Knowledge Book

Sonia García-Ruiz (s.ruiz@ucl.ac.uk)

2022-01-20

Contents

Chapter 1

Analysis of Genetic Association Studies

All information shown in this chapter has been extracted from the course "Analysis of Genetic Association Studies", taught at University of Liverpool, Department of Health Data Science.

The purpose of this section is to provide guidance on performing a GWAS analysis step-by step, as well as introducing to useful software for undertaking a GWAS.

1.1 Useful Linux commands

First, I'll show you some useful Linux commands for working with GWASs files. Linux is an operating system very well suited to dealing with large datasets.

Create a bash script:

```
touch script.sh
```

Run the script:

```
sh script.sh
```

Return the number of lines of a file:

```
wc -l ./file.map
```

Print only the first 15 lines of "./file.map" file:

```
head -15 ./file.map
```

Print the bottom of the file:

```
tail -15 ./file.map
```

Find the SNP "rs1234":

```
grep "rs1234" ./file.map
```

Find the SNP "rs1234" and print the line number of the match:

```
grep -n "rs1234" ./file.map
```

Search the file ./file.map, for "1234" in the SNP name (column 2, specified by $2) and print the whole line:

```
awk '$2~/1234/{print}' ./file.map
```

Search the file ./file.map, for "1234" in the SNP position (column 4, specified by $4) and print only the SNP name (column 2):

```
awk '$4~/1234/{print $2}' ./file.map
```

Search for an exact match of "rs1234676" in the SNP name (column 2) and print the whole line:

```
awk '$2=="rs1234676"/{print}' ./file.map
```

Search for all SNPs with a position between 90680000 up to and including 90690000 in the SNP position column (column 4) and count the lines in that output:

```
awk '$4>90680000&&$4<=90690000{print}' ./file.map | wc -l
```

Remove the chromosome 10 from the output above:

```
awk '$4>90680000&&$4<=90690000{print}' ./file.map | awk '$1!=10{print}'
```

Count the SNPs from chromosome 10 and chromosome 23:

```
awk '$1==10{print}' ./file.map| wc -l
awk '$1==23{print}' ./file.map| wc -l
```

Print the SNP position of the first SNP on chromosome 23:

```
awk '$1==23{print $4}' ./file.map | head -1
```

There are SNPs starting with other characters than "rs", identify what other starting characters there are.

```
awk '$2!~/rs/{print $2}' ./file.map
awk '$2!~/rs/&&$2!~/.../{print $2}' ./file.map
```

1.2 Useful R commands

NOTE: R has a system of libraries on the Comprehensive R Archive Network (CRAN) where R users deposit code to be verified and then distributed for wider use (https://cran.r-project.org/).

Load the file "./file.map" into R.

```
gd <- read.table("./file.map",header=F)
names(gd)<-c("CHR","SNP","cM","BP")
head(gd)
tail(gd)
```

Search for only rs1234568 in the SNP name (column 2):

```
gd[gd$SNP=="rs1234568",]
```

Search for all SNPs with a position between 90680000 up to and including 90690000 in the SNP position column:

```
gd[gd$BP>90680000&gd$BP<=90690000,]
```

Do you have any genome-wide significant (p=5x10-8) SNPs?

```
gd[gd$P<0.00000005,]
```

Report results using a Manhattan plot:

```
plot(gd$BP,-log(gd$P,base=10))
```

There were SNPs from both chromosome 10 and 23 in the .map file. Separate chromosomes by colour:

```
plot(gd$BP,-log(gd$P,base=10), col=gd$CHR)
```

Add a reference line to know which SNPs have a p-value below 1x10-5:

```
plot(gd$BP,-log(gd$P,base=10), col=gd$CHR)
abline(h=-log(1e-5,base=10),col="blue")
```

Use the package "qqman" to generate a Manhattan plot:

```
manhattan(gd)
manhattan(gd[gd$chromosome==10,], chr="chromosome",bp="position",snp="rsid",p="pvalue")
manhattan(gd[gd$chromosome==23,], chr="chromosome",bp="position",snp="rsid",p="pvalue")
```

1.3 Variant Calling from sequencing data

1.3.1 Generate genotype calls for a single sample

In this section we will generate genotype calls from aligned sequencing data, explore the calls and perform some initial quality checks.

A commonly used variant caller comes from the Genome Analysis Toolkit (GATK).

- GATK takes files containing aligned sequencing reads. These are called .bam files.
- GATK produces .vcf files containing details of the variants, to explore and manipulate these files, we'll be using vcftools. This is also freely available and can be found: http://vcftools.sourceforge.net/
- To visualise the variant calls we'll be using Integrated Genome Viewer (IGV); this is available from: http://software.broadinstitute.org/software/igv/

Usually, each .bam file has an accompanying .bai file. These are index files, which are programs used to navigate the data faster – they must be present for software programs to run.

Generate genotype calls for the sample NA12878 using GATK:

```
gatk -T HaplotypeCaller \
-R human_g1k_b37_20.fasta \
-I NA12878_wgs_20.bam \
-o yourfilename.vcf \
-L 20:10,000,000-10,200,000
```

Variant calling on a whole genome can take a long time, we're only looking at a very small region: Chromosome 20, positions 10,000,000 – 10,200,000, specified by the –L option.

(*) Details of the different options used: https://software.broadinstitute.org/gatk/documentation/tooldocs/current/org_broadinstitute_gatk_tools_walkers_haplotypecaller_HaplotypeCaller.php

To have a look at the first 5 variant calls using:

```
head -34 yourfilename.vcf | tail -6
```

Filter the vcf file and see only the variants called in the region 20:10,002,371-10,002,546:

```
vcftools --vcf yourfilename.vcf \
--chr 20 \
--from-bp 10002371 \
--to-bp 10002546 \
```

```
--out yournewfilename \
--recode
```

The code above will generate the files: yournewfilename.recode.vcf and yournewfilename.log.

To know how many variants were identified in this region:

```
cat yournewfilename.log
```

What are those variants?

```
cat yournewfilename.recode.vcf
```

1.3.2 Generating genotypes from multiple .bam files

To generate genotypes for multiple samples, one option is to run the same command than above but give multiple -I options:

```
gatk -T HaplotypeCaller \
-R human_g1k_b37_20.fasta \
-I NA12878_wgs_20.bam \
-I NA12882_wgs_20.bam \
-I NA12877_wgs_20.bam \
-o differentfilename.vcf \
-L 20:10,000,000-10,200,000
```

Another option is to run each sample independently, but generate a .g.vcf file and then merge the .g.vcf files to get a single call set.

Generate a .g.vcf file for sample NA12878 (then run the same command on the bam files for samples NA12877 and NA12882):

```
gatk -T HaplotypeCaller \
-R human_g1k_b37_20.fasta \
-I NA12878_wgs_20.bam \
-o yourchoice_NA12878.g.vcf \
-ERC GVCF \
-L 20:10,000,000-10,200,000
```

The main difference between the .g.vcf and the .vcf file is that the gvcf file contains a line per base or region if the region contains no variants. The gvcf also includes as a symbolic allele, this allows for complex variants at the site and to makes it possible to give a confidence measure in the reference allele (more info: http://gatkforums.broadinstitute.org/gatk/discussion/4017/what-is-a-gvcf-and-how-is-it-differentfrom-a-regular-vcf)

Merge the individual .g.vcf files and get a .vcf file containing the calls for all samples:

```
gatk -T GenotypeGVCFs \
-R human_g1k_b37_20.fasta \
-V yourchoice_NA12878.g.vcf \
-V yourchoice_NA12882.g.vcf \
-V yourchoice_NA12877.g.vcf \
-o yourchoice_gvcf_jointcalls.vcf \
-L 20:10,000,000-10,200,000
```

1.3.3 Performing initial QC

GWAS studies often focus only on SNPS, whereas variant callers identify more variants, such as indels. To create a file containing just SNPs (no indels or other variants) the –remove-indels option from vcftools can be used:

```
vcftools --vcf yourchoice_gvcf_jointcalls.vcf \
--remove-indels \
--out yourchoice_snps \
--recode
```

Calculate the Ts/Tv ratio:

```
vcftools --vcf yourchoice_gvcf_jointcalls.vcf \
--TsTv-summary \
--out yourchoice_tstv
```

The QualByDepth (QD) score is the variant quality (QUAL field) divided by the unfiltered depth of nonreference samples. Variants with a low QD may be unreliable and have a high false positive rate. To add a filter to the .vcf file which flags variants which have a quality-by-depth less than 2.0:

```
gatk -T VariantFiltration \
-R human_g1k_b37_20.fasta \
-V yourchoice_gvcf_jointcalls.vcf \
--filterExpression 'QD<2.0' \
--filterName 'QualityDepth' \
-o yourchoice_flagged.vcf
```

NOTE: SNPs and Indels may need to be treated differently – to do this, the file can be first splitted into 2 vcfs, one containing SNPs and one containing indels, then add the flags should be added.

The new .vcf is now different from the original as the FILTER column is now filled in, usually with PASS.

Remove all SNPs with a filter flag other than PASS:

```
vcftools --vcf yourchoice_flagged.vcf \
--remove-filtered-all \
```

```
--recode \
--out yourchoice_filtered
```

To convert the SNP filtered .vcf file to PLINK .ped and .map files:

```
vcftools --vcf yourchoice_gvcf_jointcalls.vcf \
--remove-indels \
--out yourchoice_plink
```

1.4 Quality Control for GWAS

1.4.1 Software and datasets

A commonly used program for GWAS analysis is called 'PLINK', which is freely available for download. The webpage for PLINK is available at: http://pngu.mgh.harvard.edu/~purcell/plink/ For undertaking a GWAS, two input files are required: one '.ped' file and one '.map' file.

- The PED file is a white-space (space or tab) delimited file. It includes one row for each participant within the study, and at least seven columns. Each of the SNPs genotyped is represented by two columns (column 7 onwards) –one for each of the two alleles held by the individual at the SNP. Typical coding would be A, C, G or T to represent the four possible alleles, or similarly 1, 2, 3 or 4. Coding for missing genotype is '0 0'.
- The .map file includes information on the SNPs genotyped, and each row represents a SNP. It includes 4 columns.

In this example, we will be working on analysing the GWAS dataset 'genstudy' (a .map and .ped file for 'genstudy'). The study is a case-control study with 252 diseased (cases) and 252 non-diseased (controls) individuals. All individuals have been genotyped for 198,684 SNPs on chromosome 10 and 22,379 SNPs on chromosome X. The purpose of the study is to try and identify SNPs on chromosome 10 associated with disease.

To view the first row and first eight columns of the 'genstudy.ped' file:

```
awk 'FNR == 1 { print $1,$2,$3,$4,$5,$6,$7,$8} ' genstudy.ped
```

The eight columns correspond to the first six mandatory columns plus two columns representing the first SNP genotyped.

1.4.2 Checking the input datasets

Before starting a genotype QC or GWAS association analysis, it is important to check that the input files are in good order, and contain the individuals

and SNPs that we think they should. Do this by focussing on the SNPs on chromosome 10:

```
plink --file genstudy --chr 10 --noweb
```

The output will also be saved in a log file called 'plink.log'.

Note: the '-noweb' option is included to make PLINK run faster – otherwise it connects to the web to check for updates before running the analysis.

1.4.3 Binary PED files

To save space and time, .map and .ped files can be converted to binary format. In doing this, we create three new files as follows:

- '.bed': file This includes all the genotype data from the previous '.ped' file, but in binary format.
- '.fam': file This includes the first six columns from the previous '.ped' file.
- '.bim' file This is the same as the previous '.map' file, but also includes an additional two columns with the alleles names (which are otherwise lost when'converting the genotype data from .ped file to .bed file.)

To create a set of binary format files for the 'genstudy' dataset:

```
plink --file genstudy --make-bed --out genstudy --noweb
```

To use the binary format files instead of the .ped and .map files, we just substitute the option –file with –bfile in the PLINK command line:

```
plink --bfile genstudy --chr 10 --noweb
```

1.4.4 Sample QC

In this section, we will start undertaking genotype QC on the 'genstudy' dataset – starting first of all with sample QC, one step at a time.

1.4.4.1 Checks for missing data

To obtain a summary of the amount of missing genotype data per sample and per SNP, use the option '- -missing' in the PLINK command line.

```
plink --bfile genstudy --missing --out genstudy.CRstats --noweb
```

Adding the '–out genstudy.CRstats' option ensures that the two output files are called 'genstudy.CRstats.imiss' and 'genstudy.CRstats.lmiss' respectively.

This command creates two output files:

- a '.imiss' file: summarises the proportion of missing genotype data per individual (one row per individual and six columns). The most important column for sample QC purposes is column 6, which contains the proportion of missing genotypes for the individual.
- a '.lmiss' file: summarises the proportion of missing genotype data per SNP (one row per SNP and five columns). The most important column for SNP QC purposes is column 5, which contains the proportion of missing genotypes for the SNP.

1.4.4.2 Gender Checks

PLINK allows to use genotype data from the X chromosome to determine gender (i.e. based on heterozygosity rates), and compares this to reported gender as per the .ped/.fam file. PLINK then flags any discrepancies i.e. individuals for whom reported gender does not match the gender estimated based on genotype data. To undertake this comparison, we can use the option '- - check-sex' in PLINK.

To undertake the gender comparison and save results in an output file called 'genstudy.sexcheck':

```
plink --bfile genstudy --check-sex --out genstudy --noweb
```

The output file has six columns, being the most important one (for the purpose of sample QC) column 5, which indicates whether there is a gender discordance or not.

1.4.4.3 Duplicate/related samples

PLINK will also calculated identity-by-state (IBS) and identity-by-descent (IBD) statistics for our individuals to help us identify duplicated and/or related samples. Before we ask PLINK to do this, however, we need to generate a pruned subset of SNPs using the option '–indep-pairwise' in PLINK, which is based on pairwise genotypic correlation. An example of a command line for generating a pruned subset of SNPs is as follows:

```
plink --bfile genstudy --indep-pairwise 1500 150 0.2 --noweb
```

The options 1500, 150 and 0.2 in the command above relate to the following steps, and of course these options can be changed depending on your requirements:

- consider a window of 1500 SNPs,
- calculate LD between each pair of SNPs in the window, remove one of a pair of SNPs if the LD is greater than 0.2,
- shift the window 150 SNPs forward and repeat the procedure

Running this command line will create two output files, 'plink.prune.in' and 'plink.prune.out':

- plink.prune.in: lists all SNPs remaining after pruning
- plink.prune.out: lists all SNPs removed during pruning

Both these files can subsequently be specified as the argument for PLINK's '–extract' or '–exclude' command respectively to provide a pruned set of SNPs.

To create a pruned version of the 'genstudy' dataset and call it 'p.genstudy':

```
plink --bfile genstudy --extract plink.prune.in --make-bed --out p.genstudy --noweb
```

To generate IBS and IBD estimates for all pairs of individuals, use the PLINK option '—genome':

```
plink --bfile p.genstudy --genome --out genstudy --noweb
```

Running this command creates an output file called "genstudy.genome". This includes a row per each unique pair of individuals, and fourteen columns, being the most important for the purpose of sample QC column 10, as it includes an estimate of the IBD proportion between individuals.

1.4.4.4 Heterozygosity

To run heterozygosity checks in PLINK, we can use the option '- -het':

```
plink --bfile genstudy --het --out genstudy
```

This command produces the output file "genstudy.het", which contains one row per individual and six columns. Column 6 includes the estimate of the inbreeding coefficient, F.

1.4.4.5 Removing individuals who fail sample QC

To remove individuals who fail any of the QC steps above.

1.4.4.5.1 Removing due to missingness

To investigate how many individuals are excluded at difference threshold levels (e.g. 1%, 5%, and 10%):

```
awk 'NR>1 && $6 >= 0.01 {print}' genstudy.CRstats.imiss | wc -l
awk 'NR>1 && $6 >= 0.05 {print}' genstudy.CRstats.imiss | wc -l
awk 'NR>1 && $6 >= 0.1 {print}' genstudy.CRstats.imiss | wc -l
```

To exclude all individuals with more than 5% missing data:

```
awk '$6 >= 0.05 {print}' genstudy.CRstats.imiss > genstudy.CRremove
plink --bfile genstudy --noweb --remove genstudy.CRremove --make-bed --out genstudy.CR
```

1.4.4.5.2 Removing due to gender discordance

To select only those individuals with an entry of 'PROBLEM' in column 5:

```
awk '$5=="PROBLEM"{print}' genstudy.sexcheck>genstudy.sexremove
```

To remove these individuals from the 'genstudy.CR' dataset, and create a new dataset named 'genstudy.sex':

```
plink --bfile genstudy.CR --noweb --remove genstudy.sexremove --make-bed --out genstudy.sex
```

1.4.4.5.3 Remove duplicates and related samples

We use an IBD threshold of 0.1875 to filter out either duplicated or closely related samples (column 10).

```
awk '$10 >= 0.1875 {print}' genstudy.genome > genstudy.PIremove
```

To remove this list of individuals from the 'genstudy.sex' dataset and create a new dataset 'genstudy_sampleqc':

```
plink --bfile genstudy.sex --noweb --remove genstudy.PIremove.list --make-bed --out genstudy_samp
```

1.4.4.5.4 Heterozygosity

Look at the "genstudy.Het_vs_imiss.pdf" and see if outliers exist. If not, there are no samples to remove.

1.4.4.6 SNP QC

1.4.4.6.1 Checks for missing data

Similar to what we did for sample QC, we can investigate how many SNPs fail different missingness thresholds by filtering column 5 in 'genstudy_sampleqc.CRstats.lmiss' using different threshold values. We can do this using the following command (e.g. for threshold of 1%):

```
plink --bfile genstudy_sampleqc --missing --out genstudy_sampleqc.CRstats --noweb
awk 'NR>1 && $5 >= 0.01 {print}' genstudy_sampleqc.CRstats.lmiss | wc -l
```

1.4.4.6.2 Checks for minor allele frequency (MAF)

To investigate how many SNPs have MAF<1% by first of all running the '–freq' option in PLINK:

```
plink --bfile genstudy_sampleqc --noweb --freq --out genstudy_sampleqc
awk '$5<0.01' genstudy_sampleqc.frq | wc -l
```

NOTE: Data quality tends to decrease with decreasing MAF. A low MAF implies rare genotypes which will be seen only a few times in your GWAS dataset. This in turn implies that there is less information that a genotype calling algorithm can use to call this genotype, and so calls are less certain. And the power to detect an association signal decreases with decreasing MAF. However, if a large MAF threshold is selected, this implies more SNPs are removed from study which possibly includes the true causal SNPs. A moderate MAF such as 0.05 is suggested.

1.4.4.6.3 Checks for adherence to Hardy-Weinberg Equilibrium

To generate a list of genotype counts and Hardy-Weinberg test statistics for each SNP:

```
plink --bfile genstudy_sampleqc --noweb --hardy --out genstudy_sampleqc
```

This provides and output file 'genstudy.hwe' and the p-value for the test for Hardy-Weindberg Equilibrium is in column 9 of this file.

To see how many SNPs have HWE p-value<0.0001 (in controls "UNAFF") and pvalue<0.00001:

```
awk '$3=="UNAFF" && $9<0.0001' genstudy_sampleqc.hwe | wc -l
awk '$3=="UNAFF" && $9<0.00001' genstudy_sampleqc.hwe | wc -l
```

To apply some extra SNP QC filters to our already sample-QC'd dataset:

- Minor allele frequency (MAF) >0.05 – option '–maf 0.05'
- SNP genotyping rate >95% (i.e. <5% missing genotypes for the SNP, across all individuals) – option '– geno 0.05'
- Hardy-Weinberg p-value >0.0001 in controls – option '– hwe 0.0001'

```
plink --bfile genstudy_sampleqc --noweb --maf 0.05 --geno 0.05 --hwe 0.0001 --make-bed
```

1.5 Testing for Associations

In this section, we will undertake an analysis of association:

- first, with no adjustment made for non-genetic variables (univariate analysis of association)

- then, with adjustment made for relevant non-genetic variables (multiple regression analysis)

1.5.1 Software and datasets

We will use the QC'd PLINK files (generated in the previous section) ready to run the analyses of association.

To run the analyses of association, we use a software package called 'SNPTEST (Version 2)' – it's free and available for the analysis of genome-wide association studies.

To use SNPTEST, input files need to be in a specific format:

- one '.gen' file including genotype data, which includes one row per SNP.
- one '.sample' file including phenotypic data. It has three parts (a) a header line detailing the names of the columns in the file, (b) a line detailing the types of variables stored in each column, and (c) a line for each individual including the information for that individual.

A '.gen' file includes one row per SNP. The first five entries on each line should be as follows:

- SNP ID: This entry is usually used to denote the chromosome number.
- rs number: This entry is a number which uniquely identifies the genotyped SNP. An rs number is an accession number used by researchers and databases to refer to specific SNPs. It stands for 'Reference SNP cluster ID'.
- Base pair position of the SNP: A value that described the SNP's position on the chromosome.
- Allele coded A (see below for further explanation of this): The entry here will be A, C, T or G i.e. corresponding to the four possible nucleotides.
- Allele coded B (see below for further explanation of this): The entry here will be A, C, T or G i.e. corresponding to the four possible nucleotides.

The remaining entries on each row represent the genotype of each individual at the SNP. Each individual will have three entries, representing their probability of having the 'AA', 'AB' and 'BB' genotypes respectively. As we do not have any imputed SNPs in the current dataset, the probability for each genotype will be either 0 or 1. So, an individual with genotype AA will be coded '1 0 0', with genotype AB will be coded '0 1 0', and with genotype BB will be coded '0 0 1'. If genotype is missing for an individual, all three entries are set to '0 0 0'.

It is possible to convert QC'd PLINK files to '.gen' and '.sample' format using a program called 'GTOOL'. We have also removed all chromosome X SNPs from these files, since in terms of association we are only interested in SNPs on chromosome 10.

1.5.2 Univariate analyses of association

To run univariate analyses of association between all SNPs in our 'genstudy_qc' file and our phenotype 'diseased', assuming an additive mode of inheritance, we would use the following SNPtest command:

```
snptest \
-data genstudy_qc.gen genstudy_qc.sample \
-o genstudy_qc_univariate_add.out \
-pheno diseased \
-frequentist 1 \
-method threshold
```

Due to the large number of output columns in 'genstudy_qc_univariate_dom.out' and the large number of SNPs investigated, it is useful to look at our results in summary format. A useful way of doing this is by preparing a Manhattan plot to give a graphical representation of our results. Essentially, this is a plot of pvalue versus base pair position for each SNP and is a useful way of identifying significantly associated SNPs.

First of all, it is necessary to re-format our output file into a format which is appropriate for the shell script.

1. Add chromosome number to each row (also need to delete final row in datafile since includes an additional line of text which is not needed)

```
awk 'NR>11{$1=10; print}' genstudy_qc_univariate_add.out >genstudy_qc_new.out
sed '$d' genstudy_qc_new.out>genstudy_qc_new2.out
```

2. Select only columns of interest, and label them:

```
echo "CHR SNP BP P" > genstudy_qc_Manhattan.txt;
awk '{print $1,$2,$4,$42}' genstudy_qc_new2.out >>genstudy_qc_Manhattan.txt
```

3. Create a new file ("genstudy_qc.unimp.snp") which lists all actually genotyped SNPs.

```
awk ' $9==1{print $2}' genstudy_qc_univariate_add.out > genstudy_qc.unimp.snp
```

4. Now, the data is prepared for generating the Manhattan plot.

To prepare a Q-Q plot of your results to ensure that there is no apparent genomic inflation in your results, which would suggest population substructure:

```
pvals <- read.table("genstudy_qc_Manhattan.txt", header=T)
observed <- sort(pvals$P)
lobs <- -(log10(observed))
expected <- c(1:length(observed))
lexp <- -(log10(expected / (length(expected)+1)))
pdf("qqplot.pdf", width=6, height=6)
plot(c(0,7), c(0,7), col="red", lwd=3, type="l", xlab="Expected (-logP)", ylab="Observe
```

```
xlim=c(0,7), ylim=c(0,7), las=1, xaxs="i", yaxs="i", bty="l")
points(lexp, lobs, pch=23, cex=.4, bg="black")
dev.off()
```

To extract a list of the SNPs giving the lowest p-values:

```
awk ' $4 <0.0001' genstudy_qc_Manhattan.txt > lowest_ps_univariate
awk ' $4>0' lowest_ps_univariate > lowest_ps_univariate_new #to filter out SNPs were it was not p
```

1.5.3 Multiple regression analyses

In the 'genstudy.sample' file you will notice four columns to the right of the phenotype column 'diseased'. These are labelled 'age', 'bmi', 'family' and 'sex', and represent four variables believed to be of potential interest in terms of being associated with our disease.

To decide whether to adjust for each of these variables, we first of all need to test whether they are univariately associated with our phenotype 'diseased'. As our phenotype is binary, we can use Student's t-test to test for association with age and bmi, and the chi-square test to test for association with family history and sex.

```
data=read.table("genstudy_qc.sample",header=T)
data=data[-1,]
attach(data)
res_age=t.test(as.numeric(age)~diseased)
res_bmi=t.test(as.numeric(bmi)~diseased)
res_family=chisq.test(family,diseased)
res_sex=chisq.test(sex,diseased)
res_age$p.value
res_bmi$p.value
res_family$p.value
res_sex$p.value
```

To now run our SNP association analyses using a multiple regression approach, we can add the element '-cov_names' followed by the clinical variable names to your command (assuming that we have decided to adjust for all clinical variables giving $p<0.05$ in our univariate analysis, we only have to adjust for age and family history):

```
snptest \
-data genstudy_qc.gen genstudy_qc.sample \
-o genstudy_qc_adjusted_add.out \
-pheno diseased \
-frequentist 1 \
-method threshold \
```

```
-cov_names age family
```

To prepare a Manhattan plot and QQ-plot of the results from your multiple regression analysis by adjusting the code used for plotting the univariate results:

```
awk 'NR>11{$1=10; print}' genstudy_qc_adjusted_add.out > genstudy_qc_new.out
sed '$d' genstudy_qc_new.out > genstudy_qc_new2.out
tail -n +2 genstudy_qc_new2.out > genstudy_qc_new3.out
echo "CHR SNP BP P" > genstudy_qc_adj_Manhattan.txt
awk '{print $1,$2,$4,$42}' genstudy_qc_new3.out >> genstudy_qc_adj_Manhattan.txt
awk '$9==1{print $2}' genstudy_qc_adjusted_add.out > genstudy_qc.adj.snp
sh QuickManhattan.sh
```

To obtain a list of the lowest p-values from this analysis, and compare them with your lowest pvalues from the univariate analysis:

```
awk ' $4 <0.0001' genstudy_qc_adj_Manhattan.txt > lowest_ps_adjusted
awk '$4>0' lowest_ps_adjusted > lowest_ps_adjusted_new
cat lowest_ps_adjusted_new
```

1.6 Population structure in GWAS

1.6.1 Testing for association under an additive model

To test for association under an additive model using the PLINK command, but without adjustment for any covariates:

```
plink --noweb --bfile EGCUT.clean --logistic --ci 0.95 --out plink.additive
```

To investigate the impact of population structure by generating a QQ plot of observed p-values against those that would be expected under the null hypothesis of no association. In the absence of population structure, we would expect most points to lie on the y=x line:

```
R --vanilla --slave --args plink.additive.assoc.logistic ADD qq_1.pdf < qqPlot.R
```

1.6.2 Performing multi-dimensional scaling

Multi-dimensional scaling in PLINK is performed in three steps:

1. Identifying a subset of "independent" common SNPs (minor allele frequency of at least 5%) that are not in linkage disequilibrium with each other.

```
plink --noweb --bfile EGCUT.clean --maf 0.05 --indep-pairwise 50 5 0.05
```

2. Calculating the relatedness between each pair of individuals using the set of independent SNPs which is a measure of genetic similarity.

```
plink --noweb --bfile EGCUT.clean --extract plink.prune.in --Z-genome
```

3. Perform multi-dimensional scaling using the relatedness matrix.

```
plink --noweb --bfile EGCUT.clean --read-genome plink.genome.gz --cluster --mds-plot 2
```

Note that the option –Z-genome produces a compressed genome file and saves on storage. The option –mds-plot specifies how many eigenvectors from the multi-dimensional scaling to summarise in the output file. The eigenvectors are output to the file plink.mds.

The output file has one row per individual, and provides the identifier used in the PLINK family file, together with the first two eigenvectors (columns C1 and C2). You can plot the first two eigenvectors against each other using the command:

```
R --vanilla --slave --args plink.mds EGCUT.clean.fam mds.pdf < mds.R
```

In the output plot, each point corresponds to an individual.

1.6.3 Testing for association under an additive model with adjustment for confounding

To account for population structure in our analysis, we can repeat our test of association under and additive model, but this type adjusting for eigenvectors from the multi-dimensional scaling as covariates:

```
plink --noweb --bfile EGCUT.clean --logistic --covar plink.mds --covar-name C1,C2 --ci 0.95 --out
```

To investigate the impact of population structure that is not accounted for by the first two eigenvectors from multi-dimensional scaling by generating a QQ plot with the command:

```
R --vanilla --slave --args plink.additive.mds.assoc.logistic ADD qq_2.pdf < qqPlot.R
```

To obtain association summary statistics for the SNP:

```
grep SNPID plink.additive.mds.assoc.logistic
```

1.7 Phasing and Imputation

By utilising genotype imputation methods, we can investigate whether there are neighbouring SNPs to a particular SNP that generated a significant p-value, also sharing evidence of association with the phenotype. The focus in this section

will be on imputing genotypes within close proximity of a significant SNP, and then to re-run the association analyses on the imputed dataset.

1.7.1 Pre-phasing and imputation

Prior to undertaking genotype imputation, it is necessary to "pre-phase" the genotype data to convert it into haplotype form.

Doing this affords more efficient imputation as the reference panels on which we base our imputation are all in haplotype form as well. Haplotypes span across many locus so it is important that when focusing in on a certain region of the genome, (for example the region surrounding a significant SNP as we are in this practical), that we allow a sufficient buffer zone around the region. If we didn't allow this buffer zone then we could interrupt haplotype construction with the locus of interest!

To undertake the pre-phasing we use a software package called SHAPEIT2 (https://mathgen.stats.ox.ac.uk/genetics_software/shapeit/shapeit.htm).

The causal SNP we found before had base pair position 64803579 and so a sensible region to pre-phase might be from 63Mb to 68Mb, which will include the immediate region surrounding the causal SNP, plus a buffer region at either end.

To pre-phase the "genstudy_pp" dataset, we can run the following SHAPEIT2 command:

```
shapeit --input-ped genstudy_pp.ped genstudy_pp.map --input-map chr10.map --output-max
```

Once we have pre-phased our data, we can undertake genotype imputation and to do this we will use a software package called "IMPUTE2" (https://mathgen.stats.ox.ac.uk/impute/impute_v2.html).

After running SHAPEIT2 on the "genstudy_pp" dataset, we obtained a "genstudy.ph.haps" datafile which contained phased haplotypes for our specified region. We can now run IMPUTE2 on this .haps datafile to impute genotypes at SNPs not originally genotyped, but included on our chosen reference panel. For the purpose of this practical, we will use the reference panel from the 1000 Genomes Project (http://www.1000genomes.org). To do this, we can use the following IMPUTE2 command:

```
impute2 -use_prephased_g -known_haps_g genstudy.ph.haps -m chr10.map -h chr10.haps.gz
```

Please note that the interval we specify under the option "-int" (64300000 to 65300000) is narrower than the one we specified in the SHAPEIT2 command. This is because the imputation process will produce genotypes rather than haplotypes. Therefore we can narrow down to a 1Mb region of interest.

After the Imputation has run, in the screen output there is a section called "Imputation accuracy assessment". The interval here denotes the probability range at which a SNP genotype has been called and the concordance with the patient genotypes. These figures can be improved by increasing the number of patients in the study or with better match reference haplotypes.

1.7.2 Re-running our univariate analyses of association

Previously, when we were dealing only with actually genotyped SNPs, our .gen file ("genstudy_qc.gen") included the first five compulsory columns, followed by three columns for each genotyped SNP. The three columns represented an individual's probability of having the "AA", "AB" and "BB" genotypes respectively. As we did not have any imputed SNPs, the probability for each genotype was either 0 or 1. However, now that we are dealing with imputed SNPs, the probability for each genotype can be any number between 0 and 1, representing our genotype uncertainty.

```
head -n 1 genstudy_imp.gen
```

Please, notice that there are number others than 0 and 1 in the columns representing genotype probabilities for the SNPs.

To run univariate analyses of association on a dataset which includes imputed genotypes, we can again use the SNPTest programme. However, the element "method" within the SNPTest command needs to be specified differently. The "method" element controls the way in which genotype uncertainty is accounted for in the analyses of association, bearing in mind that imputation can only provide a 'best estimate' of what the true genotype at a SNP would be.

- threshold: Assumes a particular genotype at a SNP only if accuracy of the genotype call is above a given threshold. The calling threshold is controlled by the flag -call_thresh and the default calling threshold is 0.9.
- expected: Uses expected genotype counts, also known as genotype dosages, in the analyses of association.
- score: Uses a missing data likelihood score test in the analyses of association.
- ml: Again uses missing data likelihood approach, but with multiple Newton-Raphson iterations to estimate the parameters in the missing data likelihood.
- em: Again uses missing data likelihood approach, but with an EM algorithm to estimate the parameters in the missing data likelihood.

So, to run a univariate analysis of association on our imputed dataset, assuming the threshold method we would use the following command:

```
snptest -data genstudy_imp.gen genstudy_qc.sample -o genstudy_qc_univariate_add.imp.out -pheno di
```

The SNPtest output file "genstudy_qc_univariate_add.imp.out" can now be scrutinised in much the same way as we did for the output file initially – i.e. we can produce Manhattan and QQ plots of the resulting p-values – except for one additional and important QC step which needs to occur post-imputation. This QC step involves filtering out SNPs with poor imputation accuracy. The SNPtest output file has a column labelled "INFO", which contains, for each SNP, a number indicating how accurately it has been imputed. The value of INFO will range from 0 (no certainty in imputation) to 1 (perfect accuracy in imputation/actually genotyped SNP). Poorly imputed SNPs are likely to contribute to false positive results whilst removing a large number of imputed SNPs will distort the overall results. Therefore, as a QC step we need to identify an appropriate, yet not too stringent, threshold for the level of accuracy that is allowed. Unfortunately, there is no exact science to choosing the threshold; the SNPtest website (https://mathgen.stats.ox.ac.uk/impute/impute_v2.html) views INFO thresholds between 0.3 and 0.5 as reasonable. For this analysis we will utilise an INFO threshold of 0.4.

This additional QC step can be implemented very easily by:

```
awk 'NR>11&&$9>0.4{$1=10;print}' genstudy_qc_univariate_add.imp.out >genstudy_qc_new.in
```

Chapter 2

AWS

2.1 AWS S3

2.1.1 Accessing AWS using the AWS Command Line Interface

The AWS Command Line Interface (or AWS CLI) is an open source tool that enables you to interact with the AWS services from your command-line Shell. AWS CLI tool is already installed in our server, and it is very simple to use. Below are described the most commonly used interactions with AWS.

2.1.1.1 Configure the AWS CLI service

1. To configure the service, please type "aws configure" on the console. This command is interactive, so the service will prompt you four times to enter some config information. Below an example (all RytenLab members have to ask me to send them their secret AWS credentials):

```
$ aws configure

AWS Access Key ID [None]: your_access_key
AWS Secret Access Key [None]: your_secret_key
Default region name [None]: eu-west-2
Default output format [None]: json
```

2. To check whether the connection with AWS has been correctly stablished, we can type the following command to list all our current buckets.

```
$ aws s3 ls
```

2.1.1.2 Upload a single file to AWS

Let's suppose we have a bucket on AWS called 'my-bucket'. Let's also suppose you have a file called 'myfile.txt' stored in your local that you would like to upload to AWS. To upload the file 'myfile.txt' to the bucket 'my-bucket':

```
$ aws s3 cp myfile.txt s3://my-bucket/
```

2.1.1.3 Download a single file from AWS

To download the file 'myfile.txt' from the 's3://my-bucket/' AWS bucket into your local folder:

```
$ aws s3 cp s3://my-bucket/myfile.txt ./my_local_folder
```

2.1.1.4 Upload multiple files to AWS

To upload multiple files, we can use the sync command. The **sync** command syncs objects under a specified local folder to files in a AWS bucket by uploading them to AWS.

```
$ aws s3 sync my_local_folder/ s3://my-bucket/
```

2.1.1.5 Download multiple files from AWS

To download multiple files from an AWS bucket to your local folder, we can also use the **sync** command by changing the order of the parameters.

Please, be aware the costs associated with downloading files correspond to $0.090 per GB - first 10 TB / month data transfer out beyond the global free tier.

```
$ aws s3 sync s3://my-bucket/my_remote_folder/ ./my_local_folder
```

2.1.2 Checking AWS file integrity

Considering the example used in the previous section, let's check the integrity of the local folder './my_local_folder' matches with the integrity of the remote AWS folder 's3://my-bucket/my_local_folder/'.

1. First, clone the 'aws-s3-integrity-check' repo.

```
$ git clone https://github.com/SoniaRuiz/aws-s3-integrity-check.git
```

2. Clone the 's3md5' repo.

```
$ git clone https://github.com/antespi/s3md5.git
```

3. Move the s3md5 folder within the aws-s3-integrity-check folder:

```
$ mv ./s3md5 ./aws-s3-integrity-check
```

4. Next, grant execution access permission to he s3md5 script file.

```
$ chmod 755 ./aws-s3-integrity-check/s3md5/s3md5
```

5. The aws-s3-integrity-check folder should look similar to the following:

```
total 16
-rw-r--r-- 1 your_user your_group 3573 date README.md
-rwxr-xr-x 1 your_user your_group 3301 date aws_check_integrity.sh
drwxr-xr-x 2 your_user your_group 4096 date s3md5
```

6. Execute the script 'aws_check_integrity.sh' following the this structure: **aws_check_integrity.sh 'local_path' 'bucket_name' 'bucket_folder'**

```
$ aws_check_integrity.sh ./my_local_folder my-bucket my_local_folder/
```

2.1.3 Creating a new AWS bucket

To create a new AWS bucket, we recommend using the following configuration:

1. Region: EU London
2. Block all public access: enabled
3. Bucket Versioning: enable
4. Tags:
    1. Key = "data-owner" / Value = "your name"
    2. Key = "data-origin" / Value = "the origin of the data in one word (i.e. the name of a project, the name of a server)"
5. Default encryption: enable - Amazon S3 key (SSE-S3)
6. Advanced settings
    1. Object Lock: enable/disable, depending on your type of data (more info here)

2.2 AWS EC2

All commands here shown have been used to configure an EC2 virtual instance under the AWS free-tier.

2.2.1 Generating an EC2 instance

To generate an EC2 instance under the free-tier using an Ubuntu 20.04 LTS image, please follow the next steps:

- Log in on the AWS Console using your AWS account.
- Click on Services –> EC2 –> Instances.
- Click on the "Launch Instances" button. It is an orange button located on the upper right-hand side of the web page.
  - Step 1: Choose an Amazon Machine Image (AMI). Click on the "Free tier only" checkbox (located on the left-hand side menu) and select an AIM from the list. In this example I am going to use the "Ubuntu Server 20.04 LTS (HVM), SSD Volume Type" AMI.
  - Step 2: Choose an Instance Type. Select "t2.micro" (free tier eligible).
  - Step 3: Configure Instance Details. Leave all options by default and click "Next".
  - Step 4: Add Storage. Change the size of your disk from 8GB to 25GB. Free tier allows to get up to 30 GB of (SSD) disk storage.
  - Step 5: Add Tags. Add any tags or just leave it by default and click Next.
  - Step 6: Configure Security Group. We will need to add an additional HTTP rule and two custom TCP rules to set the ports to 3838 (for RServer to run) and 8787 (for RStudio to run). See "Step 5: Configuring the Security Group" section from this post for more details.
  - Step 7: Review and Launch. Review the info and click "Launch".
  - Step 8: Select or Create Key Pair. From the dropdown, choose "Create a new key pair". Name the .pem file and store it securily in your local computer. You will need later to connect to your EC2 instance.

2.2.2 Connecting to an EC2 instance

SSH login to your EC2 instance:

- From your AWS Console. Click on "Services –> EC2 –> Instances". Check your newly created instance is running.
- Click on the "Connect" button and select the "SSH Client" tab.
- Follow the instructions indicated there to connect to your EC2 instance from an SSH client (I normally use MobaXterm).

2.2.3 Installing/Running RStudio on an EC2

```
## More info: https://jagg19.github.io/2019/08/aws-r/

## Update the Ubuntu repos
# update indices
```

```
sudo apt update -qq
# install two helper packages we need
sudo apt install --no-install-recommends software-properties-common dirmngr
# add the signing key (by Michael Rutter) for these repos
# To verify key, run gpg --show-keys /etc/apt/trusted.gpg.d/cran_ubuntu_key.asc
# Fingerprint: 298A3A825C0D65DFD57CBB651716619E084DAB9
sudo wget -qO- https://cloud.r-project.org/bin/linux/ubuntu/marutter_pubkey.asc | sudo tee -a /et
# add the R 4.0 repo from CRAN -- adjust 'focal' to 'groovy' or 'bionic' as needed
sudo add-apt-repository "deb https://cloud.r-project.org/bin/linux/ubuntu $(lsb_release -cs)-cran


# Update ubuntu package repo, to get latest R
sudo apt update

# Install R
sudo apt -y install r-base r-base-dev

# Install shiny before shiny-server
sudo R -e "install.packages('shiny')"


# Install debian package manager, gdebi
sudo apt install gdebi-core


# Install Shiny Server (https://www.rstudio.com/products/shiny/download-server/)
sudo wget https://download3.rstudio.org/ubuntu-14.04/x86_64/shiny-server-1.5.17.973-amd64.deb
sudo gdebi shiny-server-1.5.17.973-amd64.deb
sudo rm shiny-server-1.5.17.973-amd64.deb


# Dependencies for R packages like RMariaDB, devtools, tidyverse, sparklyr. Please run seperate.
sudo apt -y install libcurl4-openssl-dev
sudo apt -y install libssl-dev libxml2-dev libmariadbclient-dev build-essential libcurl4-gnutls-d


# Install RStudio (https://www.rstudio.com/products/rstudio/download-server/)
wget https://download2.rstudio.org/server/bionic/amd64/rstudio-server-2021.09.2-382-amd64.deb
sudo gdebi rstudio-server-2021.09.2-382-amd64.deb
sudo rm rstudio-server-2021.09.2-382-amd64.deb


# Install some useful R Packages
sudo R -e "install.packages('RCurl', repos='http://cran.rstudio.com')"
sudo R -e "install.packages('devtools', repos='http://cran.rstudio.com')"
```

```
sudo R -e "install.packages('tidyverse')"
sudo R -e "install.packages('RMariaDB')"


# Add user info to login RStudio
sudo adduser user_name
#Add rstudio to sudo group
sudo usermod -aG sudo user_name


# Install Java and reconfigure in R for RStudio use
sudo apt -y install default-jdk
sudo R CMD javareconf
# Change permissions for R library
sudo chmod 777 -R /usr/local/lib/R/site-library


#Restart rstudio-server to incorporate changes made in rserver.conf
sudo rstudio-server restart
```

2.2.4 Access RStudio through your browser

- From your AWS Console, click on "Services –> EC2 –> Instances".
- Click on the checkbox located next to the EC2 instance to select it.
- At the bottom of the web page it should have appeared a section with info from the selected instance.
- Copy the URL indicated under the "Public IPv4 DNS" section.
- Add the port 8787 to the URL (i.e. http://.....:8787),
- Paste the URL on your browser. RStudio should now appear on your browser.

2.3 AWS List of Cloud Services

Table 2.1: list of AWS cloud services.

name	description
Amazon API Gateway	Service to develop APIs. These APIs will act as the fron
Amazon Athena	SQL to analyse data stored in S3.
Amazon Aurora	MySQL-Compatible Relational database for the cloud.
Amazon Aurora	PostgreSQL-Compatible Relational database for the clou
Amazon Bracket	Quantum computing service.
Amazon Carrier	A Carrier IP address is the address that I will assign to a
Amazon Chime	To let users meet and chat online.
Amazon CoudWatch	Monitoring and management service providing insights fo
Amazon Code Guru Reviewer	Reviewer Service that uses program analysis and machin
Amazon Cognito	To add user sign-up, sign-in and access control to my we
Amazon Comprehend	For text processing and analysis.
Amazon Comprehend Medical	HIPAA-eligible natural language processing (NLP)
Amazon DocumentDB	Document database service
Amazon DynamoDB	Is a key-value and document database that delivers singl
Amazon EC2	Compute platform
Amazon EC2 Dedicated Hosts	Allows you to use your eligible software licences on Amaz
Amazon EKS	Amazon Elastic Kubernetes Service
Amazon Elastic Block Store (EBS)	Allows you to create persistent block storage volumes an
Amazon Elastic Container Registry (ECR)	To store, manage, and deploy Docker container images.
Amazon Elastic File System (EFS)	Provides a file system for use with AWS Cloud services.
Amazon Elastic Graphics	Allows you to attach low-cost graphics acceleration to EC
Amazon Elastic IP Static	IPv4 for dynamic cloud computing.
Amazon Elastic Transcoder	Is media transcoding in the cloud. Designed to convert (
Amazon ElastiCache	Build data-intensive apps or improve the performance of
Amazon Elasticsearch Service	To operate Elasticsearch at scale with zero downtime.
Amazon EMR	Industry-leading cloud big data platform for processing v
Amazon FSx for Lustre	Integrated with S3, is designed for fast processing of wor
Amazon FSx for Windows File Server	To move your Windows based applications that require f
Amazon GuardDuty	Is a threat detection service that monitors for malicious a
Amazon Kinesis Data Analytics	To analyse streaming data (for responding to your busine
Amazon Kinesis Data Firehose	To only pay for the volume of data ingested into the serv
Amazon Kinesis Data streams	Data streaming service.
Amazon Lookout for Vision	Machine learning (ML) service that spots defects and an
Amazon Managed Streaming for Apache Kafka (MSK)	To easily run applications that use Apache Kafka
Amazon MQ	To set up and operate message brokers in the cloud.
Amazon Neptune	Graph database service to build and run applications tha
Amazon Polly	Service that turns text into lifelike speech.
Amazon QuickSight	Business intelligence service to deliver insights to everyon
Amazon RDS for MariaDB	To set up, operate and scale MariaDB database deploym
Amazon RDS for MySQL	”
Amazon RDS for Oracle	
Amazon RDS for PostgreSQL	
Amazon RDS for SQL server	
Amazon Redshift	Petabyte-scale data warehouse
Amazon Rekognition	Images and video analysis recognition; to identify objects
Amazon Route 53	DNS web service
Amazon S3 Glacier	
Amazon SageMaker	To manage costs associated to ML instances

Chapter 3

Docker

3.1 Introduction

Nowadays, almost all companies and scientific labs which work with data are also becoming software producers. In some cases, the main reason for this software development is to publish effective scientific work. In other cases, its final aim might be just putting together a set of related functions that work for a similar final objective. However, in both cases, the final software product becomes highly shareable.

This characteristic, the shareability of the software produced, might seem very straightforward but in reality, it can turn into a scaring daunting task to be solved. Why? Because of the heterogeneity of all different platforms, operating systems, dependencies, versioning and so on that our software product makes use of. Please, be aware that prior to its first release, our software product might have been tested in a limited number of PC stations with determined characteristics that can be extremely different from any other potential user's PC station around the world.

All these reasons make Docker Platform really interesting and useful for both software producers and consumers.

3.2 What is docker?

Docker is a software platform created in 2013 by Docker, Inc. which its main objective is to *build, share, and run any app anywhere*, independently of the platform and environment where it is executed.

But what does this definition mean? It means that you admiringly can forget

about dependencies, libraries, compatibility, etc. to make the app run correctly. To some extent, you could think Docker as a *black box*, as a **snapshot** of the developer's laptop where she or he developt the software you are about to make use of. A snapshot of the precise moment when the developer decided to release the software. Thus, everything you need to run the app is already there, installed and configured inside the Docker object, ready to be used, with no compatibility issues at all.

3.3 Images and containers

Working with Docker, there are two main concepts you are going to hear about constantly: images and containers. An **image** is the virtual file or template that contains the whole set of instructions to build a container. An image is the raw Docker file you will directly download from Docker Hub developer's repository to your local. On the other hand, a **container** is the executable object directly generated from the image. A container will be the virtual object that represents the snapshot of the app developer's laptop.

In summary, the image is the virtual file that contains the raw instructions to build the executable app, and the executable app is the container itself.

3.4 Docker Hub

Docker Hub is an online platform that allows creating individual Docker repositories. A Docker repository is a personal account space where you can store one or more versions of the same Docker image, which are represented by tags.

Let us focus on the following image obtained from the Ubuntu Docker Repository:

3.5 Useful Commands

This page contains a list with some of the most common commands of Docker.

To download a Docker image from Docker Hub:

```
$ sudo docker push repository/name:tag
```

To run the image *name:tag*. The flag –**rm** indicates to remove the container after stopping the image; whereas the flag **-p** indicates the port on which we want to expose the execution of the image:

```
$ sudo docker run --rm -p 8500:80 name:tag
```

To list all docker images that are available in our local:

```
$ sudo docker images
```

To remove the image *image_name*:

```
$ sudo docker image rm image_name
```

To remove all orphaned images:

```
$ sudo docker rmi $(sudo docker images -f dangling=true -q)
```

To list all current containers:

```
$ sudo docker ps
```

To list all stopped containers:

```
$ sudo docker ps -a
```

To remove all orphaned containers:

```
$ sudo docker rm $(sudo docker ps -a -q)
```

To enter inside a container in execution:

```
$ sudo docker exec -it name_container /bin/sh
```

3.6 Docker & CoExp Web Application

This tutorial contains the instructions to install a local version of the CoExp Webpage by making use of the Docker technology. All the commands shown in this document have been tested in a Ubuntu18.04 machine.

3.6.1 Software requirements

Before downloading the CoExp Docker images, we first need to prepare the environment for the correct execution of Docker.

Thus, let's download/fetch new versions of the packages that are already installed in our Linux machine (all these commands have been tested in a Ubuntu18.04 machine):

```
$ sudo apt update
$ sudo apt upgrade
```

curl is a tool used to transfer data. We will make use of it later when we download the CoExp Docker images. To install *curl* in the machine:

```
$ sudo apt install curl
$ sudo curl --version
```

Now, we are ready to install the Docker technology in the machine. So, let's download it:

```
$ sudo apt install docker.io
```

Once the Docker installation has finished, we can enable and start it. The last instruction *sudo docker –version* will return the current Docker version installed:

```
$ sudo systemctl start docker
$ sudo systemctl enable docker
$ sudo docker --version
```

Finally, we need to install Docker-compose (more info here). Docker-compose is a brach of the Docker technology, which allows communicating different Docker images between them:

```
$ sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
$ docker-compose --version
```

3.6.2 Download Docker images of CoExp

If everything has gone as expected, the system should now be ready to download the two CoExp Docker images. There are two different images because one contains the user interface (UI) of the CoExp webpage, and the other one contains the backend of the CoExp WebPage (author Juan Botia).

In terms of the back-end of the CoExp Webpage, there are two different docker images available:

1. Complete version: this docker image contains the totality of all CoExp networks, and it is about ~4.5GB of size.
2. Lite version: this smaller docker image contains only the ROSMAP co-expression network. This image is about ~1.3GB of size.

Depending on which image you are interested in, the commands to execute are:

To download the complete version:

```
$ sudo docker pull soniaruiz/coexp:r
```

To download the lite version:

```
$ sudo docker pull soniaruiz/coexp:r-lite
```

3.6.3 Use Docker-Composer to build the images

The next step is to make the comunication between the two docker images possible. For that purpose, we need to download this **docker-compose.yml**

file (in case you have opted by the complete backend version), or this **docker-compose.yml** file (if you have opted by the lite version). In any case, this file will make possible the correct communication between the two Docker images we downloaded in the previous step.

Additionally, the location of the downloaded **docker-compose.yml** file is not really important, but we recommend to place it in your **Home** folder (in case you are using a Linux machine).

Once the download has finished, use the following command to execute the **docker-compose** file and, therefore, to run your own Docker CoExp webpage:

```
$ sudo docker-compose up
```

Finally, to test whether the execution has been correct, please type the following URL into the address bar:

```
http://localhost:8088/
```

If everything has gone as expected, you should now be able to visualize your dockerized version of the CoExp webpage in your browser. Congratulations!

3.6.4 Juan's tutorial

Suppose we want to generate local TOM (Topology Overlap Measure) modules from a specific network so we may, independently from the CoExpNets Web application or within the application, plot or use networks in graph mode. We can do it by creating their module TOMs and get the corresponding graph in terms of a connectivity matrix we can plot.

We will exemplify this by using the frontal cortex network from GTEx V6 package as follows.

We launch all package stuff so we can start working with those networks.

```
library(CoExpNets)
library(CoExpGTEx)
CoExpGTEx::initDb()

netf = getNetworkFromTissue(tissue="FCortex",
                            which.on="gtexv6",
                            only.file=T)
```

And now (we assume it is not generated yet) create the module-TOMs for the Frontal Cortex Network as follows. As we see,

```
netf
```

the beta value for that network is 9, it is in the name between the tissue and the `.it.50.rds` token in the file name.

```
getModuleTOMs(tissue="FCortex",
              beta=9,
              out.path="~/tmp/mytoms/",
              which.one="gtexv6")
```

And we can see all module-TOMs created now at the `~/tmp/mytoms/` folder

```
list.files("~/tmp/mytoms/",full.names = F,recursive = F)
```

And now we can get any module's connectivity matrix so we can represent a graph for the TOM

```
getModuleTOMGraph(tissue="FCortex",
                  which.one="gtexv6",
                  module="black",
                  topgenes=10,
                  out.path="~/tmp/mytoms/")
```

And there you have it.

Chapter 4

GitHub

4.1 Merge

To merge two branches presenting conflicts, it is possible to indicate which changes are preferred by using a –ours/–theirs flag:

First, we checkout to the branch that we want to apply/commit the merge:

```
> git checkout testing
> git log --oneline
```

Merging would generate the following error:

```
> git merge testchanges2
warning: Cannot merge binary files: obj/Debug/netcoreapp2.1/CoExp_Web.csprojAssemblyReference.cache
warning: Cannot merge binary files: obj/Debug/netcoreapp2.1/CoExp_Web.Views.pdb (HEAD vs. testchange
warning: Cannot merge binary files: obj/Debug/netcoreapp2.1/CoExp_Web.Views.dll (HEAD vs. testchange
warning: Cannot merge binary files: .vs/CoExp_Web/v16/.suo (HEAD vs. testchanges2)
Auto-merging obj/Debug/netcoreapp2.1/CoExp_Web.csprojAssemblyReference.cache
CONFLICT (content): Merge conflict in obj/Debug/netcoreapp2.1/CoExp_Web.csprojAssemblyReference.ca
Auto-merging obj/Debug/netcoreapp2.1/CoExp_Web.Views.pdb
CONFLICT (content): Merge conflict in obj/Debug/netcoreapp2.1/CoExp_Web.Views.pdb
Auto-merging obj/Debug/netcoreapp2.1/CoExp_Web.Views.dll
CONFLICT (content): Merge conflict in obj/Debug/netcoreapp2.1/CoExp_Web.Views.dll
Auto-merging libman.json
CONFLICT (content): Merge conflict in libman.json
Auto-merging Views/Shared/_Layout.cshtml
CONFLICT (content): Merge conflict in Views/Shared/_Layout.cshtml
Auto-merging Views/Run/Help_Introduction.cshtml
Auto-merging Views/Run/Help_Catalogue.cshtml
Auto-merging Views/Run/Help_Annotation.cshtml
```

```
Auto-merging Views/Run/Help.cshtml
Auto-merging Views/Run/About.cshtml
Auto-merging .vs/CoExp_Web/v16/.suo
CONFLICT (content): Merge conflict in .vs/CoExp_Web/v16/.suo
Automatic merge failed; fix conflicts and then commit the result.
```

To obtain the detail of the files that present a conflict:

```
> git log --merge
> git log --merge
commit 29cd34cc267a858fe696e86981e83bd3af1abb85 (testchanges2)
Author: Sonia Garcia <43370296+SoniaRuiz@users.noreply.github.com>
Date:   Tue Nov 24 09:03:14 2020 +0000

    commiting .suo file in testchanges2

commit 7319379b1038dc5415cf035f915d91096c70af2f (origin/testchanges2)
Author: Sonia Garcia <43370296+SoniaRuiz@users.noreply.github.com>
Date:   Tue Nov 24 00:47:04 2020 +0000

    remove unnecessary libraries

commit 5bf600b6f5f1b41b42faaca9173f33efac4958f4
Author: Sonia Garcia <43370296+SoniaRuiz@users.noreply.github.com>
Date:   Mon Nov 23 23:49:54 2020 +0000

    new built

commit 1f60b3ffb5b558849fa061efd7af1796380de30c
Author: Sonia Garcia <43370296+SoniaRuiz@users.noreply.github.com>
Date:   Mon Nov 23 23:20:51 2020 +0000

    Update .suo

    commit suo

commit d1834c7ed4a83a95e08a148c7945525e3521981c
Author: Sonia Garcia <43370296+SoniaRuiz@users.noreply.github.com>
Date:   Mon Nov 23 23:10:18 2020 +0000

> git merge --no-ff testchanges2
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.
```

As we want to apply the comming changes, we use the flag “–theirs”, and the

paths of the 7 conflictive files are updated:

```
> git checkout --theirs ./*
Updated 7 paths from the index
```

Then, we commit the changes of those 7 files and push them to the current repository (the 'testing' repository):

```
> git branch
  master
  testchanges
  testchanges2
* testing
> git add ./*
> git commit -m "checkout --theirs testing"
[testing bcc3b5c] checkout --theirs testing
> git push
...
> git status
On branch testing
Your branch is up to date with 'origin/testing'.

nothing to commit, working tree clean
```

Now, it's time to checkout to the repository we want to merge from and apply the merge:

```
> git checkout testchanges2
Switched to branch 'testchanges2'
Your branch is ahead of 'origin/testchanges2' by 1 commit.
  (use "git push" to publish your local commits)

> git merge testing
Updating 29cd34c..bcc3b5c
Fast-forward
 Program.cs                          |  2 +-
 Properties/launchSettings.json      |  4 ++--
 Views/Run/About.cshtml              | 18 ++++++++++---------
 Views/Run/Help.cshtml               |  4 ++--
 Views/Run/Help_Annotation.cshtml    |  4 ++--
 Views/Run/Help_Catalogue.cshtml     |  4 ++--
 Views/Run/Help_Introduction.cshtml |  4 ++--
 7 files changed, 20 insertions(+), 20 deletions(-)

> git checkout testing
Switched to branch 'testing'
Your branch is up to date with 'origin/testing'.
```

```
> git merge testchanges2
Already up to date.
```

Once the merge is done, we can apply the commit to the 'testing' repository and push the changes. After that, both repositories ('testchanges2' and 'testing') could be considered as merged, being the final result of their merge stored within the 'testing' repository:

```
> git add ./*

> git commit -m "testing version with the plot"
[testing 7c2f7dd] testing version with the plot
 18 files changed, 90 deletions(-)
 ...

> git push origin testing
 ...
```

4.1.1 Troubleshooting pages of interest

- This
- Merge a branch in Git
- Meaning of 'ours' and 'their' in Git.)

4.2 Stash

When we are working on our GitHub project, we may want to backup some of our changes. However, it is possible that those changes are not stable yet, or we think they are not ready to be committed into our repository. The stash command creates a backup copy of our changes, and keeps it ready to be committed into the repo once we feel it is ready.

It might happen that, once we think the stash is ready to be committed, GitHub doesn't allow us to do so because there are some 'untracked' files on it.

The only way I have found to deal with this issue, is to upload my stash into a brand new repo, and finally merge the new repo with the one I was intending to commit the changes to originally.

Chapter 5

Machine Learning Concepts

5.1 Introduction

Have you ever asked yourself what is the difference between **Artificial Intelligence** and **Machine Learning**? What about between **supervised** and **unsupervised learning**? Well, that's not surprising at all because trying to find out the right answer within the huge ocean of information that is the Internet, can become a really daunting task.

In this post, we will try first to define the most widely used Machine Learning concepts and finally trying to give clarifying examples of each one of them to try to help in the understanding of their meanings.

5.2 Artificial Intelligence vs. Machine Learning

What is the difference between Artificial Intelligence and Machine Learning? **Artificial Intelligence** is the concept of machines being able to perform tasks in a way that we would consider "smart". **Machine Learning** however is the current application of AI, where we just give machines access to data and let them learn for themselves (source Forbes).

The Machine Learning concept comprises different techniques whereby it is possible to make machines learning from diverse sets of data. Among the most important ones, we can highlight **supervised learning** and **unsupervised learning**.

5.3 Supervised learning

When the training data - the data we want machines learning about - comprises not only the input vectors but also their corresponding target vectors.

5.3.1 Classification

Classification is a supervised learning method used when the target vectors consist of a finite number of discrete categories.

The *iris* dataset available in R, for instance, can be used in classification problems because it provides different input vectors (*"Sepal.Length", "Sepal.Width", "Petal.Length" and "Petal.With"*) and a target vector (*"Species"*) with a finite number of categories (*"setosa", "versicolor" and "virginica"*).

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

5.3.2 Regression

Regression is also a supervised learning method but only used when the target vectors consist of one or more continuous variables.

The *longley* R dataset is an example of this type of data. It presents a collection of inputs vectors (*"GNP.deflator", "GNP", "Unemployed", "Armed.Forces", "Population", "Year"*) and a numeric vector output (*"Employed"*).

```
> summary(longley)
  GNP.deflator         GNP          Unemployed     Armed.Forces     Population
 Min.   : 83.00   Min.   :234.3   Min.   :187.0   Min.   :145.6   Min.   :107.6   Min.
 1st Qu.: 94.53   1st Qu.:317.9   1st Qu.:234.8   1st Qu.:229.8   1st Qu.:111.8   1st
 Median :100.60   Median :381.4   Median :314.4   Median :271.8   Median :116.8   Media
 Mean   :101.68   Mean   :387.7   Mean   :319.3   Mean   :260.7   Mean   :117.4   Mean
 3rd Qu.:111.25   3rd Qu.:454.1   3rd Qu.:384.2   3rd Qu.:306.1   3rd Qu.:122.3   3rd
 Max.   :116.90   Max.   :554.9   Max.   :480.6   Max.   :359.4   Max.   :130.1   Max.
```