Practical in Polygenic Risk Scoring using the R package qgg

Palle Duun Rohde & Peter Sørensen

2022-07-06

Introduction

The aim of this practical is to provide a simple introduction to polygenic risk scoring (PRS) 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 perform quality control of genetic data
- Session 4: Compute GWAS summary statistics
- Session 5: Compute sparse LD matrices
- Session 6: Compute PRS using clumping and thresholding (C+T)
- Session 7: Compute PRS using different Bayesian Linear Regression (BLR) models

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 an individual's relative 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:

$$PRS = \sum_{i=1}^{m} X_i \hat{b}_i$$

where X_i is the genotype vector, and $\hat{b}_i$ the weight of the i'th single genetic marker.

Genomic prediction has been used for many years in animal and plant breeding (e.g., Meuwissen et al. 2001), and genomic prediction (i.e., PRS) has gained popularity during the last decade 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 predispostion, genetic value, genomic breeding value is (more or less) the same thing.

Heritability: The heritability (h^2) 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 ($\sigma_y^2 = \sigma_a^2 + \sigma_e^2$):

$$h^2 = \sigma_a^2/(\sigma_a^2 + \sigma_e^2) \tag{1}$$

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 the disease risk or quantitative trait. 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). The majority of complex traits and common diseases in humans are heritable. The heritability determines the value of using genetics for risk prediction. In general, large data sets are required to obtain accurate marker estimates of small to moderate effects, which also improves the prediction accuracy.

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

- R (version $\geq$ 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 the 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 offers a number of different products, with one being Rstudio which is an 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

Linking R to multi-threaded math libraries (DO NOT INSTALL IN THIS PRACTICAL)

The multi-core machines of today offer parallel processing power. To take advantage of this, R should be linked to multi-threaded math libraries (e.g. MKL/ OpenBLAS/ATLAS). These libraries make it possible for so many common R operations, such as matrix multiply/inverse, matrix decomposition, and some higher-level matrix operations, to compute in parallel and use all of the processing power available to reduce computation times.

This can make a huge difference in computation times: https://mran.microsoft.com/documents/rro/multithread#mt-bench

For Windows/Linux users it is possible to install Microsoft R Open is the enhanced distribution of R from Microsoft Corporation: https://mran.microsoft.com/open

For MAC users the ATLAS (Automatically Tuned Linear Algebra Software) library can be installed from here: https://ports.macports.org/port/atlas/

Introduction to PRS workflow using the qgg package

The practical is based on the R package qgg (Rohde et al. (2021)). 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")
```

The most recent version of qgg can be obtained from github:

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

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

All functions in qgg used for polygenic risk 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 ldscores (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!)

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

Assess accuracy of polygenic scores (e.g. AUC or R^2) (DO NOT RUN!)

```
acc(yobs = y[valid], ypred = prs[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, covariable, 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 (your own) directory for downloading files:

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

Set (your own) working directory for the downloaded files:

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

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

Genetic data are commonly stored in a binary format (as used by the software PLINK), named `.bed`-files. These files must be accompanied by `.bim` (contains information about the genetic variants) and `.fam` (contains information about the individuals) 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 bed-file will be corrupted. If the data file is corrupted it can cause errors in the analyses.

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:

```
# for MacOS / Linux users
system(paste("curl -sL https://github.com/psoerensen/qgdata/raw/main/simulated_human_data
             /human.bed | md5"))
```

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 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 an overview of the data set you are working with you can use the `str()` or `head()` functions:

```
str(pheno)
str(covar)
```

```
head(pheno)
head(covar)
```

How is the phenotype distributed?

Define the response variable

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

! Task for you

1. Use the histogram and boxplot functions to visualize the distribution of the trait/covariables.

Which factors or covariated 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)
```

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, homozygozity, missingnes, 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)
saveRDS(Glist, file = "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\Glist.RDS",
    compress = FALSE)
```

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 *.RDS-file. To explore the content of the `Glist` object:

```
names(Glist)
str(Glist)
```

! Task for you

1. Why should we do quality control of the genetic data?
2. Which quality metrics are needed for genetic data?

hint - see Choi et al (2020) (doi:10.1038/s41596-020-0353-1).

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:

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

! Task for you

1. How many variants are removed during the quality control?
2. Make a distribution plot of the frequency of the A_1-allele before and after quality control.

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.

! Task for you

1. What is a GWAS
2. What are GWAS summary statistics?

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))
```

Define training and validation samples

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

Computation of GWAS summary statistics

The function `glma` can be used for computing GWAS summary statistics.

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

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

```
dim(stat)
head(stat)
```

! Task for you

1. Present the results from the GWAS as a Manhattan plot.

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)
saveRDS(Glist, file = "C:\\Users\\au223366\\Dropbox\\Projects\\Summer_course\\Glist_sparseLD_1k.RDS",
    compress = FALSE)
```

! Task for you

1. In the context of a GWAS, why is information about LD important?)

Session 6: Compute PRS using clumping and thresholding (C+T)

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()`-function 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)
```

! Task for you

1. Plot the un-adjusted marker effect (from the `stat` dataframe) against the adjusted marker effects (from the the `statAdj` dataframe).

Computing polygenic risk scores

For each of the `P`-value thresholds chosen in the C+T procedure a PRS is computed as:

$$PRS = \sum_{i=1}^{m} X_i \hat{b}_i$$

where X_i is the genotype vector, and $\hat{b}_i$ 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`), and information about the genotypes which are in the `Glist`:

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

Explore polygenic scores

It is always important to explore the PRS computed.

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

! Task for you

1. How are the PRSs distributed?
2. Are the PRSs correlated?
3. Based on the plots, which PRS do you think has the best discriminating ability?

Evalute 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 = prs[valid, ], typeoftrait = "binary")
```

! Task for you

1. Make a plot that compares the prediction accuracies using the different P-value thresholds.

Plot polygenic scores

For visualization, the PRS can be divided into groups (e.g., deciles), and the disease prevalence within each group was computed.

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

nbin <- 10
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 PRS", col = colfunc(nbin), frame.plot = FALSE)
```

! Task for you

1. Make a plot (e.g., boxplot or violin plot) that compares the PRS for cases and controls.
2. Is there statistical difference in PRS between cases and controls?

Session 7: Compute PRS using different Bayesian Linear Regression (BLR) models

Bayesian linear regression models have been proposed as a unified framework for gene mapping, prediction of genetic predispostion (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:

$$y = Xb + e \tag{2}$$

where y is the phenotype, X a matrix of SNP genotypes, where values are standardised to give the ijth 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 ith individual at the jth SNP and p_j the allele frequency of the effect allele. 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:

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

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 a LD correlation matrix B (from a population matched LD reference panel) and summary statistics:

$$X'X = D^{0.5}BD^{0.5} \tag{4}$$
$$X'y = Db_m \tag{5}$$

where $D_i = \frac{1}{\sigma^2_{b_i} + b_i^2/n_i}$ 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, and $b_m = D^{-1}X'y$ is the marginal marker effects obtained from a standard GWAS. The construction of a LD correlation matrix, B, is shown in session 5.

Estimation of parameters using Bayesian methods

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

$$p(b, \sigma_b^2, \sigma_e^2|y) \propto p(y|b, \sigma_b^2, \sigma_e^2)p(b|\sigma_b^2)p(\sigma_b^2)p(\sigma_e^2) \tag{6}$$

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. 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 priory 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 some of them next, as well as their implications.

Prior marker variance Bayes N

In the Bayes N approach the prior the marker effect, b, follows a priori a normal distribution with a variance σ_b^2 which is constant across markers:

$$p(b) = \prod_i p(b_i) \tag{7}$$

where

$$p(b_i) = N\left(0, \sigma_b^2\right) \tag{8}$$

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 $\sigma^2_{b_i}$. A natural way of doing this is using an inverted chi-squared distribution with with υ_b degrees of freedom and scale parameter $S^2_b = \upsilon_b \sigma^2_b$

$$p(b_i|\sigma^2_{b_i}) = N\left(0, \sigma^2_{b_i}\right) \tag{9}$$

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

$$p(\sigma^2_{b_i}|\upsilon_b, S^2_b) = S^2_b \chi^{-1}_{\upsilon_b} \tag{10}$$

The value of σ^2_b should be set as $\sigma^2_b = \frac{\upsilon_b - 2}{\upsilon_b} \frac{\sigma^2_g}{2\sum_i p_i(1-p_i)}$ because the variance of a t distribution is $\frac{\upsilon_b}{\upsilon_b - 2}$. It can be shown that this corresponds to a prior on the marker effects corresponding to a scaled t distribution (Gianola et al. 2009):

$$p(b_i|\sigma^2_b, \upsilon_b) = \sigma_b t\left(0, \upsilon_b\right) \tag{11}$$

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^2_b. 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 Bernouilli with a probability π of being 0. Therefore the hierarchy of priors is:

$$p(b_j|\delta_i, \sigma^2_{b_i}, \pi) = \begin{cases} 0 & \text{with probability } \pi, \\ \sim N(0, \sigma^2_{b_i}) & \text{with probability } 1-\pi, \end{cases} \tag{12}$$

$$p(\sigma^2_{b_i}|\upsilon_b, S^2_b) = S^2_b \chi^{-1}_{\upsilon_b} \tag{13}$$

where $S^2_b = \sigma^2_b \upsilon_b$ with $\sigma^2_b = \frac{\sigma^2_g}{(1-\pi)2\sum_i p_i(1-p_i)}$ because the variance of a t distribution is $\frac{\upsilon_b}{\upsilon_b - 2}$.

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^2_b,and variance scaling factors, γ:

$$b_j|\pi, \sigma^2_b = \begin{cases} 0 & \text{with probability } \pi_1, \\ \sim N(0, \gamma_2 \sigma^2_b) & \text{with probability } \pi_2, \\ \vdots & \\ \sim N(0, \gamma_C \sigma^2_b) & \text{with probability } 1 - \sum_{c=1}^{C-1} \pi_c, \end{cases} \tag{14}$$

where $\pi = (\pi_1, \pi_2, ..., \pi_C)$ is a vector prior probabilities and $\gamma = (\gamma_1, \gamma_2, ..., \gamma_C)$ 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^2_b 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^2_b is assumed to be an inverse Chi-square prior distribution,$\chi^{-1}\left(S_b, \nu_b\right)$.

The proportion of markers in each mixture class π follows a Direchlet $(C, c + \alpha)$ 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 $d = (d_1, d_2, .., d_{m-1}, d_m)$, 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_b^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 BLR models

The different BLR models are implemented in the `qgg`-package in the function `gbayes()`, where the argument `method=` specifies which prior marker variance there should be used.

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

! Task for you

1. Run the different BLR models (bayesN, bayesA, bayesC and bayesR).
2. Compare the predictive performance of the BLR models with the C+T model (make figure).
3. Compare the adjusted marker effects from the different BLR models and C+T with the un-adjusted marker effects.