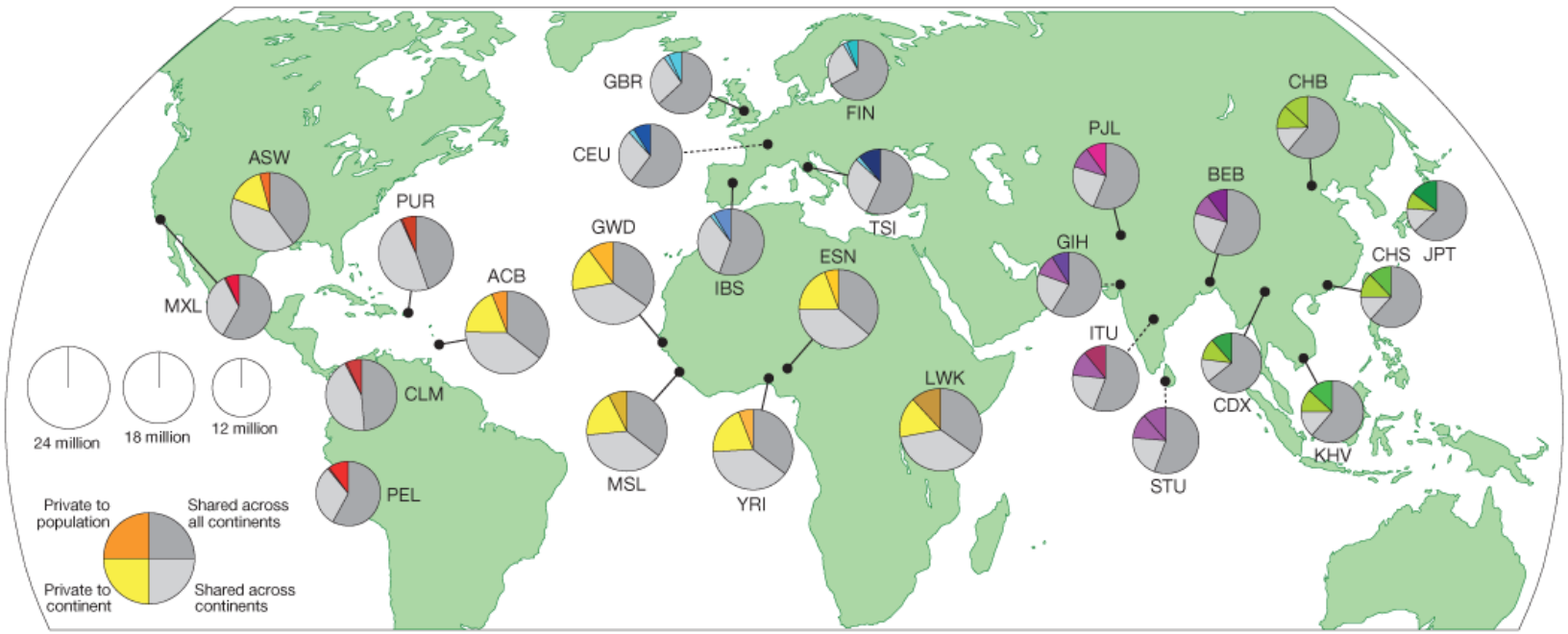
Analysis of Genetic Data 1: Inferring Population Structure

Peter Carbonetto

Research Computing Center and the Dept. of Human Genetics
University of Chicago



Aims of workshop

1. Work through the steps of a basic population structure analysis in human genetics, starting with the **“raw” source data**, and ending with a **visualization of population structure** estimated from the genetic data.
2. Understand how large genetic data sets are commonly represented in computer files.
3. Use command-line tools to manipulate genetic data.
4. Learn through “live coding”—this includes learning from our mistakes!

Our research task

We will simulate a population structure analysis commonly done in human genetics research.

1. We have a small genetic data set (5 genotype samples).
2. We would like to uncover population structure in this small sample.
3. To do so, we infer population structure *relative to a community-accepted reference*. We use the publicly available 1000 Genomes data as a reference.
4. We will use the most common statistical technique—Principal Components Analysis (PCA)—to expose, or “infer”, population structure from the genetic data.

It is your choice

You may. . .

1. Walk through the examples on the RCC cluster.
2. Walk through the examples on your laptop.
3. Pair with your neighbour.
4. Follow what I do on the projector.

I can't guarantee that all examples will work exactly the same on your laptop.

Software tools we will use today

1. PLINK
2. R
3. R packages `data.table`, `rsvd`, `ggplot2` and `cowplot`.
4. Basic shell commands such as `less` and `wc`.

Outline of workshop

1. Initial setup.
2. Download and prepare the genotype data.
3. Run PCA on the processed genotype data.
4. Visualize and interpret the PCA results.

Outline of workshop

1. **Initial setup.**
2. Download and prepare the genotype data.
3. Run PCA on the processed genotype data.
4. Visualize and interpret the PCA results.

Initial setup (part 1)

- WiFi
- Power outlets
- YubiKeys
- Helper(s)
- Pace, questions (e.g., keyboard shortcuts).

Initial setup (part 2)

If you are using the RCC cluster, set up your cluster computing environment:

1. Connect to midway2.
   - ▷ See `https://rcc.uchicago.edu/docs/connecting`.
2. Request 4 CPUs and 10 GB of memory on a midway2 compute node:

```
sinteractive --partition=broadwl \
  --time=3:00:00 --mem=10G \
  --cpus-per-task=4
```

Initial setup (part 3)

Download the workshop packet from GitHub.

- URL: `https://github.com/rcc-uchicago/genetic-data-analysis-1`

If you are using the RCC cluster, you can run these commands to download the workshop packet:

```
cd ~
git clone https://github.com/rcc-uchicago/
  genetic-data-analysis-1.git
```

If you are using the RCC cluster, also download the workshop packet on to your laptop. Browse **handout.pdf** using your favourite PDF viewer.

What's in the workshop packet

```
genetic-data-analysis-1
  /bin      # All executables are stored here.
  /code     # Source code used in analyses.
  /data     # "Raw" and processed data.
  /docs     # Additional workshop materials.
  /output   # All results are stored here.
```

Outline of workshop

1. Initial setup.
2. **Download and prepare the genotype data.**
3. Run PCA on the processed genotype data.
4. Visualize and interpret the PCA results.

Download 1000 Genomes data

We download the 1000 Genomes data from the European Bioinformatics Institute. Downloading may take 10–20 minutes—**hopefully you did this before the workshop!**

- Long URL: `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/supporting/hd_genotype_chip`
- Short URL: `http://bit.ly/2G7ZWYu`
- Download this file into the `data` folder: `ALL.chip.omni_broad_sanger_combined.20140818.snps.genotypes.vcf.gz`

On the RCC cluster, you can run these commands:

```
cd data
wget http://bit.ly/2C617oO -O 1kg.vcf.gz
```

Explore the VCF file

Let's run some simple shell commands to inspect the genotype data stored in the VCF file.

```
cd data
ls -lh 1kg.vcf.gz
zcat 1kg.vcf.gz | less -S # On Mac, use gzcat.
```

- VCF reference: `http://www.cog-genomics.org/plink2/formats#vcf`

VCF files: concepts

- The Variant Call Format (VCF) is a text format for storing many types of DNA variant data (e.g., SNPs, deletions, insertions), and for annotating these variants.
- It is one of the most commonly used data formats in genetics.
- It is not an efficient way to store genotype data (this is why we have compressed it).
- See also:
  - `https://vcftools.github.io`
  - `https://samtools.github.io/hts-specs`
  - `doi:10.1093/bioinformatics/btr330`

Download PLINK

We download the stable version of the PLINK software in the `bin folder`.

- URL: `http://www.cog-genomics.org/plink2`

On the RCC cluster, you can run these commands to download and test the 64-bit Linux binary for PLINK 1.9b5.2:

```
cd bin
wget http://bit.ly/2nUxOQX -O plink.zip
unzip plink.zip
./plink --version
```

Convert VCF to PLINK

Run this command to convert the genotypes from VCF to the PLINK text format. This may take a few minutes to complete.

```
cd data
../bin/plink --vcf 1kg.vcf.gz --recode \
  --chr 1-22 --allow-extra-chr \
  --geno 0.01 --out 1kg
```

- Command details:
  - ▷ Creates two new files: `1kg.map` and `1kg.ped`.
  - ▷ We retain only SNPs on (autosomal) chromosomes 1–22.
  - ▷ We remove any SNPs with >1% missing genotypes.
  - ▷ These steps are taken to simplify the analyses.

Explore PLINK files

Let's run some simple shell commands to inspect the 1000 Genomes genotype data stored in the PLINK files.

```
head 1kg.map
tail 1kg.map
wc -l 1kg.map
less -S 1kg.ped
wc -l -w 1kg.ped
```

Next, use these same commands to inspect the AffyMetrix Human Origins data stored in `origins.map` and `origins.ped` (they are already included in the git repository).

- Columns in `.map` file: (1) chromosome; (2) marker id; (3) genetic distance on chromosome, in cM; (4) base-pair position on chromosome.
- Columns in `.ped` file: (1) family id, (2) individual id, (3) father id, (4) mother id, (5) gender, (6) phenotype measurement, (7—) SNP genotypes.

PLINK files: concepts

- The most commonly used format for storing human genoytpe data.
- Less flexible than VCF.
- The PLINK text format is easy to view and manipulate with simple shell commands (e.g., `wc`, `grep`, `cat`, `cut`, `paste`).
- For long-term storage, use PLINK binary (`.bed`) format. It is much more efficient, but is not human readable.
- See: `http://www.cog-genomics.org/plink/1.9/formats#ped`

Prepare the 1000 Genomes data

To speed up the data processing steps, we convert to binary PLINK format, then we remove related samples (which standard population structure analyses are not designed to handle).

1. Convert to PLINK binary format:

```
cd data
../bin/plink --file 1kg --make-bed --out 1kg
```

2. Inspect the 3 new files: `1kg.bed`, `1kg.bim`, `1kg.fam`.
3. Remove 29 of 31 related samples:

```
cut -f 1 20140625_related_individuals.txt \
  > temp.txt
paste temp.txt temp.txt > samples.txt
../bin/plink --bfile 1kg --make-bed \
   --remove samples.txt --out 1kg_unrelated
```

Merge with Affymetrix Human Origins data (part 1)

In order to merge the data sets, we need to identify a set of SNPs that is common to both data sets (see below for details), and extract the genotypes for the common SNPs only.

1. Extract common SNPs from 1000 Genomes data:

```
cd data
../bin/plink --bfile 1kg_unrelated \
  --extract 1kg_origins_markers.txt \
  --make-bed --out 1kg_common
```

2. Extract common SNPs from Human Origins data, and remove 2 samples that are included in both data sets:

```
../bin/plink --file origins \
  --extract 1kg_origins_markers.txt \
  --remove dupids.txt --make-bed \
  --out origins_common
```

Merge with Affymetrix Human Origins data (part 2)

3. Merge the two data sets:

```
../bin/plink --bfile 1kg_common \
  --bmerge origins_common \
  --out 1kg_origins_combined
```

Optional exercise: It is good practice to double-check the output after each processing step. Use the `cut` and `diff` commands to compare the SNPs in `1kg_common` and `origins_common` and check that the SNPs are in the same order.

Prune SNPs in LD

Many basic population structure analyses (e.g., PCA) assume that the SNPs are independent. A common step is to "prune" SNPs that are strongly correlated with each other (*i.e.*, in linkage disequibilirium, or LD) to make analysis better supported.

```
cd data
../bin/plink --bfile 1kg_origins_combined \
  --indep-pairwise 1000 500 0.8
../bin/plink --bfile 1kg_origins_combined \
  --make-bed --extract plink.prune.in \
  --out 1kg_origins_pruned
```

Typically you will want to be more aggressive in pruning SNPs in LD.

Data preparation: take-home points

- VCFtools and PLINK have many commands for manipulating genotype data.
- For more specialized edits, you can go far with basic shell commands (e.g., `awk`, `cut`, `head`, `cat`, `paste`).
- Often the majority of the effort goes toward data processing. Careless data processing—or no data processing!—can lead to a poor quality analysis.
- It is very common to introduce errors when merging multiple data sets. Errors can be due to different genome assemblies, different allele encodings, different genotyping error rates, *etc.*
- The PLINK "merge" command will correct some of these errors, but not all of them.
- I was conservative in selecting SNP common to both data sets to avoid introducing errors. As a result, we lost a lot of data after merging.
- *Important:* Record all your data processing steps.

Outline of workshop

1. Initial setup.
2. Download and prepare the genotype data.
3. **Run PCA on the processed genotype data.**
4. Visualize and interpret the PCA results.

Run PCA on combined data set

Outline of the PCA analysis:

1. Convert genotype data to a numeric representation—a matrix.
2. Start up interactive R environment.
3. Load genotype matrix into R.
4. Fill in missing genotypes.
5. Compute PCs in R using the `rsvd` package.

Convert genotype data to a matrix

The input to PCA must be an $n \times p$ matrix, where n is the number of samples and p is the number of SNPs.

```
cd data
../bin/plink --bfile 1kg_origins_pruned \
  --recode A --out 1kg_origins_recoded
```

This command creates a new file, `1kg_origins_recoded.raw`.

Start up interactive R environment

Move to the `code` folder, and start up R. On the RCC cluster, run these commands:

```
pwd # Should be .../code
module load R/3.4.3
export OPENBLAS_NUM_THREADS=4 # Optional.
R --no-save
```

The third command dramatically speeds up the PCA using OpenBLAS multithreaded matrix operations. This is optional.

Load genotype matrix into R (part 1)

Before continuing, check your working directory:

```
getwd() # Should be .../code
```

I wrote a function to rapidly load the genotype matrix using function `fread` from the `data.table` package. If you are not using the RCC cluster, you may need to install this package.

```
# install.packages("data.table")
library(data.table)
source("geno.utils.R")
```

Load the genotype matrix into R:

```
geno <-
read.geno.raw("../data/1kg_origins_recoded.raw")
```

Load genotype matrix into R (part 2)

Run a few commands to inspect the genoytpe data, e.g.:

```
class(geno)
dim(geno)
geno[1:5,1:5]
```

Fill in missing genotypes

A problem: a small proportion (<1%) of the genotypes are missing:

```
100*mean(is.na(geno))
```

We need to fill in these missing genotypes. In this case, a reasonable choice is the mean genotype:

```
p <- ncol(geno)
for (j in 1:p) {
  i           <- which(is.na(geno[,j]))
  geno[i,j] <- mean(geno[,j],na.rm = TRUE)
}
```

Check that there are no missing genotypes:

```
sum(is.na(geno))
```

Compute PCs using rsvd package (part 1)

If you are not using the RCC cluster, you may need to install the rsvd package.

```
# install.packages("rsvd")
library(rsvd)
```

Use the `rpca` function to compute the first 10 PCs—that is, the 10 components that explain the most variation in the genotypes:

```
out.pca <- rpca(geno,k = 10,center = TRUE,
                scale = FALSE,retx = TRUE)
```

Compute PCs using rsvd package (part 2)

Let's take a quick look at the PCA results:

```
summary(out.pca)
pcs <- out.pca$x
colnames(pcs) <- paste0("PC",1:10)
head(pcs)
```

Assuming we didn't encounter problems in any of these steps, let's save the results of our analysis to the `output` folder.

```
save(file = "../output/1kg_origins_pca.RData",
     list = c("out.pca","pcs"))
```

PCA analysis: take-home points

- PCA requires a matrix (with no missing values), so the genotypes need to be encoded as numeric values.
- Other software deals more elegantly with missing data. Here it does not matter much.
- Not everyone agrees on the best numeric encoding of genotypes for PCA.
- See the `pca.sbatch` script in the `code` folder for an illustration of how to automate the steps of this analysis on the RCC cluster.

Outline of workshop

1. Initial setup.
2. Download and prepare the genotype data.
3. Run PCA on the processed genotype data.
4. **Visualize and interpret the PCA results.**

Visualize and interpret PCA results

We have now finished all the computationally intensive aspects of our analysis.

- Our final step is to create plots from the PCA results to gain insight into the genetic data.

Outline of the PCA visualization:

1. Set up R environment for plotting with `ggplot2`.
2. Create a basic PC plot.
3. Create a PC plot with population labels.
4. Add sample ids to the unlabeled samples in the PC plot.

Set up R environment for plotting with ggplot2

If you are using the RCC cluster, make sure you can display graphics in your current R session, e.g.,

```
plot(cars$dist,cars$speed)
```

You should see a scatterplot. If not, you should start a fresh R session in a new midway2 connection (**ThinLinc** is the safest approach). As a reminder, you will need to run these commands:

```
pwd # Should be .../code
module load R/3.4.3
R --no-save
```

Set up R environment for plotting with ggplot2

If you are not using the RCC cluster, you may need to install the `ggplot2` package (I also recommend the `cowplot` package).

```
# install.packages("gplot2")
# install.packages("cowplot")
library(ggplot2)
library(cowplot)
getwd() # Should be .../code
source("geno.utils.R")
```

Load the PCA results in case you don't already have them loaded:

```
load("../output/1kg_origins_pca.RData")
```

Create a basic PC plot

Use function `basic.pc.plot` to plot all the samples projected onto the first 2 PCs:

```
p <- basic.pc.plot(pcs,x = "PC1",y = "PC2",
                   size = 2)
print(p)
```

- You may want to adjust the `size` argument.
- To learn how ggplot2 is used to generate the plot, see the code in `geno.utils.R`.

This plot shows that there is clear structure in the data. But it is difficult to interpret this structure without additional information.

Create a PC plot with population labels (part 1)

To create this plot, we first need to load the 1000 Genomes population labels stored in `omni_samples.20141118.panel`:

```
labels.1kg <-
read.table("../data/omni_samples.20141118.panel",
           sep = " ",header = TRUE,as.is = "id")
```

Next, add a “label” column to the table of PC results:

```
pcs <- add.poplabels(pcs,labels.1kg)
head(pcs)
```

Use the `labeled.pc.plot` function (see `geno.utils.R`) to create the PC plot with labels:

```
p2 <- labeled.pc.plot(pcs,x = "PC1",y = "PC2",
                      label = "label",size = 2)
print(p2)
```

Create a PC plot with population labels (part 2)

Several interesting insights can be drawn from this plot—*discuss.*

- How would you explain in a concise, non-technical way the main demographic patterns captured by PCs 1 and 2?
- How well do these results agree with Supp. Fig. 4 of the 1000 Genomes paper (`doi:10.1038/nature11632`)?
- See `1kg.pop` in the `data` folder to help with interpreting these results.

In some parts of the plot, the samples are clustered closely together. To gain additional insight, it is helpful to zoom on the denser parts. This is easily done with ggplot2, e.g.,

```
p2 + xlim(c(-30,10)) + ylim(c(10,50))
```

Optional exercise: Investigate demographic patterns exposed by PCs 3 and 4.

Add sample ids to unlabeled samples in PC plot

The previous plot does not show the population labels for the 5 samples from the Human Origins data set (these are the 5 black circles in the plot). To interpret the PCA results for these 5 samples, we add sample ids to the 5 points.

```
p3 <- labeled.pc.plot2(pcs,x = "PC1",y = "PC2",
                       label = "label",size = 2)
print(p3)
```

- This creates a similar plot to before, but annotates the 5 Human Origins samples with their ids.
- Compare this PC projection for these 5 samples results against the sample information provided the Human Origins data file `affymetrix-human-origins.ind`. Are these results expected or surprising based on the provided population labels?

Save the PC plot

That was our final PC plot. Let's save our work as a PDF file using the `ggsave` function from the `ggplot2` package:

```
ggsave("../output/1kg_origins_pcs1+2.pdf",p3)
```

Visualizing and interpreting PCA results: take-home points

- PCA is the most common approach to infer population structure from genotype data.
- One reason PCA is so popular is that it can produce evocative visualizations of populations structure (see `doi:10.1038/nature07566` for a particularly famous example produced by University of Chicago researchers).
- However, there are many well-known pitfalls in interpreting the results of a PCA analysis applied to genetic data—proceed with caution!

Recap

- An effective analysis of genetic data requires a variety of programmings skills.
- We did not work with sequencing data in this workshop—genotype data from DNA sequence assays introduces many more complications!
- Please email me (`pcarbo@uchicago.edu`) with questions, or for advice on your analysis of genetic data.