Day2: Sharpening the data (morning session)

The goal of this practical is to introduce multiple key software packages commonly used for genotype data management. Specifically, we will use the following packages:

1. BCFtools (Download / Documentation). This tool is extremely efficient for data management.
2. VCFtools (Download / Documentation). Useful to perform some basic statistical analysis.
3. QTLtools (Download / Documentation). Initially developed to perform molecular QTL analysis, it also contains a useful module to perform PCA on genotype data.
4. PLINK1.9 (Download / Documentation). The very last version of PLINK (works on VCF files).

The raw material for this practical is a VCF file containing some genotype data set for multiple Spanish individuals on a single chromosome. This data has been generated with Illumina OMNI2.5M. We will now perform multiple standard QC steps to prepare the data we need for the practical of this afternoon. Note that from the command lines described below, you should be able to build an entire QC pipeline tailored to your specific needs. The code color in this practical is as follows:

1. Grey means command lines,
2. Green means questions that need answers.

0. Raw genotype data

The genotype data set we use in the practical is located in:

BASH

```
ls -halt /embo/data/olivier/chr20.RAW.vcf.gz
```

Copy the VCF file in your local directory and then open it in order to get familiar with the file format:

BASH

```
less -S /embo/data/olivier/chr20.RAW.vcf.gz
```

Q: By using combinations of bash commands (zcat, wc, grep and cut), try to determine:

(1) The chromosome we are working on,
(2) The number of variants,
(3) And the number of individuals.

BASH

```
#Number of variants
zcat /embo/data/olivier/chr20.RAW.vcf.gz | grep -v "#" | wc –l

#Number of samples
zcat /embo/data/olivier/chr20.RAW.vcf.gz | grep "CHROM" | cut -f10- | wc -w
```

1. Filtering variants with poor call rates

Check the missing data rate at each variant using VCFtools as follows:

BASH

```
vcftools --gzvcf /embo/data/olivier/chr20.RAW.vcf.gz --missing-site --stdout >
chr20.RAW.missing.txt
```

Then, load the resulting report in R as follows:

R

```
#Read the missing data report
DATA = read.table("chr20.RAW.missing.txt", head=TRUE)

#Plot the rates of missing data per variant
pdf("chr20.RAW.missing.pdf", 12, 4)
plot(DATA$F_MISS, xlab="Variant index", ylab="Missing data rate", col=ifelse(DATA$F_MISS >0.1,
"red", "grey"))
abline(h=0.1, col="red")
dev.off()
```

Now that we computed the missing data rate per variant, we can filter out poorly called variants (with more than 10% missing data) using:

BASH

```
vcftools --gzvcf /embo/data/olivier/chr20.RAW.vcf.gz --max-missing 0.1 --recode --stdout | bgzip -c >
chr20.STEP1.vcf.gz
```

Since, we've just created a new VCF file, let's rebuild an index for it using:

BASH

```
tabix -p vcf chr20.STEP1.vcf.gz
```

Q: How many poorly called variants have been removed?

2. Exclude variants based on allele frequency

To achieve this, we will spot variants for which frequencies are really discordant with some reference data set. To do so, we will compare our allele frequencies to those derived from 1000 Genomes. The 1000 Genomes data only for chromosome 20 and European samples can be found here:

BASH

```
ls -halt /embo/data/olivier/reference/chr20.EUR.vcf.gz
```

Let's compute allele frequencies in this data using VCFtools as follows:

BASH

```
vcftools --gzvcf /embo/data/olivier/reference/chr20.EUR.vcf.gz --freq2 --stdout | sed '1d' | awk
'{print $2, $4, $5, $6}' > chr20.EUR.freq
```

Do the same for the VCF file containing our genotype data and write the report in a file chr20.STEP1.freq. Then compare the two sets of frequencies in ***R***:

R

```
#Read report for reference data set
EXP = read.table("chr20.EUR.freq", head=FALSE)
colnames(EXP) = c("pos", "tot_exp", "ref_exp", "alt_exp")
```

```
#Read report for our data set
OBS = read.table("chr20.STEP1.freq", head=FALSE)
colnames(OBS) = c("pos", "tot_obs", "ref_obs", "alt_obs")

#Merge both together
M = merge(EXP, OBS, by="pos")

#Test for significant differences
M$pvalue = apply(M, 1, FUN=function(x) fisher.test(matrix(round(c(x[2]*x[3], x[2]*x[4], x[5]*x[6],
x[5]*x[7])), ncol=2))$p.value)

#Scatter plot the comparison
pdf("chr20.STEP1.frequencies.pdf")
plot(M$alt_exp, M$alt_obs, xlab="ALT frequency in Reference", ylab="ALT frequency in Observed",
col=ifelse(M$pvalue < 1e-10, "red", "black"), main=”Missing data report for chr20”)
legend("bottomright", legend=c("pvalue > 1e-10", "pvalue < 1e-10"), fill=c("black","red"),
bg="white")
dev.off()

#Write the list of variants to be excluded
write.table(cbind(rep(20, sum(M$pvalue < 1e-10)), M$pos[M$pvalue < 1e-10]),
"chr20.STEP1.filtered.txt", quote=FALSE, col.names=FALSE, row.names=FALSE)
```

Q: How many variants are we going to filter out?

Now, you can remove these frequency discordant variants using VCFtools as follows:

BASH

```
vcftools --gzvcf chr20.STEP1.vcf.gz --exclude-positions chr20.STEP1.filtered.txt --recode --stdout |
bgzip -c > chr20.STEP2.vcf.gz
```

Do not forget to index this new VCF file.

3. Exclude variants based on Hardy-Weinberg Equilibrium

The next step relies on removing variants that deviate too much from the Hardy-Weinberg Equilibrium (HWE). First, test all variants using vcftools as follows:

BASH

```
vcftools --gzvcf chr20.STEP2.vcf.gz --hardy --stdout | sed '1d' | awk '{ print $2, $3, $4, $6 }' >
chr20.STEP2.hwe.txt
```

Q: Parse the report file and give one example of a variant passing HWE and another failing HWE (i.e. their expected and observed genotype frequencies).

Then, plot the result of this analysis in R:

R

```
#Load the report file
DATA = read.table("chr20.STEP2.hwe.txt", head=FALSE)
```

```
#Plot the HWE deviations
pdf("chr20.STEP2.hwe.pdf", 12, 4)
plot(-log10(DATA$V4), xlab="Variant index", ylab="-log10(HWE test)", col=ifelse(-log10(DATA$V4) >
5, "red", "grey"), main=”HWE report for chr20”)
abline(h=5, col="red")
dev.off()
```

Extract the list of variants that deviate from HWE using a p-value threshold of 1e-5 and write it in a file chr20.STEP2.filtered.txt:

BASH

```
cat chr20.STEP2.hwe.txt | awk '{ if ($4 < 1e-5) print "20", $1 }' > chr20.STEP2.filtered.txt
```

Build a new VCF file (*chr20.STEP3.vcf.gz*) by filtering out all variants failing HWE.

Q: How many variants did you filter out here?

4. Exclude individuals based on call rates

To do so, we first need to measure the missing data rate on a per individual basis. This can be achieved using VCFtools:

BASH

```
vcftools --gzvcf chr20.STEP3.vcf.gz --missing-indv --stdout > chr20.STEP3.missing.txt
```

Then, plot the missing data rates per individual in R. To do so, adapt the R script in section 1 to produce a plot *chr20.STEP3.missing.pdf*.

Q: How many individuals exhibit high missing rates?

Build the list of individuals to remove from the data (*chr20.STEP3.filtered.txt*) in order to create a new VCF file (*chr20.STEP4.vcf.gz*):

BASH

```
vcftools --gzvcf chr20.STEP3.vcf.gz --remove chr20.STEP3.filtered.txt --recode --stdout | bgzip -c >
chr20.STEP4.vcf.gz
```

5. Exclude individuals based on relatedness

Assuming you want to work on unrelated individuals only, we will now try to remove related individuals in the data. First, compute IBD estimates using plink2 as follows:

BASH

```
plink --vcf chr20.STEP4.vcf.gz --genome --ppc-gap 100 --out chr20.STEP4
```

Then, plot the report in R:

R

```
#Load the report in a data frame
DATA = read.table("chr20.STEP4.genome", head=TRUE)
```

```
#Make 3 scatter plots comparing the IBD estimates per pair of individuals
pdf("chr20.STEP4.genome.pdf", 12, 4)
par(mfrow=c(1,3))
plot(DATA$Z0, DATA$Z1, xlab="P(IBD=0)", ylab="P(IBD=1)", main="IBD0 versus IBD1")
plot(DATA$Z0, DATA$Z2, xlab="P(IBD=0)", ylab="P(IBD=2)", main="IBD0 versus IBD2")
plot(DATA$Z1, DATA$Z2, xlab="P(IBD=1)", ylab="P(IBD=2)", main="IBD1 versus IBD2")
dev.off()
```

Q: From these plots, what are the related individuals and what are their degrees of relatedness?

Next, create a new VCF file (*chr20.STEP5.vcf.gz*) by filtering out one individual for each pair of related individuals using VCFtools. Note that you should be removing only two individuals in total.

6. Exclude individuals based on population stratification

In this last QC step, we will study the population stratification in our data in order to remove all non-European samples (called outliers). To do so, we will use a 1000 Genomes version that contains representatives of all continental groups. This data is contained in the following VCF file:

BASH

```
ls -halt /embo/data/olivier/reference/chr20.ALL.vcf.gz
```

To study population stratification, the most common approach is to use Principal Component Analysis (PCA). To do so, we first need to merge our genotype data with the 1000 Genomes one. This can be achieved using bcftools as follows:

BASH

```
bcftools merge -m id -Oz -o chr20.MERGED.vcf.gz /embo/data/olivier/reference/chr20.ALL.vcf.gz chr20.STEP5.vcf.gz
tabix -p vcf chr20.MERGED.vcf.gz
```

Then, we can perform the PCA using the QTLtools implementation:

BASH

```
QTLtools_1.1_Ubuntu16.04_x86_64 pca --vcf chr20.MERGED.vcf.gz --scale --center --distance 50000 --maf 0.05 --out chr20.MERGED
```

This performs the PCA on scaled (--scale) and centered (--center) genotype data on relatively independent (--distance 50kb) and frequent variants (--maf 0.05). Note that the population of origin of each individual can be found in the file /embo/data/olivier/reference/populations.txt. Now that we've got our PCA done, let's plot the results:

R

```
#Load all PCA coordinates in a data frame
PCA = read.table("chr20.MERGED.pca", head = TRUE)

#Subset only the two first PCs
PCA = data.frame(V1=colnames(PCA)[2:ncol(PCA)], t(PCA[1,2:ncol(PCA)]), t(PCA[2,2:ncol(PCA)]))

#Load the population of origin of each individual
```

```
POP = read.table("/embo/data/olivier/reference/populations.txt", head=FALSE)

#Merge PCA results with population of origin
DATA = merge(POP, PCA, by="V1")

#Have a look at the result
head(DATA)

#Plot PC1 versus PC2 …
pdf("chr20.MERGED.pca.pdf", 10, 5)
par(mfrow=c(1,2))

# … comparing our samples to 1000 Genomes
plot(DATA$X1[DATA$V3 != "SPA"], DATA$X2[DATA$V3 != "SPA"], xlab="PC1", ylab="PC2", pch=20,
col="grey")
points(DATA$X1[DATA$V3 == "SPA"], DATA$X2[DATA$V3 == "SPA"], pch=20, col="red")
legend("topleft", legend=c("Our samples", "1000 genomes samples"), fill=c("red","grey"))

# … and comparing all 1000 Genomes together
plot(DATA$X1[DATA$V3 != "SPA"], DATA$X2[DATA$V3 != "SPA"], xlab="PC1", ylab="PC2", pch=20,
col=DATA$V3[DATA$V3 != "SPA"])
legend("topleft", legend=unique(DATA$V3[DATA$V3 != "SPA"]), fill=unique(DATA$V3[DATA$V3 !=
"SPA"]))
dev.off()
```

Q: How many samples with non-european ancestry do we have in our data? Which population do they likely belong to?

List the IDs of these outliers and remove their genotypes in the data set using VCFtools in order to produce the final VCF file of the practical (*chr20.FINAL.vcf.gz*).

Q: How many variants and individuals remain in the data set? You should get 48,721 variants and 101 individuals. Do you get it right?

Important note: Keep the file chr20.FINAL.vcf.gz for the afternoon practical. If your numbers do not match, you can still get the right file from there: /embo/olivier/practical1_morning