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

# 1 Introduction

# 2 Data Files and Pre-Processing

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

## 2.1 Data Files

In order to perform a genome wide association study, it is essential to obtain genotype data, properly formatted for analysis. In this example, the "snpStats" package is utlized. The "snpStats" package, is an update of the "snpMatrix" package, and creates a "snpMatrix" object. It is freely available in R, and contains functions for reading in ".fam", ".bed", and ".bim" files. These file types, are created from ".ped" and ".map" files using, PLINK, an open source software package written in C/C++. ".ped" files contain participant identification information, as well as the genotype data for each participant, coded as the first letter of the nucleotide (A,C,T or G)."map" files contain the rsNumber and information regarding the location of each SNP. Upon conversion, the ".fam" files will contain the participant identification information,".bed" files will contain a binary version of the genotype data, and the ".bim" files will contain the same information as the ".map" files, as well as the nucleotide identifiers (A, C, T, or G) from the ".ped", files. The data contained in each file is demonstrated in figure 1.

### .fam File

The .fam file is a 6 column matrix which identifies each participant by "Family ID Number", "Sample ID Number","Paternal ID Number", "Maternal ID Number", "Sex", and "Phenotype". There is a row for each individual. Note that not all of these columns may contain information depending on the nature of the data collected and the type for analysis. As is often the case when when analyzing quanitfied traits, in the example to follow a separate phenotype file will be read in.

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

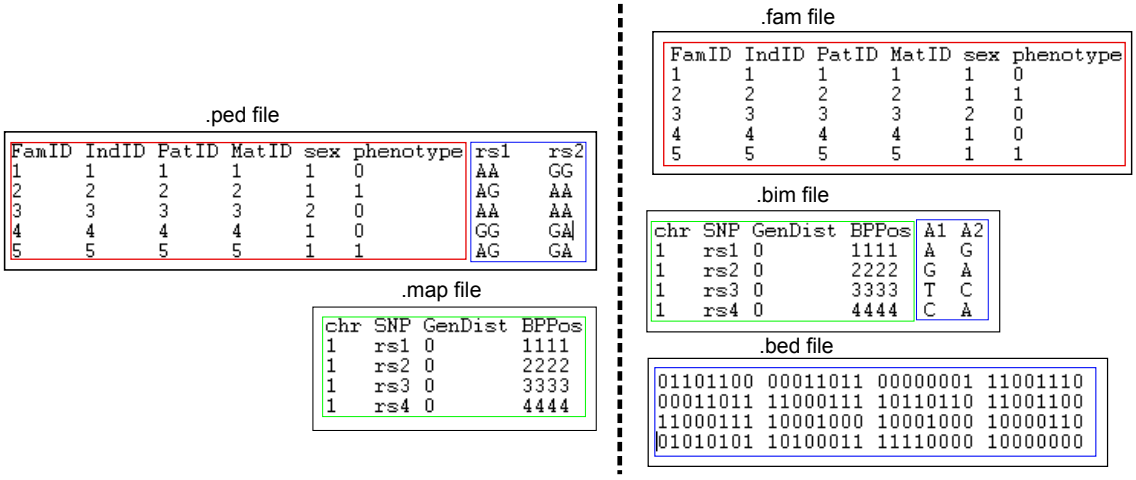


Figure 1: Data Contained in PLINK files

## .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 is a row 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)
```

## .bed File

As previously mentioned, the ".bed" file contains the genotypic data in binary format. 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. We will pull out the "genotype" slot of this object, which contains the genotype data, stored as a matrix with a column for each SNP, and a row 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 of 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 keep, only study participants 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 to 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, then 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, 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, which will remove SNPs for which the minor allele frequency is less than 1 %.

### Hardy-Weinberg Equilibrium

The Hardy-Weingberg Equilibrium is based off the principle that if two alleles are possible for a given genotype, then the sum of the probabilities of having a given allele at each chromosome is 1. This can be represented as:

$$p + q = 1$$

where $p$ and $q$ represent the probabilities of either allele. Therefore, given the frequency of a certain allele, we should not only be able predict the frequency of the other allele, but also the proportion of homozygous major, heterozygous, and homozygous dominant individuals in a given population using the formula:

$$p^2 + 2pq + q^2 = 1$$

Here, the proportion of the homozygous major genotype is predicted as $p^2$, the hetrozygous genotype is predicted as $2pq$, and the proportion of the homozygous recessive genotype is predicted as $q^2$. Since we cannot expect the observed values of each genotype to precisely match the expected values in our sample, a $\chi^2$-test statistic is used to measure the departure between the expected number of a given genotype and the observed number of a given genotype to obtain a p-value. If the p-value is greater than a certain cut-off, we assume that the allele frequencies comply with Hardy-Weinberg Equilibrium, and therefore the stringency of Hardy-Weinberg filtering is enhanced by larger p-value cutoffs.
Though Hardy-Weinberg Equilibrium can be violated via population admixture and stratification, and in such cases we would still consider the associations between genotype and phenotype to be informative, they may also occur as a result of genotyping errors. Therefore, it is common practice to remove the data for SNPS that violate Hardy-Weinberg Equilibrium. In this example we will use a commonly used cutoff of $p < 0.001$.

```
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
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:

- NA = Missing
- 0 = Homozygous Minor
- 1 = Heterozygous
- 2 = Homozygous Major

Alternatively if it is formatted to count the number of minor alleles, the meaning of the genotype data is coded as:

- NA = Missing
- 0 = Homozygous Major
- 1 = Heterozygous
- 2 = Homozygous Minor

The latter demononstrates how the genotypes are coded by genome-wide analyses through PLINK. Therefore we will want to recode data that is coded as the former.

```
flip <- function(x) {
    zero2 <- which(x == 0)
    two0 <- which(x == 2)
    one1 <- which(x == 1)
    mat <- matrix(data = NA, nrow = nrow(x), ncol = ncol(x))
    rownames(mat) <- rownames(x)
    colnames(mat) <- colnames(x)
    mat[zero2] <- 2
    mat[two0] <- 0
    mat[one1] <- 1
    mat
}

mafTest <- (2 * (sum(genoNum[, 1] == 0, na.rm = TRUE)) + sum(genoNum[, 1] ==
    1, na.rm = TRUE))/(2 * sum(!is.na(genoNum[, 1])))
mafCalc <- snpsum.col[colnames(genoNum)[1], "MAF"]
genoNum <- if (mafTest == mafCalc) {
    flip(genoNum)
} else {
    genoNum
}
```

## 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
# pcs1<-which(abs(pervar-var)==min(abs(pervar-var))) pcs1
```

```
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 Genome-Wide Association Study (GWAS) Analysis

## 3.1 Fitting the Model

Multiple methods are used to analyse GWAS data: Linkage analysis, Admixture mapping and Association analysis. since we have a high number of SNPs and these subjects are assumed not to be correlated, we will be conducting an Association analysis. Genetic models are used to incorporate the interaction between alleles on homolougous chromosomes. Genetic models commonly used are Additive Models, Dominant Models and Recessive models.

### Additive

Additive models are used to evaluate additive structure and reveal associations that depend additively based on the allele classification. These models assume that if having a single minor allele will increase the quantitative trait we are interested in, $y$, by $\beta$, then having two minor alleles in the homologs will increase $y$ by $2\beta$.
To represent this model, let $A$ and $a$ represent the possible alleles at a given SNP locus where $A$ is the major allele and $a$ is the minor allele. Let $I(x_{i,k} = a)$ be an indicator for whether the allele on the $k$th homolog $(k = 1, 2)$ is the minor allele for individual $i = 1, \cdots, n$, where $n$ is the number of observations, then this model can be written as

$$g(E(y_i)) \quad = \quad \alpha + \beta[I(x_{i,1} = a) + I(x_{i,2} = a)] \tag{1}$$

### Dominant

Dominant models assume that the homologs conatain at least one major allele for $y$ to exist. Having one or more copies of $a$ will increase $y$ by $\beta$. This model can be written as:

$$g(E(y_i)) \quad = \quad \alpha + \beta I(x_{i,1} = a \text{ or } x_{i,2} = a) \tag{2}$$

RECESSIVE
Recessive models assume both homologs contain the minor allele for $y$ to exist. This model can be written as:

$$g(E(y_i)) \quad = \quad \alpha + \beta[I(x_{i,1} = a) * I(x_{i,2} = a)] \tag{3}$$

[**?**, p. 61]
To run the GWAS analysis the additive model of association should include demographic, clinical, environmental and other relevant covariates such as Age, Sex and Race. This model can be written as

$$g(E(\underset{\sim}{Y})) \quad = \quad \mathbf{X}\beta + \mathbf{P}\alpha + \gamma_\mathbf{j}\text{SNP}_\mathbf{j} \tag{4}$$

where $\gamma_j \text{SNP}_j$ is the SNP effect, $X$ is a vector of the relevant covariates and $P$ is a vector of principle comonents, the demension of which depends on the number of principle components chosen to be used in the analysis.

### 3.2 Example of GWAS Analysis Using R

One way to run a GWAS analysis is to creat a We create a GWAS function. 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 adjusts for the variables of interest: The "a" object computes and returns a list of summary statistics of the fitted linear model, "a" 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.

```
GWAS <- function(rsNumber) {
    print(rsNumber)
    tempSNP <- data.frame(FamID = row.names(genoNum), snp = genoNum[, rsNumber])
    dat <- merge(phenoSub, tempSNP, by.x = "FamID", by.y = "FamID", all.x = TRUE)
    a <- summary(glm(fldl_wk0 ~ age + sex + raceth + pc1 + pc2 + pc3 + pc4 +
```

```
        pc5 + pc6 + pc7 + pc8 + pc9 + pc10 + snp, family = gaussian, data = dat))
    out <- as.matrix(a$coefficients["snp", ])
    out
}
```

### 3.2.1 Running a Parallel 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.

```
library(parallel)
```

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)
aa <- mclapply(rsVec, GWAS, mc.cores = 8)
out <- do.call(cbind, aa)
fldl.wk0.p3 <- data.frame(t(out))
```

## 3.3 Summarizing and Visualization

We now have a new dataset that contains rsNumbers, Estimates, Standard Errors, 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")
```

Our new dataset contains only part of the variables needed to accurately summarize the data. We need to be able to determine which SNP cooresponds to which chromosome, and where that chromosome is located. To do so we load in a dataset that contains this information for the RS numbers and merge with our existing dataset. Note that the chromosome location, that is the variable coord1, depends on which build you are using. We are currently using build...

```
baseline1 <- read.csv("fldl_wk0_p3.txt", header = T)
baseline1 <- baseline1[, -1]

merge1 <- read.csv("SnpCoord.csv")
merge1$rsNumber <- merge1$snp
merge1 <- merge1[, c(2, 3, 5)]

baseline2 <- merge(baseline1, merge1, by = "rsNumber")
```

In this example we are interested in P-values $< 5 * 10^{-5}$. Thus, we create a subset of SNPs with those P-values and summarize them in a table. Furthermore, we drop "chr" to report just chromosome numbers and save the resulting table.

```
baseTable <- baseline3[baseline3$Pvalue <= 5 * 10^(-5), ]

baseline3$chr <- substring(baseline3$chr, 4, 5)
baseline3$chr <- as.numeric(as.character(baseline3$chr))

write.csv(baseline3, "baseline.txt")

xtable(baseTable)
```

Now that we have a complete dataset with SNPs, chromosome, gene type, etc., we can create a Manhattan Plot to view the data. Below is a function that will create a Manhattan Plot. A Manhattan plot vizualizes where chromosome numbers are displayed along the $x$-axis an the negative logarithm of the association P-value for each SNP on the Y-axis.

```
one <- read.csv("baseline.txt", header = T)
position <- one$coord1/1e+06
chr <- one$chr
nsnp <- as.numeric(table(chr))[1:22]
pvalue <- one$Pvalue

pos <- NULL
pos[1:nsnp[1]] <- position[chr == 1]
pval <- NULL
pval[1:nsnp[1]] <- pvalue[chr == 1]
col <- NULL
col[1:nsnp[1]] <- "blue"
TextPos <- NULL
```

```
TextPos[1] <- mean(pos)
for (chr in 2:22) {
    TotalPrev <- length(pos)
    start <- sort(pos)[TotalPrev]
    pos[(TotalPrev + 1):(TotalPrev + nsnp[chr])] <- start + position[chr ==
        chr]
    pval[(TotalPrev + 1):(TotalPrev + nsnp[chr])] <- pvalue[chr == chr]
    if (chr%%2 == 0) {
        a = "lightblue4"
    }
    if (chr%%2 == 1) {
        a = "blue"
    }
    col[(TotalPrev + 1):(TotalPrev + nsnp[chr])] <- rep(a, nsnp[chr])
    TextPos[chr] <- start + mean(position[chr == chr])
}
```

Now, we can creat the Manhattan Plot.

```
bitmap(file = "manhattan.jpeg", type = "jpeg", width = 10, height = 6, res = 432,
    pointsize = 8)
par(mfrow = c(1, 1), las = 1, xaxs = "i", yaxs = "i")
plot(pos, -log10(pval), pch = 20, col = col, xlab = "", ylab = "-log10(p-value)",
    axes = F, ylim = c(0, 8))
axis(2)
abline(h = 0)
text(TextPos, -0.5, seq(1, 22, by = 1), xpd = TRUE)
title("Manhattan Plot")
abline(h = 3, col = "green")
abline(h = 4, col = "orange")
abline(h = 5, col = "red")
abline(h = 6, col = "purple")
abline(h = 7, col = "cyan")
dev.off()
```

# 4 Discussion

## 4.1 Gene Based Approaches

In this case, we are only interested in looking at the following gene types: exonic, intronic, splicing, UTR3, UTR5, downstream, exonic;splicing, and upstream. Thus, we keep only

these types and sort by P-values.

```
# Keeping Gene Types:exonic, intronic, splicing, UTR3, UTR5, downstream,
# exonic;splicing, upstream

keep1 <- c("exonic", "intronic", "splicing", "UTR3", "UTR5", "downstream", "exonic;splicing
    "upstream")
baseline3 <- baseline2[is.element(el = baseline2$type, set = keep1), ]
baseline3 <- baseline3[order(baseline3$Pvalue), ]
```

We can now sort the results by p-value to find our strongest associations.

```
fldlwk0 <- read.csv("fldlwk0GWAS")
wk0bypval <- fldlwk0[order(fldlwk0$Pvalue), ]
wk0bypval <- wk0bypval[, -c(1, 3, 12)]
# This is to remove numbered columns added by saving as a '.csv' file
wk0bypval[1:10, ]
```

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.