

Practical in Polygenic Risk Scoring using the R package qgg

Palle Duun Rohde, Izel Fourie Sørensen & Peter Sørensen

2022-07-05

Introduction

The aim of this practical is to provide a simple introduction of polygenic risk scoring (PRS) analyses of complex traits and diseases. The practical will be a mix of theoretical and practical exercises in R that are used for illustrating/applying the theory presented in the corresponding lecture on polygenic risk scoring:

- Data used for computing polygenic risk scores
- Methods used for computing polygenic risk scores
- Methods used for evaluating the predictive ability of the polygenic risk scores

Sessions: This practical provides a step-by-step guide to performing basic PRS analyses including the following sessions:

- Session 1: Use R for downloading data
- Session 2: Prepare and explore phenotype data
- Session 3: Prepare and quality control of genotype data
- Session 4: Compute summary statistics
- Session 5: Compute sparse LD matrices
- Session 6: Compute polygenic risk scores using a clumping and thresholding (C+T) method
- Session 7: Compute polygenic risk scores using Bayesian Linear Regression (BLR) methods

Polygenic risk scores: Polygenic risk scoring combines information from large numbers of markers across the genome (hundreds to millions) to give a single numerical score for individual's risk for developing a specific disease on the basis of the DNA variants they have inherited.

For a particular disease or trait a polygenic risk score (PRS) is calculated as:

$$PRS = \sum_{i=1}^m X_i b_i$$

where X_i is the genotype vector, and b_i the weight of the i'th single genetic marker.

Genomic prediction used for many years in animal and plant breeding (e.g. Meuwissen et al. 2001). Genomic prediction (i.e. polygenic risk scoring) is now being in humans because of:

- Larger GWAS sample size = more precision for effect estimates
- Development of methods that combine genome-wide sets of variants
- Large Biobanks for validation and testing of genetic risk scores
- Ability to identify clinically meaningful increases in disease risk predictions

Terminology: Polygenic risk scores, polygenic scores, genomic risk score, genetic scores, genetic predisposition, genetic value, genomic breeding value is (more or less) the same thing.

Heritability: Heritability quantify the degree of variation in a phenotypic trait in a population that is due to genetic variation between individuals in that population. It measures how much of the variation of a trait can be attributed to variation of genetic factors, as opposed to variation of environmental factors. The narrow sense heritability is the ratio of additive genetic variance (σ_a^2) to the overall phenotypic variance ($\sigma_y^2 = \sigma_a^2 + \sigma_e^2$):

$$h^2 = \sigma_a^2 / (\sigma_a^2 + \sigma_e^2) \quad (1)$$

A heritability of 0 implies that no genetic effects influence the observed variation in the trait, while a heritability of 1 implies that all of the variation in the trait is explained by the genetic effects. In general the amount of information provided by the phenotype about the genetic risk is determined by the heritability. Note that heritability is population-specific and a heritability of 0 does not necessarily imply that there is no genetic determinism for the trait.

Complex traits and diseases: For many complex traits and diseases there will be thousands of genetic variants that each contribute with a small effect on disease risk. Rare variant with large effects will only explain small proportion of h^2 (low predictive potential). Common variants with small effects can explain larger proportion of h^2 (high predictive potential). Many complex traits and diseases in humans are heritable. The degree of heritability determines the value of using genetics for risk prediction. In general large data sets are required to obtain accurate estimates of small to moderate effects and thereby improve prediction accuracy.

Software used: To follow the practical, you will need the following installed (see installation guides below):

- R (version ≥ 4.2)
- qgg (version $\geq 1.1.1$)

We assume you have basic knowledge on how to use R. We suggest to use R through a user-friendly interface called Rstudio (although this is not a requirement).

Install R and Rstudio

R is a free software environment for statistical computing and graphics (<https://www.r-project.org/>). Because R is free and it is available for the most commonly used operating systems such as Windows, MacOSX and Linux, it has become very popular in statistics and in data science. Furthermore, R can be extended with user-contributed code and documentation (called R-packages) in a very easy and standardised way. The number of available R-packages is growing rapidly and has reached more than 18000 (<https://cran.r-project.org/web/packages/>).

RStudio (<https://www.rstudio.com/>) is a private company that among a large number of different products distributes the RStudio Integrated Development Environment (IDE) for R. A great number of different resources about R and RStudio IDE is available.

Install R from here: <https://mirrors.dotsrc.org/cran/>

Install Rstudio (free version) from here: <https://www.rstudio.com/products/rstudio/download/>

Further information and introduction to R and Rstudio can be found here:

<https://cran.r-project.org/doc/manuals/r-release/R-intro.html>

<https://www.rstudio.com/resources/cheatsheets>

<https://www.rstudio.com/resources/webinars>

Brief introduction to workflows used for PRS analyses in the qgg package

The practical is based on the R package **qgg**. This package provides an infrastructure for efficient processing of large-scale genetic and phenotypic data including core functions for:

- fitting linear mixed models
- constructing genetic relationship matrices
- estimating genetic parameters (heritability and correlation)
- performing genomic prediction and genetic risk profiling
- single or multi-marker association analyses

qgg handles large-scale data by taking advantage of:

- multi-core processing using openMP
- multithreaded matrix operations implemented in BLAS libraries (e.g. OpenBLAS, ATLAS or MKL)
- fast and memory-efficient batch processing of genotype data stored in binary files (e.g. PLINK bedfiles)

You can install qgg from CRAN with:

```
install.packages("qgg")
```

You can install the latest version of qgg from github with:

```
library(devtools)
devtools::install_github("psoerensen/qgg")
```

Input data/objects used in polygenic scoring in the qgg package

All functions in qgg used for polygenic scoring relies on a simple data infrastructure that takes five main input:

y: vector, matrix or list of phenotypes

X: design matrix for non-genetic factors

W: matrix of centered and scaled genotypes (in memory)

Glist: list structure providing information on genotypes, sparse LD, and LD scores (on disk)

stat: data frame with marker summary statistics

Prepare genotype information in Glist format (DO NOT RUN!)

```
Glist <- gprep(bed/bim/famfiles, task = "prepare")
```

Filter markers (DO NOT RUN!)

```
rsids <- gfilter(Glist, excludeMAF = 0.01, ....)
```

Compute sparse LD matrices and ldscscores (DO NOT RUN!)

```
Glist <- gprep(Glist, rsids, ids, ldfiles, task = "sparseld")
```

Compute summary statistics (e.g. fit single marker regression model) (DO NOT RUN!)

```
stat <- glma(y = y[train], X = X[train, ], Glist = Glist)
```

Quality control of summary statistics (DO NOT RUN!)

```
stat <- qcStat(stat = stat, Glist = Glist)
```

Clumping and thresholding (DO NOT RUN!)

```
stat <- adjStat(Glist = Glist, stat = stat, r2 = 0.9, threshold = c(0.5,  
  0.01, 0.001))
```

BLR model analysis based on summary statistics (DO NOT RUN!)

```
fit <- gbayes( stat=stat, Glist=Glist, method="bayesR")
```

Polygenic scoring (DO NOT RUN!)

```
grs <- gscore(Glist = Glist, stat = fit$stat)
```

Assess accuracy of polygenic scores (e.g. AUC or R2) (DO NOT RUN!)

```
acc(yobs = y[valid], ypred = grs[valid, ], typeoftrait = "binary")
```

Session 1: Downloading the data using R

In this practical we will perform polygenic risk scoring based on simulated data. The data consist of disease phenotype, covariables, and genetic marker data. The data used in this practical are intended for demonstration purposes only.

Load required packages:

```
library(data.table)
library(tools)
```

Create directory for downloading files:

```
dir.create("C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course")
```

Set working directory for the downloaded files:

```
setwd("C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course")
```

Download PLINK genotype files (bedfile, bimfile, famfile) from github repository:

Genotype data are commonly stored in binary format in PLINK .bed files. These files must be accompanied by .bim and .fam files. Read more about these file formats here:

1. <https://www.cog-genomics.org/plink/1.9/formats#bed>
2. <https://www.cog-genomics.org/plink/1.9/formats#bim>
3. <https://www.cog-genomics.org/plink/1.9/formats#fam>

```
url <- "https://github.com/psoerensen/qgdata/raw/main/simulated_human_data/human.bed"
download.file(url = url, mode = "wb", destfile = "human.bed")
url <- "https://github.com/psoerensen/qgdata/raw/main/simulated_human_data/human.bim"
download.file(url = url, destfile = "human.bim")
url <- "https://github.com/psoerensen/qgdata/raw/main/simulated_human_data/human.fam"
download.file(url = url, destfile = "human.fam")
```

Note that mode="wb" for downloading the human.bed file. This is needed or otherwise the bedfile will be corrupted. If the data file is corrupted it can cause errors in the results.

Check md5sum:

A md5sum hash is generally included in files so that file integrity can be checked. The following command performs this md5sum check in R:

```
md5sum("C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.bed")
```

This should be compared to the md5sum value before download. However the md5sum is not available on the github repository (anyone else knows how to obtain it?).

Read more about md5sum here:

<https://en.wikipedia.org/wiki/Md5sum>

Download pheno and covar files from github repository;

```
url <- "https://github.com/psoerensen/qgdata/raw/main/simulated_human_data/human.pheno"
download.file(url = url, destfile = "human.pheno")
url <- "https://github.com/psoerensen/qgdata/raw/main/simulated_human_data/human.covar"
download.file(url = url, destfile = "human.covar")
```

Session 2: Preparing the phenotype and covariable data using R

One of the first thing to do is to prepare the phenotypic data used in the analysis. The goal is to understand the variables, how many records the data set contains, how many missing values, what is the variable structure, what are the variable relationships and more. Several commands/functions can be used (e.g. `str`, `head`, `dim`, `table`, `is.na`).

```
library(data.table)
```

Read phenotype and covariables data files

```
pheno <- fread(input = "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.pheno",  
  data.table = FALSE)
```

```
covar <- fread(input = "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.covar",  
  data.table = FALSE)
```

How many observations and which variables do we have in the data set?

To get a fast overview of the data set you are working with you can use the `str` or `head` functions:

```
str(pheno)
```

```
## 'data.frame':    5000 obs. of  3 variables:  
## $ V1: chr  "IND1" "IND2" "IND3" "IND4" ...  
## $ V2: chr  "IND1" "IND2" "IND3" "IND4" ...  
## $ V3: int   0 0 1 0 0 1 0 0 1 1 ...
```

```
str(covar)
```

```
## 'data.frame':    5000 obs. of  14 variables:  
## $ V1 : chr  "IND1000" "IND1001" "IND1002" "IND1003" ...  
## $ V2 : chr  "IND1000" "IND1001" "IND1002" "IND1003" ...  
## $ V3 : int   1 0 0 0 0 1 1 1 1 0 ...  
## $ V4 : num   61.2 62.4 65.6 55.4 64.1 ...  
## $ V5 : num   0.00488 -0.0063 -0.00522 0.02028 -0.00411 ...  
## $ V6 : num  -0.002097 0.000148 -0.000606 0.006778 0.004349 ...  
## $ V7 : num   0.01091 0.00469 0.01709 -0.00306 -0.00839 ...  
## $ V8 : num   0.0056 0.01611 -0.00442 -0.01284 0.01022 ...  
## $ V9 : num  -0.008546 -0.00262 0.034218 -0.003078 -0.000912 ...  
## $ V10: num   0.019736 0.000308 -0.005157 0.012835 0.004788 ...  
## $ V11: num  -0.00408 0.00961 -0.00289 -0.01387 0.00977 ...  
## $ V12: num   0.01435 -0.01479 -0.00901 0.01499 0.00899 ...  
## $ V13: num  -0.00152 -0.00708 0.00706 0.00474 -0.00381 ...  
## $ V14: num   0.003998 0.037763 -0.000359 -0.022958 -0.010249 ...
```

```
head(pheno)
```



```
##      V1    V2 V3
## 1 IND1 IND1  0
## 2 IND2 IND2  0
## 3 IND3 IND3  1
## 4 IND4 IND4  0
## 5 IND5 IND5  0
## 6 IND6 IND6  1
```

```
head(covar)
```

```
##      V1      V2 V3      V4      V5      V6      V7      V8      V9
## 1 IND1000 IND1000 1 61.24445  0.004881 -0.002097  0.010908  0.005597 -0.008546
## 2 IND1001 IND1001 0 62.42271 -0.006301  0.000148  0.004692  0.016107 -0.002620
## 3 IND1002 IND1002 0 65.64274 -0.005215 -0.000606  0.017091 -0.004416  0.034218
## 4 IND1003 IND1003 0 55.40703  0.020284  0.006778 -0.003061 -0.012837 -0.003078
## 5 IND1004 IND1004 0 64.10666 -0.004105  0.004349 -0.008388  0.010221 -0.000912
## 6 IND1005 IND1005 1 65.77823 -0.010636 -0.021853 -0.017450  0.010058  0.000502
##      V10      V11      V12      V13      V14
## 1  0.019736 -0.004078  0.014350 -0.001518  0.003998
## 2  0.000308  0.009612 -0.014793 -0.007077  0.037763
## 3 -0.005157 -0.002894 -0.009013  0.007064 -0.000359
## 4  0.012835 -0.013872  0.014987  0.004744 -0.022958
## 5  0.004788  0.009773  0.008988 -0.003814 -0.010249
## 6  0.003807 -0.003007  0.005646 -0.007199 -0.009874
```

How is the phenotype distributed?

Define the response variable

```
y <- pheno[, 3]
names(y) <- pheno[, 1]
```

Use the histogram and boxplot functions to visualize the distribution the trait:

Which factors or covariates influence the phenotype?

The exploratory data analysis is the process of analyzing and visualizing the data to get a better understanding of the data. It is not a formal statistical test. Which factors should we include in the statistical model? To best answer these question we can fit a logistic regression model that include these factors in the model.

This can be done using the `glm` function:

```
fit <- glm(y ~ V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 +
          V12 + V13 + V14, data = covar, family = binomial(link = "logit"))
summary(fit)
```

```
##
## Call:
## glm(formula = y ~ V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 +
##      V12 + V13 + V14, family = binomial(link = "logit"), data = covar)
```

```

##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -0.9283  -0.7765  -0.7439   1.5279   1.8371
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.181957   0.214978  -5.498 3.84e-08 ***
## V3          -0.011066   0.065316  -0.169  0.8655
## V4           0.001740   0.003818   0.456  0.6487
## V5           1.309941   2.306100   0.568  0.5700
## V6          -0.280830   2.307468  -0.122  0.9031
## V7           2.354268   2.306546   1.021  0.3074
## V8          -1.349941   2.306494  -0.585  0.5584
## V9           5.520940   2.311308   2.389  0.0169 *
## V10          -2.724333   2.307482  -1.181  0.2377
## V11          -1.929366   2.307348  -0.836  0.4031
## V12           0.578919   2.308375   0.251  0.8020
## V13           4.382604   2.307287   1.899  0.0575 .
## V14          -0.504532   2.306778  -0.219  0.8269
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 5645.2  on 4999  degrees of freedom
## Residual deviance: 5631.6  on 4987  degrees of freedom
## AIC: 5657.6
##
## Number of Fisher Scoring iterations: 4

```

The exploration (including quality control) of phenotypes and covariables is a key step in quantitative genetic analyses. It is however beyond the scope of this practical.

Session 3: Prepare genotype for simulated data

The preparation (including quality control) of genotype data is a key step in quantitative genetic analyses. The function `gprep` reads genotype information from binary PLINK files, and creates the Glist object that contains general information about the genotypes such as allele frequencies, homozygosity, missingness, number of individuals etc..

```
library(qgg)
```

Summarize genotype information using the gprep

```
bedfiles <- "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.bed"
bimfiles <- "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.bim"
famfiles <- "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.fam"
```

```
Glist <- gprep(study = "Example", bedfiles = bedfiles, bimfiles = bimfiles,
              famfiles = famfiles)
names(Glist)
```

```
## [1] "study"      "bedfiles"   "bimfiles"   "famfiles"   "ids"        "n"
## [7] "rsids"      "mchr"       "a1"         "a2"         "position"   "chr"
## [13] "cpa"        "map"        "nmiss"      "af"         "af1"        "af2"
## [19] "maf"        "hom"        "het"        "n0"         "n1"         "n2"
## [25] "nchr"
```

The output from `gprep` (Glist) has a list structure that contains information about the genotypes in the binary file. Glist is required for downstream analyses of large-scale genetic data. Typically, the Glist is prepared once, and saved as an *.Rdata-file. To explore the content of the Glist object:

```
str(Glist)
```

```
## List of 25
## $ study : chr "Example"
## $ bedfiles: chr "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.bed"
## $ bimfiles: chr "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.bim"
## $ famfiles: chr "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.fam"
## $ ids : chr [1:5000] "IND1000" "IND1001" "IND1002" "IND1003" ...
## $ n : int 5000
## $ rsids :List of 1
## ..$ : Named chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## .. ..- attr(*, "names")= chr [1:50000] "1_568670_C_T" "1_761732_C_T" "1_830791_T_C" "1_867663_T_C"
## $ mchr : int 50000
## $ a1 :List of 1
## ..$ : Named chr [1:50000] "C" "C" "T" "T" ...
## .. ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ a2 :List of 1
## ..$ : Named chr [1:50000] "T" "T" "C" "C" ...
## .. ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ position:List of 1
## ..$ : Named num [1:50000] 568670 761732 830791 867663 916549 ...
## .. ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
```

```
## $ chr      :List of 1
## ..$ : Named chr [1:50000] "1" "1" "1" "1" ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ cpra     :List of 1
## ..$ : Named chr [1:50000] "1_568670_C_T" "1_761732_C_T" "1_830791_T_C" "1_867663_T_C" ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ map      :List of 1
## ..$ : Named num [1:50000] 0.000384 0.492847 0.620827 0.620827 0.69188 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ nmiss    :List of 1
## ..$ : Named int [1:50000] 0 0 0 0 0 0 0 0 0 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ af       :List of 1
## ..$ : Named num [1:50000] 0.103 0.355 0.264 0.495 0.475 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ af1      :List of 1
## ..$ : Named num [1:50000] 0.103 0.355 0.264 0.495 0.475 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ af2      :List of 1
## ..$ : Named num [1:50000] 0.897 0.645 0.736 0.505 0.525 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ maf      :List of 1
## ..$ : Named num [1:50000] 0.103 0.355 0.264 0.495 0.475 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ hom      :List of 1
## ..$ : Named num [1:50000] 0.812 0.539 0.61 0.496 0.498 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ het      :List of 1
## ..$ : Named num [1:50000] 0.188 0.461 0.39 0.504 0.502 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ n0       :List of 1
## ..$ : Named int [1:50000] 4015 2071 2704 1266 1371 2305 4983 4229 4816 2667 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ n1       :List of 1
## ..$ : Named int [1:50000] 942 2303 1948 2518 2510 2169 17 739 180 1988 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ n2       :List of 1
## ..$ : Named int [1:50000] 43 626 348 1216 1119 526 0 32 4 345 ...
## ..- attr(*, "names")= chr [1:50000] "1:568670" "1:761732" "1:830791" "1:867663" ...
## $ nchr     : int 1
```

Which quality controls are needed for genotype data?

Why quality controls for genotype data needed?

Quality control of genotype data

The genotype data must be quality controlled, e.g. removing markers with low genotyping rate, low minor allele frequency, out of Hardy-Weinberg Equilibrium. The function `gfilter` can be used for filtering of markers based:

```
rsids <- gfilter(Glist = Glist, excludeMAF = 0.05, excludeMISS = 0.05,
  excludeCGAT = TRUE, excludeINDEL = TRUE, excludeDUPS = TRUE,
  excludeHWE = 1e-12, excludeMHC = FALSE)
```

Session 4: Compute GWAS summary statistics

One of the first step in PRS analyses is to generate or obtain GWAS summary statistics. Ideally these will correspond to the most powerful GWAS results available on the phenotype under study. In this example, we will use GWAS on the simulated disease phenotype. We will use only a subset of the data (training data) in the GWAS and the remaining subset of the data (validation data) to assess the accuracy of the polygenic risk scores.

Define the response variable

```
y <- pheno[, 3]
names(y) <- pheno[, 1]
```

Create design matrix for the explanatory variables

```
X <- model.matrix(~V3 + V4 + V5 + V6 + V7 + V8 + V9 + V10 + V11 +
  V12 + V13 + V14, data = covar)
rownames(X) <- covar$V1
X <- X[names(y), ]
sum(names(y) %in% rownames(X))
```

```
## [1] 5000
```

Define training and validation samples

```
train <- sample(names(y), 4000)
valid <- names(y)[!names(y) %in% train]
```

Computation of GWAS summary statistics

What are GWAS summary statistics?

The function `glma` can be used for computation of summary statistics.

Computation of summary statistics:

```
stat <- glma(y = y[train], X = X[train, ], Glist = Glist)
```

Explore the output (stat) form the `glma` function:

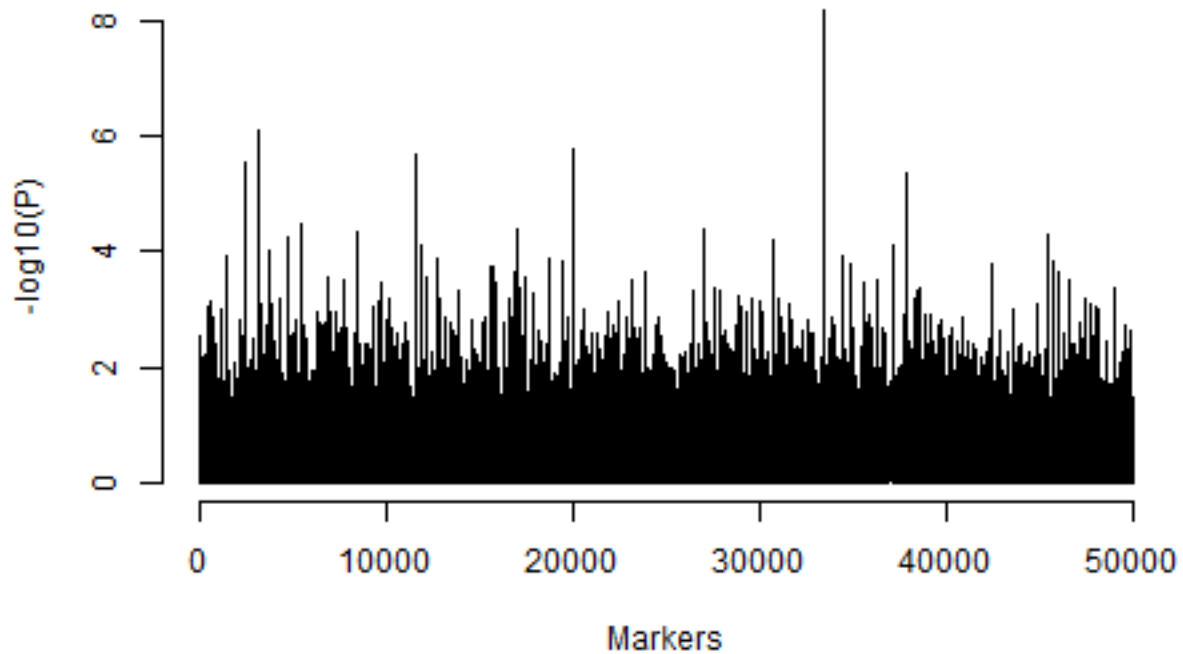
```
dim(stat)
```

```
## [1] 50000    13
```

```
head(stat)
```

##	rsids	chr	pos	a1	a2	af	b	seb	stat
##	1:568670	1	568670	C	T	0.1028	0.004372846	0.006890506	0.6346190
##	1:761732	1	761732	C	T	0.3555	0.013604759	0.006828506	1.9923478
##	1:830791	1	830791	T	C	0.2644	-0.007561625	0.006873267	-1.1001500
##	1:867663	1	867663	T	C	0.4950	-0.001595144	0.006838656	-0.2332540
##	1:916549	1	916549	A	G	0.4748	0.006745389	0.006825587	0.9882503
##	1:956852	1	956852	C	T	0.3221	-0.006363572	0.006860914	-0.9275109
##	p	n	ww	wy					
##	1:568670	0.5257132	3998	3942.836	17.24142				
##	1:761732	0.0464008	3998	4011.182	54.57116				
##	1:830791	0.2713330	3998	3961.839	-29.95794				
##	1:867663	0.8155761	3998	4003.200	-6.38568				
##	1:916549	0.3230899	3998	4017.618	27.10039				
##	1:956852	0.3537173	3998	3976.467	-25.30454				

```
plot(-log10(stat$p), frame.plot = FALSE, ylab = "-log10(P)",
     xlab = "Markers", type = "l")
```



Session 5: Compute sparse LD matrices

Polygenic risk scoring based on summary statistics require the construction of a reference linkage disequilibrium (LD) correlation matrix. The LD matrix corresponds to the correlation between the genotypes of genetic variants across the genome. Here we use a sparse LD matrix approach using a fixed window approach (e.g. number of markers, 1 cM or 1000kb), which sets LD correlation values outside this window to zero.

The function `gprep` can be used to compute sparse LD matrices which are stored on disk. The r^2 metric used is the pairwise correlation between markers (allele count alternative allele) in a specified region of the genome.

Define filenames for the sparse LD matrices.

```
ldfiles <- "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\human.ld"
```

Compute sparse LD using only the filtered rsids

```
Glist <- gprep(Glist, task = "sparseld", msize = 1000, rsids = rsids,  
  ldfiles = ldfiles, overwrite = TRUE)
```

Session 6: Polygenic risk scoring using a clumping and thresholding (C+T) method

Polygenic risk scoring using clumping and thresholding is a relative simple and robust method. Linkage disequilibrium makes identifying the contribution from causal independent genetic variants extremely challenging. One way of approximately capturing the right level of causal signal is to perform clumping, which removes markers in ways that only weakly correlated SNPs are retained but preferentially retaining the SNPs most associated with the phenotype under study. The clumping procedure uses a statistic (usually p-value) to sort the markers by importance (e.g. keeping the most significant ones). It takes the first one (e.g. most significant marker) and removes markers (i.e. set their effect to zero) if they are too correlated (e.g. $r^2 > 0.9$) with this one in a window around it. As opposed to pruning, this procedure makes sure that this marker is never removed, keeping at least one representative marker by region of the genome. Then it goes on with the next most significant marker that has not been removed yet.

Clumping and thresholding

Clumping can be performed using the `adjStat` in `qgg`. The input to the function is the summary statistic (`stat`), information about sparse LD matrices which is in the `Glist`, a threshold of linkage disequilibrium (e.g. $r^2 = 0.9$) and thresholds for p-values (`threshold = c(0.001, 0.05)`):

```
threshold <- c(1e-05, 1e-04, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2,
              0.5, 1)
statAdj <- adjStat(Glist = Glist, stat = stat, r2 = 0.9, threshold = threshold)
```

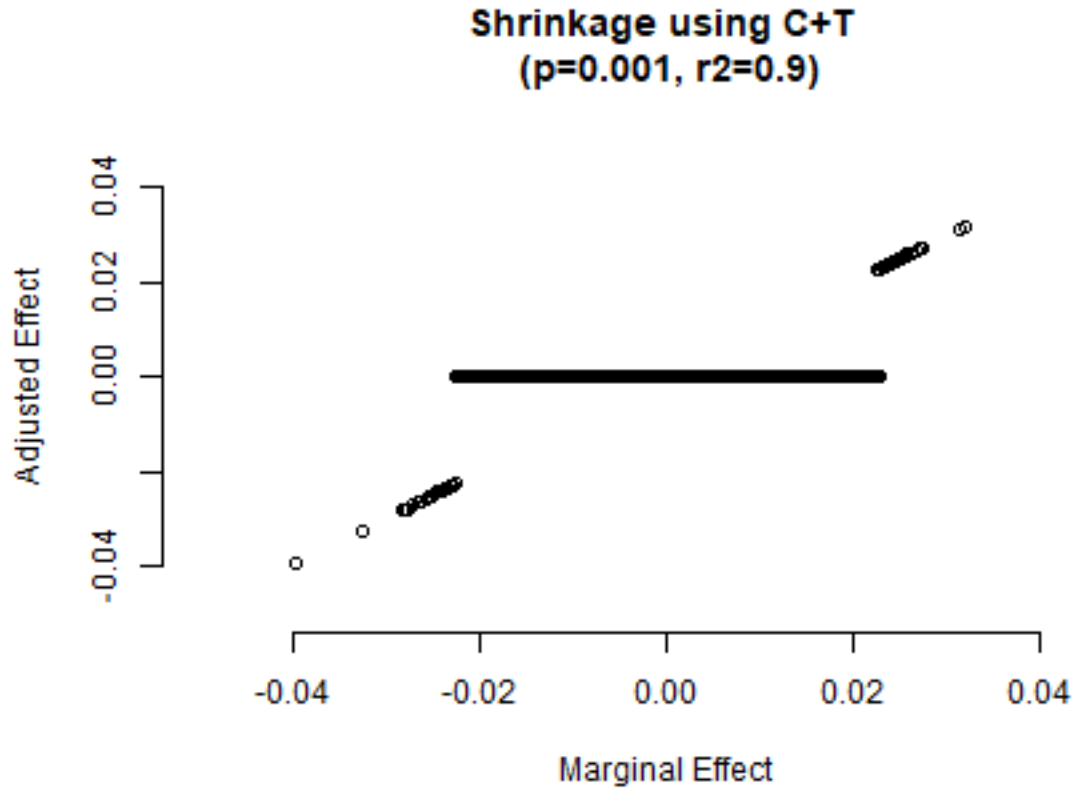
Explore the output (`statAdj`) using the `head`, `tail` and `summary` functions:

```
head(statAdj)
```

```
##           rsids chr    pos a1 a2      af      b b_1e.05 b_1e.04 b_0.001
## 1:568670 1:568670   1 568670  C  T 0.1028  0.004372846      0      0      0
## 1:761732 1:761732   1 761732  C  T 0.3555  0.013604759      0      0      0
## 1:830791 1:830791   1 830791  T  C 0.2644 -0.007561625      0      0      0
## 1:867663 1:867663   1 867663  T  C 0.4950 -0.001595144      0      0      0
## 1:916549 1:916549   1 916549  A  G 0.4748  0.006745389      0      0      0
## 1:956852 1:956852   1 956852  C  T 0.3221 -0.006363572      0      0      0
##           b_0.005 b_0.01      b_0.05      b_0.1      b_0.2      b_0.5
## 1:568670      0      0 0.00000000 0.00000000 0.00000000 0.00000000
## 1:761732      0      0 0.01360476 0.01360476 0.01360476 0.013604759
## 1:830791      0      0 0.00000000 0.00000000 0.00000000 -0.007561625
## 1:867663      0      0 0.00000000 0.00000000 0.00000000 0.00000000
## 1:916549      0      0 0.00000000 0.00000000 0.00000000 0.006745389
## 1:956852      0      0 0.00000000 0.00000000 0.00000000 -0.006363572
##           b_1
## 1:568670 0.004372846
## 1:761732 0.013604759
## 1:830791 -0.007561625
## 1:867663 -0.001595144
## 1:916549 0.006745389
## 1:956852 -0.006363572
```

To better understand what the clumping procedure is doing we can plot the unadjusted marker effect (from the `stat` data frame) against the adjusted marker effects (from the `statAdj` data frame)


```
plot(y = statAdj[rownames(stat), "b_0.001"], col = 1, x = stat$b,
     xlab = "Marginal Effect", ylab = "Adjusted Effect", frame.plot = FALSE,
     ylim = c(-0.05, 0.05), xlim = c(-0.05, 0.05), main = "Shrinkage using C+T \n (p=0.001, r2=0.9)")
```



Computing polygenic scores

For each of the p-value thresholds chosen in the C+T procedure a polygenic risk score is computed. The polygenic risk score (PRS) is calculated as:

$$PRS = \sum_{i=1}^m X_i b_i$$

where X_i is the genotype vector, and b_i the weight of the i'th single genetic marker. The PRS are computed using the `gscore` function. The input to the function is the adjusted summary statistic (`adjStat`), information about genotypes which are in the `Glist`:

```
grs <- gscore(Glist = Glist, stat = statAdj)
```

Explore polygenic scores

It is always important to explore the PRS computed.

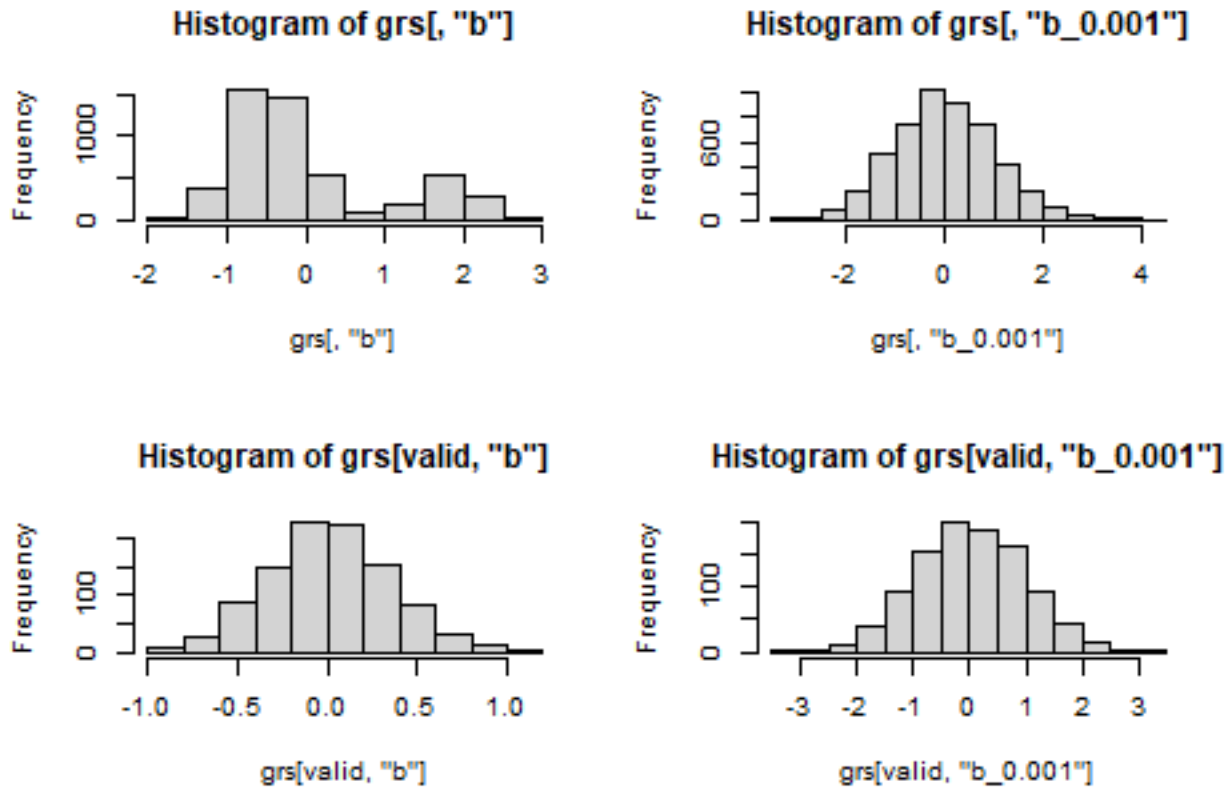
```
head(grs)
```

```
##           b      b_1e.05      b_1e.04      b_0.001      b_0.005      b_0.01
## IND1000 -0.3447964  0.3006850  0.9682779 -0.27823883 -0.3677530 -1.5748531
## IND1001 -0.3816874 -0.4505576 -1.4868376  0.05554413 -1.1175919 -0.7661481
## IND1002 -0.7026741 -1.8951543 -1.9465221 -2.10316548 -2.6121056 -2.3570986
## IND1003  1.7896168 -0.2464155  0.5340401  0.91151931  0.9935833  0.6311771
## IND1004  1.9471880  1.3078423  0.3845601  0.34988877  0.3203804  1.1870655
## IND1005 -0.5936767  0.3006850 -0.2520434  0.94297286 -0.2426019 -0.6171697
##           b_0.05      b_0.1      b_0.2      b_0.5      b_1
## IND1000 -0.6139658 -0.2724639 -0.1028791 -0.3857704 -0.3447964
## IND1001  0.1016956 -0.1582630 -0.3059590 -0.3367264 -0.3816874
## IND1002 -1.4584206 -0.8968075 -0.9579710 -0.7605080 -0.7026741
## IND1003  0.9058808  1.4272461  1.6414506  1.7389963  1.7896168
## IND1004  1.6727237  2.4602255  1.9697087  2.0756953  1.9471880
## IND1005 -0.7284329 -0.1733920 -0.4805994 -0.6074025 -0.5936767
```

```
cor(grs)
```

```
##           b      b_1e.05      b_1e.04      b_0.001      b_0.005      b_0.01      b_0.05
## b          1.0000000  0.1506419  0.2311565  0.4118399  0.5980999  0.6965628  0.8838618
## b_1e.05    0.1506419  1.0000000  0.6463747  0.3567647  0.2430431  0.2109885  0.1713807
## b_1e.04    0.2311565  0.6463747  1.0000000  0.5472769  0.3802017  0.3270058  0.2556454
## b_0.001    0.4118399  0.3567647  0.5472769  1.0000000  0.6744126  0.5792780  0.4591517
## b_0.005    0.5980999  0.2430431  0.3802017  0.6744126  1.0000000  0.8637349  0.6736186
## b_0.01     0.6965628  0.2109885  0.3270058  0.5792780  0.8637349  1.0000000  0.7878023
## b_0.05     0.8838618  0.1713807  0.2556454  0.4591517  0.6736186  0.7878023  1.0000000
## b_0.1      0.9362052  0.1593720  0.2394704  0.4337783  0.6388242  0.7418950  0.9421374
## b_0.2      0.9706233  0.1530891  0.2311189  0.4202312  0.6149745  0.7163896  0.9089219
## b_0.5      0.9956166  0.1514694  0.2307180  0.4136484  0.6003620  0.6996538  0.8873248
## b_1        1.0000000  0.1506419  0.2311565  0.4118399  0.5980999  0.6965628  0.8838618
##           b_0.1      b_0.2      b_0.5      b_1
## b          0.9362052  0.9706233  0.9956166  1.0000000
## b_1e.05    0.1593720  0.1530891  0.1514694  0.1506419
## b_1e.04    0.2394704  0.2311189  0.2307180  0.2311565
## b_0.001    0.4337783  0.4202312  0.4136484  0.4118399
## b_0.005    0.6388242  0.6149745  0.6003620  0.5980999
## b_0.01     0.7418950  0.7163896  0.6996538  0.6965628
## b_0.05     0.9421374  0.9089219  0.8873248  0.8838618
## b_0.1      1.0000000  0.9636596  0.9400458  0.9362052
## b_0.2      0.9636596  1.0000000  0.9747260  0.9706233
## b_0.5      0.9400458  0.9747260  1.0000000  0.9956166
## b_1        0.9362052  0.9706233  0.9956166  1.0000000
```

```
layout(matrix(1:4, ncol = 2, byrow = TRUE))
hist(grs[, "b"])
hist(grs[, "b_0.001"])
hist(grs[valid, "b"])
hist(grs[valid, "b_0.001"])
```



How are the PRSs distributed?

Are the PRSs correlated?

Based on the plots which PRS do you think is the best performing?

Evaluate polygenic scores

The p-value threshold that provides the “best-fit” PRS under the C+T method is usually unknown. To approximate the “best-fit” PRS, we can perform a regression between PRS calculated at a range of p-value thresholds and then select the PRS that explains the highest phenotypic variance or has the highest AUC. This can be achieved using `acc` function as follows:

```
paCT <- acc(yobs = y[valid], ypred = grs[valid, ], typeoftrait = "binary")
paCT
```

##	Corr	R2	Nagel	R2	AUC	intercept	slope	MSPE
## b	0.057	0.003	0.005	0.530	0.258	0.073	0.355	
## b_1e.05	0.104	0.011	0.016	0.569	0.259	0.044	1.225	
## b_1e.04	0.069	0.005	0.007	0.547	0.258	0.030	1.193	
## b_0.001	0.154	0.024	0.035	0.600	0.256	0.071	1.023	
## b_0.005	0.134	0.018	0.026	0.585	0.254	0.072	0.803	
## b_0.01	0.121	0.015	0.022	0.569	0.257	0.071	0.736	
## b_0.05	0.092	0.008	0.012	0.561	0.258	0.074	0.506	
## b_0.1	0.108	0.012	0.017	0.575	0.258	0.103	0.426	
## b_0.2	0.076	0.006	0.008	0.552	0.258	0.083	0.392	

```
## b_0.5    0.052 0.003    0.004 0.524    0.258 0.064 0.364
## b_1      0.057 0.003    0.005 0.530    0.258 0.073 0.355
```

Which p-value threshold gives the highest accuracy?

Plot polygenic scores

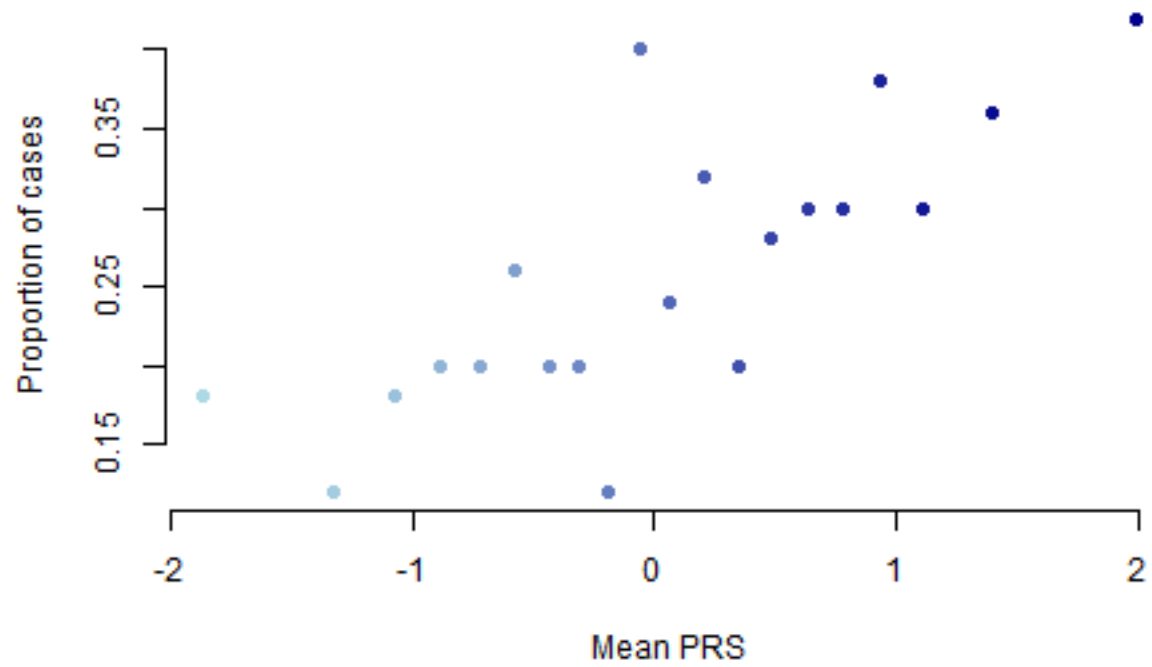
For visualization, the GRS were divided into percentiles, and the disease prevalence within each bin was computed; the OR for each percentile was computed adjusting for sex, age, UKB assessment center, and the first 10 genetic principal components, and the OR was expressed relative to the 50-th percentile.

```
yobs <- y[valid]
ypred <- grs[names(y[valid]), which.max(paCT[, "AUC"])]

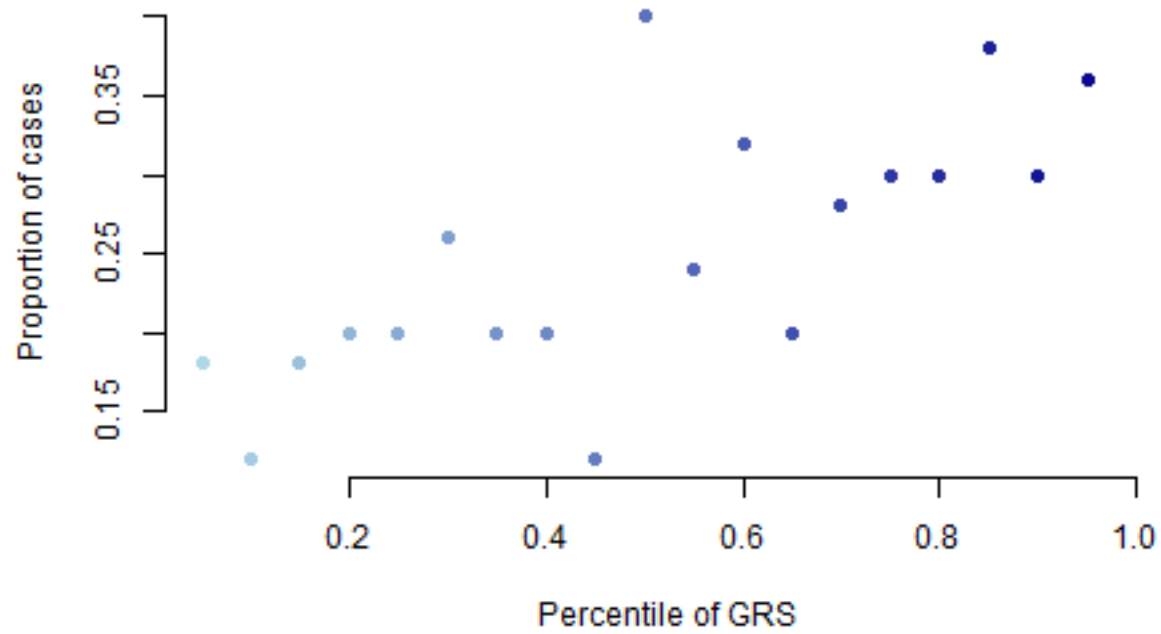
nbin <- 20
qsets <- qgg:::splitWithOverlap(names(ypred)[order(ypred)], length(ypred)/nbin,
0)
qy <- sapply(qsets, function(x) {
  mean(yobs[x])
})
qg <- sapply(qsets, function(x) {
  mean(ypred[x])
})

colfunc <- colorRampPalette(c("lightblue", "darkblue"))

plot(y = qy, x = qg, pch = 19, ylab = "Proportion of cases",
  xlab = "Mean PRS", col = colfunc(nbin), frame.plot = FALSE)
```



```
plot(y = qy, x = (1:nbin)/nbin, pch = 19, ylab = "Proportion of cases",
     xlab = "Percentile of GRS", col = colfunc(nbin), frame.plot = FALSE)
```



Session 7: Polygenic risk scoring using Bayesian Linear Regression (BLR) methods

Bayesian linear regression models have been proposed as a unified framework for gene mapping, prediction of genetic predisposition (polygenic risk scoring), estimation of genetic parameters and effect size distribution (Moser et al. 2015).

Bayesian linear regression models attempts to account for the underlying genetic architecture of the trait. This is achieved by using many linked markers covering the entire genome to jointly estimate marker effects, and by allowing the genetic signal to be heterogeneous distributed over the genome (i.e. some regions have stronger genetic signal than others). This may in some situations allow a more accurate estimate of the true underlying genetic signal leading to more accurate predictions.

Bayesian linear regression models can also be used to map genetic variants associated with phenotypes and to estimate the total variance explained by the genetic markers. Because they fit all markers simultaneously and account for linkage disequilibrium between markers, they should have greater power to detect true associations, find less false negatives and give unbiased estimates of the larger marker effects. They can also provide information about the genetic architecture of the trait from the hyper-parameters of the distribution of marker effects.

Bayesian linear regression models fit all markers simultaneously and their effects as drawn from a prior distribution that attempts to match the true distribution of marker effects as closely as possible. However the true distribution of effect sizes is unknown but a mixture of normal distributions can approximate a wide variety of distributions by varying the mixing proportions. Erbe et al. used this prior and included one component of the mixture with zero variance. A similar model was proposed by Zhou et al. but with a mixture of two normal distributions, one with a small variance and one with a larger variance.

In the following the statistical model, prior distributions of model parameters, algorithms for estimation of model parameters, for the Bayesian linear regression models is presented.

Statistical model

In the multiple regression model the phenotype is related to the set of genetic markers:

$$y = Xb + e \quad (2)$$

where y is the phenotype, X a matrix of SNP genotypes, where values are standardised to give the ij th element as: $x_{ij} = (x_{ij} - 2p_j) / \sqrt{2p_j(1 - p_j)}$, with x_{ij} the number of copies of the effect allele (e.g. 0, 1 or 2) for the i th individual at the j th SNP and p_j the allele frequency of the effect allele. \mathbf{b} are the genetic effects for each SNP, and e the residual error. The dimensions of y , X , b and e are dependent upon the number of traits, k , the number of SNP markers, m , and the number of individuals, n . The residuals, e , are a priori assumed to be independently and identically distributed multivariate normal with null mean and covariance matrix $I\sigma_e^2$.

Extensions to summary statistics

The key parameter of interest in the multiple regression model are the marker effects. These can be obtained by solving an equation system similar to:

$$b = \left(X'X + I \frac{\sigma_e^2}{\sigma_b^2} \right)^{-1} X'y \quad (3)$$

In order to solve this equation system individual level data (genotypes X and phenotypes y) is required. If these are not available, it is possible to reconstruct $X'y$ and $X'X$ from summary statistics and LD reference panel. $X'X$ is derived from an LD correlation matrix B (from population matched LD reference panel) and summary statistics:

$$X'X = D^{0.5}BD^{0.5} \quad (4)$$

where $D_i = \frac{1}{\sigma_{b_i}^2 + b_i^2/n_i}$ if the genotypes have been centered to mean 0 or $D_i = n_i$ if the genotypes have been centered to mean 0 and scaled to unit variance. $X'y$ is derived from marginal marker effects (b_m):

$$b_m = D^{-1}X'y \quad (5)$$

$$X'y = Db_m \quad (6)$$

The summary statistics methods used require the construction of a reference LD correlation matrix as shown in session 5.

Estimation of parameters using Bayesian methods

In the Bayesian multiple regression model the posterior density of the model parameters ($b, \sigma_b^2, \sigma_e^2$) depend on the likelihood of the data given the parameters and a prior probability for the model parameters:

$$p(b, \sigma_b^2, \sigma_e^2 | y) \propto p(y | b, \sigma_b^2, \sigma_e^2) p(b | \sigma_b^2) p(\sigma_b^2) p(\sigma_e^2) \quad (7)$$

The prior density of marker effects, $p(b | \sigma_b^2)$, defines whether the model will induce variable selection and shrinkage or shrinkage only. Also, the choice of prior will define the extent and type of shrinkage induced. Ideally the choice of prior for the marker effect should reflect the genetic architecture of the trait, and will vary (perhaps a lot) across traits.

Prior distributions of model parameters

Most complex traits and diseases are likely highly polygenic, with hundreds to thousands of causal variants, most frequently of small effect. So, the prior distribution must include many small and few large effects. Furthermore marker effects are a priori assumed to be uncorrelated (but markers can be in strong linkage disequilibrium and therefore a high posterior correlation). Many priors for marker effects have been proposed. These priors come more from practical (ease of computation) than from biological reasons. Each prior originates a method or family of methods, and we will describe them next, as well as their implications.

Prior marker variance Bayes N

In the BayesN approach the prior for each marker effect follows a priori a normal distribution with a variance σ_b^2 which is constant across markers:

$$p(b) = \prod_i p(b_i) \quad (8)$$

where

$$p(b_i) = N(0, \sigma_b^2) \quad (9)$$

In a normal distribution most effects are concentrated around 0, whereas few effects will be large. Therefore, the prior assumption of normality precludes few markers of having very large effects – unless there is a lot of information to compensate for this prior information.

Prior marker variance Bayes A

In the Bayes A approach it is assumed that a priori we have some information on the marker variance. For instance, this can be σ_b^2 . Thus, we may attach some importance to this value and use it as prior information for $\sigma_{b_i}^2$. A natural way of doing this is using an inverted chi-squared distribution with v_b degrees of freedom and scale parameter $S_b^2 = v_b \sigma_b^2$

$$p(b_i | \sigma_{b_i}^2) = N(0, \sigma_{b_i}^2) \quad (10)$$

In the second stage, we postulate a prior distribution for the variance themselves:

$$p(\sigma_{b_i}^2 | v_b, S_b^2) = S_b^2 \chi_{v_b}^{-1} \quad (11)$$

The value of σ_b^2 should be set as

$\sigma_b^2 = \frac{v_b - 2}{v_b} \frac{\sigma_g^2}{2 \sum_i p_i (1 - p_i)}$ because the variance of a t distribution is $\frac{v_b}{v_b - 2}$. It can be shown that this corresponds to a prior on the marker effects corresponding to a scaled t distribution (Gianola et al. 2009):

$$p(b_i | \sigma_b^2, v_b) = \sigma_b t(0, v_b) \quad (12)$$

which has the property of having “fat tails”. This means that large marker effects are more likely a priori compared to a normal distribution.

Prior marker variance Bayes C

In the Bayes C approach the marker effects, b , are a priori assumed to be sampled from a mixture with a point mass at zero and univariate normal distribution conditional on common marker effect variance σ_β^2 . This reflect a very common thought was that there were not many causal loci. This can be implemented by introducing additional variables δ_i which explain if the i -th marker has an effect or not. In turn, these variables δ have a prior distribution called Bernoulli with a probability π of being 0. Therefore the hierarchy of priors is:

$$p(\beta_j | \delta_i, \sigma_{b_i}^2, \pi) = \begin{cases} 0 & \text{with probability } \pi, \\ \sim N(0, \sigma_{b_i}^2) & \text{with probability } 1 - \pi, \end{cases} \quad (13)$$

$$p(\sigma_{b_i}^2 | v_b, S_b^2) = S_b^2 \chi_{v_b}^{-1} \quad (14)$$

where $S_b^2 = \sigma_b^2 v_b$ with

$\sigma_b^2 = \frac{\sigma_g^2}{(1 - \pi) 2 \sum_i p_i (1 - p_i)}$ because the variance of a t distribution is $\frac{v_b}{v_b - 2}$.

Prior marker variance Bayes R

In the Bayes R approach the marker effects, b , are a priori assumed to be sampled from a mixture with a point mass at zero and univariate normal distributions conditional on common marker effect variance σ_β^2 , and variance scaling factors, γ :

$$\beta_j | \pi, \sigma_\beta^2 = \begin{cases} 0 & \text{with probability } \pi_1, \\ \sim N(0, \gamma_2 \sigma_\beta^2) & \text{with probability } \pi_2, \\ \vdots & \\ \sim N(0, \gamma_C \sigma_\beta^2) & \text{with probability } 1 - \sum_{c=1}^{C-1} \pi_c, \end{cases} \quad (15)$$

where $\pi = (\pi_1, \pi_2, \dots, \pi_C)$ is a vector prior probabilities and $\gamma = (\gamma_1, \gamma_2, \dots, \gamma_C)$ is a vector of variance scaling factors for each of C marker variance classes. The γ coefficients are prespecified and constrain how the common marker effect variance σ_β^2 scales within each mixture distribution. Typically $\gamma = (0, 0.01, 0.1, 1.0)$. and $\pi = (0.95, 0.02, 0.02, 0.01)$.

The prior distribution for the marker variance σ_β^2 is assumed to be an inverse Chi-square prior distribution, $\chi^{-1}(S_\beta, \nu_\beta)$.

The proportion of markers in each mixture class π follows a Dirichlet $(C, c + \alpha)$ distribution, where c is a vector of length C that contains the counts of the number of variants in each variance class and $\alpha = (1, 1, 1, 1)'$.

Using the concept of data augmentation, an indicator variable $\mathbf{d} = (d_1, d_2, \dots, d_{m-1}, d_m)$, is introduced, where d_j indicates whether the j 'th marker effect is zero or nonzero.

Estimation of model parameters

Bayesian linear regression methods use an iterative algorithm for estimating joint marker effects. Estimation of the joint marker effects depend on additional model parameters such as a probability of being causal (π), an overall marker variance (σ_β^2), and residual variance (σ_e^2). Estimation of model parameters can be done using MCMC techniques by sampling from fully conditional posterior distributions.

Fit BayesN BLR model

```
fit <- gbayes(stat = stat, Glist = Glist, method = "bayesN",
  nit = 1000)
grs <- gscore(Glist = Glist, stat = fit$stat)
paN <- acc(yobs = y[valid], ypred = grs[valid, ], typeoftrait = "binary")
paN
```

Fit BayesA BLR model

```
fit <- gbayes(stat = stat, Glist = Glist, method = "bayesA",
  nit = 1000)
grs <- gscore(Glist = Glist, stat = fit$stat)
paA <- acc(yobs = y[valid], ypred = grs[valid, ], typeoftrait = "binary")
paA
```

Fit BayesC BLR model

```
fit <- gbayes(stat = stat, Glist = Glist, method = "bayesC",
  nit = 1000)
grs <- gscore(Glist = Glist, stat = fit$stat)
paC <- acc(yobs = y[valid], ypred = grs[valid, ], typeoftrait = "binary")
paC
```

Fit BayesR BLR model

```
fit <- gbayes(stat = stat, Glist = Glist, method = "bayesR",
  nit = 1000)
grs <- gscore(Glist = Glist, stat = fit$stat)
paR <- acc(yobs = y[valid], ypred = grs[valid, ], typeoftrait = "binary")
paR
```

```
auc <- c(paCT[-1, "AUC"], paN[, "AUC"], paA[, "AUC"], paC[, "AUC"],
  paR[, "AUC"])
plot(auc, ylab = "AUC")
names(auc) <- c("C+T 1e-5", "C+T 1e-4", "C+T 0.001", "C+T 0.005",
  "C+T 0.01", "C+T 0.05", "C+T 0.1", "C+T 0.2",
  "C+T 0.5", "C+T 1.0", "BayesN", "BayesA", "BayesC",
  "BayesR")
qqg:::plotForest(x = auc, sd = rep(0, length(auc)), reorder = FALSE,
  xlab = "AUC", main = "Accuracy")
```

Fit BLR models and compare shrinkage

```
fitN <- gbayes(stat = stat, Glist = Glist, method = "bayesN",
  nit = 1000)
fitA <- gbayes(stat = stat, Glist = Glist, method = "bayesA",
  nit = 1000)
fitC <- gbayes(stat = stat, Glist = Glist, method = "bayesC",
  nit = 1000)
fitR <- gbayes(stat = stat, Glist = Glist, method = "bayesR",
  nit = 1000)
```

Compare shrinkage

```
plot(y = fitN$stat[rownames(stat), "badj"], col = 1, pch = 1,
  x = stat$b, xlab = "Marginal Effect", ylab = "Adjusted Effect",
  frame.plot = FALSE, ylim = c(-0.05, 0.05), xlim = c(-0.05,
    0.05), main = "Shrinkage using BLR")
abline(a = 0, b = 1)
points(y = fitA$stat[rownames(stat), "badj"], x = stat$b, col = 2,
  pch = 2)
points(y = fitC$stat[rownames(stat), "badj"], x = stat$b, col = 3,
  pch = 3)
points(y = fitR$stat[rownames(stat), "badj"], x = stat$b, col = 4,
  pch = 4)
legend("top", legend = c("Bayes N", "Bayes A", "Bayes C", "Bayes R"),
  col = 1:4, pch = 1:4, bty = "n", text.col = "black", horiz = F)
```

```
plot(y = fitR$stat[rownames(stat), "badj"], col = 4, pch = 4,
  x = stat$b, xlab = "Marginal Effect", ylab = "Adjusted Effect",
  frame.plot = FALSE, ylim = c(-0.05, 0.05), xlim = c(-0.05,
    0.05), main = "Shrinkage BLR vs C+T")
abline(a = 0, b = 1)
points(y = statAdj[rownames(stat), "b_0.001"], x = stat$b, col = 2,
  pch = 2)
legend("top", legend = c("Bayes R", "C+T (P = 0.001)"), col = c(4,
  2), pch = c(4, 2), bty = "n", text.col = "black", horiz = F)
```