize at POPINFO" is described in more detail under STARTATPOPINFO, section 7.2.

8.5 Running simulations.

Now that you have a parameter set, you can start the program running by going to [Parameter Set]→[Run]. You'll be asked to set the number of populations (K). You can also stop simulations in the same place ([Parameter Set]→[Stop]).

Text data will be printed to the console at the bottom of the screen (Figure 6). You can also view real-time time-series plots of various key summary statistics: F_{ST}, α, likelihoods, etc, (Figure 7).

Once you have more than one parameter set, you need to specify which one you want to use for new MCMC runs. At any time, one parameter set is designated as "active" (see the left-hand window). You can switch the active parameter set by going to [Parameter Set]→[Parameter Set List], and highlighting the appropriate choice or by double clicking the corresponding parameter set tip in the project tree.

8.6 Batch runs.

You can schedule a series of *structure runs* by going to [Project]→[Start a Job]. This opens up a scheduler that allows you to pick (1) parameter sets (use control+mouse to select multiple parameter sets) (2) values of K, (2) number of runs for each. See Figure 8.

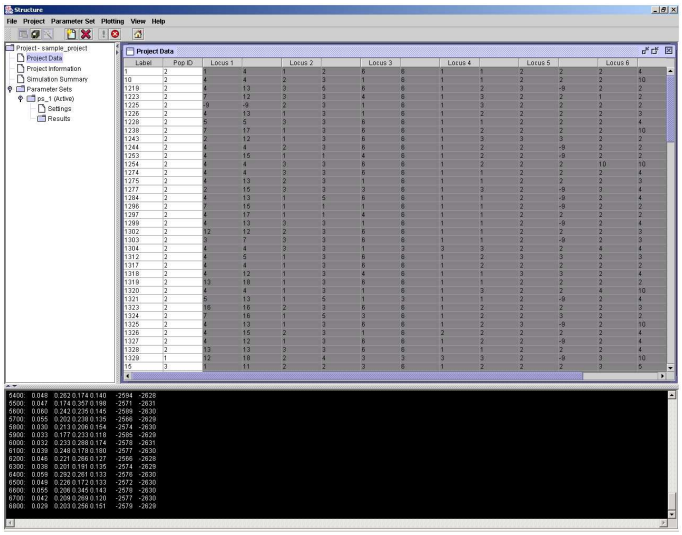

Figure 6: The run-time output is shown in the bottom console

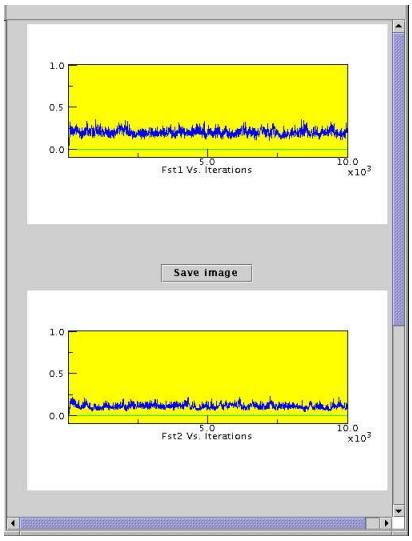

Figure 7: Time series plot of F_{ST}.

8.7 Exporting parameter files from the front end.

You can use the front end to automate writing text-based parameter files for use in the command-line version of structure. Go to [Project]→"generating parameter files...". This option may be useful, for example, in helping you set up large numbers of runs on a computing cluster.

8.8 Importing results from the command-line program.

The results from the command-line version of the *structure* program can be imported into the frontend by going to [File]→[Load structure result...]. You will be asked to provide 2 files: the structure results file (required; this file usually has an "_f" suffix) and a file containing the runtime *structure* output to the screen (optional). The latter can be obtained by running *structure* in the console with the output redirected into a file (e.g., structure > output.txt). Upon successfully loading files, you should be able to read the bar plot, triangle plot, and various time series plot (the

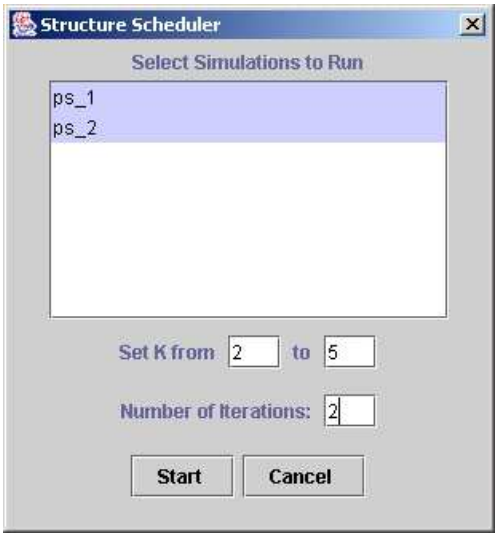

Figure 8: Creating a batch run job.

latter only if you provide the runtime file) in the front end.

8.9 Analyzing the results

Summaries. You can view a table of summary details of all runs completed so far by going to [View]→[Simulation Summary].

Text results. You can look at full text results for each run by going to the left-hand window, and clicking on the appropriate run. The full text output will appear in the right-hand window. Details for interpreting the text output are given in Section 9.

Plots of Ancestry estimates. We provide two types of plots of $\hat{Q}$ (the estimated membership coefficients for each individual, in each cluster. The first representation comes up automatically when you click on the corresponding run in the left-hand window. Each individual in the data set is represented by a single vertical line, which is partitioned into K colored segments that represent that individual's estimated membership fraction in each of the K inferred clusters.

The second representation of the ancestry of individuals plots everybody into a triangle (Figure 10). This type of plot is useful for visualizing the data for $K = 3$ (Pritchard et al., 2000a). It is kind of a fun tool for exploring higher-dimensional data, but the bar plots are usually easier to interpret.

Plots of summary statistics. We also provide plots of several interesting summary statistics. [Plotting]→[Data Plotting] contains time-series plots of the values of certain summaries during the course of a run. One example is shown in Figure 7; notice that there is a brief period at the start of the run where the values increase dramatically before reaching their stationary distribution. You should inspect these plots to be sure that the summary statistics seem to stabilize before the end of the burnin.

There are also histogram plots of F_{ST} and α (Figure 8.9). These are estimates of the posterior distribution of these parameters.

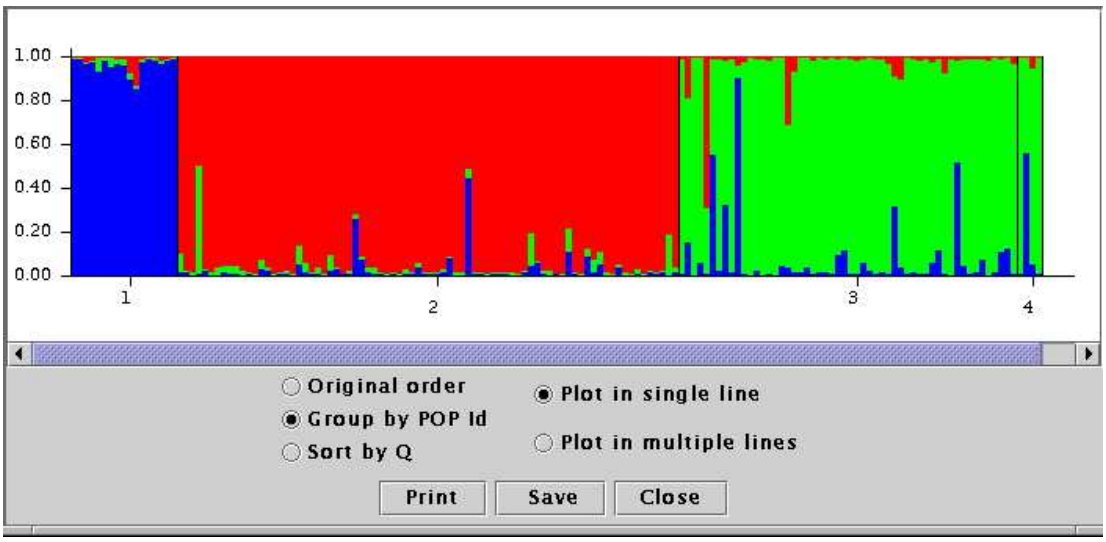

Figure 9: Summary plot of estimates of Q. Each individual is represented by a single vertical line broken into K colored segments, with lengths proportional to each of the K inferred clusters. The numbers (1..4) correspond to the predefined populations.

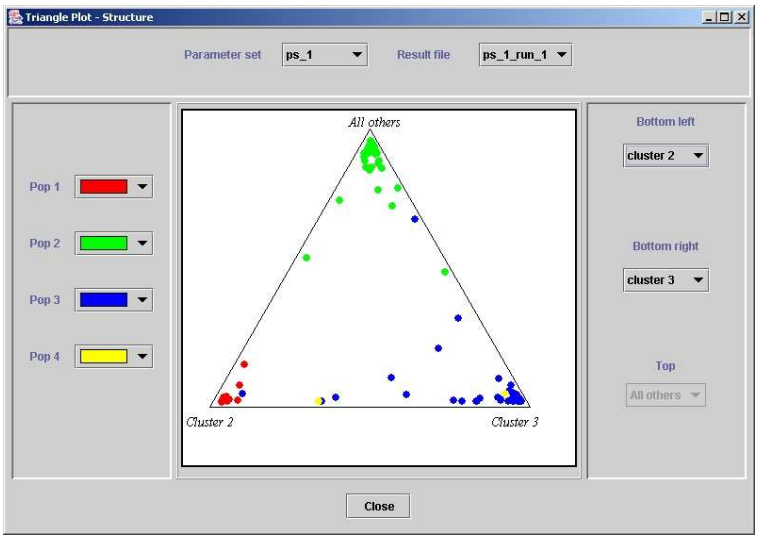

Figure 10: The triangle plot of Q. Each individual is represented by a colored point. The colors correspond to the prior population labels. The estimated ancestry vector for an individual consists of K components which add up to 1. When $K = 3$, the ancestry vectors can be plotted onto a triangle, as shown. For a given point, each of the three components is given by the distance to one edge of the triangle. Individuals who are in one of the corners are therefore assigned completely to one population or another. For $K > 3$, we represent the data by allowing the user to pick out two of the inferred clusters at a time, and then grouping all the other clusters together.

9 Interpreting the text output

This section describes the data that are printed to the console during the run, and to the output file. The front end also provides additional data plots, described below.

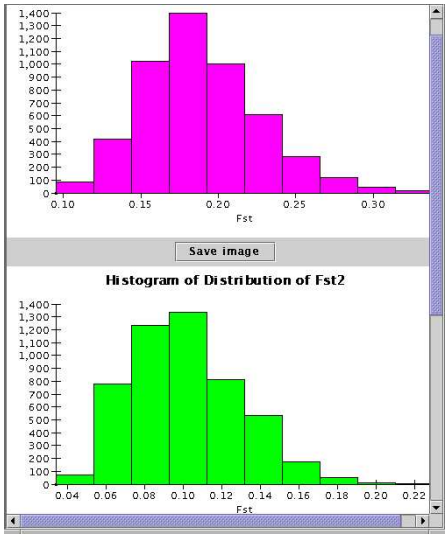

Figure 11: The histogram of Fst

9.1 Output to screen during run

```
Rep#:  Alpha   Corr   D1,2   Ln Like   Est Ln P(D)
   4500:  0.009   6.1   0.831   -2730       -2776
   4525:  0.010   9.1   0.806   -2716       -2776
   4550:  0.008   7.3   0.792   -2734       -2775
```

The example above shows part of the output to the screen during a run. Here, "rep" gives the number of MCMC iterations so far, including the burnin; "Alpha" is the current value of α; "Corr" is the current mean (across loci) value of the correlation coefficient $f^{(l)}$ (see FREQSCORR); "D1,2" is a measure of divergence between populations 1 and 2 (see PRINTNET); "Ln Like" is the log likelihood of the data given the current values of P and Q; "Est Ln P(D)" is the current estimate of $\ln(P(X|K))$ (averaging over all iterations since the end of the burnin period). Some of these columns may not be printed, depending on the program options chosen. If K gets large, it may be worth setting PRINTNET=0 in order to make the output easier to read.

9.2 Printout of Q

```
 Label (%Miss) Pop:  Inferred clusters (and 90% probability intervals)
1  17     (0)   2 :   0.977 0.023      (0.829,1.000) (0.000,0.171)
2 1219    (7)   2 :   0.997 0.003      (0.988,1.000) (0.000,0.012)
3 1223    (0)   2 :   0.833 0.167      (0.003,1.000) (0.000,0.997)
4 1329    (7)   1 :   0.005 0.995      (0.000,0.020) (0.980,1.000)
5  15     (0)   3 :   0.006 0.994      (0.000,0.016) (0.984,1.000)
```

When the program is run without using prior population information, the results for Q are presented in the format shown above (here K was set to 2). This is read as follows. Reading from row 1: Individual label (taken from data file) = 17; percentage of missing data for this individual = 0%; user-assigned population = 2; estimated membership in clusters 1 and 2 = 0.977 and 0.023 respectively (these are the mean values of $q^{(17)}$); 90% probability intervals on $q_1^{(17)}$ and $q_2^{(17)}$ are

(0.829,1.000) and (0.000,0.171), respectively. (Note that the probability intervals are shown only if ANCESTDIST is turned on.)

9.3 Printout of Q when using prior population information

When the program is run with prior population information, the printout is a bit different. Here we have run the same data with $K = 3$, USEPOPINFO=1, and GENSBACK = 1:

```
 Label (%Miss) Pop :
1  17    (0) 2 :  0.998 | Pop 1:  0.000 0.001 | Pop 3:  0.000 0.000
2 1219   (7) 2 :  1.000 | Pop 1:  0.000 0.000 | Pop 3:  0.000 0.000
3 1223   (0) 2 :  0.612 | Pop 1:  0.000 0.000 | Pop 3:  0.004 0.383
4 1329   (7) 1 :  1.000 | Pop 2:  0.000 0.000 | Pop 3:  0.000 0.000
5  15    (0) 3 :  0.999 | Pop 1:  0.000 0.001 | Pop 2:  0.000 0.000
```

In general, the first column of results (following "Pop : ") shows the posterior probability that the individual in question is correctly assigned to the given population. The subsequent columns show the probabilities that it is from, or has ancestry in, the other populations. There are GENSBACK+1 entries for each of the other populations, showing the probability that an individual is from that population, has a parent, grandparent, great-granparent,... etc, from that population (in this order).

For example, reading from row 3: Individual 1223 (who has 0% missing data) is actually from the presumed population (2) with probability 0.612. There is (approximately) zero posterior probability that this individual has recent ancestry in population 1, but it may have recent ancestry in population 3 (the probabilities are 0.004, and 0.383, that the individual is *from* population 3, or has a single *parent* from population 3, respectively).

9.4 Printout of allele-frequency divergence

```
Allele-frequency divergence among clusters (net nucleotide distance), computed using
point estimates of P.
         1         2
   1     -     0.0357
   2  0.0357      -
Average distances (expected heterozygosity) between individuals in same cluster:
   cluster 1 :  0.7845
   cluster 2 :  0.7686
```

This example shows a matrix of the pairwise net distance among $K = 2$ clusters (above), and the (expected) heterozygosity within each cluster (below). See PRINTNET in Section 7.2 for details on how these distances are computed. Note that the pairwise matrix is symmetric (ie $D_{ij} = D_{ji}$). These distances replace the Kullback-Leibler divergence of *structure* Versions < 2.2.

You may find it helpful to draw trees to represent the distances among clusters, based on the net nucleotide distances. Examples are shown in Falush et al. (2003b).

9.5 Printout of estimated allele frequencies (P)

```
Locus 5
   3 alleles
```

```
19.0% missing data
2 0.511 0.821 0.656
3 0.444 0.171 0.317
1 0.045 0.008 0.027
```

This example shows a printout for the estimated allele frequencies (P) at locus 5. Columns 2,3, and 4 show the estimated frequency (in clusters 1,2 and 3, respectively) of the allele listed in column 1.

9.6 Site by site output for linkage model.

When the SITEBYSITE option is chosen, there is a separate output file with a suffix "_ss", that contains posterior population of origin assignments for each allele copy at each locus for each individual. For large datasets, this file may be several megabases.

Each line shows the assignment probabilities for one locus for one individual.The first two columns of the line indicate the number of the individual (ranging from 1 to NUMINDS) and the number of the locus (ranging from 1 to NUMLOCI) in the order that they occur in the data file.

The format of the posterior assignment probabilities depends on the parameter combinations. If LINKAGE=0 or PHASED=1 then the first K rows of output give the probability that the first allele copy at the locus comes from populations 1..K. For diploid or polyploid data, analogous probabilities for subsequent allele copies are shown in further columns.

If the linkage model is used (LINKAGE=1) and the data is not fully phased (PHASED=0) the posterior assignment probabilites for the allele copies at each locus can be strongly co-dependent. Structure therefore outputs joint assignment probabilities for the two allele copies implying K^2 entries for each locus (note that this option is not available for PLOIDY$\neq$ 2).

If MARKOVPHASE=1 then the first K columns give the probabilities that the first allele copy in the datafile is in population 1 and the second allele copy is in population 1..K, with subsequent columns relating to probabilities with the first allele copy in populations 2..K.

If MARKOVPHASE=0, then instead of referring to the first and second listed allele copies in the data file, the probabilities refer to the population of origin of maternal and paternal strands. If there is no phase information (PHASEINFO=0), then the posterior probability matrix should theoretically be symmetric, such that the probability the maternal allele is in population k_1 and the paternal allele is in k_2 will be equal to the probability that the maternal allele is in population k_2 and the paternal allele is in population k_1. In practice, because MCMC is used to estimate the matrix, there will be noticeable deviations from symmetry if NUMREPS is small.

For example, suppose that the below is site-by-site output for two loci for a diploid individual with no phase information, with MARKOVPHASE=0.

1	1	0.001	0.000	0.008	0.000	0.000
			0.001	0.007	0.001	0.982
1	2	0.001	0.000	0.008	0.000	0.000
			0.001	0.008	0.001	0.982

Then in order to calculate the assignment probabilities of the maternal and paternal allele copies at for the first locus the numbers are summed as follows:

locus 1	pop1	pop2	pop3	origin of maternal(X) chromosome
pop1	0.001	0.000	0.008	0.009
pop2	0.000	0.000	0.001	0.001
pop3	0.008	0.000	0.982	0.990
origin of paternal chromosome (missing)	0.009	0.000	0.991	

In this example, the data is from an X chromosome of a male, so in fact the second allele copy is missing.

Note that the format is simplified from version 2.1, where the results were placed in the same file as the rest of the output. Labels and marker names are no longer printed and the output prints each number in decimal format instead of scientific. These changes were made in the interests of compactness.

10 Other resources for use with *structure*

10.1 Plotting *structure* results

CLUMPP and *distruct* are a pair of programs produced by Noah Rosenberg's lab for making nice plots of the Q matrix. Similar plots are produced by the front end, but these two programs provide much finer control of the graphical output. See
`http://rosenberglab.bioinformatics.med.umich.edu/software.html`

10.2 Importing bacterial MLST data into *structure* format

xfma2struct by Xavier Didelot and Daniel Falush takes haploid sequence data in extended Fasta format and converts them into *structure* format. See the ClonalFrame website at
`http://bacteria.stats.ox.ac.uk/`

11 How to cite this program

The appropriate citation for the basic method is to Pritchard et al. (2000a). The paper by Falush et al. (2003a) is the appropriate reference for the linkage model and the correlated allele frequencies model implemented in Version 2.0. The methods for ambiguous genotype data such as dominant markers (new in Version 2.2) are described by Falush et al. (2007).

12 Bibliography

References

Beaumont, M., Gottelli, D., Barratt, E. M., Kitchener, A. C., Daniels, M. J., Pritchard, J. K., and Bruford, M. W. (2001). Genetic diversity and introgression in the Scottish wildcat. *Molecular Ecology*, 10:319–336.

Conrad, D., Jakobsson, M., Coop, G., Wen, X., Wall, J., Rosenberg, N., and Pritchard, J. (2006). A worldwide survey of haplotype variation and linkage disequilibrium in the human genome. *Nature Genetics*, 38:1251–1260.

Evanno, G., Regnaut, S., and Goudet, J. (2005). Detecting the number of clusters of individuals using the software STRUCTURE: a simulation study. *Mol. Ecol.*, 14:2611–2620.

Falush, D., Stephens, M., and Pritchard, J. K. (2003a). Inference of population structure: Extensions to linked loci and correlated allele frequencies. *Genetics*, 164:1567–1587.

Falush, D., Stephens, M., and Pritchard, J. K. (2007). Inference of population structure using multilocus genotype data: dominant markers and null alleles. *Molecular Ecology Notes*.

Falush, D., Wirth, T., Linz, B., Pritchard, J. K., Stephens, M., and 13_others (2003b). Traces of human migrations in *Helicobacter pylori* populations. *Science*, 299:1582–1585.

Harter, A., Gardner, K., Falush, D., Lentz, D., Bye, R., and Rieseberg, L. (2004). Origin of extant domesticated sunflowers in eastern North America. *Nature*, 430:201–205.

Murgia, C., Pritchard, J. K., Kim, S., Fassati, A., and Weiss., R. (2006). Clonal origin and evolution of a transmissible cancer. *Cell*, 126:477–487.

Pritchard, J. K., Stephens, M., and Donnelly, P. (2000a). Inference of population structure using multilocus genotype data. *Genetics*, 155:945–959.

Pritchard, J. K., Stephens, M., Rosenberg, N. A., and Donnelly, P. (2000b). Association mapping in structured populations. *Am. J. Hum. Genet.*, 67:170–181.

Rosenberg, N. A., Burke, T., Elo, K., Feldman, M. W., Freidlin, P. J., Groenen, M. A., Hillel, J., Maki-Tanila, A., Tixier-Boichard, M., Vignal, A., Wimmers, K., and Weigend, S. (2001). Empirical evaluation of genetic clustering methods using multilocus genotypes from 20 chicken breeds. *Genetics*, 159:699–713.