CS4775 Final Project

Population Structure Inference with STRUCTURE

Jonathan Mares

December 13th, 2016

Abstract

This writeup details the final project for CS4775. This report discusses a partial implementation of the STRUCTURE algorithm. Models with and without admixture are considered. STRUCTURE without admixture is implemented and is shown to have good results that agree with both STRUCTURE and Admixture (UCLA). STRUCTURE with admixture is also implemented, but is just shy of returning good results. The algorithms are considered on a portion of the Hapmap3 dataset, as acquired by Admixture.

I apologize for the empty pages that exist between sections. I am using a latex template that has nice features but some excessive empty pages, which I did not get a chance to remove. Please do not take this as an attempt on my part to increase the page length.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Objectives

The primary motivation for this project was to implement a non-trivial approach to a problem in the field of computational biology. The objectives were to implement an approach discussed in the literature and apply it to a real dataset.

1.2 Contributions and Statement of Originality

I, Jonathan, am the sole contributor to both the implementation and data analysis. Guidance was generously given by Professor Williams and Melissa Hubisz.

1.3 What is Included

I have submited my implementation for jonathanStructure, the Hapmap3.ped dataset, as well as parameter files for STRUCTURE, in addition to this writeup.

1.4 Other Notes

For convenience, I refer to my implementation of STRUCTURE as jonathanStructure, to distinguish between the Stanford implementation. I refer to the Stanford implementation as STRUCTURE.

Chapter 2

Background Theory

2.1 Background

In the field of population genetics, classification of individuals into populations is a common method used to study various problems. For example, in the study of evolution, populations are of particular interest and are used to study evolutionary relationships, such as the migratory and mating patterns of humans across periods of human history. Classifying populations provides an alternative, quantitative method to other definitions of populations such as cultural or physical characteristics [PSD00]. It suffices to say that population classification is an important method employed in evolutionary biology in a broad sense.

In addition to anthropological questions, population classification is important in other contexts. Elhaik et. al has described an approach that utilizes STRUCTURE to determine geographic origin [Elh14]. One specific problem that can arise in the field of medicine is the situation where ancestry can act as a confounder [ANL09]. Researchers need to be aware of potential false positive associations that can arise. One such scenario occured in medical history; A gene was initially found to cause lower risk of type II diabetes, where later it was found that caucasian ancestry can also result in this lowered risk. Researchers may observe a gene to cause an apparent association, while in fact it is an ancestry (at least in part) that may cause some given trait to be expressed [ANL09]. Population classification can be used to account or correct for such situations. There are other interesting applications of population classification, such as the approach taken by Foreman et al. (1997) and Roeder et al.), who were interested in estimating the degree of cryptic population structure for the purpose of studying false matches at DNA fingerprint loci.

A background on clustering methods is given in the STRUCTURE paper for the purpose of shedding light on solving the problem of clustering genetically similar individuals. Distance-based models focus on calculating a pairwise distance matrix, which is followed by visual inspection of clusters [PSD00]. These methods suffer from some drawbacks; One drawback in particular is the difficulty of assessing confidence in the clustering, something that a model based approach can provide [PSD00]. Choosing the appropriate model can be difficult when taking a model-based approach, however. We will now describe in some detail the approach taken by STRUCTURE.

2.2 STRUCTURE description

STRUCTURE takes a model-based clustering method for using multilocus genotype data to infer structure and assign individuals to populations [PSD00]. We define X to be the genotypes of sampled individuals, Z the unknown populations of origin of individuals, P the unknown allele frequencies across all populations, and Q the admixture proportions for individuals. Depending on the model, these vectors may be multi-dimensional (details provided for each model). There are some notable assumptions taken by STRUCTURE: Hardy-Weinberg equilibrium within populations and complete linkage equilibrium between loci within populations [PSD00]. Finally, a Bayesian approach is taken by STRUCTURE to provide a coherent for inferring the uncertainty of parameter estimates [PSD00].

Algorithm 1 MCMC with no admixture

```
procedure MCMC-NA($a, b$)                                ▷ MCMC with no admixture
    for $i \leftarrow 0, N$ do
        $z_i \leftarrow \frac{1}{k}$                                 ▷ Sample from uniform distribution
    end for
    for $m \leftarrow 0, numberItterations$ do                       ▷ Itterate $m$ times
        for $i \leftarrow 0, N$ do
            for $l \leftarrow 0, L$ do
                $n_{klj} \leftarrow n_{klj} + 1 \; if \; x_l^{(i,a)} == j$          ▷ count allele copies
            end for
        end for
        for $k \leftarrow 0, K$ do
            for $l \leftarrow 0, L$ do
                $p_{kl} \leftarrow Dirichlet(\lambda + n_{kl1}, \lambda + n_{kl2})$▷ Sample p from the Dirichlet distribution
            end for
        end for
        for $i \leftarrow 0, N$ do
            $z_i \leftarrow \frac{b_k}{b}$                                    ▷ Sample z
        end for
    end for
end procedure
```

2.2.1 Model without admixture

We will attempt to keep the notation consistent with STRUCTURE, as well as provide some clarity to the algorithm details. We suppose we are tasked with have N diploid individuals at L loci. Under a no admixture assumption, each individual originates from one of K populations. We define X to be a vector of observed genotypes with elements $x_l^{i,1}, x_l^{i,2}$ where $l = 1, 2 \ldots L$ is the locus, $i = 1, 2 \ldots N$ is the individual, and the 1 and 2 represent the an index a into the locus' allele copies. Each individual receives 2 alelle copies, such that $a \in \{1, 2\ \}$. We also define the vector Z the unknown origin populations of individuals, which has elements z_i (one population per individual). Lastly, we define n_{klj}, which is the number of ocpies of allele j at locus l observed in individuals assigned by Z to population k. Algorithm 1 attempts to provide an overview of the MCMC model with no admixture.

We use the constant b to simplify the pseudocode. Let's define $b_k = Pr(x^i|P, z^i = k)$ and b to be $\sum_{k'=1}^{K} b_i$, which is essentially a normalization factor. In the line where we count allele copies, it is is difficult to succinctly describe the procedure. In words, we keep track of allele copies at each locus belonging to population k We need to also define b_k:

$$Pr(x^i|P, z^i = k) = \prod_{j=1}^{L} p_{klx^{i,1}} p_{klx^{i,2}}$$

which in words the product of all of the allele copies for a given indivudal.

Assumptions

While STRUCTURE has the ability to specify burn in itterations, we omit this for simplicity. We've also assigned $\lambda = 1$.

Additional Notes

We assign the minimum value of $min(n_{klj}) = 1$ as pseudocounts, to prevent 0 probabilities from causing the model to break down.

2.3 Model with Admixture

A pseudocode description is omitted, but the changes are highlighted as follows. We still sample p_{kl} as before, however we need to update the way we count n_{klj} and our definition of $z_l^{(i,a)}$. z is no longer one population assignment per individual, but rather a population assignment for each allele copy over all individuals. The snp data and the matrix z, in other words, have the same size. We initialize each element of $z_l^{(i,a)}$ as before, with a uniform distribution. We also redefine $n_{klj}$to be the number of copies of allele j at locus l as assigned by $z_l^{(i,a)}$. The only difference in the counts here is how z is defined.

We also introduce a matrix m_k^i, which is the number of allele copies in individual i that originated in population k. In other words, we sum across all allele copy assignemnts (z) for each individual, keeping track of how many assignments were made from each population. We will use m to sample admixture proportions q as follows:

$$q(i|X,Z) = Dirichlet(\alpha + m_1^i, \ldots, \alpha + m_K^i)$$

where K is the number of populations. As a simplifying assumption, we choose $\alpha = 1$ and forego using a Metropolis-Hastings update step.

Finally, we sample each element of z as follows:

$$Pr(z_l^{(i,a)} = k|X,P) = \frac{q_k^i * p_{klx_l^{i,a}}}{\sum_{k=1}^{K} q_k^i p_{k'lx_l^{i,a}}}$$

Once again, the log likelihood of the assignments were assumed to be the sum of the log probabilities in the matrix n.

2.4 ADMIXTURE description

The dataset was also analyzed with ADMIXTURE [ANL09]. The algorithm takes an EM approach with a block relxation algorithm, giving it fast computational speeds for the problem at hand. Details of the algorithm are ommitted here.

Chapter 3

Results and Discussion

In this section, I compare the results of jonathanStructure implementation with both the Stanford STRUCTURE and ADMIXTURE.

3.1 SNP Data

Hapmap3 Data was obtained from the UCLA admixture website, which was included in the freely available ADMIXTURE software download. As explained in (Alexander, et al) a subset of the 1,440,616 available markers of the HapMap Phase 3 project were chosen according to the following criteria: (1) minimize background linkage disequilibrium (no markers must be closer than 200 kb apart), and no more than 5% of genotypes missing [ANL09]. Unrelated individuals from the CEU, YRI, MEX and AWS samples were used to construct a dataset of 324 individuals with 13,298 markers.

The best K was chosen to be 3, based on running cross-validation for K values 1,2,3,4,5, as suggested by the ADMIXTURE manual:

```
for K in 1 2 3 4 5; \
do admixture −−cv hapmap3.bed $K | tee log${K}.out; done
```

It's fairly clear that for this data set $K = 3$ is a good choice:

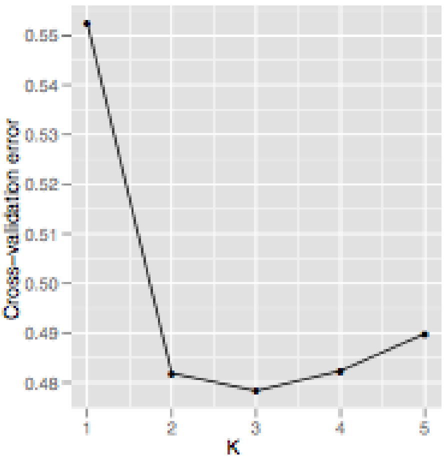

Figure 3.1: Cross-validation plot for the hapmap3 dataset (figure obtained from manual)

3.2 Running with jonathanStructure

3.2.1 No Admixture

One thing to note about STRUCTURE is the populations labels k themselves are arbitrary. When comparing data, I attempt to match up populations such that $k = 2$ is indeed the same $k = 2$ between comparisons. As such, the clustering of individuals was more relevant than a particular assignment.

The log likelihood of the data was computed as follows:

$$likelihood = \sum_k \sum_l p_{kl}$$

this was a simplifying assumption made with Melissa to reduce the complexity of analysis, as the other terms were deemed to be constant factors, and unnecessary for comparing likelihoods across runs of jonathanStructure.

The large size of the dataset made it costly with respect to time to run both STRUCTURE and jonathanStructure. Due to the large size of the dataset however, the log likelihood seemed to level out quickly, on the order of tens of itterations.

Table 3.1: Summary of a selection of jonathanSTRUCTURE no admixture runs

Run	log-likelihood	K=1	K=2	K=3
1	-85180	.04	0.46	0.5
2	-81579	0.494	0.5	.006
3	-89868	.373	.127	0.5

A selection of runs with statistics are shown in Table 3.1

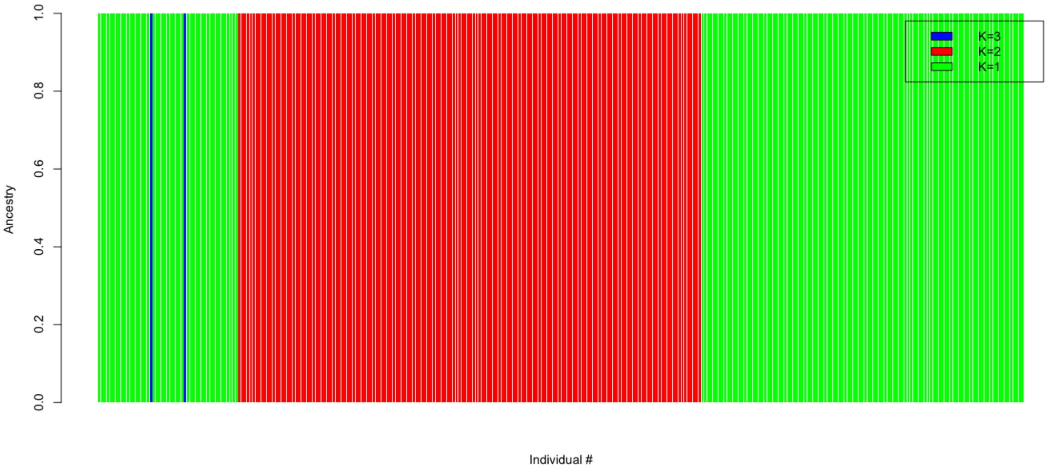

Figure 3.2: Ancestry proportions as obtained by jonathanSTRUCTURE with no admixture. Corresponds to Run 2 in Table 3.1

By inspecting the bar plots, we can see that jonathanSTRUCTURE with no admixture performs well in comparison to STRUCTURE no admixture (Figures 3.5- 3.7. When comparing the population determinations between structureJonathan Run 3 and STRUCTURE Run 1, it was found that the output matched on all but 7 individuals. A short script to find this is included in the appendix. We will also be making comparisons with the data shown in 3.10. Both STRUCTURE and jonathanAdmixture capture the green proportions of the populations well (ind 1-49 and 212-324). Red populations are also captured well.

It is likely that both the Green and Red populations are captured best because those individuals seem to be the most homogenous. Depending on the run, the individuals with blue ancestry (individuals 162-211) are captured as completely from blue or red. Many of these individuals seem to be admixed roughly equally, so it would make sense that both algorithms would struggle to assign those individuals to one population consistently. We postulate that the reason that results differ in some amount between results is due the high dimensional solution space, which could result in convergence upon a local maximum rather than the true solution.

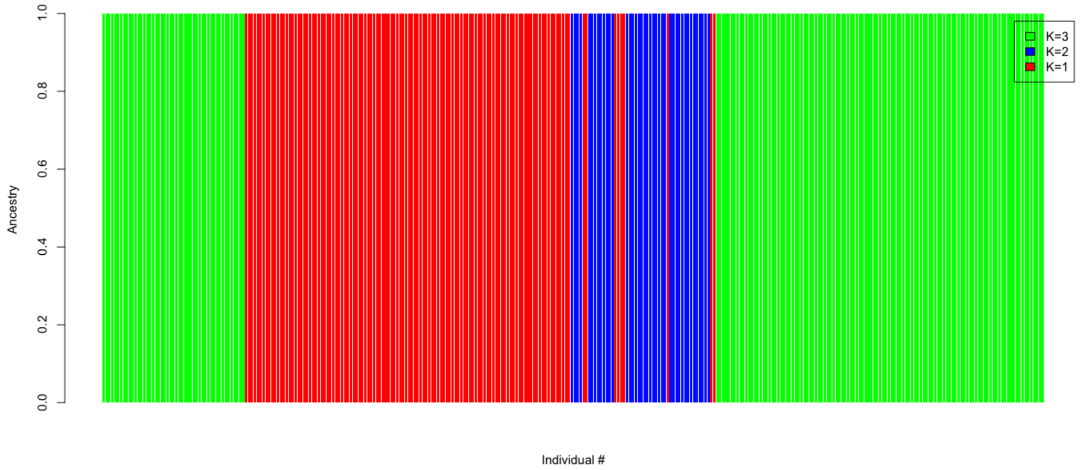

Figure 3.3: Ancestry proportions as obtained by jonathanSTRUCTURE with no admixture. Corresponds to Run 2 in Table 3.1

3.2.2 jonathanStructure with admixture

An implementation of admixture was attempted. It is believed that the implementation is close to correct, however there may be a few bugs that cause the assignments to stay uniform, as can be seen in Figure 3.4. The admixture coefficients do not agree with the output of STRUCTURE or Admixture. However, it is possible that jonathanStructure did not run for long enough (100 iterations took 7 hours). One way to have determine if this were indeed true were to cut the size of the Hapmap3 dataset, reducing the number of loci per individual. One problem with the approach taken was that the dataset was too large. For the admixture scenario, the matrix m_k has 8.5 million entries, and needs to be updated on every iteration. If a smaller dataset was initially chosen, more time could have been spent on further implementation and showing better results with admixture.

3.3 Running with STRUCTURE

3.3.1 No Admixture

STRUCTURE was run on the same dataset with 1000 burnin and 1000 iterations. Table 3.2 summarizes run statistics and Figures 3.5 - 3.7 show the population assignments. We can see

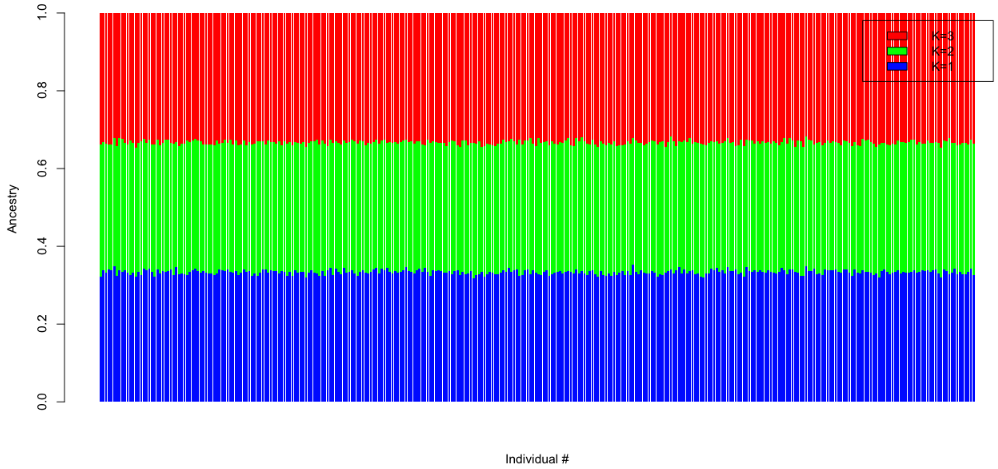

Figure 3.4: Ancestry proportions as obtained by jonathanStructure with admixture after 100 itterations. log likelihood of data=-82255

that the population assignments may be different every time, but the clusters remain somewhat consistent.

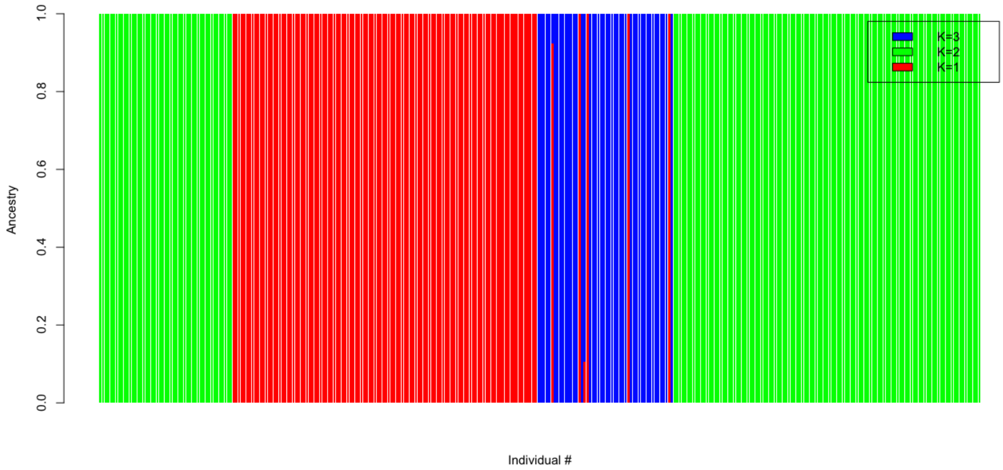

Figure 3.5: Ancestry proportions as obtained by STRUCTURE (admix=False) with Burn-In=1000 and Reps=1000. Corresponds to Run 1 in Table 3.2

Table 3.2: Summary of a selection of STRUCTURE no admixture runs

Run	log-likelihood	K=1	K=2	K=3
1	-3867176	0.361	0.5	0.139
2	-3866912	0.5	0.485	0.015
3	-3867176	0.485	0.015	0.5

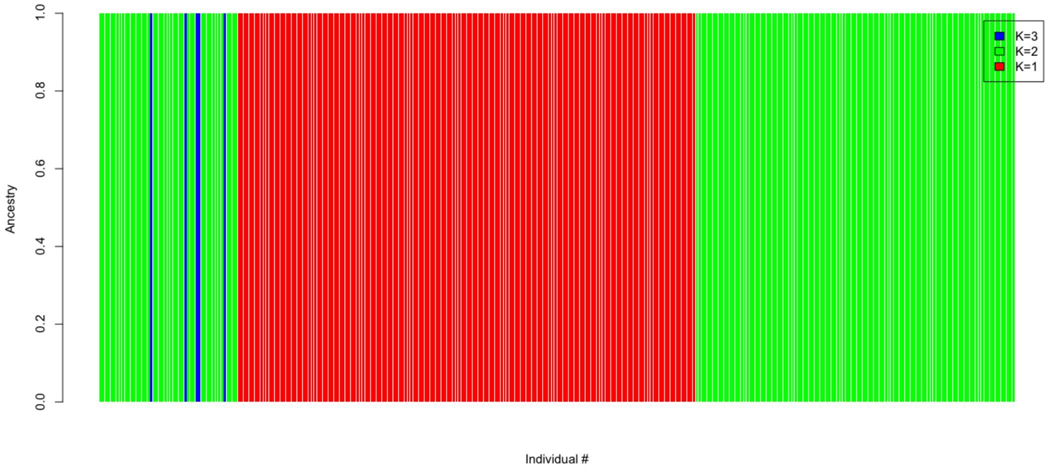

Figure 3.6: Ancestry proportions as obtained by STRUCTURE (admix=False) with Burn-In=1000 and Reps=1000 Corresponds to Run 2 in Table 3.2

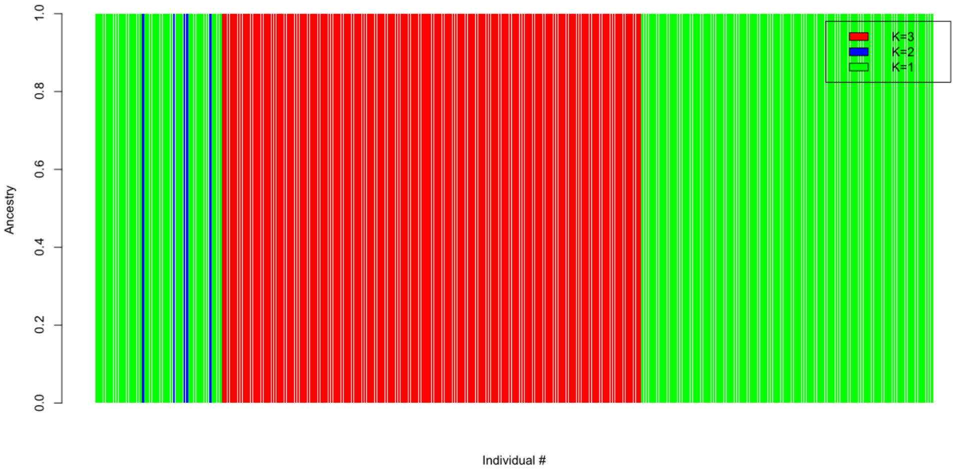

Figure 3.7: Ancestry proportions as obtained by STRUCTURE (admix=False) with Burn-In=1000 and Reps=1000 Corresponds to Run 3 in Table 3.2

3.3.2 Structure with Admixture

STRUCTURE was also run with the admixture flag. Figure 3.9 shows close matching with the output of Admixture.

3.4 Running with ADMIXTURE

It is interesting to compare output with ADMIXTURE. ADMIXTURE considers individuals to be admixed. One advantage of comparing with Admixture is the speed in which the algorithm

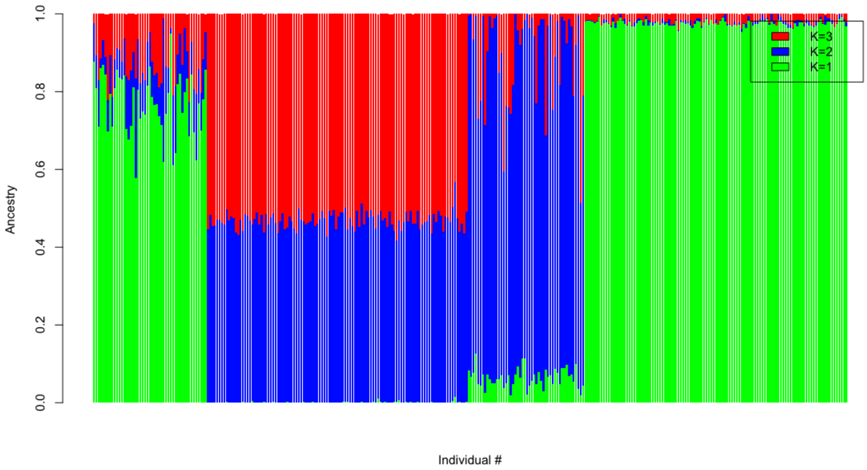

Figure 3.8: Ancestry proportions as obtained by STRUCTURE (admix=TRUE) with Burn-In=1000 and Reps=1000

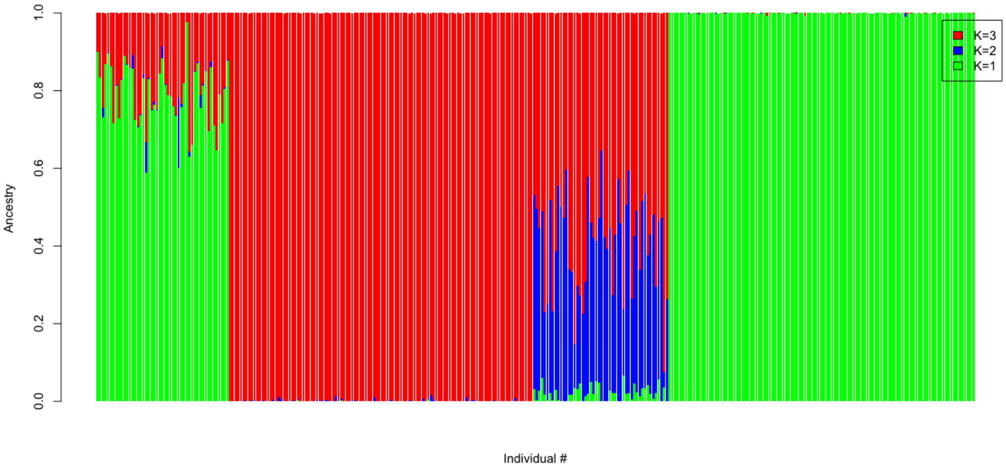

Figure 3.9: Ancestry proportions as obtained by STRUCTURE (admix=TRUE) with Burn-In=10000 and Reps=20000

runs (the Block relaxation algorithm converges in $\approx 33s$). The algorithm finishes in under 45 seconds with first running the FRAPPE EM algorithm for 5 itterations (this brings the solution to the vicinity of the maximum) followed by the Block Relaxation algorithm for 21 iterations.

Output from running ADMIXTURE can be found in the appendix.

A bar-chart representing the ancestry proportion from the individuals in the dataset as obtained by running ADMIXTURE is shown in Figure 3.10.

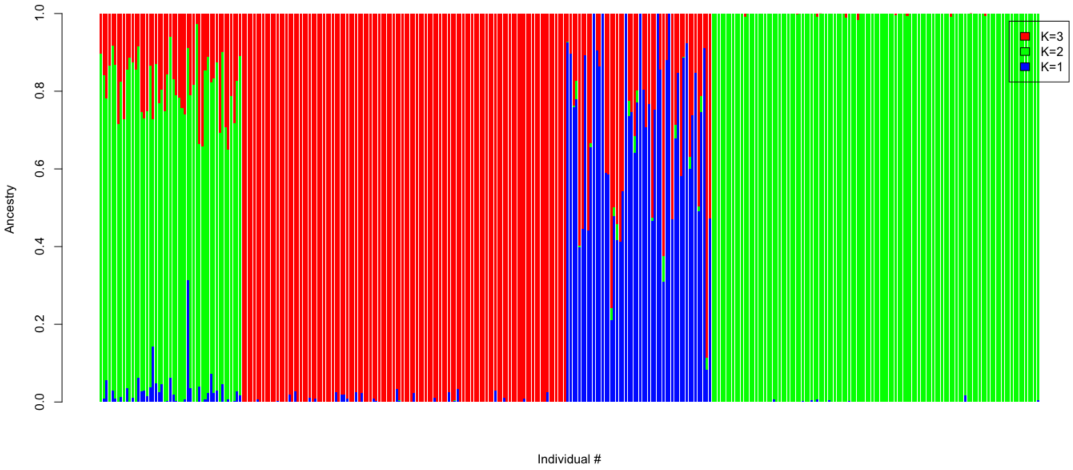

Figure 3.10: Ancestry proportions as obtained by ADMIXTURE. Red bars indicate $K = 3$, green bars indicate $K = 2$, and blue bars indicate $K = 1$ proportions.

Chapter 4

Conclusion

4.1 Remarks

It can be concluded that jonathanAdmixture with no admixture returns results that agree with both STRUCTURE and admixture. However, it cannot be definitively determined that the admixture implementation returns good results. It is believed that the admixture implementation was not far from a working one. Further steps would have included sampling α using a Metropolis-Hastings probability update step.

Appendix A

Appendix

A.1 Additional Information

The bar plots were produced in R with the following command:

```
barplot(t(as.matrix(tbl)), col=rainbow(3),xlab="Individual #", ylab="Ancestry", border=NA, legend=(c("K=1","K=2","K=3") ) )
```

A.2 Admixture Output

```
$ ./admixture hapmap3.bed 3
**** ADMIXTURE Version 1.3.0 ****
**** Copyright 2008-2015 ****
**** David Alexander, Suyash Shringarpure, ****
**** John Novembre, Ken Lange ****
**** ****
**** Please cite our paper! ****
**** Information at www.genetics.ucla.edu/software/admixture ****


Random seed: 43
```

```
Point estimation method: Block relaxation algorithm
Convergence acceleration algorithm: QuasiNewton, 3 secant conditions
Point estimation will terminate when objective function delta <
   0.0001
Estimation of standard errors disabled; will compute point estimates
    only.
Size of G: 324x13928
Performing five EM steps to prime main algorithm
1 (EM)  Elapsed: 0.609 Loglikelihood: -4.38757e+06 (delta): 2.87325e
   +06
2 (EM)  Elapsed: 0.577 Loglikelihood: -4.25681e+06 (delta): 130762
3 (EM)  Elapsed: 0.579 Loglikelihood: -4.21622e+06 (delta): 40582.9
4 (EM)  Elapsed: 0.578 Loglikelihood: -4.19347e+06 (delta): 22748.2
5 (EM)  Elapsed: 0.54 Loglikelihood: -4.17881e+06 (delta): 14663.1
Initial loglikelihood: -4.17881e+06
Starting main algorithm
1 (QN/Block)  Elapsed: 1.117 Loglikelihood: -3.94775e+06 (delta):
   231058
2 (QN/Block)  Elapsed: 1.132 Loglikelihood: -3.8802e+06 (delta):
   67554.6
3 (QN/Block)  Elapsed: 1.283 Loglikelihood: -3.83232e+06 (delta):
   47883.8
4 (QN/Block)  Elapsed: 1.782 Loglikelihood: -3.81118e+06 (delta):
   21138.2
5 (QN/Block)  Elapsed: 1.389 Loglikelihood: -3.80682e+06 (delta):
   4354.36
6 (QN/Block)  Elapsed: 1.266 Loglikelihood: -3.80474e+06 (delta):
   2085.65
7 (QN/Block)  Elapsed: 1.655 Loglikelihood: -3.80362e+06 (delta):
   1112.58
```

```
8 (QN/Block)  Elapsed: 1.541 Loglikelihood: -3.80276e+06 (delta):
    865.01
9 (QN/Block)  Elapsed: 1.304 Loglikelihood: -3.80209e+06 (delta):
    666.662
10 (QN/Block)  Elapsed: 1.493 Loglikelihood: -3.80151e+06 (delta):
    579.49
11 (QN/Block)  Elapsed: 1.356 Loglikelihood: -3.80097e+06 (delta):
    548.156
12 (QN/Block)  Elapsed: 1.391 Loglikelihood: -3.80049e+06 (delta):
    473.565
13 (QN/Block)  Elapsed: 1.268 Loglikelihood: -3.80023e+06 (delta):
    258.61
14 (QN/Block)  Elapsed: 1.385 Loglikelihood: -3.80005e+06 (delta):
    179.949
15 (QN/Block)  Elapsed: 1.484 Loglikelihood: -3.79991e+06 (delta):
    146.707
16 (QN/Block)  Elapsed: 1.373 Loglikelihood: -3.79989e+06 (delta):
    13.1942
17 (QN/Block)  Elapsed: 1.488 Loglikelihood: -3.79989e+06 (delta):
    4.60747
18 (QN/Block)  Elapsed: 1.372 Loglikelihood: -3.79989e+06 (delta):
    1.50012
19 (QN/Block)  Elapsed: 1.362 Loglikelihood: -3.79989e+06 (delta):
    0.128916
20 (QN/Block)  Elapsed: 1.264 Loglikelihood: -3.79989e+06 (delta):
    0.00182983
21 (QN/Block)  Elapsed: 1.32 Loglikelihood: -3.79989e+06 (delta):
    4.33787e-05
Summary:
Converged in 21 iterations (33.735 sec)
Loglikelihood: -3799887.171935
```

```
Fst divergences between estimated populations:
       Pop0 Pop1
Pop0
Pop1 0.163
Pop2 0.073 0.156
```

A.3 Comparing Population Assignments

```




# compare outputs of STRUCTURE and jonathanStructure
def main():
        jonathan = []
        with open(’s_j_3’) as f:
                for line in f:
                        jonathan.append(map(int,line.split()))

        structure = []
        with open(’f4_Q’) as f:
                for line in f:
                        structure.append(map(float,line.split()))


        jonathanOut = []
        for line in jonathan:
                if line[0] == 1:
                        jonathanOut.append("Red")
                elif line[1] == 1:
                        jonathanOut.append("Blue")
                elif line[2] == 1:
                        jonathanOut.append("Green")
        structureOut = []
        for line in structure:
                if line[0] == 1:
                        structureOut.append("Red")
                elif int(line[1]) == 1:
                        structureOut.append("Green")
                elif int(line[2]) == 1:
                        structureOut.append("Blue")
                else:
                        print "somethign went wrong"

        error = 0
        for i in range(len(jonathanOut)):
                if jonathanOut[i] != structureOut[i]:
                        error +=1
        print error


if __name__ == "__main__":
        main()
```

Bibliography

[ANL09] D.H. Alexander, J. Novembre, and K. Lange. Fast model-based estimation of ancestry in unrelated individuals. *Genome Research*, 19:1655–1664, 2009.

[Elh14] E Elhaik. Geographic population structure analysis of worldwide human populations infers their biogeographical origins. *Nat. Commun*, 5(3513), 2014.

[PSD00] J.K. Pritchard, M. Stephens, and P. Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945, 2000.