Day 2 AM Practical – Quality Control, Testing Association, Replication and Meta-Analysis for Genome-Wide Association Studies

In the practical, you will analyse a simulated association study consisting of 1001 Cases and 1001 Controls genotyped on the Affymetrix 500k chip. We have only given you data on chromosome 22 so that you can complete the whole analysis in the time available, but the tools and commands you use are easily extendable to the analysis of whole genomes.

Software and documentation:

You will be using the software packages **PLINK 1.9**, **SNPTEST2** and **META**. These are all freely available for academic use and can be found on the web here:

https://www.well.ox.ac.uk/~gav/snptest/#introduction (SNPTEST2)
https://jmarchini.org/software/#meta (META)
https://www.cog-genomics.org/plink2 (PLINK 1.9)

Each program has detailed documentation on its webpage for you to refer to during the practical. You may also find the original PLINK documentation at http://zzz.bwh.harvard.edu/plink/ to be useful.

IMPORTANT: all of the programs write detailed screen output summarising the data being analysed and detailing the analysis being carried out. It helps to read this screen output carefully.

Data files:

The first set of data for the practical is contained in the following 4 files:

`controls.gen` – file containing genotypes on the controls for chr22
`controls.sample` – file containing information about the controls
`cases.gen` – file containing genotypes on the cases for chr22
`cases.sample` – file containing information about the cases

The `.gen` files contain the genotype data for the cases and controls. The `.sample` files contain the phenotype and covariate information on each individual.

LOOK AT THESE TEXT FILES TO UNDERSTAND THE FORMAT. THIS CAN BE DONE USING THE "more" COMMAND IN LINUX OR BY USING A TEXT EDITOR VIA THE "SHOW APPLICATIONS" ICON AT THE BOTTOM LEFT CORNER OF THE SCREEN → TEXT EDITOR

The format of the files (sometimes known as the "Oxford" format) is described in more detail on the webpage

https://www.well.ox.ac.uk/~gav/snptest/#input_file_formats

The first 11 lines and first 11 columns of the file `cases.gen` are:

```
22 rs915677 14433758 A G 0 0 1 0 0 1
22 rs9617528 14441016 C T 0 0 1 1 0 0
22 rs140378 15257135 C G 1 0 0 1 0 0
22 rs131564 15258423 C G 0 0 1 0 1 0
22 rs5748616 15268900 C G 1 0 0 0 1 0
22 rs4010554 15274264 A C 0 0 1 0 1 0
22 rs4010550 15280134 A G 1 0 0 0 1 0
22 rs4239845 15332978 C T 0 0 1 0 1 0
22 rs2890282 15333120 C T 0 0 1 0 0 1
22 rs5747360 15370420 A G 1 0 0 1 0 0
22 rs2379981 15410792 A G 0 1 0 1 0 0
```

There is one line for each SNP in the dataset. The first column contains the chromosome number. The second column is an rsid and the third column lists the SNP position in base-pairs. The fourth and fifth columns list the alleles coded A and B respectively. The next three numbers on the line should be the probabilities of the three genotypes AA, AB and BB at the SNP for the *first* individual in the cohort. The next three numbers should be the genotype probabilities for the *second* individual in the cohort. The next three numbers are for the *third* individual and so on. The order of individuals in the genotype file should match the order of the individuals in the sample file (see below).

The first few lines of the file `cases.sample` are:

```
ID_1 ID_2 missing heterozygosity pheno1 pheno2 cov1 cov2
0 0 0 C B P C D
A1001 B1001 0.000190658 0.272121 1 -0.410022 2.2682 0
A1002 B1002 0.000381316 0.300401 1 1.17131 1.83839 0
A1003 B1003 0.000190658 0.267544 1 -2.32686 1.45202 1
```

```
A1004 B1004 0.000381316 0.328247 1 2.08926 1.17157 0
A1005 B1005 0.000381316 0.295632 1 0.484458 -0.0489582 0
```

The first line lists the names of the variables in the columns. The first three names are always the same. The first two columns list individual IDs and the third column gives the overall missing data proportion of each individual. (This is not used for most analyses and so can be set to an arbitrary number if it is unknown).

The second line lists the codes for the types of variables in the columns. The first three entries are always 0. The coding scheme is

D	Discrete covariate (coded using positive integers)
C	Continuous covariates
P	Continuous Phenotype
B	Binary Phenotype (0=Controls, 1=Cases)

So, for example, the fourth column contains a continuous covariate called heterozygosity, the fifth column contains a binary phenotype called pheno1 and the sixth column contains a continuous phenotype called pheno2. There is then one line for each sample in the dataset with the values of the variables.

Getting Started

For this practical, you will need to run **Unix/Linux** commands in a Linux terminal and at the same time visualize results within **R**. So open **two** terminals and navigate to the appropriate DAY2 directory. In one of the terminals start an **R** session by typing R

Colour Key

In the rest of the practical, we provide commands that either need to be entered into R or directly into a Linux terminal

Red commands should be entered in the **R** session.
Purple is used to show the output of the **R** session/**Unix** commands.
Cyan is used for the comment in **R** (comment is a line starting with #). Comments are ignored by the **R** session if entered.
Blue commands should be entered in the Linux terminal.
Green is used to denote data files.

Entering commands

To enter the commands into the terminal you can either type them in directly, or you can open the Word/Pdf Document for the practical and then copy and paste the commands into the appropriate window. (Be careful - sometime copying and pasting does not work due to incompatibilies between **Unix** and Word/Pdf text formats).

NOTE: most of the **Unix** commands end with the '\' character. This character allows a long **Unix** command to be split over multiple lines. This way the whole command is easier to read and can be cut-and-paste into a Linux terminal window to run.

Association Test without doing any QC

(a) Using SNPTEST

First, run **SNPTEST** on the raw data files to see what the results look like without doing any QC:

```
snptest_v2.5.2 \
-data cases.gen cases.sample controls.gen controls.sample \
-frequentist 1 \
-bayesian 1 \
-method score \
-pheno pheno1 \
-o snptest_assoc.txt
```

This command runs a frequentist test for an additive effect (via the `-frequentist 1` option) and calculates a Bayes factor for an additive effect (via the `-bayesian 1` option). The phenotype is specified by the `-pheno pheno1` option. The `-method score` option tells the program to use a fast score test to implement the tests.

Make sure you read the screen output from the program. It gives information about the input dataset and analysis options that you chose.

To understand more about the commands used by **SNPTEST** either

(a) look at the **SNPTEST** webpage
(b) type `snptest_v2.5.2 -help` for a full list of all options

The **SNPTEST** command calculates a p-value and Bayes factor at each SNP and writes the results to the file `snptest_assoc.txt`. Take a look at this file. It contains one line for each SNP. The entries in each line are described by names in the header line of the file. SNP ids, base-pair position, alleles, information measures, genotype counts in cases and controls, minor allele frequencies, odds ratios and p-values are all given. See the **SNPTEST** webpage for precise details of what is included.

The results can be read into **R** using the following command:

```
s.assoc=read.table("snptest_assoc.txt", header=T, as.is=T)
```

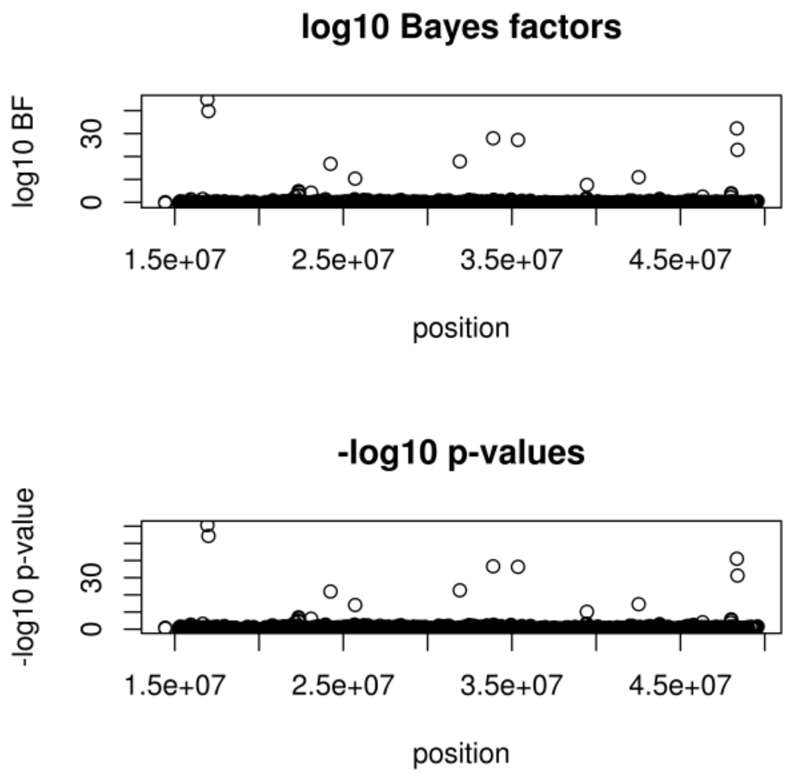
The p-values (on –log10 scale) and Bayes factors (on log10 scale) can be plotted across the chromosome using the following **R** commands:

```
#set R output format for two plots (2 rows, 1 column)
par(mfrow=c(2,1))

#plot log10 Bayes factors
plot(s.assoc$pos, s.assoc$bayesian_add_log10_bf, main="log10
Bayes factors", xlab="position", ylab="log10 BF")

#plot -log10 p-values
plot(s.assoc$pos, -log10(s.assoc$frequentist_add_pvalue),
main="-log10 p-values", xlab="position", ylab="-log10 p-value")
```

If you want to save a copy of your R plots, you can write them to a file (e.g. a .PNG file), rather than outputting them to the screen using the following commands:

```
#open .png file
png("snptest_assoc.png")

#set R output format for two plots
par(mfrow=c(2,1))

#plot log10 Bayes factors
plot(s.assoc$pos, s.assoc$bayesian_add_log10_bf, main="log10
Bayes factors", xlab="position", ylab="log10 BF")

#plot -log10 p-values
plot(s.assoc$pos, -log10(s.assoc$frequentist_add_pvalue),
main="-log10 p-values", xlab="position", ylab="-log10 p-value")
```

```
#close .png file
dev.off()
```



(b) Using PLINK

First, use **PLINK to** *convert* the Oxford format `cases.gen` and `cases.sample` files into the standard **PLINK** format:

```
plink --data cases --make-bed --allow-no-sex --out cases
```

You should now have `cases.bim,` `cases.fam` and `cases.bed` files. The file `cases.fam` has 6 columns with the following pedigree information:

1. Family ID ('FID')
2. Within-family ID ('IID'; cannot be '0')
3. Within-family ID of father ('0' if father isn't in dataset)
4. Within-family ID of mother ('0' if mother isn't in dataset)
5. Sex code ('1'=male, '2'=female, '0'=unknown)

6. Phenotype value ('1'=control, '2'=case, '-9'/'0'/non-numeric=missing data if case/control)

The file `cases.bim` has 6 columns with the following SNP information:

1. Chromosome code (either an integer, or 'X'/'Y'/'XY'/'MT'; '0' indicates unknown) or name
2. Variant (SNP) identifier
3. Position in Morgans or centiMorgans (safe to use dummy value of '0')
4. Base-pair coordinate (normally 1-based, but 0 ok; limited to 2^{31}-2)
5. Allele 1 (corresponding to clear bits in .bed; usually minor)
6. Allele 2 (corresponding to set bits in .bed; usually major)

The file `cases.bed` is a binary biallelic genotype file, which is not human readable.

Similarly, you can now convert the controls file into a **PLINK** format:

```
plink --data controls --make-bed --allow-no-sex \
--out controls
```

This generates `controls.bim, controls.fam` and `controls.bed` files.

Now, merge the two sets of **PLINK** files (one for the cases and the other for the controls) using the following **PLINK** command:

```
plink --bfile cases --bmerge controls --allow-no-sex \
--make-bed --out casecon
```

You should now have `caseson.bim, casecon.fam` and `casecon.bed` files containing the genotype data of the cases and controls.

The following command performs the association analysis in **PLINK** using logistic regression:

```
plink --bfile casecon \
--allow-no-sex --logistic beta --ci 0.95 \
--out caseconresults
```

The results are written into a file `caseconresults.assoc.logistic`

Take a look at this file. It contains one line for each SNP. The entries in each line are described by names in the header line of the file. The file contains the following columns:

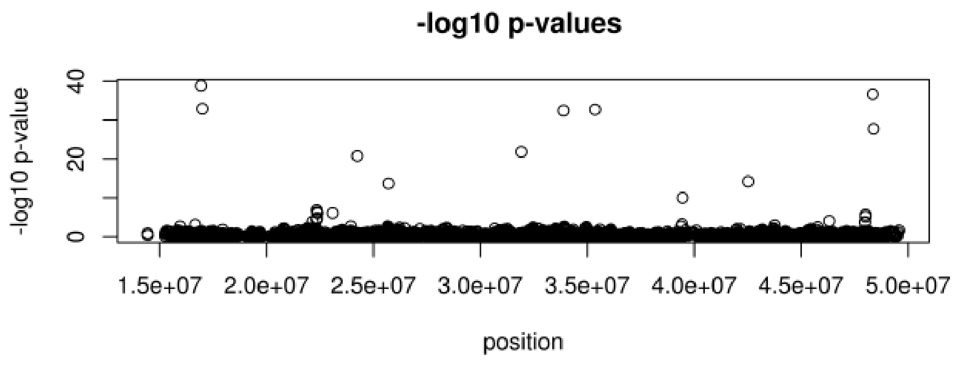
CHR	Chromosome
SNP	SNP identifier
BP	Physical position (base-pair)
A1	Tested allele (minor allele by default)
TEST	Code for the test (see below)
NMISS	Number of non-missing individuals included in analysis
BETA	Regression coefficient
SE	Standard error
L95	Lower bound on confidence interval
U95	Upper bound on confidence interval
STAT	Test statistic for coefficient (t-statistic)
P	Asymptotic p-value for t-statistic

The results can be read into **R** using the following command:

```
assoc=read.table("caseconresults.assoc.logistic", header=T,
as.is=T)
```

The p-values (on –log10 scale) can be plotted across the chromosome:

```
#open .png file
png("plink_assoc.png")

#set R output format for one plot
par(mfrow=c(1,1))

#plot -log10 p-values
plot(assoc$BP, -log10(assoc$P), main="-log10 p-values",
xlab="position", ylab="-log10 p-value")

#close .png file
dev.off()
```



The results should look very similar to that found using SNPTEST.

Quality Control

Now we will use the program **PLINK** to clean up the dataset. As you saw in the lecture, there are several different QC procedures that can be applied to a dataset. You will use **PLINK** to apply sample (person) -level filters on missing genotype call rate and heterozygosity, and to identify duplicate samples. You will then apply SNP-level filters on missing genotype call rate, allele frequency and HWE, and different genotype call rates between cases and controls. You will also examine quantile-quantile (Q-Q) plots of the p-values as a post-analysis QC check.

Sample Filters

(a) elevated missing data rates or outlying heterozygosity rate

The first step involves calculating the sample statistics (missing genotype call rate and heterozygosity) of each sample. The following command calculates the missing genotype rates:

```
plink --bfile casecon --missing \
--allow-no-sex --out casecon.mis
```

The sample statistics are written to files
`casecon.mis.lmiss`
`casecon.mis.imiss`

The *fourth* column in the file `casecon.mis.imiss` (N_MISS) gives the number of missing SNPs and the *sixth* column (F_MISS) gives the proportion of missing SNPs per individual.

The following command calculates the heterozygosity rate:

```
plink --bfile casecon --het \
--allow-no-sex --out casecon.het
```

The sample statistics are written to a file `casecon.het.het`

The *third* column in the file `casecon.het.het` gives the observed number of homozygous genotypes [O(HOM)] and the *fifth* column gives the number of non-missing genotypes [N(NM)], per individual.

To understand more about the commands used by **PLINK** either

(a) look at the **PLINK** webpage

(b) type `plink --help` for a full list of all options

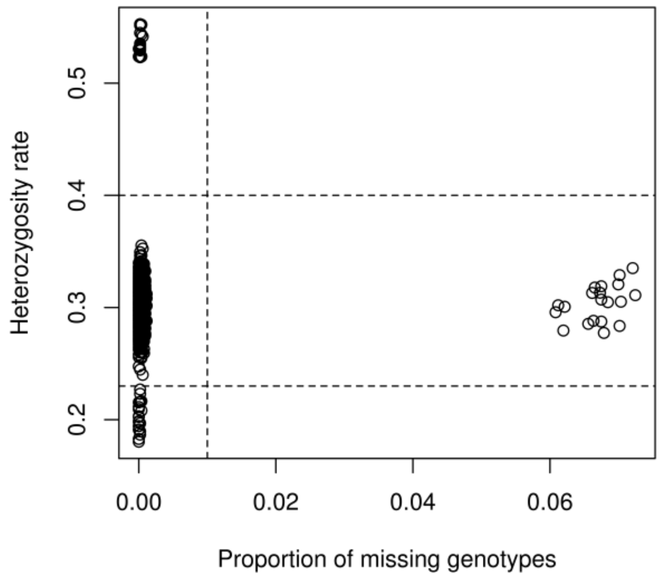
The results can be visualized from within an **R** session by plotting a graph where the observed heterozygosity rate per individual
is plotted on the x-axis and the proportion of missing SNPs per individuals is plotted on the y-axis:

```
#read in the output files
het=read.table("casecon.het.het", header=T)
mis=read.table("casecon.mis.imiss", header=T)

#Calclate the observed heterozygosity rate per individual using
#the formula (N(NM) - O(Hom))/N(NM)
mishet=data.frame(FID=het$FID, IID=het$IID,
het.rate=(het$N.NM - het$O.HOM)/het$N.NM, mis.rate=mis$F_MISS)

#Plot the proportion of missing genotypes and the
#heterozygosity rate
plot(y=mishet$het.rate, x=mishet$mis.rate, ylab="Heterozygosity
rate", xlab="Proportion of missing genotypes")
```

As you can see, there are samples with high levels of missing data and some samples with unusually high and low heterozygosity.
Based on this plot you can decide upon thresholds for removing individuals. For example, you might decide to remove all individuals with more than 1% missing data and a heterozygosity outside the range [0.23, 0.4].

Use the following R commands to visualise the thresholds and to save the R plot into a file:

```
#open .png file
png("mis_het.png")

#plot het.rate vs. mis.rate
plot(y=mishet$het.rate , x=mishet$mis.rate,
ylab="Heterozygosity rate", xlab="Proportion of missing
genotypes")
abline(v=0.01, lty=2) #vertical dashed line
abline(h= 0.23, lty=2)  #horizontal dashed line
abline(h=0.4, lty=2)  # horizontal dashed line

#close .png file
dev.off()
```



Can we figure out how many samples failed each of the filters? And what is the total number of failures? These numbers often need to be reported in papers.

To answer these questions, you can extract the individuals you decided to exclude using the following **R** commands:

```
#individuals with mis.rate > 0.1
fail_mis_qc = mishet[mishet$mis.rate > 0.01,]

#individuals with het.rate <0.23 or with  het.rate >0.4
fail_het_qc = mishet[mishet$het.rate < 0.23 | mishet$het.rate >
0.4,]
```

You can then create files `fail_mis_qc.txt` and `fail_het_qc.txt` with the individuals that do not pass the thresholds:

```
write.table(fail_mis_qc, "fail_mis_qc.txt", row.names=F,
col.names=T, quote=F)
write.table(fail_het_qc, "fail_het_qc.txt", row.names=F,
col.names=T, quote=F)
```

Take a look at these files to understand the format.

(b) duplicated or related individuals

Duplicated or related individuals can be identified using the identity-by-descent (IBD) report constructed from a reduced subset of frequent SNPs.

First, create a subset of frequent snps (MAF > 0.35) while filtering out SNPs that have > 5% missing rate and SNPs that don't pass the HWE test at p-value =0.00000001:

```
plink --bfile casecon \
--maf 0.35 --geno 0.05 --hwe 0.00000001 \
--allow-no-sex --make-bed --out frequent
```

Reduce the subset of frequent SNPs by pruning so that no pair of SNPs (within 50 base-pairs) has an r^2 greater than 0.2:

```
plink --bfile frequent --indep-pairwise 50 5 0.2 \
--allow-no-sex --out prunedsnplist
```

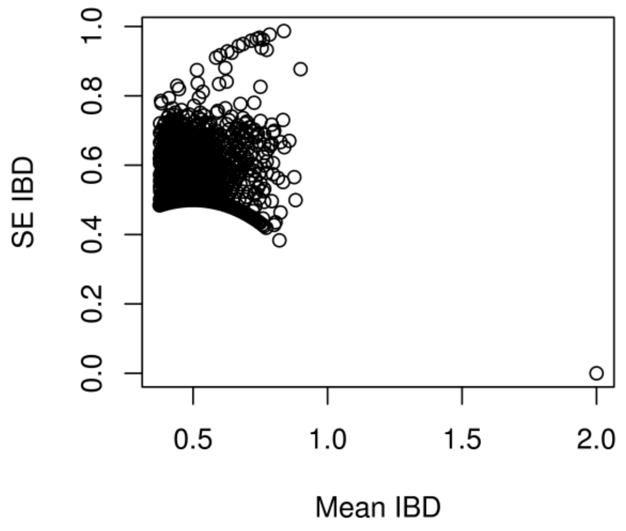
Generate the identity-by-descent (IBD) report from the reduced subset of frequent SNPs:

```
plink --bfile frequent \
--extract prunedsnplist.prune.in \
--genome --allow-no-sex \
--out caseconpruned
```

Take a look at the file `caseconpruned.genome`. The columns Z0, Z1 and Z2 contain the probability of sharing 0, 1 or 2 alleles IBD. The column PI_HAT is a mean IBD per individual, i.e.
(0*genome$Z0 + 1*genome$Z1 + 2*genome$Z2)/2
This file can be very large as it contains results for every pair of individuals. So you may prefer to use the `--Z-genome` command instead, whch outputs a gzipped version.

In **R**, you can visualise the IBD report and identify duplicated individuals by plotting the standard error of the IBD sharing vs. the mean IBD sharing:

```
#read in caseconpruned.genome file
genome=read.table("caseconpruned.genome", header=T, as.is=T)

#The caseconpruned.genome file is very big as it contains all
possible pairs of
# individuals (2,003,001 in our case). Therefore, we look only
at the pairs of
# individuals with PI_HAT > 0.1875, which corresponds to a half
way between second # and third degree relatives.

genome=genome[genome$PI_HAT > 0.1875,]

#compute Mean(IBD)
mean.ibd=0*genome$Z0 + 1*genome$Z1 + 2*genome$Z2

#compute Var(IBD)
var.ibd=((0 -mean.ibd)^2)*genome$Z0 +
          ((1 -mean.ibd)^2)*genome$Z1 +
          ((2 -mean.ibd)^2)*genome$Z2

#compute SE(IBD)
se.ibd=sqrt(var.ibd)

#open .png file
png("ibd.png")

#plot SE(IBD) vs Mean(IBD)
plot(mean.ibd, se.ibd, xlab="Mean IBD", ylab="SE IBD")

#close .png file
dev.off()
```



People who on average share two alleles IBD are either monozygotic twins or duplicates. Using the following R commands, you can see two pairs of identical individuals (genome$Z2=1, i.e. they share all their alleles IBD).

```
duplicate=genome[mean.ibd == 2,]
duplicate
```

```
           FID1  IID1  FID2  IID2 RT EZ Z0 Z1 Z2 PI_HAT PHE DST PPC RATIO
1501500 A1000 B1000 A3000 B3000 UN NA  0  0  1      1  -1   1   1    NA
2002999 A2000 B2000 A2001 B2001 UN NA  0  0  1      1   1   1   1    NA
```

Save the ID of one in each pair into a file fail_ibd_qc.txt for subsequent removal:

```
fail_ibd_qc=data.frame(FID=duplicate$FID2, IID=duplicate$IID2)

write.table(fail_ibd_qc, "fail_ibd_qc.txt", row.names=F,
col.names=T, quote=F)
```

NOTE: In practice, you might want to looks at the individual call rates stored in `casecon.mis.imiss` and output the IDs of the individual with the lowest call rate to `fail_ibd_qc.txt` for subsequent removal.

Remove all individuals failing QC

In **R**, concatenate all the files listing individuals failing the previous QC steps into a single file (with no duplicate individuals) `fail_qc.txt`:

```
fail_mis_qc=read.table("fail_mis_qc.txt",header=T,as.is=T)
fail_het_qc=read.table("fail_het_qc.txt", header=T, as.is=T)
fail_ibd_qc=read.table("fail_ibd_qc.txt", header=T, as.is=T)
fail_qc=data.frame(FID=c(fail_mis_qc$FID, fail_het_qc$FID,
fail_ibd_qc$FID), IID=c(fail_mis_qc$IID, fail_het_qc$IID,
fail_ibd_qc$IID))
fail_qc=unique(fail_qc)

write.table(fail_qc, "fail_qc.txt", row.names=F, col.names=F,
quote=F)
```

The file `fail_qc.txt` should now contain a list of unique individuals failing the previous QC steps.

Use **PLINK** to remove these from the dataset:

```
plink --bfile casecon --remove fail_qc.txt \
--allow-no-sex --make-bed --out casecon.qc.ind
```

(note: make sure to look at the screen output when these commands run. Check the number of the removed individuals.)

This command creates new files:
`casecon.qc.ind.nosex`
`casecon.qc.ind.bim`
`casecon.qc.ind.fam`
`casecon.qc.ind.bed`

The new `casecon.qc.ind` dataset contains 1939 individuals that passed the QC (you could check it by calculating the number of lines in the file with the `wc -l casecon.qc.ind.fam` command in **R**).

SNP Filters

In a similar way to the above sample filters, the first step involves calculating the SNP statistics (missing rate, allele frequency, p-value for the test of HWE and for the different call rates between cases and controls) for each SNP.

Excessive missing data rate

The following **PLINK** command computes the missing data rate:

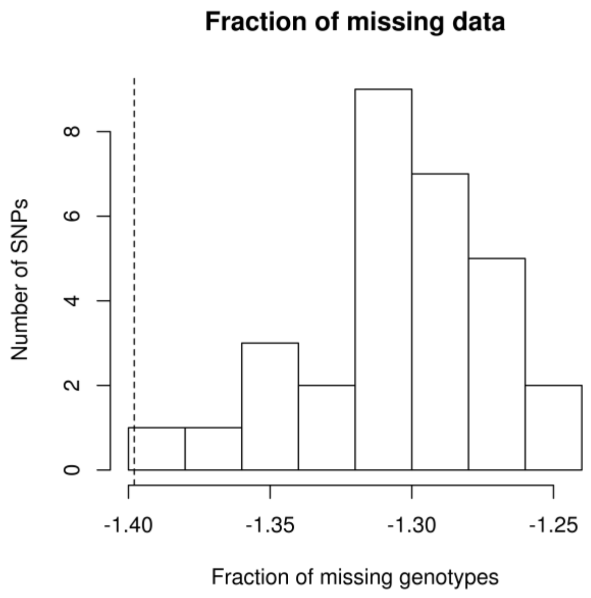
```
plink --bfile  casecon.qc.ind \
--missing --allow-no-sex \
--out casecon.qc.ind.mis
```

The column F_MISS in the file `casecon.qc.ind.mis.lmiss` contains the fraction of missing genotypes per SNP. It can be visualised in **R** by plotting a histogram (here we choose to plot it on a log scale):

```
#read in casecon.qc.ind.mis.lmiss file
lmis=read.table("casecon.qc.ind.mis.lmiss",header=T)

#plot histogram of the fraction of missing genotypes:
hist(log10(lmis$F_MISS), ylab="Number of SNPs",xlab="Fraction
of missing genotypes",main="Fraction of missing data")
```

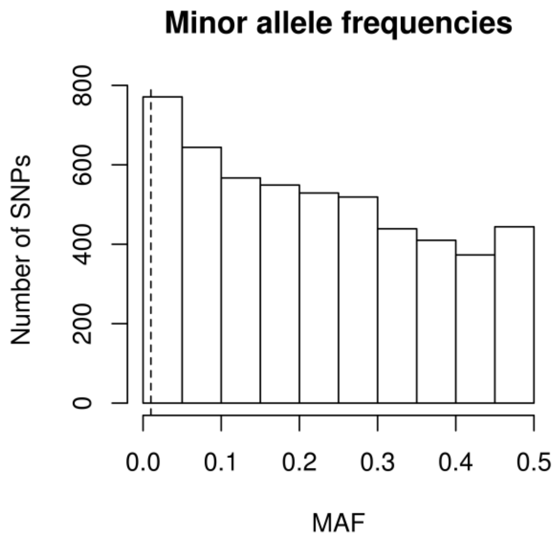
Based on this plot you can decide upon thresholds for removing SNPs, e.g. you might want to remove SNPs with more than 4% missing data:

```
#open .png file
png("casecon_qcind_lmiss.png")

#set output format
par(mfrow=c(1,1))

#plot histogram of log10(lmis$F_MISS)
hist(log10(lmis$F_MISS), ylab="Number of SNPs",xlab="Fraction
of missing genotypes",main="Fraction of missing data")

#dashed vertical line corresponding to 4% missing data
abline(v=log10(0.04),lty=2)

#close file
dev.off()
```



Next, you can examine the frequency of the minor alleles by computing them in **PLINK** and visualising in R**:**

```
plink --bfile casecon.qc.ind --freq  \
--allow-no-sex --out casecon.qc.ind.freq

#read in frequencies file
freq=read.table("casecon.qc.ind.freq.frq", header=T)

#open .png file
png("casecon_qcind_freq.png", res=1200, width=4, height=4,
units="in")

#set output format
par(mfrow=c(1,1))
#plot histogram of frequencies
hist(freq$MAF, ylab="Number of SNPs",xlab="MAF",main="Minor
allele frequencies")

#dashed vertical line corresponding to 1% MAF
abline(v=0.01,lty=2)

#close .png file
dev.off()
```



Different genotype call rates between cases and controls

The following PLINK command performs a test for differences between cases/controls missing call rates at each variant and outputs the results into a file `casecon.qc.ind.call.rate.missing`

```
plink --bfile casecon.qc.ind \
--test-missing --allow-no-sex  \
--out casecon.qc.ind.call.rate
```

Next, in R, you create a file `fail_diffmiss_qc.txt` that contains SNPs with different call rates between cases and controls at p-value < 0.000001 for subsequent removal:

```
#read in casecon.qc.ind.call.rate.missing file
diffmiss = read.table("casecon.qc.ind.call.rate.missing",
header=T, as.is=T)
#save SNPs with p-value < 0.000001 into a file
diffmiss = diffmiss[diffmiss$P<0.000001,]
write.table(diffmiss$SNP, "fail_diffmiss_qc.txt", row.names=F,
col.names=F, quote=F)
```

Remove all SNPs failing QC

The following **PLINK** command removes the SNPs listed in the file `fail_diffmiss_qc.txt`, and also SNPs with MAF<0.01, SNPs with > 4% missing genotypes, and SNPs that depart from the HWE at p-value<0.000001 (in controls):

```
plink --bfile casecon.qc.ind  \
--exclude fail_diffmiss_qc.txt  \
--maf 0.01 --hwe 1e-6  --geno 0.04  \
--make-bed --allow-no-sex --out casecon.qc
```

The new dataset `casecon.qc` contains 5016 SNPs that passed the QC.

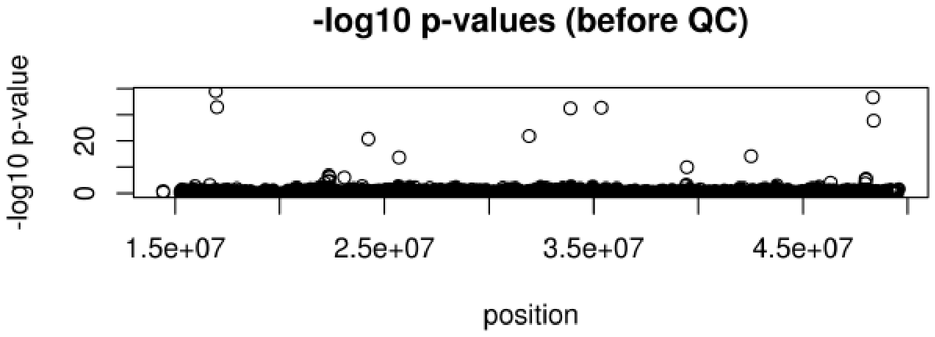
Testing for association after QC

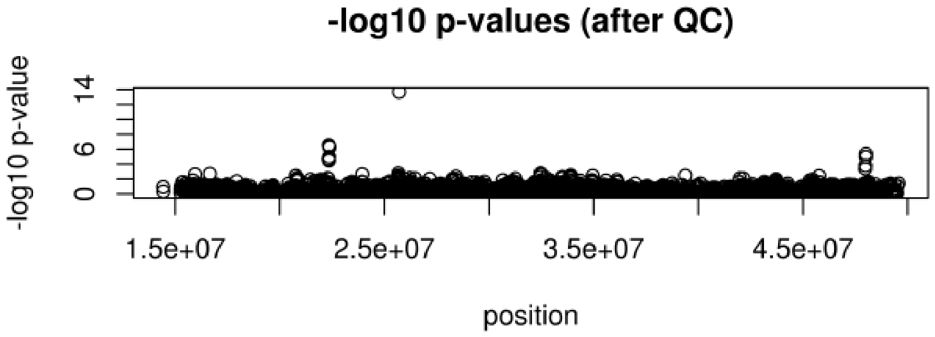
Now the QC filtering has been carried out, you can re-test for association to see if it has made any difference to the results using **PLINK**:

```
plink --bfile casecon.qc --allow-no-sex \
--logistic beta --ci 0.95 --out casecon.qc.log
```

The results can be visualized in **R** using the following commands:

```
#read in the association test results (before QC)
assoc=read.table("caseconresults.assoc.logistic", header=T,
as.is=T)
#read in the association test results (after QC)
assoc.qc=read.table("casecon.qc.log.assoc.logistic", header=T,
as.is=T)
#open .png file
png("casecon_before_afer_qc_assoc.png")
#set the output format
par(mfrow=c(2,1))
#plot -log10 p-values (before QC)
plot(assoc$BP,-log10(assoc$P), main="-log10 p-values (before
QC)", xlab="position", ylab="-log10 p-value")
#plot -log10 p-values (after QC)
plot(assoc.qc$BP,-log10(assoc.qc$P), main="-log10 p-values
(after QC)", xlab="position", ylab="-log10 p-value")
#close .png file
dev.off()
```



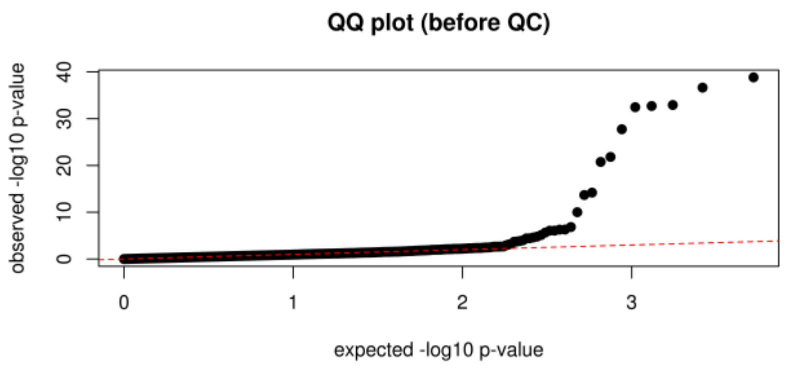


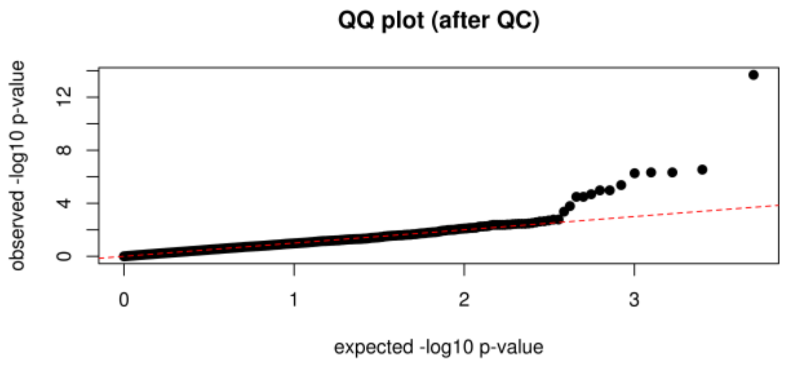
It seems as if most of the highly significant results have disappeared after QC. There is one lonely (possibly slightly suspicious?) signal with a p-value around 1e-14 and a few better-supported signals with p-values around 1e-06 or 1e-07.

Q-Q plots of p-values

You can examine the quantile-quantile (Q-Q) plot of the p-values before and after the QC by plotting the observed p-values vs. the expected p-values:

```
#open .png file
png("QQplot_before_afer_qc.png")

#set the output format
par(mfrow=c(2,1))

#observed -log10 p-values (before QC)
p.obs=-log10(sort(assoc$P,decreasing=F))

#expected -log10 p-values (before QC)
p.exp=-log10( 1:length(p.obs)/length(p.obs) )

#plot observed vs. expected -log10 p-values (before QC)
plot(p.exp, p.obs, pch=19, main= "QQ plot (before QC)",
xlab="expected -log10 p-value",
ylab="observed -log10 p-value")

#equality line
abline(0, 1, lty=2, col="red")

#observed -log10 p-values (after QC)
p.obs=-log10(sort(assoc.qc$P,decreasing=F))

#expected -log10 p-values (after QC)
p.exp=-log10( 1:length(p.obs)/length(p.obs) )

#plot observed vs. expected -log10 p-values (after QC)
plot(p.exp, p.obs, pch=19, main= "QQ plot (after QC)",
xlab="expected -log10 p-value",
ylab="observed -log10 p-value")

#equality line
abline(0, 1, lty=2, col="red")

#close .png file
dev.off()
```





Identifying regions of association

Now the dataset has been cleaned you can identify a set of SNPs for follow-up and replication. To do this you will need to pick a threshold for the p-values. For example, SNPs with a -log10 (p-value) > 4 can be identified by typing in **R**:

```
assoc.qc[-log10(assoc.qc$P)>4,]
```

The command will produce the following output:

```
      CHR        SNP       BP A1 TEST NMISS    BETA      SE     L95     U95
774    22  rs5751704 22331975  A  ADD  1939  0.2853 0.06700  0.1539  0.4166
776    22 rs12157657 22344229  T  ADD  1939  0.3990 0.07774  0.2466  0.5513
777    22   rs738785 22352631  C  ADD  1939  0.2783 0.06695  0.1471  0.4095
778    22  rs2330578 22352975  A  ADD  1939  0.2783 0.06695  0.1471  0.4095
779    22 rs11090280 22358412  T  ADD  1939  0.2969 0.06737  0.1649  0.4290
781    22  rs7287369 22362015  A  ADD  1939  0.3932 0.07804  0.2403  0.5462
782    22 rs13057362 22362771  A  ADD  1939  0.3932 0.07804  0.2403  0.5462
785    22 rs17629956 22380471  A  ADD  1939  0.3919 0.07820  0.2387  0.5452
1372   22  rs5761885 25700273  T  ADD  1939 -0.7937 0.10380 -0.9971 -0.5903
4809   22  rs2858612 48009185  C  ADD  1939 -0.3148 0.07145 -0.4548 -0.1747
4810   22  rs2858613 48011290  T  ADD  1939 -0.3182 0.06916 -0.4538 -0.1827
```

```
       STAT          P
774    4.257 2.070e-05
776    5.132 2.866e-07
777    4.157 3.228e-05
778    4.157 3.228e-05
779    4.407 1.047e-05
781    5.039 4.675e-07
782    5.039 4.675e-07
785    5.012 5.394e-07
1372 -7.647 2.050e-14
4809 -4.406 1.055e-05
4810 -4.601 4.198e-06
```

Replication

There is another case control dataset (with QC already applied) available to test for replication. The files are:

```
rep.bim, rep.bed, rep.fam
```

Perform logistic regression with **PLINK** on this new dataset using the command

```
plink --bfile rep --allow-no-sex  \
--logistic beta --ci 0.95 --out rep.log
```

Read the results into **R**:

```
assoc.rep=read.table("rep.log.assoc.logistic", header=T,
as.is=T)
```

and check the SNPs you identified for evidence of replication. For example, if you want to check SNP rs5751704 use the command

```
assoc.rep[assoc.rep$SNP == "rs5751704",]
```

```
        CHR        SNP        BP A1 TEST NMISS   BETA      SE    L95    U95  STAT
827   22 rs5751704 22331975  A  ADD  2000 0.2864 0.06395 0.1611 0.4118 4.479
            P
827 7.501e-06
```

To check the slightly suspicious signal, use the command

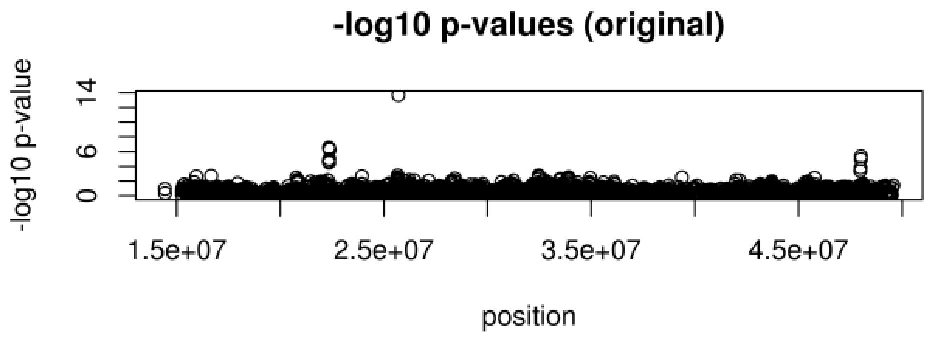
```
assoc.rep[assoc.rep$SNP == "rs5761885",]
```

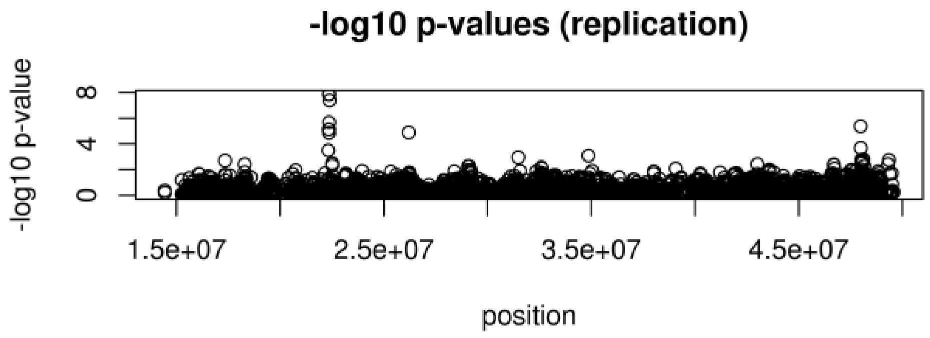
You should find little evidence of replication (p-value =0.2514) at this SNP, suggesting the original result may have been spurious.

Meta-analysis

Let us now combine the original dataset together with the replication dataset in a meta-analysis. First, look at the signal in the replication dataset. How does it compare to the signal from the original dataset? We can answer this question using the **R** commands below:

```
#read the association test results (original data set)
assoc.qc=read.table("casecon.qc.log.assoc.logistic", header=T,
as.is=T)
#read the association test results (replication data set)
assoc.rep=read.table("rep.log.assoc.logistic", header=T,
as.is=T)

#open .png file
png("casecon_with_rep.png")
#set output format for the plots
par(mfrow=c(2,1))
#plot -log10 p-values for the original data set
plot(assoc.qc$BP,-log10(assoc.qc$P), main="-log10 p-values
(original)", xlab="position", ylab="-log10 p-value")
#plot -log10 p-values for the replication data set
plot(assoc.rep$BP,-log10(assoc.rep$P), main="-log10 p-values
(replication)", xlab="position", ylab="-log10 p-value")
#close .png file
dev.off()
```





Visually it seems as if there may be signals in the same genetic locations in both the original and the replication data sets. However, we already know that the slightly suspicious SNP did not replicate, so the top SNP seen in that region must be a different SNP. We can check this in **R** by typing:

```
assoc.rep[assoc.rep$P<0.0001,]
```

It seems that the top replication SNP in that region is rs5752554 at position 26192808 (with p-value 1.318e-05), while the top original SNP was rs5761885 at position 25700273 (with p-value 2.050e-14).

You can combine the results from the two datasets using the **META** program. First, you need to prepare input files for the META program (one file for each data set). Here is the required structure of the input file (see https://mathgen.stats.ox.ac.uk/genetics_software/meta/meta.html):

rsid	SNP id.
pos	base-pair position of SNP.
allele_A	non-coded allele (a.k.a non-effect allele, non-reference allele).
allele_B	coded allele (a.k.a effect allele, reference allele).

`info`	imputation quality score (**RSQR_HAT** column in **MACH**; **INFO** column in **PLINK**; **PROPER_INFO** column in **SNPTEST**).
`P_value`	p-value of each SNP.
`beta`	effect size of each snp.
`se`	standard error of effect size.

The values for the columns `rsid, pos, allele_B`, and `P_value` can be taken from the association test output files (you have already read them into `assoc.qc` and `assoc.rep` **R** objects).
The value for the column `allele_A` (non-effect allele) can be taken from the *sixth* column of the files `casecon.qc.bim` /`rep.bim`:

```
#read casecon.qc.bim and rep.bim files
qc.bim=read.table("casecon.qc.bim", as.is=T)
rep.bim=read.table("rep.bim", as.is=T)

#create vectors A2 and rep.A2 with the non-effect allele
A2=qc.bim$V6
rep.A2=rep.bim$V6
```

You can now prepare the META input file for the original data set:

```
#META input file for the original dataset
casecon.meta=data.frame(chr=assoc.qc$CHR, rsid=assoc.qc$SNP,
pos=assoc.qc$BP, allele_A=A2,
allele_B=assoc.qc$A1,P_value=assoc.qc$P, info=rep(1, times=
nrow(assoc.qc)), beta=assoc.qc$BETA, se=assoc.qc$SE)

write.table(casecon.meta, "casecon_meta.txt", row.names=F,
col.names=T, quote=F)
```

Similarly, you create an input file for the replication data set:

```
#META input file for the replication dataset
rep.meta=data.frame(chr=assoc.rep$CHR,rsid=assoc.rep$SNP,
pos=assoc.rep$BP, allele_A=rep.A2, allele_B=assoc.rep$A1,
P_value=assoc.rep$P, info=rep(1, times= nrow(assoc.rep)),
beta=assoc.rep$BETA, se=assoc.rep$SE)

write.table(rep.meta, "rep_meta.txt", row.names=F, col.names=T,
quote=F)
```

NOTE: We set the imputation quality score (info) to 1, as if “pretending” that these genotyped SNPs are perfectly imputed SNPs.

The following command runs the **meta analysis**:

```
meta_v1.7 \
--cohort casecon_meta.txt  rep_meta.txt \
--method 1 --output meta.txt
```

and produces the output on the screen:

```
META v1.7
=========

==> Standard output used
==> Method being used: inverse-variance method (fixed effects
model)
==> OUTPUT: meta.txt

Reading the data ...
==> 2 cohorts being used:
Cohort 1: casecon_meta.txt, 5016 out of 5016 (100%) SNPs are
used (threshold >= 0.5)
Cohort 2: rep_meta.txt, 5072 out of 5072 (100%) SNPs are used
(threshold >= 0.5)

There are 5147 SNPs in the union list.
0 SNPs have had their alleles flipped to make strand consistent
205 SNPs have had the order of their alleles normalized
0 SNPs have been removed as their alleles are not consistent
across cohorts

Writing the data ...

DONE!
```

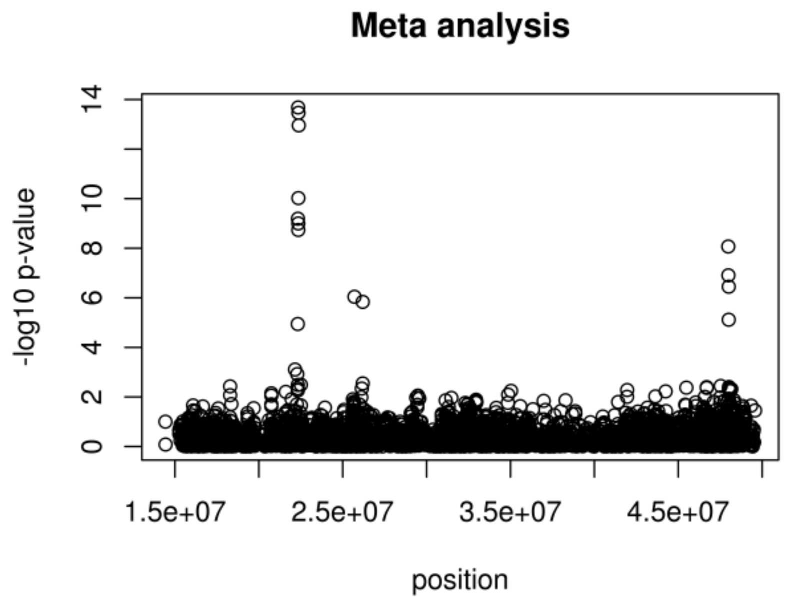
NOTE: For 205 SNPs, the effect and the non-effect alleles were different across the two cohorts, e.g. use the **Unix** grep command to have a look at SNP rs4400:

```
grep rs4400 casecon.qc.bim
22      rs4400  0       47091268        G       A
grep rs4400 rep.bim
22      rs4400  0       47091268        A       G
```

META takes this into account by changing the coefficients of the logistic regression accordingly.

You can read the results into **R** and make a plot:

```
#read in the results file for the meta analysis
meta.res=read.table("meta.txt", header=T, as.is=T)
#open .png file
png("meta.png")
#set the output format
par(mfrow=c(1,1))
#plot -log10 p-value for the meta analysis
plot(meta.res$pos, -log10(meta.res$P_value), ylab="-log10 p-
value", xlab="position", main="Meta analysis")
#close .png file
dev.off()
```



We can investigate the top SNPs in R by typing:

```
meta.res[meta.res$P_value<0.0001,]
```

In relation to the second signal, it seems that there is significant heterogeneity in the results from the two cohorts at rs5761885 (P_heterogeneity=2.33538e-09), while there is more consistency at rs5752554, although the results are certainly stronger in the replication cohort (p-value=1.318e-05) than in the original cohort (p-value=0.02585).

The signal could be visualised with the following **R** commands:

```
plot(meta.res$pos, -log10(meta.res$P_value), ylab="-log10 p-
value", xlab="position", main="Meta analysis")

#colour rs5761885 in green
points(meta.res$pos[meta.res$rsid == "rs5761885"], -
log10(meta.res$P_value[meta.res$rsid == "rs5761885"]), col =
"green", pch = 20)

#colour rs5752554 in blue
points(meta.res$pos[meta.res$rsid == "rs5752554"], -
log10(meta.res$P_value[meta.res$rsid == "rs5752554"]), col =
"blue", pch = 20)
```