Genomic Prediction

Contents

1 SNP Best Linear Unbiased Prediction

We use snpBLUP to simultaneously estimate all SNP effects, despite what happens in single SNP regression analyses.

1.1 Import data

```
# Import data
gwas = readRDS("Data/gwasData.rds")
pheno = read.table("Data/phenotypes.txt", header = T, sep = "\t")$Pheno
map = read.table("Data/map.txt", header = T, sep = "\t")
```

There are 10,000 SNP (rows) for 2,000 individuals (columns) with the SNP coded as 0, 1 and 2. Also, there are no missing genotypes, the population is completely unrelated, genotypes are in linkage equilibrium, allele frequencies are roughly the same at 0.5, and there are **ten true QTL** in the data which are actual SNP.

```
# Check genotype data
gwas[1:4, 1:4]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    2    0
## [2,]    1    1    0    1
## [3,]    1    2    1    2
## [4,]    0    2    0    1
```

The phenotypes are mean centered and all known fixed effects have been accounted for.

```
# Check phenotype data
round(pheno[1:4], 3)
```

```
## [1] 5.814 4.894 3.119 0.993
```

The SNP are already ordered by map position.

```
# Check map data
map[1:4, ]
```

```
##   snp chrom loc
## 1   1     1   1
## 2   2     1   2
## 3   3     1   3
## 4   4     1   4
```

1.2 Single SNP regression

```
# Build a variable for effect sizes (coefficients)
effect = numeric(10000)

# Build a variable for p-values
pval = numeric(10000)

# Perform regression with lm for one SNP and check the output
summary(lm(formula = pheno ~ gwas[1, ]))
```

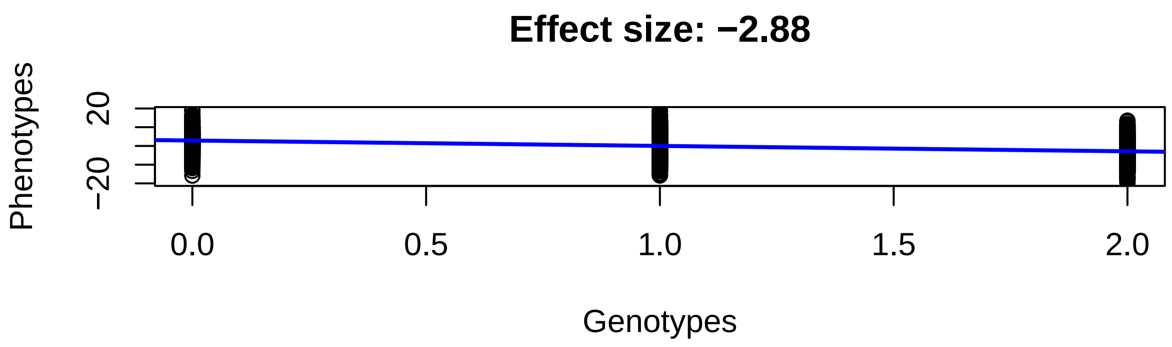
```
##
## Call:
## lm(formula = pheno ~ gwas[1, ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.2988  -4.5255  -0.0773   4.3939  19.1710
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.5069     0.2512   2.018   0.0437 *
## gwas[1, ]    -0.4989     0.2035  -2.451   0.0143 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.376 on 1998 degrees of freedom
## Multiple R-squared:  0.002999,   Adjusted R-squared:  0.0025
## F-statistic:  6.01 on 1 and 1998 DF,  p-value: 0.01431
```

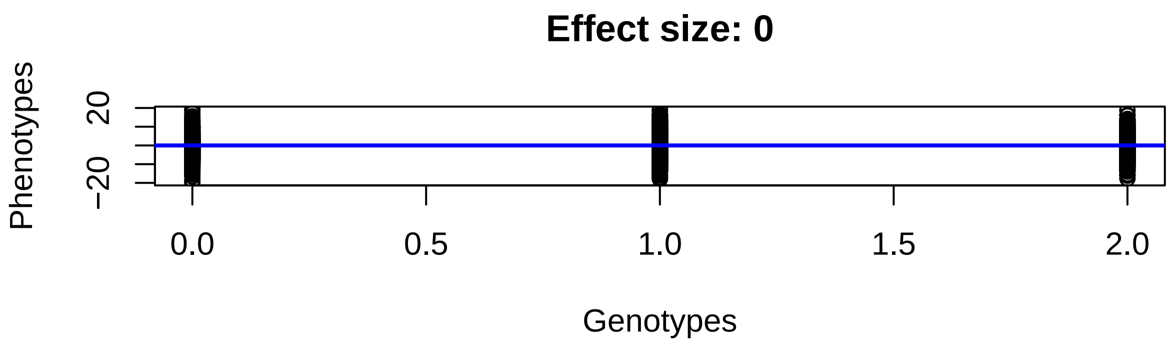
```
# Perform regression with lm for the genotype of all SNPs
for (i in 1:10000) {
  res = coef(summary(lm(formula = pheno ~ gwas[i, ])))[2, c(1, 4)]
  effect[i] = res[1]
  pval[i] = res[2]
}
```

In GWAS terms, it translates to: how much does phenotype (pheno) change when genotype (gwas) changes. Genotypes are now coded as numeric values according to a predefined mode of inheritance that the researcher wants to test; an additive model that assumes that alleles have a dose effect: one allelic variant has no effect (e.g., A) on the trait and the other has an effect (e.g., B). A heterozygous individual (AB) will express the effect and a homozygous individual with two copies of the allele (BB) will express twice the value of the effect. This naturally suggests coding SNP genotypes as 0 (AA), 1 (AB), and 2 (BB).

Now, let's compare the results with the highest and lowest p-values:

```
# Split screen for two plots
par(mfrow = c(2, 1))

# Most significant SNP
most_sig_snp = which(pval == min(pval))
plot(
  gwas[most_sig_snp, ],
  pheno,
  xlab = "Genotypes",
  ylab = "Phenotypes",
  main = paste("Effect size:", round(effect[most_sig_snp], 2))
)
mod = lm(pheno ~ gwas[most_sig_snp, ])
abline(mod, lwd = 2, col = "blue")

# Least significant SNP
least_sig_snp = which(pval == max(pval))
plot(
  gwas[least_sig_snp, ],
  pheno,
  xlab = "Genotypes",
  ylab = "Phenotypes",
  main = paste("Effect size:", round(effect[least_sig_snp], 2))
)
mod = lm(pheno ~ gwas[least_sig_snp, ])
abline(mod, lwd = 2, col = "blue")
```





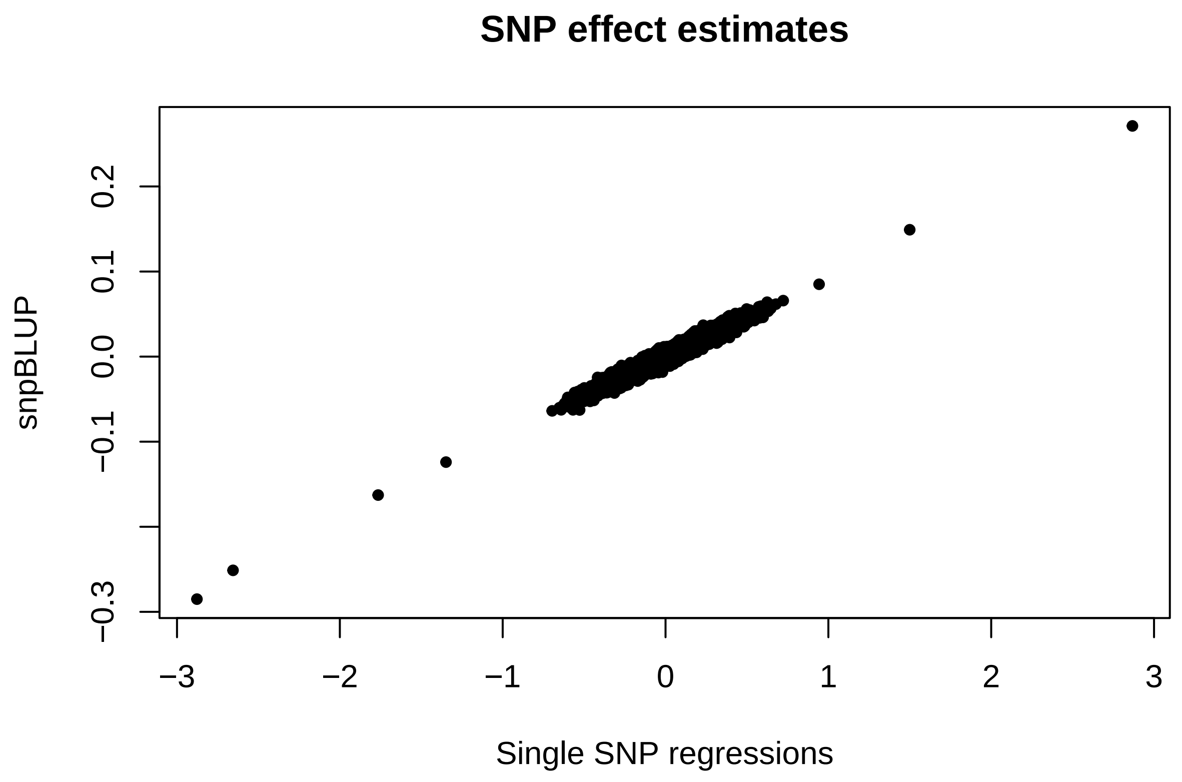
1.3 snpBLUP

It involves fitting SNP as random effects. While snpBLUP solves the problem of multiple testing and over estimation of SNP effects, it does strongly shrink back the estimates of SNP effects.

```
# Heritability
h2 = 0.5

# Frequency of second allele
p = rowMeans(gwas) / 2

# Lambda
d = 2 * sum(p * (1 - p))
lambda = (1 - h2) / (h2 / d)

# snpBLUP with fitting the mean
## y = pheno
## X = t(gwas - (p * 2))
## XtX = t(X) %*% X
## diag(XtX) = diag(XtX) + lambda
## ones = rep(1, length(y))
## oto = t(ones) %*% ones
## otX = t(ones) %*% X
## Xto = t(X) %*% ones
## lhs = rbind(cbind(oto, otX), cbind(Xto, XtX))
## oty = t(ones) %*% y
## Xty = t(X) %*% y
## rhs = rbind(oty, Xty)
## effectBLUP = solve(lhs) %*% rhs

# snpBLUP without fitting the mean
y = pheno
X = t(gwas - (p * 2))
XtX = t(X) %*% X
diag(XtX) = diag(XtX) + lambda
Xty = t(X) %*% pheno
effectBLUP2 = solve(XtX) %*% Xty
```

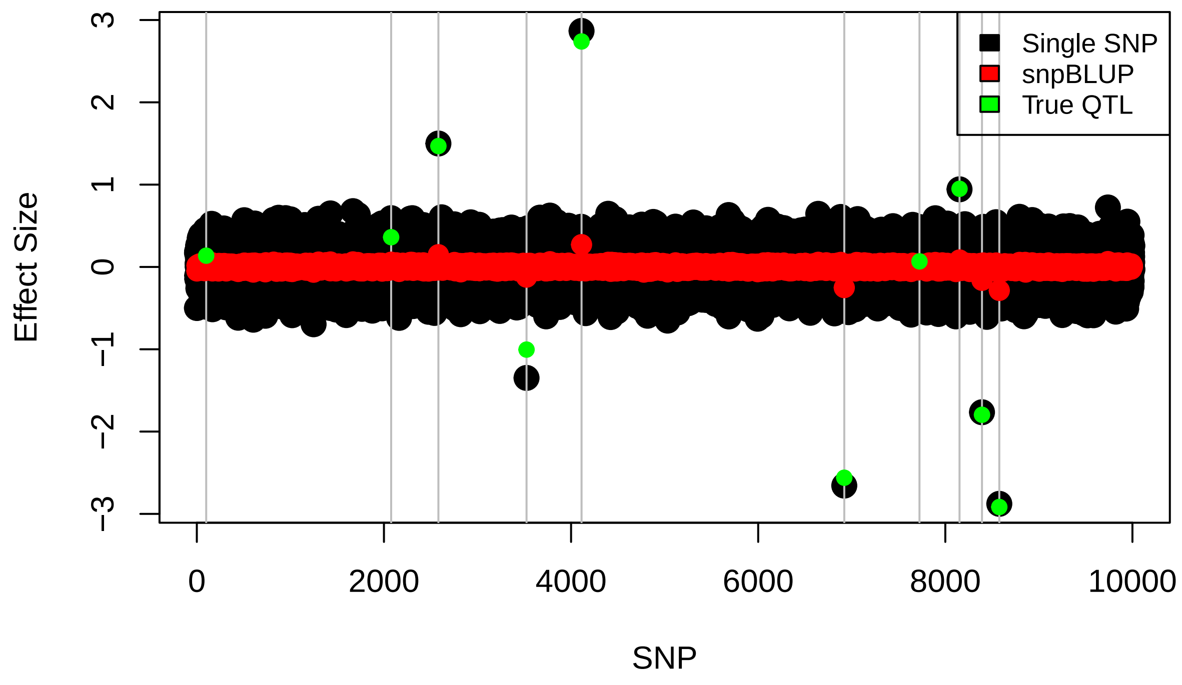
Now, let's compare the effects from the single marker regressions with snpBLUP:

```
# Make variable names more meaningful
effSingle = effect
effBlup = effectBLUP2

# Plot
plot(
  effSingle,
  effBlup,
  xlab = "Single SNP regressions",
  ylab = "snpBLUP",
  main = "SNP effect estimates",
  pch = 20
)
```



And now, let's compare the results from both methods with the truth:

```
# Import data of the ten true QTL effects
QTL = read.table("Data/trueQTL.txt", header = T, sep = "\t")

# Plot
plot(
  effSingle,
  pch = 20,
  cex = 2.5,
  xlab = "SNP",
  ylab = "Effect Size",
  main = ""
)
abline(v = QTL$indexQTL, col = "gray")
points(effBlup,
       col = "red",
       pch = 20,
       cex = 2)
points(QTL$indexQTL,
       QTL$QTLval,
       col = "green",
       pch = 20,
       cex = 1.5)
legend(
  "topright",
  c("Single SNP", "snpBLUP", "True QTL"),
  fil = c("black", "red", "green"),
  cex = 0.8
)
```



First, note the strong shrinkage of effect sizes in snpBLUP. Also note, the single marker regression did a better job at approximating the true effects in the data, but notice how three of the true QTL effects are overestimated and many other non-associated SNP show sizeable effects. snpBLUP, on the other hand, underestimated effects but the non-associated SNP are closer to zero.

Association significance can be tested and compared with the true known QTL:

```
# Single regression analysis
## FDR correction
sig = 0.01 / length(pval)

## Significant SNPs
sigSNP = which(pval < sig)
length(sigSNP)
```

```
## [1] 6
```

```
## Number of true QTL found
length(intersect(sigSNP, QTL$indexQTL))
```

```
## [1] 6
```

```
# snpBLUP
## FDR correction
sig = 0.01 / length(pval)

## Significant SNPs
pvalBlup = 2 * pt(-abs(effBlup / sd(effBlup)), df = length(effBlup) - 1)
sigSNP = which(pval < sig)
length(sigSNP)
```

```
## [1] 6
```

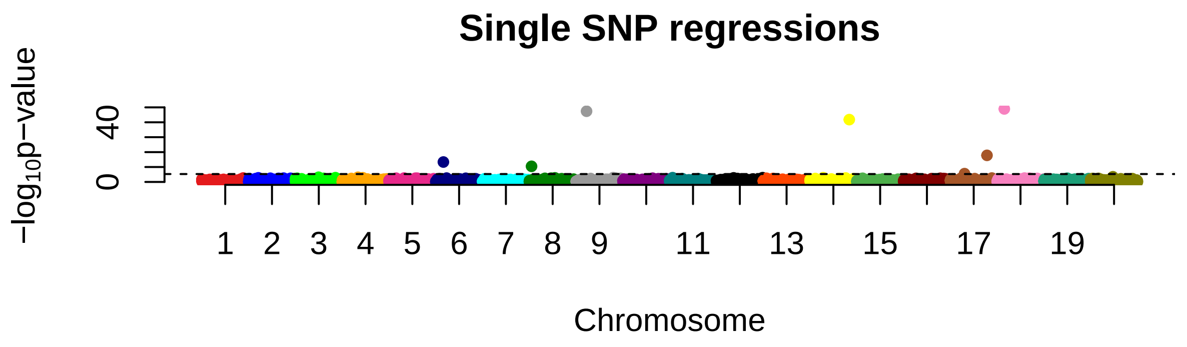
```
## Number of true QTL found
length(intersect(sigSNP, QTL$indexQTL))
```

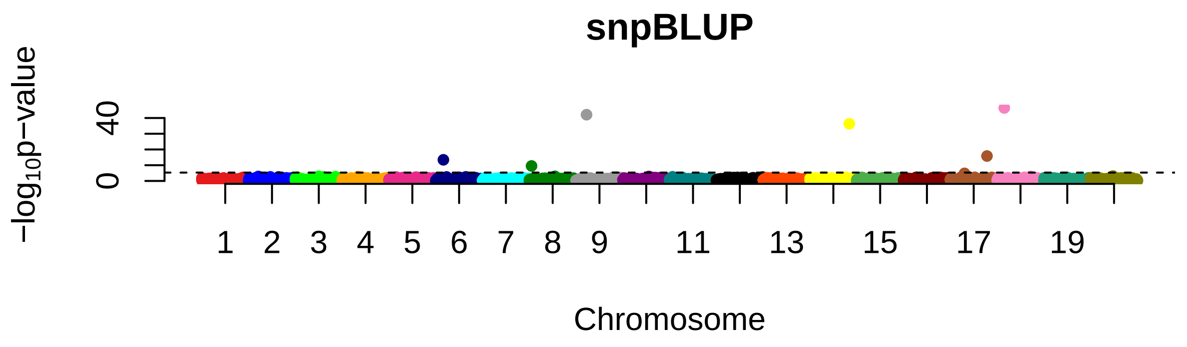
```
## [1] 6
```

Both methods found the same six true QTL and missed four of them. Let's build Manhattan plots for both:

```
library(made4)
mbcol = as.factor(map$chrom)
chroms = unique(map$chrom)
mapcol = getcol(length(chroms))
chromseps = numeric(length(chroms))
xdist = length(map[, 3])
cum = 0
chrpos = numeric(length(chroms))
for (i in 1:length(chroms)) {
  index = which(map[, 2] == chroms[i])
  chromseps[i] = index[length(index)]
  xdist[index] = map[, 3][index] + cum
  cum = cum + map[, 3][index[length(index)]]
  chrpos[i] = cum
}
chrpos[2:length(chroms)] = chrpos[2:length(chroms)] -
  ((chrpos[2:length(chroms)] - chrpos[1:(length(chroms) - 1)]) / 2)
chrpos[1] = chrpos[1] / 2
par(mfrow = c(2, 1))

# Single SNP regressions
plot(
  xdist,
  -log10(pval),
  col = (mapcol[mbcol]),
  pch = 20,
  xlab = "Chromosome",
  ylab = expression(paste("-", log[10], "p-value", sep = "")),
  axes = F,
  main = "Single SNP regressions"
)
abline(h = -log10(0.05 / nrow(map)), lty = 2)
axis(1, at = chrpos, labels = chroms, las = 1)
axis(2, )
```

```
# snpBLUP
plot(
  xdist,
  -log10(pvalBlup),
  col = (mapcol[mbcol]),
  pch = 20,
  xlab = "Chromosome",
  ylab = expression(paste("-", log[10], "p-value", sep = "")),
  axes = F,
  main = "snpBLUP"
)
abline(h = -log10(0.05 / nrow(map)), lty = 2)
axis(1, at = chrpos, labels = chroms, las = 1)
axis(2, )
```





The package *rrBLUP* can be used for snpBLUP as well:

```
# snpBLUP using package
library(rrBLUP)
sol = mixed.solve(y, X)
effPack = sol$u
```

Save all outputs:

```r
effect = data.frame(single = effSingle,
                    snpblup = effBlup,
                    rrblup = effPack)
```

2 Genomic Prediction

The objective of genomic prediction is to make use of genotypic data to predict phenotypic outcomes. This can be achieved using snpBLUP or gBLUP.

- Broad sense heritability (H2): the proportion of total phenotypic variation due to all genetic effects.
- Narrow sense heritability (h2): the proportion of the total phenotypic variation due to the additive genetic effects only.

To dissect a complex trait we first need to know what proportion of the total phenotypic variance is determined by genetics; and then we can try to find the underlying genetic loci that controls this variation.

2.1 Prediction with snpBLUP

First, load the file that contains true additive genetic effects of each individual with environmental variance removed:

```r
# Load file
tgv = read.table("Data/trueGeneticValue.txt",
                 header = T,
                 sep = "\t")$TGV
```

Now, scale genotypes based on allele frequencies:

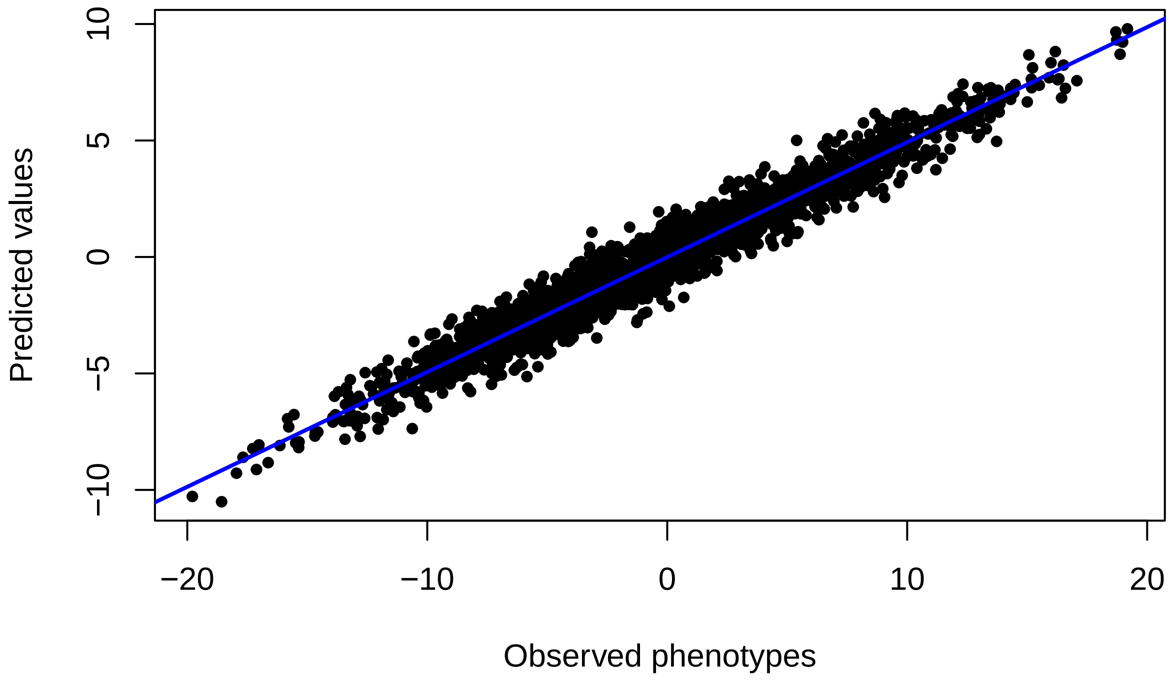
```r
p = rowMeans(gwas) / 2
X = t(gwas - (p * 2))
```

Phenotypes of individuals can be predicted with:

```r
# Using effects from snpBLUP
pred = X %*% effect$snpblup
```

To evaluate how well this worked, the predicted values can be compared directly with the observed phenotypes:

```r
# Plot predicted vs. observed
plot(pheno,
     pred,
     ylab = "Predicted values",
     xlab = "Observed phenotypes",
     pch = 20)
mod = lm(pred ~ pheno)
abline(mod, lwd = 2, col = "blue")
```

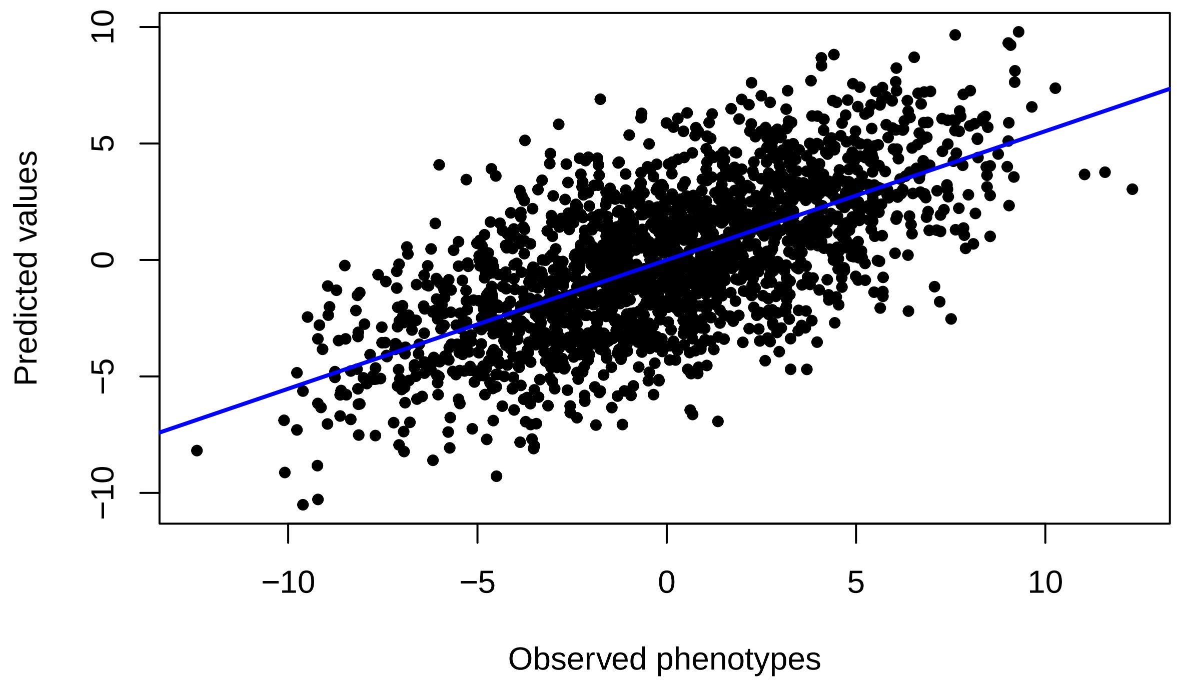


```
# Evaluate the correlation
cor(pheno, pred)
```

```
##           [,1]
## [1,] 0.9750789
```

The correlation is very good and the plot confirms it. Unfortunately, SNP effects were estimated using these same genotypes and phenotypes, which is the reason for such good correlation. Using the true genetic values, we see that our predictions are not so perfect anymore:

```
# Plot predicted vs. observed
plot(tgv,
     pred,
     ylab = "Predicted values",
     xlab = "Observed phenotypes",
     pch = 20)
mod = lm(pred ~ tgv)
abline(mod, lwd = 2, col = "blue")
```

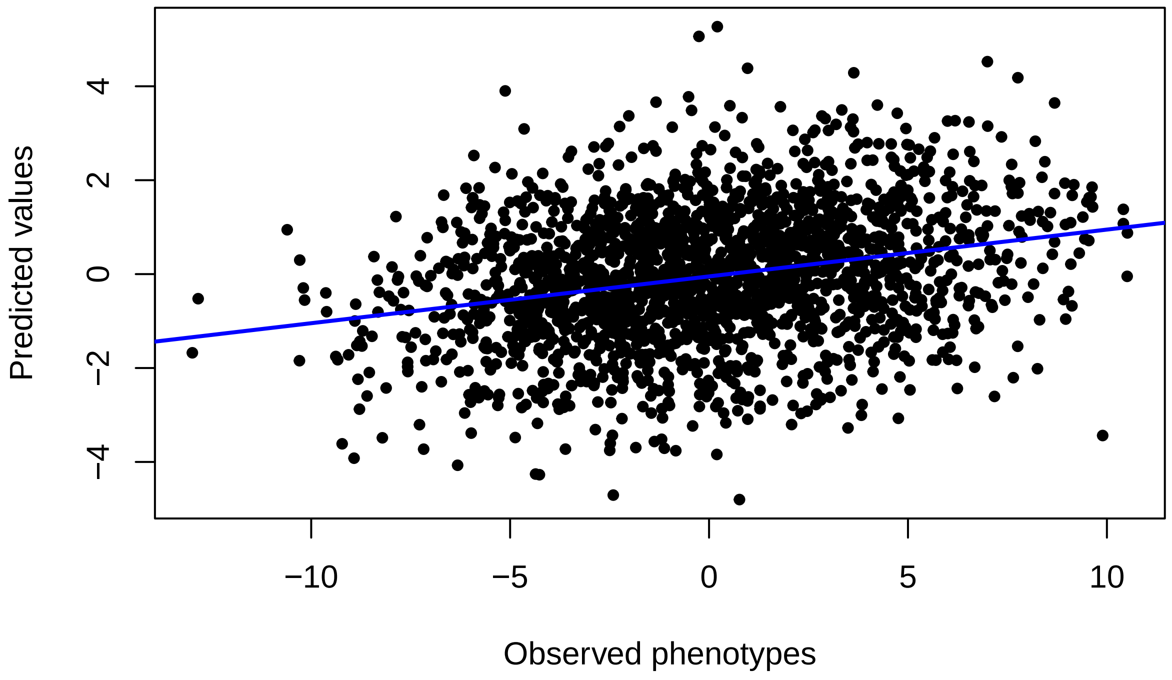


```
# Evaluate the correlation
cor(tgv, pred)
```

```
##           [,1]
## [1,] 0.6615204
```

In practice, effects should be estimated on one dataset (discovery) and evaluated on an independent one (validation). Let's repeat the prediction steps with this new validation dataset:

```
# Import validation data
validG = readRDS("Data/validGeno.rds")
validP = read.table("Data/validPheno.txt", header = T, sep = "\t")$Pheno
validT = read.table("Data/validTGV.txt", header = T, sep = "\t")$TGV

# Scale genotypes based on allele frequencies
validX = t(validG - (p * 2))

# Predict phenotypes
validPred = validX %*% effect$snpblup

# Plot predicted vs. observed
plot(validT,
     validPred,
     ylab = "Predicted values",
     xlab = "Observed phenotypes",
     pch = 20)
mod = lm(validPred ~ validT)
abline(mod, lwd = 2, col = "blue")
```

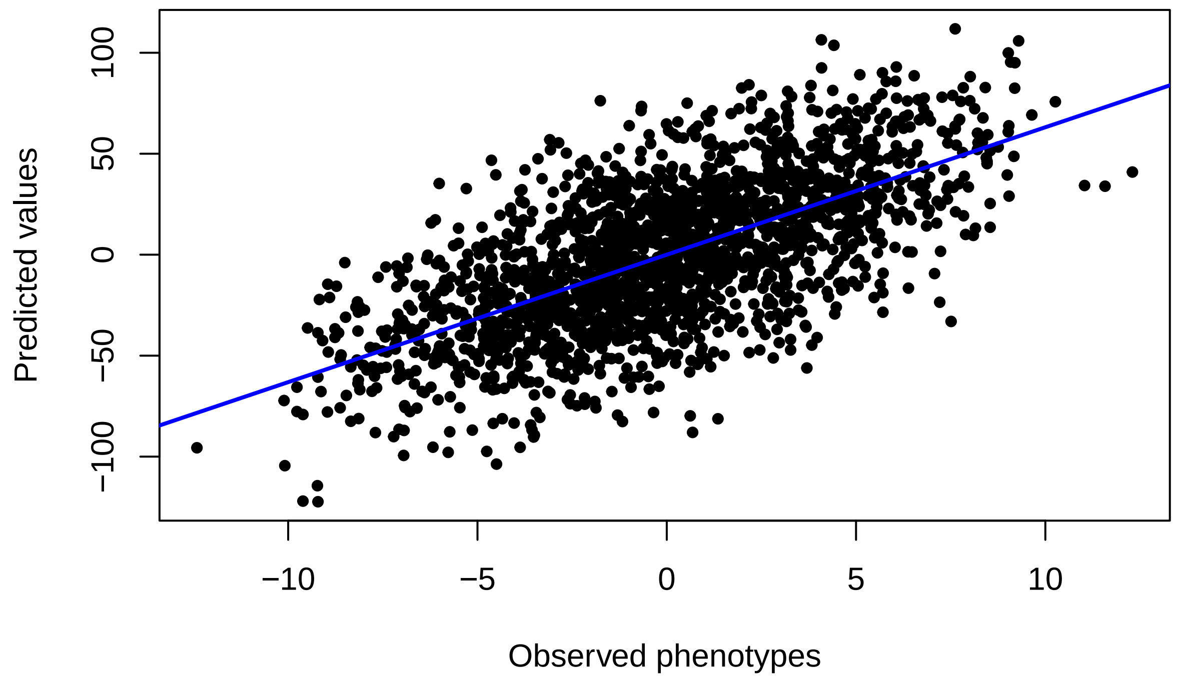


```
# Evaluate the correlation
cor(validT, validPred)
```

```
##           [,1]
## [1,] 0.2719149
```

Now, we use the SNP effects from the single SNP regressions to make predictions. Note how the range of predictions is almost 12x larger than with using the snpBLUP effects:

```
# Using effects from snpBLUP
predS = X %*% effect$single

# Plot predicted vs. observed
plot(tgv,
     predS,
     ylab = "Predicted values",
     xlab = "Observed phenotypes",
     pch = 20)
mod = lm(predS ~ tgv)
abline(mod, lwd = 2, col = "blue")
```



```
# Evaluate the correlation
cor(pheno, predS)
```

```
##           [,1]
## [1,] 0.9213455
```

2.2 Prediction with gBLUP

Instead of using pedigree information to define the relationships between individuals (a numerator relationship matrix) it uses information from a large number of markers. The marker data is used to define genomic relationships in a genomic relationship matrix (GRM). The GRM is a measure of the additive relatedness between individuals and, in simple terms, can be thought of as a correlation matrix between individuals corrected for variation in allele frequencies:

```
# Calculate GRM
## Calculate the allelic frequency
freqAvg = rowMeans(gwas, na.rm = T)
p = freqAvg / 2
## Scale genotypes to -1, 0, and 1
M = gwas - 1
## Subtract M from twice the deviation from 0.5 of the allele frequencies
P = 2 * (p - 0.5)
W = M - P
## Calculate G
```

```r
WtW = t(W) %*% W
d = 2 * sum(p * (1 - p))
G = WtW / d
rm(P, WtW, p, freqAvg)
```

Once the GRM has been calculated, phenotypes can be predicted with gBLUP:

```r
# Heritability
h2 = 0.5

# Lambda
ve = var(y) * (1 - h2)
va = var(y) * h2
lambda = (1 - h2) / (h2 / d)

# gBLUP
Z = matrix(0, ncol(gwas), ncol(gwas))
diag(Z) = 1
ZtZ = t(Z) %*% Z
Gil = solve(G) * lambda
lhs = solve(ZtZ + Gil)
rhs = t(Z) %*% y
sol = lhs %*% rhs
```

Predicted values for individuals in the validation dataset can be obtained by rebuilding the GRM with all individuals and then using the relationships between individuals with and without phenotypes to estimate those without:

```r
All = cbind(gwas, validG)
freqAvg = rowMeans(gwas, na.rm = T)
p = freqAvg / 2
M = All - 1
P = 2 * (p - 0.5)
W = M - P
WtW = t(W) %*% W
d = 2 * sum(p * (1 - p))
Gall = WtW / d
rm(P, W, WtW, d, p, freqAvg)

# Index of individuals in the validation dataset
missindex = 2001:4000
Ginv = Gall[-missindex, -missindex]

# Predict
diag(Ginv) = diag(Ginv) + lambda
Ginv = solve(Ginv)
solAll = Gall[, -missindex] %*% Ginv %*% pheno

# Evaluate the correlation with their phenotype
cor(validPred, solAll[2001:4000])
```

```
##           [,1]
## [1,] 0.9788139
```