Genotype Imputation

Matthew Parker
Mostafavi Lab

June 26, 2019

Overview

The following steps detail a workflow for preparing and imputing the PNC genome data for analysis.

The files are available on github at mrparker909/PNCgwasWorkflow and at namheegordonkim/gwasqc.

Useful Background Reading

Anderson, C. A., Pettersson, F. H., Clarke, G. M., Cardon, L. R., Morris, A. P., & Zondervan, K. T. (2010). **Data quality control in genetic case-control association studies**. Nature Protocols, 5(9), 1564–1573. https://doi.org/10.1038/nprot.2010.116

Ellingson, S. R., & Fardo, D. W. (2016). **Automated quality control for genome wide association studies**. F1000Research, 5. https://doi.org/10.12688/f1000research.9271.1

Reed, E., Nunez, S., Kulp, D., Qian, J., Reilly, M. P., & Foulkes, A. S. (2015). **A guide to genome-wide association analysis and post-analytic interrogation**. Statistics in Medicine, 34(28), 3769–3792. https://doi.org/10.1002/sim.6605

McVean, G. (2009). **A Genealogical Interpretation of Principal Components Analysis**. PLoS Genetics, 5(10). https://doi.org/10.1371/journal.pgen.1000686

1) Extract only subjects of Race==EA, from the genotype data

Race info contained in

".../phenotype/phs000607.v1.pht003445.v1.p1.c1.Neurodevelopmental_Genomics_Subject_Phenotypes.GRU-NPU.txt"

We can use `plink2` in the terminal to create a subset of the subjects, with Race == "EA"

The variable encoding for Race is:

`AA=Black or African American; AI=American Indian or Alaska Native; AS=Asian; EA=European American; HI=Hispanic/Latino; PI=Native Hawaiian/Pacific Islander; OT=Other; NA=Not Available/Pending Validation`

To use `plink`, we need to append to the Phenotype file FID and IID as first two columns:

00a-createPheno.R

```
# R code to create phenotype file for plink.
# The phenotype file requires FID and IID as first two columns,
# phenotype file currently missing FID, but has IID labelled as SUBJID
# .fam files have column1=FID, column2=IID
library(dplyr)
library(magrittr)

pathToPheno  = "/zfs3/scratch/saram_lab/PNC/data/phenotype/phs000607.v1.pht003445.v1.p1.
               c1.Neurodevelopmental_Genomics_Subject_Phenotypes.GRU-NPU.txt"
pathToFam1   = "/zfs3/scratch/saram_lab/PNC/data/genotype/raw/GO_Affy60.fam"
pathToFam2   = "/zfs3/scratch/saram_lab/PNC/data/genotype/raw/GO_Axiom.fam"
```

```
pathToFam3   = "/zfs3/scratch/saram_lab/PNC/data/genotype/raw/GO_Omni.fam"
pathToFam4   = "/zfs3/scratch/saram_lab/PNC/data/genotype/raw/GO_Quad.fam"
pathToFam5   = "/zfs3/scratch/saram_lab/PNC/data/genotype/raw/GO_v1_hg18.fam"
pathToFam6   = "/zfs3/scratch/saram_lab/PNC/data/genotype/raw/GO_v3.fam"
pathToOut    = "/zfs3/scratch/saram_lab/PNC/data/phenotype/FID_IID_
               Neurodevelopmental_Genomics_Subject_Phenotypes.GRU-NPU.txt"

# 1) read in phenotype file, rename SUBJID to IID
phen1 <- read.csv(pathToPheno, skip = 10, header = TRUE, sep = "\t")
phen2 <- phen1 %>% rename(IID=SUBJID)

# 2) read in .fam files
fam1   <- read.csv(pathToFam1, header = FALSE, sep = " ")
names(fam1) <- c("FID", "IID", 1:4)

fam2   <- read.csv(pathToFam2, header = FALSE, sep = " ")
names(fam2) <- c("FID", "IID", 1:4)

fam3   <- read.csv(pathToFam3, header = FALSE, sep = " ")
names(fam3) <- c("FID", "IID", 1:4)

fam4   <- read.csv(pathToFam4, header = FALSE, sep = " ")
names(fam4) <- c("FID", "IID", 1:4)

fam5   <- read.csv(pathToFam5, header = FALSE, sep = " ")
names(fam5) <- c("FID", "IID", 1:4)

fam6   <- read.csv(pathToFam6, header = FALSE, sep = " ")
names(fam6) <- c("FID", "IID", 1:4)

fam <- rbind(fam1,fam2,fam3,fam4,fam5,fam6)

# 3) subset .fam to FID and IID
FAM  <- fam %>% select(FID, IID)

# 4) left join phenotype with FID and IID by IID
newP  <- phen2 %>% left_join(FAM, by="IID")

missingIIDs <- setdiff(phen2$IID, FAM$IID)
write.csv(x = missingIIDs, file = "./pheno_missingIIDS.csv")

# 5) ensure first two columns are FID and IID
newP2 <- mutate_all(newP[c("FID", "IID", setdiff(names(newP), c("FID","IID")))],
                    stringr::str_replace_all, pattern=" ", replacement="")

# 6) save new phenotype file
write.table(x = newP2, pathToOut, sep = " ", quote=F, row.names=F)
```

Next, we create a text file listing the FID and IID for each EA:

00b-createFID_IID.R

```
# R code to create EAsubjectFID_IID.txt
library(dplyr)
inp <- "/zfs3/scratch/saram_lab/PNC/data/phenotype/FID_IID_
          Neurodevelopmental_Genomics_Subject_Phenotypes.GRU-NPU.txt"
out <- "/zfs3/scratch/saram_lab/PNC/data/phenotype/EAsubjectID/EAsubjectFID_IID.txt"
df  <- read.csv(inp, header = TRUE, stringsAsFactors = FALSE, quote = "", sep = " ")
df2 <- dplyr::filter(df, Race %in% c("EA"))
df3 <- dplyr::select(df2,FID,IID)
write.table(df3, out, sep=" ", row.names = F,col.names=F, quote=F)
```

Next we can use the ID list with `plink` to subset the genome data:

01a-subsetEA.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N subsetEA
#PBS -l mem=10gb
#PBS -l walltime=1:00:00
#PBS -l procs=1

#PBS -M mrparker909@gmail.com
#PBS -m abe
#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

echo "Starting script at `date`"
cd $PBS_O_WORKDIR

# PLINK2 CODE TO SUBSET BY RACE=="EA"
module load plink2

pFile="/zfs3/scratch/saram_lab/PNC/data/phenotype/FID_IID_Neurodevelopmental_
       Genomics_Subject_Phenotypes.GRU-NPU.txt"
genDir="/zfs3/scratch/saram_lab/PNC/data/genotype/raw"
outDir="/zfs3/scratch/saram_lab/PNC/data/genotype/raw/EAsubjectOnly"
idFile="/zfs3/scratch/saram_lab/PNC/data/phenotype/EAsubjectID/EAsubjectFID_IID.txt"

plink --bfile $genDir/GO_Affy60 --make-bed --out $outDir/GO_Affy60 --pheno $pFile --mpheno 3 --keep $idFile
plink --bfile $genDir/GO_Axiom --make-bed --out $outDir/GO_Axiom --pheno $pFile --mpheno 3 --keep $idFile
plink --bfile $genDir/GO_Omni --make-bed --out $outDir/GO_Omni --pheno $pFile --mpheno 3 --keep $idFile
plink --bfile $genDir/GO_Quad --make-bed --out $outDir/GO_Quad --pheno $pFile --mpheno 3 --keep $idFile
plink --bfile $genDir/GO_v1_hg18 --make-bed --out $outDir/GO_v1_hg18 --pheno $pFile --mpheno 3 --keep $idFile
plink --bfile $genDir/GO_v3 --make-bed --out $outDir/GO_v3 --pheno $pFile --mpheno 3 --keep $idFile

echo "script completed at `date`"
```

The output genome data files located:

`/zfs3/scratch/saram_lab/PNC/data/genotype/raw/EAsubjectOnly/`

are ready for quality checking, imputation, and merging.

NOTE: *Due to low sample sizes (zero for Affy60 and 3 for Axiom), we will exclude those two chips from the EAsubjectOnly analysis.*

2) Perform genotype data quality check (QC), which removes "noisy" subjects and SNPs

Fix allele encoding

The chips Omni, Quad, v1, and v3 are coded using a 1-2 encoding, while we require an ACGT encoding.

- determine exact chip by uploading .bim files to chipendium
- download corresponding snpTables from Will Rayner
- convert snpTables from AB encoding to 12 encoding, for example:

```
Rscript recodeABto12.R omni.txt omni.snptable
```

- where recodeABto12.R is

recodeABto12.R

```
args = commandArgs(trailingOnly = T)
snpTable = args[1]
outFile = args[2]

snpDat = read.csv(snpTable, sep="\t", header=F, stringsAsFactors=F)
for(r in 1:nrow(snpDat)) {
  row = snpDat[r,]
  row$V2 = gsub(pattern = "A", replacement = "1", x = row$V2)
  row$V2 = gsub(pattern = "B", replacement = "2", x = row$V2)
  snpDat[r,] = row
}
write.table(snpDat, file = outFile, sep="\t", col.names = F, row.names=F, quote=F)
```

- convert genotype data from 12 encoding to ACGT encoding:

01b-recodeAlleles.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N recodeAlleles
#PBS -l mem=10gb
#PBS -l walltime=10:00:00
#PBS -l procs=1

#PBS -M mrparker909@gmail.com
#PBS -m abe
#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

cd $PBS_O_WORKDIR
echo "Current working directory is now: " `pwd`
echo "Starting script at `date`"

# PLINK2 CODE TO RECODE ALLELES
module load plink2

genDir="/zfs3/scratch/saram_lab/PNC/data/genotype/raw/EAsubjectOnly"
outDir="/zfs3/scratch/saram_lab/PNC/data/genotype/raw/EAsubjectOnly/recoded"
snpTableDir="/zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/snpTables"

# NOTE: skipping affy since no EA subjects
# plink --bfile $genDir/GO_Affy60 --make-bed --out $outDir/GO_Affy60 --update-alleles $snpTableDir/affy.snptable
# echo "affy alleles updated"

# NOTE: skipping axiom since only 3 EA subjects
#plink --bfile $genDir/GO_Axiom --make-bed --out $outDir/GO_Axiom --update-alleles $snpTableDir/axiom.snptable
#echo "axiom alleles updated"

plink --bfile $genDir/GO_Omni --make-bed --out $outDir/GO_Omni --update-alleles $snpTableDir/omni.snptable
echo "omni alleles updated"
plink --bfile $genDir/GO_Quad --make-bed --out $outDir/GO_Quad2 --update-alleles $snpTableDir/quad.snptable
echo "quad alleles updated"
plink --bfile $genDir/GO_v1_hg18 --make-bed --out $outDir/GO_v1_hg18 --update-alleles $snpTableDir/v1.snptable
echo "v1 alleles updated"
plink --bfile $genDir/GO_v3 --make-bed --out $outDir/GO_v3 --update-alleles $snpTableDir/v3.snptable
echo "v3 alleles updated"

echo "script completed at `date`"
```

- finally, the Quad chip contains 28369 CNV ids as well as the rsIDs, so we will need to remove those (plink has output `GO_Quad2.allele.no.snp`, a file listing all of the CNV ids)

```
plink --bfile GO_Quad2 --exclude GO_Quad2.allele.no.snp --make-bed --out GO_Quad
```

a) Clone GWAS qc

Prior to using GWAS qc, we will need to clone it to the environment we are using.

This can be done by navigating to the (empty) folder you will be using, and entering the following terminal command:

```
git clone https://github.com/namheegordonkim/gwasqc .
```

Note that some changes will need to be made to the cloned files depending on your use case. We made these changes:

- `update_sexinfo.R`: changed header references from *Subject* and *Gender* to *IID* and *Sex* (to match our phenotype file)
- `imiss-vs-het-custom.R`: removed line `library("geneplotter")` (library not used)

b) Perform Quality Check

Use: GWAS qc

Run: `pre_impute_qc.sh` (QC to remove poor quality observations)

```
# Usage:
# sh pre_impute_qc.sh <input filename> <subject info> <output filename>
# <input filename>    -- For .bed, .bim, .fam files named exactly the same
#                        except for extensions. Do not include extensions.
#                        e.g. IMAGEN_20110404
# <subject info file> -- For updating sex information.
# <output filename>   -- For .vcf files to be created. Do not include extensions.

# Example Code:
sh pre_impute_qc.sh ./binaryFileName ./subjectInfo.txt ./out/testout.pre
```

Output: sorted *.vcf.gz files ready for alignment in step # 3

In our case, `subjectSexInfo.txt` needs to be built from the phenotype file (we will subset to use the columns FID, IID, and Sex).

00c-createSexInfo.R

```
# R code to create subjectSexInfo.txt
library(dplyr)
inp <- "/zfs3/scratch/saram_lab/PNC/data/phenotype/FID_IID_
          Neurodevelopmental_Genomics_Subject_Phenotypes.GRU-NPU.txt"
out <- "/zfs3/scratch/saram_lab/PNC/data/phenotype/subjectSexInfo.txt"
df  <- read.csv(inp, header = TRUE, stringsAsFactors = FALSE, quote = "", sep = " ")
df2 <- dplyr::select(df,FID,IID,Sex)
write.table(df2, out, sep=" ", row.names = F,col.names=F, quote=F)
```

We also need to change the `pre_impute_qc.sh` file so that it does not split the genotype data into chromosome files, and so it doesn't compress the files (the split will happen when we run the HRC prep tool, and we will compress them manually after). Just remove these lines (63 to 71):

```
# step 5: Splitting into chromosomes (MIS requirement)
numchr=22
for i in `seq 1 $numchr`
do
  plink --noweb --bfile $out --chr $i --make-bed --out $out.chr$i
  plink --noweb --bfile $out.chr$i --recode vcf --out $out.chr$i
  # compress
  vcf-sort $out.chr$i.vcf | bgzip -c > $out.chr$i.vcf.gz
done
```

Replace them with this (to produce .frq files):

```
# freqency file post chopping
plink --bfile $out --freq --out $out.post
```

Now we can run `pre_impute_qc.sh` for each genome data set:

Genome data sets: `GO_Omni, GO_Quad, GO_v1_hg18, GO_v3`

02a-preImputeQc.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N preImputeQc
#PBS -l mem=10gb
#PBS -l walltime=1:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

echo "Changing to directory gwasqc."
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc
echo "Starting script at `date`"

module load plink2
module load R
module load eigensoft
module load shapeit
module load impute2
module load vcftools # for vcf-sort
module load samtools # for bgzip

genDir="/zfs3/scratch/saram_lab/PNC/data/genotype/raw/EAsubjectOnly/recoded"
outDir=""
scrDir="/zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc"
sexFil="/zfs3/scratch/saram_lab/PNC/data/phenotype/subjectSexInfo.txt"

sh $scrDir/pre_impute_qc_MOD.sh $genDir/GO_Omni $sexFil $outDir/GO_Omni
sh $scrDir/pre_impute_qc_MOD.sh $genDir/GO_Quad $sexFil $outDir/GO_Quad
sh $scrDir/pre_impute_qc_MOD.sh $genDir/GO_v1_hg18 $sexFil $outDir/GO_v1_hg18
sh $scrDir/pre_impute_qc_MOD.sh $genDir/GO_v3 $sexFil $outDir/GO_v3

echo "script completed at `date`"
```

NOTE: *It is important that the folders ./tmp and ./out (located in /scrDir/) are empty at the start of a run. If the run fails and needs to be rerun, remove all generated files from ./tmp and ./out.*

Upon completion, this will have produced for each genome data set binary files (`.bim`, `.fam`, `.bed`). These files are ready for alignment in step 3.

3) "Align" all QC'ed genotype data to a reference panel

What is HRC?

Use: Prep Tool
From: www.well.ox.ac.uk/~wrayner/tools/

Step 1)

Download HRC-1000G-check-bim-v4.2.11-NoReadKey.zip

```
url="https://www.well.ox.ac.uk/~wrayner/tools/HRC-1000G-check-bim-v4.2.11-NoReadKey.zip"
loc="/zfs3/scratch/saram_lab/PNC/data/genotype/qc/EAsubjectOnly"
fil="HRC-1000G-check-bim-v4.2.11-NoReadKey"
wget  -P $loc $url
unzip $loc/$fil.zip -d $loc/$fil/
```

This is the perl script which will generate plink commands for alignment.

Step 2)

Download Reference File

```
url="ftp://ngs.sanger.ac.uk/production/hrc/HRC.r1-1/HRC.r1-1.GRCh37.wgs.mac5.sites.tab.gz"
loc="/zfs3/scratch/saram_lab/PNC/data/genotype/qc/EAsubjectOnly"
fil="HRC.r1-1.GRCh37.wgs.mac5.sites.tab"
wget  -P $loc $url
bgzip $loc/$fil.gz -d $loc/$fil/
```

This is the reference genome against which we will be aligning our SNPs.

Step 3)

Run Prep Tool:

WARNING: This step requires quite a lot of memory, or the process will be killed without any helpful error or log output. For us, 10 gb was not enough, but 30gb worked fine.

This will need to be run sequentially for each chip (NOT SIMULTANEOUSLY! or Run-plink.sh will be overwritten):

02b-alignment_Omni.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N preImputeAlign
#PBS -l mem=30gb
#PBS -l walltime=1:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

echo "Changing to directory containing gwasqc."
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc/out
echo "Starting script at `date`"
```

```
module load plink2

HRCDir="../../HRC-1000G-check-bim-v4.2.11-NoReadKey/HRC-1000G-check-bim-NoReadKey.pl"
REFDir="../../HRC.r1-1.GRCh37.wgs.mac5.sites.tab"

perl $HRCDir -b GO_Omni.bim -f GO_Omni.post.frq -r $REFDir -h

echo "script completed at `date`"
```

Then execute `sh Run-plink.sh`, which was generated in the `/gwasqc/out/` folder by the above script. Repeat this for each chip, running the `Run-plink.sh` script prior to continuing with the next chip:

02b-alignment_Quad.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N preImputeAlign
#PBS -l mem=30gb
#PBS -l walltime=1:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

echo "Changing to directory containing gwasqc."
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc/out
echo "Starting script at `date`"

module load plink2

genDir="/zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation"
HRCDir="../../HRC-1000G-check-bim-v4.2.11-NoReadKey/HRC-1000G-check-bim-NoReadKey.pl"
REFDir="../../HRC.r1-1.GRCh37.wgs.mac5.sites.tab"

perl $HRCDir -b GO_Quad.bim -f GO_Quad.post.frq -r $REFDir -h

echo "script completed at `date`"
```

02b-alignment_v1.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N preImputeAlign
#PBS -l mem=30gb
#PBS -l walltime=1:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

echo "Changing to directory containing gwasqc."
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc/out
echo "Starting script at `date`"

module load plink2

HRCDir="../../HRC-1000G-check-bim-v4.2.11-NoReadKey/HRC-1000G-check-bim-NoReadKey.pl"
REFDir="../../HRC.r1-1.GRCh37.wgs.mac5.sites.tab"

perl $HRCDir -b GO_v1_hg18.bim -f GO_v1_hg18.post.frq -r $REFDir -h

echo "script completed at `date`"
```

02b-alignment_v3.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N preImputeAlign
#PBS -l mem=30gb
#PBS -l walltime=1:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

echo "Changing to directory containing gwasqc."
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc/out
echo "Starting script at `date`"

module load plink2

HRCDir="../../HRC-1000G-check-bim-v4.2.11-NoReadKey/HRC-1000G-check-bim-NoReadKey.pl"
REFDir="../../HRC.r1-1.GRCh37.wgs.mac5.sites.tab"

perl $HRCDir -b GO_v3.bim -f GO_v3.post.frq -r $REFDir -h

echo "script completed at `date`"
```

The aligned, uncompressed, vcf files for each chromosome and chip (initially output to `./gwasqc/out/`) are stored here:

`/zfs3/scratch/saram_lab/PNC/data/genotype/qc/EAsubjectOnly/post-align/aligned`

For example, chromosome 20, chip Omni, has output file: `GO_Omni-updated-chr20.vcf`

Step 4)

We need to produce a list of assayed SNPs (so that we can use just the assayed SNPs for principal components extraction later on). Note that we are using the .bim files produced as a biproduct of running the alignment scripts. This will produce a file `assayed.txt`:

02e-assayed.sh

```
#!/bin/bash
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/chromo/aligned/

# extract rsIDs from .bim files (chromosome files aligned to HRC reference)
for f in *.bim
do
  cut -f2 $f > ${f}.rsIDs
done

# concatenate rsIDs into one file
cat *.rsIDs > assayed.txt
```

The file is stored here: `/zfs3/scratch/saram_lab/PNC/data/genotype/qc/EAsubjectOnly/post-align/assayed.txt`

Step 5)

After running HRC alignment tool, we need to compress the aligned files so they can be uploaded to the Michigan Imputation Server.

02c-bgzip.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N preImputeCompress
#PBS -l mem=30gb
#PBS -l walltime=1:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

echo "Changing to directory containing gwasqc."
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/
echo "Starting script at `date`"

module load vcftools # for vcf-sort
module load samtools
module load htslib   # for bgzip

numchr=22
for i in `seq 1 $numchr`
do
  # compress
  vcf-sort ./chromo/aligned/GO_Omni-updated-chr$i.vcf | bgzip -c > ./chromo/compressed/GO_Omni.chr$i.vcf.gz
  vcf-sort ./chromo/aligned/GO_Quad-updated-chr$i.vcf | bgzip -c > ./chromo/compressed/GO_Quad.chr$i.vcf.gz
  vcf-sort ./chromo/aligned/GO_v1_hg18-updated-chr$i.vcf | bgzip -c > ./chromo/compressed/GO_v1_hg18.chr$i.vcf.gz
  vcf-sort ./chromo/aligned/GO_v3-updated-chr$i.vcf | bgzip -c > ./chromo/compressed/GO_v3.chr$i.vcf.gz
done

echo "script completed at `date`"
```

This will have generated updated bim/bed/fam files for the whole sample and each chromosome, and compressed them into `*.vcf.gz` files for upload to MIS. These files were stored here:

`/zfs3/scratch/saram_lab/PNC/data/genotype/qc/EAsubjectOnly/post-align/compressed`

4) Upload the QC'ed "aligned" genotype data to Michigan Imputation Server

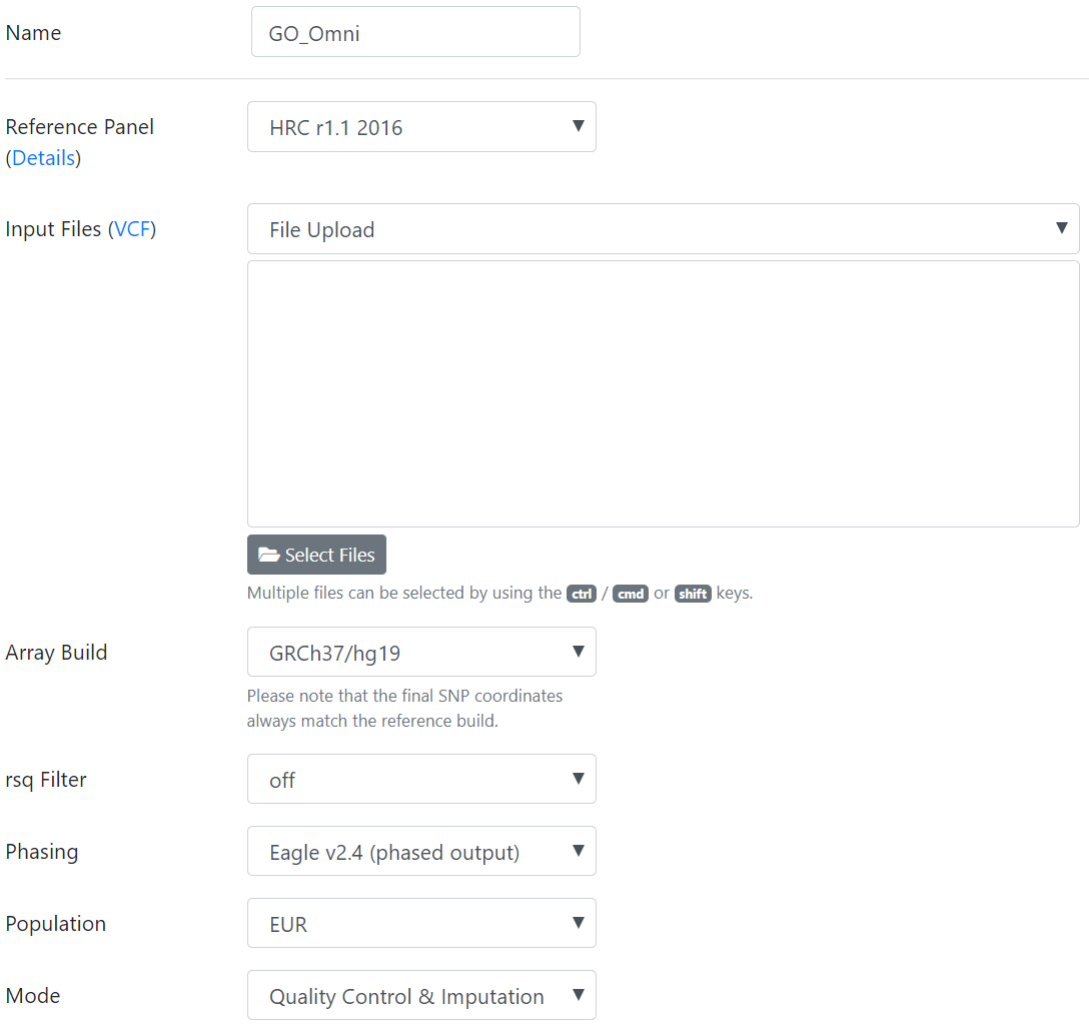
MIS: Getting Started Guide

Follow the MIS Getting Started Guide, and upload the compressed chromosome files using sftp, for example, the sftp command for one chromosome file could be: `sftp://orenthal.stat.ubc.ca/path/to/file/GO_Omni.chr1.vcf.gz`.

Note that when we did this, MIS claimed that separating files could be done with a space between file names, however this did not work. Inserting a full line break between file names did however work, eg:

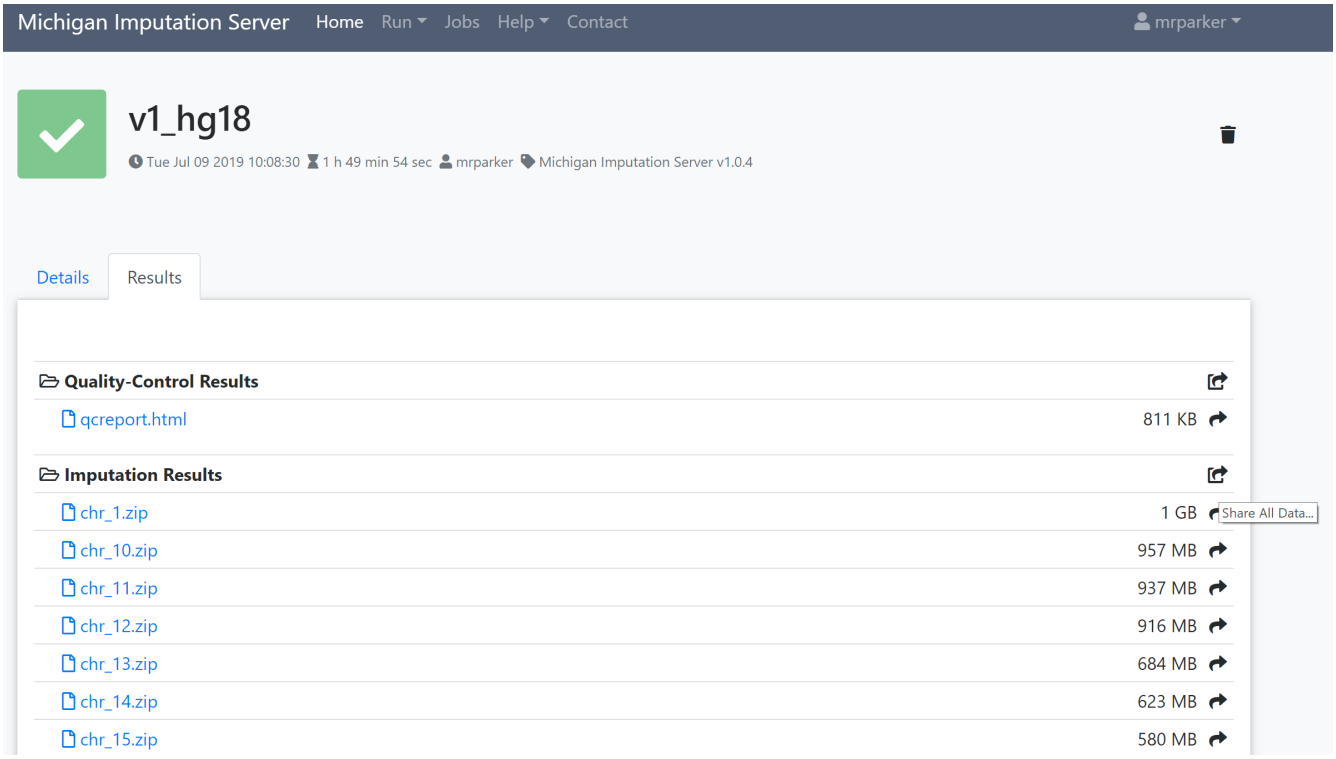
```
sftp://orenthal.stat.ubc.ca/path/to/files/GO_v3.chr1.vcf.gz

sftp://orenthal.stat.ubc.ca/path/to/files/GO_v3.chr2.vcf.gz

sftp://orenthal.stat.ubc.ca/path/to/files/GO_v3.chr3.vcf.gz

(up to chr22)
```

Here is an example of what the job submission would look like on the MIS website:



Imputation will take some time, and the progress can be monitored through the jobs dashboard on the MIS website. For us, it took between 2 and 20 hours per chip.

Once the imputation has finished, we then need to download the imputed files, click on the share all button to get the `wget` commands necessary:



Run the given commands to download the imputed files, place the different sets of chr zip files into their own folders, and unzip (this will give you eg `chr_1.dose.vcf.gz`).

For example:

```
wget https://imputationserver.sph.umich.edu/share/results/1234512345/chr_1.zip
unzip chr_1.zip
mv chr1.dose.vcf.gz ./GO_Omni/chr1.dose.vcf.gz
```

5) Perform post imputation reheadering/annotating

After imputation, we would like to reheader the chromosome files, and to annotate them with rsID, since those were replaced during imputation by CHR:POS IDs.

Note: Steps 5 and 6 are heavily modified from GWAS QC, `post_impute_merge_hrc.sh`. Due to the file sizes and computation time for dealing with these large and numerous chromosome files, the task was split so that each file could be reheadered as a separate job, with merging done in a new script after all reheadering is completed.

First, we need to download the **hg19 reference FASTA** file and the **HRC** file:

```
wget --timestamping 'ftp://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/*'
gunzip *.fa.gz
echo "$(ls chr*.fa | sort -V | grep -vP 'chr[^X|Y|\d]'; ls chr*.fa | sort -V |
    grep -vP 'chr[\d|X|Y]')" | xargs cat > hg19.fa
cat hg19.fa | sed 's/>chr/>/g' > hg19.nochr.fa

# just checking that the files contain actual genomics data
# cat hg19.nochr.fa | grep -v "N" > check_hg19.txt

wget ftp://ngs.sanger.ac.uk/production/hrc/HRC.r1-1/HRC.r1-1.GRCh37.wgs.mac5.sites.vcf.gz

# the HRC file needs to be indexed
module load bcftools
```

```
bcftools index HRC.r1-1.GRCh37.wgs.mac5.sites.vcf.gz
```

Before we reheader and annotate, we also need to index the imputed `*.vcf.gz` files, run this for each chip:

```
#!/bin/bash

#PBS -q medium

#PBS -N indexOmni
#PBS -l mem=10gb
#PBS -l walltime=10:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

module load bcftools

cd /zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_Omni

for i in {1..22}
do
        bcftools index chr${i}.dose.vcf.gz
done
```

Now we are ready to reheader and annotate.

Script for reheadering:

post_impute_reheader.sh

```
#!/usr/bin/bash

# Reheader the results of imputation.
# Perform corrections of errors in Michigan Imputation Server's output
# Including:
# 1. reheadering
# 2. recoding REF and ALT alleles

# Input: folder specifying dataset; number specifying chromosome number; assume each folder has chr1 through chr22
#        and each is named chr1.dose.vcf.gz, chr2.dose.vcf.gz, etc.
#        (this is how MIS likes to give you the output)
# Output: reheadered chromosome file

# Required: BCFTools, samtools, tabix, hg19 reference FASTA file
#           (hg19.nochr.fa) where chromosomes are named "1", "2", instead of
#           "chr1", "chr2", etc.

source ./params/pre_impute_params
missing_header_line_file=scripts/missing_header_lines.txt
missing_header_line_number=11
hg19=/zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/hg19FASTA/hg19.nochr.fa
hrc=/zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/HRC.r1-1.GRCh37.wgs.mac5.sites.vcf.gz

chrNum=$1

echo chr$chrNum: Current Date and Time is: `date +"%Y-%m-%d %T"`
merge_args=""
outputFilename=""
dirname=$2
echo "chr$chrNum: dirname=$dirname"
dataname=$(basename $dirname)
chrfname=chr$chrNum.dose
echo "chr$chrNum: dataname=$dataname"
```

```
echo "chr$chrNum: Beginning reheader of /$chrfname.vcf.gz"

# reheader
tabix -H $dirname/$chrfname.vcf.gz > $tmpdir/$dataname.$chrfname.header
sed -i "${missing_header_line_number}r $missing_header_line_file" $tmpdir/$dataname.$chrfname.header

bcftools reheader -h $tmpdir/$dataname.$chrfname.header -o $tmpdir/$dataname.$chrfname.reheadered.vcf.gz
      $dirname/$chrfname.vcf.gz

echo "chr$chrNum: Reheader of /$chrfname.vcf.gz completed"

echo "chr$chrNum: Rebuilding index of /$chrfname.vcf.gz"
# rebuild index
bcftools index $tmpdir/$dataname.$chrfname.reheadered.vcf.gz

bcftools annotate -a $hrc -c CHROM,POS,ID -O z -o $tmpdir/$dataname.$chrfname.annotated.vcf.gz
      $tmpdir/$dataname.$chrfname.reheadered.vcf.gz
rm -f $tmpdir/$dataname.$chrfname.reheadered*
tabix $tmpdir/$dataname.$chrfname.annotated.vcf.gz

#recode REF and ALT (mutate the tmp file!)
bcftools norm -d both -N -c ws -f $hg19 -O z -o $tmpdir/$dataname.$chrfname.normed.vcf.gz
      $tmpdir/$dataname.$chrfname.annotated.vcf.gz
rm -f $tmpdir/$dataname.$chrfname.annotated*
tabix $tmpdir/$dataname.$chrfname.normed.vcf.gz
echo "chr$chrNum: Finished recoding"
```

The above script **would have been** called once per chip, with a job array indexing over chromosome number (1-22). However, the PBS arrays were not working on Orenthal at the time of computing. Thus the script was instead called once per chip per chromosome (without array indexing).

This is what the calling script would look like with array indexing:

03a-reheader_Omni.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N reheaderOmni
#PBS -l mem=10gb
#PBS -l walltime=10:00:00
#PBS -l procs=1
#PBS -M mrparker909@gmail.com
#PBS -m abe

#PBS -t 1-22

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

echo "Changing to directory containing gwasqc."
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc
echo "Starting script at `date`"

module load plink2
module load R
module load vcftools # for vcf-sort
module load samtools
module load htslib   # for bgzip
module load bcftools

GO_Omni="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_Omni/"
GO_Quad="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_Quad/"
GO_v3="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_v3/"
GO_v1_hg18="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_v1_hg18/"

sh post_impute_reheader.sh $PBS_ARRAYID $GO_Omni
```

Here is the script that was actually used (varying chr from 1 to 22 and varying the chip to be Omni, Quad, v1_hg18, and v3):

```
#!/bin/bash
#PBS -q medium

#PBS -N reheaderOmni
#PBS -l mem=30gb
#PBS -l walltime=4:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

echo "Changing to directory containing gwasqc."
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc
echo "Starting script at `date`"

module load plink2
module load R
module load vcftools # for vcf-sort
module load samtools
module load htslib   # for bgzip
module load bcftools

chr=1
GO_Omni="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_Omni/"

sh post_impute_reheader.sh $chr $GO_Omni

echo "script completed at `date`"
```

Note that each reheadering (one per chromosome per chip) will take about 2-4 hours to complete, so that sequential reheadering is not advisable (it would take over 200 hours to sequentially reheader in our case).

The reheadered files are stored here: `/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/`, under their respectively named folders.

The files were also renamed to remove the chip name suffix, this is necessary for running the Step 6 merge:

rename.sh

```
#!/bin/bash

chip="GO_v3"

for i in {1..22}
do
  mv ${chip}.chr${i}.dose.normed.vcf.gz reheadered.chr${i}.dose.normed.vcf.gz
  mv ${chip}.chr${i}.dose.normed.vcf.gz.tbi reheadered.chr${i}.dose.normed.vcf.gz.tbi
  mv ${chip}.chr${i}.dose.header reheadered.chr${i}.dose.header
done
```

6) Post Reheadering: Merge Files

Script for merging: post_reheader_merge.sh

post_reheader_merge.sh

```
#!/usr/bin/bash

# Merge the results of imputation into one dataset
# In case the output of imputation is chromosome-by-chromosome,
# merge across datasets so there is one file per chromosome.

# Input: starting chromosome number; ending chromosome number; folders specifying dataset;
#        assume each folder has chr1 through chr22
#        and each is named chr1.dose.normed.vcf.gz, chr2.dose.normed.vcf.gz, etc.
#        (after we do the reheadering, this is how MIS likes to give you the output)
# Output: 22 chromosome files for the merged dataset

# Required: BCFTools, samtools, tabix, hg19 reference FASTA file
#           (hg19.nochr.fa) where chromosomes are named "1", "2", instead of
#           "chr1", "chr2", etc.

source ./params/pre_impute_params
missing_header_line_file=scripts/missing_header_lines.txt
missing_header_line_number=11
hg19=/zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/hg19FASTA/hg19.nochr.fa
hrc=/zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/HRC.r1-1.GRCh37.wgs.mac5.sites.vcf.gz

start=$1
end=$2

for i in `seq $start $end`
do
  (
  echo chr$i: Current Date and Time is: `date +"%Y-%m-%d %T"`
  merge_args=""
  outputFilename=""
  for dirname in "${@:3}"
  do
    echo "chr$i: dirname=$dirname"
    dataname=$(basename $dirname)
    chrfname=chr$i.dose

    merge_args="$merge_args $tmpdir/$dataname.$chrfname.normed.vcf.gz"
    outputFilename="$outputFilename+$dataname"
  done
  # merge
  echo "chr$i: Merging files: $merge_args"
  merge_args=${merge_args:1}
  outputFilename=${outputFilename:1}.$chrfname.vcf.gz
  time bcftools merge $merge_args -O z -o $outdir/$outputFilename
  ) &
done
wait
```

Here is the script we ran to complete the merge:

04-merge.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N MergeChips
#PBS -l mem=10gb
#PBS -l walltime=60:00:00
#PBS -l procs=1
#PBS -M mrparker909@gmail.com
#PBS -m abe

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt
```

```
echo "Changing to directory containing gwasqc."
cd /zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc
echo "Starting script at `date`"

module load plink2
module load R
module load eigensoft
module load shapeit
module load impute2
module load vcftools # for vcf-sort
module load samtools
module load htslib   # for bgzip
module load bcftools

GO_Omni="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_Omni/"
GO_Quad="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_Quad/"
GO_v3="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_v3/"
GO_v1_hg18="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/imputed_premerge/GO_v1_hg18/"

sh post_reheader_merge.sh 1 22 $GO_Omni $GO_Quad $GO_v3 $GO_v1_hg18

echo "script completed at `date`"
```

The 22 merged chromosome files are located: `/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/merged`

They will need to be compiled into a single data set before PCA can be done to extract the SNP principal components. Chromosome merging not implemented in plink2 yet, a workaround is to use "bcftools concat" on the VCF files.

05-final-merge.sh

```
#PBS -q medium
#PBS -N MergeChromosomes
#PBS -l mem=30gb
#PBS -l walltime=100:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

module load samtools
module load htslib   # for bgzip
module load bcftools

path="/zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/merged"
suffix=".vcf.gz"
c1=$path/chr1$suffix
c2=$path/chr2$suffix
c3=$path/chr3$suffix
c4=$path/chr4$suffix
c5=$path/chr5$suffix
c6=$path/chr6$suffix
c7=$path/chr7$suffix
c8=$path/chr8$suffix
c9=$path/chr9$suffix
c10=$path/chr10$suffix
c11=$path/chr11$suffix
c12=$path/chr12$suffix
c13=$path/chr13$suffix
c14=$path/chr14$suffix
c15=$path/chr15$suffix
c16=$path/chr16$suffix
c17=$path/chr17$suffix
c18=$path/chr18$suffix
c19=$path/chr19$suffix
c20=$path/chr20$suffix
c21=$path/chr21$suffix
c22=$path/chr22$suffix
```

```
out=$path/hrc_merged.vcf.gz

bcftools concat $c1 $c2 $c3 $c4 $c5 $c6 $c7 $c8 $c9 $c10 $c11 $c12 $c13 $c14 $c15 $c16
      $c17 $c18 $c19 $c20 $c21 $c22 -o $out -O z
```

7) Quality Control, LD and IBD Pruning

Perform LD pruning, and IBD pruning. This helps with dimension reduction for performing Principal Components analysis.

06-final-qc.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N Final_QC
#PBS -l mem=10gb
#PBS -l walltime=4:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

module load R
module load plink2

cd /zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/merged/

inFile="hrc_merged"
outFile="hrc.cleaned"
assayFile="/zfs3/scratch/saram_lab/PNC/data/genotype/qc/EAsubjectOnly/post-align/assayed.txt"
ibdScript="/zfs3/users/matthew.parker/matthew.parker/PBSscripts/GenotypeImputation/gwasqc/scripts/ibd-cut.R"

# make binary files
plink --vcf ${inFile}.vcf.gz --make-bed --out tmp

# scan for incorrect strand assignment
# see plink.flipscan for main report
# echo "Beginning flip-scan..."
# plink --bfile tmp --flip-scan

echo "Beginning LD pruning..."
# LD-based pruning
plink --bfile tmp --extract $assayFile --indep-pairwise 1500 150 0.2
plink --bfile tmp --extract plink.prune.in --genome --make-bed --out pruned

echo "Beginning IBD cuts..."
# Cryptic relatedness check
Rscript $ibdScript pruned.genome pruned.ibdcuts
plink --bfile pruned --remove pruned.ibdcuts --make-bed --out $outFile

echo "Done"
```

Results of LD Pruning:

- 39018346 total SNPs loaded (from 4740 individuals)
- 860219 assayed SNPs remaining
- Total genotyping rate is 0.976697.
- 860219 variants and 4740 people pass filters and QC.
- Pruned 58867 variants from chromosome 1, leaving 9877.

- Pruned 61892 variants from chromosome 2, leaving 9191.
- Pruned 50827 variants from chromosome 3, leaving 7917.
- Pruned 44779 variants from chromosome 4, leaving 7216.
- Pruned 45904 variants from chromosome 5, leaving 7420.
- Pruned 51534 variants from chromosome 6, leaving 7311.
- Pruned 40917 variants from chromosome 7, leaving 6574.
- Pruned 40894 variants from chromosome 8, leaving 5980.
- Pruned 35062 variants from chromosome 9, leaving 5735.
- Pruned 40356 variants from chromosome 10, leaving 6381.
- Pruned 37877 variants from chromosome 11, leaving 5869.
- Pruned 36916 variants from chromosome 12, leaving 6118.
- Pruned 29036 variants from chromosome 13, leaving 4665.
- Pruned 24472 variants from chromosome 14, leaving 4133.
- Pruned 22288 variants from chromosome 15, leaving 3969.
- Pruned 22477 variants from chromosome 16, leaving 4359.
- Pruned 19233 variants from chromosome 17, leaving 4042.
- Pruned 22171 variants from chromosome 18, leaving 3989.
- Pruned 13264 variants from chromosome 19, leaving 3232.
- Pruned 18444 variants from chromosome 20, leaving 3633.
- Pruned 10548 variants from chromosome 21, leaving 2013.
- Pruned 10559 variants from chromosome 22, leaving 2278.
- LD Results: 860219 assayed SNPs reduced to 121902 SNPs (4740 individuals)
- IBD Pruning: 4740 individuals reduced to 4481 individuals (121902 SNPs)

Resulting in 4481 individual subjects and 121902 SNPs ready for principal components analysis.

8) Extract 10 PCs from the imputed genotype data for modeling ancestry

We use only the assayed SNPs in computing the PCs, as per Ellingson et al. (2016):

"a thinned dataset created with PLINK and starting from assayed markers only is used to calculate PCs. The SMARTPCA tool is used to calculate PCs from this thinned dataset and identify outliers for removal"

The assayed SNPs are listed in the file `assayed.txt`, and we will restrict the PCA to that subset of SNPs.

To use smartPCA we first need to convert the plink data into EIGENSTRAT data.

- convert binaries to ped/map

```
plink --bfile hrc.cleaned --recode12
```

- convertf from ped/map to EIGENSTRAT (convertf documentation)

07a-convert.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N convertPCA
#PBS -l mem=10gb
#PBS -l walltime=10:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

module load eigensoft

cd /zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/postQC/

convertf -p convPAR.txt
```

convPAR.txt

```
genotypename: plink.ped
snpname:      plink.map
indivname:    plink.ped
outputformat:    EIGENSTRAT
genooutfilename:   pca.eigenstratgeno
snpoutfilename:    pca.snp
indoutfilename:    pca.ind
```

- perform PCA with smartpca (smartpca documentation)

07b-pca.sh

```
#!/bin/bash
#PBS -q medium

#PBS -N PCA
#PBS -l mem=10gb
#PBS -l walltime=10:00:00
#PBS -l procs=1

#PBS -o /zfs3/users/matthew.parker/matthew.parker/PBSlogs/log_$PBS_JOBID.txt
#PBS -e /zfs3/users/matthew.parker/matthew.parker/PBSlogs/err_$PBS_JOBID.txt

module load eigensoft

cd /zfs3/scratch/saram_lab/PNC/data/genotypeImputed/hrc/postQC/

smartpca -p eigensoft_params.txt
```

eigensoft_params.txt

```
genotypename: pca.eigenstratgeno
snpname:      pca.snp
indivname:    pca.ind
evecoutname:  pca.eigenvec
evaloutname:  pca.eigenval
altnormstyle: NO
numoutevec:   20
numoutlieriter: 5
numoutlierevec: 10
outliersigmathresh: 6.0
outlieroutname:     pca.outliers
snpweightoutname:   pca.snp.weights
```

The output will be several files:

- `pca.eigenvec`: file containing the eigenvectors
- `pca.eigenval`: file containing the eigenvalues
- `pca.outliers`: file containing the removed outliers
- `pca.snp.weights`: file containing the SNP weights