Tutorial 4
Economics and Genetics (FEB13089)

Erasmus School of Economics – Bachelor (Block 2)

Learning goals

- Being able to perform power calculations in *R*.
- Being able to run a Genome-wide Association study in PLINK.
- Being able to visualize Genome-wide Association study results in *R*.

Power calculations in *R*

As discussed in Lecture 4, it is essential to well-powered from a statistical point of view to find reliable and replicable associations between SNPs and phenotypes. We perform the power calculation in *R* (see Tutorial 1 for how to get *R* running on your computer). Launch *R* by starting *RStudio.* To perform power calculations, we make use of the *R* package “pwr”. This add-on package needs to be installed first. Go to *Tools* ⟶ *Install packages…* Type *pwr*, select the package, and click “install”. Alternatively, install the *pwr* package from the command line using the following command:

```
install.packages("pwr")
```

To be able to work with add-on *R* packages, you should first load them using the “library” command:

```
library(pwr)
```

For our interest, the function “pwr.r.test” is most important. This function takes the following five arguments as input:

`n`	Number of observations (individuals)
`r`	Linear correlation coefficient (in our case, squareroot of the explained variance R^2)
`sig.level`	Significance level (Type I error probability, α)
`power`	Power of test (1 minus Type II error probability, π)
`alternative`	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less"

Exactly one of the input arguments “r”, “n”, “power” and “sig.level” must be passed as NULL, and that parameter is then determined from the others. Through three examples, we practice how power calculations help to determine the conditions under which a well-powered GWAS can be conducted. Remember, in a GWAS the significance threshold α is set to $5x10^{-8}$, and 80% power is usually considered to be adequate.

Example 1: How many observations are needed to find a SNP that explains 0.02% of the phenotypic variance at the genome-wide significance level of $5x10^{-8}$ with 80% power?

The “pwr.r.test” works with standardized regression coefficients as input, which is in our case the square root of the variance explained. Therefore, we use r = sqrt(0.0002) as input parameter. We perform a two-sided test, because here we are not interested in the sign of the association but in the variance explained.

```
pwr.r.test(n = NULL, r = sqrt(0.0002), sig.level = 5e-8, power = 0.80, alt
ernative = "two.sided")

     approximate correlation power calculation (arctangh transformation)

              n = 197984.4
              r = 0.01414214
      sig.level = 5e-08
          power = 0.8
    alternative = two.sided
```

Hence, 197984 observations are needed in this setting.

Example 2: How much power do we have for a in a sample of 100,000 individuals for a SNP that explains 0.02% of the phenotypic variance, if we impose a significance level of 5x10^{-8}?

```
pwr.r.test(n = 100000, r = sqrt(0.0002), sig.level = 5e-8, power = NULL, a
lternative = "two.sided")

    approximate correlation power calculation (arctangh transformation)

              n = 1e+05
              r = 0.01414214
      sig.level = 5e-08
          power = 0.1637795
    alternative = two.sided
```

The results show that in this setting, we only have 16% power to detect the SNP. You'll see that large samples are needed in a GWAS.

Example 3: What is the minimum variance explained you could detect in a sample of 100,000 individuals at the genome-wide significance level of 5x10^{-8} with 80% power?

```
pwr.r.test(n = 100000, r = NULL, sig.level = 5e-8, power = 0.80, alternati
ve = "two.sided")

     approximate correlation power calculation (arctangh transformation)

              n = 1e+05
              r = 0.01989582
      sig.level = 5e-08
          power = 0.8
    alternative = two.sided
```

As output we get a standardized regression coefficient of 0.01989582. We have to square this number, and multiply the result with 100% to obtain what we want:

```
0.01989582*0.01989582*100
[1] 0.03958437
```

Hence, the minimum variance explained we can detect with 80% power in this setting is 0.04%. Thus, SNPs with a smaller effect are unlikely to be detected in this setting.

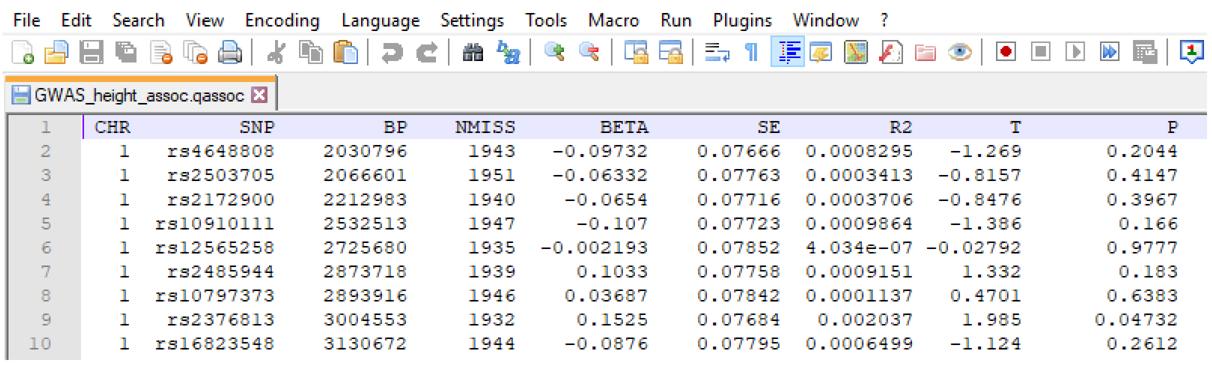
Running a GWAS in PLINK

Now that we can determine the conditions for a sufficiently powered GWAS, let's practice with running a GWAS. For this purpose, we switch to PLINK (see Tutorial 2 for how to run PLINK from the command line). In this tutorial, we make again use of the same example data as in Tutorial 2 and Tutorial 3. The files Example.bed, Example.bim, and Example.fam are probably still in your working directory ("C:\Users\Niels\Documents\EUR\E&G\Tutorials") If not, please put these files there again. These data files include information about 23,825 SNPS for 2,000 individuals (have a look at the BIM and FAM file to check this). Also, make sure that the file "Example_height.pheno" (Tutorial 2) is available in this folder.

In Tutorial 2, we used the --assoc command to perform a basic association analysis. The --linear flag (for continuously distributed traits) comes with more flexibility. Let's compare the two output files; let's run the --assoc command first:

```
plink --bfile Example --assoc --pheno Example_height.pheno --out GWAS_height_assoc
```

The output file "GWAS_height_assoc.qassoc" looks as follows (we already discussed the contents of this file in Tutorial 2):

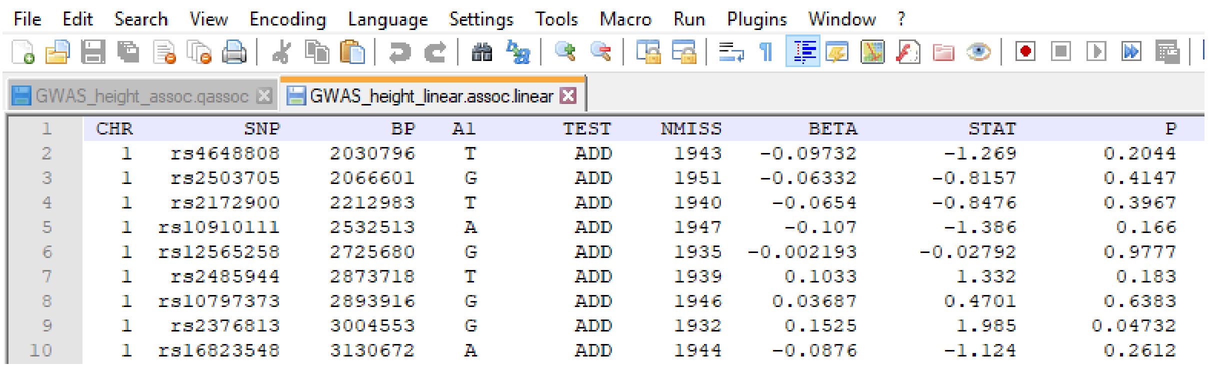

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
GWAS_height_assoc.qassoc

```
 CHR         SNP         BP    NMISS       BETA         SE         R2        T            P
   1   rs4648808    2030796     1943   -0.09732    0.07666  0.0008295   -1.269       0.2044
   1   rs2503705    2066601     1951   -0.06332    0.07763  0.0003413  -0.8157       0.4147
   1   rs2172900    2212983     1940    -0.0654    0.07716  0.0003706  -0.8476       0.3967
   1  rs10910111    2532513     1947     -0.107    0.07723  0.0009864   -1.386        0.166
   1  rs12565258    2725680     1935  -0.002193    0.07852  4.034e-07 -0.02792       0.9777
   1   rs2485944    2873718     1939     0.1033    0.07758  0.0009151    1.332        0.183
   1  rs10797373    2893916     1946    0.03687    0.07842  0.0001137   0.4701       0.6383
   1   rs2376813    3004553     1932     0.1525    0.07684   0.002037    1.985      0.04732
   1  rs16823548    3130672     1944    -0.0876    0.07795  0.0006499   -1.124       0.2612
```

Let's now run the GWAS with the --linear command:

```
plink --bfile Example --linear --pheno Example_height.pheno --out GWAS_height_linear
```

The resulting file "GWAS_height_linear.assoc.linear" looks as follows:


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
GWAS_height_assoc.qassoc
GWAS_height_linear.assoc.linear

```
 CHR         SNP         BP   A1       TEST    NMISS       BETA         STAT            P
   1   rs4648808    2030796    T        ADD     1943   -0.09732       -1.269       0.2044
   1   rs2503705    2066601    G        ADD     1951   -0.06332      -0.8157       0.4147
   1   rs2172900    2212983    T        ADD     1940    -0.0654      -0.8476       0.3967
   1  rs10910111    2532513    A        ADD     1947     -0.107       -1.386        0.166
   1  rs12565258    2725680    G        ADD     1935  -0.002193     -0.02792       0.9777
   1   rs2485944    2873718    T        ADD     1939     0.1033        1.332        0.183
   1  rs10797373    2893916    G        ADD     1946    0.03687       0.4701       0.6383
   1   rs2376813    3004553    G        ADD     1932     0.1525        1.985      0.04732
   1  rs16823548    3130672    A        ADD     1944    -0.0876       -1.124       0.2612
```

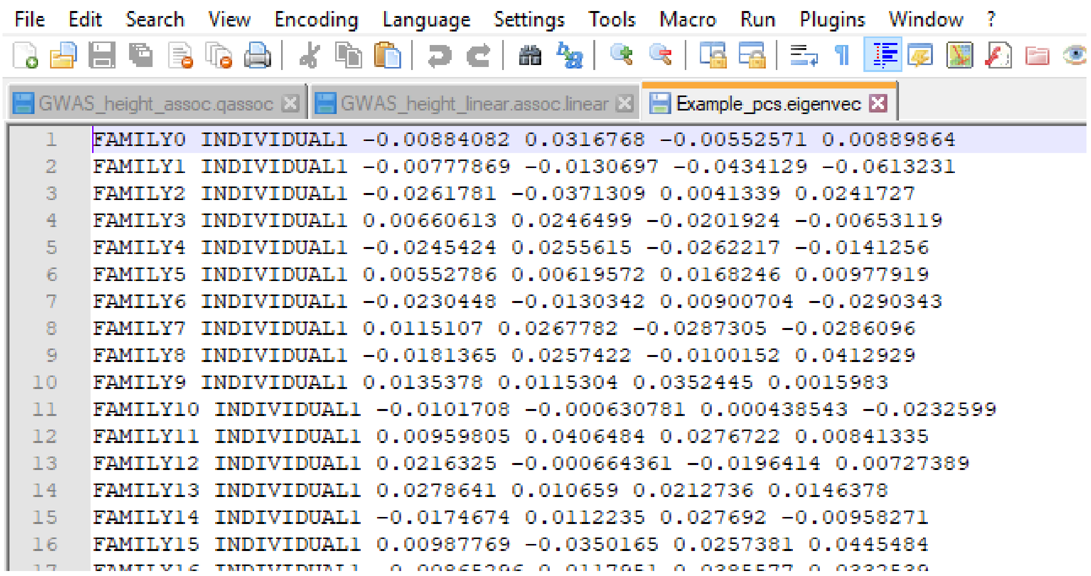
The columns in this file represent: Chromosome, SNP identifier, Basepair position, Reference allele, Type of model (here: additive), Number of non-missing individuals for this analysis, Regression coefficient, Standard error of the coefficient, *t*-statistic for regression of phenotype on allele count (number of reference alleles; 0, 1, or 2), Asymptotic significance value for coefficient. You can see that --assoc and --linear give precisely the same results, and that many columns in the output files are identical.

Principal components

As discussed in Lecture 4, an important advantage of GWAS analysis is the possibility to control for population stratification with principal components. Let's construct these principal components first, and then add them to the GWAS model as control variables. We can construct the principal components with the --pca command. Let's construct the first four principal components (because for many samples, analysts use four principal components to control subtle population stratification):

```
plink --bfile Example --pca 4 --out Example_pcs
```

The command above creates two files in your working directory: "Example_pcs.eigenval" and "Example_pcs.eigenvec". The first file contains the eigenvalues of the genetic relationship matrix. The second file contains the first four principal components:



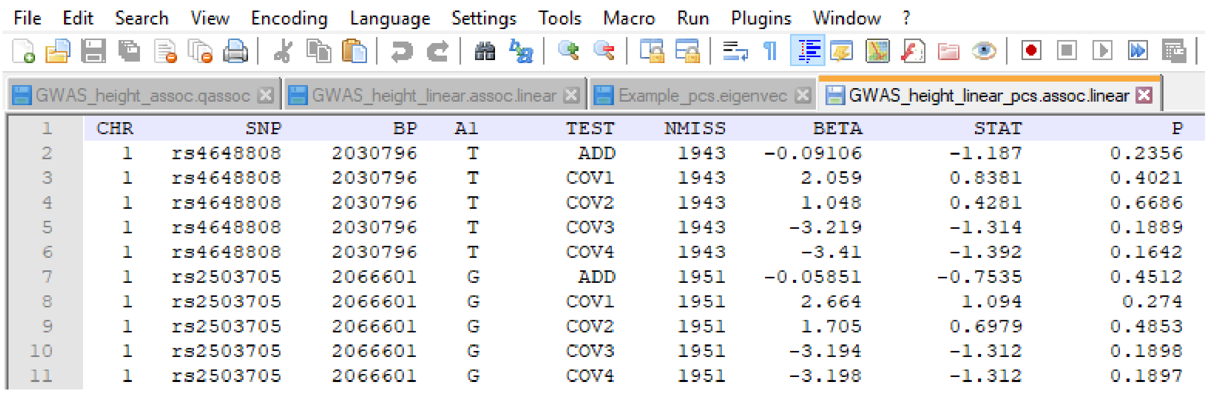
```
FAMILY0 INDIVIDUAL1 -0.00884082 0.0316768 -0.00552571 0.00889864
FAMILY1 INDIVIDUAL1 -0.00777869 -0.0130697 -0.0434129 -0.0613231
FAMILY2 INDIVIDUAL1 -0.0261781 -0.0371309 0.0041339 0.0241727
FAMILY3 INDIVIDUAL1 0.00660613 0.0246499 -0.0201924 -0.00653119
FAMILY4 INDIVIDUAL1 -0.0245424 0.0255615 -0.0262217 -0.0141256
FAMILY5 INDIVIDUAL1 0.00552786 0.00619572 0.0168246 0.00977919
FAMILY6 INDIVIDUAL1 -0.0230448 -0.0130342 0.00900704 -0.0290343
FAMILY7 INDIVIDUAL1 0.0115107 0.0267782 -0.0287305 -0.0286096
FAMILY8 INDIVIDUAL1 -0.0181365 0.0257422 -0.0100152 0.0412929
FAMILY9 INDIVIDUAL1 0.0135378 0.0115304 0.0352445 0.0015983
FAMILY10 INDIVIDUAL1 -0.0101708 -0.000630781 0.000438543 -0.0232599
FAMILY11 INDIVIDUAL1 0.00959805 0.0406484 0.0276722 0.00841335
FAMILY12 INDIVIDUAL1 0.0216325 -0.000664361 -0.0196414 0.00727389
FAMILY13 INDIVIDUAL1 0.0278641 0.010659 0.0212736 0.0146378
FAMILY14 INDIVIDUAL1 -0.0174674 0.0112235 0.027692 -0.00958271
FAMILY15 INDIVIDUAL1 0.00987769 -0.0350165 0.0257381 0.0445484
FAMILY16 INDIVIDUAL1 -0.00865296 0.0117951 0.0385577 0.0333539
```

Although a bit difficult to see, there are six columns in this file (separated by spaces): The family ID, the individual ID, and the first four principal components.

Controlling for principal components in a GWAS

In Tutorial 3, we excluded cryptically related individuals from the GREML analysis. In a GWAS, we use principal components to control for cryptic genetic relatedness. A nice feature of the --linear command is the possibility to include covariates in the model with the --covar option. Let's use the constructed principal components as control variables in our GWAS:

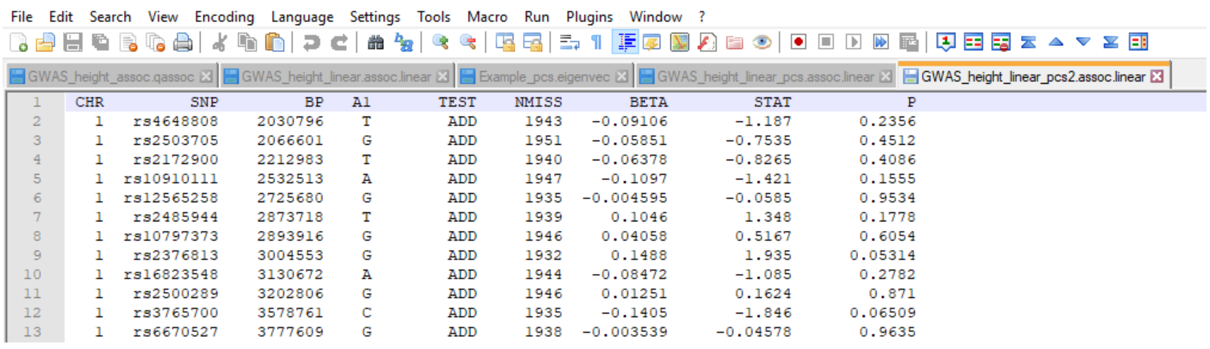
```
plink --bfile Example --linear --pheno Example_height.pheno --out
GWAS_height_linear_pcs --covar Example_pcs.eigenvec
```


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
GWAS_height_assoc.qassoc
GWAS_height_linear.assoc.linear
Example_pcs.eigenvec
GWAS_height_linear_pcs.assoc.linear

```
 CHR         SNP         BP   A1       TEST    NMISS       BETA         STAT            P
   1   rs4648808    2030796    T        ADD     1943   -0.09106       -1.187       0.2356
   1   rs4648808    2030796    T       COV1     1943      2.059       0.8381       0.4021
   1   rs4648808    2030796    T       COV2     1943      1.048       0.4281       0.6686
   1   rs4648808    2030796    T       COV3     1943     -3.219       -1.314       0.1889
   1   rs4648808    2030796    T       COV4     1943      -3.41       -1.392       0.1642
   1   rs2503705    2066601    G        ADD     1951   -0.05851      -0.7535       0.4512
   1   rs2503705    2066601    G       COV1     1951      2.664        1.094        0.274
   1   rs2503705    2066601    G       COV2     1951      1.705       0.6979       0.4853
   1   rs2503705    2066601    G       COV3     1951     -3.194       -1.312       0.1898
   1   rs2503705    2066601    G       COV4     1951     -3.198       -1.312       0.1897
```

For every SNP, PLINK reports the regression results for the control variables on a separate row. In practice, this is quite inconvenient, because you are often only interested in the regression results for the SNPs. If you add --hide-covar, only the regression results for the SNPs are included in the output file:

```
plink --bfile Example --linear --pheno Example_height.pheno --out GWAS_height_linear_pcs2 --covar Example_pcs.eigenvec --hide-covar
```


File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
GWAS_height_assoc.qassoc
GWAS_height_linear.assoc.linear
Example_pcs.eigenvec
GWAS_height_linear_pcs.assoc.linear
GWAS_height_linear_pcs2.assoc.linear

```
 CHR         SNP         BP   A1       TEST    NMISS       BETA         STAT            P
   1   rs4648808    2030796    T        ADD     1943   -0.09106       -1.187       0.2356
   1   rs2503705    2066601    G        ADD     1951   -0.05851      -0.7535       0.4512
   1   rs2172900    2212983    T        ADD     1940   -0.06378      -0.8265       0.4086
   1  rs10910111    2532513    A        ADD     1947    -0.1097       -1.421       0.1555
   1  rs12565258    2725680    G        ADD     1935  -0.004595      -0.0585       0.9534
   1   rs2485944    2873718    T        ADD     1939     0.1046        1.348       0.1778
   1  rs10797373    2893916    G        ADD     1946    0.04058       0.5167       0.6054
   1   rs2376813    3004553    G        ADD     1932     0.1488        1.935      0.05314
   1  rs16823548    3130672    A        ADD     1944   -0.08472       -1.085       0.2782
   1   rs2500289    3202806    G        ADD     1946    0.01251       0.1624        0.871
   1   rs3765700    3578761    C        ADD     1935    -0.1405       -1.846      0.06509
   1   rs6670527    3777609    G        ADD     1938  -0.003539     -0.04578       0.9635
```

- **Exercise:** Check that the SNP with the lowest *p*-value is rs1953405.

Visualizing GWAS results in *R*

We are now going to visualize the GWAS results using QQ-plots and Manhattan plots. For this purpose, we switch back to *R*. Launch *R* by starting *RStudio*. First, we need to read in our GWAS results in *R*. Therefore, we first change R's working directory to our folder with data and results:

```
setwd("C:/Users/Niels/Documents/EUR/E&G/Tutorials")
```

Subsequently, we read in the GWAS results (the model with PCs) using the "read.table" function. We store the results in the object "results".

```
results<-read.table("GWAS_height_linear_pcs2.assoc.linear",header=TRUE)
```

In the "Environment" you now see the object "results" under "Data". By clicking on the object, you can inspect it contents. Importantly, the *p*-value for each SNP is available in the column "P".

We will create both a Manhattan plot and a QQ-plot to visualize the GWAS results. In both plots, it is common work with logarithmically transformed *p*-values, -log10(*p*-value), rather than with the raw *p*-values. Let's do this transformation, and store the results in the column "observed_log_pvalues":

```
results$observed_log_pvalues <- (-log10(results$P))
```

Manhattan plot

Based on the –$\log_{10}$(*p*-values) of the GWAS results, we can create a Manhattan plot. In a Manhattan plot, you plot the (transformed) *p*-values according to their position in the genome. In the GWAS results, the chromosome number and basepair position of the SNP is available. The numbering of basepairs restarts every chromosome, so we first need to determine the plotting position of a SNP as a number reflecting its chromosome and its BP position on the chromosome.

First, we need to determine per chromosome the minimum and maximum Base Pair (BP) position:

```
for(i in 1:22) {
  results$MinBP[results$CHR==i]<-min(results$BP[results$CHR==i])
  results$MaxBP[results$CHR==i]<-max(results$BP[results$CHR==i])
}
```

Within a chromosome, we make sure that the plotting position start at 1. Therefore, we need to subtract the minimum BP in a chromosome from the SNP's BP position, and add 1 to it:

```
results$Plot_position<-results$BP - results$MinBP + 1
```

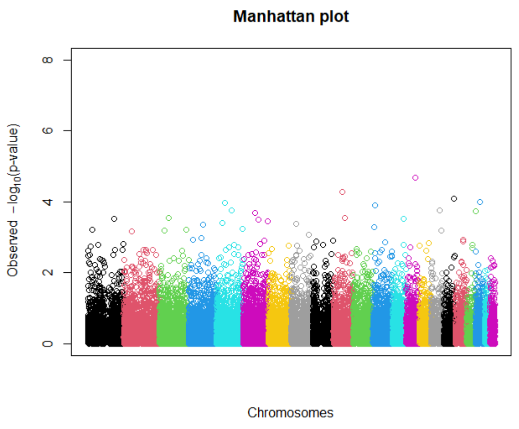
The only thing left to do now (to get a continuous plotting position), is to add the maximum plotting position from a SNP on the preceding chromosome to the plotting position of a SNP within a chromosome. For this purpose we could use a "loop", but for simplicity I spell out the commands for each chromosome here. We skip chromosome 1, because there is no preceding chromosome.

```
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==1])
results$Plot_position[results$CHR==2]<-results$Plot_position[results$CHR==2]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==2])
results$Plot_position[results$CHR==3]<-results$Plot_position[results$CHR==3]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==3])
results$Plot_position[results$CHR==4]<-results$Plot_position[results$CHR==4]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==4])
results$Plot_position[results$CHR==5]<-results$Plot_position[results$CHR==5]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==5])
results$Plot_position[results$CHR==6]<-results$Plot_position[results$CHR==6]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==6])
results$Plot_position[results$CHR==7]<-results$Plot_position[results$CHR==7]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==7])
results$Plot_position[results$CHR==8]<-results$Plot_position[results$CHR==8]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==8])
results$Plot_position[results$CHR==9]<-results$Plot_position[results$CHR==9]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==9])
results$Plot_position[results$CHR==10]<-results$Plot_position[results$CHR==10]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==10])
results$Plot_position[results$CHR==11]<-results$Plot_position[results$CHR==11]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==11])
results$Plot_position[results$CHR==12]<-results$Plot_position[results$CHR==12]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==12])
results$Plot_position[results$CHR==13]<-results$Plot_position[results$CHR==13]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==13])
results$Plot_position[results$CHR==14]<-results$Plot_position[results$CHR==14]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==14])
results$Plot_position[results$CHR==15]<-results$Plot_position[results$CHR==15]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==15])
results$Plot_position[results$CHR==16]<-results$Plot_position[results$CHR==16]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==16])
results$Plot_position[results$CHR==17]<-results$Plot_position[results$CHR==17]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==17])
results$Plot_position[results$CHR==18]<-results$Plot_position[results$CHR==18]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==18])
results$Plot_position[results$CHR==19]<-results$Plot_position[results$CHR==19]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==19])
results$Plot_position[results$CHR==20]<-results$Plot_position[results$CHR==20]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==20])
results$Plot_position[results$CHR==21]<-results$Plot_position[results$CHR==21]+MAX_position_preceding_chromosome
MAX_position_preceding_chromosome<-max(results$Plot_position[results$CHR==21])
results$Plot_position[results$CHR==22]<-results$Plot_position[results$CHR==22]+MAX_position_preceding_chromosome
```

With the transformed *p*-values and plotting positions calculated, we can create the Manhattan plot.[1] In doing, we are going to give every chromosome a different color:

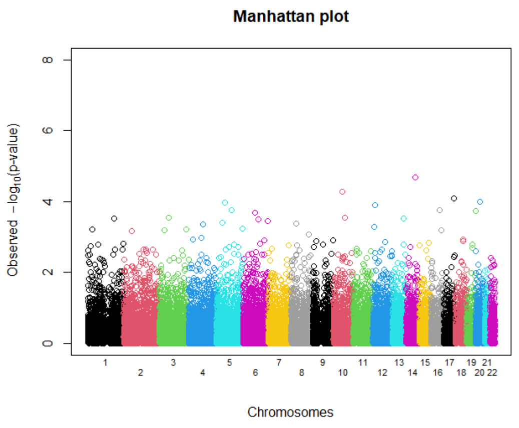
[1] Sometimes, you get the error message "Error in plot.new() : figure margins too large". This error occurs when your "Plots" panel is too small on your screen. You can easily solve this issue by making your "Plots" panel

```
plot(results$Plot_position,
results$observed_log_pvalues,col=results$CHR,xaxt="n",ylab=expression(paste(Observed~~-log[10],"(p-
value)")),main="Manhattan plot",ylim=c(0,8),xlab="Chromosomes")
```



In the resulting plot, all chromosomes have a different color. Let’s also put the chromosome numbers on the x-axis.

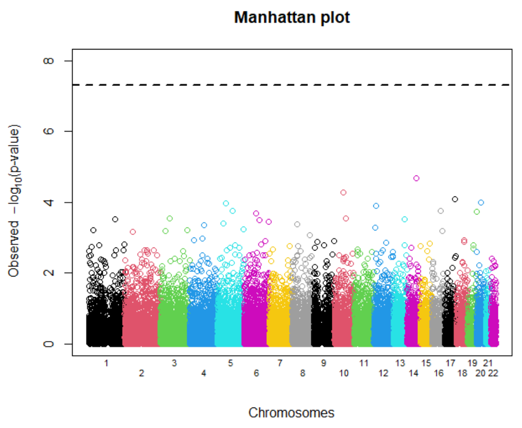
```
for(i in 1:22) {
  text(mean(results$Plot_position[results$CHR==i]),-0.50-0.30*((i %% 2) ==
0), adj=0.5,label=i,cex=0.7, xpd=TRUE)
}
```



The genome-wide significance level is 5×10^{-8} (Note that $-\log_{10}(5\times10^{-8})=7.3$). Let’s add this line to the Manhattan plot:

```
abline(h=-log10(5e-8),lwd="2",lty="dashed")
```

large, see e.g., https://support.rstudio.com/hc/en-us/articles/200488548-Problem-with-Plots-or-Graphics-Device.



The resulting Manhattan plot nicely presents the results of your GWAS. Unfortunately, you can see that there is no single SNP genome-wide significant in our analysis.

QQ-plot

As explained in Lecture 4, a Manhattan plot visualizes specific SNP associations. A QQ-plot is used to inspect the distribution of *p*-values. Under the null hypothesis of no association, the *p*-values in a GWAS are randomly distributed on the interval [0,1]. That is, they are uniformly distributed on the interval [0,1]. Let's "draw" values from this distribution for exactly as many SNPs we have in our results (length(results$P)), and transform them to the logarithmic scale. For this purpose, we create a series of numbers from 1 to the number of SNPs using "1:length(results$P)". We divide these numbers by length(results$P) to get to our distribution on the interval [0,1]:

```
null_log_pvalues <- -log10(1:length(results$P)/length(results$P))
```

To plot the observed and null distributions of *p*-values against each other, we first need to sort the *p*-values:

```
observed_log_pvalues_sorted <- sort(results$observed_log_pvalues, decreasing=TRUE)
```

```
null_log_pvalues_sorted <- sort(null_log_pvalues, decreasing=TRUE)
```

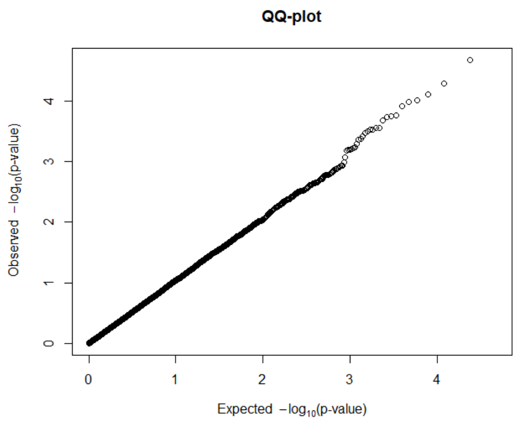
For the sorted *p*-values under the null distribution, we also construct the 95% confidence intervals because we want to add these to the QQ-plot. Fortunately, some researchers have derived that the *j*-th order statistic from a uniform (0,1) sample follows a beta(*j*,*n*-*j*+1) distribution. Hence, we can use the quantiles of this distribution to determine the upper (97.5% quantile) and lower bound (2.5% quantile) of the 95% confidence interval:

```
c975 <- rep(0, length(results$P))
c025 <- rep(0, length(results$P))
for(i in 1: length(results$P)){
  c975[i] <- qbeta(0.975,i, length(results$P)-i+1)
  c025[i] <- qbeta(0.025,i, length(results$P)-i+1)
}
```

In a QQ-plot, you plot the expected distribution of *p*-values against the observed distribution of *p*-values. We add some labels to the axes and figure itself as well. Moreover, we make sure to have a squared plotting area.

```
plot(null_log_pvalues_sorted, observed_log_pvalues_sorted,xlab=expression(paste(Expected~~-
log[10],"(p-value)")),ylab=expression(paste(Observed~~-log[10],"(p-value)")),main="QQ-
plot",xlim=c(0,max(observed_log_pvalues_sorted)),ylim=c(0,max(observed_log_pvalues_sorted)))
```

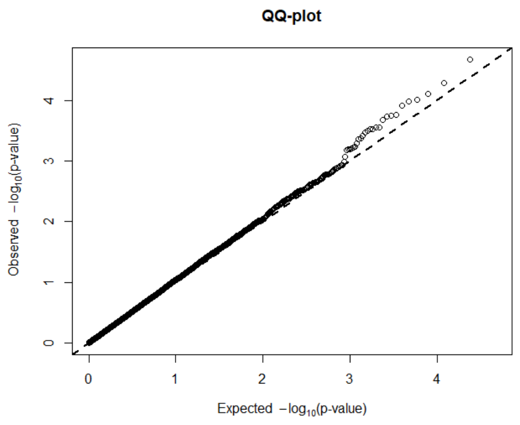
The resulting plot looks like:



With the following lines of code, we add a diagonal line to the plot. The diagonal represent the null hypothesis of no association. The command "par(new=T)" makes sure that the diagonal is added to the earlier created QQ-plot.

```
par(new=T)
abline(0,1,lwd="2",lty="dashed")
```

Your plot should like look this now:



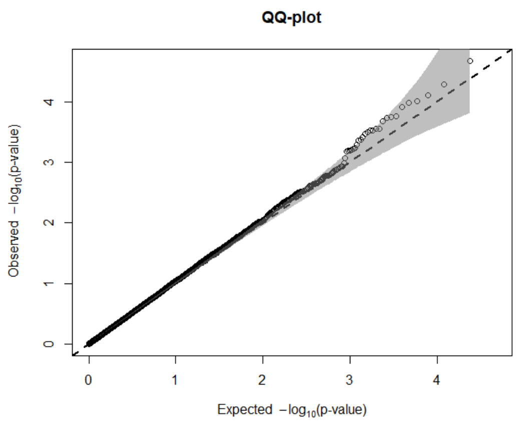
In the plot, you can see that there is some deviation from the diagonal. However, the deviation is small.

Finally, we add the confidence intervals to the QQ-plot:

```
polygon(x=c(null_log_pvalues_sorted,rev(null_log_pvalues_sorted)),y=c(-
log(c975,10),rev(null_log_pvalues_sorted)),col= rgb(0.5, 0.5, 0.5, 0.5),border=NA)

polygon(x=c(null_log_pvalues_sorted,rev(null_log_pvalues_sorted)),y=c(-
log(c025,10),rev(null_log_pvalues_sorted)),col= rgb(0.5, 0.5, 0.5, 0.5),border=NA)
```

Your final result should look like:



We see that all *p*-values are more or less within the 95% confidence interval (a few SNPs outside the 95% confidence interval is not surprising given the large number of SNPs we analyze). Moreover, there is no SNP with a *p*-value below the genome-wide significance level of 5×10^{-8} (i.e., above ~7.3 on the y-axis). Hence, we conclude that the distribution of *p*-values does not provide evidence for association between SNPs and height in our data.

Individual assignment[2]

- Use power calculations to assess under what statistical conditions (e.g., the expected R^2 of SNP) a GWAS on personal income is feasible. Use reasonable assumptions (e.g., based on earlier genetic associations reported in scientific papers) in your calculations, and use at least 1 scientifically formatted table to report the results.
- Run two times a GWAS on personal income, one for females and one for males, using the data from Tutorial 3. Use at least 2 scientifically formatted figures (QQ-plots and Manhattan plots) to report the results. Given your power calculations, discuss whether you are surprised by the GWAS results.

[2] Don't forget to include your analysis code in the Appendix of your report.