**MalariaGEN Practical**

**Wellcome Trust Advanced Course – September 2015 –Durban, South Africa**

**Introduction**

This data uses sample data from MalariaGEN to illustrate how to use R to read in genotype data and clinical and other phenotype data, to perform some basic quality control (QC), to describe the data and to run association analyses. It assumes that you have some basic knowledge of 'R'

**Required Inputs**

The following R library should have been be installed and will be loaded for use as required,

* *genetics*

The following files are required. They are available here in a directory called MalariaGEN'

* *clinical.csv* Clinical phenotype data
* *genotypes.csv* Genetic data (genotypes)
* *phenotypes.csv* Additional phenotype data
* *custom\_functions.R* Customised R functions

There is also a file called dictionary.txt which is a dictionary file giving the definitions and values used in the fields in *clinical.csv*

**Practical**

There are 3 sections in this practical

1. Load all data, match data in clinical, genotype and phenotype files and perform quality control of clinical and phenotype data.
2. Prepare a separate genotype file for analysis
3. Perform association analyses

**Section 1** **- Load all data, match data in clinical, genotype and phenotype files and perform quality control of clinical and phenotype data**

We upload several different data files and merge then to create a single file which we use for a case-control analysis. We will do a simple exploration of the data to get some sense of the contents and then perform some preliminary quality control (QC)

* 1. Reading data into R from a comma-delimited format

1. Find the R default working directory:

getwd()

To specify a particular directory, rather than the default, assign the required path to a suitable variable, e.g. filepath and set the default directory to that path

filepath <-"/media/ubuntu/data/GEIA/Practicals/03\_MalariaGEN/"

setwd(filepath)

NOTE: Folder separators are forward slash not Windows reverse slash!

This practical assumes that the path in *filepath* is the correct path for accessing the data files.

* 1. Upload the clinical files required into dataframes

1. Read genotype data file

Use *paste* to concatenate the filepath and filename

help(paste)

genotype\_file <- paste(filepath,"genotypes.csv", sep="")

Use read.csv to load the file into R

help(read.csv)

geno <- read.csv(genotype\_file, header = T)

1. Read clinical phenotype data file

clinical\_file <- paste(filepath,"clinical.csv", sep="")

clin <- read.csv(clinical\_file, header = T)

1. Read additional phenotype data file

phenotype\_file <- paste(filepath, "phenotypes.csv", sep="")

pheno <- read.csv(phenotype\_file, header = T)

1. Read map file

map\_file <- paste(filepath, "map.csv" , sep="")

map <- read.csv(map\_file, header = T)

* 1. Inspecting the data

1. Check the dimensions of each file

help(dim)

dim(geno) # 4190 rows by 73 columns

dim(clin) # 4190 x 16

dim(pheno) # 4190 x 18

1. View the files

*fix* is a useful R function which can be used in to view and change the data directly in the R editor

help(fix)

Note: This function is not functional in RStudio used here. Use Excel instead to view the .csv files directly or to view the whole file in R, just type the name of the file. However, if the file is large, just view selected rows and columns:

geno[1:6,1:7]

pheno [1:6,1:7]

Note: *[a,b]* – *a* specifies which rows and *b* specifies which columns to view.

Or, to simply view the first 6 rows and all columns:

head(clin)

Note: Each row in each file has a unique identifier *released\_key* which

identifies the individuals to whom the data in the row corresponds.

1. Check the column names.

help(names)

Note: Sometimes R may add an X at the beginning of a name and/or replace certain characters. E.g. An X is added to any filed name beginning with a number; fullstops replace brackets and other symbols.

names(geno) #73 fields

names(clin) #16 fields

names(pheno) #18 fields

1. Determine and check the column number for a particular field.

HbS.i0 <- which(names(geno) =="X11.5204808.rs334.HBD.hHbS")

HbS.i0 #44

names(geno)[44]

1. Summarise the data in each column.

help(summary)

summary(geno)

summary(clin)

summary(pheno)

Note: This is not always the easiest result to interpret if the function is used on the full data table.

* 1. Merging data

1. Merge the clinical and genetic data files.

Use *merge* to merge data from two files with common identifiers.

help(merge)

Providing two files are appropriately formatted and share a common identifier or key in any column (i.e. sample name), they can be merged for analysis. First identify the common identifiers or key in each file. In the clinical file, *clin* and the genetic file, *geno*, this is released\_key

clin$released\_key[1:10]

geno$released\_key[1:10]

cg.dat <- merge(clin,geno, by.x="released\_key", by.y="released\_key")

1. Merge the merged clinical and genetic data file with the phenotype file.

cgp.dat <- merge(cg.dat,pheno, by.x="released\_key", by.y="released\_key")

Check the dimension of the new file:

dim(cgp.dat)# 4190 x 105

Note: The new dataframe *cgp.dat*, created from merging of clinical phenotype, genetic and additional phenotype file has the same number of rows as each of the unmerged files

* 1. Subsetting data

1. Subset the data according to family membership.

In this dataset we have trios: child (C), father (F) and mother (M). This information is available from the field in the clinical data file named *family\_member.*

Use *unique* to see what types of record are present in a given variable

help(unique)

unique(cgp.dat$family\_member) # C F M

unique(cgp.dat[,3])

Use *table* to see how many and what type of records there are

help(table)

table(cgp.dat$family\_member)

# C F M

#2510 840 840

Here we wish to work with offspring only. We could just use an exclusion clause in each analysis to remove the parents but since we do not require parents at all, it is more efficient to create a new data set with parents removed.

To create a new dataset which only keeps samples of family\_member="C"

cgp.C.dat <- cgp.dat[which(cgp.dat$family\_member == "C"), ]

This selects only rows with family\_member value "C" and all columns

table(cgp.C.dat$family\_member)

# C F M

# 2510 0 0

* 1. Recoding, reviewing and checking (quality controls) the data.

1. Case control status.

Case control status is indicted in the field in the clinical data file named *TYPE:*

table(cgp.C.dat$TYPE)

# 1 2

# 1010 1500

Typical coding for cases and controls is 1 for cases and 0 for controls. Here we create a new variable called *status* with the typical coding:

cgp.C.dat$status <- NA

cgp.C.dat$status[cgp.C.dat$TYPE == 1] <- 1 #case

cgp.C.dat$status[cgp.C.dat$TYPE == 2] <- 0 #control

NOTE: In general it is good practice to create a new field so that you still have the original for reference, or in case you make any errors.

Check that the new variable corresponds to the old one.

table(cgp.C.dat$TYPE, cgp.C.dat$status)

NOTE: The *table* function can also be used to look at summaries across multiple variables

To calculate the proportion of controls and cases:

help(sum)

100\* (table(cgp.C.dat$status)/sum(table(cgp.C.dat$status)))

# 0 1

# 59.76096 40.23904

Round results for presentation:

round(100\* (table(cgp.C.dat$status)/sum(table(cgp.C.dat$status))),1)

# 0 1

#59.8 40.2

1. Ethnicity and location.

table(cgp.C.dat$ethnicity, cgp.C.dat$location)

Add the status variable:

table(cgp.C.dat$ethnicity, cgp.C.dat$location, cgp.C.dat$status)

For ease of viewing, add criteria to the function to display controls and cases separately by assigning indicators for cases and controls

help(which)

ictl <- which(cgp.C.dat$status==0)

icase <- which(cgp.C.dat$status==1)

Show ethnicity and location for controls:

table(cgp.C.dat$ethnicity[ictl],cgp.C.dat$location[ictl])

Show ethnicity and location for cases:

table(cgp.C.dat$ethnicity[icase],cgp.C.dat$location[icase])

When we look at location data or ethnicity or other similar data, we may want to group categories; either some groups are too small for power, or, for ethical considerations we may want to reduce the liklihood of identification of individuals or minority groups. Let's say we want to recode any locations that make up <10% of samples:

Look at the proportion of individuals at each location:

round(100\*(table(cgp.C.dat$location)/sum(table(cgp.C.dat$location))),2)

#Balau Clindau Dallau Findau Gunjau Monnau Nondau Pindau

#13.90 8.69 15.14 23.90 8.25 9.68 7.85 12.59

Assign location "other" to all groups < 10% => help(ifelse)

cgp.C.dat$locnew <- ifelse(cgp.C.dat$location == "Balau"|cgp.C.dat$location == "Dallau"| cgp.C.dat$location == "Findau"|cgp.C.dat$location == "Monnau"|cgp.C.dat$location == "Pindau" , as.character(cgp.C.dat$location) , "other")

NOTE: R uses | for  *or*. Since the variable being tested is the same in each *or* command, the following is equivalent

cgp.C.dat$locnew <- ifelse(cgp.C.dat$location %in% c("Balau","Dallau","Findau","Monnau","Pindau"),

as.character(cgp.C.dat$location), "other")

Check outputs:

table(cgp.C.dat$location, cgp.C.dat$locnew)

Repeat similarly for Ethnicity:

round(100\*(table(cgp.C.dat$ethnicity)/sum(table(cgp.C.dat$ethnicity))),1)

# CAN FAN JAN PAN WAN

29.9 29.9 16.7 16.7 6.8

There is only one group with ethnicity < 10%, so there is no need to group ethnicity data.

1. Gender

Clinical gender is available in the field in the clinical data file named *gender*.

table(cgp.C.dat$gender)

# F M

# 68 1185 1257

68 individuals have missing clinical gender, which is coded as a blank.

NOTE: R requires NA for missing data

Create a new variable called *clin\_gender* for clinical gender where NA is coded instead of a blank for individuals with missing gender:

cgp.C.dat$clin\_gender <- NA

cgp.C.dat$clin\_gender[cgp.C.dat$gender == "M"] <- 1

cgp.C.dat$clin\_gender[cgp.C.dat$gender == "F"] <- 2

We can now see the 68 rows of blanks genders are gone:

table(cgp.C.dat$clin\_gender)

# 1 2

#1257 1185

NOTE: R ignores the NA cells; if you want to table these as well, then include *exclude=NULL* in the table command:

table(cgp.C.dat$clin\_gender, exclude=NULL)

# 1 2 <NA>

#1257 1185 68

There are 3 SNPs in this data set that were typed exclusively to check gender. We can use these to confirm the DNA gender matches the clinical and also to possibly fill in any missing clinical genders. They are in columns 86:88 of *cgp.C.dat*.

names(cgp.C.dat[,c(86:88)])

table(cgp.C.dat$XY.11223656..AMELX.amelogenin\_XY\_SNP1)

table(cgp.C.dat$XY.11226027..AMELX.amelogenin\_XY\_SNP2)

table(cgp.C.dat$XY.11226571..AMELX.amelogenin\_XY\_SNP6)

Notice that there are only 2 valid genotypes for each SNP plus *XX* for failure:

The homozygotes are females (with two X chromosome copies);

The heterozygotes are males (with one X and one Y chromosome).

To calculate genetic gender, we look at each of the 3 SNPs trying successively to use them to assign gender. When an *XX* is detected it is coded as missing. If a mismatch between the gender assigned by any SNP is detected, we flag the data.

Start by creating a new variable called *gen\_gender* initially set to NA for all values

cgp.C.dat$gen\_gender <- NA

Use SNP 1 to define gender as female for heterozygotes and male for heterozygotes and missing otherwise.

cgp.C.dat$gen\_gender <- ifelse(cgp.C.dat[,86] == "GG", 2, ifelse(cgp.C.dat[,86] == "AG", 1, NA))

table(cgp.C.dat$gen\_gender, exclude=NULL)

# 1 2 <NA>

#1272 1199 39

Now use SNP 2 similarly:

cgp.C.dat$gen\_gender <- ifelse(is.na(cgp.C.dat$gen\_gender) & cgp.C.dat[,87] == "TT", 2, ifelse(is.na(cgp.C.dat$gen\_gender) & cgp.C.dat[,87] == "CT",1, ifelse(cgp.C.dat$gen\_gender == 2 & cgp.C.dat[,87] == "CT",NA, ifelse(cgp.C.dat$gen\_gender == 1 & cgp.C.dat[,87] == "TT",NA, cgp.C.dat$gen\_gender))))

table(cgp.C.dat$gen\_gender, exclude=NULL)

# 1 2 <NA>

#1282 1208 20

Now use SNP 3 similarly:

cgp.C.dat$gen\_gender <- ifelse(is.na(cgp.C.dat$gen\_gender) & cgp.C.dat[,88] == "CC", 2, ifelse(is.na(cgp.C.dat$gen\_gender) & cgp.C.dat[,88] == "AC",1, ifelse(cgp.C.dat$gen\_gender == 2 & cgp.C.dat[,88] == "AC",NA, ifelse(cgp.C.dat$gen\_gender == 1 & cgp.C.dat[,88] == "CC",NA, cgp.C.dat$gen\_gender))))

table(cgp.C.dat$gen\_gender, exclude=NULL)

# 1 2 <NA>

#1286 1207 17

Compare the clinical and genetic genders

table(cgp.C.dat$gen\_gender, cgp.C.dat$clin\_gender, exclude=NULL)

# 1 2 <NA>

# 1 1250 0 36

# 2 1 1174 32

# <NA> 6 11 0

NOTE: The last column shows samples without clinical gender (68) but these have all now been assigned a genetic gender (32 females and 36 males). The last row shows samples without a genetic gender; these individuals had already been assigned a clinical gender (11 females and 6 males). Notice that there is one sample where the clinical and genetic genders conflict.

Now we can clinical and genetic gender information to assign gender: we set gender according to genetic gender where clinical and genetic agree and where clinical is missing; when clinical and genetic genders disagree or when genetic gender is not available, we set the gender to missing . We call this field *curated\_gender*.

cgp.C.dat$curated\_gender <- NA

Update where both clinical and genetic agree

cgp.C.dat$curated\_gender[cgp.C.dat$clin\_gender ==1 & cgp.C.dat$gen\_gender==1] <- 1

cgp.C.dat$curated\_gender[cgp.C.dat$clin\_gender ==2 & cgp.C.dat$gen\_gender==2] <- 2

Update where clinical missing and genetic available

cgp.C.dat$curated\_gender[is.na(cgp.C.dat$clin\_gender) & cgp.C.dat$gen\_gender == 1] <-1

cgp.C.dat$curated\_gender[is.na(cgp.C.dat$clin\_gender) & cgp.C.dat$gen\_gender == 2] <-2

table(cgp.C.dat$curated\_gender, exclude=NULL)

# 1 2 <NA>

#1286 1206 18

NOTE: 17 individuals with missing genetic gender and 1 individual with contradicting clinical and genetic genders have been set to missing in *curated\_gender.*

We will use *curated\_gender* to subset the data later on.

1. Age.

Age in months is available in the field in the clinical data file named *AGEMTH*:

summary(cgp.C.dat$AGEMTH)

We can summarise the data by splitting according to a third variable, e.g. using the case control indicators assigned earlier

summary(cgp.C.dat$AGEMTH[icase]) #cases

summary(cgp.C.dat$AGEMTH[ictl]) #controls

NOTE: All controls are AGEMTH=0 (cord bloods for example)

1. Parasite Density.

Parasite density is a continuous variable available as *parasites per l* in the field in the clinical data file named *PARASITE\_DENSITY*. To look at ordered unique values in this field:

help(sort)

help(unique)

sort(unique(cgp.C.dat$PARASITE.DENSITY))

It is sometimes necessary to transform data in order to make it easier to analyse.

Transformation is usually required for parasiteamia. This is a good example of how to deal with numbers that cannot be transformed, e.g log(0)!

Here we make a logarithmic transform of all non zero parasite density data; non zero values are set to -1.

cgp.C.dat$log10\_PD <- ifelse(cgp.C.dat$PARASITE.DENSITY > 0, log10(cgp.C.dat$PARASITE.DENSITY) , -1)

Take a look at the transformed values;

sort(unique(cgp.C.dat$log10\_PD) )

Rather than treat the transformed parasite density as a continuous variable, we group the data into categories 0-1, 1-2, 2-3, and 3-4:

help(cut)

cgp.C.dat$log10\_PD\_cat <- cut(cgp.C.dat$log10\_PD,breaks = c(-1,0,1,2,3,4))

table(cgp.C.dat$log10\_PD\_cat)

#(-1,0] (0,1] (1,2] (2,3] (3,4]

# 77 163 294 452 15

* 1. Creating a final data set

1. Filter data on gender, location and ethnicity

For our example analysis, we only want to include individuals that have a valid gender, a specifically defined location and who are not of ethnicity “WAN”. Gender, location and ethnicity will be used as covariates for adjustment in association analyses. Create a final filtered quality controlled data set called *QC.dat*:

QC.dat <- cgp.C.dat[which(cgp.C.dat$ethnicity != "WAN" & cgp.C.dat$locnew != "other" & (is.na(cgp.C.dat$curated\_gender) == FALSE)), ]

dim(QC.dat) # 1751 x 112

NOTE: *QC.dat* has 1751 rows of data compared with the original data frame, which had 2510: 759 individuals comprising 392 males and 349 females with location or ethnicity *other* and 18 individuals with missing curated gender have been removed).

* 1. Section 1 - Analyses for you to try

Using the unfiltered cases or controls cgp.C.dat:

1. What proportion of unfiltered case or control samples have missing gender?
2. Calculate the proportion of individuals ged 0, (0,6] years, (6,12] years and greater than 12 years within each ethnic group.
3. As for (iii) but split by cases and controls?
4. How many samples died before leaving hospital ?

**Section 2** - **Prepare a separate genotype file for analysis**

This section reviews and recodes the genotype data to create a filtered file of numerically coded genotypes for use in association analyses.

Many R packages exist with pre-coded functions. Here we will make use of R package *genetics*. To load the functions from these packages into the work space:

library(genetics)

Manuals for these packages have been provided for your use.

1. Review SNPs.

Take a look at all the SNPs in the filtered data frame

names(QC.dat)

Autosomal SNPs are in columns 21 to 81. X chromosome SNPs are in columns 82 to 85. Gender SNPs are in columns 86 to 88.

In this analysis, we concentrate on two SNPs: *X11.5204808.rs334.HBD.hHbS*, which identifies sickle-cell trait and *X9.135121143.rs8176746.ABO.rs8176746*, which identifies blood group B versus other blood groups. We look at their distribution, association and use as covariates in analyses.

1. Find the column number of these 2 SNPs:

HbS.i1 <- which(names(QC.dat) == "X11.5204808.rs334.HBD.hHbS")

BGB.i1 <- which(names(QC.dat) == "X9.135121143.rs8176746.ABO.rs8176746")

1. Tabulate SNPs to see which genotypes are present, to check that genotype values look sensible and to identify polymorphic SNPs

table(QC.dat[HbS.i1])

# AA AT TT XX

#1555 147 10 39

table(QC.dat[BGB.i1])

# AA AC CA CC XX

# 53 490 2 1193 13

Notice that we have genotypes are coded as alphanumeric values representing alleles by base (e.g. *AA, AT, TT*,…) with *XX* for missing data. Blood Group B has two versions of the heterozygote genotype (i.e. both *AC* and *CA* genotypes are present).

R genetics package has a useful summary function:

summary(genotype(QC.dat[,HbS.i1], sep=""))

summary(genotype(QC.dat[,BGB.i1], sep=""))

1. Recode SNPs

Depending on the software and analyses required, it is often necessary to recode the genotypes. In this section we illustrate different ways to recode SNPs for ease of future analyses: to assign correct missing values or to share a common format either as genotype or counts of derived alleles. We do this here according to ancestral/derived alleles. We can also recode according to major/minor alleles but the value of the major/minor allele may vary across populations making comparison difficult. For example, consider the distribution of the Duffy blood group when comparing genotypes between Europe and Africa for example (<http://www.ncbi.nlm.nih.gov/SNP/snp_ref.cgi?rs=2814778>)

Ancestral and derived alleles can be identified from Ensembl or dbSNP. Go to www.ensembl.org then enter the SNP id into the search box and click through the screens to see that

* *rs334* has ancestral allele *T*. Here, because of the genotyping platform we actually have ancestral allele *A* and derived allele *T*.
* *rs8176746* has ancestral allele *G*. Here, because of the genotyping platform we actually have ancestral allele *C* and derived allele *A*.

Ancestral and derived allele information is available for each SNP in the *map* file in columns *anc* and *der* respectively; where ancestral or derived allele information was not available, *anc* and *der* contain the major and minor alleles respectively.

1. Remove missing values and standardize heterozygotes

Include a recode of the genotypes to reset missing value code to *NA* instead of *XX*:

We can do this easily using R gsub function

help(gsub)

For sickle:

HbS <- gsub("XX", NA, QC.dat[,HbS.i1])

table(HbS, exclude=NULL)

#AA AT TT <NA>

#1555 147 10 39

For blood group B:

BGB <- gsub("XX", NA, QC.dat[,BGB.i1])

table(BGB, exclude=NULL)

#AA AC CA CC <NA>

#53 490 2 1193 13

and to additionally change to a common format of the heterozygote genotype

BGB <- gsub("CA","AC", BGB)

table(BGB, exclude=NULL)

#AA AC CC <NA>

#53 492 1193 13

1. Alternatively, recode as *11, 12, 22* and *NA* for homozygous ancestral (wild-type), heterozygote and homozygous derived (mutant) and missing genotypes respectively. This format is common to all SNPS making them easier to work with. We can do this by simply reassigning each value:

For sickle, to set *AA* = *11*, *AT* or *TA* = *12*, *TT*=*22*, *XX*=*NA*:

HbS <- rep(NA, nrow(QC.dat))

HbS[QC.dat[,HbS.i1] == "AA"] <- "11"

HbS[QC.dat[,HbS.i1] == "AT"] <- "12"

HbS[QC.dat[,HbS.i1] == "TA"] <- "12"

HbS[QC.dat[,HbS.i1] == "TT"] <- "22"

HbS[QC.dat[,HbS.i1] == "XX"] <- NA

table(HbS, exclude=NULL)

#11 12 22 <NA>

#1555 147 10 39

For Blood Group B, to set *CC* = *11*, *AC* or *CA* = *12*, *AA*=*22*, *XX*=*NA*:

BGB <- rep(NA, nrow(QC.dat))

BGB[QC.dat[,BGB.i1 ] == "CC"] <- "11"

BGB[QC.dat[,BGB.i1 ] == "AC"] <- "12"

BGB[QC.dat[,BGB.i1 ] == "CA"] <- "12"

BGB[QC.dat[,BGB.i1 ] == "AA"] <- "22"

BGB[QC.dat[,BGB.i1 ] == "XX"] <- NA

table(BGB, exclude=NULL)

#11 12 22 <NA>

#1193 492 53 13

1. Plotting SNP Data.

We will not cover much plotting here as there are many websites showing how to use R to make plots, as well as the R help function itself.

Here is a simple example of a bar plot of the genotypes by case control status.

barplot(table(HbS, QC.dat$status), beside=T, main = "BARPLOT: HbS genotype numbers" , ylab = "frequency", xlab = "Controls = 0, Cases = 1", col= c(1:3))

Plot is shown in *MalariaGEN/plots/01-HbS\_genotype\_frequencies.pdf*

1. Testing Hardy Weinberg Equilibrium (HWE).

*HWE*.chisq function in the *genetics* package can be used to check HWE. We use the recoded format where genotype *XX* has been set to *NA,* as shown in either 2.b(i) or 2.b(ii) to ensure that missing genotypes are correctly handled.

Note the Chi-Squared statistic and associated p-value for the 2 SNPs:

summary(genotype(HbS, sep=""))

HWE.chisq(genotype(HbS, sep=""))

summary(genotype(BGB, sep=""))

HWE.chisq(genotype(BGB, sep=""))

We use the p-value from the HWE test to remove SNPs that are not in HWE. We usually use a very small threshold e.g. P<0.001 or P<0.0001 to ensure that interesting SNPs are not removed.

1. Customised Functions.

Packages do not always contain exactly the functions required so then we must write them. We have written some customised functions which are available in the R script file called custom\_functions.R. Functions in this R script file can be loaded into the workspace using *source*:

help(source)

source(paste(filepath, "custom\_functions.R", sep=""))

1. *num.recodeSNP*

This is a customized function to recode genotypes to specified values according to reference and derived alleles. Have a look at this function to understand how it works. Here we use it to recode each genotypes to the corresponding numerical count of derived alleles. The base of the reference and derived allele is obtained from the map file*, map*:

We established in 2.a(i) that the column number of QC.dat containing the sickle genotype is HbS.i1. Now, get the row number corresponding to sickle in the map file:

map.i1 <- match(names(QC.dat)[HbS.i1], map[,1])

Now run the function to create a new variable called HbS\_add containing the number of derived alleles carried by each individual. This is referred to as the additive count.

HbS\_add <- num.recodeSNP (data=QC.dat, SNPcol=HbS.i1, ref.allele=map[map.i1,]$anc, nonref.allele=map[map.i1,]$der, recode=c(0,1,2))

table(HbS\_add)

# 0 1 2

#1555 147 10

Now use the customized function and a loop to create a new data set containing case-control status, gender, ethnicity, location and genotype data (coded as numeric counts of derived allele) for each individual.

final.dat <- QC.dat[,which(names(QC.dat) %in% c('released\_key', 'status','curated\_gender', 'ethnicity', 'locnew'))]

for(i in 21:85){

map.i1 <- match(names(QC.dat)[i], map[,1])

num\_geno\_i <- num.recodeSNP(data=QC.dat, SNPcol=i, ref.allele=map[map.i1,]$anc, nonref.allele=map[map.i1,]$der, recode=c(0,1,2))

final.dat <- data.frame(final.dat, num\_geno\_i)}

names(final.dat)[-c(1:5)] <- names(QC.dat)[21:85]

Here we have first created a data frame called *final.dat* containing sample identifier, status, gender, ethnicity and location columns from the original file. Then we have looped through each of the 65 autosomal columns of SNP genotype data in *QC.dat* using our customized function each time to recode the SNP genotype data as counts of derived alleles. Each recoded column is then stored successively in *final.dat*. Finally the 65 columns of recoded data in *final.dat* are named using the same names in the original data. The derived and ancestral alleles are taken from the *map* file.

Check some of the fields

table(QC.dat$X19.6870624.rs373533.EMR1.rs373533,final.dat$X19.6870624.rs373533.EMR1.rs373533, exclude=NULL)

table(QC.dat$X6.31651080.rs361525.TNF.hTNF.238,final.dat$X6.31651080.rs361525.TNF.hTNF.238, exclude=NULL)

table(QC.dat$X1.205849512.rs17047661.CR1.rs17047661,final.dat$X1.205849512.rs17047661.CR1.rs17047661, exclude=NULL)

In each case you should only see data on the diagonal and failed samples coded as *NA*.

NOTE: *final.dat* contains a filtered set of numeric genotype data for case control samples with a unique identifier, case control status and covariates for location, ethnicity and gender.

1. Section 2 - Analyses for you to try
2. Can you make a summary and test for HWE by ethnic group?

**Section 3 - Perform association analyses.**

We use the final data frame with numerically coded genotypes representing counts of derived alleles called *final.dat*, as created in 2.e(ii) above.

Find the column numbers of SNPs for sickle cell trait and Blood Group B in the final data frame:

HbS.i2 <- which(names(final.dat) == "X11.5204808.rs334.HBD.hHbS"); HbS.i2

#44

BGB.i2 <- which(names(final.dat) == "X9.135121143.rs8176746.ABO.rs8176746"); BGB.i2

#42

1. Simple Tests of Association.

For association analysis we typically use a test that compares the frequency distribution of a SNP between cases and controls.

1. Pearson Chi-Square test.

The Pearson Chi-square test is the most basic simple association test and can be performed using function *chisq.test*

help(chisq.test)

HbS.chi <- chisq.test(table(final.dat$status, final.dat[,HbS.i2]))

Note: Ignore any warning messages but not error messages.

The Pearson Chi-Square test cannot adjust for covariates or allow for a stratified analysis.

1. Mantel-Haenszel test

The Mantel-Haenszel test using ethnicity as a stratum in an extension of the chi-squared test. It estimates odds ratios (ORs) across the levels of the stratum and the makes a summary of these ORs.

HbS.mh <- mantelhaen.test(final.dat$status,final.dat[,HbS.i2],factor(final.dat$ethnicity))

1. Generalised Linear Models

Generalized linear models naturally allow for a variety of genetic models of inheritance and for adjustment for covariates and stratification of the data. Here we need to stratify by ethnicity and place-of-origin and include covariates known to affect SNPs. For example, individuals who are heterozygous for sickle are up to 10 times less likely to develop severe illness with a malaria infection. Therefore this could mask SNP associations unless account for it. We do this by including it as a covariate in a regression model.

Commonly used genetic models of inheritance are additive, dominant, recessive and heterozygote. In a GLM, the SNP needs to be correctly coded in order to apply a specific model of inheritance.

* An additive model of inheritance assumes that each copy of the derived allele increases risk by the same amount. SNP counts 0,1,2 of derived alleles are required to run association under an additive model of inheritance.
* A dominant model of inheritance assumes heterozygous and homozygotes for the derived allele have the same risk and compares these pooled genotypes to homozygotes for the ancestral allele. SNP counts 0,1,2 of derived alleles must be recoded to 0,1,1 respectively to run association under a dominant model of inheritance
* A recessive model of inheritance assumes heterozygotes and homozygotes for the ancestral allele have the same risk and compares homozygotes for the derived allele to these pooled genotypes. Counts 0,1,2 of derived alleles must be recoded to 0,0,1 respectively to run association under a recessive model of inheritance
* A heterozygous (over-dominant) model assumes homozygous genotypes have the same risk and compares heterozygotes to the pooled homozygous genotypes. Counts 0,1,2 of derived alleles must be recoded to 0,1,0 respectively to run association under a heterozgote model of inheritance
* A genotypic model allows heterozygotes and derived homozygotes to have different risks in comparison to ancestral homozygotes. Counts 0,1,2 of derived alleles must be treated as a factor to run association under a genotypic model of inheritance.

1. Recode genotypes

We have written a simple function *model.recodeSNP* which takes a numerically coded SNP of counts of derived alleles and outputs the recoded version required for a dominant, additive or recessive model of inheritance. This can be loaded, as described earlier using the *source* function

source(paste(filepath, "custom\_functions.R", sep=""))

Run for dominant, recessive and heterozygote coding for sickle-cell trait and check it works correctly by comparing it to the additive coding from final.dat or equivalently, *HbS\_add* created in 2.e(i).

HbS\_dom <- model.recodeSNP(final.dat, "X11.5204808.rs334.HBD.hHbS", model="dom")

HbS\_rec <- model.recodeSNP(final.dat, "X11.5204808.rs334.HBD.hHbS", model="rec")

HbS\_het <- model.recodeSNP(final.dat, "X11.5204808.rs334.HBD.hHbS", model="het")

table(HbS\_add, HbS\_dom)

table(HbS\_add, HbS\_rec)

table(HbS\_add, HbS\_het)

1. Simple logistic regression using an additive model of inheritance with no stratification or adjustment for covariates

Here we simply regress SNP against the binary disease status using the numerically coded genotypes in final.dat, which represent counts of derived alleles and hence an additive mode of inheritance. The coefficient in this model is then the log odds ratio of disease associated with each copy of the derived allele. This is an additive model in terms of the log-odds ratio but is also referred to as a multiplicative model when considering the odds ratio. This is often referred to as a Cochran Trend test.

help(glm)

glm\_m1\_add <- glm(status ~ X11.5204808.rs334.HBD.hHbS, data=final.dat, family="binomial" ,na.action=na.omit)

The summary of the test provides all necessary information:

help(summary)

glm\_m1\_add\_sum <- summary(glm\_m1\_add)

glm\_m1\_add\_sum

To see what does the summary contains

names(glm\_m1\_add\_sum)

This show all the different data frames/matrices in this summary object. To view any one of these use e.g. coefficients use

glm\_m1\_add\_sum$coefficients

Estimate ORs, 95% confidence interval (CI) to 3 significant places and Wald p-values directly from the model:

help(signif)

coef <- glm\_m1\_add\_sum$coefficients

nn <- dim(coef)[1]

OR <- signif(exp(coef[nn,1]),3)

OR.lower <- signif(exp(coef[nn,1]-qnorm(0.975)\*coef[nn,2]),3)

OR.upper <- signif(exp(coef[nn,1] + qnorm(0.975)\*coef[nn,2]),3)

glm\_m1\_add\_waldp <- coef[nn,4]

glm\_m1\_add\_OR95CI <- paste(OR,"(", OR.lower, " - ",OR.upper,")",sep="")

1. Analysis of Variance (ANOVA)

To test how well the data fits the predicted model we use ANOVA to get a likelihood ratio (LR) *p*-value

help(anova)

glm\_m1\_add\_anova <- anova(glm\_m1\_add,test='Chisq')

The anova tests whether the addition of the SNP to the model results in a better fit that the null model; the smaller the *p*-value the greater the reduction in residual deviance achieved by inclusion of the SNP to the model.

nn <- length(glm\_m1\_add\_anova[,ncol(glm\_m1\_add\_anova )])

glm\_m1\_add\_LRp <- signif(glm\_m1\_add\_anova[nn, ncol(glm\_m1\_add\_anova)],3)

glm\_m1\_add\_LRp

The LR *p*-value and the Wald *p*-value are asymptotically equivalent. Estimation of the LR *p*-value requires computation of 2 models but is generally preferred.

1. Compare the results from the 4 different tests performed so far: Pearson Chi-square, Mantel Haensel and GLM and ANOVA

HbS.chi$p.value

# 9.112792e-27

HbS.mh$p.value

# 6.905762e-27

glm\_m1\_add\_OR95CI

# 0.129(0.0813 - 0.206)

glm\_m1\_add\_waldp

# 8.029405e-18

glm\_m1\_add\_LRp

# 9.67e-27

1. Model 1: Simple logistic regression using dominant, recessive and heterozygote models of inheritance without adjustment for any covariates.

We prepare data frames to include status, recoded SNP and covariates (for use later) for each mode of inheritance. We use a customized function, *model\_result*, from source code *custom\_functions.R*, to extract the wald and LR *p*-values from the regression.

* Dominant

dom\_dat <- data.frame(status=final.dat$status, ethnicity=final.dat$ethnicity, gender=final.dat$curated\_gender, SNP=HbS\_dom)

glm\_m1\_dom <- glm(status ~ SNP, data=dom\_dat, family=binomial(link=logit))

model\_result(glm\_m1\_dom)

# "OR(95%CI)=0.0956(0.0572-0.16), wald p.value=3.11158e-19, LR p.value=4.55e-30"

* Recessive

rec\_dat <- data.frame(status=final.dat$status, ethnicity=final.dat$ethnicity, gender=final.dat$curated\_gender, SNP=HbS\_rec)

glm\_m1\_rec <- glm(status ~ SNP, data=rec\_dat, family=binomial(link=logit))

model\_result(glm\_m1\_rec)

# "OR(95%CI)=0.618(0.174-2.2), wald p.value=0.457683, LR p.value=0.452637"

* Heterozygote

het\_dat <- data.frame(status=final.dat$status, ethnicity=final.dat$ethnicity, gender=final.dat$curated\_gender, SNP=HbS\_het)

glm\_m1\_het <- glm(status ~ SNP, data=het\_dat, family=binomial(link=logit))

model\_result(glm\_m1\_het)

# "OR(95%CI)=0.0767(0.043-0.137), wald p.value=3.063e-18, LR p.value=3.12409e-31"

We see that there is evidence in each model, including the additive model, for association with a protective effect of HbS on disease status. In general, we would select the lowest *p*-value, which here is the heterozygote model and here also has the smallest OR. This does not necessarily mean that this is the model by which HbS exerts it effect: that would have to be tested functionally

1. Model 2: Including covariates to take account of different ethnicities and gender

Covariates can simply be included as extra variables in the model. They can be put anywhere in the model formula. Here we add them in last place as the customized function model\_result assumes they are last. Genetic models of inheritance

* Additive

add\_dat <- data.frame(status=final.dat$status, ethnicity=final.dat$ethnicity, gender=final.dat$curated\_gender, SNP=HbS\_add)

glm\_m2\_add <- glm(status ~ ethnicity + gender + SNP, data=add\_dat, family=binomial(link=logit))

model\_result(glm\_m2\_add)

# "OR(95%CI)=0.129(0.0808-0.206), wald p.value=8.16183e-18, LR p.value=1.17815e-26"

* Dominant

glm\_m2\_dom <- glm(status ~ ethnicity + gender + SNP, data=dom\_dat, family=binomial(link=logit))

model\_result(glm\_m2\_dom)

# "OR(95%CI)=0.0951(0.0569-0.159), wald p.value=3.02815e-19, LR p.value=5.38216e-30"

* Recessive

glm\_m2\_rec <- glm(status ~ ethnicity + gender + SNP, data=rec\_dat, family=binomial(link=logit))

model\_result(glm\_m2\_rec)

# "OR(95%CI)=0.628(0.176-2.24), wald p.value=0.472483, LR p.value=0.467636"

* Heterozygote

glm\_m2\_het <- glm(status ~ ethnicity + gender + SNP, data=het\_dat, family=binomial(link=logit))

model\_result(glm\_m2\_het)

# "OR(95%CI)=0.0762(0.0427-0.136), wald p.value=2.87158e-18, LR p.value=3.52108e-31"

Note: The differences are slight between the models 2 adjusted for gender and ethnicity and models 1 where there is no adjustment.

1. Section 3 - Analyses for you to try
2. Run additive, dominant and recessive tests of association adjusted for ethnicity and gender for Blood Group B
3. Run additive models of association separately for each ethnicity.