Crop 545 Lab 7

Lec 9 Phenotype Simulation

Y.Song

2/12/2018

This document is an annotated demonstration of the R codes provided from **Lecture 9** (genetic architechture & phenotype simulation) to **Lecture 12** (PCA). For the time we have in lab, we will not be able to cover all codes in detail. Please practice the script, discuss with your peers, and ask questions.

We cannot fully understand and make progress in coding without actually typing our own ideas!

'Please notice warnings and messages in this document has been muted. When you run the script in R console, you might get warning signs that were not shown here.

Lec 9: genetic architechture & phenotype simulation

For Lecture 8 we need to understand and be able to implement the G2P function. The script of the function is already provided through the zzlab.net website. The target for this section is to understand it.

9.1 phenotype distribution

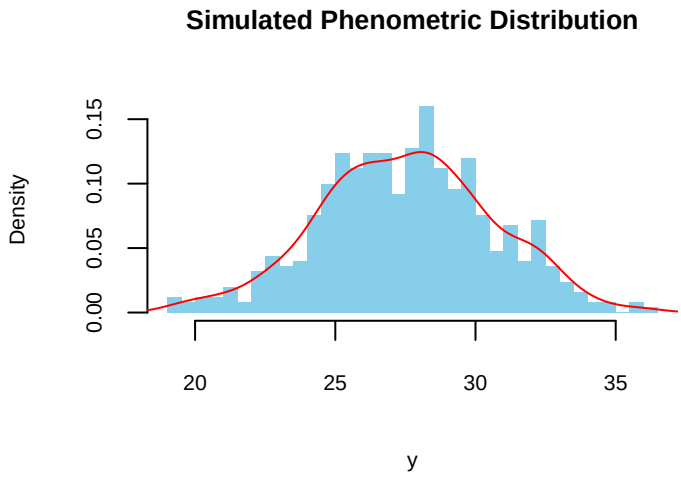
The R chunk below is from slide 9.6, which is a realization of 8.5. Here, we are simulating a phenometric distribution in a trait. A brief but helpful explaination is here https://www.ncbi.nlm.nih.gov/books/NBK22080/.

Imagine we have m number of gene, and n number of individual in a trait.

```
m=100 # number of gene
n=500  # number of individual
x=runif(m) # x = a uniform r.v. with the length of m
gene=matrix(x,n,m,byrow = T)
# gene = a matrix with n row and m col rows are repeats of the x we created above
# The above create the matrix in 8.6 "assign to individuals"

# next we will assign True/False values to each elemant in gene matrix by 50/50
# chance. This is a review of how to flip a coin with R. Please make sure you
# understand.
galton=matrix(runif(n*m),n,m)
# galton is a matrix full of uniform (0,1) r.v., with same dimension as 'gene'
galton.binary=galton<.5
# the above generate the matrix in 8.6 "Uniform random variable"

gene[galton.binary]=0 # at the 'TRUE' location, the orignal 'gene' value
# will be turned into zero
# till here, certain genes in some individuals are 'expressed', in some are 'muted'.
y=rowSums(gene) # sum up the total gene effect in each individual
# The following part contain a lot of plotting setting, please ignore.
# The kernel is hist() for histogram diagram.
par(cex.lab=0.7, cex.axis=0.7, cex.main=0.8)
hist(y, freq=F, breaks=round(n/10), col="skyblue",lty="blank",
       main="Simulated Phenometric Distribution")
lines(density(y), col="red")
```



We can wrap the above idea into a function for convinience, so we can quickly compare the different distribution under various combination of this two number. Please try write a function `phe.sim()` function which take `m` and `n` as input and directly plot the histogram out. (The function is provided in the r script).

9.2 QTN position plot

The corresponding result slide is 9.17.

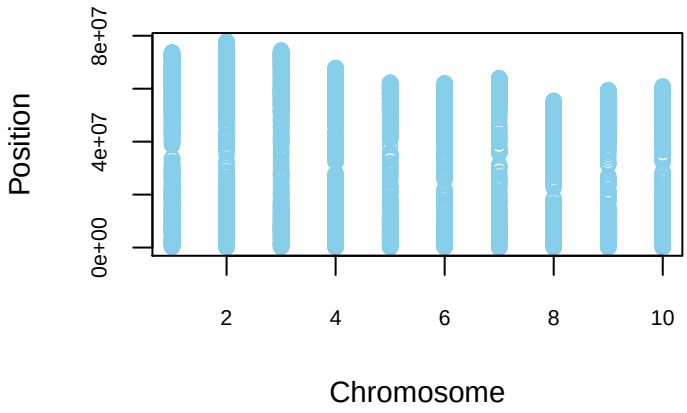
For plotting a QTN plot, the only information we need is 'Chromosome' and 'Position' information. With a 'standard & clean' hapmap file, such information is already provided. For example:

```
# QTN position plot
# the hapmap example data is a subset of the dataset downloaded from the Dryland
# website.
hapmapEX=read.table("hapmapEX.txt", header = T)
hapmapEX[1:6, 1:8]
```

```
##           rs. alleles chrom      pos strand assembly. center protLSID
## 1 S1_68306052     C/T     1 68306052      +         NA     NA        NA
## 2 S2_64594459     C/T     2 64594459      +         NA     NA        NA
## 3 S3_45663174     G/A     3 45663174      +         NA     NA        NA
## 4 S5_54683496     T/G     5 54683496      +         NA     NA        NA
## 5 S2_20427728     A/C     2 20427728      +         NA     NA        NA
## 6  S7_2890041     C/G     7  2890041      +         NA     NA        NA
```

```
# col3 = chrommosome
# col4 = position
# It should be a standard in all Hapmap format

plot(hapmapEX[, c(3,4)],
     # below are plot settings:
     col="skyblue",
     xlab="Chromosome", ylab="Position",
     cex.lab=0.9, cex.axis=0.7)
```



In the lecture notes, we randomly pick some marker locations and circled it in the QTN plot. Please read the lecture slides 8.16 - 8.19 and figure out how it was achieved. (Hint: QTN.position)

9.3 Simulate phenotype

Statistical simulation is a common and useful way in data analysis. The intuitition is such that since we have some prior knowledge about the population - some kind of rule (e.g. the distribution, or some information mean, variance...) of the population, we can construct a realization of data under that rule. Since the simulated data was coming from the same rule of the population, we then believe it can somehow help us understand the unobserved population.

In this section, we use the genotype data that we have, make some 'rules' about the distribution of the phenotype (normally distributed), and construct some phenometic trait simulation.

In this part, we simply say $y_i = g_i + e_i$, where y_i is the phemometric trait, $ g_i$ is the gene effect, and e_i is the residual term. Our target is to get a simulated sample of y_i. We will simulate g_i, calculate e_i and y_i. Recall from the lecture:

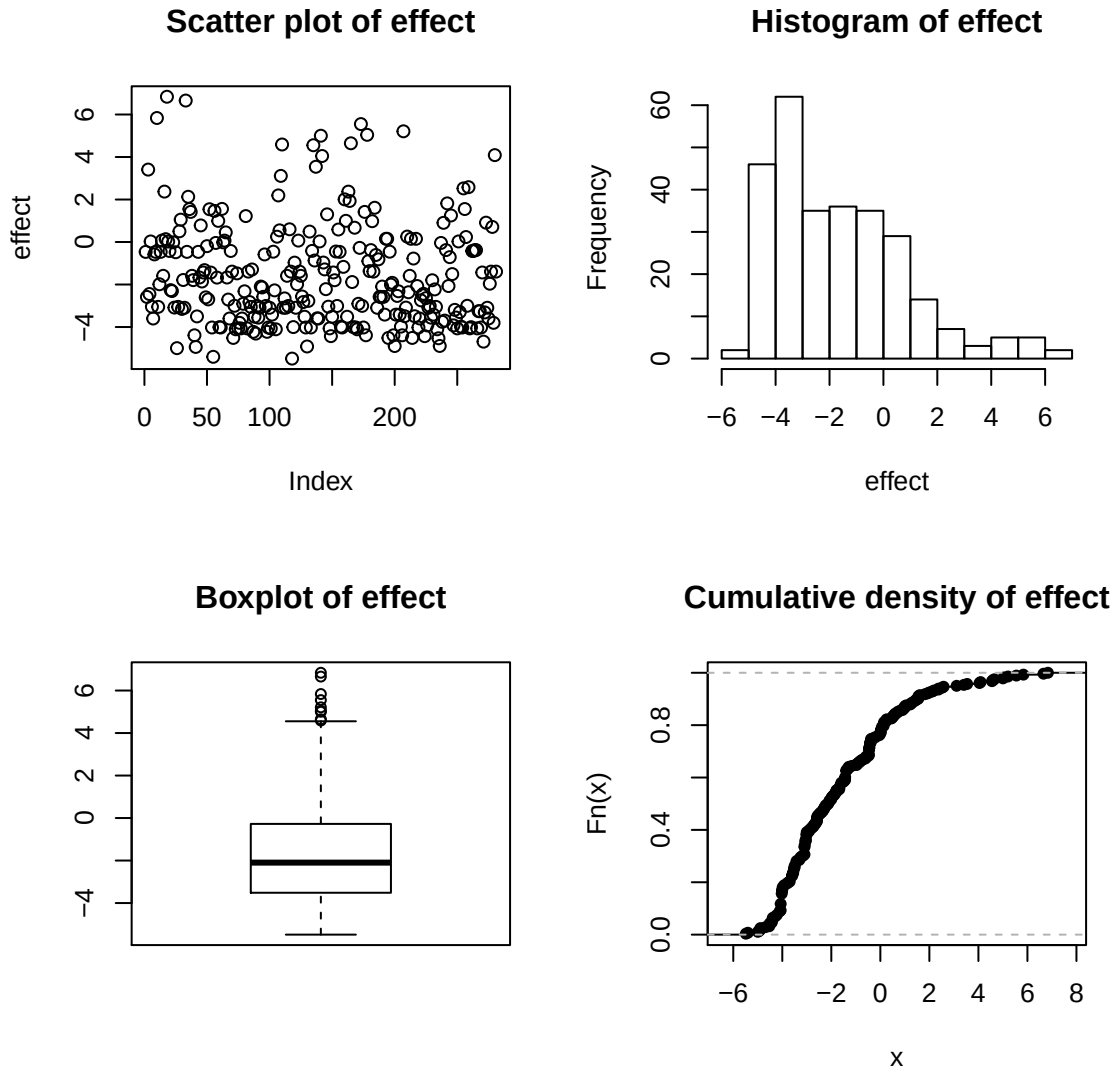
$$h^2 = \frac{var(g)}{var(g) + var(e)}$$

then

$$var(e) = \frac{var(g) - var(g) \times h^2}{h^2}$$

```
# Sampling QTN
myGD=read.table(file="http://zzlab.net/GAPIT/data/mdp_numeric.txt",head=T)
NQTN=10
X=myGD[,-1]
n=nrow(X); m=ncol(X)
QTN.position=sample(m,NQTN,replace=F)
SNPQ=as.matrix(X[,QTN.position])
# randomly simulate some additive effect.
# recall the add effect follows a normal distribution
addeffect=rnorm(NQTN,0,1)
effect=SNPQ%*%addeffect
# %*% stands for matrix multiplication
```

The above gives us an $n \times 1$ column of g_i's:

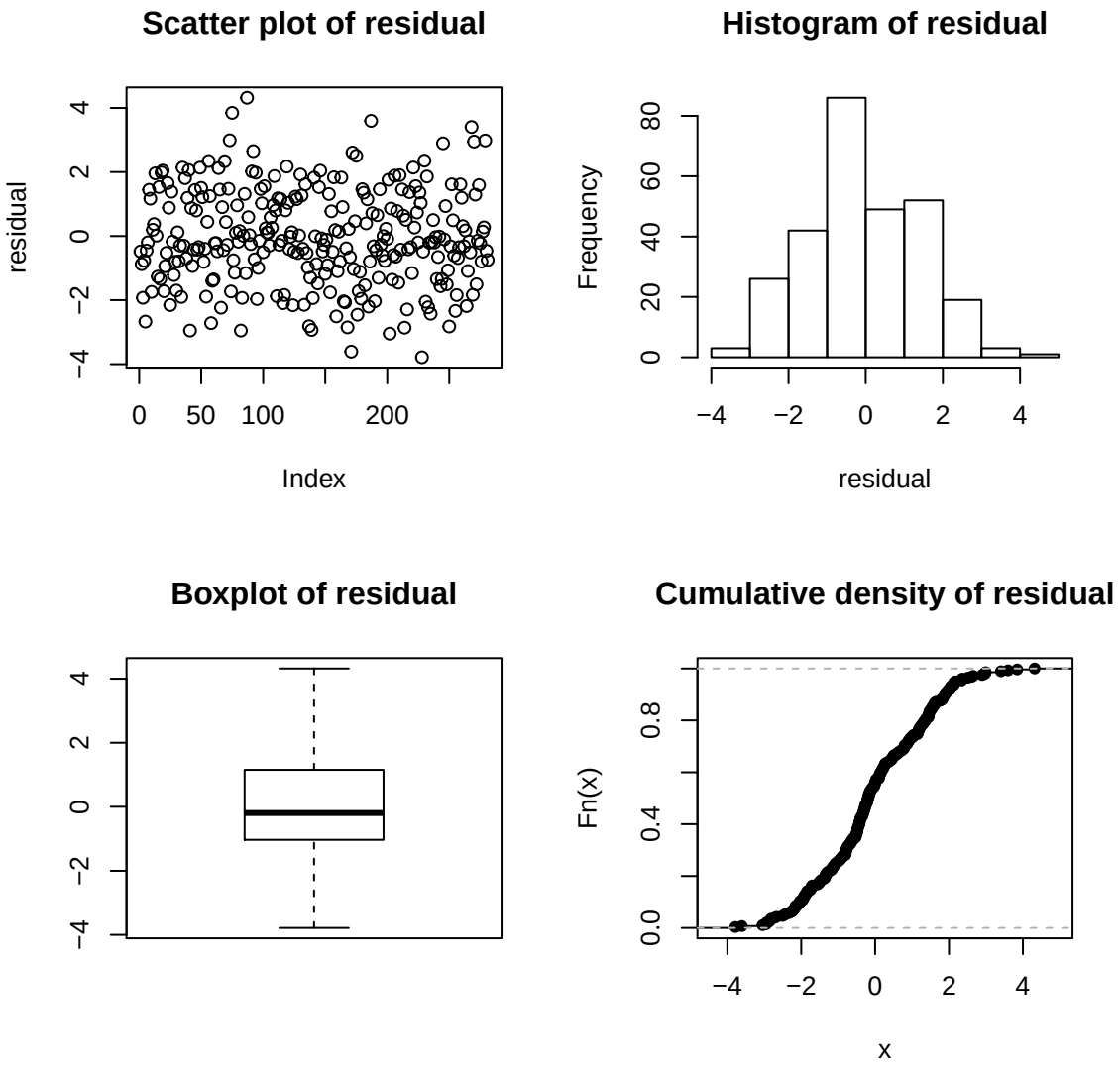


Now we assume an fixed h-sqaure score, and calculate residual variance $var(e)$. Why? Because we assume $e_i \sim N(0, \sigma^2)$, and with the calculated $var(e)$ value, we can say $\hat{\sigma}^2 = var(e)$ and be able to let R to generate some normally distributed r.v.'s for us.

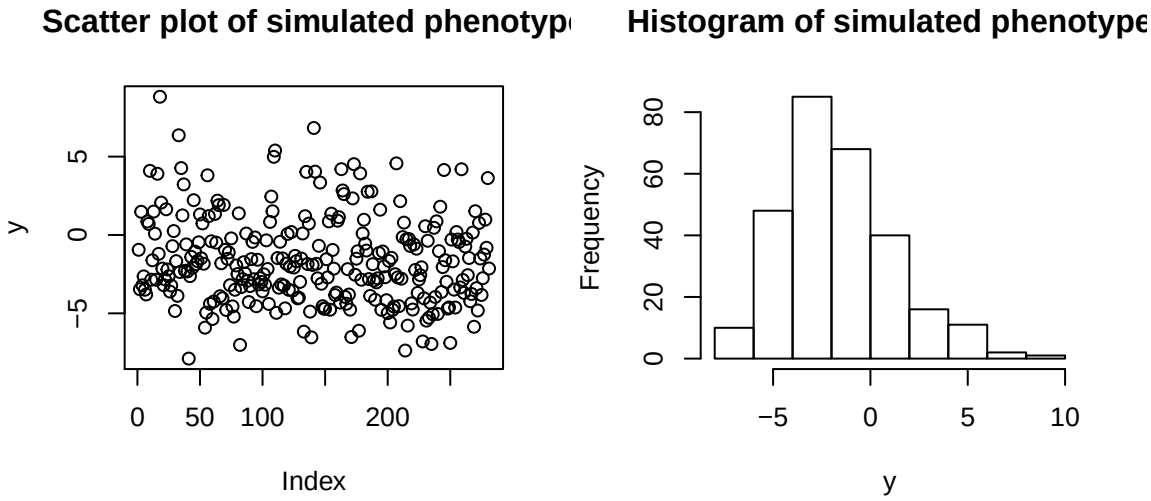
```
h2=.7
effectvar=var(effect)
residualvar=(effectvar-h2*effectvar)/h2 # see this equation in lecture notes
residual=rnorm(n,0,sqrt(residualvar))
y=effect+residual
```

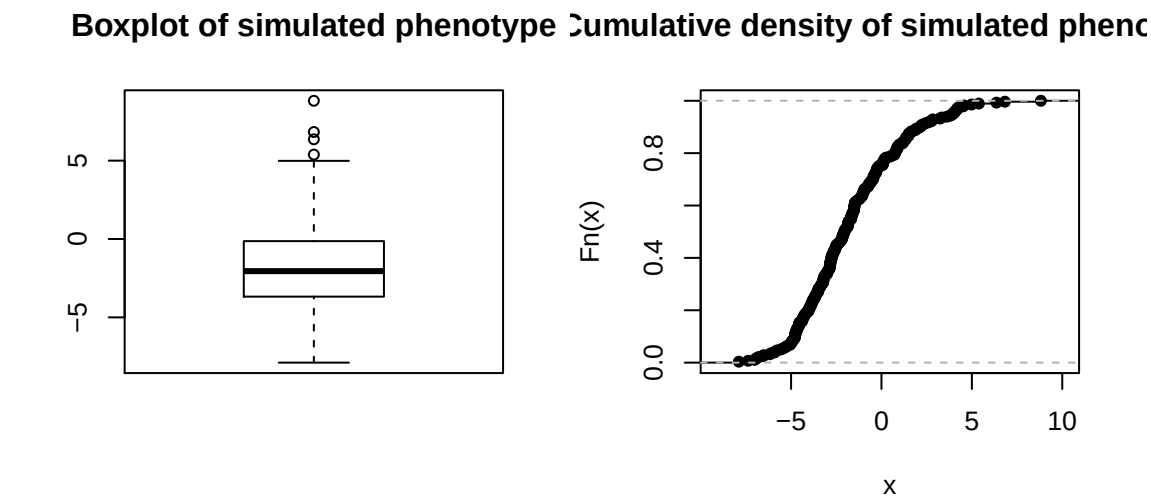
In the above, we obtained the residuals, and then get the phenometric value by plus effect and residual together.

```
par(mfrow=c(2,2))
plot(residual,main="Scatter plot of residual")
hist(residual)
boxplot(residual, main="Boxplot of residual")
plot(ecdf(residual), main="Cumulative density of residual")
```




```
par(mfrow=c(2,2))
plot(y,main="Scatter plot of simulated phenotype")
hist(y,main="Histogram of simulated phenotype")
boxplot(y,main="Boxplot of simulated phenotype")
plot(ecdf(y),main="Cumulative density of simulated phenotype")
```



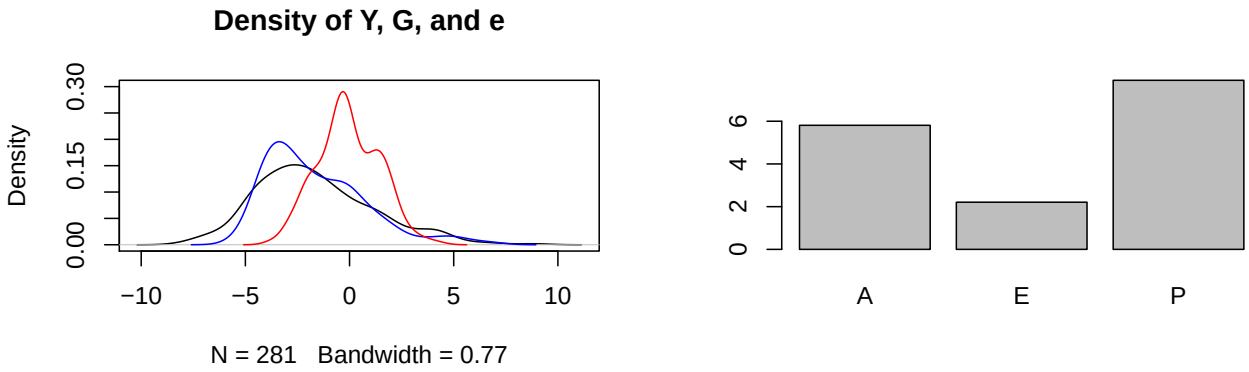


9.4 Heritability

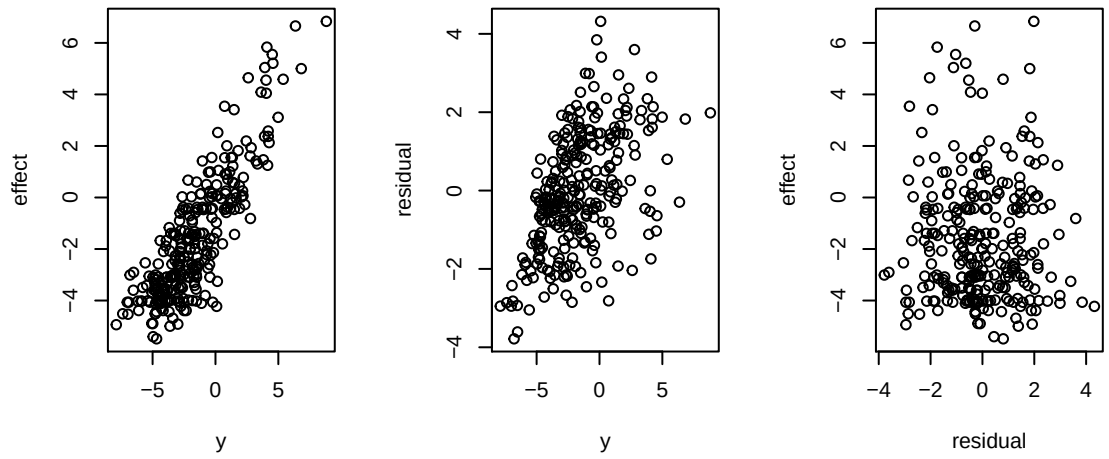
As we mentioned before, we simulated the phenometric trait data so we can use it to help us learn how much of variation in the phenotypic trait was due to genetic variation. After we got the data, we simply plot the variability out:

```
va=var(effect)
ve=var(residual)
vp=var(y)
v=matrix(c(va,ve,vp),1,3)
colnames(v)=c("A", "E","P") # assign names to columns

par(mfrow=c(1,2))
plot(density(y),ylim=c(0,.3), main="Density of Y, G, and e")
lines(density(effect),col="blue")
lines(density(residual),col="red")
barplot(v,col="gray")
```




```
# Correlations between y and effects
# Plot
par(mfrow=c(1,3))
plot(y,effect)
plot(y,residual)
plot(residual,effect)
```




```
cor(y,effect)
```

```
##           [,1]
## [1,] 0.8491922
```

9.5 G2P Function

In real life, we need to do the previous steps more than once. We can wrap all the idea into one function. This is what the G2P function is useful for. It's a long function, but you have already seen most of it in the above section. Please read and try to understand it.

```
G2P=function(X,h2,alpha,NQTN,distribution){
  n=nrow(X)
  m=ncol(X)
  #Sampling QTN
  QTN.position=sample(m,NQTN,replace=F)
```

```
  SNPQ=as.matrix(X[,QTN.position])
  QTN.position
  #QTN effects
  if(distribution=="norm")
  {addeffect=rnorm(NQTN,0,1)
  }else
  {addeffect=alpha^(1:NQTN)}
  #Simulate phenotype
  effect=SNPQ%*%addeffect
  effectvar=var(effect)
  residualvar=(effectvar-h2*effectvar)/h2
  residual=rnorm(n,0,sqrt(residualvar))
  y=effect+residual
  return(list(addeffect = addeffect,
              y=y, add = effect,
              residual = residual,
              QTN.position=QTN.position,
              SNPQ=SNPQ))
}
```