We thank you for your time spent taking this survey.
Your response has been recorded.

Below is a summary of your responses

Download PDF

# 2022 International Statistical Genetics Workshop
# Polygenic risk score tutorial

Today we'll be working through the some of the tasks involved in running a polygenic risk score analysis. You'll be working together as a group to help each other with the tasks.

In each task you will be asked to work out a solution to a problem and share your solution using this questionnaire. There is more than one solution to some of these tasks and you can use whatever method works best for you.
Please spend a couple of minutes introducing yourselves and choose one person from your group to be the scribe. This person will submit the groups solution(s) using this questionnaire.

Remember that you can share screen and also share code or instructions via chat.

As you're going through this tutorial we'll call you back into the main room to check in and take a break.

There are hints built into the questionnaire but if you need more help please click the 'Ask for help' button.

Today you will be using both the ssh server and Rstudio.

To get started ssh into the workshop server.

Use `mkdir` to make a working directory for today's tutorial and copy the files from `/faculty/sarah/2022/Day7/` to your working directory.

```
cp -r /faculty/sarah/2022/Day7/* .
```

In your Rstudio window set your working directory to be the directory you have just made.

Today's icebreaker question - what was the last TV series or movie you watched (or game you played or book you read)

What number zoom breakout room are you in?

What are the first names of your group members?

Has anyone in your group run a polygenic risk score analysis

○ Yes

○ No

In today's scenario you are a group of analysts working with the data from the twinData data set that you used last week.

To run these analyses you will work with both the terminal window and Rstudio. Code to be run in the terminal will be in `blue` and code for Rstudio will be in `black`.

You have been asked to replicate some results from another group that you collaborate with.

They are sceptical about the reliability of your height and weight data because they were collected with self report questionnaires.

To show them that the data are reliable they have asked you to run a polygenic risk score analysis to check that the self reported height data can be predicted by the GWAS of height from the UKBiobank.

They have asked you run the analysis using a program called PRSice (https://www.prsice.info/). PRSice is often used to run analyses to try and find the best predictor; however, as the aim of this analysis is to make sure that the data are behaving as expected your collaborators have asked you to run the prs at a series of set thresholds instead.

You agree to do this, to save some time you download a GWAS of height from the Neale lab

UKB GWAS (http://www.nealelab.is/uk-biobank) and start running the analyses...

The first step to running the analysis is to set up the files and make sure everything is in the right format.

The PRSice analyses require 6 files: 3 PLINK files, a phenotype file, a covariate file and a file containing weights you will use to build the polygenic risk score

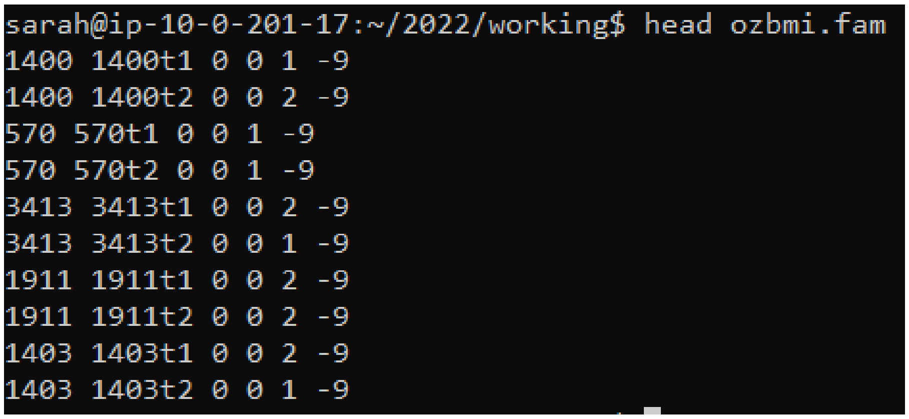
These files have already been created and are in the folder you have just copied, but we'll go through them anyway.

First, the 3 PLINK files: ozbmi.fam ozbmi.bim  ozbmi.bed

ozbmi.fam contains the pedigree information

You can take a look at ozbmi.fam by typing `head ozbmi.fam` into your ssh terminal

The .fam file has 6 columns: Family ID, Individual ID, Fathers ID (0=missing), Mothers ID (0=missing), Sex (1=M, 2=F), Phenotype (-9=missing)

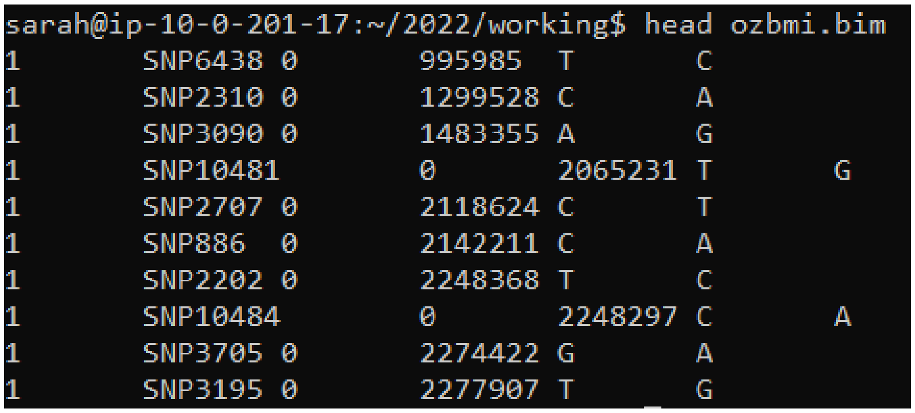


```
sarah@ip-10-0-201-17:~/2022/working$ head ozbmi.fam
1400 1400t1 0 0 1 -9
1400 1400t2 0 0 2 -9
570 570t1 0 0 1 -9
570 570t2 0 0 1 -9
3413 3413t1 0 0 2 -9
3413 3413t2 0 0 1 -9
1911 1911t1 0 0 2 -9
1911 1911t2 0 0 2 -9
1403 1403t1 0 0 2 -9
1403 1403t2 0 0 1 -9
```

ozbmi.bim contains the information about the genetic variants that have been genotyped

You can take a look at ozbmi.bim by typing `head ozbmi.bim` into your ssh terminal

The .bim file has 6 columns: Chromosome Number, Variant Name, cM location (set to 0 as it is not needed), base pair location, Allele 1, Allele2

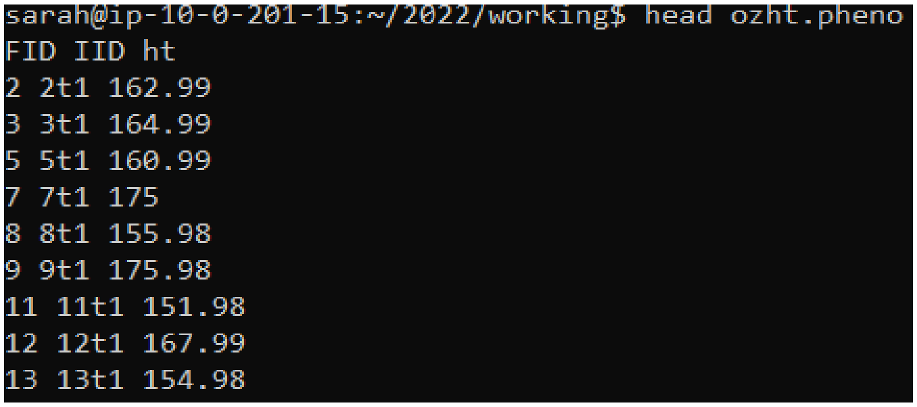


```
sarah@ip-10-0-201-17:~/2022/working$ head ozbmi.bim
1       SNP6438 0       995985  T       C
1       SNP2310 0       1299528 C       A
1       SNP3090 0       1483355 A       G
1       SNP10481        0       2065231 T       G
1       SNP2707 0       2118624 C       T
1       SNP886  0       2142211 C       A
1       SNP2202 0       2248368 T       C
1       SNP10484        0       2248297 C       A
1       SNP3705 0       2274422 G       A
1       SNP3195 0       2277907 T       G
```

ozbmi.bed contains the genotype data. It is a binary file and you can't look at it using `head`.

Next we'll take a look at the phenotype file by typing `head ozht.pheno`

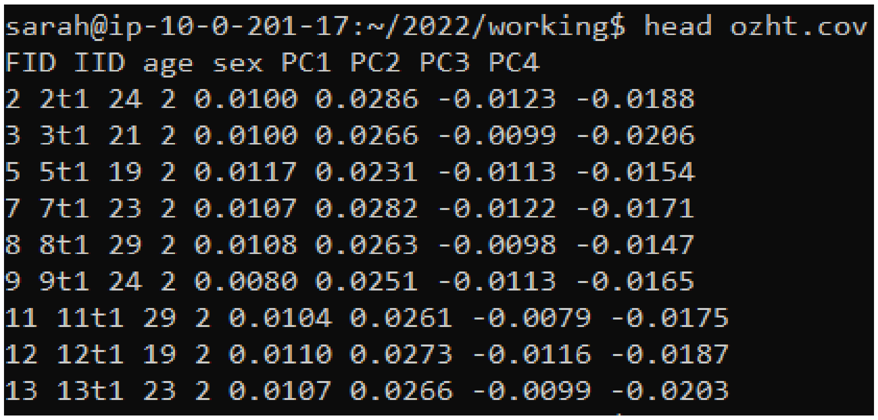
This file contains 3 columns: Family ID, Individual ID, Height (in cm)



```
sarah@ip-10-0-201-15:~/2022/working$ head ozht.pheno
FID IID ht
2 2t1 162.99
3 3t1 164.99
5 5t1 160.99
7 7t1 175
8 8t1 155.98
9 9t1 175.98
11 11t1 151.98
12 12t1 167.99
13 13t1 154.98
```

You can take a look at the covariate file using by typing `head ozht.cov`

This file contains 8 columns: Family ID, Individual ID, Age, Sex (1=M,2=F), and 4 genetic principle components which we will use to account for the effects of ethnicity in our analyses

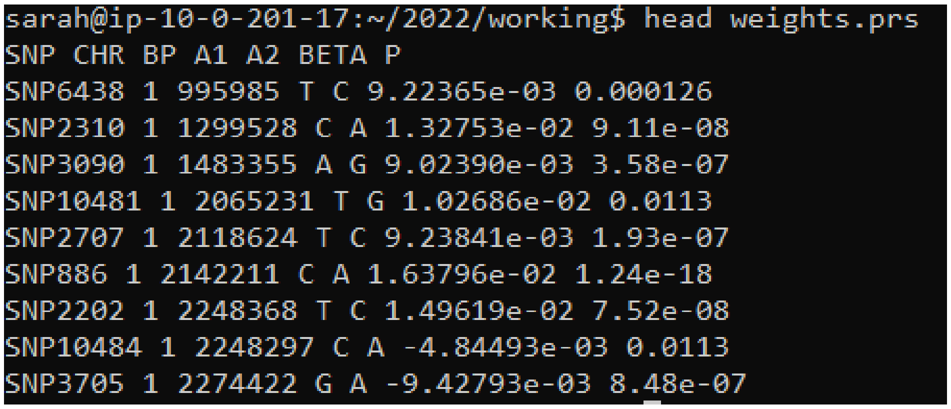


```
sarah@ip-10-0-201-17:~/2022/working$ head ozht.cov
FID IID age sex PC1 PC2 PC3 PC4
2 2t1 24 2 0.0100 0.0286 -0.0123 -0.0188
3 3t1 21 2 0.0100 0.0266 -0.0099 -0.0206
5 5t1 19 2 0.0117 0.0231 -0.0113 -0.0154
7 7t1 23 2 0.0107 0.0282 -0.0122 -0.0171
8 8t1 29 2 0.0108 0.0263 -0.0098 -0.0147
9 9t1 24 2 0.0080 0.0251 -0.0113 -0.0165
11 11t1 29 2 0.0104 0.0261 -0.0079 -0.0175
12 12t1 19 2 0.0110 0.0273 -0.0116 -0.0187
13 13t1 23 2 0.0107 0.0266 -0.0099 -0.0203
```

The last file contains a weights which you will use to build the polygenic risk score

You can take a look at the covariate file using by typing `head weights.prs`

This file contains 7 columns: Variant Name, Chromosome Number, base pair location, Allele 1, Allele 2, Beta (from the GWAS in the UK Biobank) and P-values (from the GWAS in the UK Biobank)



```
sarah@ip-10-0-201-17:~/2022/working$ head weights.prs
SNP CHR BP A1 A2 BETA P
SNP6438 1 995985 T C 9.22365e-03 0.000126
SNP2310 1 1299528 C A 1.32753e-02 9.11e-08
SNP3090 1 1483355 A G 9.02390e-03 3.58e-07
SNP10481 1 2065231 T G 1.02686e-02 0.0113
SNP2707 1 2118624 T C 9.23841e-03 1.93e-07
SNP886 1 2142211 C A 1.63796e-02 1.24e-18
SNP2202 1 2248368 T C 1.49619e-02 7.52e-08
SNP10484 1 2248297 C A -4.84493e-03 0.0113
SNP3705 1 2274422 G A -9.42793e-03 8.48e-07
```

Now that we have checked the files we are ready to run our analyses.

The PRSice analysis will go through the following steps

- The data in the `weights.prs` file will be clumped. Across the genome there is a high amount of autocorrelation (called Linkage disequilibrium or LD). When calculating the PRS we want to include weights from a set of independent variants. The clumping process allows us to do this by keeping the strongest result from the GWAS analysis for each set of correlated variants.

- The polygenic risk scores (PRS) will then be calculated
- The predictive ability of each PRS would usually then be examined by including it in a linear regression analysis to try and work out the best threshold - we will not be doing this
- The results of these analyses would usually then be plotted - we will not be doing this

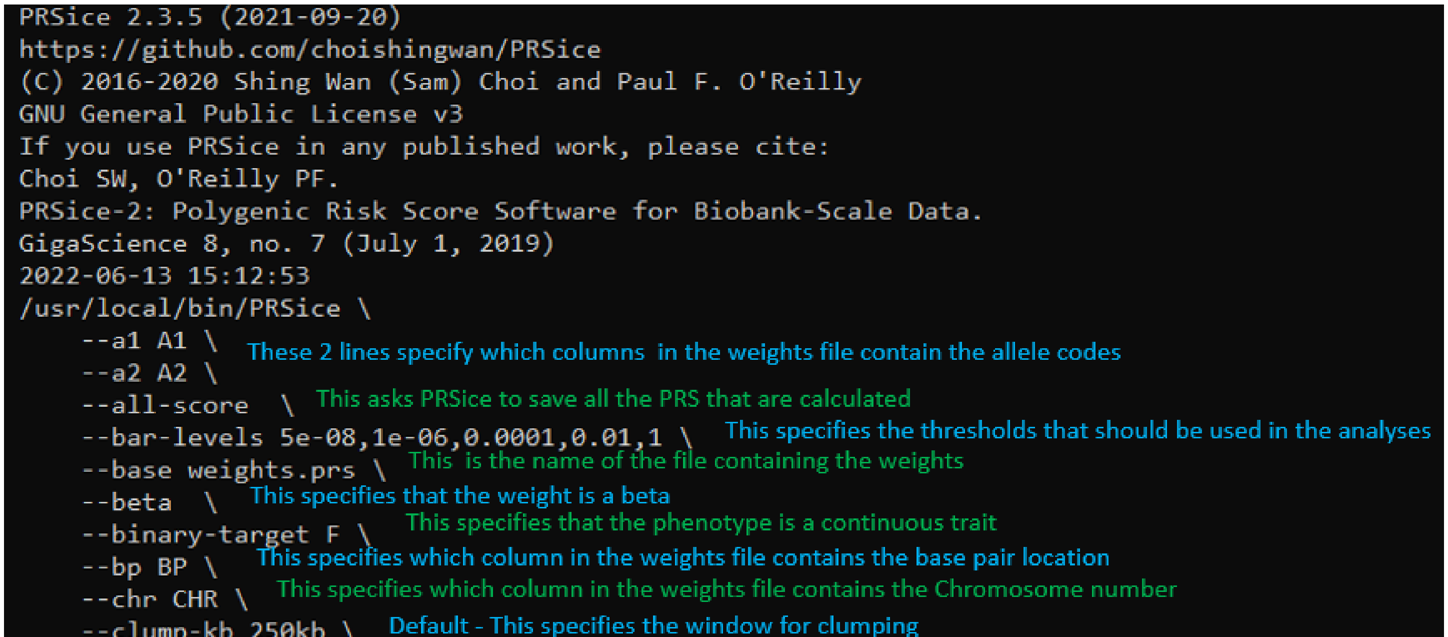
Copy the following code and paste it into the terminal window

```
Rscript /usr/local/bin/PRSice.R --dir . --prsice
/usr/local/bin/PRSice  --base weights.prs --target ozbmi  --thread 1
--snp SNP --chr CHR --bp BP --A1 A1 --A2 A2 --stat BETA --pvalue P  -
-beta --pheno ozht.pheno --score sum --all-score --bar-levels
0.00000005,0.000001,0.0001,0.01,1  --fastscore --quantile 5 --quant-
ref 1 --binary-target F --no-regress
```

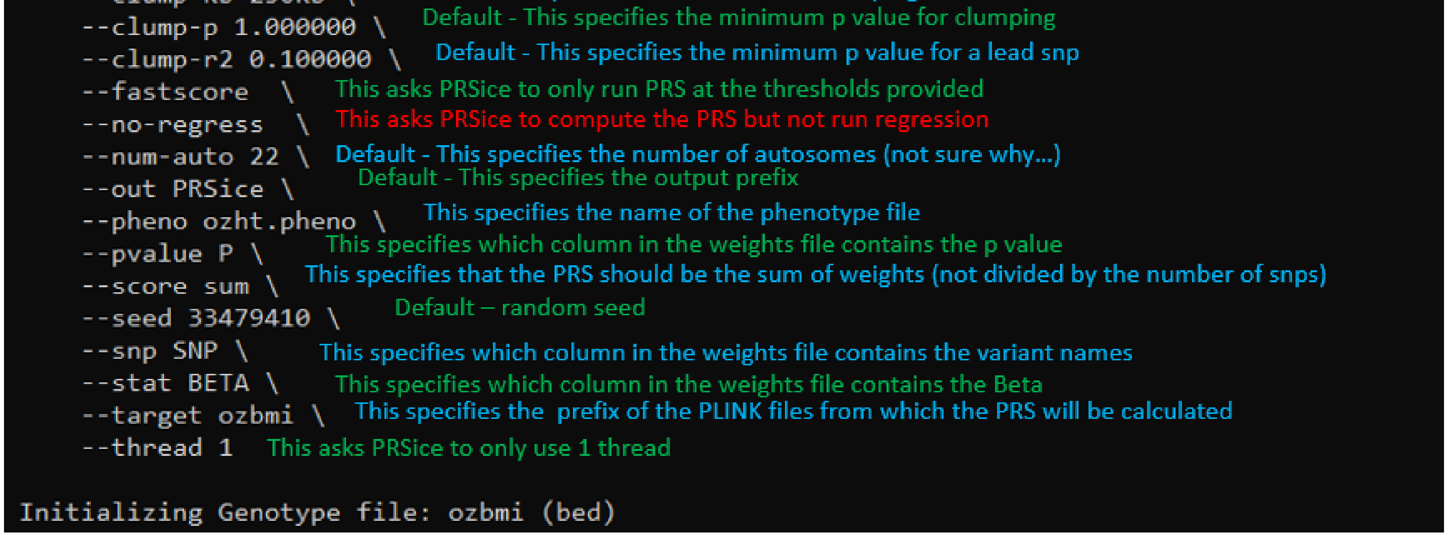
PRSice will run and print a lot of information on the screen - this information will also be printed to the `PRSice.log` file and we will look at it step by step...

hint: If you are having trouble copying and pasting you can find the code we are using today in `2022day7.txt`

The first section of the output reports back all the options that we included in our PRSice analysis and let us know what the defaults were for options we did not specify



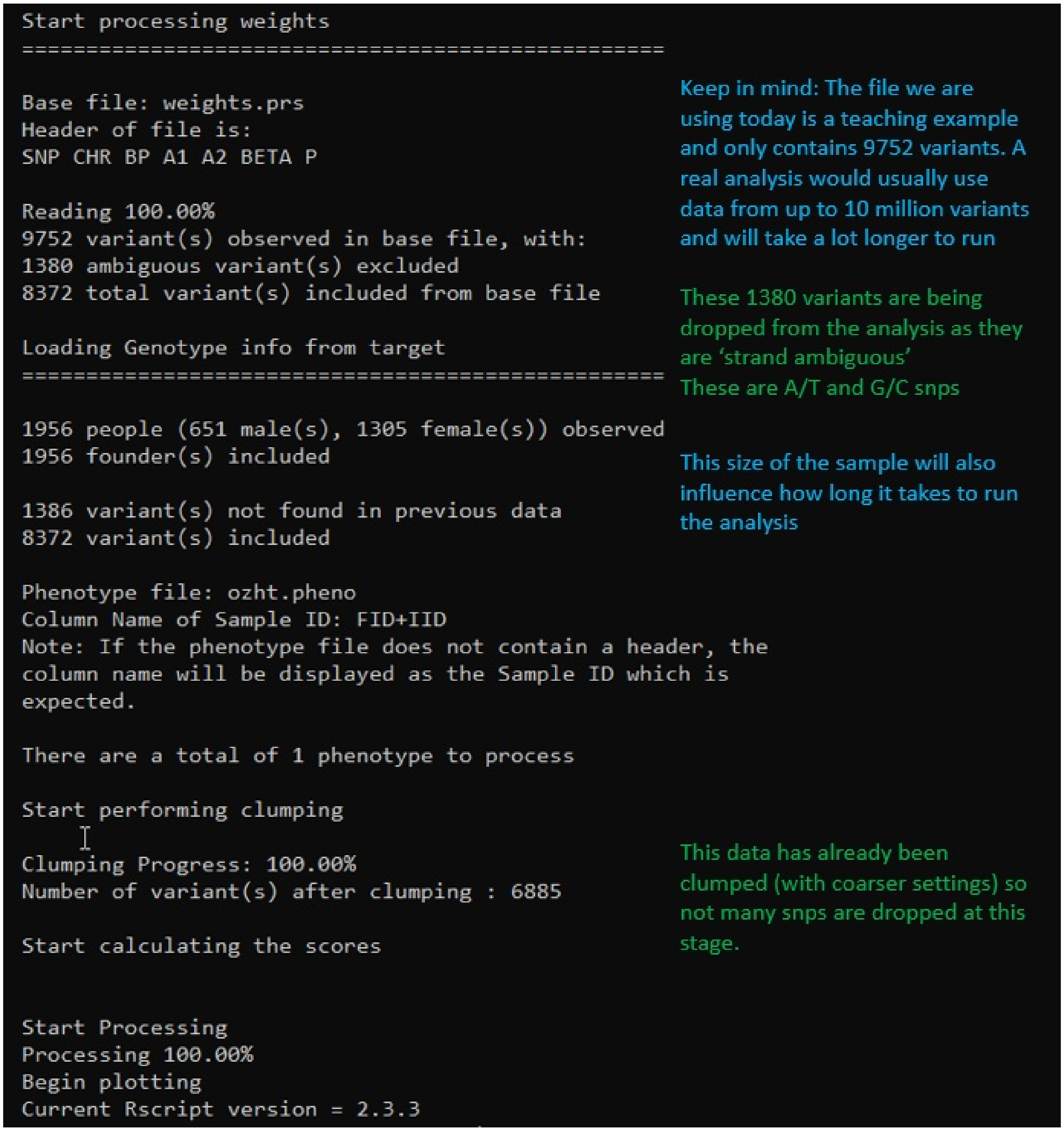
```
PRSice 2.3.5 (2021-09-20)
https://github.com/choishingwan/PRSice
(C) 2016-2020 Shing Wan (Sam) Choi and Paul F. O'Reilly
GNU General Public License v3
If you use PRSice in any published work, please cite:
Choi SW, O'Reilly PF.
PRSice-2: Polygenic Risk Score Software for Biobank-Scale Data.
GigaScience 8, no. 7 (July 1, 2019)
2022-06-13 15:12:53
/usr/local/bin/PRSice \
    --a1 A1 \     These 2 lines specify which columns  in the weights file contain the allele codes
    --a2 A2 \
    --all-score  \   This asks PRSice to save all the PRS that are calculated
    --bar-levels 5e-08,1e-06,0.0001,0.01,1 \   This specifies the thresholds that should be used in the analyses
    --base weights.prs \   This  is the name of the file containing the weights
    --beta  \   This specifies that the weight is a beta
    --binary-target F \   This specifies that the phenotype is a continuous trait
    --bp BP \   This specifies which column in the weights file contains the base pair location
    --chr CHR \   This specifies which column in the weights file contains the Chromosome number
    --clump-kb 250kb \   Default - This specifies the window for clumping
```



```
    --clump-p 1.000000 \     Default - This specifies the minimum p value for clumping
    --clump-r2 0.100000 \     Default - This specifies the minimum p value for a lead snp
    --fastscore  \   This asks PRSice to only run PRS at the thresholds provided
    --no-regress  \  This asks PRSice to compute the PRS but not run regression
    --num-auto 22 \  Default - This specifies the number of autosomes (not sure why...)
    --out PRSice \      Default - This specifies the output prefix
    --pheno ozht.pheno \    This specifies the name of the phenotype file
    --pvalue P \     This specifies which column in the weights file contains the p value
    --score sum \  This specifies that the PRS should be the sum of weights (not divided by the number of snps)
    --seed 33479410 \    Default – random seed
    --snp SNP \     This specifies which column in the weights file contains the variant names
    --stat BETA \   This specifies which column in the weights file contains the Beta
    --target ozbmi \  This specifies the  prefix of the PLINK files from which the PRS will be calculated
    --thread 1   This asks PRSice to only use 1 thread

Initializing Genotype file: ozbmi (bed)
```

The next section of reports the process of clumping and PRS calculation



```
Start processing weights
==================================================

Base file: weights.prs
Header of file is:
SNP CHR BP A1 A2 BETA P

Reading 100.00%
9752 variant(s) observed in base file, with:
1380 ambiguous variant(s) excluded
8372 total variant(s) included from base file

Loading Genotype info from target
==================================================

1956 people (651 male(s), 1305 female(s)) observed
1956 founder(s) included

1386 variant(s) not found in previous data
8372 variant(s) included

Phenotype file: ozht.pheno
Column Name of Sample ID: FID+IID
Note: If the phenotype file does not contain a header, the
column name will be displayed as the Sample ID which is
expected.

There are a total of 1 phenotype to process

Start performing clumping

Clumping Progress: 100.00%
Number of variant(s) after clumping : 6885

Start calculating the scores


Start Processing
Processing 100.00%
Begin plotting
Current Rscript version = 2.3.3
```

Keep in mind: The file we are using today is a teaching example and only contains 9752 variants. A real analysis would usually use data from up to 10 million variants and will take a lot longer to run

These 1380 variants are being dropped from the analysis as they are 'strand ambiguous'
These are A/T and G/C snps

This size of the sample will also influence how long it takes to run the analysis

This data has already been clumped (with coarser settings) so not many snps are dropped at this stage.

The PRSice analysis made 3 files

- PRSice.all_score - this contains the PRS that were calculated for each individual. Take a look at this file using `head`.
- PRSice.prsice - this contains the output from the regression analyses for each PRS. IGNORE THIS FILE
- PRSice.log - this contains a copy of the output that was printed to the screen

To see how well the PRS predicts height we will read the data into R and run a series of regression models
For this section of the tutorial we are going use Rstudio and you can either copy the code from qualtrics and paste into R or open the `2022Day7.txt` file and run the code from there

We will start by reading the phenotype, covariate and PRS files into R.
# Read in the phenotype covariance and prs files
```
pheno<-read.table("ozht.pheno", header=T)
covar<-read.table("ozht.cov", header=T)
PRS<-read.table("PRSice.all_score", header=T)
```

#lets give the PRS variables some more user friendly names
```
head(PRS)
names(PRS)<-c("FID","IID","PRS1","PRS2","PRS3","PRS4","PRS5")
head(PRS)
```

#and convert the height to cm
```
pheno$ht<-pheno$ht*100
```

We will then merrge the covar and PRS files using the individual ID "IID" as the key.
# Merge the pheno covar and PRS files
# we are using [ ] to subset the columns so that the FID is not duplicated in the merged files
```
temp=merge(pheno, covar[,2:8], by="IID")
longData=merge(temp, PRS[,2:7], by="IID")
```

#Take a look at the result to check that it worked
```
head(longData)
```

#Next we are going to run a Null model that includes the covariates but it doesn't include any PRS
```
nullModel<-lm(ht~age+sex+PC1+PC2+PC3+PC4, data=longData)
summary(nullModel)
```

#Then we are going to run a model that includes the covariates and the first PRS

```
thresh1<-lm(ht~age+sex+PC1+PC2+PC3+PC4+PRS1, data=longData)
summary(thresh1)
```

#We can compute the difference in R2 between the 2 models using the following code

```
summary(thresh1)$r.squared-summary(nullModel)$r.squared
```

Now working together as a group run the a series of regression analyses to predict height from each of the other PRS and record the results of your analyses in the table below

If you need help running the regression analyses take a look at this website https://www.statmethods.net/stats/regression.html

| | Beta | SE | P | R2 |
|---|---|---|---|---|
| PRS1 | | | | |
| PRS2 | | | | |
| PRS3 | | | | |
| PRS4 | | | | |
| PRS5 | | | | |

Do you think the PRS predict height well?

You put the results in an email and send it off to your collaborators.
Just after you hit send you have a sinking feeling as you realize that PRSice does not correct for relatedness and these analyses have assumed that your participants are independent of each other.
Do you...

- ○ A - Ignore this fact, after all the collaborators asked you to do this
- ○ B - Send one of those annoying recall emails in a futile attempt to turn back time
- ○ C - Email your collaborators and let them know you will rerun the analyses correcting for relatedness

I'm going to pretend you didn't say that and that you had picked option C instead.

Your sample contains MZ and DZ twins and we can rerun these analyses in openMx to correct for relatedness but we need to add the zygosity information to longData

# Read in the zygosity information
```
zyg<-read.table("ozht.zyg", header=T)
```

#We will then merge on the zyg data using the individual ID "IID" as the key.
```
longData1=merge(longData, zyg[,2:5], by="IID")
```

#Take a look at the result to check that it worked
```
head(longData1)
```

#For openMx we need our data to be in a *wide* format with one line per family so we will need to reshape the data.
#In this code the variables listed in v.names are present for both twins
#The variable Twin specifies if the person is twin 1 or 2
```
wideData=reshape(data = longData1, direction = "wide", v.names =
c("IID", "ht", "age", "sex", "PC1", "PC2", "PC3", "PC4", "PRS1",
"PRS2", "PRS3", "PRS4", "PRS5"), idvar = "FID", timevar = "Twin",
sep="_" )
```

# We will replace all missingness in the definition variables with 0
# the people with missing definition variables don't have phenotypic data so this will not bais estimates
```
wideData[,c(6:12,19:25)][is.na(wideData[,c(6:12,19:25)])] <- 0
```

We are now ready to analyse these data in openMx but before we do, let think about what we expect.

Do you think the pvalue will...

- ○ Stay the same
- ○ Get bigger (closer to 1)
- ○ Get smaller (closer to 0)

Do you think the beta will...

- ○ Stay about the same
- ○ Get bigger
- ○ Get smaller

Do you think the standard error will...

- ○ Stay about the same
- ○ Get bigger
- ○ Get smaller

Next we are going to run the model in openMx to correct for relatedness
You can find the openMx code in the `2022Day7.txt` file
We'll walk through some sections of the code to make sure we know what is going on

At the start of the code we'll define the variables and start values so we can pull these in later

```
# Load Libraries & Options
library(OpenMx)
mxOption( NULL, "Default optimizer", "NPSOL" )
source("miFunctions5272022.R")

# Select Variables for Analysis
vars <- 'ht' # list of variables names
nv <- 1 # number of variables
ntv <- nv*2 # number of total variables
depVar <- c("ht_1","ht_2") # dependent variables
indVar <- c("age_1", "sex_1", "PC1_1", "PC2_1", "PC3_1", "PC4_1",
"PRS1_1",
"age_2", "sex_2", "PC1_2", "PC2_2", "PC3_2", "PC4_2", "PRS1_2") #
independent variables or predictors
selVars <- c(depVar,indVar)

# Select Data for Analysis
mzData <- subset(wideData, MZDZ==1, selVars)
dzData <- subset(wideData, MZDZ==2, selVars)

# Set Starting Values
```

```
svMe <- 170 # start value for means
svVa <- 100 # start value for variance
lbVa <- 0.00001 # lower bound for variance
svBe <- 0.0001 # start value for means

# Create Data Objects for Multiple Groups
dataMZ <- mxData( observed=mzData, type="raw" )
dataDZ <- mxData( observed=dzData, type="raw" )
```

Next we'll set up the objects to hold the definition or independent variables and the betas for the regression

```
# Objects to hold definition variables for the regression
dPC1 <- mxMatrix( type="Full", nrow=1, ncol=2, free=FALSE,
labels=c("data.PC1_1","data.PC1_2"), name="dPC1" )
dPC2 <- mxMatrix( type="Full", nrow=1, ncol=2, free=FALSE,
labels=c("data.PC2_1","data.PC2_2"), name="dPC2" )
dPC3 <- mxMatrix( type="Full", nrow=1, ncol=2, free=FALSE,
labels=c("data.PC3_1","data.PC3_2"), name="dPC3" )
dPC4 <- mxMatrix( type="Full", nrow=1, ncol=2, free=FALSE,
labels=c("data.PC4_1","data.PC4_2"), name="dPC4" )
dsex <- mxMatrix( type="Full", nrow=1, ncol=2, free=FALSE,
labels=c("data.sex_1","data.sex_2"), name="dsex" )
dage <- mxMatrix( type="Full", nrow=1, ncol=2, free=FALSE,
labels=c("data.age_1","data.age_2"), name="dage" )
dPRS <- mxMatrix( type="Full", nrow=1, ncol=2, free=FALSE,
labels=c("data.PRS1_1","data.PRS1_2"), name="dPRS" )

# Objects to hold betas for the covariates
bPC1 <- mxMatrix( type="Full", nrow=1, ncol=1, free=TRUE,
values=svBe, label="b1", name="bPC1" )
bPC2 <- mxMatrix( type="Full", nrow=1, ncol=1, free=TRUE,
values=svBe, label="b2", name="bPC2" )
bPC3 <- mxMatrix( type="Full", nrow=1, ncol=1, free=TRUE,
values=svBe, label="b3", name="bPC3" )
bPC4 <- mxMatrix( type="Full", nrow=1, ncol=1, free=TRUE,
values=svBe, label="b4", name="bPC4" )
bsex <- mxMatrix( type="Full", nrow=1, ncol=1, free=TRUE,
values=svBe, label="b5", name="bsex" )
bage <- mxMatrix( type="Full", nrow=1, ncol=1, free=TRUE,
values=svBe, label="b6", name="bage" )
bPRS <- mxMatrix( type="Full", nrow=1, ncol=1, free=TRUE,
values=svBe, label="b7", name="bPRS" )
```

Then we will build the model for the means that includes the regression tems
(we are using Kronecker products here to mutliply the values of the independent variables by the betas)

```
meanG <- mxMatrix( type="Full", nrow=1, ncol=1, free=TRUE,
values=svMe, labels=labVars("mean",vars), name="meanG" )
ExpMean <- mxAlgebra( expression= (meanG + bPC1%x%dPC1 + bPC2%x%dPC2
+ bPC3%x%dPC3 + bPC4%x%dPC4 + bsex%x%dsex + bage%x%dage +
bPRS%x%dPRS) , name="expMean" )
```

and do some house keeping (making a list of the objects so we can pull these into the model later)

```
pars<- c(bPC1, bPC2, bPC3, bPC4, bsex, bage, bPRS, meanG)
defs<- c(dPC1, dPC2, dPC3, dPC4, dsex, dage, dPRS)
```

The covariance model in this script is very simple as we are only trying to correct for relatedness
We are specifying 1 variance for all individuals, an MZ covariance and a DZ covariance

```
# Create Algebra for expected Variance/Covariance Matrices
covMZ <- mxMatrix( type="Symm", nrow=ntv, ncol=ntv, free=TRUE,
values=c(.1,.05,.1),
lbound=valDiag(lbVa,ntv),labels=c("var","cMZ21","var"), name="covMZ"
)
covDZ <- mxMatrix( type="Symm", nrow=ntv, ncol=ntv, free=TRUE,
values=c(.1,.05,.1), lbound=valDiag(lbVa,ntv),
labels=c("var","cDZ21","var"), name="covDZ" )
```

Next we build the expection and model objects

```
# Create Expectation Objects for Multiple Groups
expMZ <- mxExpectationNormal( covariance="covMZ", means="expMean",
dimnames=depVar )
expDZ <- mxExpectationNormal( covariance="covDZ", means="expMean",
dimnames=depVar )
funML <- mxFitFunctionML()
```

```
# Create Model Objects for Multiple Groups
modelMZ <- mxModel( pars, defs, ExpMean, covMZ, dataMZ, expMZ, funML,
name="MZ" )
modelDZ <- mxModel( pars, defs, ExpMean, covDZ, dataDZ, expDZ, funML,
name="DZ" )
multi <- mxFitFunctionMultigroup( c("MZ","DZ") )

# Build Saturated Model
model_wPRS <- mxModel( "wPRS", pars, modelMZ, modelDZ, multi )
```

We run the model

```
# RUN MODEL
# Run Saturated Model
fit_wPRS <- mxRun( model_wPRS )
(sum_wPRS <- summary( fit_wPRS ))
```

And then run a submodel whic drops the PRS effect from the model

```
# Drop the PRS from the model (running the null model)
model_Null <- mxModel( fit_wPRS, name="noPRS" )
model_Null <- omxSetParameters( model_Null, labels="b7", free=FALSE,
values=0 )
fit_Null <- mxRun( model_Null)
(sum_Null <- summary( fit_Null ))
```

We can work out the significance of the PRS effect by comparing the fit of the models

```
mxCompare( fit_wPRS, fit_Null )
```

and calculate r2 as the difference in variance between the models gives variance explained by PRS, divided by total variance

```
(fit_Null$MZ$covMZ$values[1,2]-
fit_wPRS$MZ$covMZ$values[1,2])/fit_Null$MZ$covMZ$values[1,2]
```

Now working together as a group run the a series of openMx analyses to predict height from each of the other PRS and record the results of your analyses in the table below

|  | Beta | SE | P | r2 |
|---|---|---|---|---|
| PRS1 |  |  |  |  |
| PRS2 |  |  |  |  |
| PRS3 |  |  |  |  |

| | | | | |
|---|---|---|---|---|
| PRS4 | Beta | SE | P | r2 |
| PRS5 | | | | |

● **I'm stuck please give me a hint**

You will need to update the name of the PRS variable being included in the model in two places within the script
1. In the list of variables being included in the model
2. In the matrix that holds the PRS definition variable

How do the openMx results (correcting for relatedness) compare to those from the lm regression (that assume independence)?

Consider your results from PRS2.

| | **Beta** | **SE** | **P** | **r2** |
|---|---|---|---|---|
| **lm** | | | | |
| **openMx** | | | | |

You have just finished running all the analyses and are ready to send the results off to the collaborators when your colleague asks how your openMx analysis accounts for the fact that some of your twins might be related (ie twins in family 1 are cousins of twins in famly 2).

By this stage you are very frustrated and you decide to reanalyse the data again this time using a method that doesn't require you to know about the relatedness between participants before you run the analyses.

After doing some searching online you decide to use a program called gcta (https://yanglab.westlake.edu.cn/software/gcta/#Overview)

gcta is usually used to estimate heritability in unrelated individuals BUT if you run this in a sample that includes close relatives it produces heritability estimates that match a twin/family AE model.
Today we are going to exploit this capability to use gcta to run regressions that correct for relatedness. The advantage of using gcta in this case is that as we will be calculating the relatedness from genotype data we do not need to know family structure before we the

relatedness from genotype data we do not need to know family structure before we the analyses.

As a reminder code to be run in the terminal will be in `blue` and code for Rstudio will be in `black`.

Our first step is to calculate a genetic relatedness matrix or GRM
# Make GRM using GCTA

```
gcta64 --bfile ozbmi --make-grm --out ozGRM
```

We can take a look at the contents of the GRM using the following code

```
#  Open GRM and visualise off-diagonal distribution
ReadGRMBin=function(prefix, AllN=F, size=4){
  sum_i=function(i){
    return(sum(1:i))
  }
  BinFileName=paste(prefix,".grm.bin",sep="")
  NFileName=paste(prefix,".grm.N.bin",sep="")
  IDFileName=paste(prefix,".grm.id",sep="")
  id = read.table(IDFileName)
  n=dim(id)[1]
  BinFile=file(BinFileName, "rb");
  grm=readBin(BinFile, n=n*(n+1)/2, what=numeric(0), size=size)
  NFile=file(NFileName, "rb");
  if(AllN==T){
    N=readBin(NFile, n=n*(n+1)/2, what=numeric(0), size=size)
  }
  else N=readBin(NFile, n=1, what=numeric(0), size=size)
  i=sapply(1:n, sum_i)
  return(list(diag=grm[i], off=grm[-i], id=id, N=N))
}

GRM=ReadGRMBin(prefix = "ozGRM")
```

We can look at the distribution of diagonal values (ie the covariation of a person with themselves) using the following code.

```
hist(GRM$diag, breaks=100) # Large values can indicate data problems
or outstanding homozygosity rate
```

Usually we would expect the values to be close to 1. This teaching example only includes

~10,000 snps so there is more variation here than we would usually expect.

How many off diagonal elements are in this matrix (ie covarations between people)?

Use `length(GRM$off)` to find out and report the number in the box below

We can look at the off diagonal elements using the following code

```
hist(GRM$off, breaks = 100)
```

This plot isn't very informative as most relationship coefficients are close to 0. We can update our plot to only show relationship coefficients greater than 0.05 using the following code

```
hist(GRM$off[which(GRM$off>0.05)], breaks=100)
```

This plot clearly shows we have a lot of siblings and MZ twin pairs. If we zoom in even further we might spot those cousins

```
hist(GRM$off[which(GRM$off>0.05)], breaks=100, xlim=c(0.05,0.3)
```

We will use R to write out the files to run GCTA

#We'll start by reading in the data files again in case they have been lost
```
prs=read.table("PRSice.all_score", header=T)
covar<-read.table("ozht.cov", header=T)
pheno<-read.table("ozht.pheno", header=T)
```

#Then rescaling and renaming
```
pheno$ht<-pheno$ht*100
names(prs)<-c("FID","IID","PRS1","PRS2","PRS3","PRS4","PRS5")
```

#Then merging the data into 1 dataframe
```
all=merge(covar, prs, by=c("IID", "FID"))
all=merge(all, pheno, by=c("IID", "FID"))
```

# Then we'll write out the covariate files for the continuous covariates that we want to include in the GCTA model
# It's important that the PRS is in the last column of this file

```
write.table(all[,c("FID", "IID", "age", "PC1", "PC2", "PC3", "PC4",
"PRS1")], "ozht.prs1.qcovar", quote = F, row.names = F)
write.table(all[,c("FID", "IID", "age", "PC1", "PC2", "PC3", "PC4",
"PRS2")], "ozht.prs2.qcovar", quote = F, row.names = F)
write.table(all[,c("FID", "IID", "age", "PC1", "PC2", "PC3", "PC4",
"PRS3")], "ozht.prs3.qcovar", quote = F, row.names = F)
write.table(all[,c("FID", "IID", "age", "PC1", "PC2", "PC3", "PC4",
"PRS4")], "ozht.prs4.qcovar", quote = F, row.names = F)
write.table(all[,c("FID", "IID", "age", "PC1", "PC2", "PC3", "PC4",
"PRS5")], "ozht.prs5.qcovar", quote = F, row.names = F)
```

# Then we'll write out the covariate files for the binary covariates that we want to include in the GCTA model

```
write.table(all[,c("FID", "IID", "sex")], "ozht.covar", quote = F,
row.names = F)
```

# Then we'll write out the phenotype files for the GCTA model

```
write.table(all[,c("FID", "IID", "ht")], "ozht.phen", quote = F,
row.names = F)
```

Now we are ready to run our gcta analyses. As a reminder you will run these analyses by pasting this code into the ssh terminal

```
gcta64 --pheno ozht.phen --qcovar ozht.prs1.qcovar --covar ozht.covar
--grm ozGRM --out ozhtPRS1 --reml --reml-est-fix
gcta64 --pheno ozht.phen --qcovar ozht.prs2.qcovar --covar ozht.covar
--grm ozGRM --out ozhtPRS2 --reml --reml-est-fix
gcta64 --pheno ozht.phen --qcovar ozht.prs3.qcovar --covar ozht.covar
--grm ozGRM --out ozhtPRS3 --reml --reml-est-fix
gcta64 --pheno ozht.phen --qcovar ozht.prs4.qcovar --covar ozht.covar
--grm ozGRM --out ozhtPRS4 --reml --reml-est-fix
gcta64 --pheno ozht.phen --qcovar ozht.prs5.qcovar --covar ozht.covar
--grm ozGRM --out ozhtPRS5 --reml --reml-est-fix
```

GCTA will make a .log file and an .hsq file for each model.
The beta and SE of the PRS effect can be found in the line labeled X_7 in the log files.
We can pull these out of the log files and save them to a file with the following code

```
grep X_7 *.log > gcta_results.txt
```

*An important note:*
*The beta and se from the PRS are in the line labeled X_7 because X_1 is the mean and the covariates are included in the model based on the column order (continuous covariates are listed before binary covariates) with the first covariate in the line labeled X_2.*
*The PRS is the 6th continuous covariate included in the analysis so it ends up in the line labeled X_7.*
*If you were to remove age from continuous covariate file so that the PRS was now in column 5 the beta and se for the PRS would now be labeled X_6*

Moving back to R we can process the output from the gcta analyses

```
# We can read in the gcta results
res<-as.data.frame(read.table("gcta_results.txt"))
names(res)<-c("Model","beta","se")

# To calculate R2 we scale beta into a marginal correlation and then
square it
# We are calculating this and adding it to the res object
res$r2<-rbind((res[1,2]/sd(all$ht, na.rm = T)*sd(all$PRS1 , na.rm =
T))**2,
    (res[2,2]/sd(all$ht, na.rm = T)*sd(all$PRS2 , na.rm = T))**2,
    (res[3,2]/sd(all$ht, na.rm = T)*sd(all$PRS3 , na.rm = T))**2,
    (res[4,2]/sd(all$ht, na.rm = T)*sd(all$PRS4 , na.rm = T))**2,
    (res[5,2]/sd(all$ht, na.rm = T)*sd(all$PRS5 , na.rm = T))**2)



# To test the significance of the betas we can use t-tests
# We are calculating this and adding it to the res object

res$tStat<-res$beta/res$se # Follows a student distribution
res$pval<-(1-pt(q=res$tStat, df = 1887, lower.tail = T))*2

# Now we can look at the results
res
```

How do these results compare to those from lm and openMx?

Consider your results from PRS2.

| | Beta | SE | P | r2 |
|---|---|---|---|---|
| **lm** | | | | |
| **openMx** | | | | |

Our last job in this section of the practical is to make some plots. First we will plot the R-squared by threshold.

```
# Data
r2 <- data.frame(
  name = c("5e-08", "1e-06", "0.0001", "0.01", "1"),
  R2 = c(1.236,1.913,2.218,2.340,2.561 ),
  P=c(0.000195,5.04e-06,6.91e-07,2.67e-07, 6.12e-08)
)
# Increase bottom margin
par(mar=c(4,4,4,4))
# Barplot
jpeg("height.R2.jpg", width = 800, height = 800, quality=75, bg =
"white", type = "cairo")
my_bar <- barplot(r2$R2 , border=F , names.arg=r2$name ,
                  las=2 ,
                  col=c('skyblue1','skyblue2','skyblue3','skyblue4')
,
                  ylim=c(0,3) ,
                  main="Predicting height from height PRS",
                  ylab="% variance explained", xlab="PRS Thresholds")
# Add P values to the bars
text(my_bar, r2$R2+0.04 , paste("P=", r2$P, sep="") ,cex=1)
dev.off()
```

Next we'll plot height by quintile of PRS (often these plots are made with deciles but the sample is small so we're going to use quintiles here)

```
library(Hmisc)
library(gplots)
# the following command divides the data into quintiles.
longData$percentiles <- factor(cut2(longData$PRS5, g=5,
levels.mean=TRUE))
```

```
# Plot the mean of height by precentile with 95% CIs
jpeg("Quintile.jpg", width = 800, height = 800, quality=75, bg =
"white", type = "cairo")
plotmeans(ht ~ percentiles, data = longData, xlab="Quintile of
PRS", ylab="Height in Cm",legends = c("1","2","3","4","5"))
dev.off()
```

Do the plots tell us anything about the genetic architecture of height in this sample?

**Discussion point 1**

Your group presents these results at a conference. After your talk you are approached by a researcher who would like to join the project and replicate the results.  They have access to the variables you need in a cohort of 10,000 individuals from a non-European ancestry cohort.

In your group discuss the ways in which these data could be included in the project. What are the advantages of including this cohort?
Summarize the main points of your discussion in the box below.

**Discussion point 2**

You've now completed all your analyses and you're finalizing the draft of the paper. Unfortunately you realize that the author list is probably incomplete and you need to provide extra authors from your group.

- How will you decide who has contributed enough to be an author?
- Can a paper have too many authors?
- If you place a limit on the number of authors a cohort can have who is most likely to be left off the list?

Discuss these questions within your group.

Summarise the main points of your discussion in the box below.

| Yes | No | It depends |
|---|---|---|

| | Yes | No | It depends... |
|---|---|---|---|
| Should someone who processed the biological samples, extracted DNA and organised the genotyping be an author? | ○ | ○ | ○ |
| Should someone who collected phenotypes be an author? | ○ | ○ | ○ |
| Should someone who ran the PRS analyses be an author? | ○ | ○ | ○ |
| Should someone who wrote the grant that funded the study and designed the study be an author? | ○ | ○ | ○ |

**Stretch goal 1!**

Awesome work - congratulations on getting to this point

So far we have been running saturated or AE models. We can also include PRS in an ACE or ADE model. Intuitively you might think that adding a PRS will only change your A estimates but this is not necessarily the case. (There is a paper by Conor Dolan et al in the folder which might shed some light on this).

In this stretch goal we're going to run an ACE model with a PRS in GCTA (this is another hack and not a built in option). To do this we are going to add a second relatedness matrix that summarises the common environmental relatedness (a CRM).

If we take another look at the contents of the GRM we can see that the sample contains a large amount of siblings

```
hist(GRM$off[which(GRM$off>0.20)], breaks=100)
```

We are going to reformat the GRM and turn it into an NxN table

```
# Format GRM into a NxN matrix
asNxNGRM<-function(BRMbin){
  mat<-matrix(0, nrow = length(BRMbin$diag), ncol =
length(BRMbin$diag))
mat[upper.tri(mat)]<-BRMbin$off
mat<-mat+t(mat)
diag(mat)<-BRMbin$diag
colnames(mat)<-BRMbin$id[,2]
rownames(mat)<-BRMbin$id[,2]
  return(mat)
}

# convert GRM into NxN matrix
GRMmat=asNxNGRM(GRM)
```

Next we are going to hack the GRM to produce a CRM

```
CRM=GRMmat
CRM[which(CRM> 0.20, arr.ind = T)]=1 # we assume that individuals
with GRM>0.2 have grown up together -> C=1
CRM[which(CRM< 0.20, arr.ind = T)]=0 # other less related individuals
have no shared environment -> C=0
```

We can check how many 0s and 1s (families) we have with the following code - how many families did you get?

```
table(CRM[upper.tri(CRM)])
```

We can check how many individuals are in the CRM with the following code - how many did you get?

```
table(diag(CRM)) # check diagonal
```

Next we will format and write out the CRM

```
# We get the info about family ID and IID - needed to create the
.grm.id file
fam=GRM$id
colnames(fam)=c("fam", "id")

# The genio package allows the writing of a GRM
library(genio) # needed to write_grm

genio::write_grm(kinship = CRM, name = "ozCRM", fam = fam )
```

# We will also write out a continuous covariate file that does not include a PRS

```
write.table(all[,c("FID", "IID", "age", "PC1", "PC2", "PC3", "PC4")],
"ozht.noprs.qcovar", quote = F, row.names = F)
```

```
ozht.noprs.qcovar", quote = F, row.names = F)
```

Now we are going to switch to the ssh terminal window to run the gcta analyses

We need to create a text file, which stores the name of the different GRM/CRM matrices so GCTA knows which to open

```
echo "ozGRM" > GRM-CRM.txt
echo "ozCRM" >> GRM-CRM.txt
head GRM-CRM.txt
```

Then we can run the gcta analyses

```
gcta64 --pheno ozht.phen --qcovar ozht.prs1.qcovar --covar ozht.covar --mgrm GRM-CRM.txt --out ozhtPRS1.ACE --reml --reml-est-fix
gcta64 --pheno ozht.phen --qcovar ozht.prs2.qcovar --covar ozht.covar --mgrm GRM-CRM.txt --out ozhtPRS2.ACE --reml --reml-est-fix
gcta64 --pheno ozht.phen --qcovar ozht.prs3.qcovar --covar ozht.covar --mgrm GRM-CRM.txt --out ozhtPRS3.ACE --reml --reml-est-fix
gcta64 --pheno ozht.phen --qcovar ozht.prs4.qcovar --covar ozht.covar --mgrm GRM-CRM.txt --out ozhtPRS4.ACE --reml --reml-est-fix
gcta64 --pheno ozht.phen --qcovar ozht.prs5.qcovar --covar ozht.covar --mgrm GRM-CRM.txt --out ozhtPRS5.ACE --reml --reml-est-fix
gcta64 --pheno ozht.phen --qcovar ozht.noprs.qcovar --covar ozht.covar --mgrm GRM-CRM.txt --out ozht.ACE.noPRS --reml --reml-est-fix
```

The we will pull out the results - remember the result is in the line labeled X_7 because of the order of the variables in the analysis (see the note above for more information)

```
grep X_7 *ACE.log > gcta_results_ACE.txt
```

Now we are going back to R

# First we can read in the gcta results

```
resACE<-as.data.frame(read.table("gcta_results_ACE.txt"))
names(resACE)<-c("Model","beta","se")
```

# To calculate R2 we scale beta into a marginal correlation and then square it
# We are calculating this and adding it to the resACE object

```
resACE$r2<-rbind((resACE[1,2]/sd(all$ht, na.rm = T)*sd(all$PRS1 ,
na.rm = T))**2,
    (resACE[2,2]/sd(all$ht, na.rm = T)*sd(all$PRS2 , na.rm = T))**2,
    (resACE[3,2]/sd(all$ht, na.rm = T)*sd(all$PRS3 , na.rm = T))**2,
    (resACE[4,2]/sd(all$ht, na.rm = T)*sd(all$PRS4 , na.rm = T))**2,
    (resACE[5,2]/sd(all$ht, na.rm = T)*sd(all$PRS5 , na.rm = T))**2)
```

# To test the significance of the betas we can use t-tests
# We are calculating this and adding it to the resACE object

```
resACE$tStat<-resACE$beta/resACE$se # Follows a student distribution
resACE$pval<-(1-pt(q=resACE$tStat, df = 1887, lower.tail = T))*2
```

# Now we can look at the results

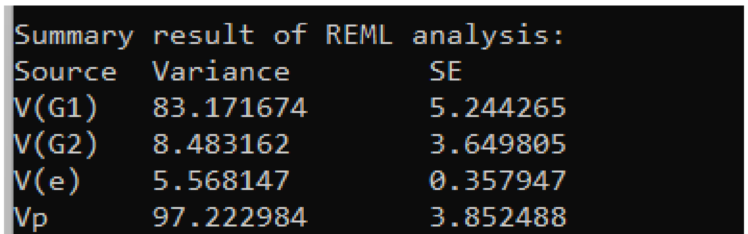
```
resACE
```

How do these results compare to your earlier results which are located in the object `res` ?

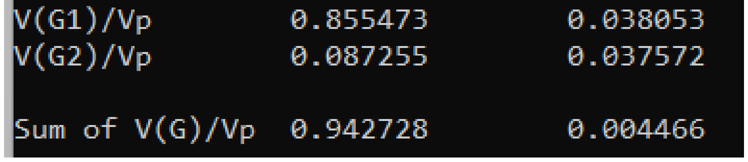
Take a look at the estimates of A and C from the model with no PRS - do this in the terminal

```
cat ozht.ACE.noPRS.log
```

This section contains the estimates
V(G1) is the variance explained by the first relatedness matrix in the GRM-GRM.txt file. This is the GRM and this is the unstandadised estimate of A with the SE of the estimate.
V(G2) is the variance explained by the second relatedness matrix in the GRM-GRM.txt file. This is the CRM and this is the unstandadised estimate of C with the SE of the estimate.
V(e) is the unstandardised estimate of E and Vp is the total phenotypic variance (A+C+E).
The standardised estimates of A and C are also provided.



```
Summary result of REML analysis:
Source  Variance        SE
V(G1)   83.171674       5.244265
V(G2)   8.483162        3.649805
V(e)    5.568147        0.357947
Vp      97.222984       3.852488
```


```
V(G1)/Vp        0.855473        0.038053
V(G2)/Vp        0.087255        0.037572

Sum of V(G)/Vp  0.942728        0.004466
```

What is the standardised estimate of E?

Now take a look at the unstandardised estimates of A, C and E from the models with PRS. These values are also in the .hsq files so to make this easier you can look at the values from all the models with the following code

```
grep "V(G1)" *hsq | grep -v Vp
grep "V(G2)" *hsq | grep -v Vp
```

How do the results compare - do both A and C change?

Now take a look at the standardised estimates of A, C and E from the models with PRS. These values are also in the .hsq files so to make this easier you can look at the values from all the models with the following code

```
grep "V(G1)" *hsq | grep Vp
grep "V(G2)" *hsq | grep Vp
```

Does looking at the standardised estimates tell you what is really happening in these models?

CONGRATULATIONS you have reached the end of stretch goal 1

**Stretch goal 2!**

If you made it this far and would like to do more PRS analyses you can.

In the files you copied there is one called `ozMass.txt`
This file contains the `twinData` in a long format (1 line per person) and 5 PRS calculated from a GWAS of Whole Body Fat Mass (WBFM) and 5 PRS calculated from a GWAS of Whole Body Fat Free Mass (WBFFM)

The PRS were calculated for the following thresholds 0.00000005, 0.000001, 0.0001, 0.01, 1

Read the data into R.
Reshape the data to make it wide (you can borrow the code we used in the height example)
Analyze the data in openMx or gcta and record the beta, se, p and r2 for each of the 10 PRS.
Make a plot of the r2 - see if you can put both WBFM and WBFFM on the same plot

What predicts better Fat Mass or Fat Free Mass?

CONGRATULATIONS you have reached the end of stretch goal 2

**Super Stretch Goal!**

Check to see if there is an interaction between sex and the PRS

There is a catch here as to model this correctly you actually need to include all possible interaction terms not just the sex*PRS term.
You can find out more about this by reading the Keller_2014_GxE.pdf that is included in the files you copied today

Did you find an interaction?

You are AMAZING and you have made it all the way through the practical!

You can find a copy of all the files made in the tutorial in /faculty/sarah/2022/Answers