This document provides the steps necessary to run a Genome Wide Association Study (GWAS) analysis.

1 Introduction

2 Data Files and Pre-Processing

To analyize the Plink files (*.bim*, *.fam*, *.bed*) in R, the "snpStats" package is required.

```
source("http://bioconductor.org/biocLite.R")
biocLite("snpStats")
library(snpStats)
```

2.1 Data Files

The .fam File

The .fam file is a matrix that identifies each study participant. It contains 6 columns which identify each individual by "Family ID Number", "Sample ID Number","Paternal ID Number", "Maternal ID Number", "Sex", and "Phenotype". There are n rows, one for each individual. Note that not all of these columns may contain information depending on the nature of the data collected and the strategies for analysis. In the example to follow, a separate phenotype file will be read in, containing quantified phenotype data.

```
genoFam <- read.table("genodata.fam")
colnames(genoFam) = c("FamID", "IndID", "PatID", "MatID", "sex", "phenotype")
```

The .bim File

The .bim file is a matrix that provides information about each SNP in our study. There are 6 colums that contain information for the "Chromosome Number", "rsNumber", "Genetic Distance", "Position ID", "Allele 1", "Allele 2". There are n rows, one for each SNP in the analysis. By default, the rsNumbers are read in as factor variables, and will be changed to character variables.

```
genoBim <- read.table("genodata.bim")
colnames(genoBim) = c("chr", "SNP", "GenDist", "BPPos", "g1", "g2")
genoBim$SNP <- as.character(genoBim$SNP)
```

The .bed File

The .bed file is a matrix that contains the genotypic data in binary format. It's the result of the conversion of .ped and .map files by the PLINK software. This is the largest of the three files because it contains every SNP in the study, as well as the genotype at this snp for each individual. In order to interpret the binary data, we need to employ the "read.plink()" function, from the "snpStats" package, to read in the .fam, .bim, and .bed functions together, and to interpret the binary data in the .bed file and put it into the proper format. The "snpStats" package is an update of the "snpMatrix" package, and creates a "snpMatrix" object. We will pull out the "genotype" slot of this object, which contains the genotype data, stored as a matrix of p columns, one for each SNP by "rsNumber", and n rows, one for each study participant by "Family ID Number", containing the genotype of each study participant at each individual SNP, as either "01", "02", or "03".

```
geno <- read.plink("genodata.bed", "genodata.bim", "genodata.fam")$genotype
```

The Phenotype File

The phenotype file contains a matrix, with the phenotype variables as columns and the participant IDs as the unit of observation.

```
pheno <- read.csv("phenodata.csv")
```

Only phenotypes of patients who have genotype data should be included in the genotype file.

```
phenoSub <- pheno[pheno$FamID %in% genoFam$FamID, ]
```

Next, make sure that the variables in the phenotype file are formatted correctly. Categorical variables, such as race and sex, are often coded as integers, and may need to be reformatted as a factor variable. This is performed using the "str()" function to check the format of each variable, and then the "as.factor()" function to format a variable to a factor. Note this can also be done in reverse, if there is a numeric variable stored as categorical data.

```
str(phenoSub)
phenoSub$race <- as.factor(phenoSub$race)
```

2.2 SNP Level Filtering

The "snpStats" package will allow us to perform analysis on the raw "geno" file. We will use the "col.summary()" function, which will produce a matrix with values for a variety of summary analyses at each SNP. This we will allow us to filter our data based on a certain set criteria. For this study we have chosen to filter our data as follows:

Call Rate

The call rate is the most straight forward of the three criteria, as it means to only keep that have genotype data above a certain threshold. In the following example we chose a call rate of 90 %. This is essentially stating that we only want to keep the SNPs for which there is data for more than or equal 90% of the study particpants, or for which there is missing data for less than or equal to 10% of the study participants.

Minor Allele Frequency

If every study participant is homozygous at a given SNP, than there is nothing to infer about the relationship between genotype and phenotype, as each participant has the same genotype at this SNP. This is a common occurence with the major allele, as it is often much more common in a given population. Therefore, we it is important to keep only the SNPs for which we have a certain proportion of minor alleles present in the study sample. In the example to follow we will use a Minor Allele Frequency, that will remove SNPs for which the minor allele frequency is less than 1 %.

Hardy-Weinberg Equilibrium

The Hardy-Weingberg Equilibrium is based off the principal that if two alleles are possible for a given genotype, then the sum of the probabilities of recieving each allele is 1.

$$p + q = 1$$

Given the frequency of a certain allele we can, therfore, not only predict the frequency of the other allele, but also the proportion of homozygous recessive, heterozygous, and homozygous dominant individuals in a population. If violated can assume that there was a possible genotyping error, and we must remove the data for this SNP. We can't expect our samples to exhibit this principle perfectly, however we can calculate the p-value that this principle is being followed, and set a threshold to keep only data with Hardy-Weinberg equlibrium p-values below a certain p-value. In the example to follow we have chosen a p-value of 0.001.

```
call <- 0.9
minor <- 0.01
hardy <- 0.001
snpsum.col <- col.summary(geno)
use <- (!is.na(snpsum.col$MAF) & snpsum.col$MAF > minor) & (!is.na(snpsum.col$z.HWE) &
    snpsum.col$z.HWE^2 < qnorm(hardy/2)^2) & snpsum.col$Call.rate >= call  #This creates a
use[is.na(use)] <- FALSE
genoBim <- genoBim[use, ]  #This will filter the .bim file
geno <- geno[, use]  #This will filter our genotype file.
```

Formatting the Genotype File

Once we have filtered our data using the formatting necessary for the "snpStats" package we will need to convert our genotype file into a matrix in numeric format.

```
genoNum <- as(geno, "numeric")
```

Our final genotype ("genoNum") matrix contains the genotype for each individual at each individual SNP, stored as either the number of Major or Minor alleles, depending on how the data was originally formatted. For example, if the data was formatted to count the number of major alleles, the meaning of the genotype data is coded as follows:

- 0 = Homozygous Recessive
- 1 = Heterozygous
- 2 = Homozygous Dominant

Make a table one of the columns to make sure the data is formatted correctly.

```
table(genoNum[, 1])
```

2.3 Principal Components

Principal Component Analysis (PCA) is commonly performed, as a means for adjusting for population substructure. The goal is to account for as most of the genetic variation between the study participants. This is vital to a GWAS because it allows us to account for varying allele frequencies among populations of various ethnic backgrounds, which could cause confounding, and produce false positive results.
If we want to report the number of PCs needed to account for 90% of the variation I wrote the following code. It suggests that 1020 PCs are needed of 1184 total. 10 PCs account for less than 1% of the total variation.

```
pervar <- cumsum(evv$values)/sum(evv$values)
var <- 0.9
pcs <- which(abs(pervar - var) == min(abs(pervar - var)))
pcs
```

```
num.princ.comp <- 10
```

```
xxmat <- xxt(geno, correct.for.missing = FALSE)
evv <- eigen(xxmat, symmetric = TRUE)
pcs <- evv$vectors[, 1:num.princ.comp]
evals <- evv$values[1:num.princ.comp]
btr <- snp.pre.multiply(geno, diag(1/sqrt(evals)) %*% t(pcs))
pcs <- snp.post.multiply(geno, t(btr))
colnames(pcs) <- paste("pc", 1:num.princ.comp, sep = "")
rm(xxmat)
```

Next, we will merge the principal components with the phenotype file. In order to do so, we must first attach participant IDs to the principal components.

```
pcs <- data.frame(FamID = genoFam$FamID, pcs)
phenoSub <- merge(phenoSub, pcs, by.x = "FamID", by.y = "FamID", all.x = TRUE)
```

3 GWAS Analysis

To run the linear regression on our data, we first need to install the "parallel" package to be able to run a parallel analysis, an analysis on multiple cores. l

```
library(parallel)
```

We create a GWAS function that will run our analysis. This function takes a subset of our data "tempSNP" of ID numbers and RS Numbers. The linear regression model compares our filtered and merged dataset (in this case we are running an analysis on our baseline data) and adjusting for the variables of interest: The "a" matrix contains the SNP genotype on the x-axis and the phenotype on the y-axis. This function then maps the regression model for the filtered, merged dataset (in this case we are running an analysis on our baseline data) and adjusting for the variables of interest: Age, Sex, Race, the ten PCs and SNPs. The "out" matrix pulls just the row that contains the estimate for the SNP from the linear model summary.

```
ldlGWAS <- function(rsNumber) {
    print(rsNumber)
    tempSNP <- data.frame(rpid = row.names(genoNum), snp = genoNum[, rsNumber])
    dat <- merge(phenoSub, tempSNP, by.x = "rpid", by.y = "rpid", all.x = TRUE)
    a <- summary(lm(fldl_wk0 ~ age + sex + raceth + pc1 + pc2 + pc3 + pc4 +
        pc5 + pc6 + pc7 + pc8 + pc9 + pc10 + snp, data = dat))
    out <- as.matrix(a$coefficients["snp", ])
```

```
    out
}
```

We use "mclapply()" to run parallel analysis on 8 different cores to save time. Since "mclapply()" must analyze a list, we first make "rsVec" into a list. After running the analysis, we put the data back together.

```
rsVec <- as.matrix(colnames(genoNum))
rsVec <- as.list(rsVec)
start <- Sys.time()
aa <- mclapply(rsVec, ldlGWAS, mc.cores = 8)
out <- do.call(cbind, aa)
fldl_wk0_p3 <- data.frame(t(out))
```

After analysis, We have a new dataset that we write to a file that contains rsNumber, Estimate, Standard Error, Zvalues and Pvalues

```
names(fldl_wk0_p3) <- c("Estimate", "SE", "Zvalue", "Pvalue")
fldl_wk0_p3$rsNumber <- rsVec
fldl_wk0_p3 <- as.matrix(fldl_wk0_p3)
write.csv(fldl_wk0_p3, "/home/ramoser/fldl_wk0_p3")
```

We can now sort the results by p-value to find our strongest associations.
From this GWAS we can see a high level of association between baseline LDL and snp rsNumber, rs7412, on the APOE gene, with a p-value of 2.69e-12.
*I'm not sure how to finish. I think we should go on to discuss the shear number of analysis we are running and how the Bonferroni correction returns p-value threshold of:

```
0.05/834279

## [1] 5.993e-08
```

If we were to only go by this number we'd only associate high significance to rs7412. This is where we could point to other analysis such as MixMAP.