

# Finestructure Version 0.0.2

## User Manual

### *Population assignment for dense genetic data*

[www.paintmychromosomes.com](http://www.paintmychromosomes.com)

Last updated: 28<sup>th</sup> October 2011

**Daniel Lawson**

Department of Mathematics

University of Bristol

University Walk, Bristol, BS8 1TW. UK.

[dan.lawson@bristol.ac.uk](mailto:dan.lawson@bristol.ac.uk)

[danjlawson2000@yahoo.com](mailto:danjlawson2000@yahoo.com)

In collaboration with:

**Garrett Hellenthal** (Oxford)

**Simon Myers** (Oxford)

**Daniel Falush** (Max Planck Institute, Leipzig)

## Table of Contents

Finestructure Version 0.0.2.....	1
User Manual.....	1
About.....	2
Pipeline overview.....	2
Getting Started.....	2
What you need.....	2
Installation.....	2
GUI Example.....	3
Load a datafile.....	3
Perform MCMC: generate the MCMC file.....	3
Find optimal assignment: generate the tree file.....	3
Save the GUI state.....	4
Experiment with the view.....	4
Main Window.....	4
Screen Layout.....	5
Tree order manipulation.....	5
Menu options.....	6
Left Panel options.....	7
Manage Files.....	8
MCMC Input/Output/Generation.....	9
MCMC Traces.....	9
Principal Components Analysis, PCA .....	9
Command Line.....	10
Continents, super-individuals and force files.....	11
Tree types.....	12
Output file format.....	12

## About

FineSTRUCTURE is software to perform population assignment on genetics data. The correct reference is: Lawson, Hellenthal, Myers and Falush 2011 “Inference of population structure using dense genotype data”, (submitted to PloS Genetics), which contains the motivation and justification behind the method. See [www.paintmychromosomes.com](http://www.paintmychromosomes.com) for the most up to date information; where this manual is at odds with the website, the website is likely to be correct.

Similar in concept to STRUCTURE (<http://pritch.bsd.uchicago.edu/software.html>) fineSTRUCTURE assigns individuals to populations using a model for the expected variability. The advantage of our approach is that very large numbers of SNPs (Single Nucleotide Polymorphisms) can be used and these do not have to be in linkage equilibrium (in fact, it can be better if they are not). However, to achieve this efficiency **the data must be pre-processed** either using a linkage or no-linkage model, as described in the paper. Software to do this called ChromoPainter is included with this package.

Software is available both for Windows and for Linux compatible operating systems. There is both a command-line and GUI (Graphical User Interface) version of the software; the command line is only available for Linux.

**This software is Beta:** please report all bugs to the author at [dan.lawson@bristol.ac.uk](mailto:dan.lawson@bristol.ac.uk).

## Pipeline overview

- 0a. Obtain phased data.
- 0b. Run ChromoPainter to obtain a **input coancestry matrix** and  $c$  value.
  1. Run fineSTRUCTURE to obtain an results **MCMC sample file**.
  2. Run fineSTRUCTURE to obtain a results **tree file** containing a tree relating populations as well as a 'best population assignment' from the MCMC file.
  3. View the results in the GUI. (Steps 1 and 2 can be done through the GUI or command-line).

## Getting Started

### What you need

1. A genetics data set, processed into a “Coancestry matrix” of the form  $X_{ij}$  = the expected number of “genetic elements” donated to individual  $i$  from individual  $j$ .
2. An estimate of the “inflation factor”  $c$  for this dataset.
3. Correctly installed/compiled Finestructure software.

Parts 1 and 2 should be output for your dataset by the included ChromoPainter software. Note that a “genetic element” is a SNP when markers are considered as unlinked, or a contiguous segment of DNA uninterrupted by recombination in the case of linked markers. The inflation factor  $c$  computes the “effective number of independent genetic elements” and therefore accounts for the differences in these two models.

### Installation

On Windows, simply unzip, and run “finegui” from its directory. The required DLL files should be included. You can alternatively use the installer which will install it like any other program.

On Linux, a smooth installation procedure is not yet completed. You need to compile from source, for which you need the libraries the code depends on.

For the command line program: You need the Gnu Scientific Library (GSL, package libgsl0-dev on Ubuntu) and automake (package automake in Ubuntu), as well as the GCC c++ compiler (package g++ on ubuntu). Then do the following, from the directory that you downloaded the code:

```
tar -xzf fineststructure-0.0.2GUI.tar.gz # (check correct version. The GUI includes the command
                                         # line version)
cd fineststructure-0.0.2
./configure
make
sudo make install # optional
```

For the GUI: You need GSL, as well as wxwidgets (package wx2.8-headers on Ubuntu). After doing the above, the following commands should work:

```
cd gui
./configure
make
sudo make install # optional
```

If these fail, please feel free to contact the author for help. It is clear that there may be other dependencies that are system specific, or configuration of the GUI Makefile that may require attention on your system.

It should be possible to compile Finestructure for the Mac. If you succeed in doing this, or would like assistance doing so, please get in touch with the author (as we have no Mac to test on).

## GUI Example

An example dataset is provided, which is a simulated dataset of 150 regions of 5Mb (so 75Mb) of dense genetic data as described in Lawson Et al (2011). The process of generating the input dataset from “phase” file format is described in the manual for ChromoPainter. The resulting **input file** is a  $N \times N$  matrix where  $N=100$  is the number of individuals in the sample.

### *Load a datafile*

To load the dataset, start “Finegui” and go to “**File->Manage Files**” from the drop-down menu. Under the “**Raw data file**” section, first select “**Change file**”. Using the dialog, select the file “data\_unlinked.chunkcounts.out” from the “examples” folder. Then select “**Read file**” in the same section. This should read the raw data matrix into the main window (and also the correct value of the inflation factor  $c$ ).

### *Perform MCMC: generate the MCMC file*

Choose a new file for MCMC by selecting in the “**MCMC output file**” section “**Change file**”; we will use “unlinked\_test.xml”. The default run length is sufficient for this simple dataset so press “**Generate Sample**”. A progress box will appear; it takes roughly 1-2 minutes on a 2008 desktop PC. When completed, the file is automatically read and the pairwise coincidence shown in the main window.

### *Find optimal assignment: generate the tree file*

Finding an optimal assignment is performed simultaneously with the tree building algorithm.

Choose a new file for the tree under the “**Processed Tree file**” section by selecting “**Change file**”; we will use “unlinked\_test\_tree.xml”. Then press “**Generate tree**”. A (less useful) progress bar will appear, taking less than 1 minute to perform the computation. When completed, the main window will display the tree and reorder by the best population state, as well as switch to “Aggregated data” view.

### *Save the GUI state*

In order to not have to re-enter your favourite defaults and all the file names, you can now save the GUI's state. Under the “**Meta file**” section, choose a new file with the “**Change file**” button; we will use “unlinked\_test.fs”. Press the “**Write file**” button in this section to save the state. If you make changes to any of the display options, come back here and write the file again to save the updated settings. *Note: if you change the order that the tree displays, you also need to “Write Tree” in the “Processed tree file” section!*

We are now done with the file IO window; press “Done” at the top to close it.

### *Experiment with the view*

You might want to now “have a play” with the display options. The menus at the top are the place to start: try going to “View->Raw copy data” to see the data matrix ordered by the best state. “View->Pairwise coincidence” shows the average MCMC pairwise coincidence, which in this case shows partial separation between the two most difficult populations (IND41-60 and IND81-100).

Try closing everything down and **reloading the state of FineGui**; you can do this from the command line (see Command Line section) or by “**file->Manage Files**”, in the “**meta**” section choosing the file “unlinked\_test.fs” and pressing “**Read file**”.

Possibly useful exercises to familiarise yourself with the GUI: after looking at the manual, try repeating this analysis for the linked dataset provided in the examples directory. It could be helpful to try performing inference at the wrong value of  $c$ , and loading these results as the “second dataset”.

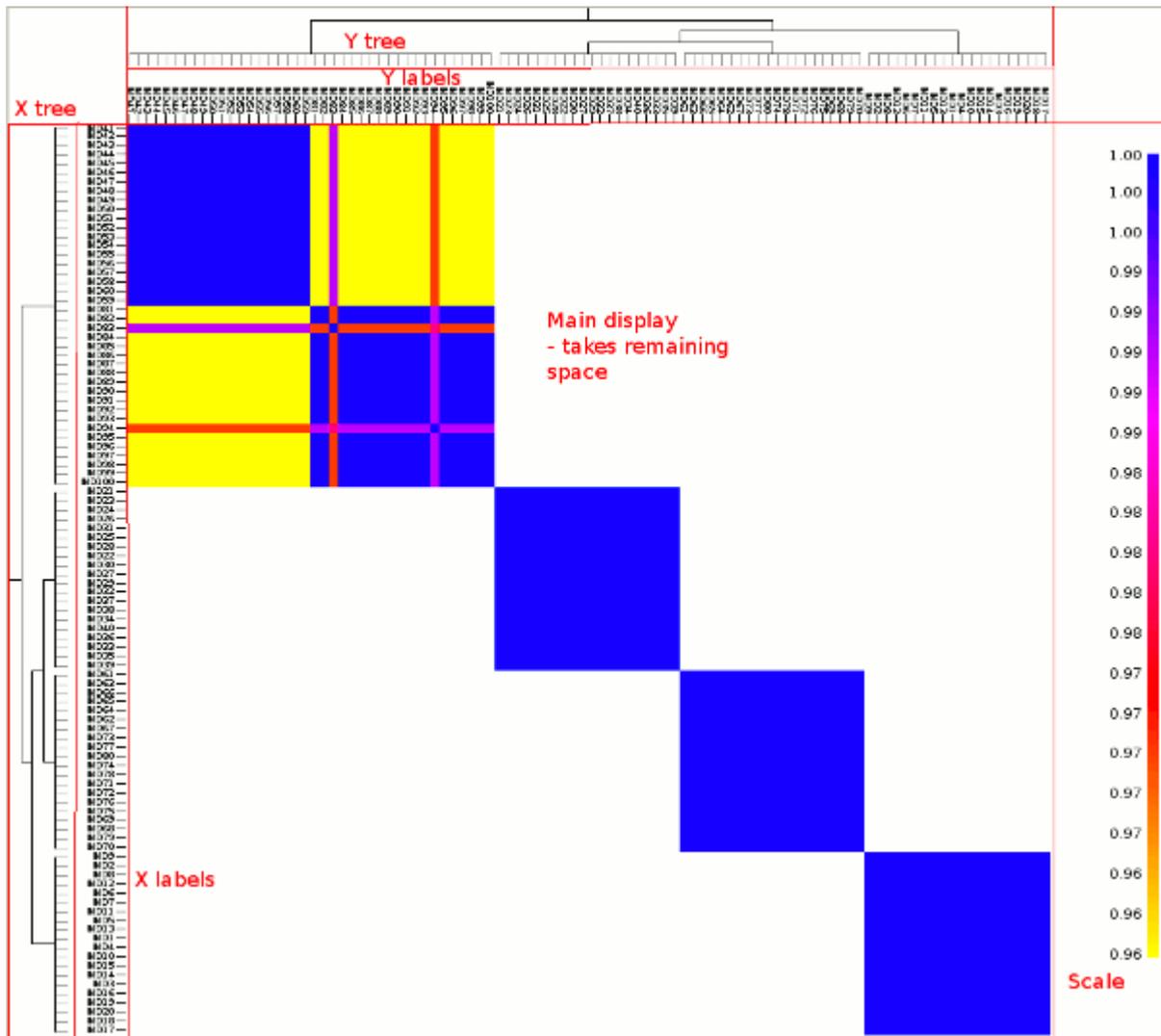
It is extremely good practice to **perform inference twice** at the same parameter values, and check for convergence both of the parameters (i.e. the MCMC traces), and population assignments (i.e. the pairwise coincidence matrix).

If you intend to use the **command line version** of Finestructure, try to recreate these results with the command line as described below, and read in the MCMC output file and the tree that you create.

## **Main Window**

The main window has some **menus at the top** (where everything is accessed from) and some **display options on the left**. It displays the **main matrix** as an “Individual by individual” figure, which can show a variety of matrices. Which matrix is shown is chosen from the top, and the optional tree/labels/scale, the dimensions, and other features are chosen from the left.

## Screen Layout



Example screen layout, showing how the sections make up the display.

Note that screen is made up of several (removable) sections of various size. The user fixes the overall dimensions, and the tree, label and scale sizes, with the display taking up the remainder of the space. The image is shown in real size, so may not fit in a window. "Screen real estate" is limited so use a small image for exploring data, but remember to make the image large before exporting it. Try to choose the size of components to keep the "main display" square.

*Note: you can disable the display to the screen using the "Display" checkbox in the main window. This is helpful if operations are taking a long time to perform. You can still export to an image, which can be quicker and easier to view in very large datasets. The "aggregated (alternative)" View option is significantly faster to display and is useful for getting populations named correctly and trees ordered correctly in such datasets.*

## Tree order manipulation

The "tree" is interactive. Try clicking a branch to swap the order they are sorted in. Try clicking an end branch to pop up a window allowing you to sort individuals within a population. **Individuals are moved** by either: a) pressing "Control and <mouse scroll wheel up/down>", b) "Control and <keypad up/down>" then pressing return when the location is correct, or c) using the "Up/Down" buttons.

## Menu options

- **File->New Session:** clears the current data files (keeping incidental settings).
- **File->Manage Files:** The main input and output window; see “Input Data” section.
- **File->Manage Second Dataset:** Related to the “manage files” option, this allows a second data set to be read for comparison purposes (see “view...”).
- **File->Export to image:** export the current display to a bitmap image of various formats. Currently supported are BMP, PNG or JPG.
- **File->Quit:** Exit without saving.
- **Organize->Rename individuals:** Allows each individual to be given a new name. Enter new names in the second column and press Accept to apply.
- **Organize->Rename populations:** Allows each population to be given a name; default is Pop<i>. Enter new names in the second column and press Accept to apply; individuals in each population are shown.
- **Organize->Reorder...:** change the order of populations (for PCA labelling purposes). Populations are moved by either: a) pressing “Control and <mouse scroll wheel up/down>”, b) “Control and <keypad up/down>” then pressing return when the location is correct, or c) using the “Up/Down” buttons.
- **Organize->Classify individuals:** Classify each individual, by e.g. ethnicity, sample location, etc. Either a) Press “guess” which assigns labels assuming that individuals are labelled <population><samplenum> (e.g Basque1,Basque2,French1,French2), or b) enter a label for each individual. Colours are assigned randomly for each unique label. Press “New Colours” to re-choose at random, or click on a colour to replace it via your operating system's native colour chooser.
- **Organize->Change Colour Scale:** This allows you to choose how the colours are displayed. The scale is often not normal so it is helpful to change the relative “length” each colour is merged into the next over. You should be able to figure out how to use this from the hints and by comparing the “Simple” and “High Contrast” colour schemes provided.
- **View->Raw copy data:** View the raw copy data as the main display.
- **View->Aggregated copy data:** View the mean of the raw copy data between populations (available if a tree file is used).
- **View->Aggregated (Alternative):** This views the aggregated data in a population-by-population way, rather than individual-by-individual. Although less flexible, this is significantly faster to work with. Note that you cannot use an alternative diagonal in this mode.
- **View->Pairwise coincidence:** View the pairwise coincidence in the MCMC sample (available if an MCMC sample is used).
- **Second View->Enable alternative second view:** allows the top right to display something different to the bottom left; they will no longer be synchronised once a second view is chosen. Useful to show raw vs aggregated copy data.
- **Second View->Use second dataset:** the data shown in the top right will be drawn from the dataset loaded from file->Manage second dataset. Useful to compare pairwise coincidences.
- **Second View->Raw copy data:** as above.
- **Second View->Aggregated copy data:** as above.

- **Second View->Pairwise coincidence:** as above.
- **Plot->MCMC traces:** Shows the traces for both the main and the second data set. Useful for establishing convergence. See “MCMC traces” section.
- **Plot->Principal Components Analysis:** Performs PCA on the data; useful for visualisation of population assignment decisions. See “PCA” section.

## Left Panel options

- **Display:** Toggle to enable/disable drawing to the screen. This is useful because when the screen is enabled, all changes cause a possibly slow refresh. Disable it to set things up as you like and then re-enable it (or use the export function without re-enabling).
- **Show X labels?** Disables the displaying of the X labels. (default: enabled with data)
- **Show Y labels?** Disables the displaying of the Y labels. (default: enabled with data)
- **Show X tree?** Disables the displaying of the X tree. (default: enabled with tree loaded)
- **Show Y tree?** Disables the displaying of the Y tree. (default: enabled with tree loaded)
- **Show scale?** Disables the scale bar. (default: enabled)
- **Dimensions:**  $\langle x \rangle$  by  $\langle y \rangle$ . The number of pixels ( $\langle width \rangle$  by  $\langle height \rangle$ ) for the entire display.
- **Tree size:**  $\langle X \rangle$ ,  $\langle Y \rangle$ . The number of pixels for the tree (if enabled).
- **Tree width**  $\langle x \rangle$ . The thickness of pen used to draw the tree, in pixels.
- **Labels size:**  $\langle X \rangle$ ,  $\langle Y \rangle$ . The number of pixels for the labels (if enabled).
- **Label size**  $\langle x \rangle$ : the size of the labels. Negative values are relative to the maximum that will fit in a single individual (i.e. 1). Larger values are useful for population labels. Positive values allow you to specify the size in points.
- **Population Labels?** Whether to use population labels instead of individual labels.
- **Perpendicular?** Whether to plot population labels perpendicular to the population.
- **Label classification?** Whether to show boxes colour coded by classification instead of small tick marks by labels.
- **Label box size**  $\langle x \rangle$ : the size of ticks/boxes for classification labels, in pixels.
- **Scale size**  $\langle x \rangle$ : the width of the scale bar box, in pixels (if shown).
- **Scale bar size**  $\langle x \rangle$ : the width of the coloured part of the scale, in pixels.
- **Scale text size**  $\langle x \rangle$ : the size of the scale text, in points.
- **Scale format**  $\langle s \rangle$ : the format for the scale to be displayed in, using c++ “printf” format. Try “%0. $\langle n \rangle$ f” for n significant figures.
- **Scale min**  $\langle x \rangle$ , **Scale max**  $\langle x \rangle$ : the range that the scale will start and end at. Lower values will be white and higher values will be black.
- **Rescale continents?** If using a “fixed file” to fix some individuals into a population, should the scale for each of these be renormalised? (useful for contrast).
- **Continent Rows?** Should “Y”, i.e. continent rows be shown?
- **Continent Cols?** Should “X”, i.e. continent columns be shown?

- **Continent Size <x>**: The relative width of a continent row/column compared to ordinary individuals.
- **Population scale <x>**: When in the aggregated (alternative) view mode, you can rescale population sizes by this power. 1 means no scaling, 0 means all populations are given the same area, 0.5 means they are square-rooted, etc.

## Manage Files

The screenshot shows the 'Input Finestructure data' window with the following sections and controls:

- Meta file section:** Includes a 'Meta file' checkbox (checked), a 'Change file' button, and a text field for 'Filename: unlinked\_test.fs'. Below are 'Read file' and 'Write file' buttons.
- Raw data file section:** Includes a 'Raw data file' checkbox (checked), a 'Use fixed' checkbox (unchecked), and a 'Change fixed file' button. Below are 'Change data file' and 'Read data file' buttons, and a text field for 'Filename: data\_unlinked.chunkcounts.out'. There are also 'Skip lines: 0' and 'Inflation 'c': 1.84071' fields, and checkboxes for 'X labels present?' and 'Y labels present?'.
- MCMC output file section:** Includes an 'MCMC output file' checkbox (checked), a 'Change file' button, and a 'Help on MCMC options' button. Below are 'Generate sample' and 'Read Pairwise Coincidence' buttons, and text fields for 'Burnin: 10000', 'Runtime: 10000', and 'Skip: 100'.
- Processed Tree file section:** Includes a 'Processed Tree file' checkbox (checked), a 'Change file' button, and a text field for 'Filename: unlinked\_test\_tree.xml'. Below are 'Generate tree' and 'Read Tree' buttons, and a dropdown menu for 'Type: Full Hill Climb'. There are also text fields for 'Max states per merge: 10000' and 'Hill climbing iterations: 10000'.

Annotations on the left side of the window include:

- Close the window (keeping changes)
- Hide "meta" section
- Change current meta file name
- Read from current meta file
- Hide "data" section
- Fix "some individuals as super individuals" . choose file name . current file name
- Change current data file name
- Read current data file
- Hide "MCMC" section
- Hide "Tree" section
- Change Tree file name
- Generates a tree
- Overwrites current tree file
- Read a tree
- Write current tree to file
- This includes individual order within and between populations so write the tree file after any changes

Annotations on the right side of the window include:

- Status of current activity
- Help screen
- Current meta file name (Editable)
- Write to current meta file
- Current data file name
- Number of lines to skip before reading
- Does the data file have individual labels as the first row? ... first column?
- Current tree file name
- Limit to the number of comparisons per merge- there are  $N(N-1)/2$  possible merges
- "Greedy" hill climbing steps (Full Hill Climb only)

Text at the bottom of the window: Type of tree: "Full.." is the most robust but slowest. "Observed" respects the MAP posterior state, "Merge" only allows merge moves from the observed MAP state

Accessed via **File->Manage Files**, this is the second most important window.

Broadly, there are 4 types of files:

- **Meta file:** The file that finestructure GUI keeps all of its information in. You can save/load (almost) all settings you might change, and restore simply. The tree, if present, is *not* saved here and so changes to the tree require saving the tree *as well*. **Files are stored as absolute path names** which might make moving ".fs" files between PCs tricky however, you can edit these by hand to make them transferrable. (Use a dot for the current directory, etc).
- **Raw data file:** Finestructure needs a raw data file to perform any actions. The data file currently can contain an arbitrary length header which will be ignored, and optional row and column names. Separation by tabs, spaces or commas should work. The "fixed file", if used, must match the one used for MCMC sample generation. The first line is a comment line containing the c value (if you used ChromoCombine to make your ".out" file).
- **MCMC output file:** Finestructure GUI can generate this, or you can use the command line version to generate it on a dedicated compute cluster. The main visualisation tool for this is the pairwise coincidence, i.e. the average number of MCMC samples two individuals are placed together in.
- **Processed Tree file:** Finestructure makes sense of MCMC samples by creating a Maximum

Aposteriori (MAP) set of populations (and parameters) called a state. This state is still too complex so we impose a tree structure on it to find similar populations. This file contains the MAP state and tree, which is helpful for visualisation.

## MCMC Input/Output/Generation

From the MCMC Help Button:

1. If you have already generated an MCMC file, e.g. using the fineSTRUCTURE command line, then 'Change File' to this file and press 'Read Pairwise Coincidence'. This is all you need to do. For more help on how to use the fineSTRUCTURE command line, see the manual (later).
2. If you want to generate an MCMC sample, think about your data. How many individuals do you have? If it is more than a few hundred, you will need to run fineSTRUCTURE for a long time (days).
  - a) You should not have to worry about "c" factors if you used ChromoCombine to generate your data. You can however change this in the data section if you want to e.g. explore how it affects inference.
  - b) For small easy datasets (100 individuals or less), you should set burnin and runtime to around 100000; 10x longer if the populations are not well defined. Larger datasets (e.g. 1000 individuals) have been successfully mixed in 10,000,000 iterations (each for burnin and runtime). You probably want S=1000 MCMC samples for a publication and S=100 for exploration; set the skip so that S='runtime/skip'.
  - c) When you are done, press 'Generate Sample'. If you aren't sure if the MCMC has mixed, you should generate a second dataset with identical parameters (Main Window->File->Second Dataset), and display it on the second diagonal (Main Window->Second View->Enable Alternative Diagonal View, and then Main Window->Second View->Use Second Dataset).

## MCMC Traces

Accessed via **plot->MCMC Traces**, but be warned: the MCMC Traces window is currently incomplete!

It can show any of the parameter values (selected from the "Plot Type" menu), and export the data either as an image (PNG, BMP or JPG) or as a CSV file for analysis in another program, for example the statistical software R.

In the future, standard MCMC convergence diagnostics will be added here.

## Principal Components Analysis, PCA

Accessed via **plot->Principal Components Analysis**, but be warned: the Principal Components Analysis window is currently incomplete!

The PCA is done by

1. Setting the diagonal (which is zero by construction) to be the column mean (excluding the diagonal),  $X_{i,i} = \sum_{j=1, j \neq i}^N X_{i,j} / (N-1)$ .
2. Subtracting the column mean from each column to zero mean the matrix,  $X_{i,j} = X_{i,j} - X_{i,i}$ .

3. Considering the Eigenvalue composition of the symmetric matrix  $X_{i,j} \times X_{i,j}^T$ .

Eigenvectors are sorted by eigenvalue in decreasing order. Currently you can select which component to place on the X and Y axis, choose the location of the legend, the legend size and point size. Exporting to CSV creates a matrix with row and column names, with a leading row containing the eigenvalue.

If you want to classify individuals by something other than their population assignments, this can be done by creating a modified Tree/population assignment file. Create a fake “MCMC output file” that has only 1 iteration, and the population structure you want (see below “Output File Format”). Then read it in as an MCMC file, create a Tree file using “Observed” rather than “Full Hill Climb”, and a correct tree will be created. This can also be useful for comparing population assignments to what was “expected” based on labels or geography.

## Command Line

The finestructure GUI can be called to open a particular finestructure meta file with:

```
> finegui <metafile.fs>
```

It is also possible to specify basic options on the command line, to make reading in files less arduous. See the internal help for details:

```
> finegui -h
```

One particularly useful option is to read in some metafile with the display disabled. This is done with the “-d” flag:

```
> finegui -d <other arguments>
```

The **finestructure command line** has a lot more flexibility than the GUI in terms of modelling. It can use a wider variety of models and create suitable MCMC and tree files for the GUI to read. The helpfile can be accessed by running finestructure with no options, or the -h flag:

```
> finestructure -h
```

This is the most complete place for help. Many of the options available are not recommended for use on real data, and the defaults are for the large part, correct. To **perform the MCMC analysis** performed by the GUI, do:

```
> finestructure -X -Y -i <skiplines> -F <forcefile> -c <cval> -x <burnin> -y  
<mcmcsteps> -z <mcmcskip> <rawdatafile> <mcmcoutfile>
```

Where these the parameters are identical to their GUI counterparts. The “-F <forcefile>” and “-i <skiplines>” are not necessary if you don't use a force file and you don't have any header to skip. The “-X -Y” tell finestructure that the datafile has row and column headers, but are usually not necessary because it can figure out these values (under most circumstances).

To **create a tree** as the GUI does:

```
> finestructure -X -Y -i <skiplines> -F <forcefile> -c <cval> -x  
<hillclimbingiterations> -m T -t <maxstatesmerge> <rawdatafile> <mcmcoutfile>  
<treefile>
```

Note the additional “-m T -t <..>”, and the **modified meaning** for “-x”. Additional features you may be interested in are:

- -s <seed>: Set the random number seed, for repeatability.
- Running with <rawdatafile> <oldmcmcoutfile> <newmcmcoutfile> to continue a run e.g. that wasn't long enough.
- -I <option>: changing the initial conditions, to make sure that the MCMC convergence does not depend on them.

- -K: fix the number of populations to the initial value specified by “-I”. Note that the tree will give a more interpretable answer than successive runs at different K!
- -a <num>, -b <vector>: changing the hyperparameters, to make sure that the populations inferred are not sensitive to these defaults. You can often change these by several orders of magnitude and see no difference in the observed populations.
- -M <modeltype>: For truly unlinked loci only (i.e. simple simulated data), with a large number of individuals, artefacts of the data construction algorithm can create problems with “banding”, i.e. an individual receiving a lot of copies from another individual by chance (but more than we allow for). This is due to anti-correlations caused by sharing a single rare SNP. The solution is to remove the information about the relative copying rates. The Normalized version of the model does this. (This model is less powerful than the standard finestructure algorithm).
- -e <extract>: allows extraction of some features from a MCMC output file. This is useful if you want to do your analysis in another program. Useful for extracting marginal likelihoods, thinning the samples, getting the realised coancestry matrix, etc. Note that you still have to go into the mcmc output file for many features.
- -k <treetype>: Different tree types. See below.

## Continents, super-individuals and force files

The Finestructure algorithm motivates two reasons to not simply run the algorithm once on the whole dataset:

1. The CPU effort required is approximately  $O(N)$  per iteration, and the number of iterations required is  $O(N)$ , making the algorithm  $O(N^2)$ . Roughly 1000 individuals (with 'nice' but detailed population structure, e.g. HGDP) take roughly 1 week to mix well; more is testing the limits. However, 3000 individuals with a low number of populations still runs well.
2. The prior assumes all populations are equally distant, which is not true. Many populations are very similar to each other and very different to some others. This means that we do not identify all substructure in one go (though we find all the large substructure).

To address these points, Finestructure uses the concept of “**Super-individuals**”, who look like (reweighted) normal individuals, but cannot be split and do not contribute to parameter inference. This allows them to be included in the algorithm without additional computational cost.

Additionally, finestructure can have a concept of “**continents**”, which are just like super-individuals except that they are ignored by the tree building algorithm. Both continents and super-individuals exist primarily to provide chunks copied to (and from) the remaining population.

A force-file may contain a combination of super-individuals and continents. Each is defined on a single line using the following format:

```
# comment lines start with #.
# Define a “super individual”
<superindname>(<ind1>,<ind2>,...<indN>)
# Define a “continent”
* <continentname>(<ind1>,<ind2>,...<indN>)
# any unmentioned individuals are treated normally.
#* <continentname>(<ind1>,<ind2>,...<indN>) is therefore ignored.
```

This uses our standard population format of a comma separated list, contained in brackets. The algorithm will see the individuals only in their merged form, and will display them as such in the output file. There is an example related to the tutorial in “example/data\_super.force” complete with command line generated MCMC file in “unlinked\_test\_F.xml” (see the comment in that file), as well as a tree for the same thing at “unlinked\_test\_tree.xml”. Interestingly, although the MCMC is

very similar for this and the tutorial, the MAP state is split correctly in this case but not split without using super-individuals. **Using super-individuals allows for greater power.**

## Tree types

For most purposes the default tree type is recommended. This ignores the order that merges happen in, giving the correct topology, but removing any additional information about which splits occurred when. (Recall the tree is built by finding the merge that least decreases the posterior probability). This is correct for display purposes, and also deliberately discourages over-interpretation of the tree (since it is somewhat affected by sample sizes).

If you would like to extract the complete tree, for example to look for large jumps in the posterior probability to identify natural clusterings at smaller K, then you can use the “-k” flag in the command-line version:

```
-k <num>          Change the tree building algorithm.  
0:               Discard all ordering and likelihood information  
(default).  
1:               Maintain ordering.  
2:               Maintain ordering and likelihood.
```

The “-k 2” option for example allows a complete description of the Posterior of the merged model at all (smaller) K, and you can use this to find sensible clusterings. It is likely to perform significantly better than running MCMC at a fixed K, because our “coancestry flattening” algorithm during tree building can partially account for presence of weak relatedness within the samples. It also ensures that results at different K are hierarchically related.

## Output file format

It is useful to understand the output file format if you wish to extract the results for processing, or for high quality graphical creation (since the GUI can not yet handle export to postscript, PDF or other vector graphics format). The format is our own XML style, which can therefore be read and processed by a variety of alternative programs. The basic structure is:

- <outputFile>
  - <header>
  - <comment>
  - <Iteration1>
  - <Iteration2>
  - ...
  - <IterationN>

Each iteration consists of the following:

- <Pop>: the current population sample, with each population consisting of (<ind1>,...<indNi>).
- <K>: the current number of populations.
- <parametername>: the current value of the parameters: the default model has alpha, beta, delta, F.
- <Acc<x>>: running acceptance rate for the move <x>, e.g. AccSAMS is the “split-or-merge” proposal, AccMS is the merge-and-split proposal, AccIndiv is the move single individual proposal, and AccHyper is the hyper-parameter updates.

- `<Number>`: the iteration number.

Don't forget that each tag is closed by `</tag>`, so e.g. the whole K tag is `<K>x</K>`, where x was the actual value of K. All tags are single line, with the exception of `<header>` and `<Iteration>`.

The tree output format is very similar, consisting of:

- `<outputFile>`
  - `<Iteration1>`
  - `<Tree>`

Here the Iteration may only contain the population, or may contain the MAP parameters, depending on how it was created. The `<Tree>` tag contains a newick format **binary** tree, with populations separated by a very tiny amount so that they appear to be multi-furcating. The GUI can reorder these trees so this should not be a problem, and allows them to be imported into a wider range of software.

Under Linux it is very easy to extract all of a particular tag. For example, the following code extracts all `<Pop>` tags, and removes the tag:

```
> grep "<Pop>" <file> | sed 's#<Pop>##' | sed 's#</Pop>##'
```

Other tags can be extracted similarly. Under Windows, your best bet is to use a script ready environment such as R.