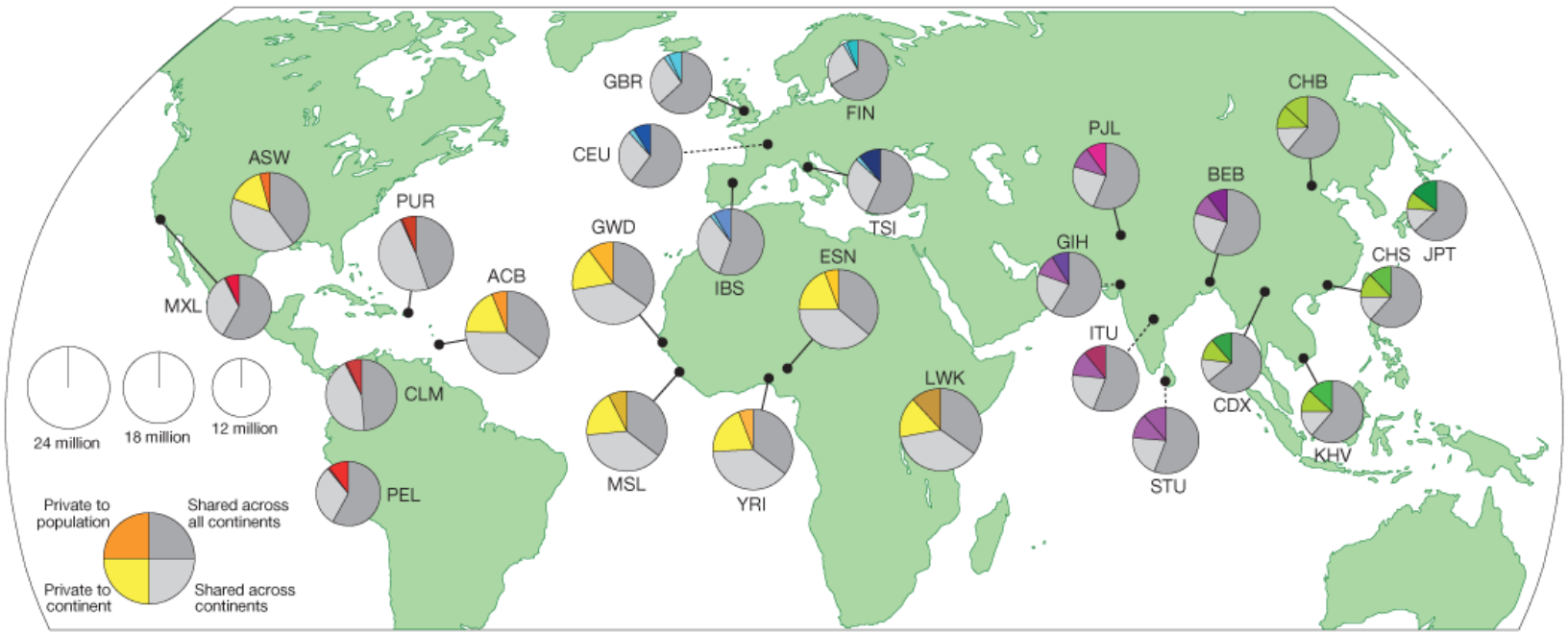
Analysis of Genetic Data 1: Inferring Population Structure

Peter Carbonetto

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



Workshop aims

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.

Workshop aims

- This is a *hands-on workshop*—you will get the most out of this workshop if you work through the exercises on your computer.
- All the examples are intended to run on the RCC cluster.
- You may try to run the examples on your laptop. *However, I cannot guarantee all examples will work the same on your laptop. My instructions will assume you are using midway2.*

Software tools we will use today

1. PLINK
2. R
3. Several R packages: data.table, rsvd, ggplot2 and cowplot.
4. Basic shell commands such as “cp” and “less”.

Our research task

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

1. We have collected genotype samples as part of our study.
2. We would like to uncover population structure in this sample.
3. We will use the most common statistical technique—Principal Components Analysis (PCA)—to investigate population structure from the genetic data.

Outline of workshop

- **Preliminaries**
- Programming challenges:
  1. Setting up your environment for genetic data analysis.
  2. Downloading and preparing genotype data for PCA.
  3. Run PCA on genotype data.
  4. Visualize and interpret PCA results.

Preliminaries

- WiFi.
- Power outlets.
- Reading what I type.
- Pace & questions (e.g., keyboard shortcuts).
- Yubikeys.
- What to do if you get stuck.

Preliminaries

- The workshop packet is a repository on GitHub. Go to:
  - ▷ github.com/rcc-uchicago/genetic-data-analysis-1
- Download the workshop packet to your computer.

What's included in the workshop packet

- **slides.pdf:** These slides.
- **slides.Rmd:** R Markdown source used to create these slides.
- **pca.R**, **pca.sbatch**: Example R and Slurm script implementing the PCA analysis.
- **functions.R**: Some R functions used to run PCA and plot the results.
- **omni_samples.20141118.panel**: 1000 Genomes population labels.
- **1kg.pop**: description of population labels.
- **20140625_related_individuals.txt**: 31 closely related samples.

Outline of workshop

- Preliminaries
- Programming challenges:
    1. **Setting up your environment for genetic data analysis.**
    2. Downloading and preparing genotype data for PCA.
    3. Run PCA on genotype data.
    4. Visualize and interpret PCA results.

Challenge #1: Setting up your HPC environment

- Aim: Configure your HPC environment for the next programming challenges.
- Steps:
  1. Connect to midway2.
  2. Download workshop packet.
  3. Install PLINK.
  4. Download 1000 Genomes data.
  5. Connect to a midway2 compute node.
  6. Launch R, and check your R environment.
  7. Set up R for plotting.
  8. Open another midway2 connection (*optional*).

Connect to midway2

- **If you have an RCC account:** I'm assuming you already know how to connect to midway2. *ThinLinc is recommended if you do not know how activate X11 forwarding in SSH.* See: `https://rcc.uchicago.edu/docs/connecting`
- **If you do not have an RCC account:** I can provide you with a Yubikey. This will give you guest access to the RCC cluster (see the next slide).

Using the Yubikeys

- Prerequisites:
  1. SSH client (for Windows, please use **MobaXterm**)
  2. USB-A port
- Steps:
  1. Insert Yubikey into USB port.
  2. Note your userid: `rccguestXXXX`, where `XXXX` is the last four digits shown on Yubikey.
  3. Follow instructions to connect to midway2 via SSH, replacing the cnetid with your `rccguestXXXX` user name: `https://rcc.uchicago.edu/docs/connecting`
  4. When prompted for password, press lightly on metal disc.
- Important notes:
  - ▷ Yubikeys do not work with ThinLinc.
  - ▷ *Please return the Yubikey at the end of the workshop.*

Download workshop packet

Once you have connected to a midway2 login node, download the workshop packet to your scratch directory on the cluster (**note:** there are no spaces in the URL below):

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

Install PLINK

Download the most recent stable version of PLINK in the same place as the other workshop materials.

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

On midway2, you can run these commands to download PLINK, and check the version:

```
cd $SCRATCH/genetic-data-analysis-1
wget https://bit.ly/2UgRH36 -O plink.zip
unzip plink.zip
./plink --version
```

Download 1000 Genomes data

Download the 1000 Genomes data from the European Bioinformatics Institute. *Downloading from the EBI may take a long time (10–20 minutes), so if possible copy the previously downloaded files instead.*

- Copying previously downloaded data:

```
cd $SCRATCH/genetic-data-analysis-1
cp ~pcarbo/share/1kg.vcf.gz .
```

- Downloading from EBI:
  - ▷ Short URL: `http://bit.ly/2G7ZWYu`
  - ▷ Download this file: `ALL.chip.omni_broad_sanger_combined.20140818.snps.genotypes.vcf.gz`
  - ▷ Make sure you put the file in same location as the other workshop materials.

Connect to a midway2 compute node

Set up an interactive session on a midway2 compute node with 8 CPUs and 19 GB of memory:

```
screen -S workshop
sinteractive --partition=broadwl \
  --reservation=workshop --cpus-per-task=8 \
  --mem=19G --time=3:00:00
echo $HOSTNAME
```

Launch R

Start up an interactive R session:

```
cd $SCRATCH/genetic-data-analysis-1
module load R/3.5.1
which R
R --no-save
```

Check your R environment

Check that you are running R 3.5.1:

```
sessionInfo()
```

Check that you are starting with an empty environment:

```
ls()
```

Check that you have the correct working directory—it should be set to the “genetic-data-analysis-1” repository:

```
getwd()
```

Set up R for plotting

Make sure you can display graphics in your current R session.

```
library(ggplot2)
library(cowplot)
data(cars)
quickplot(cars$dist,cars$speed)
```

You should see a scatterplot. If not, your connection is not set up to display graphics. An alternative is to save the plot to a file, and download the file to your computer using sftp or SAMBA:

- rcc.uchicago.edu/docs/data-transfer

Quit R

We will quit R, and return to it later.

```
quit()
```

Open another connection to midway2

- *Optionally,* open a new SSH connection, following the same steps as before (see the “Connect to midway2” slide).
- This second connection can be used to monitor your computations on the cluster.

At this point, you have completed the initial setup. You are now ready to move on to the next programming challenge.

Outline of workshop

- Preliminaries
- Programming challenges:
    1. Setting up your environment for genetic data analysis.
    2. **Downloading and preparing genotype data for PCA.**
    3. Run PCA on genotype data.
    4. Visualize and interpret PCA results.

Challenge #2: Download & prepare genotype data

- Aim: Prepare genotype data for PCA analysis in R.
- Steps:
  1. Examine VCF file.
  2. Convert VCF file to PLINK format.
  3. Convert PLINK text file to binary format.
  4. Remove related 1000 Genomes samples.
  5. Prune SNPs in LD.

Examine the VCF file

Run a few simple shell commands to inspect the genotype data stored in the VCF file.

```
cd $SCRATCH/genetic-data-analysis-1
ls -lh 1kg.vcf.gz
zcat 1kg.vcf.gz | less -S
```

- Reference: www.cog-genomics.org/plink2/formats#vcf

What is a VCF file?

- 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.
- See also:
  - vcftools.github.io
  - samtools.github.io/hts-specs
  - `doi:10.1093/bioinformatics/btr330`

Convert VCF to PLINK

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

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

Note: This step will require about 18 GB of free space.

Explore PLINK files

Run a few simple shell commands to inspect the 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
```

PLINK files: concepts

- Probably most commonly used format for storing human genoytpe data.
- Less flexible than VCF.
- 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 more efficient, but not human readable.
- See: www.cog-genomics.org/plink/1.9/formats#ped

Convert to binary format, remove related samples

To speed up the data processing steps, we convert to binary PLINK format (and remove the very large .ped file).

```
./plink --file 1kg --make-bed --out 1kg
rm 1kg.map 1kg.ped
```

Remove 29 of 31 related samples (because most population structure analyses are not designed to handle related samples):

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

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.

```
./plink --bfile 1kg_unrelated \
  --indep-pairwise 1000 500 0.08
./plink --bfile 1kg_unrelated \
  --make-bed --extract plink.prune.in \
  --out 1kg_pruned
```

Data preparation: take-home points

- VCFtools and PLINK have *many* commands for manipulating genotype data.
- For more specialized manipulations, 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 can lead to a poor quality analysis.
- It is important to record all your data processing steps.

Outline of workshop

- Preliminaries
- Programming challenges:
    1. Setting up your environment for genetic data analysis.
    2. Downloading and preparing genotype data for PCA.
    3. **Run PCA on genotype data.**
    4. Visualize and interpret PCA results.

Challenge #3: Run PCA on genotype data

- Aim: Compute principal components (PCs) from genotype data, which we will use to gain insight into the genetic data.
- Steps:
  1. Convert genotypes to a matrix.
  2. Import genotypes into R.
  3. Fill in missing genotypes.
  4. Compute PCs using rsvd package, and save results.

Convert genotype data to a matrix

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

```
./plink --bfile 1kg_pruned \
  --recode A --out 1kg_recoded
```

Import genotypes into R

Start up an interactive R session:

```
R --no-save
```

I implemented a function to import the genotypes into a matrix. It uses `fread` from the `data.table` package.

```
library(data.table)
source("functions.R")
```

Load the genotype matrix into R:

```
geno <- read.geno.raw("1kg_recoded.raw")
```

Examine genotypes

Run a few commands to examine the genotype matrix:

```
class(geno)
nrow(geno)
ncol(geno)
geno[1:4,1:4]
```

Fill in missing genotypes

Problem: A small fraction (<1%) of the genotypes are missing:

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

For PCA, 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)
}
```

Double-dheck that there are no missing genotypes:

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

Compute PCs

Use the “rpca” function from the rsvd package to compute the first 10 PCs—that is, the 10 components that explain the most variation in the genotypes:

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

Compute PCs

Take a quick look at the PCA results:

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

Save the PCA results analysis:

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

PCA analysis: take-home points

- The input to PCA must be a (numeric) matrix with no missing 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 file “pca.sbatch” for an example of automating the PCA analysis using Slurm.

Outline of workshop

- Preliminaries
- Programming challenges:
    1. Setting up your environment for genetic data analysis.
    2. Downloading and preparing genotype data for PCA.
    3. Run PCA on genotype data.
    4. **Visualize and interpret PCA results.**

Challenge #4: Visualize and interpret PCA results

- Aim: Create plots from the PCA results to gain insight into the genetic data.
- Steps:
  1. Set up R for plotting.
  2. Create a basic PC plot.
  3. Create a PC plot with population labels.

Set up R for plotting

Load the plotting packages and some functions I defined for creating the PCA plots.

```
library(ggplot2)
library(cowplot)
source("functions.R")
```

If necessary, load the PCA results.

```
load("1kg_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.

Create a PC plot with population labels

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

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

Add a new column, “label”, to the PCA results table:

```
pcs     <- as.data.frame(pcs)
ids     <- sapply(strsplit(rownames(pcs),"_"),
                  function (x) x[2])
labels <- subset(labels,is.element(labels$id,ids))
rows    <- match(ids,labels$id)
pcs$label <- factor(labels$pop)
```

Create a PC plot with population labels

Create the PC plot with labels:

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

Save the labeled PC plot

Save your work as a PDF file using the “ggsave” from the ggplot2 package:

```
ggsave("1kg_pca.pdf",p2)
```

Visualizing and interpreting PCA results: take-home points

- PCA is the most commonly used approach to infer population structure from genotype data.
- One reason PCA is so popular is that it can produce evocative visualizations of populations structure.
- However, there are many pitfalls in interpreting PCA results—proceed with caution!

Recap

Some genetic data analysis techniques we used today:

- Using PLINK and basic shell commands to process and convert genotype data.
- Importing genotype data into R.
- Running PCA on a genotype matrix in R.
- Using ggplot2 to visualize and interpret the results of the PCA analysis.