Inferring population structure and admixture histories (16/3/2023)

Leo Speidel

Contents

About this document: *This is an Rmarkdown notebook which includes* `code` *as well as explanation and generates the practical .html document. From this document you should be able to copy and paste the code and run it yourself. You can also use the* `.Rmd` *document opened in RStudio to extract and edit code. This practical is largely reworked from a practical and dataset kindly shared by Dr Garrett Hellenthal. All data you will need is present in the* `data` *folder with software available in the* `software` *folder. All required software has been installed on the HPC in /usr/local/bin except for GLOBETROTTER.R which is available under software/GLOBETROTTER.tgz. Relevant output is in* `output`*, though have a go running yourself first!*

Introduction

During this mornings lectures we discussed the determination of population structure using genome-wide SNP data. We discussed allele frequency-based clustering, for example using ADMIXTURE, as well as the power that can be gained when incorporating haplotype information. We discussed chromosome painting as a way of identifying haplotype sharing patterns between donors and recipients in populations. Clustering based on haplotype sharing can often yield much finer scale resolution of population structure, for example in the British Isles. Patterns of LD decay can also be used to identify the timings and possible sources of admixture in populations.

For this practical, we will be applying the statistical software ADMIXTURE, CHROMOPAINTER and fineSTRUCTURE to cluster (real and simulated) individuals based on genetic similarity. We will be using a dataset explored in Hellenthal et al 2014, which is freely available and consists of data from the Human Genome Diversity Panel and other resources. The SNPs were ascertained using Illumina chip technology; here we will work only with chromosome 22, which has 6,812 SNPs.

For this practical, we will further only use the following populations:

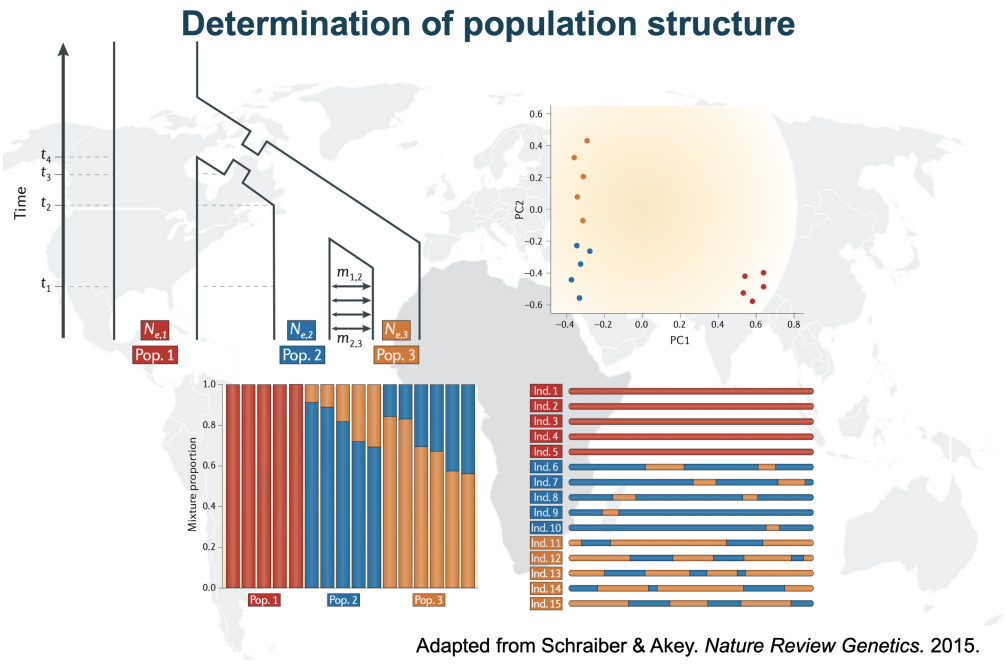


Figure 1: **Determination of population structure**

Population	Country	Region	number of individuals
Balochi	Pakistan	Central South Asia	21
BantuKenya	Kenya	Africa	11
BantuSouthAfrica	South Africa	Africa	8
Burusho	Pakistan	Central South Asia	25
English	Britain	Europe	6
HanNchina	China	East Asia	10
Kalash	Pakistan	Central South Asia	23
Makrani	Pakistan	Central South Asia	22
Mandenka	Senegal	Africa	22
MbutiPygmy	Congo	Africa	13
Mongola	Mongolia	East Asia	10
NorthItalian	Italy	Europe	12
Orcadian	Britain	Europe	15
Pathan	Pakistan	Central South Asia	22
Sardinian	Italy	Europe	28
Tuscan	Italy	Europe	8
Total			256

Added to the above is also a **simulated population** consisting of 20 individuals simulated as descendents of an admixture event occurring **30** generations ago, where 80% of the DNA was contributed from present-day **Brahui** individuals (from Pakistan, Central South Asia) and the remaining 20% from present-day **Yoruba** individuals (from Nigeria, Africa). This simulation is from Hellenthal et al. 2014 (see figure below) and is the example file included with CHROMOPAINTER and mentioned in the lecture.

By the end of the practical, you should be able to:

- Run an ADMIXTURE clustering analysis
- Run Chromopainter
- Run fineStructure

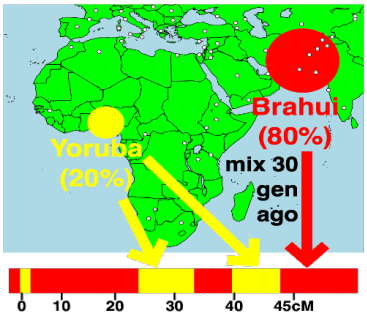


Figure 2: **Simulated population**

- Run GLOBETROTTER

Additionally, throughout this practical, there are questions about the material for you to have a think about.

Allele frequency based clustering in ADMIXTURE

First we will apply ADMIXTURE to these data. The programs we use should have been installed in `/usr/local/bin` on the workshop HPC or you can save admixture linux-1.22.tar.gz available in the **software folder** to any given folder, and then unzip and extract files with:

```
tar -xzvf admixture_linux-1.22.tar.gz
```

We will cluster individuals running admixture with several numbers of clusters K. The command to run for $K = 2$ clusters:

```
admixture BrahuiYorubaSimulationChrom22.admixture.geno 2
```

Here **BrahuiYorubaSimulationChrom22.admixture.geno** contains the genotypes of each individual. This is for all 6,793 (non-monomorphic) SNPs, though it is often recommended to thin SNPs based on linkage disequilibrium (LD) prior to running admixture (or principal components analysis). But that would leave too few SNPs when considering only a single chromosome, so we will ignore this advice here.

Question: Why should you LD prune using allele frequency based methods to detect population structure?

Two output files will be made:

- **BrahuiYorubaSimulationChrom22.admixture.2.Q**, which gives the probability each individual is assigned to each of the K clusters
- **BrahuiYorubaSimulationChrom22.admixture.2.P**, which gives each SNP's allele frequencies for each of the K clusters.

We can have a quick look at the **BrahuiYorubaSimulationChrom22.admixture.2.Q** file:

```
qfile <- read.table('output/BrahuiYorubaSimulationChrom22.admixture.2.Q',as.is=T,header=F)
```

```
head(qfile)
```

```
##         V1       V2
## 1 0.037718 0.962282
## 2 0.017349 0.982651
## 3 0.064632 0.935368
## 4 0.007161 0.992839
## 5 0.059206 0.940794
## 6 0.000010 0.999990
```

Lets now run this for `K=3,4,5,6,7,8` using the above command. Have a go, all the output is in the `output` folder.

ADMIXTURE results are typically presented as barplots ordered by population. There are many ways of generating these plots. You'll find a simple R script in the `scripts` folder **ADMIXTUREBarplots.R**.

Let's run this script to plot the ADMIXTURE inferred proportions:

```
R CMD BATCH ADMIXTUREBarplots.R
```

As you can see from the top of the R code, this program makes a file called **BrahuiYorubaSimulationChrom22ADMIXTURE.pdf**.

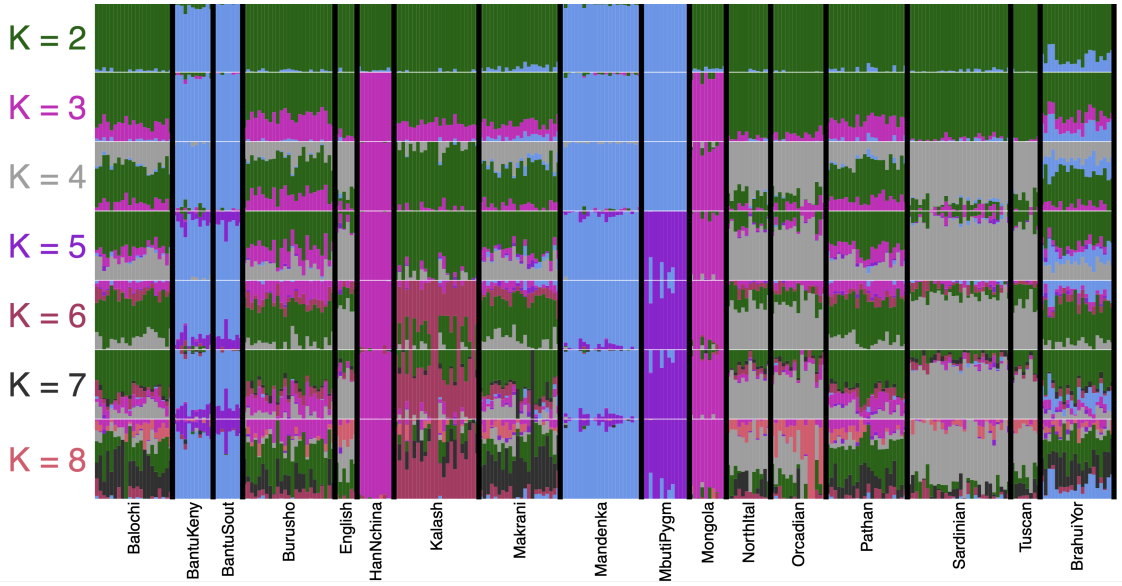


Figure 3: **ADMIXTURE plot**

Question: How do you interpret these findings? What populations "emerge" as you increase K at each step? What about the simulated population?

This is generally step one in an ADMIXTURE analysis. Note you can also run cross-validation (`--cv`) to help infer the **best** value of K. And you can run bootstrap re-sampling (`-B`) to infer standard errors around estimates.

Chromosome painting using CHROMOPAINTER

Next we will apply CHROMOPAINTER to the same dataset. **Chromopainterv2** should be installed on the cluster or you can save ChromoPainterv2.tar.gz from the `software` folder and the file `BrahuiYorubaSimulation.poplistReduced.txt` available in the `data` folder to any given location, and then unzip and extract files with:

```
tar -xzvf ChromoPainterv2.tar.gz
```

Followed by compilation with:

```
gcc -o ChromoPainterv2 ChromoPainterv2.c -lm -lz
```

ChromoPainterv2 can be run in lots of different ways. At this point in the practical we are interested in using the ChromoPainterv2 output to cluster individuals in our dataset into genetically homogeneous groups using fineSTRUCTURE.

The recommended way of doing this is described in Section 8.1 of ChromoPainterv2Instructions.pdf which is included in the `ChromoPainterv2.tar.gz` folder. The three steps we outline:

1. First run Chromopainterv2:
    a) Use Expectation-Maximization (EM) to estimate the **switch** and **mutation (emission)** rates running ChromoPainterv2 on a subset of the data (i.e. only a subset of individuals and chromosomes). EM values are stored in the **EMprobs.out** file.
    b) Using the estimated parameters from (a), run ChromoPainterv2 again on **all** individuals and chromosomes.
2. Second, sum the ChromoPainterv2 output files from step 1(b) across chromosomes: XX.chunkcounts.out, XX.regionchunkcounts.out, XX.regionsquaredchunkcounts.out.
3. Third, run fineSTRUCTURE (a bit later in the practical)

We will skip step 1(a) here, and instead use default values. (In most applications, step 1(a) probably will not make much or any difference, but it is good practice!). We will also skip step 2, as we are only running this on chromosome 22 for illustration.

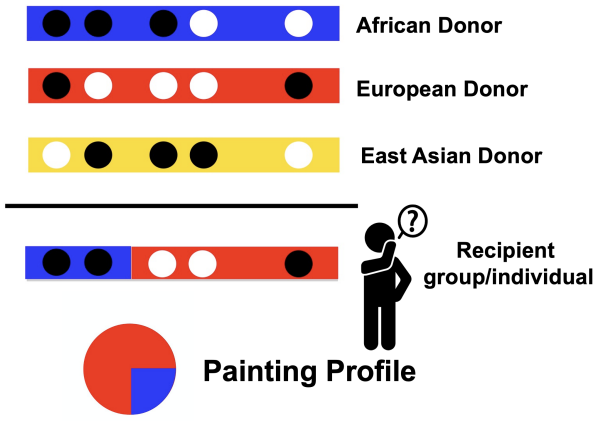


Figure 4: **Chromosome Painting**

We will do a Chromopainter run on our data. The files you need are:

- phased haplotype (*-g*) input file `data/BrahuiYorubaSimulationChrom22.haplotypes`
- recombination rate (*-r*) input file - a header line followed by one line for each SNP in haplotype infile. The first column denotes the basepair position values given in the haplotype infile, in the same order. The second column should give the genetic distance per basepair between the SNP at the position in the first column of the same row and the SNP at the position in the first column of the subsequent row `data/BrahuiYorubaSimulationChrom22.recomrates`
- label file (*-t*) - provides the population and identifier labels for each individual in haplotype infile `data/BrahuiYorubaSimulation.idfile.txt`
- population file (*-f*) - the file population list infile provides information on the populations to be used as donors and/or recipients when running ChromoPainterv2.

You can find out more information on the format of these files in the ChromoPainterv2Instructions.pdf instructions manual. For this dataset, we paint each individual in `data/BrahuiYorubaSimulation.idfile.txt` that is from a population in `data/BrahuiYorubaSimulation.poplistReduced.txt`, using every other individual from these populations as donors. We do this with the Chromopainter *-a 0 0* switch.

The command to run:

```
ChromoPainterv2 -g data/BrahuiYorubaSimulationChrom22.haplotypes \
-r data/BrahuiYorubaSimulationChrom22.recomrates \
-t data/BrahuiYorubaSimulation.idfile.txt \
-f data/BrahuiYorubaSimulation.poplistReduced.txt 0 0 \
-o output/BrahuiYorubaSimulationAllVersusAllChrom22 -a 0 0
```

This may be too slow for the practical (takes ~25 minutes) so I have provided the output files in the `output` folder. We'll focus on two of the output files:

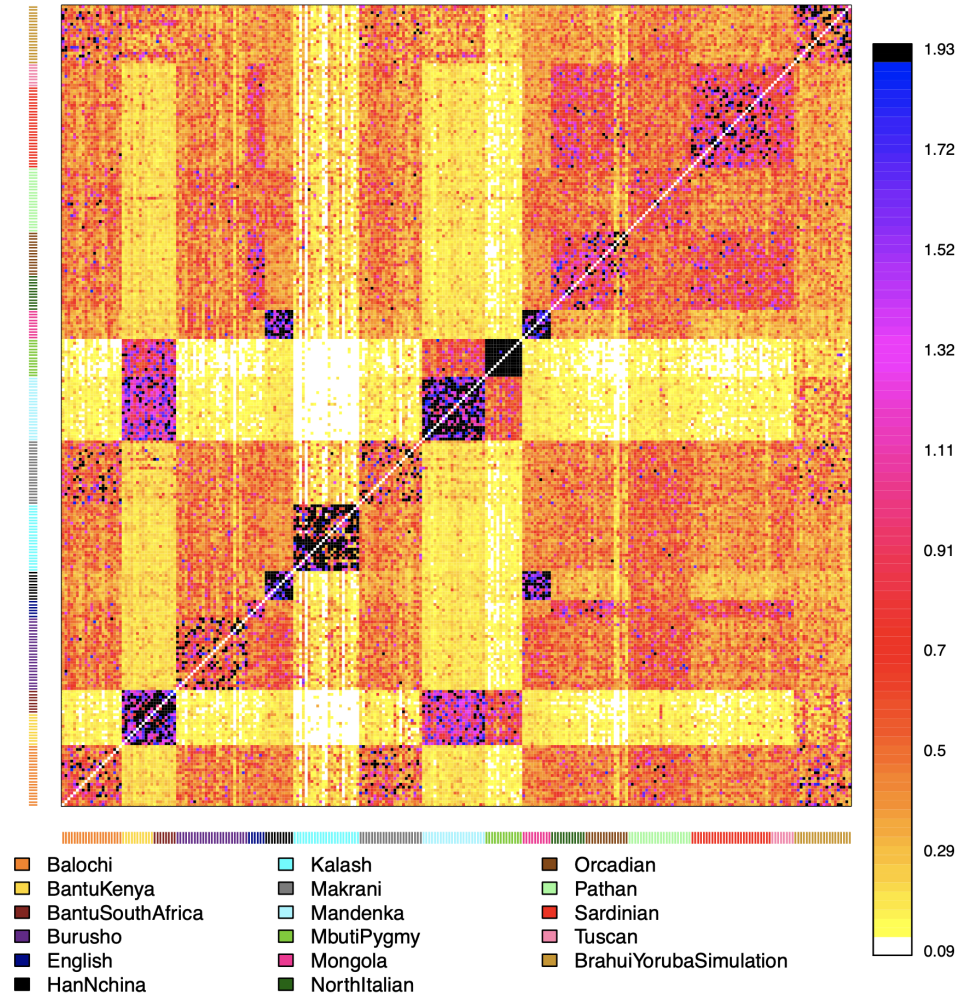
- `output/BrahuiYorubaSimulationAllVersusAllChrom22.chunklengths.out`
  - the total cM *length* of DNA segments (chunks) that each recipient individual (row) copies from each donor individual (column)
- `output/BrahuiYorubaSimulationAllVersusAllChrom22.chunkcounts.out`
  - the total *number* of DNA segments (chunks) that each recipient individual (row) copies from each donor individual (column)

Question: The lengths and counts of haplotypic chunks shared carry subtly different information. Discuss!

A convenient way to interpret these output files is by plotting them as a heatmap where the degree of colour provides the length/count of chunks shared between every combination of donor and recipient.

```
R CMD BATCH scripts/CHROMOPAINTERHeatMapPlot.R
```

This should generate a heatmap of the **chunklengths** output in the `/output` folder:



What does the output from this (i.e. BrahuiYorubaSimulationAllVersusAllChrom22HEATMAP.pdf) show (note recipients are

columns, donors are rows)? In particular answer the following questions:

Question: Which groups copy the most from within the same group and not from others?

Question: How does the simulated population BrahuiYorubaSimulation look different from the other groups?

Haplotype-based clustering using fineSTRUCTURE

Now let us use fineSTRUCTURE to cluster these individuals based on the ChromoPainter output. Again, fineSTRUCTURE should be installed on the HPC or you can download this from the `software` folder:

```
unzip fs_4.0.0.zip
```

We will use the pre-compiled binary **fs**.

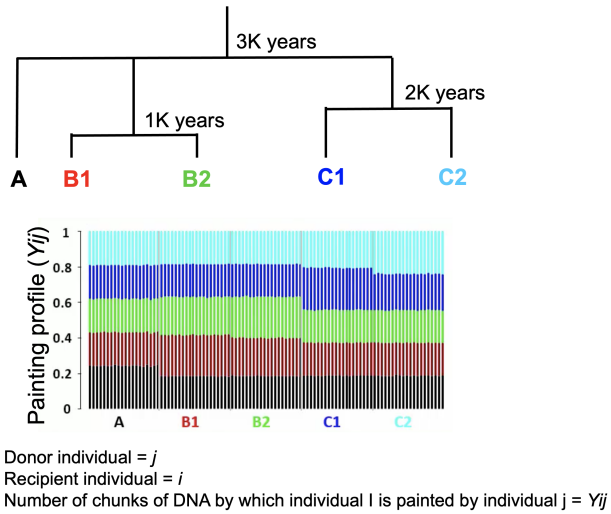


Figure 5: **fineSTRUCTURE**

We first need to calculate a nuisance parameter c, which takes into account the fact that not all entries in the coancestry matrix are actually independent, though the basic multinomial model of fineSTRUCTURE assumes they are (see Lawson et al 2012).

A simple way of doing this is by using the `calcC Continents.R` script in the `/scripts` folder which prints the c value to screen.

You can run using:

```
Rscript scripts/calcC_Continents.R output/BrahuiYorubaSimulationAllVersusAllChrom22
```

which should report a value of: **0.1844115577994475** which we will specify when running fineSTRUCTURE.

The first step of fineSTRUCTURE is to perform the MCMC run which uses the *chunkcounts.out* file to assign individuals into clusters.

Important parameters are:

- -x the number of burn-in iterations for the MCMC
- -y the number of MCMC iterations
- -z the thin interval for the output file

To run the MCMC step on our data we use:

```
fs finestructure \
-I 1 -c 0.184411557794475 -x 10000 -y 20000 \
-z 100 output/BrahuiYorubaSimulationAllVersusAllChrom22.chunkcounts.out \
output/BrahuiYorubaSimulationAllVersusAllChrom22.finestructure.out
```

In real applications, you should probably have each of *-x*, *-y*, *-z* a factor of ~100 higher. However, for reasons of time we use these values here.

Note that in the above example *-I 1* specifies that you want to start with all individuals assigned to one cluster, then split from there. The step above generates MCMC samples that assign individuals to clusters.

The second step is to apply the fineSTRUCTURE tree building step. This uses the output from the MCMC (`output/BrahuiYorubaSimulationAllVersusAllChrom22.finestructure.out`) to obtain a tree relating populations as well as best population assignments.

Important parameters are:

- *-x* the number of hill-climbing iterations for the tree buildling method
- *-t* the the maximum number of tree comparisons for splitting and merging

To run the tree-building step on our data we use:

```
fs finestructure \
-c 0.184411557794475 -x 10000 -k 2 -m T -t 1000000 \
output/BrahuiYorubaSimulationAllVersusAllChrom22.chunkcounts.out \
output/BrahuiYorubaSimulationAllVersusAllChrom22.finestructure.out \
output/BrahuiYorubaSimulationAllVersusAllChrom22.finestructureTREE.out
```

Similar to above, in real applications you should probably have *-x* a factor of 10 higher. This is the number of additional hill-climbing iterations to perform before build- ing the tree. *-m T* specifies you want to perform tree-building, rather than clustering, so that the tree file output will go into the third listed file: `output/BrahuiYorubaSimulationAllVersusAllChrom22.finestructureTREE.out`. Finally *-t 1000000* specifies that, at each level of the tree, you will compare 1000000 pairs of current populations to identify which pair to merge.

Now fineSTRUCTURE has finished clustering, look at the resulting clusters and tree again using `CHROMOPAINTERHeatMapPlot.R` in the 'scripts/ folder, but changing plot.tree at the top of the program to **yes** before running as before.

This will make a new file called `BrahuiYorubaSimulationAllVersusAllChrom22HEATMAPWithTree.pdf`, which should look something like (this is a stochastic process) the below:

What do you see? In particular answer the following questions:

Question: Do the inferred clusters seem sensible?

Question: How do the clusters compare to those when using ADMIXTURE?

Question: Does the inferred tree seem sensible?

Question: What about the simulated population?

`**If time**`, perform a second fineSTRUCTURE run and tree build. You can do this by specifying a new seed with e.g. *-s 2* and changing the output name. How do results change? (Note example runs are provided in the `output` folder.)

Note that fineSTRUCTURE also provides a function to calculate the proportion of MCMC samples for which every pair of individuals are assigned to the same cluster, by typing:

```
fs finestructure -c 0.184411557794475 -e meancoincidence \
output/BrahuiYorubaSimulationAllVersusAllChrom22.chunkcounts.out \
output/BrahuiYorubaSimulationAllVersusAllChrom22.finestructure.out \
```

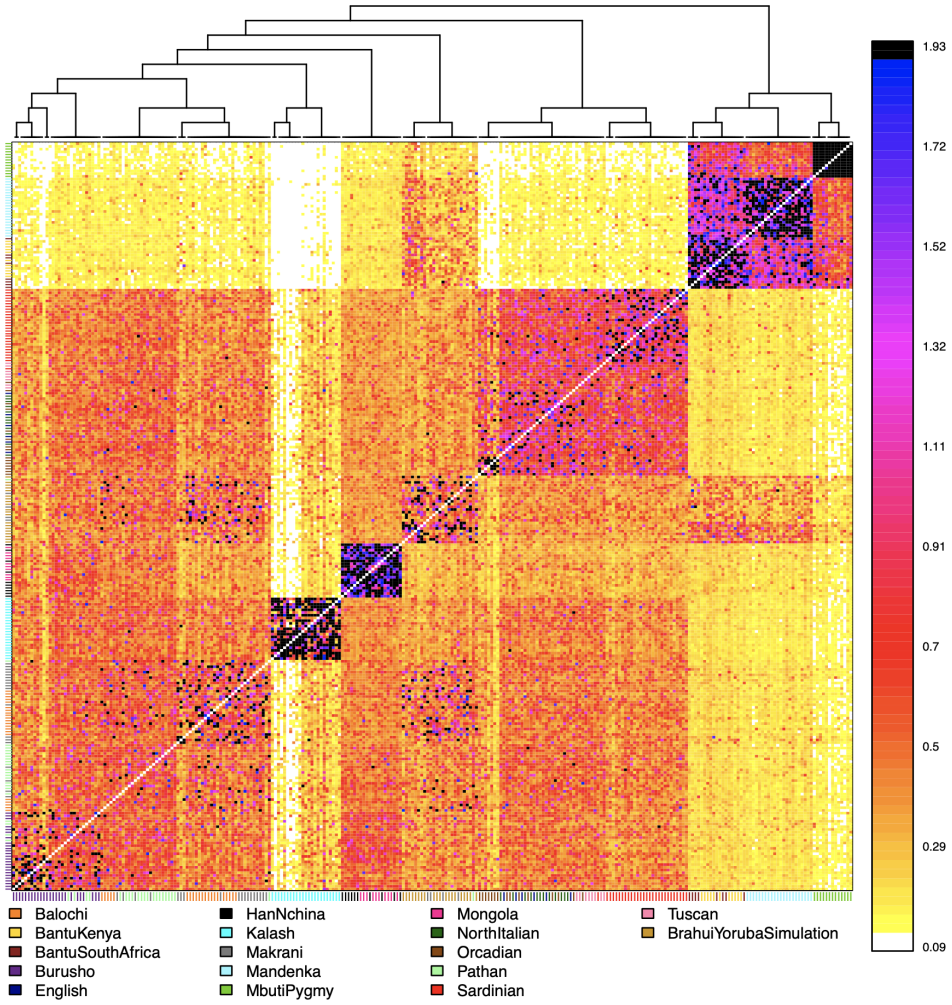


Figure 6: **fineSTRUCTURE heatmap**

`output/BrahuiYorubaSimulationAllVersusAllChrom22.finestructureCOINCIDENCE.out`

You can apply the above to both of the trees you have made (saving the second one as BrahuiYorubaSimulationAllVersusAllChrom22.finestructureCOINCIDENCE2.out).

Then you can use `FineStructureCoincidenceMatrixVisualize2Seeds.R` in the `scripts/` folder to plot the pairwise coincidence matrices for both of these two runs.

This will produce the file `BrahuiYorubaSimulationAllVersusAllChrom22FSCoincidencePlot.pdf` that shows the proportion of MCMC runs for which each pair of individuals is clustered together, for the first finestructure run (top left triangle) and the second finestructure run (bottom right triangle), with individuals ordered along the axes according to the inferred finestructure tree from the first run (i.e. ordered as in your heatmap plot for the first finestructure run). Here is an example if you don't have time to run:

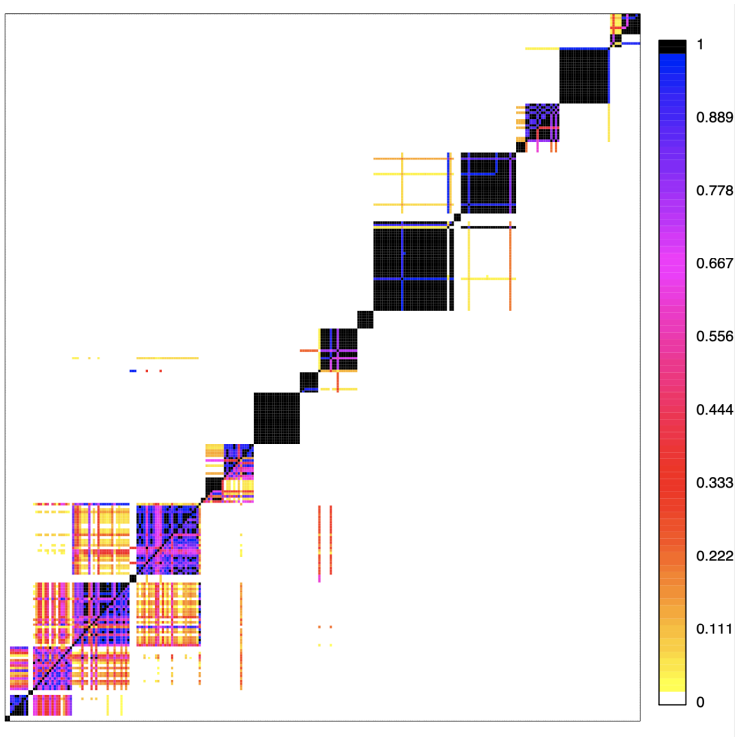


Figure 7: **Coincidence heatmap**

Question: How consistent do the results from the two runs appear to be?

Inferring admixture using GLOBETROTTER

We will use the same dataset to test for sources and timing of admixture using GLOBETROTTER. The aim is to see how well GLOBETROTTER can reconstruct an admixture event in the simulated *population* (80% Brahui and 20% Yoruba mixing 30 generations ago).

To do we first run a similar implementation of Chromopainter as earlier in the practical:

```
ChromoPainterv2 -g data/BrahuiYorubaSimulationChrom22.haplotypes \
-r data/BrahuiYorubaSimulationChrom22.recomrates \
-t data/BrahuiYorubaSimulation.idfile.txt`\
-f data/BrahuiYorubaSimulation.poplistReduced.txt 0 0 \
-o output/BrahuiYorubaSimulationGTAllVersusAllChrom22
```

though note that, importantly, this time there is no `-a 0 0`.

Instead, this command uses the file `BrahuiYorubaSimulation.poplistReduced.txt` to determine which populations to paint and which to use as donors for this painting. Have a look at BrahuiYorubaSimulation.poplistReduced.txt, you'll see that the population BrahuiYorubaSimulation is the only recipient population (R), while other populations are specified as donors (D).

This means each **BrahuiYorubaSimulation** individual will be painted using all individuals from all other listed populations as donors. This is the same as the painting we conducted previously, except we do NOT allow the BrahuiYorubaSimulation individuals to be painted using **themselves** as donors.

This is because we want to identify specific DNA segments inherited from admixing sources in order to generate the **coancestry curves** used to date admixture. When doing so, segments that **best match** to (i.e. are painted by) other BrahuiYorubaSimulation individuals would simply be discarded, because GLOBETROTTER does not allow a population to be an admixture source for itself, and so we would throw away information.

Next we will run GLOBETROTTER using the output of this painting.

You can find the source files under `software/GLOBETROTTER.tar.gz`. Please extract these locally as follows:

```
tar -xzvf software/GLOBETROTTER.tar.gz
```

Next compile with:

```
R CMD SHLIB -o GLOBETROTTERCompanion.so GLOBETROTTERCompanion.c -lz
```

To run GLOBETROTTER you have to specify three files:

1. The **parameter input file**, which describes all settings, including which population to detect admixture in and which populations to use as ancestry surrogates. The file we will use here is `data/BrahuiYorubaSimulationAdmixture.paramfile.txt`.
2. The **painting samples file**, which points to the ChromoPainterv2 output file(s) con- taining the painting samples of the putatively admixed population. The file we will use here is `data/BrahuiYorubaSimulationAdmixture.samplesfile.txt`.
3. The **recombination rate file**, which points to the recombination rate file(s) used when running ChromoPainterv2. The file we will use here is `data/BrahuiYorubaSimulationAdmixture.recomfile.txt`.

File 2 (the painting samples file) will point to the `output/BrahuiYorubaSimulationAdmixtureChrom22.samples.out` file we just made using ChromoPainterv2, which contains 10 sampled paintings for each haplotype of each individual from our simulated admixture population.

File 3 (the recombination rate file) will point to the `output/BrahuiYorubaSimulationChrom22.recomrates` we used when generating these paintings.

For File 1 (paramter input file), we have specified the input file `BrahuiYorubaSimulation.idfile.txt` we used when running ChromoPainterv2 above, as well as the output filenames (`save.file.XX`). We have also specified the painting we made **previously**, which painted the target population and each of the surrogate populations using the **same set of donors**. (In this case the donor populations – copyvector.popnames – are the same as the surrogate populations – surrogate.popnames.) This information is used in the linear model GLOBETROTTER uses to assign admixing sources.

The other parameters specify ways to run GLOBETROTTER. Most of these will likely not be changed, except for the first three, which specify whether to infer dates and admixture proportions (prop.ind) and/or bootstrap re-sample to determine uncertainty in date estimation (bootstrap.date.ind), and/or whether to standardize estimates by a "NULL" individual to help eliminate spurious signals of admixture (null.ind). (As long as you are detecting admixture in greater than three individuals, the latter is recommended as used below, though it is good practice to assess the results with and without specifcation of a null individual.)

To run GLOBETROTTER under these settings type:

```
R < GLOBETROTTER.R data/BrahuiYorubaSimulationAdmixture.paramfile.txt \
```

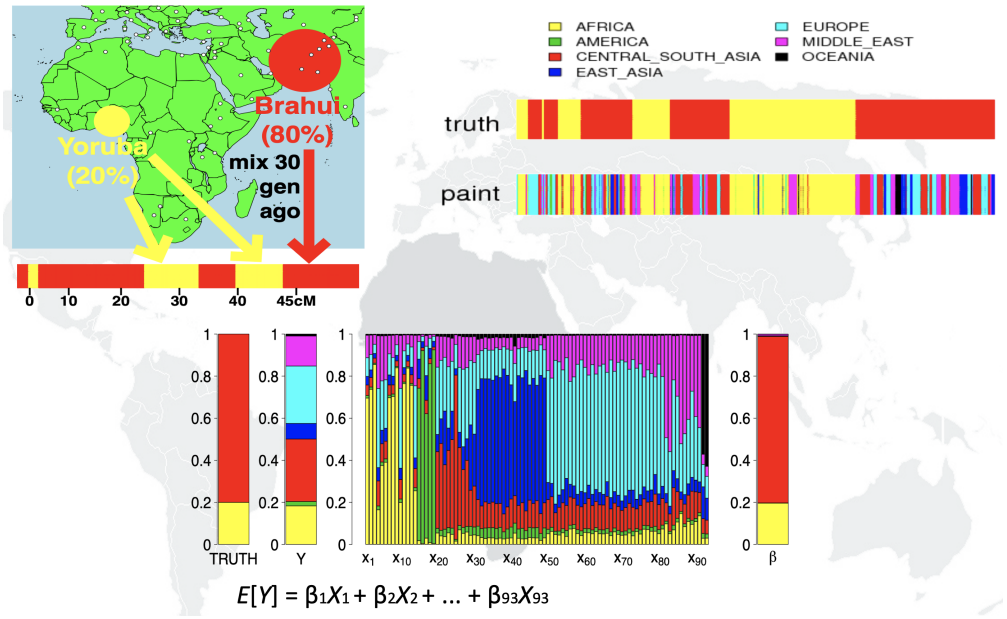


Figure 8: **Globetrotter Summary**

```
data/BrahuiYorubaSimulationAdmixture.samplesfile.txt \
data/BrahuiYorubaSimulationAdmixture.recomfile.txt --no-save > output.out
```

This will take a few minutes to run and note you will need the `nnls` package installed in R.

```
install.packages('nnls')
```

If you run into problems this is often because file paths are incorrectly specified - check these carefully.

Once successfully finished, the following output files will be produced, each in the `output` folder:

- `BrahuiYorubaSimulationAdmixed.globetrotter.main.txt` – this gives the results, including GLOBETROTTER's **best-guess** conclusion regarding admixture and the inferred admixture dates and proportions
- `BrahuiYorubaSimulationAdmixed.globetrotter.main.pdf` – this gives you **coancestry curves** for every combination of surrogate populations that are inferred to have contributed >0.1% ancestry to the target population
- `BrahuiYorubaSimulationAdmixed.globetrotter.main curves.txt` – this gives all of the raw data used to produce the curves in, in case you want to make your own plots

These are also provided for you to inspect.

Question: Check the `BrahuiYorubaSimulationAdmixed.globetrotter.main.txt` file. What is the point estimate of the admixture date?

Question: What are the major contributing sources (remember Figure 1 of Hellenthal et al)?

Question: Do we recover the same curves?

Generally we provide a confidence interval around our estimated dates. If `**enough time**`, change bootstrap.date.ind to **1**, which will provide 20 bootstrap resample estimates of the date. Then try changing the surrogates (i.e. remove some populations listed in surrogate.popnames).

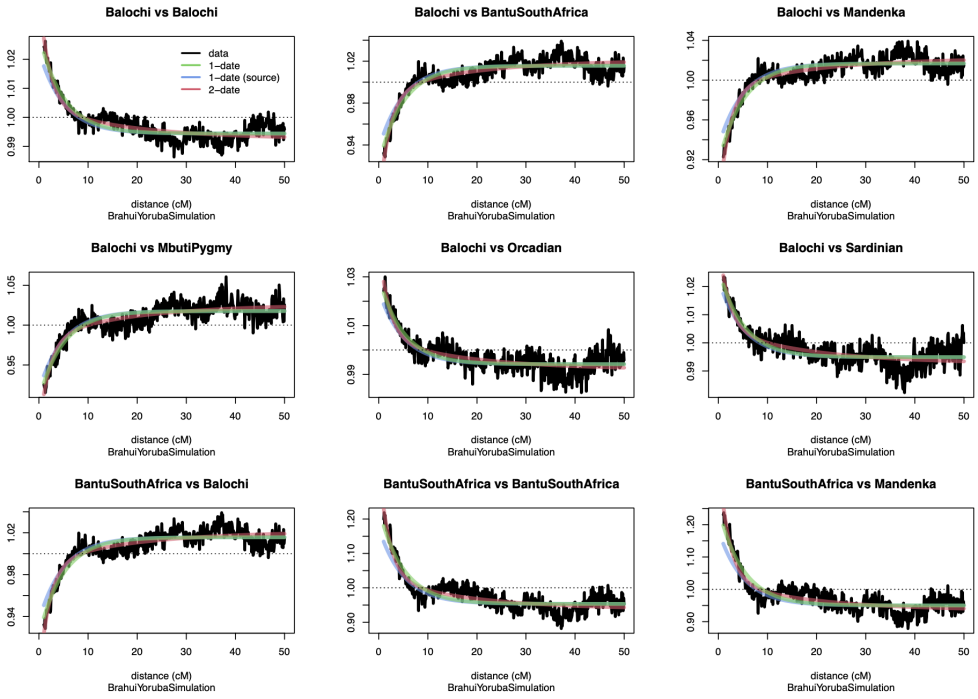


Figure 9: **Coancestry curves**

Conclusions

I hope this practical has served as an introduction to the some of the tools that can be used for analysing population structure and admixture history using both allele frequency and haplotype-based methods.

The commands used should all be quite portable and flexible. If you're interested, I encourage you to try them for yourself with variations and on your own datasets. Each software folder contains an associated manual.

Extensions/Additional Reading

If you're excited about haplotype-based analysese of human population structure here are some further papers (not mentioned in the lecture) which may be of interest and which make use of some of the methods we've tried out today:

- Patterns of genetic differentiation and the footprints of historical migrations in the Iberian Peninsula
- Genetic legacy of state centralization in the Kuba Kingdom of the Democratic Republic of the Congo
- Latin Americans show wide-spread Converso ancestry and imprint of local Native ancestry on physical appearance
- Admixture into and within sub-Saharan Africa
- Evidence for a Common Origin of Blacksmiths and Cultivators in the Ethiopian Ari within the Last 4500 Years: Lessons for Clustering-Based Inference

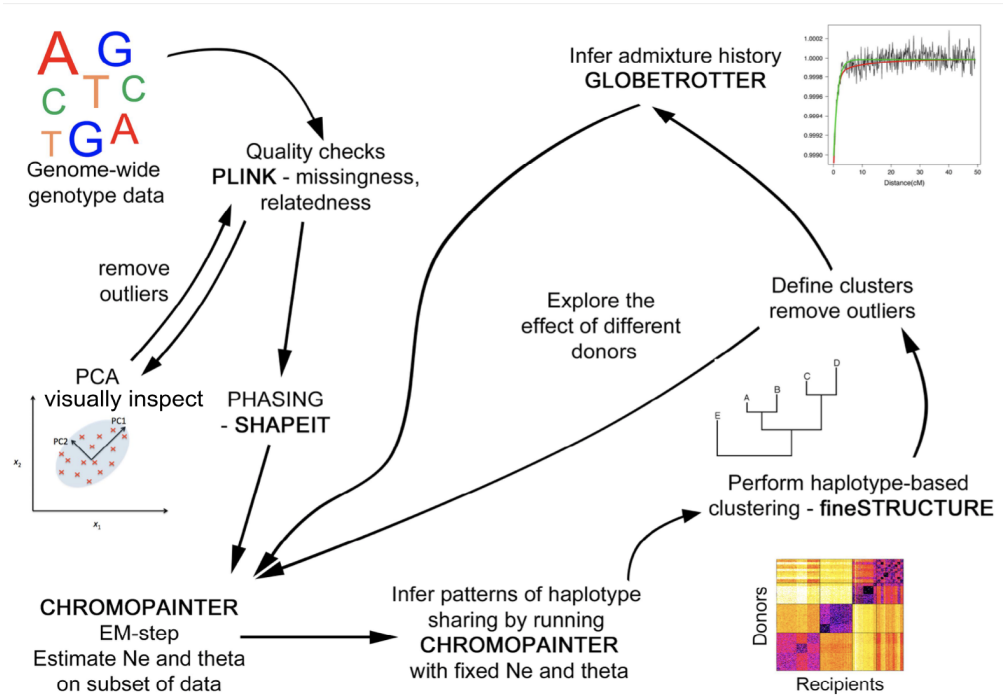


Figure 10: **Typical workflow**