Practical 5 Summary-data-based methods for polygenic prediction

Winter School 2022 Module 4 Quantitative Genetics

2022-06-23

In this practical you will perform polygenic prediction using SBLUP and SBayesR, the variations of BLUP and BayesR that only reauire GWAS summary statistics. We will use the same data sets in the session of individual-level BLUP and BayesR analysis.

Analysis of a small data set using R script

Load data in R.

```
nmarkers <- 10       #number of markers
nrecords <- 325      #number of records

x <- matrix(scan("/data/module4/prac5/xmat.inp"),ncol=nmarkers,byrow=TRUE)
y <- matrix(scan("/data/module4/prac5/yvec.inp"),byrow=TRUE)
x_prog <- matrix(scan("/data/module4/prac5/xmat_prog.inp"),ncol=nmarkers,byrow=TRUE)
y_prog <- matrix(scan("/data/module4/prac5/yvec_prog.inp"),byrow=TRUE)
```

Obtain GWAS summary data

First, let us assume the genotype matrix is standardised (scaled) to have mean zero and variance one in each column. Run GWAS using the standardised genotypes:

```
x_scaled = apply(x, 2, scale)
# run GWAS on the scaled genotypes
fit = apply(x_scaled, 2, function(x){summary(lm(y~x))$coefficients[2,1:2]})
b_scaled = fit[1,]
```

Compute LD correlation matrix

```
R = cor(x)
```

SBLUP

Run SBLUP using marginal SNP effects from GWAS and LD correlation matrix:

```
lambda = 10
I = diag(nmarkers)
coeff = R + I*lambda/nrecords
rhs = b_scaled
beta_sblup = solve(coeff, rhs)
```

BLUP

Run BLUP using the individual-level data as benchmark:

```
lambda = 10
I = diag(nmarkers+1)
X = cbind(1, x_scaled)
coeff = crossprod(X) + I*lambda
rhs = crossprod(X, y)
beta_blup = solve(coeff, rhs)[-1]
```

Question 1 Are BLUP and SBLUP solutions the same?

```
cor(beta_blup, beta_sblup)
```

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

In practice, GWAS is often done with unscaled genotypes (i.e., coded as 0, 1, 2). In this case, we need to first scale the marginal effects and then unscale the joint effect estimates to put them back to the scale of per-allele effects.

```
# run GWAS on the 0/1/2 genotypes
fit = apply(x, 2, function(x){summary(lm(y~x))$coefficients[2,1:2]})
b = fit[1,]
se = fit[2,]
# calculate the scale factor for each SNP (i.e., sqrt(heterozygosity/vary)),
# assuming each SNP effect is vanishingly small
scale = sqrt(1/(nrecords*se^2))
# scale the marginal effects
b_scaled = b*scale
```

Perform SBLUP using scaled marginal effects

```
lambda = 10
I = diag(nmarkers)
coeff = R + I*lambda/nrecords
rhs = b_scaled
beta_sblup_scaled = solve(coeff, rhs)
beta_sblup_unscaled = beta_sblup_scaled/scale
```

Then we can predict GEBV of selection candidates.

```
# get the predicted genetic value (GEBV)
ghat=x_prog%*%beta_sblup_unscaled

# prediction R-square
summary(lm(y_prog~ghat))$r.squared
```

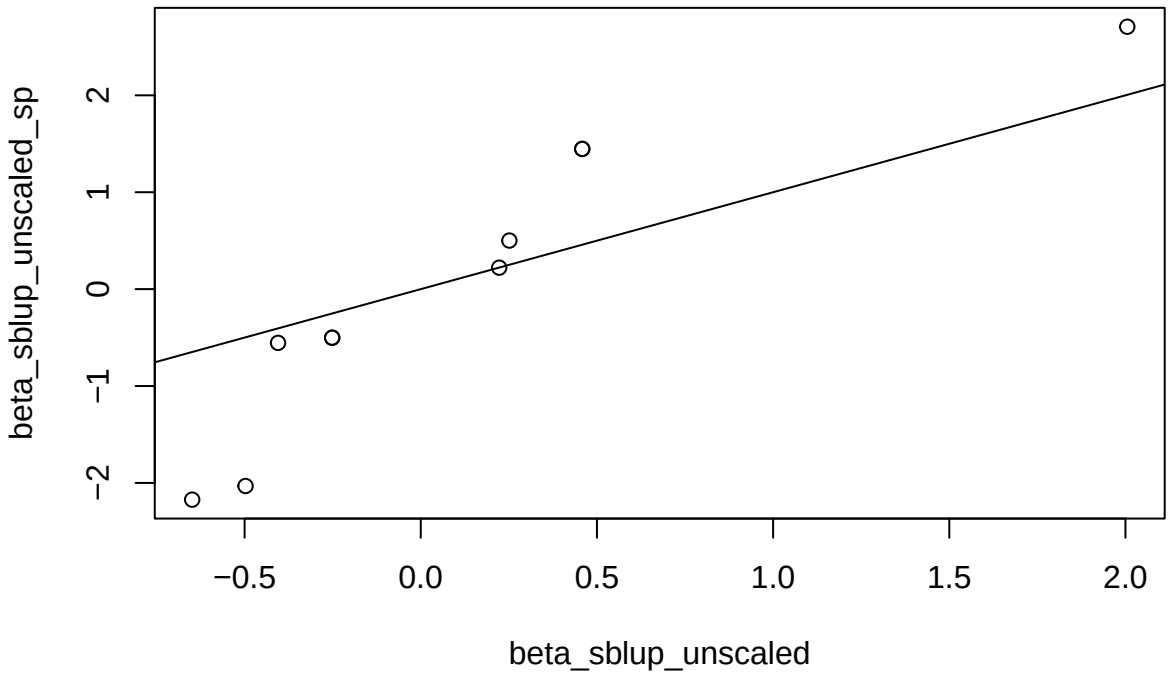
```
## [1] 0.589458
```

In practice, we often use a sparse LD correlation matrix that removes LD correlations that are almost zero. Let's see how this would influence the BLUP performance.

```
# arbitrarily set a LD threshold such that R becomes sparse
R_sp = R
R_sp[abs(R_sp)<0.2] = 0
```

```
lambda = 10
I = diag(nmarkers)
coeff = R_sp + I*lambda/nrecords
rhs = b_scaled
beta_sblup_scaled = solve(coeff, rhs)
beta_sblup_unscaled_sp = beta_sblup_scaled/scale
```

```
plot(beta_sblup_unscaled, beta_sblup_unscaled_sp)
abline(a=0, b=1)
```



SBayesR

SBayesR method is implemented in `sbayesr.R`. We also include the BayesR method in `bayesr.R`. Have a look at the code and see what the differences are.

```
source("/data/module4/prac5/sbayesr.R")
```

Run SBayesR. Note that here we are using the marginal effects from GWAS with unstandardised genotypes as input data, and the scaling factors are computed using standard errors.

```
sr.res = sbayesr(b, se, nrecords, R)
```

```
##
##  iter  100, nnz =    6, sigmaSq =  1.324, h2 =  0.694, vare =  1.000
##
##  iter  200, nnz =    2, sigmaSq =  1.806, h2 =  0.546, vare =  1.000
##
##  iter  300, nnz =   10, sigmaSq =  1.552, h2 =  0.776, vare =  1.000
##
##  iter  400, nnz =    9, sigmaSq =  1.973, h2 =  0.843, vare =  1.000
##
##  iter  500, nnz =    4, sigmaSq =  4.521, h2 =  0.768, vare =  1.000
##
##  iter  600, nnz =    8, sigmaSq =  0.714, h2 =  0.838, vare =  1.000
##
##  iter  700, nnz =    8, sigmaSq =  6.703, h2 =  0.653, vare =  1.000
##
##  iter  800, nnz =    4, sigmaSq =  1.479, h2 =  0.860, vare =  1.000
##
##  iter  900, nnz =    6, sigmaSq =  1.776, h2 =  1.012, vare =  1.000
```

```
##
##  iter 1000, nnz =    9, sigmaSq =  3.481, h2 =  0.643, vare =  1.000
##
## Posterior mean:
##       Pi1       Pi2       Pi3       Pi4       Nnz   SigmaSq        h2      Vare
## 0.3229407 0.3158747 0.2114597 0.1497249 6.4550000 2.2029482 0.7962096 1.0000000
```

```
beta.sr = colMeans(sr.res$beta)
```

Run BayesR as benchmark:

```
source("/data/module4/prac5/bayesr.R")
```

```
r.res = bayesr(x, y)
```

```
##
##  iter  100, nnz =    7, sigmaSq =  1.126, h2 =  0.479, vare =  2.492, varg =  2.291
##
##  iter  200, nnz =   10, sigmaSq =  3.623, h2 =  0.475, vare =  3.125, varg =  2.833
##
##  iter  300, nnz =    9, sigmaSq =  0.920, h2 =  0.373, vare =  3.707, varg =  2.208
##
##  iter  400, nnz =    8, sigmaSq =  1.837, h2 =  0.427, vare =  2.674, varg =  1.995
##
##  iter  500, nnz =    6, sigmaSq =  3.713, h2 =  0.430, vare =  2.867, varg =  2.161
##
##  iter  600, nnz =    6, sigmaSq =  4.808, h2 =  0.516, vare =  2.876, varg =  3.070
##
##  iter  700, nnz =    4, sigmaSq =  3.861, h2 =  0.420, vare =  2.787, varg =  2.017
##
##  iter  800, nnz =   10, sigmaSq =  7.312, h2 =  0.455, vare =  3.154, varg =  2.632
##
##  iter  900, nnz =    3, sigmaSq =  1.872, h2 =  0.403, vare =  3.127, varg =  2.107
##
##  iter 1000, nnz =    5, sigmaSq =  1.998, h2 =  0.465, vare =  2.586, varg =  2.252
##
## Posterior mean:
##       Pi1       Pi2       Pi3       Pi4       Nnz   SigmaSq        h2      Vare
## 0.2237633 0.2863551 0.2598939 0.2299876 7.8200000 2.7732762 0.4550046 2.8230172
##      Varg
## 2.3650496
```

```
beta.r = colMeans(r.res$beta)
```

Question 2: Are BayesR and SBayesR SNP effect estimates the same? What could possibly cause the difference? (hint: the scaling process implies an assumption of a negligible effect size for each individual SNP.)

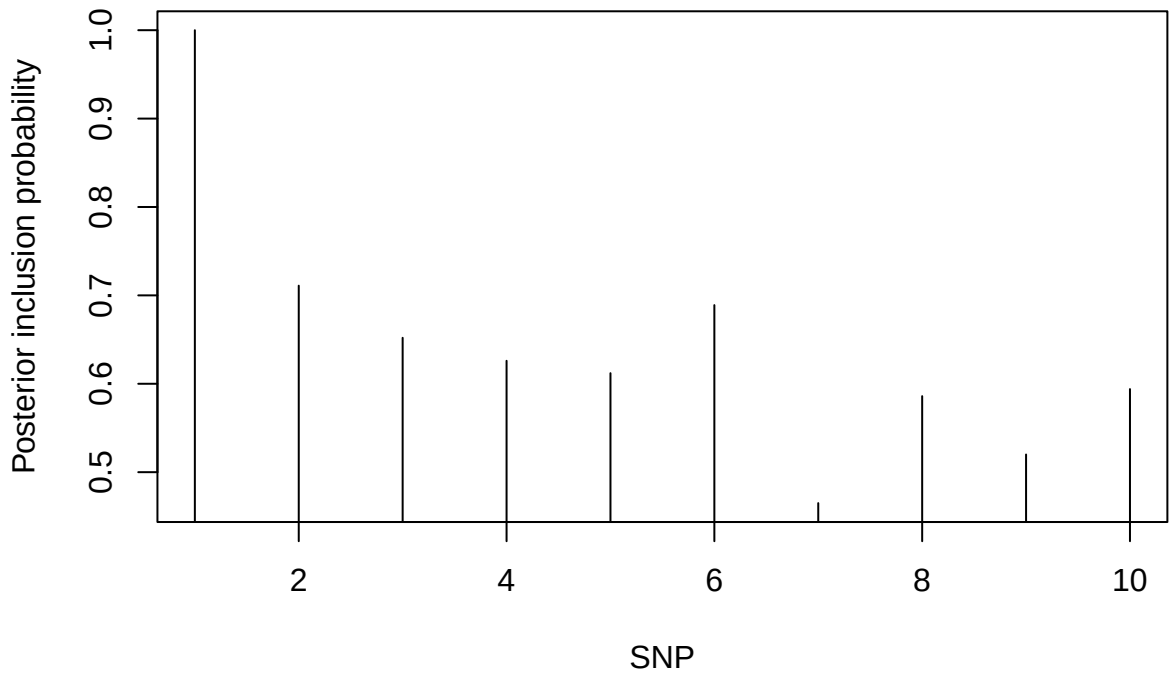
```
cor(beta.r, beta.sr)
```

```
## [1] 0.9634846
```

Question 3: Can you tell which SNPs are likely to be the causal variants based on the posterior inclusion probability (PIP)? PIP is a Bayesian measurement for SNP-trait association, which is calculated as the frequency of the SNP being fitted as a non-zero effect in the model across MCMC samples.

```
delta = (sr.res$beta != 0)
pip = colMeans(delta)
```

```
plot(pip, type="h", xlab="SNP", ylab="Posterior inclusion probability")
```



Analysis of a larger data set using GCTA and GCTB

Run SBLUP using GCTA

Command manual can be found at https://yanglab.westlake.edu.cn/software/gcta/#SBLUP.

DO NOT run this code in the practical because it will take a while. The result generated from this code is in our practical folder.

```
bfile="/data/module4/prac5/gwas"
ma="/data/module4/prac5/simu.ma"
gcta --bfile $bfile \
     --cojo-file $ma \
     --cojo-sblup 277719 \
     --cojo-wind 1000 \
     --thread-num 8 \
     --out simu
```

The SNP results are stored in file `simu.sblup.cojo`. You can use these SBLUP estimates for polygenic prediction. This code is quick to run.

```
target="/data/module4/prac5/target"
sblup="/data/module4/prac5/simu.sblup.cojo"
plink --bfile $target --score $sblup 1 2 4 sum center --out simu.sblup

phenFile="/data/module4/prac5/simu.phen"
covFile="/data/module4/prac5/covariates.cov"
indlistFile="/data/module4/prac5/target.indlist"
prsFile="simu.sblup.profile"
Rscript /data/module4/prac5/get_pred_r2.R $phenFile $covFile $indlistFile $prsFile
```

Question 4: How does it compare to the BLUP result using individual-level data?

Run SBayesR using GCTB

A tutorial for using GCTB to run SBayesR can be found at https://cnsgenomics.com/software/gctb/#Tutorial.

Step 1: get GWAS summary data. GCTB also uses `.ma` file to input GWAS summary data.

Step 2: compute LD matrix. DO NOT run this step in the practical because it can take a while. Instead, use the data that we have generated for you as shown in the next step.

GCTB provides options to compute different types of LD matrix. In our SBayesR paper (https://doi.org/10.1038/s41467-019-12653-0), we use shrunk LD matrix, which can be built by the following command.

```
# Build shrunk LD matrix for each chromosome
bfile="/data/module4/prac5/gwas"
genmap="/data/module4/prac5/interpolated_genetic_map.txt"
> mldm.txt
for i in `seq 1 22`
  do
    gctb --bfile $bfile \
         --chr $i \
         --make-shrunk-ldm \
         --gen-map $genmap \
         --out chr$i
    echo "/data/module4/prac5/chr$i.ldm.shrunk" >> mldm.txt
done
```

In practice, if you are using summary statistics from GWAS of European ancestry, we recommend use of Banded matrix or Shrunk sparse matrix, which are computed using 1M HapMap3 SNPs in the UKB and can be download from our website (https://cnsgenomics.com/software/gctb/#Download).

Step 3: Run SBayesR SBayesR can be carried out using command below. It is **not recommended** to run this genome-wide analysis in this practical session.

```
ma="/data/module4/prac5/simu.ma"
mldm="/data/module4/prac5/mldm.txt"
gctb --sbayes R \
     --mldm $mldm \
     --gwas-summary $ma \
     --original-model \
     --chain-length 1000 \
     --burn-in 200 \
     --out sbayesr
```

If you are interested, you can just run with SNPs on chromosome 1 using the command below:

```
ma="/data/module4/prac5/simu.ma"
ldm="/data/module4/prac5/chr1.ldm.shrunk"
gctb --sbayes R \
     --ldm $ldm \
     --gwas-summary $ma \
     --original-model \
     --chain-length 1000 \
     --burn-in 200 \
     --out sbayesr
```

Result file `sbayesr.parRes` shows the posterior estimates of model parameters. Result file `sbayesr.snpRes` shows the estimates of joint SNP effects.

Compute PRS using PLINK You can run polygenic prediction using the effect estimates that have been generated for genome-wide SNPs. They are in `/data/module4/prac5/sbayesr.snpRes`.

```
target="/data/module4/prac5/target"
snpRes="/data/module4/prac5/sbayesr.snpRes"
plink --bfile $target --score $snpRes 2 5 8 header sum center --out simu.bayesr

phenFile="/data/module4/prac5/simu.phen"
covFile="/data/module4/prac5/covariates.cov"
indlistFile="/data/module4/prac5/target.indlist"
prsFile="simu.bayesr.profile"
Rscript /data/module4/prac5/get_pred_r2.R $phenFile $covFile $indlistFile $prsFile
```

Question 5: How many SNPs are fitted with non-zero effects? Is the number consistent with the number of causal variants (1,000) in the simulation? Is the prediction accuracy better than C+PT and SBLUP?

```
cat /data/module4/prac5/sbayesr.parRes
```

```
## Posterior statistics from MCMC samples:
##
##               Mean            SD
##    NumSnp1    275744.968750   377.253632
##    NumSnp2    414.148743      408.834747
##    NumSnp3    854.283752      74.896935
##    NumSnp4    10.565000       9.237466
##        Vg1    0.000000        0.000000
##        Vg2    0.041509        0.040976
##        Vg3    0.903924        0.060739
##        Vg4    0.054568        0.047466
##    SigmaSq    8.946647        0.583960
##     ResVar    1067.291382     51.483097
##     GenVar    865.669373      54.767822
##        hsq    0.447784        0.026227
```

`NumSnp1-NumSnp4` are the numbers of SNPs in the mixture components 1 to 4 (component 1: zero effect; component 2: small effects that explain 0.01% heritability; component 3: medium effects that explain 0.1% heritability; component 4: large effects that explain 1% heritability). `Vg1-Vg4` are the proportions of variance explained by the SNPs in each component. `hsq` is the SNP-based heritability estimate.