Brief Introduction to R package qgg

Palle Duun Rohde, Izel Fourie Sørensen, & Peter Sørensen

2022-10-13

Contents

1 Introduction 1

2 Prepare genotype data 3

3 Compute sparse LD matrices 4

4 Simulation of quantitative traits 7

5 Compute GWAS summary statistics 10

6 Clumping and thresholding (C+T) 12

7 Genomic risk scoring using clumping and thresholding (C+T) 14

8 Gene mapping and estimation of genetic parameters using Bayesian Linear Regression (BLR) models 16

9 Genomic risk scoring using Bayesian Linear Regression (BLR) models 22

10 Gene Set Enrichment Analysis 24

1 Introduction

The practical is based on the R package `qgg` (Rohde et al. (2021, 2022)). 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 (i.e., PLINK bedfiles)

You can install qgg from CRAN with:

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

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

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

Input data/objects commonly used in the qgg package

All functions in `qgg` used for analysis of complex traits relies on a simple data infrastructure that takes the following 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
`sets`:	list of sets with marker ids
`ids`:	vector of ids of individuals
`rsids`:	vector marker marker ids

Linking R to multi-threaded math libraries

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 many common R operations, such as matrix multiplication/inversion/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/

2 Prepare genotype data

The preparation (including quality control) of genotype data is a key step in quantitative genetic analyses.

```
library(qgg)
```

Summarize genotype information in PLINK files

The function `gprep()` reads genotype information from binary PLINK files, and creates the `Glist` object that contains general information about the genotypes:

```
bedfiles <- system.file("extdata", paste0("sample_chr", 1:2, ".bed"), package = "qgg")
bimfiles <- system.file("extdata", paste0("sample_chr", 1:2, ".bim"), package = "qgg")
famfiles <- system.file("extdata", paste0("sample_chr", 1:2, ".fam"), package = "qgg")

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

names(Glist)
```

```
##  [1] "study"    "bedfiles" "bimfiles" "famfiles" "ids"      "n"
##  [7] "rsids"    "mchr"     "a1"       "a2"       "pos"      "chr"
## [13] "cpra"     "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 provided in the qgg package. Typically, the `Glist` is prepared once, and saved as an *.RDS-file.

```
saveRDS(Glist, file = "Glist.RDS", compress = FALSE)
```

Quality control of genotype data

In general it advisable to perform quality control of the genotype data. The quality control include removing markers with low genotyping rate, low minor allele frequency, not in Hardy-Weinberg Equilibrium. The function `gfilter()` can be used for filtering of markers:

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

3 Compute sparse LD matrices

A number of methods used in the genetic analyses of complex traits (e.g. Bayesian linear regression analyses, genomic risk scoring and LD score regression) are based on summary statistics and 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.

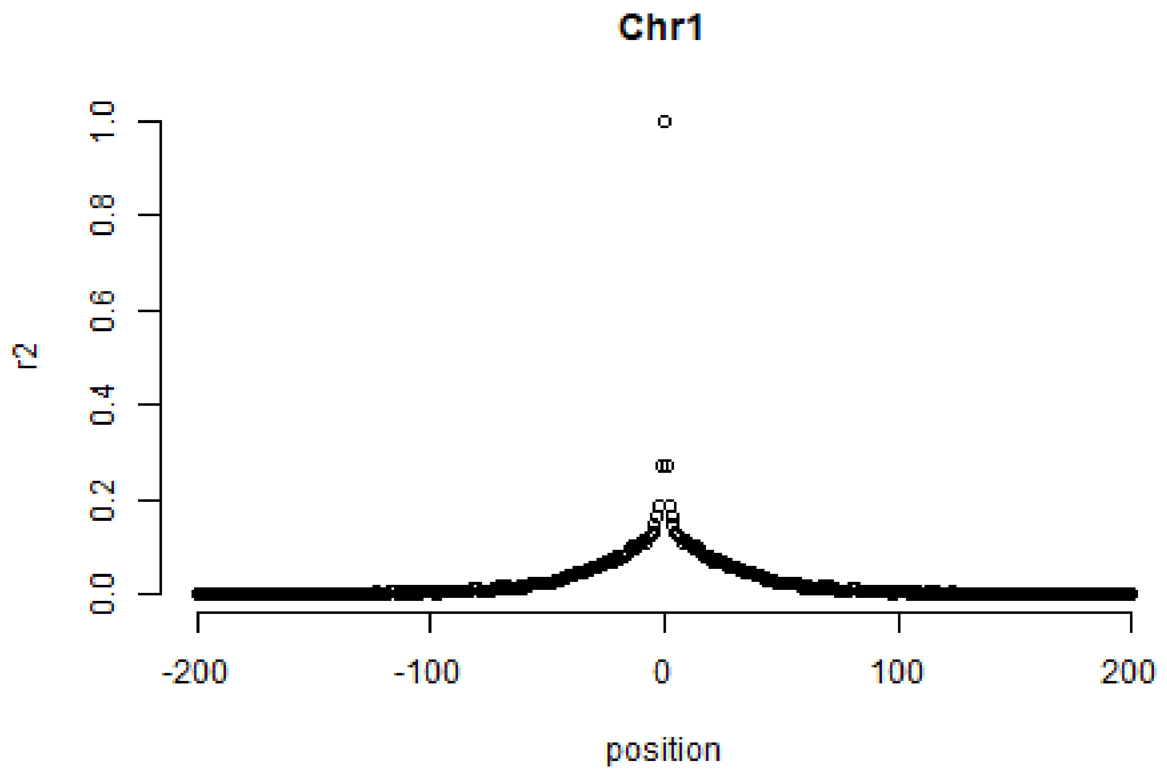
The function `gprep` can be used to compute sparse LD matrices which are stored on disk. The r^2 metric used is the pairwise correlation between markers (allele count alternative allele) in a specified region of the genome. Although this step can be slow unless R is linked to a fast BLAS it is typically only done once (or a few times).

```r
# Define filenames for the sparse LD matrices
nchr <- Glist$nchr
ldfiles <- paste0(getwd(), "/sample_chr", 1:nchr, ".ld")

# Compute sparse LD matrices using the filtered rsids only
Glist <- gprep(Glist, task = "sparseld", msize = 200, rsids = rsidsQC,
    ldfiles = ldfiles, overwrite = TRUE)

# Save the updated Glist object
saveRDS(Glist, file = "Glist.RDS", compress = FALSE)
```

Get the sparse LD matrix for a chromsome

The ´getLD´ function can be used to extract the sparse LD matrix stored on disk. Here we extract the sparse LD for chromosome 1 and plot the mean r2 in a genomic window around the index marker illustrating that marker in close proximity with the index marker have (on average) a higher r2 as compared to distantly located markers:

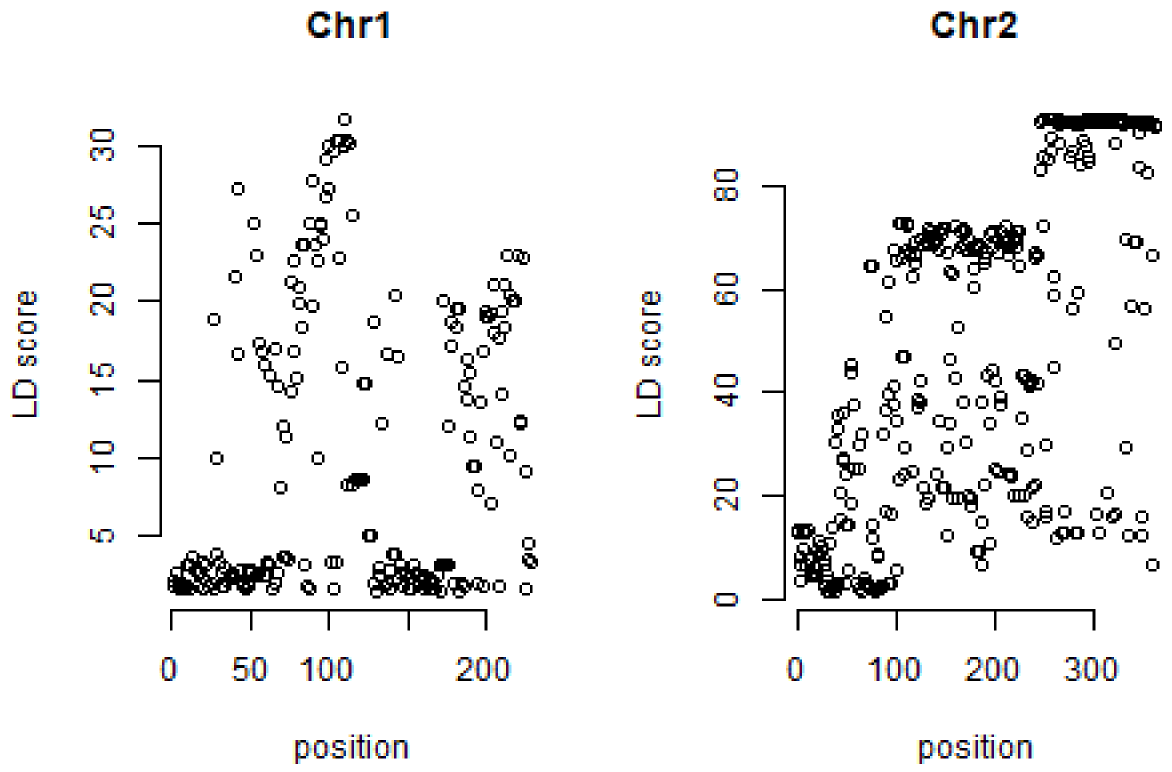
```r
ld <- getLD(Glist, chr = 1)
# Plot mean r2 f
plot(y = rowMeans(ld^2), x = rownames(ld), frame.plot = FALSE, ylab = "r2",
    xlab = "position", main = "Chr1")
```



Get the LD scores for a chromosome

The ld scores quantify the degree of linkage disequilibrium in a genomic region and are used LD score regression. They can be extracted from the Glist object in the following way:

```
layout(matrix(1:2, ncol = 2))
plot(Glist$ldscores[[1]], frame.plot = FALSE, ylab = "LD score", xlab = "position",
    main = "Chr1")
plot(Glist$ldscores[[2]], frame.plot = FALSE, ylab = "LD score", xlab = "position",
    main = "Chr2")
```



Get LD sets for chromosome 1

It can be useful to identify markers linked in a genomic regions. The ´getLDsets´ function can be used to extract linked marker based on a LD threshold such as r2=0.5:

```
sets <- getLDsets(Glist = Glist, chr = 1, r2 = 0.5)
str(sets, list.len = 5)
```

```
## List of 228
##  $ rs560838421: chr "rs560838421"
##  $ rs561578877: chr "rs561578877"
##  $ rs575349295: chr "rs575349295"
##  $ rs551139107: chr "rs551139107"
##  $ rs570124710: chr "rs570124710"
##   [list output truncated]
```

4 Simulation of quantitative traits

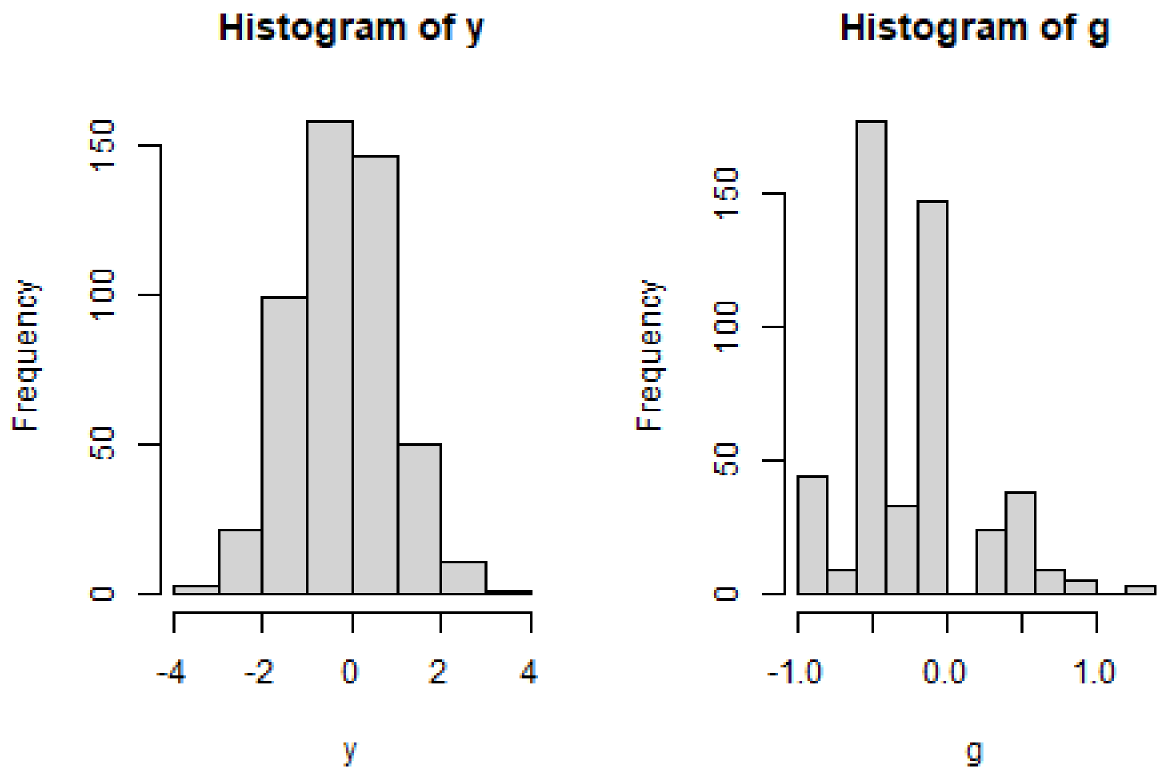
It is possible to simulate a quantitative trait using the ´gsim´ function. The trait is simulated based on the genotypes in the Glist and current implementation is mainly used for exploring the different modeling approaches provided in the qgg package. In a future release we plan to further expand on this functionality.

Single trait

```
# Simulate phenotype
sim <- gsim(Glist = Glist, nt = 1)

str(sim)
```

```
## List of 9
##  $ y     : Named num [1:489] 1.08 1.51 1.02 -1.58 1.42 ...
##   ..- attr(*, "names")= chr [1:489] "HG00096" "HG00097" "HG00099" "HG00101" ...
##  $ W     : int [1:489, 1:228] 0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:489] "HG00096" "HG00097" "HG00099" "HG00101" ...
##   .. ..$ : chr [1:228] "rs560838421" "rs561578877" "rs575349295" "rs551139107" ...
##  $ e     : num [1:489] 1.58 1.01 1.52 -1.08 1.42 ...
##  $ g     : num [1:489, 1] -0.5 0.5 -0.5 -0.5 0 -0.5 -0.25 -1 -0.5 -0.5 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:489] "HG00096" "HG00097" "HG00099" "HG00101" ...
##   .. ..$ : chr "D1"
##  $ b0    : num [1:2] 0.5 0.25
##  $ b1    :List of 1
##   ..$ : num [1:2] -0.5 0.5
##  $ set0  : int [1:2] 5 59
##  $ set1  :List of 1
##   ..$ : int [1:2] 92 10
##  $ causal: int [1:4] 5 59 92 10
```

```
y <- sim$y  # phenotype
g <- sim$g  # genetic values
e <- sim$e  # residuals

h2 <- var(g)/var(y)  # heritability

# Some plots of the simulated data
layout(matrix(1:2, ncol = 2))
hist(y)
hist(g)
```



Multiple trait

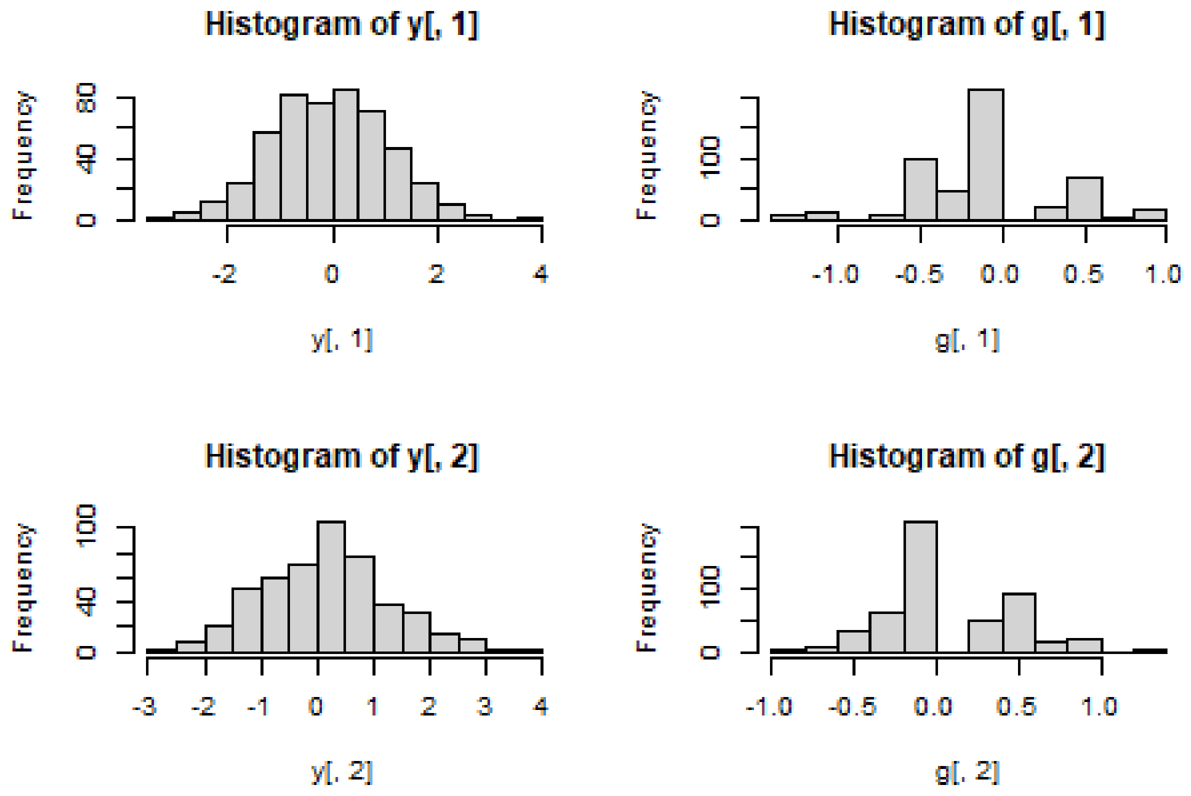
```
# Simulate phenotype
sim <- gsim(Glist = Glist, nt = 2)

y <- sim$y  # phenotype
g <- sim$g  # genetic values
e <- sim$e  # residuals

# Simulated heritability and covariance
h2 <- diag(var(g)/var(y))
h2
```

```
##        D1        D2
## 0.1494355 0.1187142
```

```
# Simulated genetic correlation
cor(g)
```

```
##           D1        D2
## D1 1.0000000 0.7611184
## D2 0.7611184 1.0000000
```

```r
# Some plots of simulated data
layout(matrix(1:4, ncol = 2))
hist(y[, 1])
hist(y[, 2])
hist(g[, 1])
hist(g[, 2])
```



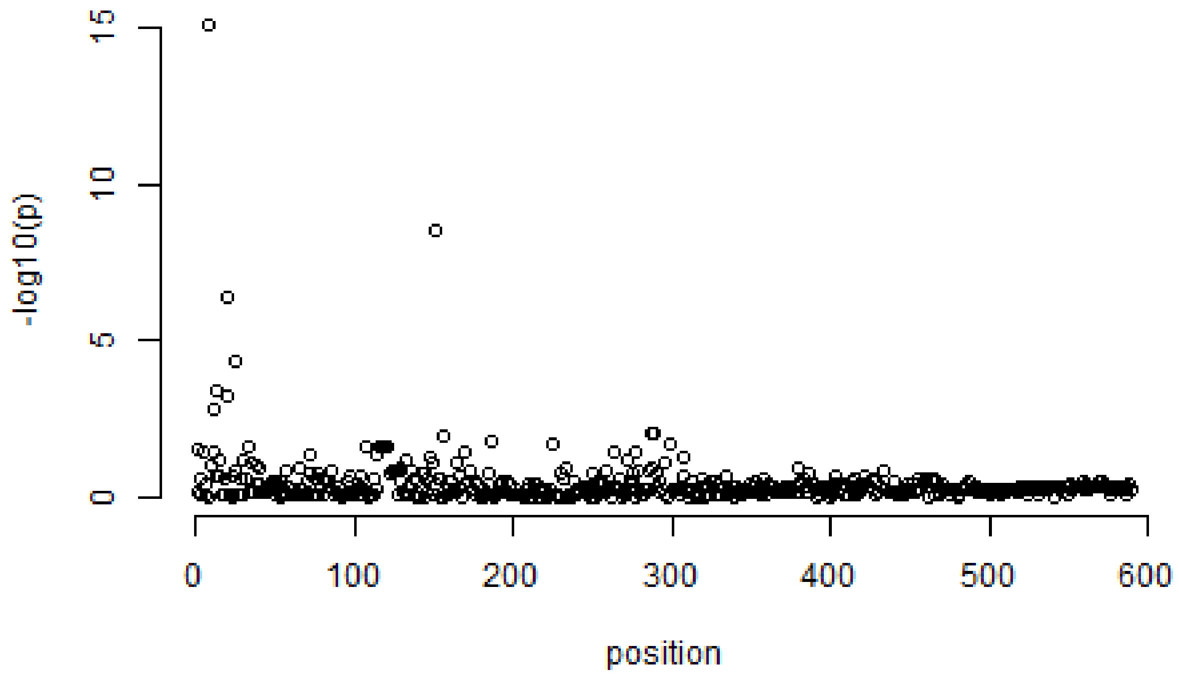
5 Compute GWAS summary statistics

A number of methods used in the genetic analyses of complex traits (e.g. Bayesian linear regression analyses, genomic risk scoring and LD score regression) are based on GWAS summary statistics. The function `glma` can be used for computing GWAS summary statistics. Currently this function only fit a simple linear regression model, but we plan to add further modeling approached in a future release. In the example below we only compute summary statistics for the markers that fullfill the quality control criteria.

```
# Simulate phenotype
sim <- gsim(Glist = Glist, nt = 1)
y <- sim$y  # phenotype

# Define deign matrix for covariates
X <- model.matrix(y ~ 1)

# Compute summary statistics
stat <- glma(y = y, X = X, Glist = Glist, rsids = rsidsQC)

head(stat)
```

```
##                   rsids chr     pos ea nea        eaf           b        seb
## rs560838421 rs560838421  21 9442544  G   A 0.06237219  0.11892338 0.05374288
## rs561578877 rs561578877  21 9450695  T   C 0.11963190 -0.01957349 0.05026082
## rs575349295 rs575349295  21 9455527  T   C 0.17075665 -0.06289456 0.05599531
## rs551139107 rs551139107  21 9466700  T   C 0.08179959 -0.01338403 0.05467394
## rs570124710 rs570124710  21 9471511  T   C 0.06850716 -0.11576592 0.05396238
## rs558507375 rs558507375  21 9471670  G   A 0.05828221  0.01085157 0.05283748
##                   stat          p   n       ww         wy
## rs560838421  2.2128210 0.02737301 487 456.4711  54.285086
## rs561578877 -0.3894384 0.69712213 487 526.9940 -10.315113
## rs575349295 -1.1232113 0.26190117 487 423.6168 -26.643193
## rs551139107 -0.2447973 0.80671653 487 445.4365  -5.961736
## rs570124710 -2.1453079 0.03242209 487 453.0362 -52.446153
## rs558507375  0.2053765 0.83736382 487 476.9559   5.175723
```

```
# A simple Manhatten plot
plot(-log10(stat$p), frame.plot = FALSE, ylab = "-log10(p)", xlab = "position")
```



15
10
5
0
-log10(p)
0
100
200
300
400
500
600
position

6 Clumping and thresholding (C+T)

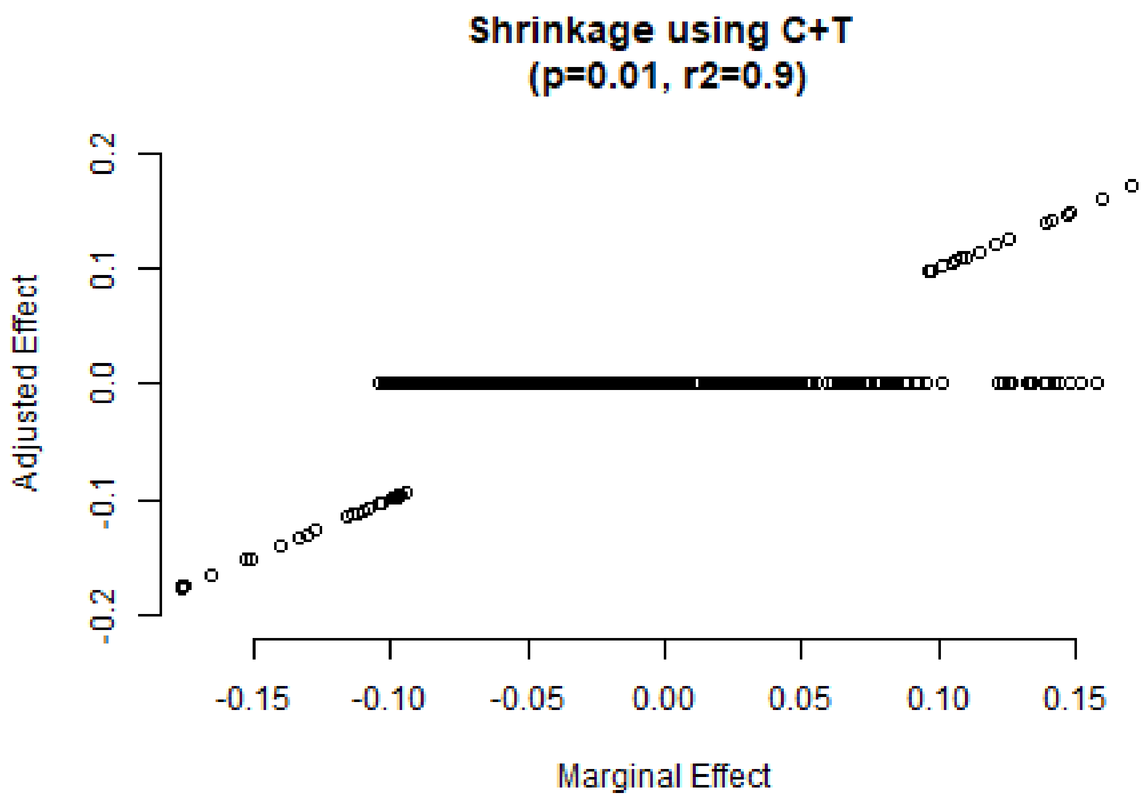
The clumping and thresholding procedure is used in both gene mapping and genomic risk scoring methods. It is used because linkage disequilibrium makes identifying the contribution from causal independent genetic variants extremely challenging. One way of approximately capturing the right level of causal 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 and thresholding 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. Then it goes on with the next most significant marker that has not been removed yet. 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, ...)`):

```r
# Simulate phenotype
sim <- gsim(Glist = Glist, nt = 1)
y <- sim$y  # phenotype

# Define deign matrix for covariates
X <- model.matrix(y ~ 1)

# Compute summary statistics
stat <- glma(y = y, X = X, Glist = Glist, rsids = rsidsQC)

# Adjust summary statistics using clumping and p-value thresholding
statAdj <- adjStat(Glist = Glist, stat = stat, r2 = 0.9, threshold = c(0.01,
    0.05, 0.1))

# Output (statAdj) is a data frame
head(statAdj)
```

```
##                   rsids chr     pos ea nea        eaf            b b_0.01
## rs560838421 rs560838421  21 9442544  G   A 0.06237219  0.002320816      0
## rs561578877 rs561578877  21 9450695  T   C 0.11963190  0.026367380      0
## rs575349295 rs575349295  21 9455527  T   C 0.17075665  0.003107604      0
## rs551139107 rs551139107  21 9466700  T   C 0.08179959  0.042402431      0
## rs570124710 rs570124710  21 9471511  T   C 0.06850716 -0.033646478      0
## rs558507375 rs558507375  21 9471670  G   A 0.05828221  0.044593198      0
##             b_0.05 b_0.1
## rs560838421      0     0
## rs561578877      0     0
## rs575349295      0     0
## rs551139107      0     0
## rs570124710      0     0
## rs558507375      0     0
```

A plot of the un-adjusted marker effect (from the `stat` data frame) against the adjusted marker effects (from the the `statAdj` data frame) illustrates that the C+T procedure keep only the most significant marker effects and is setting a large number of marker effects to zero (i.e. remove their effect).

```
plot(y = statAdj$b_0.05, col = 1, x = stat$b, xlab = "Marginal Effect",
    ylab = "Adjusted Effect", frame.plot = FALSE, ylim = c(-max(statAdj$b_0.05) *
        1.2, max(statAdj$b_0.05) * 1.2), xlim = c(-max(stat$b), max(stat$b)),
    main = "Shrinkage using C+T \n (p=0.01, r2=0.9)")
```



7 Genomic risk scoring using clumping and thresholding (C+T)

Genomic risk scoring using clumping and thresholding is a relative simple and robust approach. It is based on GWAS summary statistics and require the construction of a reference linkage disequilibrium (LD) correlation matrix. Ideally these will correspond to the most powerful GWAS results available on the phenotype under study. In this example, we will use the summary statistics output the ´glma´ function that fit a linear regression model on the quantitative trait. 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 genomic risk scores.

```
# Simulate phenotype
sim <- gsim(Glist = Glist, nt = 1)

y <- sim$y  # phenotype
X <- model.matrix(y ~ 1)  # design matrix

# Define training and validation data
train <- sample(names(y), 400)
valid <- names(y)[!names(y) %in% train]

# Compute summary statistics
stat <- glma(y = y[train], X = X[train, ], Glist = Glist, rsids = rsidsQC)

# Adjust summary statistics using clumping and p-value thresholding
statAdj <- adjStat(Glist = Glist, stat = stat, r2 = 0.9, threshold = c(0.01,
    0.05, 0.1))
```

Computing genomic risk scores (GRS)

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

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

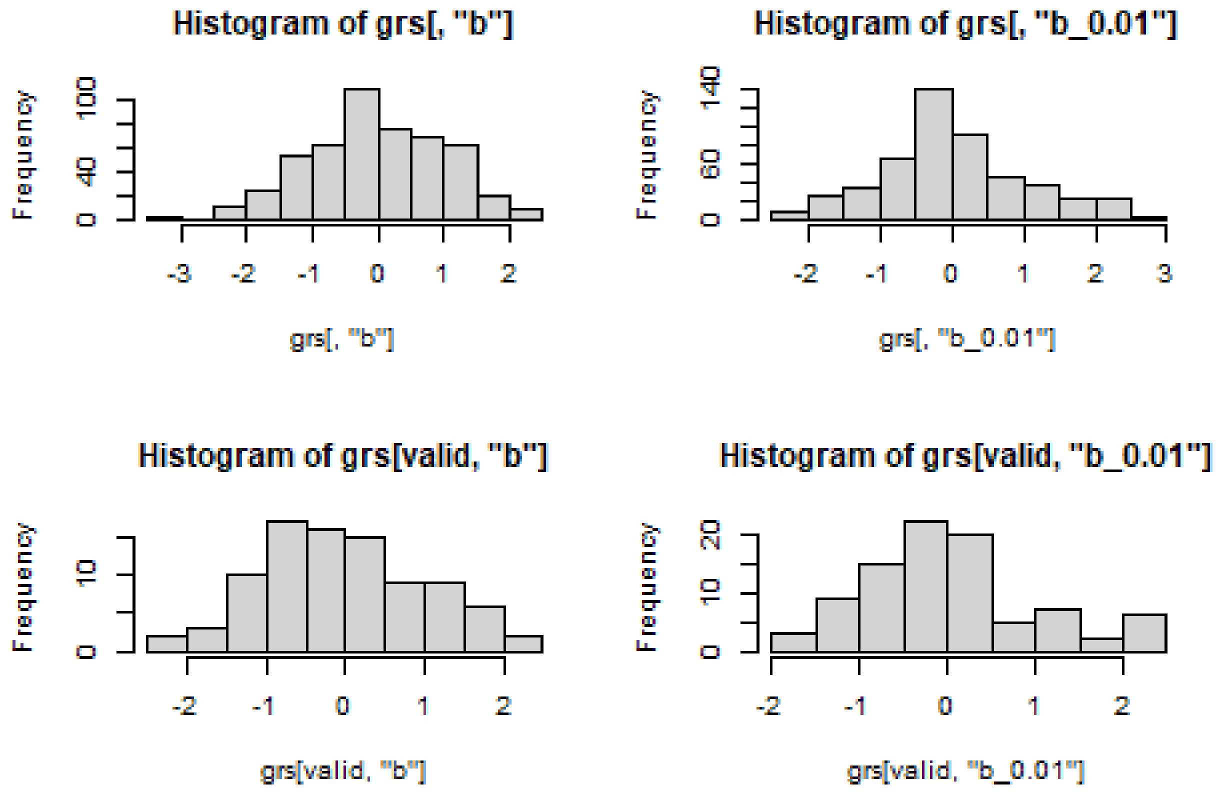
where X_i is the genotype vector, and $\hat{b}_i$ the weight of the i'th single genetic marker. The GRS are computed using the `gscore()` function. The input to the function is the adjusted summary statistic (`adjStat`), and information about the genotypes which are in the `Glist`:

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

```
##                  b     b_0.01     b_0.05      b_0.1
## HG00096 -1.6299645 -2.0846986 -2.4652260 -2.3963841
## HG00097  1.4399053  1.5338616  1.1443649  1.2957340
## HG00099  0.5787080  1.1663725  1.2344716  1.1510191
## HG00101 -0.6324188 -0.7944948 -1.1366107 -1.1847580
## HG00102  1.0609252 -0.4764918 -0.1308125  0.2384472
## HG00103  0.9682464  1.1836317  1.1902317  1.1388768
```

It is always important to explore the PRS computed using some simple plots.

```
layout(matrix(1:4, ncol = 2, byrow = TRUE))
hist(grs[, "b"])
hist(grs[, "b_0.01"])
hist(grs[valid, "b"])
hist(grs[valid, "b_0.01"])
```



Evalute genomic risk scores

The *P*-value threshold that provides the "best-fit" GRS under the C+T method is usually unknown. To approximate the "best-fit" GRS, we can perform a regression between GRS calculated at a range of *P*-value thresholds and then select the GRS that explains the highest proportion of phenotypic variance (e.g. R2) or has the highest AUC. This can be achieved using `acc()`-function as follows:

```
pa <- acc(yobs = y[valid], ypred = grs[valid, ])
pa
```

```
##          Corr    R2 intercept slope  MSPE
## b       0.078 0.006    -0.110 0.088 2.129
## b_0.01  0.241 0.058    -0.113 0.280 1.721
## b_0.05  0.227 0.051    -0.115 0.260 1.777
## b_0.1   0.212 0.045    -0.116 0.244 1.804
```

8 Gene mapping and estimation of genetic parameters using Bayesian Linear Regression (BLR) models

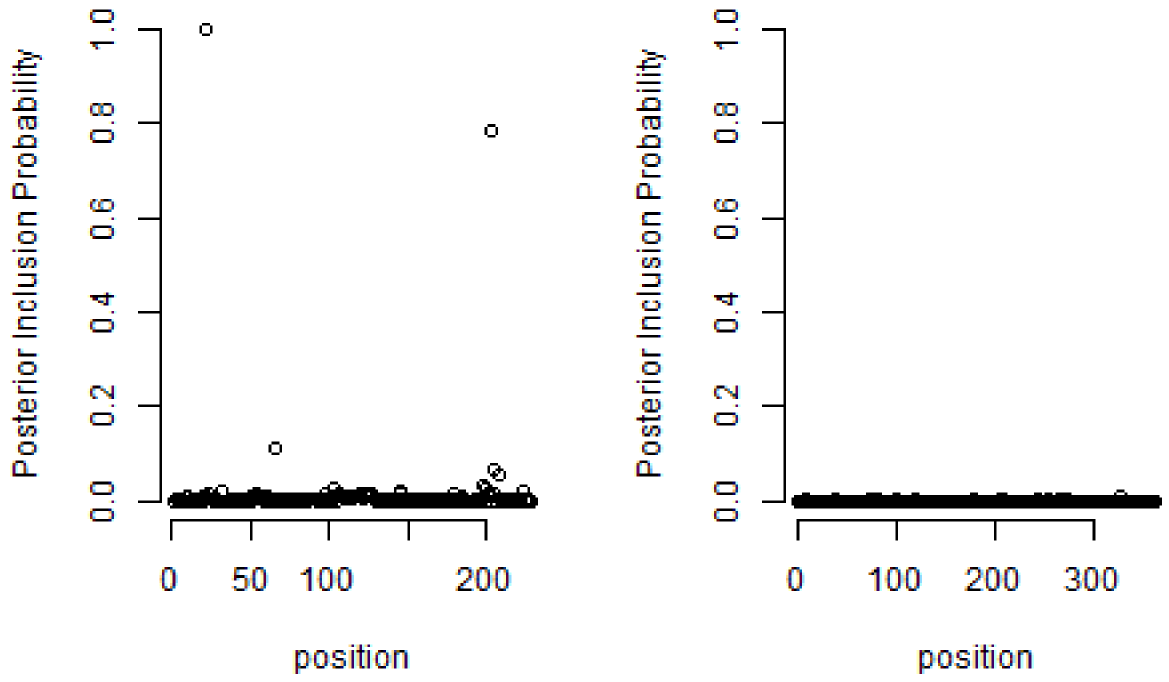
Bayesian linear regression models have been proposed as a unified framework for gene mapping, genomic risk scoring, estimation of genetic parameters and effect size distribution. BLR methods use an iterative algorithm for estimating joint marker effects that account for LD and allow for differential shrinkage of 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.

The different BLR models are implemented in the `qgg`-package in the function `gbayes()`, where the argument `method=` specifies which prior marker variance should be used. BLR models can be fitted using individual level genotype and phenotype data or be based on GWAS summary statistics and a reference linkage disequilibrium (LD) correlation matrix.

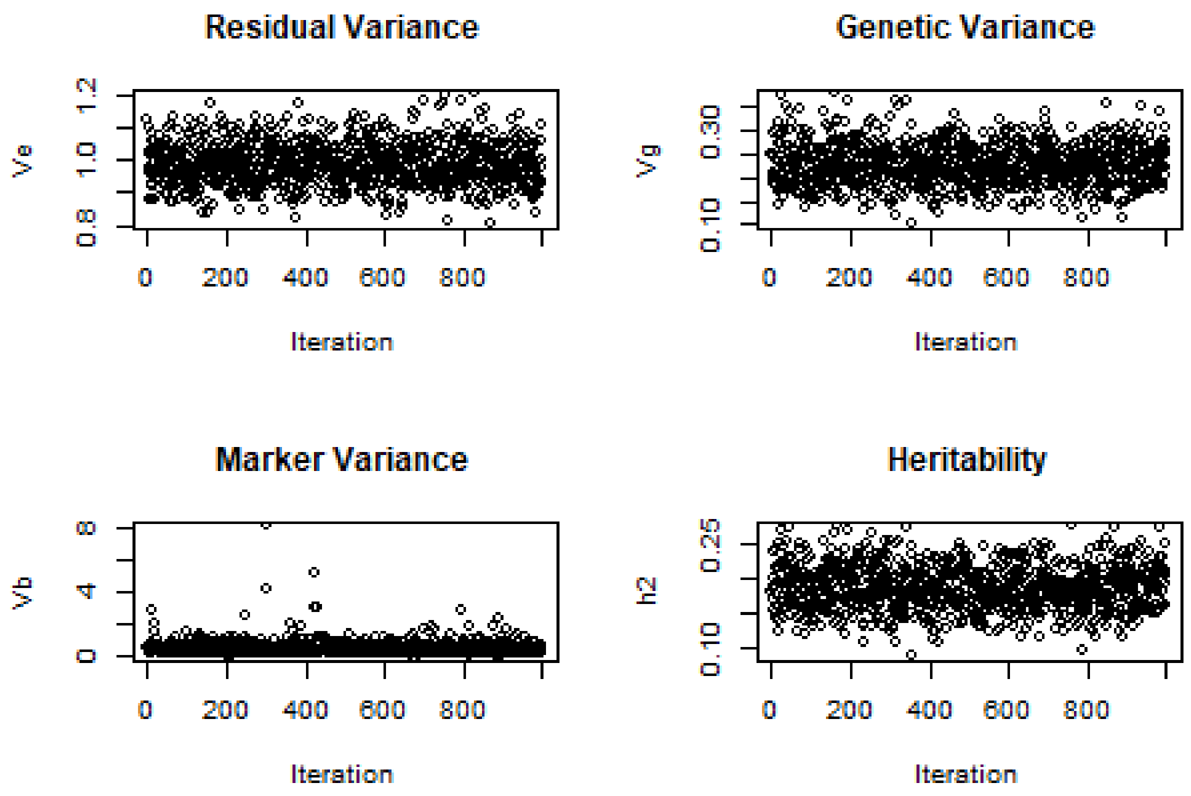
Fit single trait BLR models fitted using summary statistic data

In this example we will simulate a single quantitative trait and fit a BLR model using the Bayes C approach where 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 a common marker effect variance:

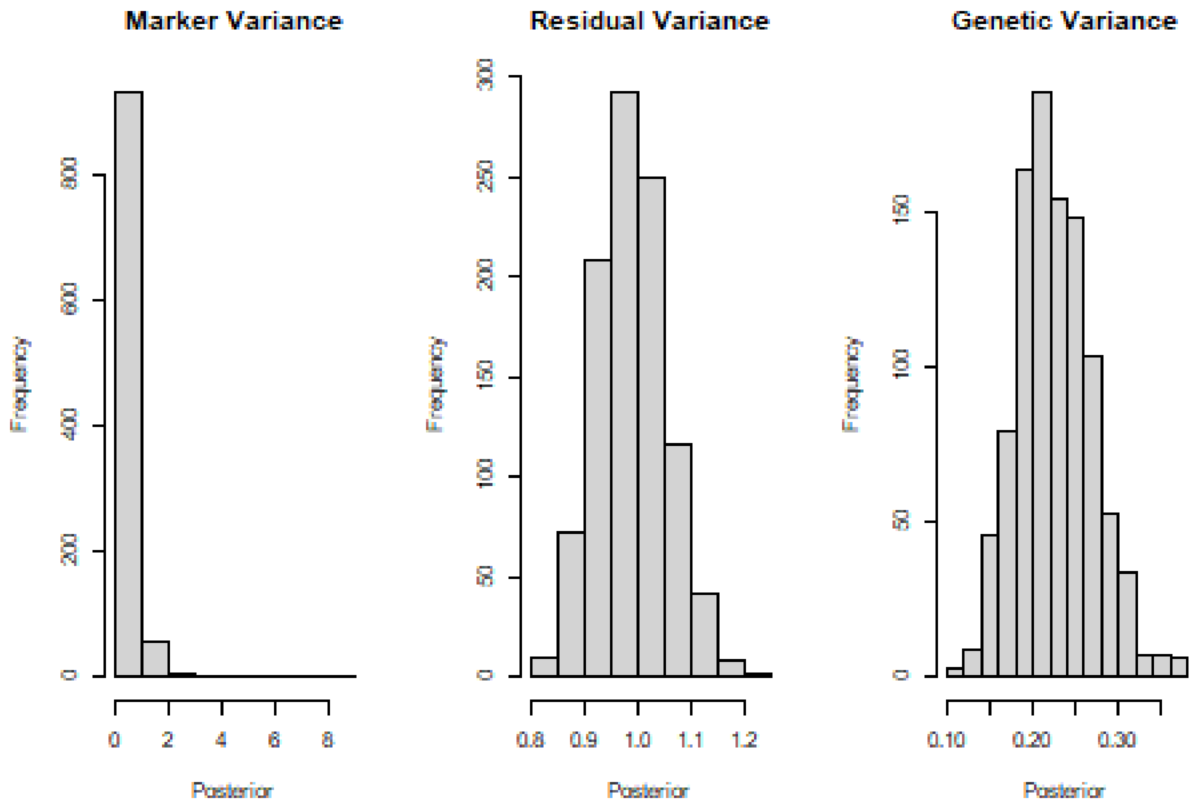
```r
# Simulate phenotype
sim <- gsim(Glist = Glist, nt = 1)

y <- sim$y  # phenotype
X <- model.matrix(y ~ 1)  # design matrix

# Compute summary statistics
stat <- glma(y = y, X = X, Glist = Glist, rsids = rsidsQC)

# Fit BLR model using summary statistics and sparse LD matrix
# information in Glist
fit <- gbayes(stat = stat, Glist = Glist, method = "bayesC", pi = 0.001,
    nit = 1000)

# Plot posterior inclusion probability for the two chromosomes
layout(matrix(1:2, ncol = 2))
plot(fit[[1]]$dm, ylab = "Posterior Inclusion Probability", xlab = "position",
    ylim = c(0, 1), frame.plot = FALSE)

plot(fit[[2]]$dm, ylab = "Posterior Inclusion Probability", xlab = "position",
    ylim = c(0, 1), frame.plot = FALSE)
```



```
# Trace plot of variance components
plotBayes(fit = fit, what = "trace", chr = 1)
```



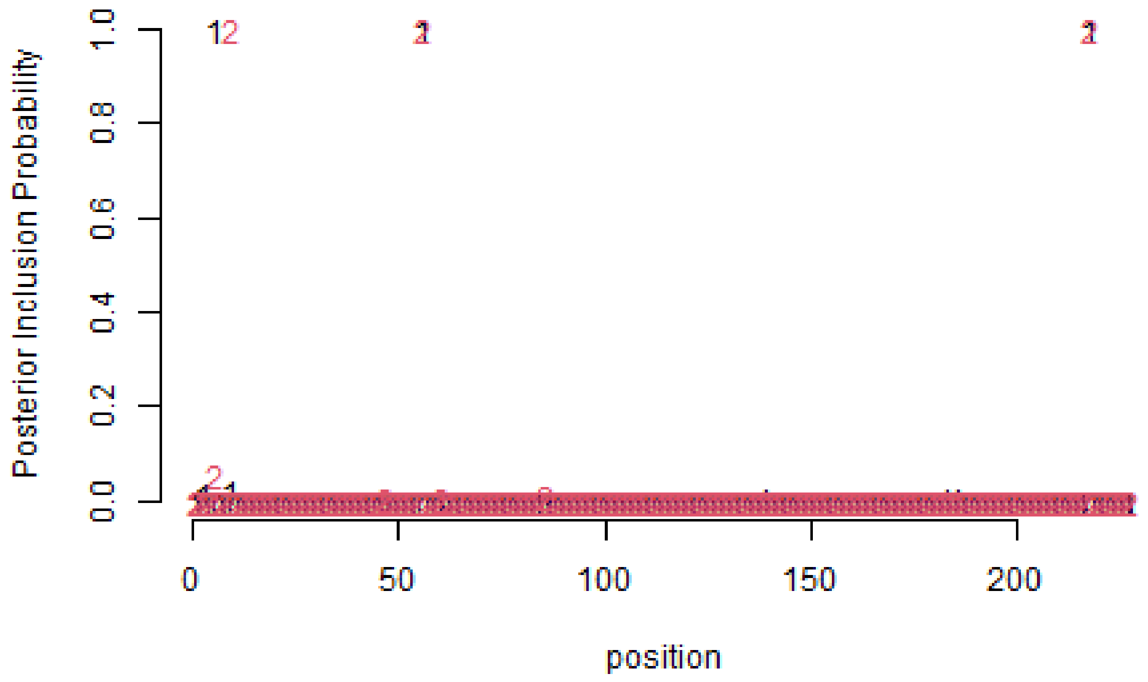
```r
# Plot of marker variance
layout(matrix(1:3, ncol = 3))
plotBayes(fit = fit, what = "vb", chr = 1)
plotBayes(fit = fit, what = "ve", chr = 1)
plotBayes(fit = fit, what = "vg", chr = 1)
```



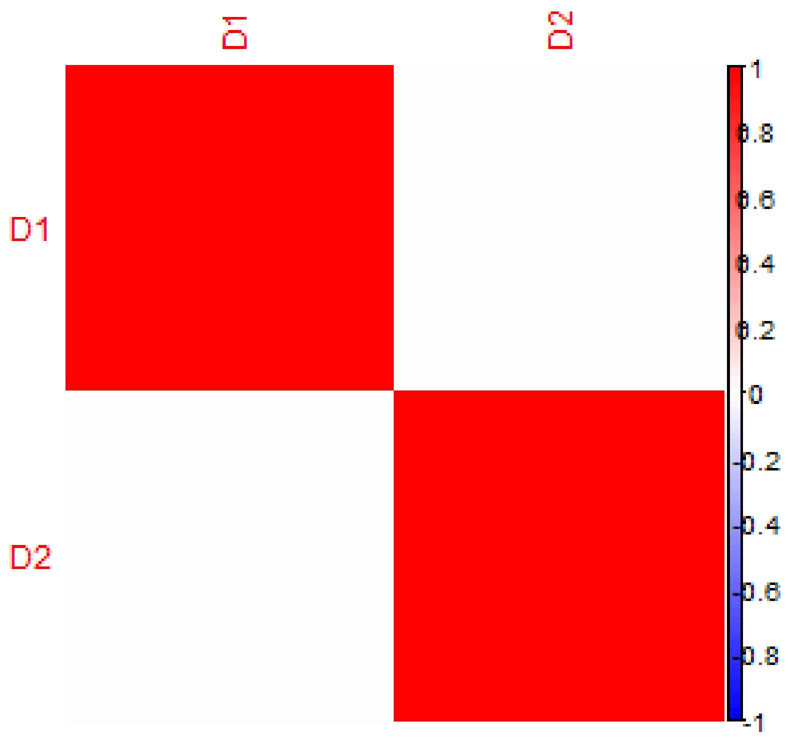
Fit multiple trait BLR models fitted using summary statistic data

In this example we will simulate two genetically correlated quantitative trait and fit a multiple trait BLR model using the Bayes C approach where marker effects, b, are a priori assumed to be sampled from a mixture with a point mass at zero and multivaiate normal distribution conditional on a common marker effect covariance:

```r
# Simulate phenotype
sim <- gsim(Glist = Glist, nt = 2)

y <- sim$y  # phenotype
X <- model.matrix(y[, 1] ~ 1)  # design matrix

# Compute summary statistics
stat <- glma(y = y, X = X, Glist = Glist)

# Fit BLR model using summary statistics and sparse LD matrix
# information in Glist
fit <- gbayes(stat = stat, Glist = Glist, method = "bayesC", updatePi = FALSE,
    updateE = FALSE, updateB = TRUE, nit = 1000)

# Plot posterior inclusion probability for the two traits
matplot(fit[[1]]$dm, col = 1:2, ylab = "Posterior Inclusion Probability",
    xlab = "position", ylim = c(0, 1), frame.plot = FALSE)
```



```r
# Plots marker effect correlation
library(corrplot)
library("RColorBrewer")

col <- colorRampPalette(c("blue", "white", "red"))
corrplot(fit[[1]]$rb, method = "color", bg = "white", outline = FALSE,
    col = col(100))
```


D1
D2
D1
D2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

9 Genomic risk scoring using Bayesian Linear Regression (BLR) models

Genomic risk scoring using Bayesian Linear Regression (BLR) models is a very flexible approach for accounting for the underlying genetic architecture of the traits. It can be implemented using GWAS summary statistics and a reference linkage disequilibrium (LD) correlation matrix. Ideally the summary statistics will correspond to the most powerful GWAS results available on the phenotype under study. In this example, we will use the summary statistics output the ´glma´ function that fit a linear regression model on the quantitative trait. 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 genomic risk scores.

Different BLR models can be fitted in the `qgg`-package in the function `gbayes()`, where the argument `method=` specifies which prior marker variance should be used. BLR models can be fitted using individual level genotype and phenotype data or based on GWAS summary statistics and a reference linkage disequilibrium (LD) correlation matrix.

```r
# Simulate phenotype
sim <- gsim(Glist = Glist, nt = 1)

y <- sim$y  # phenotype
X <- model.matrix(y ~ 1)  # design matrix

# Define training and validation data
train <- sample(names(y), 450)
valid <- names(y)[!names(y) %in% train]

# Compute summary statistics
stat <- glma(y = y[train], X = X[train, ], Glist = Glist)
```

Fit BLR model using a Bayes C prior for the marker variance

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 a common marker effect variance:

```r
fit <- gbayes(stat = stat, Glist = Glist, method = "bayesC", nit = 1000)
head(fit$stat)
```

```
##                   rsids chr     pos ea nea        eaf           bm    pm
## rs560838421 rs560838421   1 9442544  G   A 0.06237219 0.0001251063 0.002
## rs561578877 rs561578877   1 9450695  T   C 0.11963190 0.0001643679 0.002
## rs575349295 rs575349295   1 9455527  T   C 0.17075665 0.0016884162 0.015
## rs551139107 rs551139107   1 9466700  T   C 0.08179959 0.0002826718 0.007
## rs570124710 rs570124710   1 9471511  T   C 0.06850716 0.0003598147 0.004
## rs558507375 rs558507375   1 9471670  G   A 0.05828221 0.0052080605 0.045
##                       vm
## rs560838421 1.830669e-09
## rs561578877 5.690826e-09
## rs575349295 8.073254e-07
## rs551139107 1.200282e-08
## rs570124710 1.652354e-08
## rs558507375 2.977411e-06
```

```r
grs <- gscore(Glist = Glist, stat = fit$stat)
pa <- acc(yobs = y[valid], ypred = grs[valid, ])
pa
```

```
##      Corr    R2 intercept slope  MSPE
## res 0.198 0.039     0.262 0.225 1.871
```

Fit BLR model using a Bayes R prior for the marker variance

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 a common marker effect variance:

```r
fit <- gbayes(stat = stat, Glist = Glist, method = "bayesR", nit = 1000)
head(fit$stat)
```

```
##                   rsids chr     pos ea nea        eaf          bm    pm
## rs560838421 rs560838421   1 9442544  G   A 0.06237219 0.002357652 0.106
## rs561578877 rs561578877   1 9450695  T   C 0.11963190 0.003897447 0.110
## rs575349295 rs575349295   1 9455527  T   C 0.17075665 0.012282142 0.245
## rs551139107 rs551139107   1 9466700  T   C 0.08179959 0.005808456 0.147
## rs570124710 rs570124710   1 9471511  T   C 0.06850716 0.002168158 0.095
## rs558507375 rs558507375   1 9471670  G   A 0.05828221 0.041266449 0.613
##                       vm
## rs560838421 6.501460e-07
## rs561578877 3.199645e-06
## rs575349295 4.272065e-05
## rs551139107 5.068040e-06
## rs570124710 5.999669e-07
## rs558507375 1.869308e-04
```

```r
grs <- gscore(Glist = Glist, stat = fit$stat)
pa <- acc(yobs = y[valid], ypred = grs[valid, ])
pa
```

```
##      Corr    R2 intercept slope  MSPE
## res 0.208 0.043      0.29 0.284 1.691
```

10 Gene Set Enrichment Analysis

The general procedure gene set enrichment analysis is to obtain single marker statistics (e.g. summary statistics), from which it is possible to compute and evaluate a test statistic for a set of genetic markers that measures a joint degree of association between the marker set and the phenotype. The marker set is defined by biologically informed entities, such as genes, biological pathways, protein interaction complexes etc., and analysing those sets of markers jointly constitute a valuable addition to single-marker analyses. The ´gsea´ function in qgg can perform four different gene set enrichment analyses. Here we show how to perform enrichment analysis based on summary statistics from a standard single marker regression model or based on a multiple marker Bayesian Linear Regression (BLR) model.

```r
# Simulate phenotype
sim <- gsim(Glist = Glist, nt = 1)

y <- sim$y  # phenotype
X <- model.matrix(y ~ 1)  # design matrix
```

Gene set enrichment analysis using standard approach

In the first example, we will use the summary statistics output the ´glma´ function that fit a linear regression model on the quantitative trait followed by the clumping and thresholding procedure. This is important as 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.

```r
# Compute summary statistics
stat <- glma(y = y, X = X, Glist = Glist)

# Marker sets defined by chromosomes
sets <- Glist$rsidsLD

# Adjust for LD
statAdj <- adjStat(stat = stat, Glist = Glist, r2 = 0.9, threshold = c(0.001,
    0.01, 0.05))

# Gene set enrichment analysis
setstat <- gsea(stat = statAdj, sets = sets)
setstat
```

```
## $m
## Set1 Set2
##  228  362
##
## $stat
##              b   b_0.001     b_0.01     b_0.05
## Set1 1.4818174 0.3242214 0.55934405 0.76302602
## Set2 0.4586276 0.0000000 0.01750526 0.01750526
##
## $p
##          b b_0.001 b_0.01 b_0.05
## Set1 0.009   0.000  0.000  0.000
## Set2 0.755   0.583  0.651  0.723
```

Gene set enrichment analysis using BLR model approach

In this example, we will use the summary statistics output the ´glma´ function that fit a linear regression model on the quantitative trait followed by the BLR model approach. The BLR models account for the linkage disequilibrium and the underlying genetic architecture of the traits. In this example we use the Bayes C prior for the marker effects, b, which a priori are assumed to be sampled from a mixture with a point mass at zero and univariate normal distribution conditional on a common marker effect variance:

```
# Compute summary statistics
stat <- glma(y = y, X = X, Glist = Glist)

# Marker sets defined by chromosomes
sets <- Glist$rsidsLD

# Fit BLR model
fit <- gbayes(stat = stat, Glist = Glist, pi = 0.001, method = "bayesC")

# Gene set enrichment analysis
setstat <- gsea(stat = fit$stat, sets = sets)
setstat
```

```
## $m
## Set1 Set2
##  228  362
##
## $stat
##                bm     pm           vm
## Set1 8.851347e-02 1.5827 2.347553e-04
## Set2 4.217775e-06 0.0014 3.037543e-13
##
## $p
##      bm pm    vm
## Set1  0  0 0.021
## Set2  1  1 0.863
```