# HGDP Population Structure Workshop

John Novembre with contributions from Joe Marcus, Chi-Chun Liu, Kate Farris

July, 2021

A note on dependencies:

This workshop uses the unix command-line programs 'plink' and 'admixture' and the R libraries Rcolorbrewer, ggplot2, cowplot, and PCAviz. There is also the option for additional visualization using the software 'pong'.

Run this cell once at the beginning of the workshop…

```r
tryCatch({
  library(devtools)
},error = function(err){
  install.packages("devtools")
})
tryCatch({
library(PCAviz)
},error = function(err){
  devtools::install_github("NovembreLab/PCAviz",build_vignettes = FALSE)
})
```

# HGDP Population Structure Workshop

Introduction

Population structure is a commonplace feature of genetic variation data, and it has importance in numerous application areas, including evolutionary genetics, conservation genetics, and human genetics. At a broad level, population structure is the existence of differing levels of genetic relatedness among some subgroups within a sample. This may arise for a variety of reasons, but a common cause is that samples have been drawn from geographically isolated groups or different locales across a geographic continuum. Regardless of the cause, understanding the structure in a sample is necessary before more sophisticated analyses are undertaken. For example, to infer divergence times between two populations requires knowing two populations even exist and which individuals belong to each.

Two of the most commonly used approaches to describe population structure in a sample are principal components analysis [@Menozzi78;@CavSforza94; @Price06;@Patterson06] and admixture proportion inference [@Pritchard00;@Novembre16a]. In brief, principal components analysis reduces a multi-dimensional dataset to a much smaller number of dimensions that allows for visual exploration and compact quantitive summaries. In its application to genetic data, the numerous genotypes observed per individual are reduced to a few summary coordinates. With admixture proportion inference, individuals in a sample are modeled as having a proportion of their genome derived from each of several source populations. The goal is to infer the proportions of ancestry in each source populations, and these proportions can be used to produce compact visual summaries that reveal the existence of population structure in a sample.

The history and basic behaviors of both these approaches have been written about extentsively, and so we refer readers to several previous publications to learn the basic background and interpretative nuances of these approaches and their derivatives [@Pritchard00; @Falush03; @Rosenberg05; @Hubisz09; @Raj14; @Novembre14;@Falush16; @Alexander09; @Alexander11; @Price06;@Patterson06; @Novembre08a; @McVean09; @Novembre16b]. In this workshop you will gain experience running these analyses.

Materials

The protocol we present is based on two pieces of software: 1) the `ADMIXTURE` software that our team developed [@Alexander09] for efficiently estimating admixture proportions in the “Pritchard-Stephens-Donnelly” model of admixture [@Pritchard00;@Novembre16a]. 2) The `smartpca` software developed by Nick Patterson and colleagues for carrying out PCA [@Price06]. Both of these pieces of software are used widely. We also pair them with downstream tools for visualization, in particular `pong` [@Behr16], for visualizing output of admixture proportion inferences, and `PCAviz` [@Novembre17], a novel R package for plotting PCA outputs. We also use `PLINK` [@Purcell07; @Chang15] as a tool to perform some basic manipulations of the data.

The example data we use is derived from publicly available single-nucleotide polymorphism (SNP) genotype data from the CEPH-Human Genome Diversity Panel [@Cann02]. Specifically, we will look at Illumina 650Y genotyping array data as first described by Li et al [@Li08]. This sample is a global-scale sampling of human diversity with 52 populations in total, and the raw files are available from the following link: http://hagsc.org/hgdp/files.html. These data have been used in numerous subsequent publications and are an important reference set.

A few technical details are that the genotypes were filtered with a cutoff of 0.25 for the Illumina GenCall score [@GenCall] (a quality score generated by the basic genotype calling software). Further, individuals with a genotype call rate <98.5% were removed, with the logic being that if a sample has many missing genotypes it may due to poor quality of the source DNA, and so none of the genotypes from that individual should be trusted. Beyond this, to prepare the data, we have filtered down the individuals to a set of 938 unrelated individuals. We exclude related individuals as we are not interested in population structure that is due to family relationships and methods such as PCA and `ADMIXTURE` can inadvertently mistake family structure for population structure. The starting data are available as plink-formatted files `H938.bed` `H938.fam`, `H938.bim`, and an accompanying set of population identifiers `H938.clst.txt`, from this link: https://www.dropbox.com/sh/66p1hhbtx0rb3wt/AAA0aDXVTNhjUi5nH03___D4za?dl=0.

As a pragmatic side note, it is common (and recommended) when carrying out analyses of population structure to merge one’s data with other datasets that contain populations which may be representative sources of admixing individuals. For example, in analyzing a dataset with African American individuals, it can be helpful to include datasets containing African and European individuals in the analysis. These datasets can be merged with your dataset using software such as `plink`. However, when merging several datasets, one should be aware of potential biases that can be introduced due to strand flips (i.e. one dataset reports genotypes on the ‘+’ strand of the reference human genome, and another on the ‘-’ strand). One precautionary step to detect strand flips is to group individuals by what dataset they derive from and then produce a scatterplot of allele frequencies for pairs of groups at a time. If strand flips are not being controlled correctly, one will observe numerous variants on the $y = 1 - x$ line, where x is the frequency in one dataset and y is the frequency in a second dataset. (Note: this rule of thumb assumes levels of differentiation are low between datasets, as is the case in human datasets in general, but one should still keep this in mind interpreting results).

Note about logistics

You may have been given a single tarball with this workshop, or you may have downloaded it from github. In either case, look in the `data` subdirectory for files. If the `data` directory is empty (or if you find it is missing any files referred to below, eg, the H938… or H938_Euro… files), then navigate to this link http://bit.ly/1aluTln and download all the files as a `.zip`, unzipp them, and put them in the data directory.

The commands below assume you are doing the exercises in a sister subdirectory to `data`. As such, binary commands are often denoted as being run by typing `../bin/plink` and the data files referenced are in `../data` for example. You may want to modify these if you are working in other directories. You can also updating your PATH variable to make entering commands easier (look on the web to learn this).

Subsetting Data [read-only]

For running some simple examples below, we will first create a subset of the HGDP sample that is restricted to only European populations. The European populations in the HGDP have the labels 'Adygei', 'Basque', 'French', 'Italian', 'Orcadian', 'Russian', 'Sardinian' and 'Tuscan', so we create a list of the 156 individuals matching these labels using an `awk` command, and then use `plink`'s `--keep` option to make a new dataset with output prefix 'H938_Euro'.

```
awk '$3=="Adygei"||$3=="French_Basque"||$3=="French"|| $3=="North_Italian"||$3=="Orcadian"||$3=="Russia
$3=="Sardinian"||$3=="Tuscan" {print $0}' data/H938.clst.txt > data/Euro.clst.txt

DATADIRECTORY=data
gzip -d $DATADIRECTORY/H938.bed.gz
plink --noweb --bfile $DATADIRECTORY/H938 --keep data/Euro.clst.txt --make-bed --out data/H938_Euro
```

Filter out SNPs to remove linkage disequilibrium (LD) [read-only]

SNPs in high LD with each other contain redundant information. This is not problematic in and of itself but there is the potential for some regions of the genome to have a disproportionate influence on the results and thus distort the representation of genome-wide structure. A nice empirical example of the problem is in Figure 5 of Tian et al [@Tian2008aa], where PC2 of the genome-wide data is shown to be reflecting the variation in a 3.8Mb region of chromosome 8 that is known to harbor an inversion. A standard approach to address this issue is to filter out SNPs based on pairwise LD to produce a reduced set of more independent markers. Here we use `plink`'s commands to produce a new LD-pruned dataset with output prefix `H938_Euro.LDpruned` and limiting our analyses to just the automsomes. The approach considers a chromosomal window of 50 SNPs at a time, and for any pair whose genotypes have an association r^2 value greater than 0.1, it removes a SNP from the pair. Then the window is shifted by 10 SNPs and the procedure is repeated:

```
plink --noweb --bfile data/H938_Euro --autosome --indep-pairwise 50 10 0.1

plink --noweb --bfile data/H938_Euro --autosome --extract plink.prune.in --make-bed --out data/H938_Eur
```

[Advanced note: For particularly sensitive results, we recommend additional rounds of SNP filtering based on observed principal component loadings and/or population differentiation statistics. For example, a robust approach is to filter out large windows around any SNP that has a high PC loading in an initial PCA analysis, see @Novembre08b].

Exploring Hardy-Weinberg predictions

In this section, you will assess how well the genotypes at each SNP fit Hardy-Weinberg proportions. Given the population structure in this dataset, we might have a chance to observe the -Wahlund effect- in which the observed propotion of heterozygotes is less than expected due to hidden population structure (thought of another way, each sub-population is in a sense inbred, lowering the heterozygosity). One might also see the evidence of low-quality data, where SNPs wildly depart from Hardy-Weinberg proportions (e.g. 100% heterozygosity)

Using plink to get basic gentoype counts

To begin, run the plink `--hardy` command. It formally tests for departures from Hardy-Weinberg proportions. To keep the analysis simple, use the `--chr` command to limit the analysis to SNPs on chromosome 15.

```
plink --bfile data/H938_Euro.LDprune --hardy --chr 15 --out out/H938_Euro.LDprune
```

Next, you are going to read the output of this command into R and visually explore the predictions of Hardy-Weinberg proportions.

Plotting in R

Now, we will use the following R function to make a plot of the frequency of each genotype relative to its allele frequency (note: actually the code plots only a sampling of 3000 SNPs to avoid an overloaded plot). (Credit to Graham Coop for this specific function - see his lab blog http://gcbias.org/ as a reference).

```
ggplot.geno.vs.HW<-function(file,title=""){

      #read in the HW file from plink
      plink.hwe<-read.table(file,as.is=TRUE,header=TRUE)

        names(plink.hwe)<-c("CHR","SNP","TEST","A1","A2",
          "GENO","oHET","eHET","Pval")
        counts<-sapply(plink.hwe$GENO,function(x){as.numeric(strsplit(x,"/")[[1]])})
        counts<-t(counts)
        nObs<-rowSums(counts)
        geno.freq<-counts/nObs

        MAF <- (geno.freq[,1]+.5*geno.freq[,2])
      g <- data.frame(nObs,MAF,geno.freq)
      row.names(g)<-plink.hwe$SNP.id
      names(g)<-c('nObs','MAF','p11','p12','p22')
        gTidy <- pivot_longer(g,cols=c(p11,p12,p22),names_to="Genotype",values_to="Genotype.Proportion"

      head(gTidy)
      gp <- ggplot(gTidy) + geom_point(aes(x = MAF,
                                y = Genotype.Proportion,
                                color = Genotype,
                                shape = Genotype))+
        geom_point(aes(x=MAF,y=Genotype.Proportion,color=Genotype,shape=Genotype))+
        stat_function(fun=function(p) p^2, geom="line", colour="red",size=2.5) +
        stat_function(fun=function(p) 2*p*(1-p), geom="line", colour="green",size=2.5) +
        stat_function(fun=function(p) (1-p)^2, geom="line", colour="blue",size=2.5)
    }
```

Use this function now to make a plot…

```
gp <- ggplot.geno.vs.HW("out/H938_Euro.LDprune.hwe")
gp
```

Questions

1. Do the genotypic frequencies roughly follow the basic patterns expected for Hardy-Weinberg proportions (e.g. P_{AA} is approximately quadratic in p)?

2. Looking more carefully, is the HW prediction for the proportion of heterozygotes given allele frequency generally too high or too low relative to the empirically observed values? What might explain the deviation?
3. [SKIP] Now, go through the same steps for the full H938 set of plink files. Compare the deficiency in heterozygotes between the world-wide data and the European only data. In which is the deficiency smaller? Why might that be the case?

Allele frequency spectra

The allele frequency spectra is a count of the number of variant positions that have a particular allele frequency count (i.e. the "frequency of different frequecies"). This can be done using the `hist` function in R to make a histogram. A challenge is that there is a variable amount of missing data in the sample. As a way to avoid this issue without doing imputation or taking other steps, let's focus only on SNPs that are fully observed (i.e. the total counts of individuals = all 938 individuals for the full data problem).

Computing and plotting a MAF frequency spectra

```r
plot.MAF <- function(file){
  # Read in the HWE table and compute counts and allele frequencies
  hwe<-read.table(file,as.is=TRUE,header=TRUE)
  names(hwe)<-c("chr","SNP.id","which.inds","a1","a2","genotype","obs.het","exp.het","HWE.pval")
  counts<-sapply(hwe$genotype,function(x){as.numeric(strsplit(x,"/")[[1]])})
  counts<-t(counts)
  tot.counts<-rowSums(counts)
  allele.counts<-(2*counts[,1]+counts[,2])

  # Flip allele counts so that we are sure we always have the minor
  # allele frequency
  # (Note: this uses a trick based on boolean math where true/false = 1/0).
  counts.maf = allele.counts*(allele.counts<=2*tot.counts-allele.counts) + (2*tot.counts-allele.counts)

  # Set the number of individuals by looking at the sites w/ the most
  # observed data
  n=max(tot.counts)

  # Make the plot but filter on using only sites with fully observed
  # data (i.e. totcounts==n)
  hist(counts.maf[tot.counts==n],
   xlab="Minor allele count",
   ylab="# of SNPs",
   main="Allele frequency spectra",breaks=n)

  # Plot the expected minor allele frequency spectra for the standard
  # neutral model (i.e. constant size population, all loci neutral)
  # To do so we compute, Watterson's estimator of Theta
  S=sum(tot.counts==n & counts.maf>0)
  thetaW=S/sum(1/seq(1,2*n-1))
  # Which determines the expected AFS
  expectedAFS=(1/seq(1,n)+1/(n*2-seq(1,n))) * thetaW
  # And then plot
  lines(seq(1,n),expectedAFS,col=2)
  # Note: This adds a red line displaying the expected AFS shape
  # controlled to match the data w.r.t to Watterson's Theta (i.e. the total number of SNPs).
}
```

```
plot.MAF("out/H938_Euro.LDprune.hwe")
```

Questions

1. The distribution of MAF's does not have follow the shape one would expect for neutral varaints in a constant-sized population (the red-line). Is there an excess of rare or common variants relative to the expectation?
2. What is at least one plausible explanation for the departures? (Hint: Consider the data collection strategy and filtering here).

Follow-up Activities (Optional)

1. Carry out the same exercise data with a sequencing data set (for example, 1000 Genomes data) or exome chip data. Now do you see an excess of rare or common variants?

Running `ADMIXTURE`

Though structure within Europe is subtle, we can run the program `admixture` on our set of 8 European sub-populations to explore the structure within it.

The `ADMIXTURE` software (v 1.3.0 here) comes as a pre-compiled binary executable file for either Linux or Mac operating systems. To install, simply download the package and move the executable into your standard execution path (e.g. '/usr/local/bin' on many linux systems). Once installed, it is straightforward to run `ADMIXTURE` with a fixed number of source populations, commonly denoted by K. For example, to get started let's run ADMIXTURE with K=8 as there are 8 distinctly labelled populations in the dataset (Note: this command takes about 15 minutes to run):

```
admixture data/H938_Euro.LDprune.bed 8
```

`ADMIXTURE` is a maximum-likelihood based method, so as the method runs, you will see updates to the log-likelihood as it converges on a solution for the ancestry proportions and allele frequencies that maximize the likelihood function. The algorithm will stop when the difference between successive iterations is small (the 'delta' value takes a small value). A final output is an estimated F_{ST} value [@Kent09] between each of the source populations, based on the inferred allele frequencies. These estimates reflect how differentiated the source populations are, which is important for understanding whether the population structure observed in a sample is substantial or not (values closer to 0 reflect less population differentiation). For this dataset, it should take about 75 iterations to finish.

After running, `ADMIXTURE` produces two major output files. The file with suffix `.P` contains an $L \times K$ table of the allele frequencies inferred for each SNP in each population. The file with suffix `.Q` contains an $N \times K$ table of inferred individual ancestry proportions from the K ancestral populations, with one row per individual.

For our example dataset with K=8, this will be a file called `H938.LDpruned.8.Q`. This file can be used to generate a plot showing individual ancestry. In `R`, this can be done using the following commands:

```
library(RColorBrewer)
clst_file = "data/Euro.clst.txt"
fam_file = "data/H938_Euro.fam"
q_file = "out_prep/H938_Euro.LDprune.8.Q"
#Read in the Qmatrix
Q = read.table(q_file)
```

```r
Qmat = as.matrix(Q)

# To be able to label the graph we read in a
# .clst file with population "cluster" labels for each indiv
clst = read.table(clst_file)

# And a fam file from the plink data
fam = read.table(fam_file)

# Use the match function to link the family ids with the cluster id
clst_unord=clst$V3[match(fam$V2,clst$V2)]

# Re-order alphabetically
ordered_indices=order(clst_unord)
QmatO=Qmat[ordered_indices,]

# Compute where we will place the population labels in the barplot
n=length(ordered_indices)
clst_ord=clst_unord[ordered_indices]
breaks=c(0,which(clst_ord[1:(n-1)]!=clst_ord[2:n]),n)
nbrks=length(breaks)
midpts=(breaks[1:(nbrks-1)]+breaks[2:nbrks])/2
```

```r
# Make the barplot
barplot(t(QmatO),col=brewer.pal(8,"Set1"),border=NA,space=0)
abline(v=breaks,lwd=2)
mtext(levels(clst_ord),side=1,at=midpts,las=2)
```

Each thin vertical line in the barplot represents one individual and each color represents one inferred ancestral population. The length of each color in a vertical bar represents the proportion of that individual's ancestry that is derived from the inferred ancestral population corresponding to that color. The above image suggests there are some genetic clusters in the data, but it's not a well organized data display.

Questions

1. Are individuals from the population isolates (Adygei, French_Basque, Orcadian, and Sardinian) inferred to have distinct ancestral populations?
2. Are the Tuscan and North_Italian individuals completely distinguished as being from distinct populations? How about these two relative to French?
3. Which sampled population would seem to have the most internal population structure?

Follow-up Activities (Optional)

1. Run the method with K=6 and describe the results.
2. Refilter the dataset to create a worldwide sampe and run it with K=6 or K=7.

Exploring ADMIXTURE results with the PONG software [Optional]

To improve the visualization, one can use a package dedicated to plotting ancestry proportions [@Rosenberg04; @Kopelman15; @Behr16]. Here we use a post-processing tool `pong` [@Behr16], which visualizes individual ancestry with similarity between individuals within clusters. You will most likely want to install 'pong' on a local machine as it initializes a local web server to display the results.

To run pong requires setting up a few files: 1) an ind2pop file that maps individuals to populations; 2) a Qfilemap file that points pong towards which '.Q' files to display; These are easy to build up from the command-line using the Euro.clst.txt file we built above, and an awk command to output tab-seperated text to a file with the Qfilemap suffix added to whatever file prefix we're using to organize our runs:

```
cut -d' ' -f3 data/Euro.clst.txt > out/H938_Euro.ind2pop
FILEPREFIX=H938_Euro.LDprune
K=8

awk -v K=$K -v file=$FILEPREFIX 'BEGIN{ \
  printf("ExampleRun\t%d\t%s.%d.Q\n",K,file,K)
}' > out/$FILEPREFIX.Qfilemap
```

Note when building the .Qfilemap one needs to use tabs to seperate the columns for pong to read the file correctly.

Then to run pong, we use following command:

```
pong -m out/H938_Euro.LDprune.Qfilemap -i out/H938_Euro.ind2pop
```

and open a web browser to http://localhost:4000/ to view the results.

From this visualization we can see the admixture model fits most individuals of the Adygei, Orcadian, and Russian samples as being derived each from a single source population (represented by orange, green, and blue respectively); the Sardinian and the French Basque samples are modeled as comprising individuals from two source populations each (yellow/red for Sardinia, brown/pink for French Basque); and the French, Tuscan, and North Italian samples are generally estimated to have a majority component of ancestry from a single source population ('purple') though with notable admixture with other sources. A first conclusions is that the population labels do not capture the complexity of the population structure. There is apparent cryptic structure within some samples (e.g., Sardinia) and minimal differentiation between other samples (North Italian and Tuscan samples for instance).

We can run ADMIXTURE multiple times with the -s flag specifying random seeds. This is useful determining if different ADMIXTURE runs find a single global optima or if there is evidence for multimodality in the solutions. Pong summarizes modality and provides a `check to highlight multimodality` checkbox whitening populations with ancestry matrices agreeing with the modal solution. One can also click on and visualize only one cluster.

(NOTE: To save time, use the exisiting results files and skip this next code block which runs admxiture several times.)

```
prefix=H938_Euro.LDprune
K=8

# Run admiixture multiple times with K fixed
for r in {1..10}
do
  admixture -s ${RANDOM} data/H938_Euro.LDprune.bed $K
  mv out/${prefix}.${K}.Q out/${prefix}.K${K}r${r}.Q
done

# create a pong parameter file
for r in {1..10}
do
  awk -v K=$K -v r=$r -v file=${prefix}.K${K}r${r} 'BEGIN{ \
    printf("K%dr%d\t%d\t%s.Q\n",K,r,K,file)
  }' >> out/${prefix}.multiplerun.Qfilemap
done
```

Check the outputs via pong…

```
pong -m out/${prefix}.multiplerun.Qfilemap -s .99 -i out/H938_Euro.ind2pop
```

Here inferred ancestry components for North Italian are different across multiple runs of ADMIXTURE.

Considering different values of K

In a typical analysis, one typically wants to explore the sensitivity of the results to the choice of K. One approach is to run ADMIXTURE with various plausible values of K and compare the performance of results visually and using cross-validation error rates. Here is a piece of `bash` command-line code that will run `ADMIXTURE` for values of K from 2 to 12, and that will build a file with a table of cross-validation error rates per value of K.

(NOTE: To save time, use the exisiting results files and skip this next code block which runs admxiture several times.)

```
# Run for different values of K
prefix=H938_Euro.LDprune
Klow=1
Khigh=12

for ((K=$Klow;K<=$Khigh;K++)); \
do
  admixture --cv data/$prefix.bed $K | tee log.$prefix.${K}.out;
done
```

Then let's compile results on cross-validation error across values of K:

```
prefix=H938_Euro.LDprune
Klow=1
Khigh=12

echo '# CV results' > $prefix.CV.txt
for ((K=$Klow;K<=$Khigh;K++)); do
  awk -v K=$K '
    $1=="CV"{print K,$4}' out/log.$prefix.$K.out >> out/$prefix.CV.txt;
done
```

and build a Qmapfile for pong:

```
prefix=H938_Euro.LDprune
Klow=1
Khigh=12

echo '# Qfilemap' > out/$prefix.Qfilemap
for ((K=$Klow;K<=$Khigh;K++)); \
do
  awk -v K=$K -v prefix=$prefix 'BEGIN{ \
    printf("K%d\t%d\t%s.%d.Q\n",K,K,prefix,K)
  }' >> out/$prefix.Qfilemap;
done
```

Now let's inspect the outputs. First let's make a plot of the cross-validation error as a function of K:

```
tbl = read.table("out_prep/H938_Euro.LDprune.CV.txt")
names(tbl) <- c( "K", "Error")
```

```
ggplot(data = tbl, aes(K,Error)) + geom_point() + geom_line() + xlab("K") + ylab("Cross-validation erro
```

What do you see for the cross-validation error versus K? Usually, the cross-validation error decreases with K and we are interested in when there is no more improvement. This dataset is unique though. The cross-validation error suggests a single source population can model the data adequately and larger values of K lead to over-fitting. Let's now inspect the 'pong' outputs.

```
pong -m out/H938_Euro.LDprune.Qfilemap -i out/H938_Euro.ind2pop
```

Here we see how as K increases through to $K = 6$, the Sardinian, French Basque, Adygei, and Russian samples are modeled as descended from unique sources, at $K = 7$ and $K = 8$ structure within the Sardinian and Basque samples is revealed, though the Basque sub-structure is not stable at higher values of K. The values of $K = 9$ and above make increasingly finer scale divisions that are difficult to interpret.

Overall, it's worth noting that the visual inspection of the results suggests several "real" clusters in the data, though in this case the cross-validation supports a value of $K = 1$. This highlights a long-standing known issue with admixutre modeling: the selection of K is a difficult problem to automate in a way that is robust. It's also worth noting that population structure in this European dataset is very subtle in absolute terms (e.g. recall the Hardy-Weinberg plots), so it's perhaps not surprising the cross-validation does not strongly favor higher values of K.

Some advanced options

Running `ADMIXTURE` with the `-B` option provides estimates of standard errors on the ancestry proportion inferences. The `-l` flag runs `ADMIXTURE` with a penalized likelihood that favors more sparse solutions (i.e., ancestry proportions that are closer to zero). This is useful in settings where small, possibly erroneous ancestry proportions may be overinterpreted. By using the `-P` option, the population allele frequencies inferred from one dataset can be provided as input for inference of admixture proportions in a second dataset. This is useful when individuals of unknown ancestry are being analyzed against the background of a reference sample set. Please see the `ADMIXTURE` manual for a complete listing of options and more detail, and we encourage testing these options in test datasets such as the one provided here.

PCA

Principal components analysis is a commonly used way to investigate population structure in a sample (though it is also sensitive to close relatedness, batch effects, and long runs of LD, and you should watch for these potential effects in any analysis). Here you will run PCA on the Euroepan subset of the data with the LD pruned data.

PCA with SMARTPCA

Comparing ADMIXTURE and PCA results often helps give insight and confirmation regarding population structure in a sample. To run PCA, a standard package that is well-suited for SNP data is the `smartpca` package maintained by Nick Patterson and Alkes Price (at http://data.broadinstitute.org/alkesgroup/EIGENSOFT/). To run it, we first set-up a basic `smartpca` parameter file from the command-line of a `bash` shell:

```
PREFIX=H938_Euro.LDprune
echo genotypename: data/$PREFIX.bed > out/$PREFIX.par
echo snpname: data/H938_Euro.LDprune.bim >> out/$PREFIX.par
echo indivname: data/H938_Euro.LDprune.PCA.fam >> out/$PREFIX.par
echo snpweightoutname: out/H938_Euro.LDprune.snpeigs >> out/$PREFIX.par
echo evecoutname: out/H938_Euro.LDprune.eigs >> out/$PREFIX.par
echo evaloutname: out/H938_Euro.LDprune.eval >> out/$PREFIX.par
echo phylipoutname: out/H938_Euro.LDprune.fst >> out/$PREFIX.par
echo numoutevec: 20 >> out/$PREFIX.par
echo numoutlieriter: 0 >> out/$PREFIX.par
echo outlieroutname: out/H938_Euro.LDprune.out >> out/$PREFIX.par
echo altnormstyle: NO >> out/$PREFIX.par
echo missingmode: NO >> out/$PREFIX.par
echo nsnpldregress: 0 >> out/$PREFIX.par
echo noxdata: YES >> out/$PREFIX.par
echo nomalexhet: YES >> out/$PREFIX.par
```

This input parameter file runs `smartpca` in its most basic mode (i.e. no automatic outlier removal or adjustments for LD - features which you might want to explore later).

As a minor issue, `smartpca` ignores individuals in the `.fam` file if they are marked as missing in the phenotypes column. This `awk` command provides a new `.fam` file that will automatically include all individuals.

```
awk '{print $1,$2,$3,$4,$5,1}' data/H938_Euro.LDprune.fam > data/H938_Euro.LDprune.PCA.fam
```

Now run `smartpca` with the following command (takes about 10 seconds).

```
bin/smartpca -p ./out/H938_Euro.LDprune.par
```

You will find the output files in the `out` sub-directory as specified in the parameter file.

For use in R, let's read in the PCA data as a dataframe…

```
prefix = "out/H938_Euro.LDprune"
nPCs = 20

# Read in eigenvectors file
PCA = read.table(paste0(prefix,".eigs"))
names(PCA) = c("ID",paste0("PC",(1:nPCs)),"CaseControl")
PCA = PCA[,1:(nPCs+1)]

# Read in eigenvalues file
eig.val = sqrt(unlist(read.table(paste0(prefix,".eval")))[1:nPCs])
sum.eig = sum(unlist(read.table(paste0(prefix,".eval"))))

# Read in snp weightings matrix
snpeigs = read.table(paste0(prefix,".snpeigs"))
names(snpeigs) = c("ID","chr","pos",paste0("PC",(1:nPCs)))
snpeigs$chr = factor(snpeigs$chr)
rownames(snpeigs) <- snpeigs$ID; snpeigs = snpeigs[,-1]

# Clean up IDs
tmp = unlist(sapply(as.character(PCA$ID),strsplit,":")) # Note smartpca pushes the plink family and ind
ids = tmp[seq(2,length(tmp),by=2)]
PCA$ID = ids

# Read in the group/cluster labels and assign abbreviation labels
clst = read.table("data/Euro.clst.txt")
clst_unord = clst$V3[match(ids,clst$V2)]  # Order them to match the ids of PCA object
abbrev = substr(clst_unord,1,2) # Build simple-minded abbreviations
PCA = as.data.frame(PCA)
PCA = cbind(PCA,clst_unord,abbrev)
```

```r
names(PCA)[nPCs+2] <- "sample"
```

Simple plots of the results

And make a simple plot…

```r
ggplot(data = PCA, aes(x = PC1, y = PC2, label = abbrev, color = abbrev)) + geom_text()
```

This plot is nice but the legend is awkward, the French and French Basque were collapsed into "Fre", and it would be nice to plot the medians for each groups and several other features of the PCA results…

Plotting PCA results with PCAviz

The PCAviz package can be found at https://github.com/NovembreLab/PCAviz. It provides a simple interface for quickly creating plots from PCA results. It encodes several of our favored best practices for plotting PCA (such as using abberviations for point characters and plotting median positions of each labelled group). (If you didn't already, see the top of this file for installation help with PCAviz).

We first need to create a PCAviz object. We use the `PCA` dataframe defined above. Crucially, the sample IDs need to be in the next column after the actual numeric PC values.

```r
library(PCAviz)

# Build the PCAviz object
hgdp <- pcaviz(dat = PCA, sdev=eig.val, var= sum.eig, rotation = snpeigs)
# PCAviz has an automated function for building abbreviations
hgdp <- pcaviz_abbreviate_var(hgdp,"sample")
```

The default plot method with PCAviz generates a plot showing each individual sample's position in the PCA space and the median position of each labelled group in PCA space:

```r
plot(hgdp)
```

And we can make a set of plots…

```r
# Customize
geom.point.summary.params = list(shape = 16,stroke = 1,size = 7,
                                 alpha = .7)
theme_local <- function () {
  theme_cowplot(font_size=10)
}

plot0 <- plot(hgdp)
plot_legend <- get_legend(plot0)

plot1 <- plot(hgdp,coords = paste0("PC",c(1,2)), color = "sample",
              geom.point.summary.params = geom.point.summary.params,
              preserve.scale = F ,show.legend = FALSE) + theme_cowplot(font_size=10)

plot2 <- plot(hgdp,coords = paste0("PC",c(3,4)), color = "sample",
              geom.point.summary.params = geom.point.summary.params,
              preserve.scale = F ,show.legend = FALSE) + theme_cowplot(font_size=10)

plot3 <- plot(hgdp,coords = paste0("PC",c(5,6)), color = "sample",
              geom.point.summary.params = geom.point.summary.params,
               preserve.scale = F ,show.legend = FALSE) + theme_cowplot(font_size=10)

plot_grid(plot1, plot2, plot3, plot_legend, labels = list("A","B","C",""))
```

First one may notice several populations are separated with PC1 and PC2, with the more isolated populations being those that were most distinguished from the others by `ADMIXTURE`. PC4 distinguishes a subset of Orcadian individuals and PC5 distinguishes two Adygei indiviuals. PC6 corresponds to the cryptic structure observed within Sardinians in the `ADMIXTURE` analysis.

As an alternative visualization it can be helpful to see the distribution of PC coordinates per population for each labelled group in the data:

```
pcaviz_violin(hgdp,pc.dims =paste0("PC", c(1:6)),plot.grid.params = list(nrow = 3))
```

As mentioned above in the section on LD, it is useful to inspect the PC loadings to ensure that they broadly represent variation across the genome, rather than one or a small number of genomic regions [@Duforet-Frebourg16]. SNPs that are differentiated in the same direction as genome-wide structure can show high loadings, but what is particularly pathological is if the only SNPs that show high loadings are all concentrated in a single region of the genome, as might occur if the PCA is explaining genomic structure (such as an inversion) rather than population structure.

```
for (i in 1:5){
  plotname <- paste("plot",i,sep="")
  plot <- pcaviz_loadingsplot(hgdp,pc.dim=paste0("PC",i),
                              min.rank=0.8,gap=200,color = 'chr',
                              geom.point.params = list(show.legend = FALSE)) +
          xlab("SNPs") + ylab(paste0("PC",i," loading"))
  assign(plotname,plot)
}
# pull out common legend
plot <- pcaviz_loadingsplot(hgdp,pc.dim=paste0("PC",1),
                            min.rank=0.8,gap=200,color = 'chr') +
        guides(color = guide_legend(nrow=2, byrow=TRUE)) +
        theme(legend.position="bottom", legend.justification = "center")
plot_legend <- get_legend(plot)
# plot loadings
prow <- plot_grid(plot1,plot2,plot3,plot4,plot5, nrow=5, align="vh")
plot_grid(prow, plot_legend, ncol = 1, rel_heights = c(1, .2))
```

The proportion of total variance explained by each PC is a useful metric for understanding structure in a sample and for evaluating how many PCs one might want to include in downstream analyses. This can be computed as $\lambda_i / \sum_k \lambda_k$, with λ_i being eigenvalues in decreasing order, and is plotted below:

```
screeplot(hgdp,type='pve')  + ylim(0,0.018) +
      theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
      theme(axis.line = element_line(size = 1, linetype = "solid"))
```

The results show that the top PCs only explain a small fraction of the variance (<1.5%) and that after about $K = 6$ the variance explained per PC becomes relatively constant; roughly in line with the visual inspection of the `admixture` results that revealed $K = 6$ may be reasonable for this dataset.

Questions (Part A)

1. Are individuals from the population isolates (Adygei, French Basque, Orcadian, and Sardinian) clearly separated by at least one of the top PCs you've plotted?

2. Are the Tuscan and North Italian individuals completely separated in at least one of the top PCs you've plotted?
3. Do any of the PCs replicate the structure within Sardinia that was inferred in the admixture analysis above?
4. Do the admixture results and PCA results seem to agree with regards to the relationship of the French, Italian, and Orcadian samples?

Questions (Part B)

1. Based on the proportion of the variation explained - there are a number of PCs that stand out as being more relevant for explaining variation. About how many?
2. From your plots of PC1-PC8 you should see that the lower PCs seem to be picking up individual-level structure, isolating single individuals. At what PC does this first happen?

Follow-up Activities (Optional)

1. Read in the `.snpeigs` file and plot the weight of each SNP along the genome for each of the top PCs. If the spatial clustering of the weights is not distributed genome-wide it may indicate a PC is identifying some local genomic structure rather than genome-wide structure. For example, in many European samples, a PC might indentify a common inversion polymorphism on chr 8p23 or 17q.
2. Run PCA on the worldwide pruned LD data and inspect the results.

Discussion

Our protocol above is relatively straightfoward and presents the most basic implementation of these analyses. Each analysis sofware (`ADMIXTURE` and `smartpca`) and each visualization package (`pong` and `PCAviz`) contain numerous other options that may be suitable for specific analyses and we encourage the readers to spend time in the manuals of each. Nonetheless, what we have presented is a useful start and a standard pipeline that we use in our research.

Two broad perspecives we find helpful useful to keep in mind are: 1) How the admixture model and PCA framework are related to each other indirectly as different forms of sparse factor analyss [@Engelhardt10]; 2) How the PCA framework in particular can be considered as a form of efficient data compression. Both of these perspectives can be helpful in interpreting the outputs of the methods and for appreciating how these approaches best serve as helpful visual exploratory tools for analyzing structure in genetic data. These methods are ultimately relatively simple statistical tools being used to summarize complex realities. They are part of the toolkit for analysis, and often are extremely useful for framing specific models of population structure that can be further investigated using more detailed and explicit approaches.

Demonstration of spurious association due to population structure [Time-permitting]

One consequence of population structure is that it can cause spurious associations with phenotypes. In this final exercise you will generate a phenotype that has no dependence on genetics - but that does depend on population membership (imagine a trait determined by diet or some other non-genetic factor that varies among populations). You will try to map it and inspect whether the resulting association test p-values are consistent with the null of no genetic effects.

Generate a phenotype [read-only]

First - let's make a file where each individual is assigned a somewhat arbitrary base phenotypic value given by what population they are from (Adygei = 5, French Basque = 2, North Italian = 5, Tuscan =5, Sardinian = 0; French = 8; Orcadian = 10, Russian=4)

```
awk '$3=="Adygei"{print $1,$2,5}\
     $3=="French Basque"{print $1,$2,2}\
     $3=="North Italian"{print $1,$2,5}\
     $3=="Tuscan"{print $1,$2,5}\
    $3=="Sardinian"{print $1,$2,0}\
    $3=="French"{print $1,$2,8}\
    $3=="Orcadian"{print $1,$2,10}\
     $3=="Russian"{print $1,$2,4}' \
    data/Euro.clst.txt > out/pheno.base.txt
```

Now, using R, let's add some variation around this base value to produce individual-level phenotypes. Note: Nothing genetic about this phenotype!

```
pheno.base=read.table("out/pheno.base.txt")
# Scale the base phenotype to mean 0, sd 1
pheno.base.scale=scale(pheno.base$V3)
# Add some normally distributed noise (with as much variance as the base phenotype itself already has)
pheno.sim=rnorm(length(pheno.base$V3),mean=pheno.base$V3,sd=1)
# Output the phenotype to a file
write.table(cbind(pheno.base[,1:2],pheno.sim),"out/pheno.sim.txt",quote=FALSE,row.names=FALSE,col.names
```

We will use the `pheno.sim.txt` as our phenotype file for mapping.

Map the trait using plink mapping functions

The `--assoc` command in plink will produce p-values for a basic regression of phenotype on additive genotypic score. (< 10 sec)

```
plink --bfile data/H938_Euro --pheno out/pheno.sim.txt --assoc --out out/H938_Euro_sim.pheno --autosome
```

Exploring the results: A Manhattan plot

Read in the plink results contained in the `.qassoc` output file and make a Manhattan plot of the results.

```
qassoc=read.table("out/H938_Euro_sim.pheno.qassoc",header=TRUE)

# Make a Manhattan plot
# First set-up a plot of the right size
plot(1:length(qassoc$SNP),type="n",xlab="SNP index",ylab="-log10(p-value)",ylim=c(3,max(-log10(qassoc$P
# Next add the points (note: we only plot points with
# -log10(p-value)>3 to minimze the number of points plotted)
plot.these=which(-log10(qassoc$P)>3)
points(plot.these,-log10(qassoc$P[plot.these]),col=1+qassoc[plot.these,"CHR"]%%2,pch=16)
# Put in a line for a Bonferroni correction (0.05 / length(qassoc$SNP)
abline(h=-log10(0.05/length(qassoc$SNP)),lty=2,col="gray")
```

Questions (Part A)

1. Which chromosomes locations would you be tempted to follow up here?

Inspect a table of the most extreme hits:

```r
options(width=100)
print(qassoc[head(order(qassoc$P),n=20),])
```

Questions (Part B)

1. The peak on chromosome 6 is near what famous region of the human genome?
2. The peak on chromosome 4 spans the gene for TLR6, a toll-like receptor involved in bacterial recognition that was noted as being highly differentied in Europe (i.e. high FST) by Pickrell et al (2009) in their analysis of this data. Why might a highly differentiated SNP show a stronger signal of spurious association than other SNPs?

Exploring the results: A quantile-quantile plot.

Use the following code in R to make a plot of the observed vs. expected p-values matched by quantile.

```r
# Read in the p-values
qassoc=read.table("out/H938_Euro_sim.pheno.qassoc",header=TRUE)
# Produce expected p-values from the null (i.e. perfectly uniformly
# distributed).
nTests=length(qassoc$SNP)
Unif=seq(1/nTests,1-1/nTests,length=nTests)

# Sort the -log10 p-values (i.e. match on quantile)
logUnifOrder=order(-log10(Unif),decreasing=TRUE)
SNPorder=order(-log10(qassoc$P),decreasing=TRUE)

# Plot the p-values against against each other (Note: we do for only
# the top 150K SNPs to make the number of points plotted smaller)
nPlot = 150e3  # Number of test we'll plot

df <- data.frame("exp" = -log10(Unif[logUnifOrder][1:nPlot]),
                 "obs" = -log10(qassoc$P[SNPorder][1:nPlot]))

ggplot(df,aes(exp,obs)) + geom_point() +
  xlab("-log(p) Expected") + ylab("-log(p) Observed") +
  geom_hline(yintercept = -log10(0.05/length(qassoc$SNP)),
             color = "grey",linetype = "dashed") +
  geom_abline(slope = 1)
```

Questions (Part C)

1. Does there appear to be evidence for a genome-wide inflation of p-values?

Follow-up Activities

1. Simulate p-values from the null uniform distribution and draw a qq-plot.
2. Consider how genomic control be applied in this situation to control population stratficiation (See Price et al. Nature Reviews Genetics to get a better idea of genomic control)

3. Use the PCs you've computed already (or plink's MDS functions) to rerun the association test controlling for population stratification and plot a QQ-plot