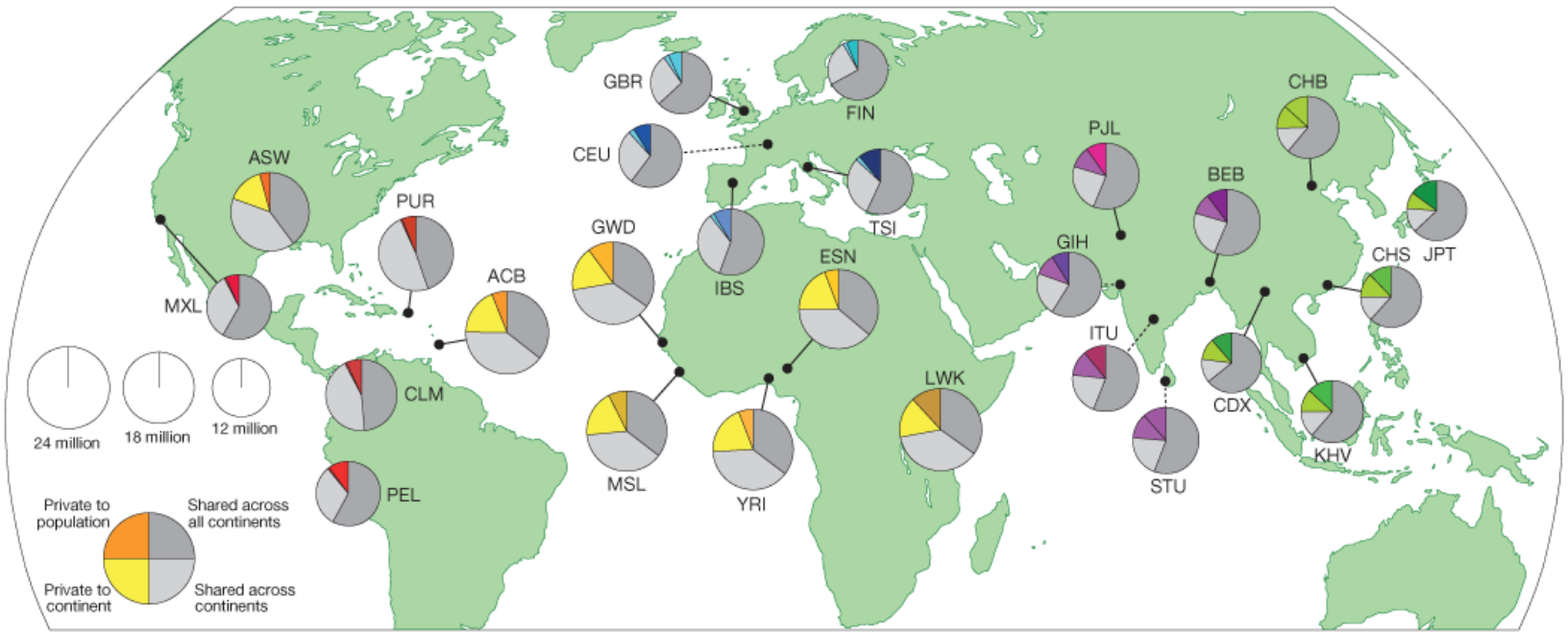
Analysis of Genetic Data 1: Inferring Population Structure

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 5 genotype samples as part of our study.
2. We would like to uncover population structure in this small sample.
3. To do so, we infer population structure *relative to a "reference" data set*. 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 investigate population structure from the genetic data.

Outline of workshop

- **Preliminaries**
- Programming challenges:
  - ▷ *Add programming challenges here.*

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.

Outline of workshop

- Preliminaries
- Programming challenges:
  - ▷ *Add programming challenges here.*

What's included in the workshop packet

Give overview of contents here.

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.csv.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 move the file to 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 18 GB of memory:

```
screen -S workshop
sinteractive --partition=broadwl \
  --reservation=workshop --cpus-per-task=8 \
  --mem=18G --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, e.g.,

```
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:
  - ▷ *Add programming challenges here.*

Download and prepare the genotype data

Outline of the data preparation 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 0.1
./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:
  - ▷ *Add programming challenges here.*

Run PCA on genotype data

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
R
```

Note: If you are not using the RCC cluster, make sure you have a recent version of R installed on your laptop, preferrably 3.4.0 or greater. You can check which version you have by typing `version$version.string` in R.

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.

```
library(data.table)
source("functions.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: <1% of the genotypes are missing:

```
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))
```

Note that the genotypes are now not always 0, 1 or 2! e.g., `geno[1200,1]`. How to interpret these “fractional” genotypes?

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.
- *Optional:* See file `pca.sbatch` in the `code` folder for an illustration of how to automate the steps of the this analysis on the RCC cluster using the SLURM job engine.

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. Add population labels to the PC plot.
4. Add sample ids to the unlabeled study samples in the PC plot.

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) # Optional.
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")
```

This adds a “label” column to the `pcs` table:

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

Create the PC plot with labels:

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

Add population labels to PC plot (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

To interpret the PCA results for the 5 study samples, we add sample ids to the 5 points in the PC plot:

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

- 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—8proceed 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 analyzing your genetic data.