Linux practical with genetic data

The goal of this practical is to gain experience working with genomic file formats on the command line.

Directions are in black
`Code snippets are in gray Menlo`
Questions are in bold blue.

For this practical, the first step is to log on to the HPC, then get an interactive job for 5 hours:

```
qsub -I -l walltime=5:00:00 -l nodes=1:ppn=1
```

change to the session2 directory and create a directory named <your_name>. This directory will be your base directory for this project

```
cd /primary/projects/training/session2/
mkdir <your_name>
cd <your_name>
```

The raw data for this practical is a VCF file with genotype data for some individuals. This data was generated with the Illumina OMNI2.5M array. We will now work through several standard QC steps using this original VCF file.

First, copy the VCF file into your newly created directory:

```
cp ../raw_data/chr20.RAW.vcf.gz .
```

Take a look at the VCF file:

```
less -S chr20.RAW.vcf.gz
```

Using a combination of bash commands (zcat, wc, grep, and cut), try to determine

1. **The chromosome that we are working on**
2. **The number of variants**
3. **The number of individuals**

Step 1: Filtering Variants with Low Call Rates

In the case of this array data, what would a low call rate mean for a variant?

First load VCFTools and htslib:

```
module load bbc/vcftools/vcftools-0.1.16
module load bbc/htslib/htslib-1.10.2
```

Now check the missing rate for each sample:

```
vcftools --gzvcf chr20.RAW.vcf.gz --missing-site --stdout >
chr20.RAW.missing.tsv
```

Now read the missing report data into R (If needed load R: `module load bbc/R/R-3.6.0/`):

```
DATA = read.table("chr20.RAW.missing.tsv",head=TRUE)
# Plot the rates of missing data per variant
# open a pdf
pdf("chr20.RAW.missing.pdf",12,4)
# make the plot with coloring based on F_MISS
plot(DATA$F_MISS, xlab="Variant index", ylab="Missing data rate",
col=ifelse(DATA$F_MISS > 0.1,"red","grey"))
# add a straight line
abline(h=0.1,col="red")
# close the pdf
dev.off()
```

Now exit R (`q()`), no need to save your session.

Now that we have examined the plot of the missing data rate, we can exclude variants with greater than 10% missing data and save these to a new file:

```
vcftools --gzvcf chr20.RAW.vcf.gz --max-missing 0.1 --recode --stdout | bgzip -c >
chr20.step1.vcf.gz
```

How many variants have been removed?

Index the filtered file using htslib:

```
tabix -p vcf chr20.step1.vcf.gz
```

Step 2: Exclude variants based on allele frequency

In this step, variants with allele frequencies discordant with some reference data set will be excluded. Our reference set will be the European samples of the 1000 Genomes project

Why would you want to exclude these variants?

```
less -S ../raw_data/reference/chr20.EUR.vcf.gz
```

Get the allele frequencies from the European population:

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

How would you go about testing the individual steps of this command so that you can understand the final output?
Using a combination of zcat and grep, can you confirm position chr20:61651?

Using nearly the same command, except this time with our observed data, save the observed frequencies to “chr20.step1.freq”.

Compare the two frequencies in R:

```
# Read the expected European frequencies
EXP = read.table("chr20.EUR.freq", head=FALSE)
colnames(EXP) = c("pos","tot_exp","ref_exp","alt_exp")

# Read the observed frequencies in our data
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
pdf("chr20.step1.frequencies.pdf")
plot(M$alt_exp,M$alt_obs,
     xlab="ALT freq in Ref",
     ylab="ALT freq in Observed",
     col=ifelse(M$pvalue < 1e-10,"red","black"),
     main="Missing data report for chr20"
```

```
)
legend("bottomright",legend=c("pvalue > 1e-10","pavlue < 1e-
10"),fill=c("black","red"),bg="white")
dev.off()

# Write the variants list 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,row.names=FALSE,sep="\t"
)
```

How many variants are we going to filter out?

Remove the discordant variants with vcftools:

```
vcftools --gzvcf chr20.step1.vcf.gz --exclude-positions
chr20.step1.filtered.txt --recode --stdout|bgzip -c >
chr20.step2.vcf.gz
```

Don’t forget to index the file!

Step 3: Exclude individuals based on call rates

In this step, individuals with a high number of missing calls are excluded

```
vcftools --gzvcf chr20.step2.vcf.gz --missing-indv --stdout >
chr20.step2.missing.tsv
```

What does each line in chr20.step2.missing.tsv represent?
Can you sort this file based on the missing data rate?
What individual has the highest missing data rate?
How many individuals have a missing rate greater than 2%?

Build a list of individuals to exclude in chr20.step2.removeIndividuals.tsv (plain text file with 1 individual per line). Then remove the individuals with high missing data:

```
vcftools --gzvcf chr20.step2.vcf.gz --remove
chr20.step3.removeIndividuals.tsv --recode --stdout | bgzip -c >
chr20.step3.vcf.gz
```

Confirm that you excluded the individuals from the new file using zcat, cut, and wc. What command did you use?

Step 4: Exclude individuals based on relatedness

Assuming that we want to only work with unrelated individuals, we can exclude samples based on identity-by-descent. For this we will work with the software package PLINK:

```
module load bbc/plink/plink-v1.90b6.18

plink --vcf chr20.step3.vcf.gz --genome --ppc-gap 100 --out
chr20.step4
```

What does the --genome option do?
what does the --pca-gap option do? Why is this a good choice?

Now that we have calculated the identity-by-descent, make a plot of the data in R:

```
# Load the report in a data frame
DATA = read.table("chr20.step4.genome",header=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 vs. IBD1")
plot(DATA$Z0, DATA$Z2, xlab="P(IBD=0)", ylab="P(IBD=2)",
main="IBD0 vs. IBD2")
plot(DATA$Z1, DATA$Z2, xlab="P(IBD=1)", ylab="P(IBD=2)",
main="IBD1 vs. IBD2")
dev.off()

# After looking at the pdf, check which pair have Z2>0.5
DATA[which(DATA$Z2>0.5),]

# Which pair have Z1>0.2
DATA[which(DATA$Z1>0.2),]
```

Which individuals are related? What is their relationship?

Make a new list (chr20.step4.removeIndividuals.tsv) to exclude one of the two individuals as was done in section 3.

What is a data-driven way to select which individual of the pairs to remove?

Then remove the selected individuals:

```
vcftools --gzvcf chr20.step3.vcf.gz --remove
chr20.step4.removeIndividuals.tsv --recode --stdout | bgzip -c >
chr20.step4.vcf.gz
```

Step 5: Exclude individuals based on their ancestry

In this final quality control step, individuals with a non-European background will be removed from the dataset. To achieve this goal, we will use a 1000 Genomes version of the file that contains representatives of all continental groups. Then we can use principal component analysis to identify individuals that are not of European ancestry.

Is it OK to exclude an individual based on their genetic background? How does sampling bias affect results, conclusions, and efficacy?

We will need bcftools:

```
module load bbc/bcftools/bcftools-1.10.2
```

Take a look at the reference file:

```
ls ../raw_data/reference/chr20.ALL.vcf.gz
```

How many samples are in this reference? Is this more than 1000?

Now we want to merge the 1000 genomes reference with our own data

```
bcftools merge -m id -Oz -o chr20.merged.vcf.gz
../raw_data/reference/chr20.ALL.vcf.gz chr20.step4.vcf.gz

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

Now we can perform a PCA on this genetic data using QTLtools. You can download QTLtools from https://qtltools.github.io/qtltools/. Click the link then go to *Downloads and Installation*. Right click on the Centos prebuilt binary and select *Copy Link Location*. Now go to your terminal type “wget” and then paste the link with Cmd + v.

```
wget
https://qtltools.github.io/qtltools/binaries/QTLtools_1.2_CentOS
7.8_x86_64.tar.gz
```

QTLtools is now downloaded and ready to run! Use it to compute a PCA:

```
./QTLtools_1.2_CentOS7.8_x86_64/QTLtools_1.2_CentOS7.8_x86_64
pca --vcf chr20.merged.vcf.gz --scale --center --distance 50000
--maf 0.05 --out chr20.merged
```

This performs PCA on scaled and centered genotype data on more or less independent (--distance 50kb) variants with a frequency greater than 5%. The population of origin for each individual is in “../raw_data/reference/populations.txt”. Let us now use this to label

each sample by population and plot our results in R. First, however, so we don't get confused, let us create a file that has our samples in it:

```
zcat chr20.step4.vcf.gz|grep '^#CHROM'| cut -f10- | tr '\t' '\n'
> mySamples.txt
```

Now start R and run the following code:

```
# Load the PCA as a data.frame
PCA = read.table("chr20.merged.pca",head=TRUE)

# subset to the first two PCAs
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 file
POP =
read.table("../raw_data/reference/populations.txt",head=FALSE)

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

# Now load our list of samples to use this
samples =
read.table('mySamples.txt',sep="\t",header=FALSE,stringsAsFactor
s=FALSE)$V1

# Now get an index of which samples are ours
index = which(DATA$V1 %in% samples)

# plot PC1 vs. PC2
pdf("chr20.merged.pca.pdf", 10, 5)
par(mfrow=c(1,2))

# Plot all the points
plot(DATA$X1,DATA$X2,xlab='PC1',ylab='PC2',pch=20,col='grey')
# Add our samples on top
points(DATA$X1[index],DATA$X2[index],col='red')
legend("topleft", legend=c("My samples","1000 Genome
samples"),fill=c("red","grey"))

# Comparing our samples to 1000 Genomes
plot(DATA$X1,DATA$X2,xlab='PC1',ylab='PC2',pch=20,col=DATA$V3)
legend("topleft", legend=unique(DATA$V3), fill=unique(DATA$V3))

# check the ancestry assignment from the reference
table(DATA$V3[index])
```

```
dev.off()
```

What is the assigned population in the reference for our samples?
How many individuals of non-European ancestry do we have in our data?
List the ids of these outliers and remove them from the VCF file.
How many variants and individuals remain?