Chapter 4: Generation based genetics analysis

Jixiang Wu

Associate Professor of Quantitative Genetics/Biostatistics

AHPS Department

South Dakota State University, SD 57007

Email: jixiang.wu@sdstate.edu

Phpone: 605-688-5947

Introduction

In general analysis of quantitative traits can not follow the standard procedures used to analyze major gene traits, such as for a one-gene (3:1) or two-gene (9:3:3:1) phenotypic ratio in an F2 becuase it is impossible to follow the segregation of the separate underlying polygenes. Instead, it is necessary to look at the degree of similarity or difference among related individuals and families using various statistics such as means, variances, covariances, regressions and correlations, parts of which have been mentioned so far. In this chapter, we will learn how to use generation means to estimate genetic effects and generation variances to estimate genetic variance components and heritability with R. We will do such data analyses with generation means and variances onlly and individual data as well. In addition, we will integrate regression methods and resampling techniques into our data analysis.

Analysis based on generation means and variances

Six-generation means and variances

The data used for our demonstration in this chapter include six-generation means and variances (P1, P2, F1, F2, BC11, BC12), which are calculated from 100 observations, separately. The data are provided as follows.

```
gm=c(69.44, 59.04, 83.44, 74.36, 76.03, 71.28)
names(gm)=c("P1","P2","F1","F2","BC11","BC12")
gv=c(59.73,65.71,51.81,100.75,81.05,90.83)
df=rep(99,6)
gv=gv/(df+1)
dat=data.frame(gm,gv,df)
dat
```

	gm	gv	df
P1	69.4	0.597	99
P2	59.0	0.657	99
F1	83.4	0.518	99
F2	74.4	1.008	99
BC11	76.0	0.810	99
BC12	71.3	0.908	99

Estimating genetic effects

Estimate population mean:mu

We can use P1 and P2 means to estimate the population mean.

```
mu.est=(gm[1]+gm[2])/2
```

The variance for mu estimate can be calculated as follows.

```
mu.var=(gv[1]+gv[2])/4
```

Thus we can use t-test to test the significance of mu from a constant such as zero or any given value with degrees of freedom 198 (100+100-2).Calculation of the variance for the estimated mu can refer to Chapter 2 in this course.

```
tvalue=mu.est/sqrt(mu.var)
pvalue=1-pt(abs(tvalue),(df[1]+df[2]))
data.frame(mu=mu.est,var=mu.var,df=(df[1]+df[2]),tvalue,pvalue)
```

	mu	var	df	tvalue	pvalue
P1	64.2	0.314	198	115	0

```
res.mu=c(mu.est,mu.var,(df[1]+df[2]),tvalue,pvalue)
```

Estimate additive effects: A

We can use P1 and P2 means to estimate the population mean.

```
A.est=(gm[1]-gm[2])/2
```

The variance for mu estimate can be calculated as follows.

```
A.var=(gv[1]+gv[2])/4
```

Thus we can use t-test to test the significance of additive effects from a constant such as zero or any given value with degrees of freedom 198 (100+100-2). In the same manner, we can calculate the variance for the estimate of A.

```
tvalue=A.est/sqrt(A.var)
pvalue=1-pt(abs(tvalue),(df[1]+df[2]))
data.frame(A=A.est,var=A.var,df=(df[1]+df[2]),tvalue,pvalue)
```

	A	var	df	tvalue	pvalue
P1	5.2	0.314	198	9.29	0

```
res.A=c(A.est,A.var,(df[1]+df[2]),tvalue,pvalue)
```

Estimate dominance effect: D

We can use P1 and P2 means to estimate the population mean.

```
D.est=gm[3]-(gm[1]+gm[2])/2
```

The variance for mu estimate can be calculated as follows.

```
D.var=gv[3]+(gv[1]+gv[2])/4
```

Thus we can use t-test to test the significance of dominance effect from a constant such as zero or any given value with degrees of freedom 297 (100+100+100-3).Like A, estimated variance can be obtained through the properties of variance.

```
tvalue=D.est/sqrt(D.var)
pvalue=1-pt(abs(tvalue),(df[1]+df[2]+df[3]))
data.frame(D=D.est,var=D.var,df=(df[1]+df[2]+df[3]),tvalue,pvalue)
```

	D	var	df	tvalue	pvalue
F1	19.2	0.832	297	21.1	0

```
res.D=c(D.est,D.var,(df[1]+df[2]+df[3]),tvalue,pvalue)
```

We can summarize three estimates for population mean, additive effect, and dominance effect

```
res=rbind(res.mu,res.A,res.D)
rownames(res)=c("mu","A","D")
colnames(res)=c("Est","Var","DF","t-value","p-value")
data.frame(res)
```

	Est	Var	DF	t.value	p.value
mu	64.2	0.314	198	114.71	0
A	5.2	0.314	198	9.29	0
D	19.2	0.832	297	21.05	0

Estimated genetic effects and their estimated variances could be slightly different if different normal equations are used to estimate these effects.

Obviously, the above used method may not use all information (equations) to estimate genetic parameters. In fact, we can use more equations/geneartions to achieve this

goal.This approach is linear regression based method, which can directly use the built-in function lm. Now we will use six generation means to estimate genetic effects as demonstrated in the following R. However, it is important to generate the coefficients for each of six generations.

```
X=matrix(0,6,2)
name=names(gm)
id=which(name=="P1")
if(length(id)>0)X[id,]=c(1,0)
id=which(name=="P2")
if(length(id)>0)X[id,]=c(-1,0)
id=which(name=="F1")
if(length(id)>0)X[id,]=c(0,1)
id=which(name=="F2")
if(length(id)>0)X[id,]=c(0,0.5)
id=which(name=="BC11")
if(length(id)>0)X[id,]=c(0.5,0.5)
id=which(name=="BC12")
if(length(id)>0)X[id,]=c(-0.5,0.5)
colnames(X)=c("A","D")
dat=data.frame(dat,X)
dat
```

	gm	gv	df	A	D
P1	69.4	0.597	99	1.0	0.0
P2	59.0	0.657	99	-1.0	0.0
F1	83.4	0.518	99	0.0	1.0
F2	74.4	1.008	99	0.0	0.5
BC11	76.0	0.810	99	0.5	0.5
BC12	71.3	0.908	99	-0.5	0.5

```
reg=lm(gm~A+D,data=dat)
summary(reg)
```

```
##
## Call:
## lm(formula = gm ~ A + D, data = dat)
##
## Residuals:
##      P1      P2      F1      F2    BC11    BC12
##  0.0724 -0.1076 -0.0353  0.4935 -0.3915 -0.0315
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   64.258      0.239   268.9  1.1e-07 ***
## A              5.110      0.235    21.7  0.00021 ***
## D             19.218      0.442    43.4  2.7e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.372 on 3 degrees of freedom
## Multiple R-squared:  0.999,  Adjusted R-squared:  0.998
## F-statistic: 1.18e+03 on 2 and 3 DF,  p-value: 4.53e-05

b=reg$coef
names(b)=c("mu","A","D")
b

##    mu     A     D
## 64.26  5.11 19.22
```

Though we can obtain the estimates for three parameters: mu, A, and D with a multiple linear regression analysis, the statistical tests here could be a little different from a conventional t-test.

On the other hand, since variances for these six generations are not the same if variances for additive and/or dominance effects are significant, a weighted linear regression analysis may be applied as well. The analysis is demonstrated as follows.

```
###Weighted linear regression
#Var=c(59.73,65.71,51.81,100.75,81.05,90.83)
reg.wls<- lm(gm~A+D, weights=1/gv, data=dat)
summary(reg.wls)

##
## Call:
## lm(formula = gm ~ A + D, data = dat, weights = 1/gv)
##
## Weighted Residuals:
##       P1       P2       F1       F2     BC11     BC12
##  0.08385 -0.10073 -0.01136  0.50979 -0.42396 -0.00516
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   64.248      0.201   319.6  6.8e-08 ***
## A              5.127      0.201    25.5  0.00013 ***
## D             19.200      0.351    54.6  1.4e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.39 on 3 degrees of freedom
## Multiple R-squared:  0.999,  Adjusted R-squared:  0.999
## F-statistic: 1.78e+03 on 2 and 3 DF,  p-value: 2.44e-05

b=reg.wls$coef
names(b)=c("mu","A","D")
b
```

```
##    mu     A     D
## 64.25  5.13 19.20
```

Results from a regular and weighted linear regression analyses, however, are very comparable.

Estimating genetic variance components with six generation variances

As used to estimate genetic parameters like population mean, additive efffect, and dominance effect, genetic variance components can be mannually estimated easily. However, we may use a linear regression analysis method to estimate these genetic components as demonstrated as follows. Again, first we need to generate nummerical coefficient for each variance components.

```
#Var=c(59.73,65.71,51.81,100.75,81.05,90.83)
#name=names(Var)=c("P1","P2","F1","F2","BC11","BC12")

X=matrix(0,6,3)
id=which(name=="P1")
if(length(id)>0)X[id,]=c(0,0,0)
id=which(name=="P2")
if(length(id)>0)X[id,]=c(0,0,0)
id=which(name=="F1")
if(length(id)>0)X[id,]=c(0,0,0)
id=which(name=="F2")
if(length(id)>0)X[id,]=c(1,1,0)
id=which(name=="BC11")
if(length(id)>0)X[id,]=c(0.5,1,-1)
id=which(name=="BC12")
if(length(id)>0)X[id,]=c(0.5,1,1)
colnames(X)=c("VA","VD","VAD")
dat=data.frame(dat,X)
dat
```

	gm	gv	df	A	D	VA	VD	VAD
P1	69.4	0.597	99	1.0	0.0	0.0	0	0
P2	59.0	0.657	99	-1.0	0.0	0.0	0	0
F1	83.4	0.518	99	0.0	1.0	0.0	0	0
F2	74.4	1.008	99	0.0	0.5	1.0	1	0
BC11	76.0	0.810	99	0.5	0.5	0.5	1	-1
BC12	71.3	0.908	99	-0.5	0.5	0.5	1	1

```
reg=lm(gv~VA+VD,data=dat)
summary(reg)

##
## Call:
## lm(formula = gv ~ VA + VD, data = dat)
```

```
##
## Residuals:
##        P1        P2        F1        F2      BC11      BC12
##  6.47e-03  6.63e-02 -7.27e-02 -1.73e-18 -4.89e-02  4.89e-02
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.5908     0.0401   14.72  0.00068 ***
## VA            0.2962     0.1703    1.74  0.18042
## VD            0.1205     0.1270    0.95  0.41268
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0695 on 3 degrees of freedom
## Multiple R-squared:  0.92,   Adjusted R-squared:  0.866
## F-statistic: 17.2 on 2 and 3 DF,  p-value: 0.0227
```

```
v=reg$coef
names(v)=c("Ve","VA","VD")

(VP=sum(v))
```

```
## [1] 1.01
```

```
(VG=v[2]+v[3])
```

```
##    VA
## 0.417
```

```
(Hn_Square=v[2]/VP)   ## Narrow-sense heritability
```

```
##    VA
## 0.294
```

```
(Hb_Square=VG/VP)   ## Broad-sense heritability
```

```
##    VA
## 0.414
```

Generation mean analysis based on individual data

When individual data are used, we only need to do an extra step of analysis: obtaining generation means and variances.The following data were saved in P1, P2, F1, and F2. Please follow the following R codes for demenstration.

```
library(DAAG)
```

```
## Warning: package 'DAAG' was built under R version 3.4.4
```

```
## Loading required package: lattice
```

```
## Input the actual data for four generation #####
P1=c(72.8,  70.3,  73.4,    71.8,   70.8,   73.1,   72.3,   71.3,   72.2,
71.3)
P2=c(62.8,  60.3,   63.8,   64.3,   63, 61.4,   64.8,   62.1,   64.3,   63.9)
F1=c(70.3,  69.3,   68.3,   70.1,   72.1,   70.8,   71.3,   69.8,   70.4,
71.2)
F2=c(63.4,  72.8,   82.1,   60.8,   69.3,   84.1,   80.2,   72.8,   70.6,
80.9,   69.8,   71.3,   70.4,   80.3,   67.4,   70.4,   79.4,   60.3,   82.1,
73.4,62.3,85.3,72.1,79.5,80.3,68.3,72.1,80.3,79.8,71.3)
```

First we will estimate the means for these four generations.

```
m=c(mean(P1),mean(P2),mean(F1),mean(F2))
v=c(var(P1),var(P2),var(F1),var(F2))
s=c(length(P1),length(P2),length(F1),length(F2))
names(m)=c("P1","P2","F1","F2")
names(v)=c("P1","P2","F1","F2")
data.frame(m,v,s)
```

	m	v	s
P1	71.9	1.03	10
P2	63.1	2.08	10
F1	70.4	1.17	10
F2	73.8	49.90	30

```
X=matrix(c(1,0,-1,0,0,1,0,0.5),nrow=4,byrow=T) # coeffiecient for additive
and dominance effects
colnames(X)=c("A","D")
data.frame(X)
```

A	D
1	0.0
-1	0.0
0	1.0
0	0.5

```
dat=data.frame(m,X)
reg=lm(m~.,data=dat)
summary(reg)

##
## Call:
## lm(formula = m ~ ., data = dat)
##
## Residuals:
##    P1    P2    F1    F2
## -0.88 -0.88 -1.76  3.52
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    68.38       2.78   24.57    0.026 *
## A               4.43       2.92    1.52    0.371
## D               3.74       4.98    0.75    0.590
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.13 on 1 degrees of freedom
## Multiple R-squared:  0.741,  Adjusted R-squared:  0.224
## F-statistic: 1.43 on 2 and 1 DF,  p-value: 0.508
```

```
b=reg$coef
b0=b
df=vb=numeric(length(b))
vb[1]=vb[2]=(v[1]/s[1]+v[2]/s[2])/4
vb[3]=v[3]/s[3]+vb[1]

df[1]=df[2]=s[1]+s[2]-2
df[3]=s[3]+df[1]-1
t=b/sqrt(vb)
#p=numeric(3)
#i=1
#for(i in 1:3)p[i]=1-pt(abs(t[i]),df[i])
pvalue<-(1-pt(abs(t),df))
res=data.frame(b,sqrt(vb),t,df,pvalue)
colnames(res)=c("Est.","SE","t_value","DF","P_value")
res
```

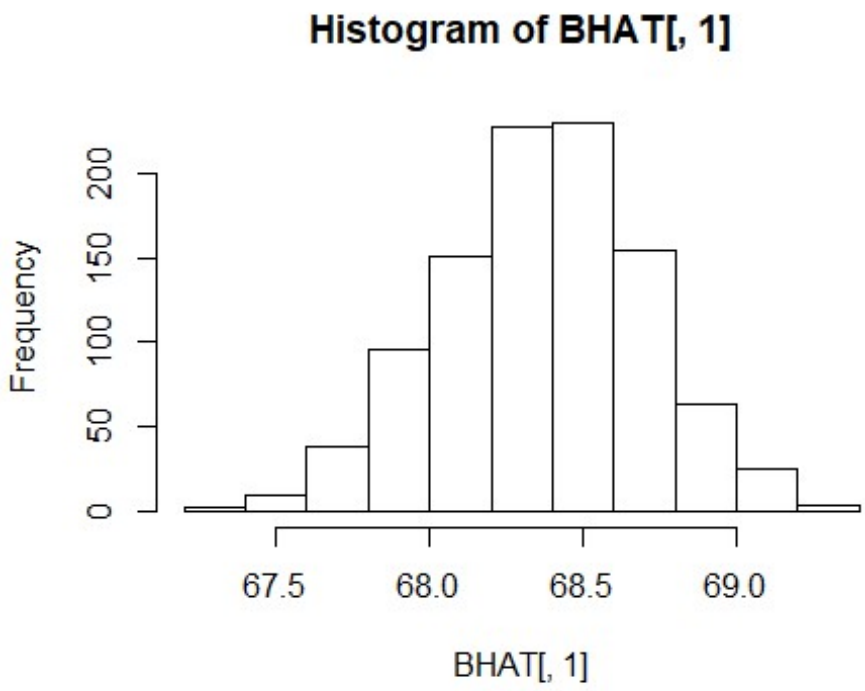
	Est.	SE	t_value	DF	P_value
(Intercept)	68.38	0.279	245.36	18	0
A	4.43	0.279	15.89	18	0
D	3.74	0.442	8.47	27	0

Bootstrapping methods for genetic effect estimation

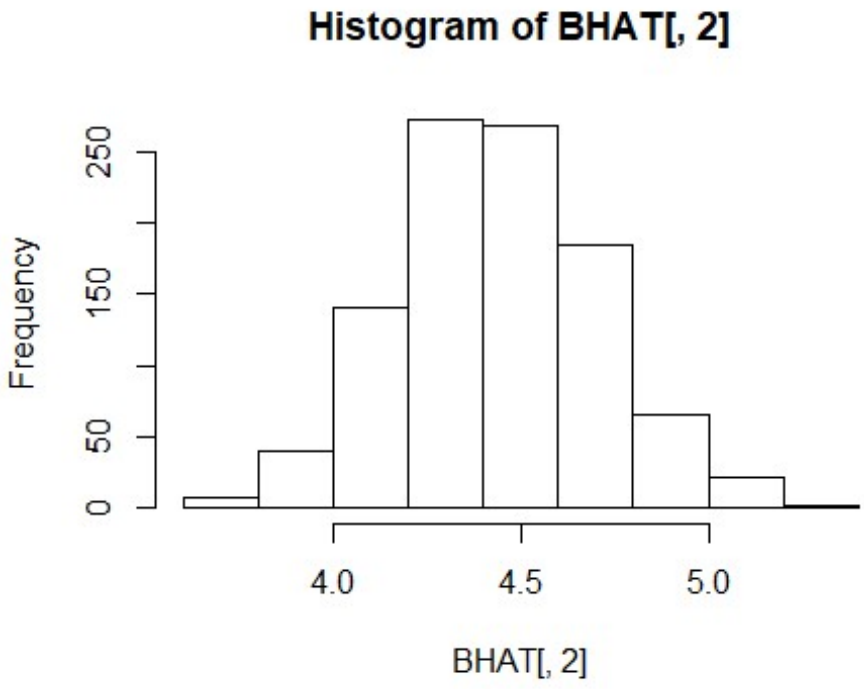
Bootstrapping samples within each population

```
B=1000
X=cbind(1,X)
BHAT=matrix(0,nrow=B,ncol=3)
for(i in 1:B){
  p1=sample(P1,replace=T)
  p2=sample(P2,replace=T)
  f1=sample(F1,replace=T)
  f2=sample(F2,replace=T)
  y=c(mean(p1),mean(p2),mean(f1),mean(f2))
  dat$y=y
  reg=lm(y~A+D,data=dat)
  summary(reg)
```

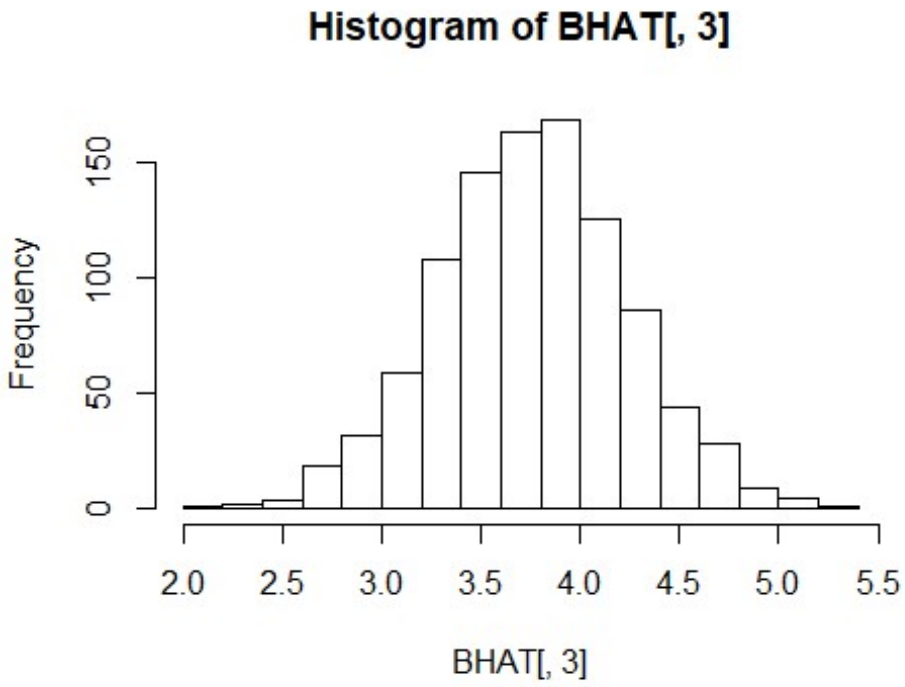
```
  b=reg$coef
  BHAT[i,]=b
}


hist(BHAT[,1])
```




```
hist(BHAT[,2])
```




```
hist(BHAT[,3])
```




```
summary(BHAT)
```

```
##        V1              V2              V3
##  Min.   :67.3   Min.   :3.62   Min.   :2.20
##  1st Qu.:68.1   1st Qu.:4.25   1st Qu.:3.44
##  Median :68.4   Median :4.43   Median :3.75
##  Mean   :68.4   Mean   :4.44   Mean   :3.76
##  3rd Qu.:68.6   3rd Qu.:4.62   3rd Qu.:4.07
##  Max.   :69.4   Max.   :5.33   Max.   :5.23
```

```
m1=mean(BHAT[,1])
m2=mean(BHAT[,2])
m3=mean(BHAT[,3])

m=c(m1,m2,m3)
CI1=quantile(BHAT[,1],p=c(0.025,0.975))
CI2=quantile(BHAT[,2],p=c(0.025,0.975))
CI3=quantile(BHAT[,3],p=c(0.025,0.975))



CI=rbind(CI1,CI2,CI3)
Result=cbind(b0,m,CI)
colnames(Result)=c("OrigMean","BootMean","LL","UL")
rownames(Result)=c("Mu","A","D")
data.frame(Result)
```

	OrigMean	BootMean	LL	UL
Mu	68.38	68.37	67.71	69.02
A	4.43	4.44	3.93	4.97
D	3.74	3.76	2.83	4.67

Bootstrapping samples across four populations

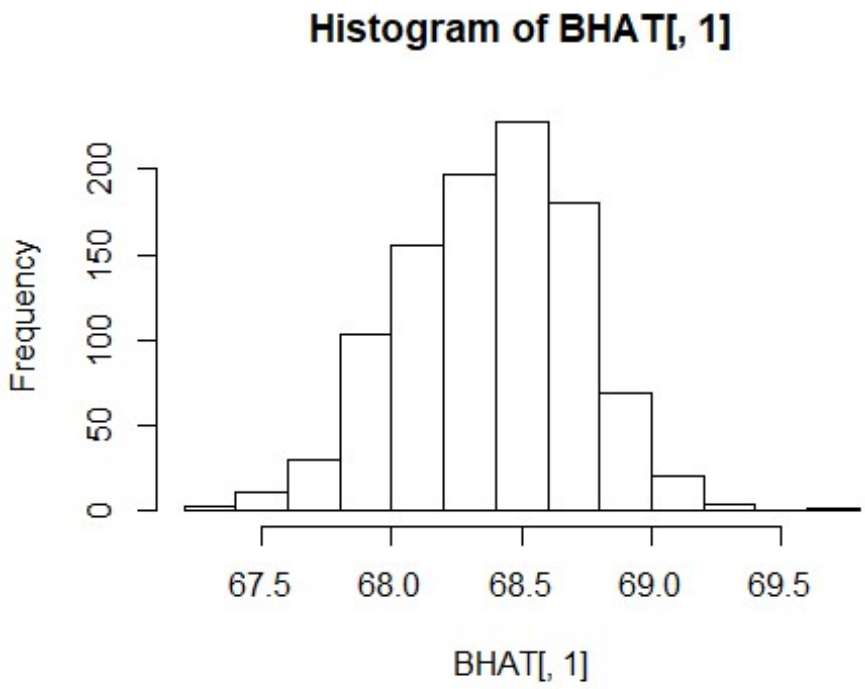
We can sometimes do bootstrapping across populations. By this way, each population may not keep the same population size.

```
obs=c(P1,P2,F1,F2)
Gen=c(rep("A1",length(P1)),rep("A2",length(P2)),rep("A3",length(F1)),rep("A4"
,length(F2)))
tapply(obs,Gen,mean)
```

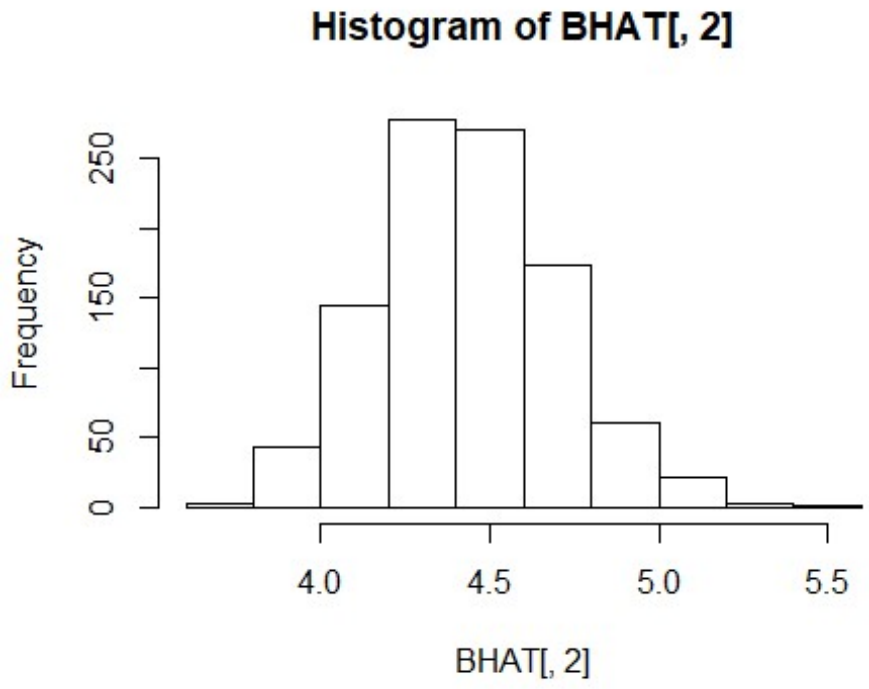
```
##   A1   A2   A3   A4
## 71.9 63.1 70.4 73.8
```

```
B=1000
BHAT=matrix(0,nrow=B,ncol=3)
n=length(obs)
for(i in 1:B){
  id=sample(n,replace=T)
  gen=Gen[id]
```

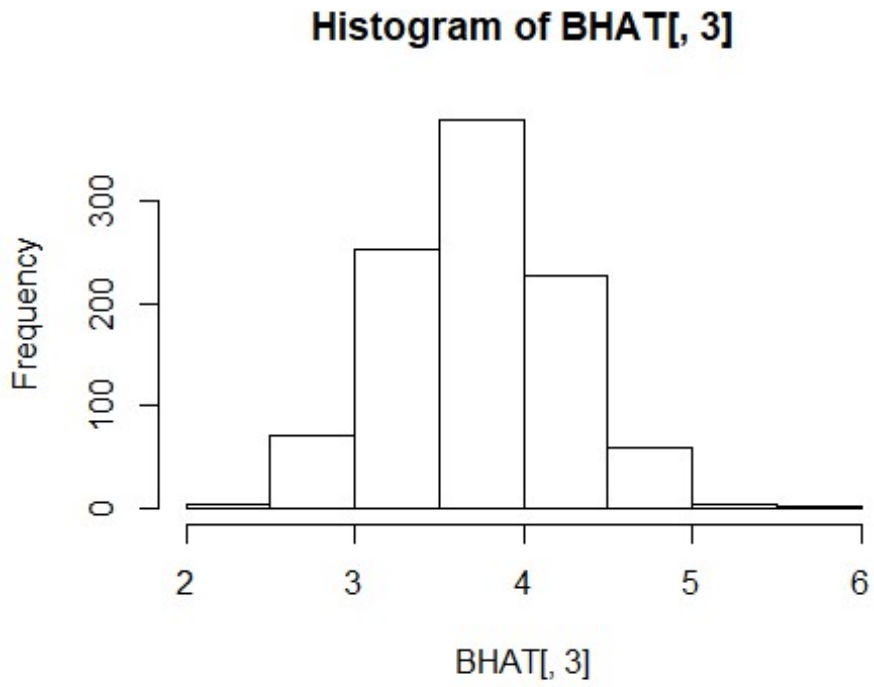
```
  obs1=obs[id]
  y=tapply(obs1,gen,mean)
  dat$y=y
  reg=lm(y~A+D,data=dat)
  summary(reg)
  b=reg$coef
  BHAT[i,]=b
}

hist(BHAT[,1])
```




```
hist(BHAT[,2])
```


Histogram of BHAT[, 2]
Frequency
0
50
150
250
4.0
4.5
5.0
5.5
BHAT[, 2]

```r
hist(BHAT[,3])
```


Histogram of BHAT[, 3]
Frequency
0
100
200
300
2
3
4
5
6
BHAT[, 3]

```
m1=mean(BHAT[,1])
m2=mean(BHAT[,2])
m3=mean(BHAT[,3])

m=c(m1,m2,m3)
CI1=quantile(BHAT[,1],p=c(0.025,0.975))
CI2=quantile(BHAT[,2],p=c(0.025,0.975))
CI3=quantile(BHAT[,3],p=c(0.025,0.975))



CI=rbind(CI1,CI2,CI3)
Result=cbind(b0,m,CI)
colnames(Result)=c("OrigMean","BootMean","LL","UL")
rownames(Result)=c("Mu","A","D")
data.frame(Result)
```

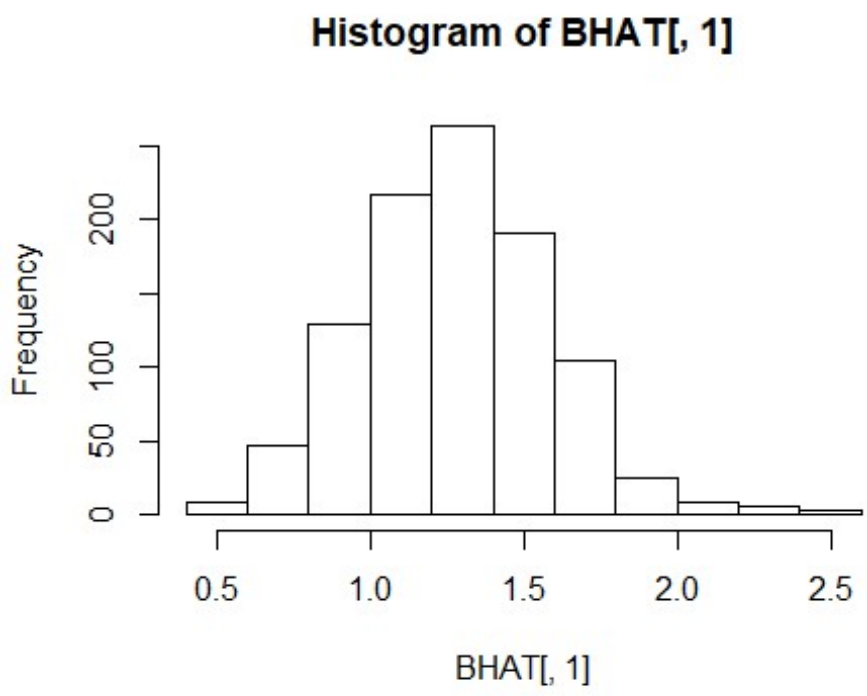
	OrigMean	BootMean	LL	UL
Mu	68.38	68.38	67.70	69.00
A	4.43	4.43	3.94	4.99
D	3.74	3.73	2.76	4.73

Bootstrapping method for genetic variance components estimation

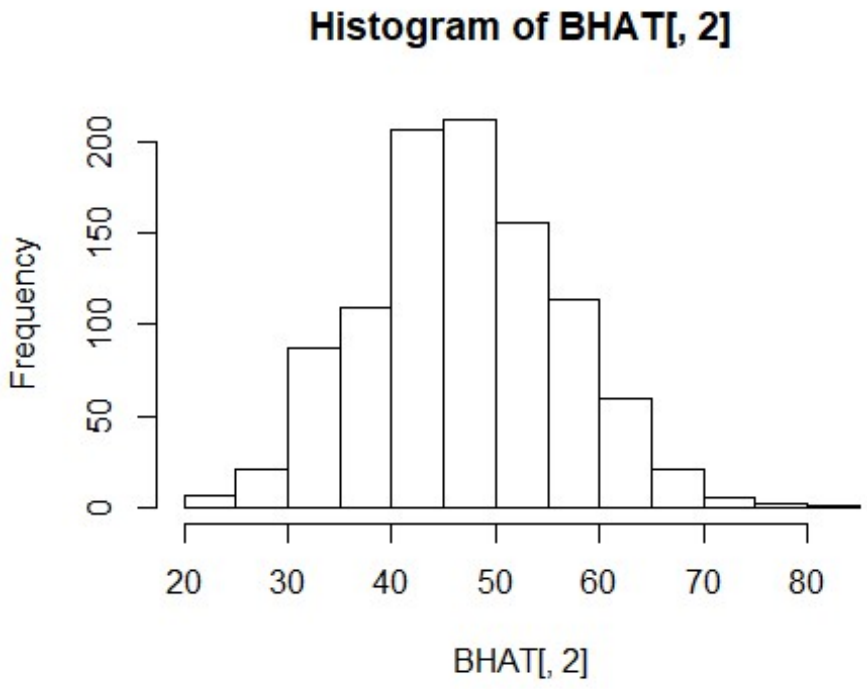
```
X=matrix(c(1,0,0,1,0,0,1,0,0,1,1,1),nrow=4,byrow=T)
## this matrix was designated for four-generation data analysis
## Since with only F1 and F2, additive and dominance variance components are
confound and can not be estimated at the same time
## However, we may be able to get Ve and VG, so the last column of matrix
needs to be dropped


y=c(var(P1),var(P2),var(F1),var(F2))

X1=X[,3]

dat=data.frame(y,X1)
colnames(dat)=c("gv","VG")
reg=lm(gv~VG,data=dat)
bhat1=reg$coef
bhat1

## (Intercept)          VG
##        1.43       48.47

(Hb_square=bhat1[2]/sum(bhat1))

##    VG
## 0.971
```

By bootstrapping within each population

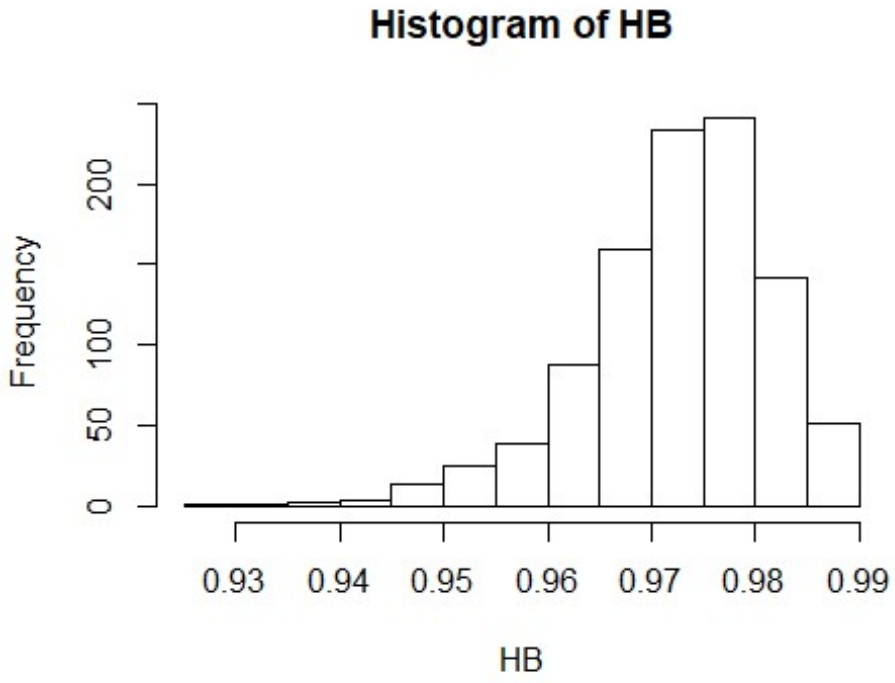
```
B=1000
BHAT=matrix(0,nrow=B,ncol=2)
HB=numeric(B)
for(i in 1:B){
  p1=sample(P1,replace=T)
  p2=sample(P2,replace=T)
  f1=sample(F1,replace=T)
  f2=sample(F2,replace=T)
  gv1=c(var(p1),var(p2),var(f1),var(f2))
  dat$gv1=gv1
  reg=lm(gv1~VG,data=dat)
  bhat=reg$coef
  BHAT[i,]=bhat
  HB[i]=bhat[2]/sum(bhat)
}
##Histogram for Ve
hist(BHAT[,1])
```




```
##Histogram for VG
hist(BHAT[,2])
```




```
##Histogram for heritability
hist(HB)
```



```r
mean(BHAT[,1])
```

```
## [1] 1.28
```

```r
mean(BHAT[,2])
```

```
## [1] 47
```

```r
mean(HB)
```

```
## [1] 0.972
```

```r
m1=mean(BHAT[,1]) # Ve
m2=mean(BHAT[,2]) # VG

m3=mean(HB)   #Heritability

m=c(m1,m2,m3)
CI1=quantile(BHAT[,1],p=c(0.025,0.975))
CI2=quantile(BHAT[,2],p=c(0.025,0.975))
CI3=quantile(HB,p=c(0.025,0.975))



CI=rbind(CI1,CI2,CI3)
orig=c(bhat1,Hb_square)
Result=cbind(orig,m,CI)
colnames(Result)=c("OrigVar","BootVar","LL","UL")
rownames(Result)=c("Ve","VG","Hb")
data.frame(Result)
```

	OrigVar	BootVar	LL	UL
Ve	1.427	1.278	0.695	1.918
VG	48.469	47.017	29.881	65.798
Hb	0.971	0.972	0.951	0.986

Bootstrapping sampling across populations

```r
Gen=c(rep("A1",length(P1)),rep("A2",length(P2)),rep("A3",length(F1)),rep("A4"
,length(F2)))
## Here A1=P1, A2=P2, A3=F1, and A4=F2 ####

obs=c(P1,P2,F1,F2)
dat=data.frame(y,X1)
colnames(dat)=c("gv","VG")
reg=lm(gv~VG,data=dat)
bhat1=reg$coef
bhat1
```

```
## (Intercept)          VG
##        1.43       48.47
```

```
(Hb_square=bhat1[2]/sum(bhat1))
```

```
##    VG
## 0.971
```

```
B=1000
BHAT=matrix(0,nrow=B,ncol=2)

HB=numeric(B)
n=length(obs)
for(i in 1:B){
  id=sample(n,replace=T)
  gen=Gen[id]
  obs1=obs[id]
  v=tapply(obs1,gen,var)
  dat$gv1=v
  reg=lm(gv1~VG,data=dat)
  bhat=reg$coef
  BHAT[i,]=bhat
  HB[i]=bhat[2]/sum(bhat)

}

m1=mean(BHAT[,1])
m2=mean(BHAT[,2])

m3=mean(HB)

m=c(m1,m2,m3)
CI1=quantile(BHAT[,1],p=c(0.025,0.975))
CI2=quantile(BHAT[,2],p=c(0.025,0.975))
CI3=quantile(HB,p=c(0.025,0.975))



CI=rbind(CI1,CI2,CI3)
orig=c(bhat1,Hb_square)
Result=cbind(orig,m,CI)
colnames(Result)=c("OrigVar","BootVar","LL","UL")
rownames(Result)=c("Ve","VG","Hb")
data.frame(Result)
```

	OrigVar	BootVar	LL	UL
Ve	1.427	1.264	0.652	2.018
VG	48.469	46.461	29.426	66.328

Hb 0.971 0.972 0.950 0.987

Permuation sampling across populations

```
###We may also try permutation test with slight changes of R codes
#######################

Gen=c(rep("A1",length(P1)),rep("A2",length(P2)),rep("A3",length(F1)),rep("A4"
,length(F2)))
## Here A1=P1, A2=P2, A3=F1, and A4=F2 ####

obs=c(P1,P2,F1,F2)
dat=data.frame(y,X1)
colnames(dat)=c("gv","VG")
reg=lm(gv~VG,data=dat)
bhat1=reg$coef
bhat1

## (Intercept)          VG
##        1.43       48.47

(Hb_square=bhat1[2]/sum(bhat1))

##    VG
## 0.971

B=1000
BHAT=matrix(0,nrow=B,ncol=2)

HB=numeric(B)
n=length(obs)
for(i in 1:B){
  id=sample(n)
  gen=Gen
  obs1=obs[id]
  v=tapply(obs1,gen,var)
  dat$gv1=v
  reg=lm(gv1~VG,data=dat)
  bhat=reg$coef
  BHAT[i,]=bhat
  HB[i]=bhat[2]/sum(bhat)

}

m1=mean(BHAT[,1])
m2=mean(BHAT[,2])

m3=mean(HB)

m=c(m1,m2,m3)
CI1=quantile(BHAT[,1],p=c(0.025,0.975))
```

```
CI2=quantile(BHAT[,2],p=c(0.025,0.975))
CI3=quantile(HB,p=c(0.025,0.975))



CI=rbind(CI1,CI2,CI3)
orig=c(bhat1,Hb_square)
Result=cbind(orig,m,CI)
colnames(Result)=c("OrigVar","PermVar","LL","UL")
rownames(Result)=c("Ve","VG","Hb")
data.frame(Result)
```

	OrigVar	PermVar	LL	UL
Ve	1.427	40.087	25.59	54.498
VG	48.469	-0.356	-28.57	26.213
Hb	0.971	-0.074	-1.04	0.494