Population Genomics: background, tools and programming

Clustering Algorithms

Garrett Hellenthal, University College London

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 [1], which is freely available and consists of data from the Human Genome Diversity Panel (http://www.cephb.fr/hgdp/) 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:

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

I've also added to these 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 [1] (see Figure 1) and is the example file included with `CHROMOPAINTER`.

1 ADMIXTURE

First we will apply ADMIXTURE [2] to these data. Save admixture_linux-1.22.tar.gz 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_linux-1.22/./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. Two output files will be made: BrahuiYorubaSimulationChrom22.admixture.2.Q, which gives the probability each individual is assigned to each of the K clusters, and BrahuiYorubaSimulationChrom22.admixture.2.P, which gives each SNP's allele frequencies for each of the K clusters.

1. Run admixture for K=2,3,4,5,6,7,8.

2. Plot results using the program ADMIXTUREBarplots.R:

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

3. As you can see from the top of the R code, this program makes a file called BrahuiYorubaSimulationChrom22ADMIXTURE.pdf. How do you interpret these findings? What populations "emerge" as you increase K at each step? At $K = 2$, Africa (blue) splits from non-African pops. At $K = 3$, East Asia (pink) splits. At $K = 4$, Europe (grey) splits out. At $K = 5$, MbutiPygmy (purple) splits out. At $K = 6$, the Kalash (maroon) split out. At $K = 7$, a single Makrani individual (black) splits out. At $K = 8$, a few Orcadian individuals (red) split out. What about the simulated population? The plots show that this population is admixed between Africans and non-Africans (e.g. at $K = 2$), at roughly the correct proportions.

4. Note you can also run cross-validation (--cv) to help infer the "best" K. And you can run bootstrap re-sampling (-B) to infer standard errors around estimates.Apparently cross-validation suggests that $K = 4$ has the lowest standard error. Looking at the barplots, it's not obvious that this is "best" (though note we are only looking at a single chromosome). In general, inferring K is a notoriously challenging problem, in part because there is no true K in reality – it's just a way to find "interesting" patterns in the data.

2 CHROMOPAINTER

Next we will apply CHROMOPAINTER [3] to the same data. As before, save ChromoPainterv2.tar.gz and BrahuiYorubaSimulation.poplistReduced.txt to any given folder, and then unzip and extract files with:

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

Next compile with:

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

ChromoPainterv2 can be run in lots of different ways; e.g. in the "Haplotype-based methods for demography" practical we will discuss a different way to run ChromoPainterv2 when exploring *admixture*. Today we are interested in using the ChromoPainterv2 output to cluster individuals into genetically homogeneous groups using fineSTRUCTURE. We describe our recommended way of doing this in Section 8.1 of ChromoPainterv2Instructions.pdf. The three steps we outline:

(I) 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).

  (b) Using the estimated parameters from (a), run ChromoPainterv2 again on all individuals and chromosomes.

(II) Second, sum the following ChromoPainterv2 output files from step 1(b) across chromosomes: XX.chunkcounts.out, XX.regionchunkcounts.out, XX.regionsquaredchunkcounts.out.

(III) Third, run fineSTRUCTURE (which we will do later today).

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. The command to run:

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

The command ''-a 0 0'' specifies to paint each individual in example/BrahuiYorubaSimulation.idfile.txt that is from a population in BrahuiYorubaSimulation.poplistReduced.txt, using every other individual from these populations as donors. The files example/BrahuiYorubaSimulationChrom22.haplotypes and example/BrahuiYorubaSimulationChrom22.recomrates are the haplotype data for all individuals and genetic map for this region, respectively, while `example/BrahuiYorubaSimulationAllVersusAllChrom22` is where the `ChromoPainterv2` output will go.

Run the above code. A problem – it may be too slow for this practical (takes ≈25min)! Therefore I have already done this painting for you. We will look at the output using `CHROMOPAINTERHeatMapPlot.R`, which plots a heatmap of the `ChromoPainterv2` "coancestry matrix". In particular we will look at the `XX.chunklengths.out` file, which gives the total cM length of DNA segments ("chunks") that each recipient individual (row) copies from each donor individual (column). (You can also look at the `XX.chunkcounts.out` file, which gives the total number of "chunks" that each recipient individual copies from each donor individual.) The plot will flip this, so that (in the *pdf file) columns are recipients and rows are donors. The tick marks along each axis color individuals based on their population labels (see legend at bottom).

```
R CMD BATCH CHROMOPAINTERHeatMapPlot.R
```

What does the output from this (i.e. `BrahuiYorubaSimulationAllVersusAllChrom22HEATMAP.pdf`) show? In particular answer the following questions:

1. Which groups are copied (painted from) least by the other groups? It looks as if African populations (BantuKenya, BantuSouthAfrica, Mandenka) are copied least (note the yellow streaks going left-to-right for these populations). Also, Kalash does not copy much from others (note the vertical yellow streak for Kalash), indicating it is an isolated population.

2. Which groups copy the most from each other? The African populations appear to copy a lot from each other (dark colors), and note that the East Asian populations (HanNChina, Mongola) copy a lot from each other.

3. How does the simulated population `BrahuiYorubaSimulation` look different from the other groups? The simulated population (tan lines at the far right and top) matches a bit more to Africa than other non-Africans match to Africa (i.e. the simulated inds have darker colors on the African population rows).

3 fineSTRUCTURE

Now we will use fineSTRUCTURE [3] to cluster these individuals based on their ChromoPainterv2 output. Again unzip and un-pack fineSTRUCTURE version 4.0.0:

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

We will use the pre-compiled binary fs_linux_glibc2.3 in the directory fs_4.0.0/. We first need to calculate a nuisance parameter "c", which takes into account the fact that all entries coancestry matrix are NOT actually independent though the basic multinomial model of fineSTRUCTURE assumes they are. Do this by typing the following:

```
Rscript calcC_Continents.R BrahuiYorubaSimulationAllVersusAllChrom22
```

Note that the value printed to screen is 0.184411557794475. We will use this when we run finestructure:

```
fs_4.0.0/fs_linux_glibc2.3 finestructure -I 1 -c 0.184411557794475 -x
10000 -y 20000 -z 100
BrahuiYorubaSimulationAllVersusAllChrom22.chunkcounts.out
BrahuiYorubaSimulationAllVersusAllChrom22.finestructure.out
```

In real applications, you should probably have each of ‘‘-x’’, ‘‘-y’’, ‘‘-z’’ a factor of 100 higher. These are the number of burn-in MCMC runs, total number of MCMC runs, and MCMC thinning parameter, respectively. But for reasons of time we use the values here. Here ‘‘-I 1’’ specifies that you want to start with all individuals assigned to 1 cluster, then split from there. The step above generates MCMC samples that assign individuals to clusters. To generate a tree using this output, type:

```
fs_4.0.0/fs_linux_glibc2.3 finestructure -c 0.184411557794475 -x 10000 -k
2 -m T -t 1000000
BrahuiYorubaSimulationAllVersusAllChrom22.chunkcounts.out
BrahuiYorubaSimulationAllVersusAllChrom22.finestructure.out
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 building 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: 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. (If not too slow, I recommend setting this ‘‘-t’’ parameter very high as I do here, in which case fineSTRUCTURE will do *all* pairwise comparisons among current populations at each level of the tree.)

Look at the resulting clusters and tree using CHROMOPAINTERHeatMapPlot.R, 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. What do you see? In particular answer the following questions:

1. Do the inferred clusters seem sensible? Yes – all African individuals cluster together, the Kalash cluster together, and East Asians clearly cluster together, etc.

2. How do the clusters compare to those when using ADMIXTURE? Maybe a bit more resolution, or at least it may be easier to see the clusters on the heatmap relative to using the barplots.

3. Does the inferred tree seem sensible? Yes, vaguely – happily the African populations merge together before merging with non-Africans, for example.

4. What about the simulated population? These individuals are placed somewhere on the tree between Africans and (some) non-Africans. The reason for this is that they are admixed between the two, and fineSTRUCTURE only uses a “no-admixture” model (i.e. does not allow individuals to be assigned to multiple clusters), meaning you should treat the tree with some caution.

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? You can also calculate the proportion of MCMC samples for which every pair of individuals are assigned to the same cluster, by typing:

```
fs_4.0.0/fs_linux_glibc2.3 finestructure -c 0.184411557794475 -e
meancoincidence BrahuiYorubaSimulationAllVersusAllChrom22.chunkcounts.out
BrahuiYorubaSimulationAllVersusAllChrom22.finestructure.out
BrahuiYorubaSimulationAllVersusAllChrom22.finestructureCOINCIDENCE.out
```

Do this for both of the trees you have made (saving the second one as `BrahuiYorubaSimulationAllVersusAllChrom22.finestructureCOINCIDENCEII.out`). Then use `FineStructureCoincidenceMatrixVisualize2Seeds.R` 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). How consistent do results from the two runs appear to be? Fairly consistent – Individuals that are uncertain (e.g. in CentralSouthAsia) are uncertain in both fineSTRUCTURE runs, meaning that they are sometimes clustered together and sometimes apart (i.e. non-black colors on `BrahuiYorubaSimulationAllVersusAllChrom22FSCoincidencePlot.pdf`).

References

[1] G. Hellenthal, G.B.J. Busby, G. Band, J.F. Wilson, C. Capelli, D. Falush, and S. Myers. A genetic atlas of human admixture history. *Science*, 343:747–751, 2014.

[2] D.H. Alexander, J. Novembre, and K. Lange. Fast model-based estimation of ancestry in unrelated individuals. *Genome Research*, 19:1655–1664, 2009.

[3] D.J. Lawson, G. Hellenthal, S. Myers, and D. Falush. Inference of population structure using dense haplotype data. *PLoS Genet*, 8(1):e1002453, 2012.