Workshop on GWAS

Regression slope p-value and multiple hypothesis testing via simualtion studies

Prof. Lei Sun

Department of Statistical Sciences, FAS

Division of Biostatistics, DLSPH

University of Toronto

23 June, 2021

Recall what's next from yesterday:

How to use simulation to obtain the emprical p-value for

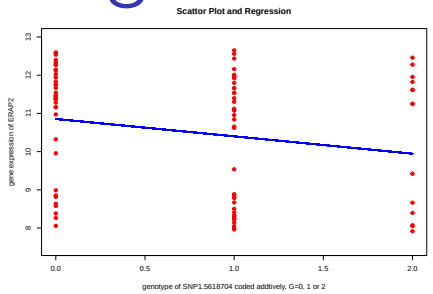
the association testing between the gene expression of *ERAP*2 (Y) and the genotypes of SNP1.5618704 coded additively.(X)

$$\text{Expected or average value of } Y = \beta_0 + \beta X.$$

That is, determine if the slope is zero, $H_0 : \beta = 0$.

What if we have a bag/family of 10^6 coins/SNPs to evaluate?

Recall the data and the regression line



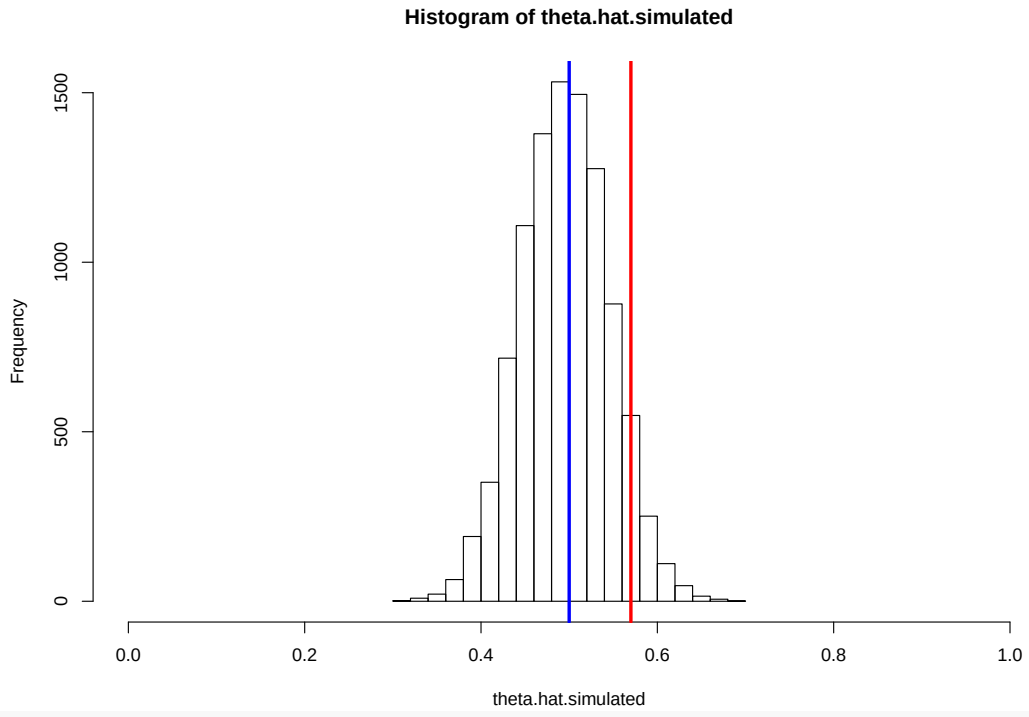
The slope (the regression coefficient) is -0.4545. The slope is not statistically different from zero: the **p-value of testing the slope = 0 is 0.0594, not statistically significant.**

```
summary(lm(y~x))
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7969 -1.7987  0.5538  1.3051  2.5135
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.8544     0.2445  44.402   <2e-16 ***
## x            -0.4545     0.2380  -1.909   0.0594 .
## ---
```

Conceptually similar to the coin example

```
set.seed(1234)
n=100; x=57; theta.0=0.5 # the sample size, observed data, null hypothesis=fair coin
theta.hat=x/n # the point estimate of the parameter based on the observed data
n.rep=10000 # the number of experiments
x.simulated=rbinom(n.rep,n,theta.0) # Draws x's using a fair coin
theta.hat.simulated=x.simulated/n # Calcuates the corresponding theta estmiates
hist(theta.hat.simulated,xlim=c(0,1)) # Displays all the estimates
abline(v=theta.0, col="blue", lwd=3) # Marks theta.0
abline(v=theta.hat, col="red", lwd=3) # Marks theta.hat inferred from the actual observed data
```




```
2*sum(theta.hat.simulated>=theta.hat)/n.rep # The empirical 2-sided p-value
```

```
## [1] 0.1956
```

Goal of this simulation, or generally the Monte Carlo method

- Create say n.rep=10,000 $\hat{\beta}_{simu,k}$'s, $k = 1, \ldots, 10,000$ where
- $\hat{\beta}_{simu,k}$'s were the regression slopes estimated from
- Datasets with the same n but we know $Y_{simu,k}$ (gene expression) and $X_{simu,k}$ (genotype of SNP1.5618704) are independent of each other (i.e. not associated with each other).

  That is, many datasets generated under the null hypothesis, $H_0 : \beta = 0$.

- The set of $\hat{\beta}_{simu,k}$'s will provide the 'background' (center, standard deviation and the shape) to interpret the actual observed $\hat{\beta} = -0.4545$.

Simulation approach 1: Permutation

We can randomly permute elements in the vector of

$$y = (y_1, y_2, \ldots, y_n)' = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{pmatrix}$$

so the original (y_i, x_i) pairing/association is destroyed!

(We can also permute the $x = (x_1, \ldots, x_n)'$)

One try

First, recall our **observed data** y (gene expression of *ERAP*2) and x (genotype of SNP1.5618704 coded additively)

```
##  [1] 10.7 10.3 12.0 11.2 11.8  8.3 12.5 12.3 11.9  9.4  8.4 12.2 11.5  8.8  8.7
## [16]  9.5 11.8  8.1  8.1  8.0 11.7  8.8 11.1 11.6 11.6 12.0 12.0 11.8 11.3  8.8
## [31] 11.3 12.4  8.9  8.1 12.4 11.4  8.4 11.4  8.2 12.6 12.6 11.4  8.2 11.4  8.3
## [46]  8.1 12.5 11.3 11.2 12.3 10.6  8.7  8.3 11.7  8.1 11.1 11.4 11.7 11.2 10.0
## [61] 11.5  8.3 12.1 12.6 12.0  7.9 11.0 11.9  8.5 11.7 11.4  8.6 11.5  8.3 11.5
## [76] 12.3 11.5  9.0  8.9  8.0 12.3 12.1  8.9 11.8  8.8 12.6  8.4 11.0 10.8  8.6
## [91] 12.0
```

```
##  [1] 1 0 0 0 0 1 0 0 0 2 1 1 1 1 1 1 1 1 1 0 0 1 2 2 1 1 0 0 1 1 0 0 0 1 0 0 0
## [39] 1 1 0 0 1 0 1 2 2 2 2 0 1 2 1 0 2 1 0 1 0 0 1 1 0 1 1 2 1 1 1 1 1 0 0 0 0
## [77] 0 0 1 1 2 0 1 2 1 0 2 0 1 0 2
```

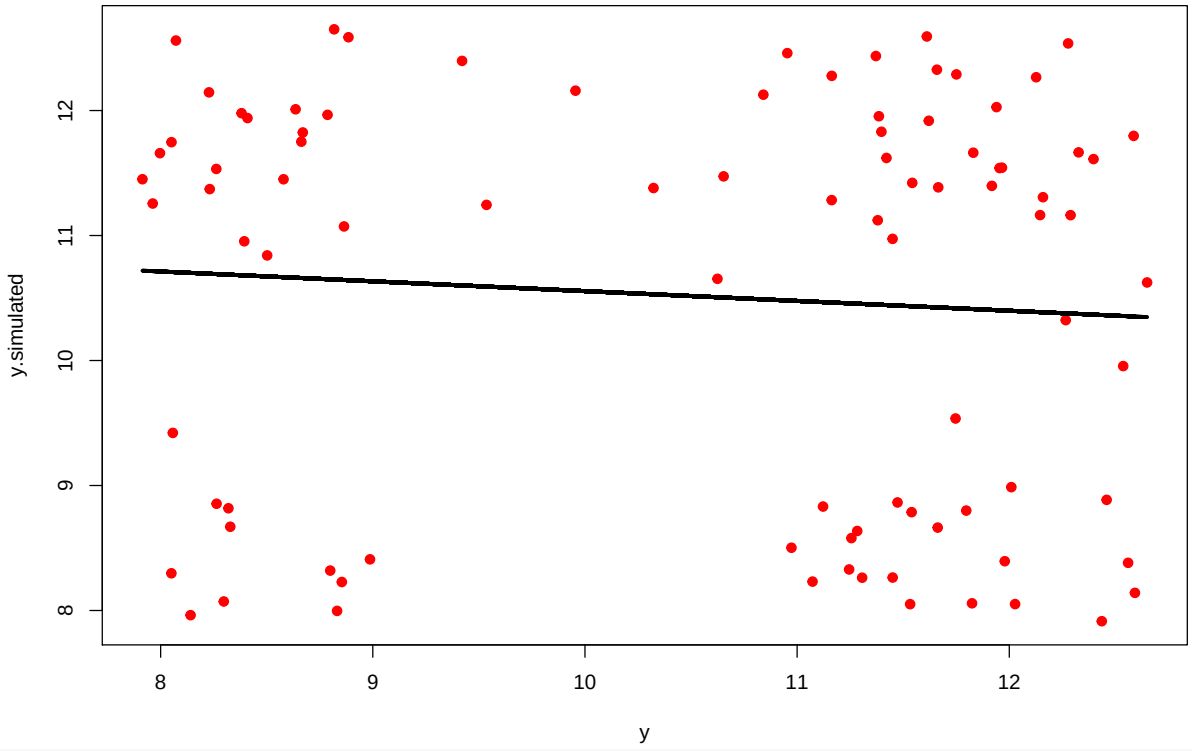
permute the observed y values

```
set.seed(101)
y.simulated=sample(y) # default is the same sample size without replacement
round(y.simulated,1)
```

```
##  [1] 11.5 11.4  8.1 12.3 11.7 11.5 10.0 11.2 12.0 12.4 11.9 11.3  8.8  8.3 11.8
## [16] 11.2  8.8  8.3  8.0 11.7 11.4 12.6  8.2 12.6 11.9  8.4 11.5 12.3  8.6  8.0
## [31]  8.3 11.6  8.2  9.4  7.9 11.6 12.0 11.0 12.1  8.4 11.8 12.0 11.4 12.4  8.1
## [46] 11.7  8.9  8.6  8.3 10.3 10.7 11.8  8.8  9.5 12.6  8.8 11.1  8.7 11.3 12.2
## [61]  8.1  8.7 12.3 10.6  9.0 11.5 12.5 11.4 10.8 12.3 11.8 12.0  8.9  8.9 11.4
## [76] 11.7  8.3  8.4 11.1 11.3 12.5 11.2 12.6  8.1 12.0  8.1 11.0  8.5 12.1 11.4
## [91] 11.5
```

Visualize the (no) correlation between the observed y and $y.simulated$

```
plot(y,y.simulated,pch=19,col="red")
lines(y,fitted(lm(y.simulated~y)),col="black",lwd=3)
```

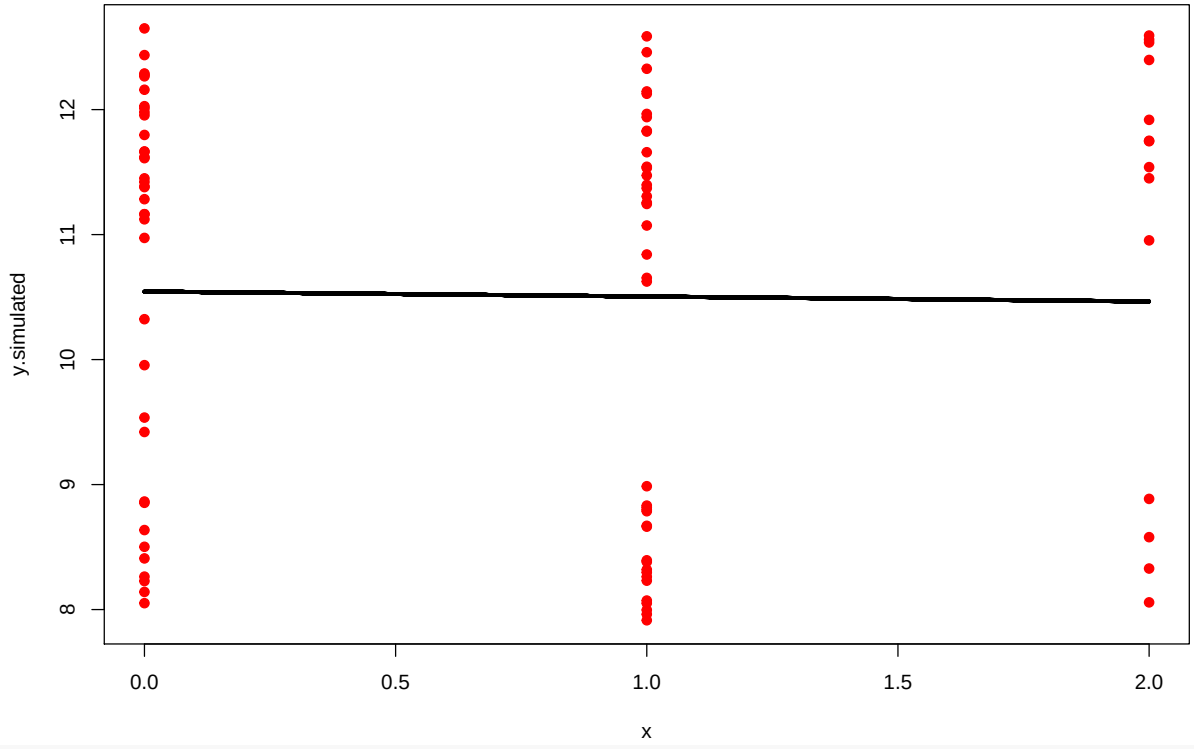



```
fit=summary(lm(y.simulated~y));fit$coefficients
```

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) 11.33983328   1.124150 10.0874752 2.112811e-16
## y           -0.07846311   0.105673 -0.7425086 4.597345e-01
```

Visualize the (no) correlation between the *y.simulated* and the observed *x*

```
plot(x,y.simulated,pch=19,col="red")
lines(x,fitted(lm(y.simulated~x)),col="black",lwd=3)
```

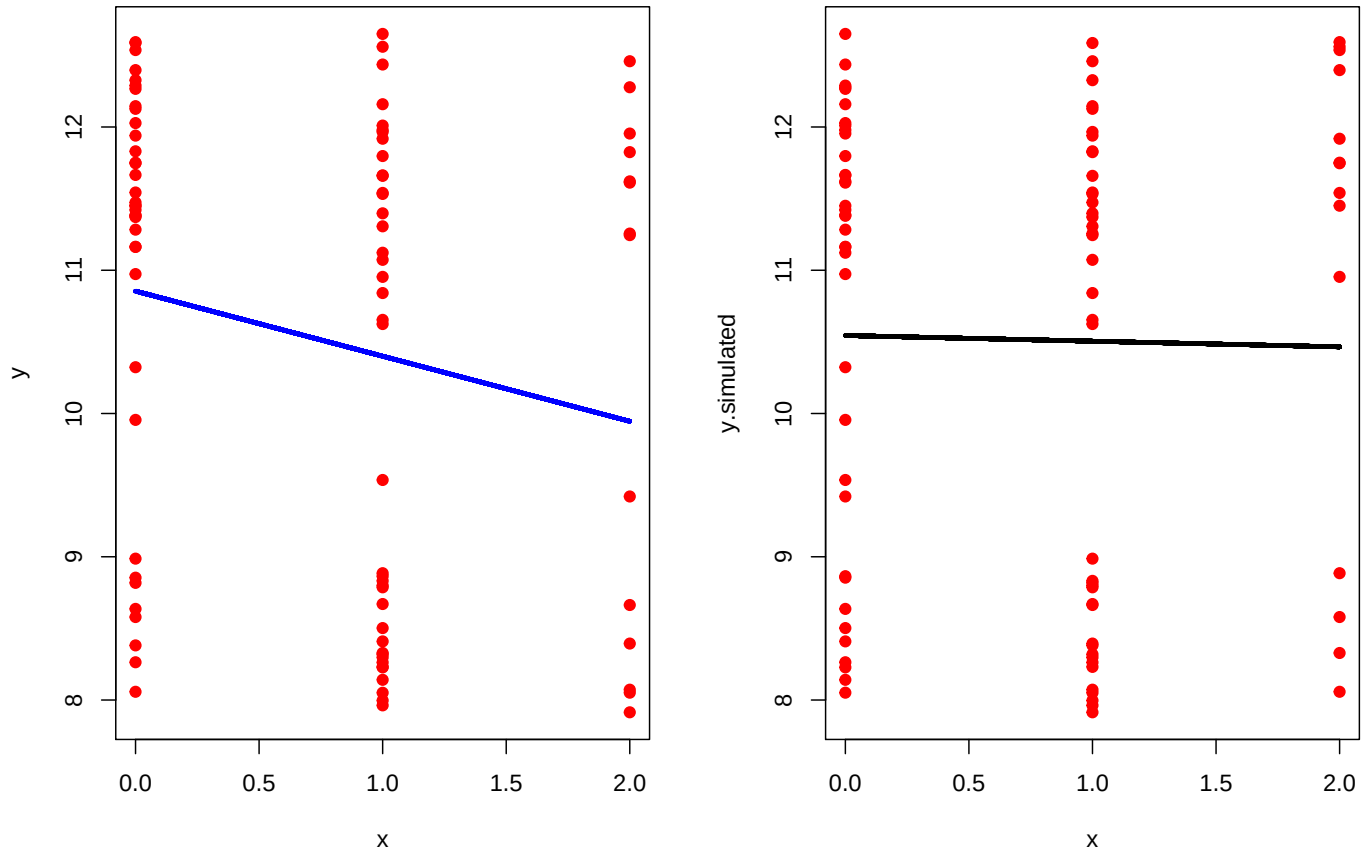



```
fit=summary(lm(y.simulated~x));fit$coefficients
```

```
##               Estimate Std. Error    t value     Pr(>|t|)
## (Intercept) 10.5444887  0.2493746 42.2837362 1.037323e-60
## x           -0.0397189  0.2427936 -0.1635912 8.704239e-01
```

Compare the observed slope, $\hat{\beta}$, with the simulated one

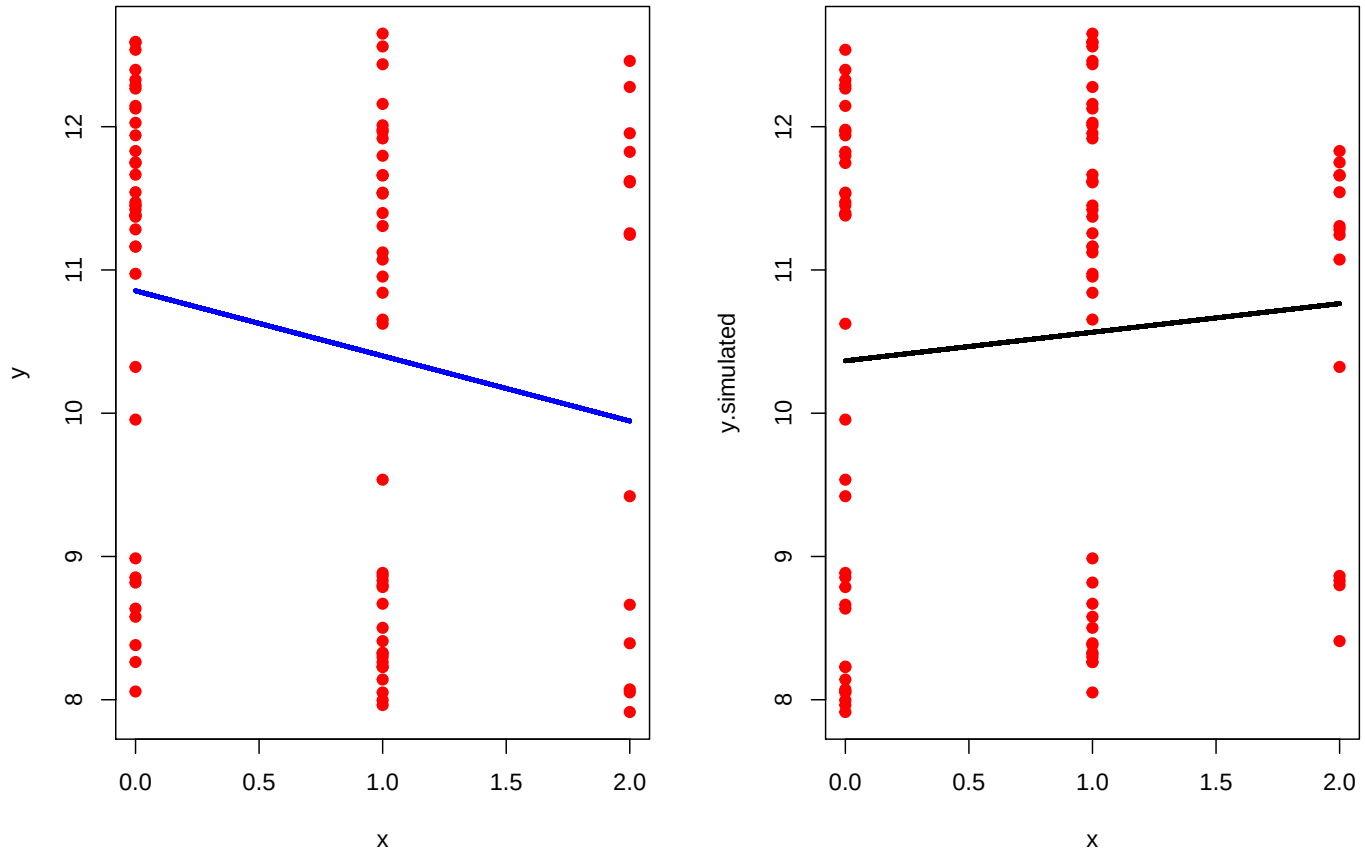
```
par(mfrow=c(1,2))
plot(x,y,pch=19,col="red");lines(x,fitted(lm(y~x)),col="blue",lwd=3)
plot(x,y.simulated,pch=19,col="red");lines(x,fitted(lm(y.simulated~x)),col="black",lwd=3)
```



"homework": Use different colors for the original data points, so we can 'track' each genotype better.

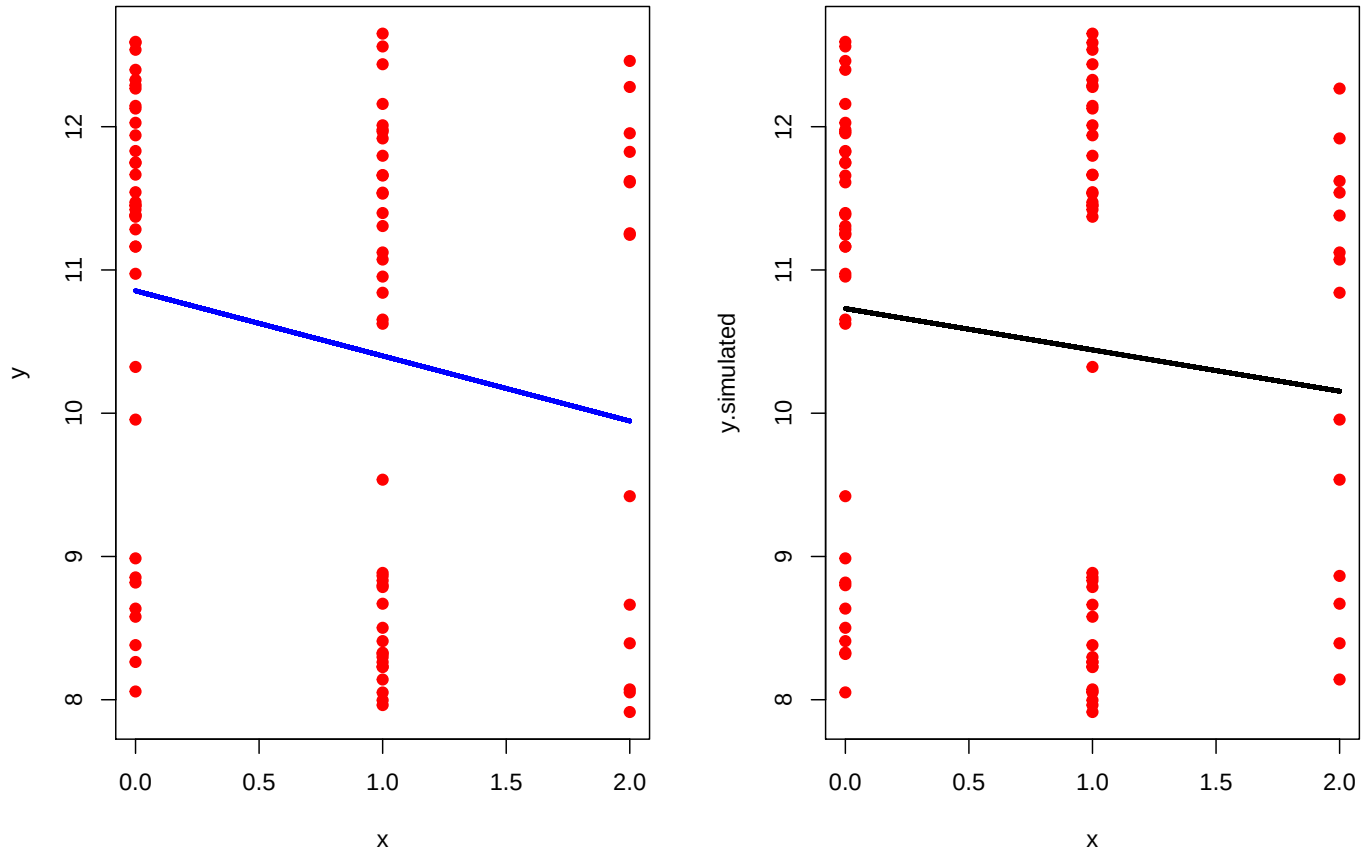
Let's try a different run

```
set.seed(102) # using a different seed now
y.simulated=sample(y)

par(mfrow=c(1,2))

plot(x,y,pch=19,col="red")
lines(x,fitted(lm(y~x)),col="blue",lwd=3)

plot(x,y.simulated,pch=19,col="red")
lines(x,fitted(lm(y.simulated~x)),col="black",lwd=3)
```



Yet another run

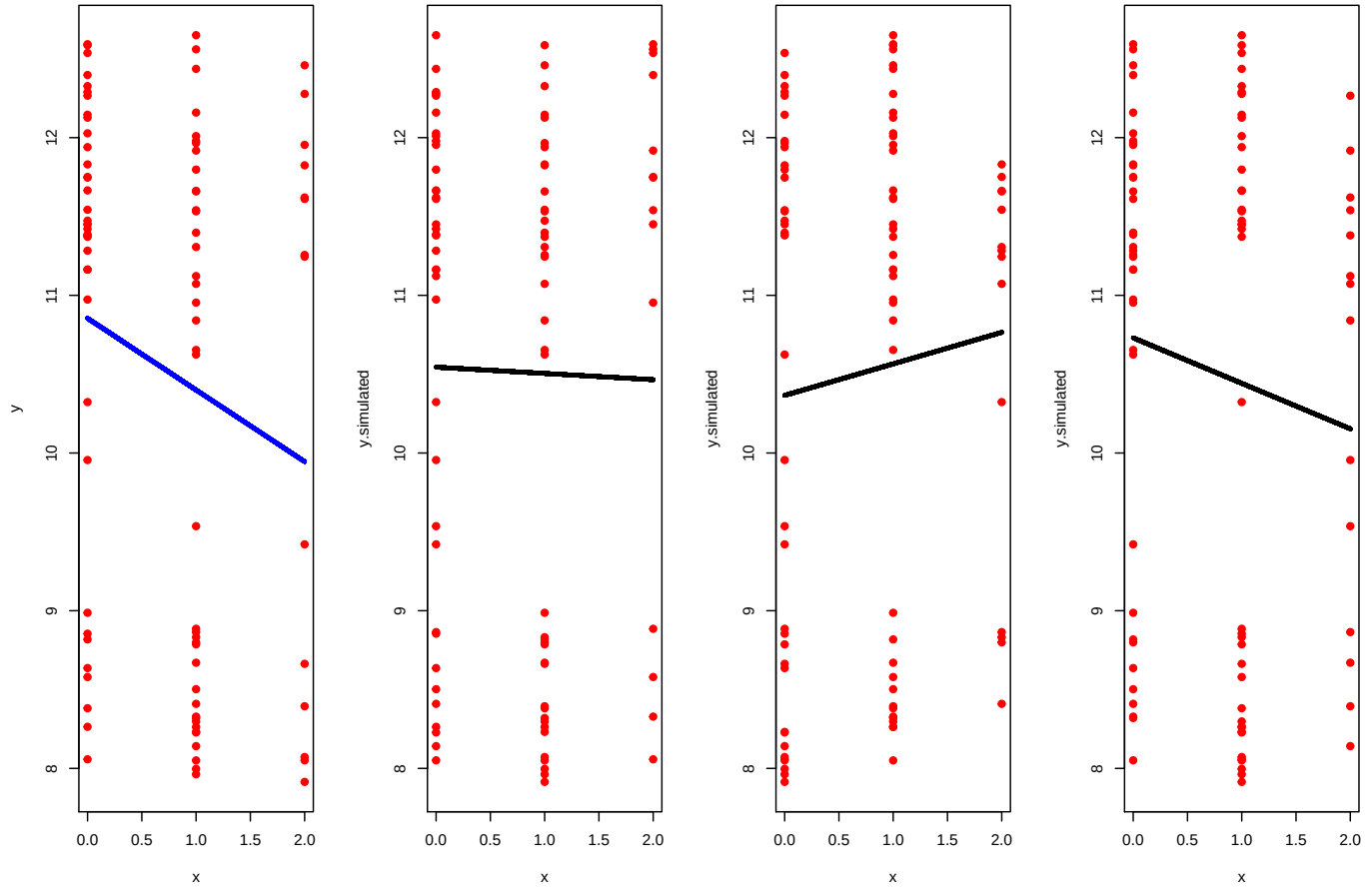
```
set.seed(121) # using a different seed now
y.simulated=sample(y)

par(mfrow=c(1,2))

plot(x,y,pch=19,col="red")
lines(x,fitted(lm(y~x)),col="blue",lwd=3)

plot(x,y.simulated,pch=19,col="red")
lines(x,fitted(lm(y.simulated~x)),col="black",lwd=3)
```



From three runs

We already have three slopes, $\hat{\beta}$'s, that are ≈ 0, $+$ and -.

The negative slope is as 'steep' as the observed one!



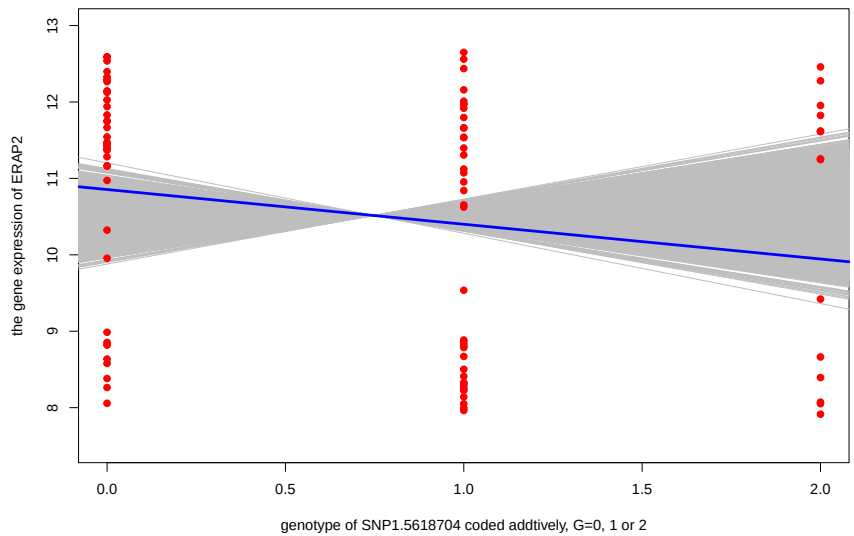
Let's try 10,000 runs all 'at once'!

```
set.seed(1234)
n.rep=10000

# to store the regression results from each replicate
# Estimate    Std. Error    t value    Pr(>|t|)
slope.result.simulated=matrix(-9,nrow=n.rep,ncol=4)

# to store the intercept information
intercept.result.simulated=matrix(-9,nrow=n.rep,ncol=4)

# run the loop; codes are clear for teaching but not efficient for large data an
for (k in 1:n.rep) {
  y.simulated=sample(y) # shuffle the data
  fit=summary(lm(y.simulated~x)) # fit the regression line
  slope.result.simulated[k,]=fit$coefficients[2,] # capture the results
  intercept.result.simulated[k,]=fit$coefficients[1,]
}

# for each replicate, results for the slope are captured by
fit$coefficients[2,]
```

```
##   Estimate Std. Error    t value   Pr(>|t|)
## -0.2437988  0.2414511 -1.0097234  0.3153652
```

All the 10,000 + 1 slopes

```
plot(x,y,type="n",ylim=c(7.5,13),
     xlab="genotype of SNP1.5618704 coded addtively, G=0, 1 or 2",
     ylab="the gene expression of ERAP2")

# plot all the n.rep regression lines
for(k in 1:n.rep)
  abline(a=intercept.result.simulated[k,1],
         b=slope.result.simulated[k,1],col="gray",lwd=.5)

# add the observed data and regression line
points(x,y,pch=19,col="red")
fit=summary(lm(y~x))
slope.obs=fit$coefficients[2,1];intercept.obs=fit$coefficients[1,1]
abline(a=intercept.obs,b=slope.obs,col="blue",lwd=3)
```



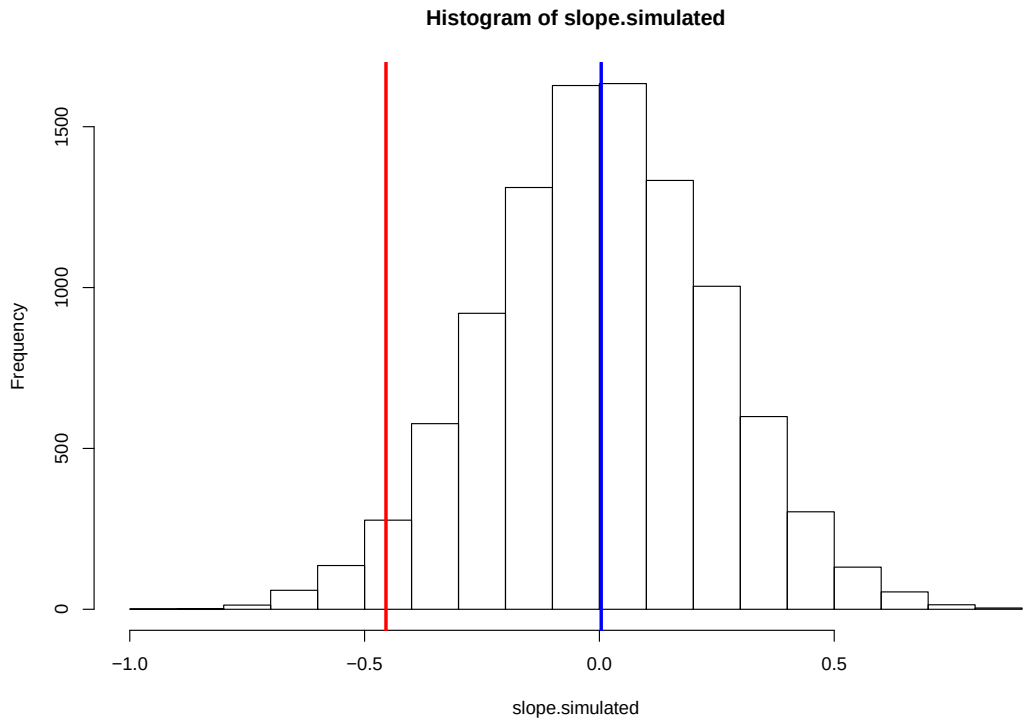
N.B. There is an over-plotting issue here.

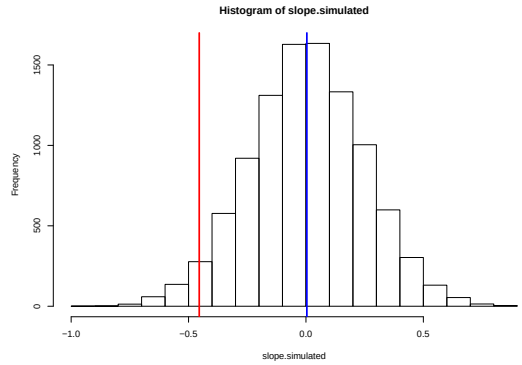
Digesting the results

```
slope.simulated=slope.result.simulated[,1]
hist(slope.simulated); abline(v=mean(slope.simulated), col="blue", lwd=3)

# On average, the beta.simulated should centered at zero
#    since there were no Y.simulated and X association
mean(slope.simulated)
```

```
## [1] 0.003843196
```

```
# Marks the beta hat inferred from the actual observed data
abline(v=slope.obs, col="red", lwd=3)
```






```
# The observed beta Estimate
slope.obs
```

```
## [1] -0.4544541
```

```
# How varied are the slope.simulated as measured by Std. Error
sqrt(var(slope.simulated))
```

```
## [1] 0.2430519
```

```
# The t-value obtained from our simulation
(slope.obs-0)/sqrt(var(slope.simulated))
```

```
## [1] -1.869782
```

```
# The empirical p-value
2*sum(slope.simulated<=slope.obs)/n.rep
```

```
## [1] 0.0618
```

Compared with the results from the lm() function

```
fit=summary(lm(y~x));fit$coefficients[2,]
```

```
##    Estimate  Std. Error     t value    Pr(>|t|)
## -0.45445405  0.23800403 -1.90943848  0.05942701
```

Permute X instead Y, and redo all analyses and plots.

Discussion 1: How to permute if the regression model is multivariate, i.e. including additional covariates such as age and sex, e.g.

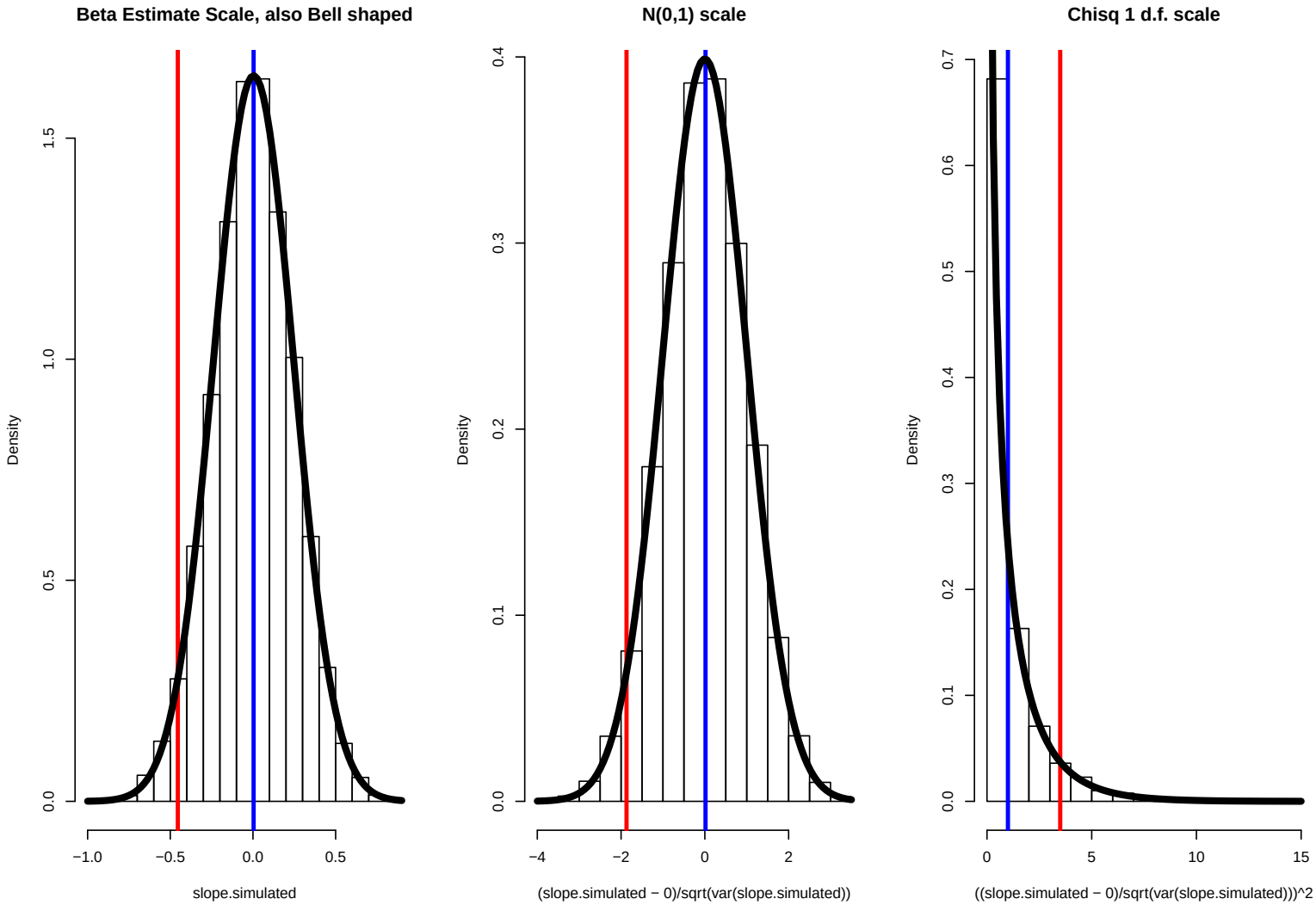
$$E(Y) = \beta_0 + \beta_X X \text{ (SNP genotype)} + \beta_Z Z \text{ (Sex)}$$

Berrett et al. (2020). *Journal of the Royal Statistical Society Series B (Statistical Methodology)*. The conditional permutation test for independence while controlling for confounders.

Discussion 2: How to permute if there are genetic related individuals?

Abney (2015). *Genetic Epidemiology*. Permutation Testing in the Presence of Polygenic Variation.

Different test statistics and different 'distances' between obs and expected under the null, but the same statistical conclusion!

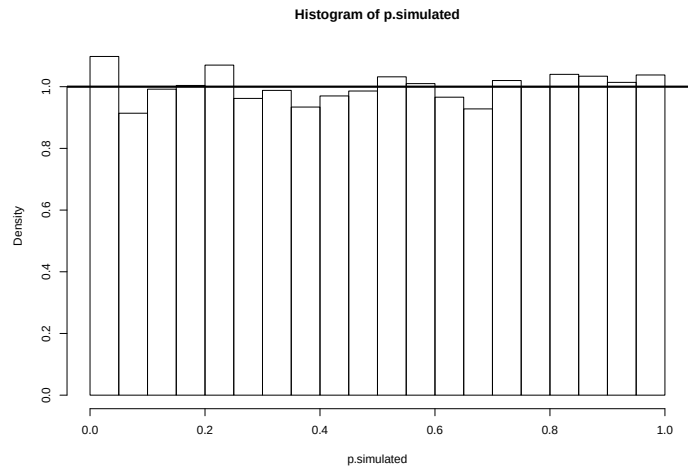


Towards multiple hypothesis testing

From the previous simulation study, we already have

10,000 p-values under the null hypothesis of no association, for testing the 10,000 slopes, $H_0 : \hat{\beta}_{simulated,\, k} = 0$, from the $Y_{permutated,\, k}$ vs. X regression analyses.

```
p.simulated=slope.result.simulated[,4]
hist(p.simulated,freq=F,breaks=seq(0,1,0.05));abline(h=1,lwd=3)
```




```
summary(p.simulated)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0000791 0.2452301 0.5040046 0.5017130 0.7564964 0.9999855
```

There are MANY small p-values even though $Y_{permutated,\, k}$ and X not associated!

```
sum(p.simulated<0.05)
```

```
## [1] 549
```

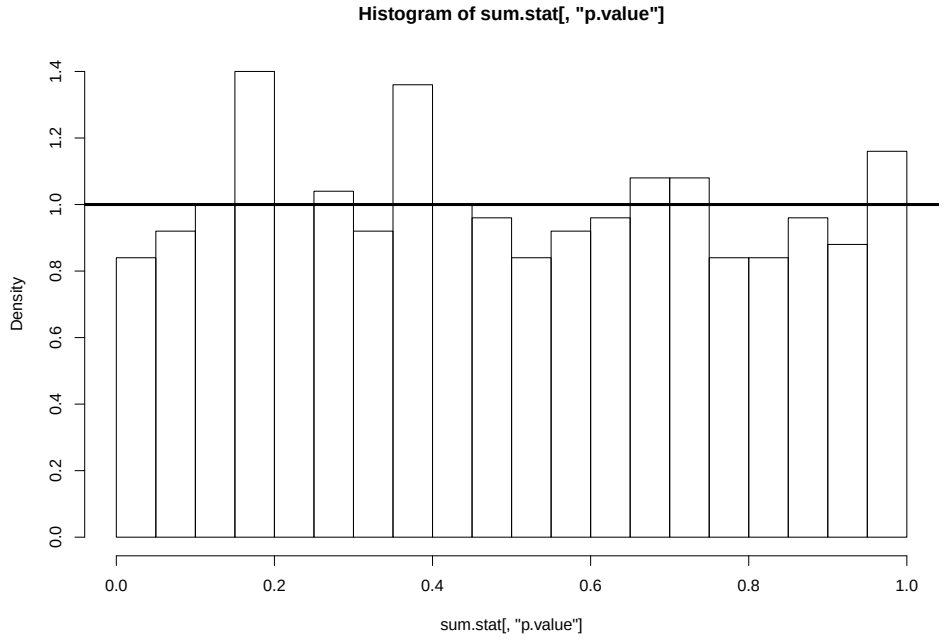
A more GWAS-mimicing simulation study

Y drawn from N(0,1), and G_j for each of the nsnp SNPs drawn based on HWE, using a MAF randomly drawn from Unif(0.05,0.5). **No relationship/asssociation between Y and all the G_j's.**

```
set.seed(101)

nsample=1000; nsnp=500 # less than 10^6 and no LD (correlation between SNPs) for now

G=matrix(-9,nrow = nsample,ncol = nsnp) # the genotype matrix
maf=runif(nsnp,min=0.05,max=0.5) # MAF randomly drawn from Unif(0,05,0.5)
maf.hat=rep(-9,nsnp)
nsnp.true=0 # number of truly associated SNPs
beta.true=0 # non effect to study type 1 error
beta=c(rep(beta.true,nsnp.true),rep(0,(nsnp-nsnp.true)))
betaG=rep(0,nsample)

for(j in 1:nsnp){ # Can be sped-up without the loop.
  nG=rmultinom(1,size=nsample,prob=c((1-maf[j])^2,2*maf[j]*(1-maf[j]), maf[j]^2))
  maf.hat[j]=(2*nG[3]+nG[2])/(2*nsample) # MAF estimated from the sample
  G[,j]=sample(c(rep(0,nG[1]),rep(1,nG[2]),rep(2,nG[3]))) # shuffle the G; no LD
  betaG=betaG+beta[j]*G[,j]
}

beta.0=0;sigma=1;e=rnorm(nsample,mean=0,sd=sigma)
Y=beta.0+betaG+e # the phenotype vector

sum.stat=matrix(-9,nrow=nsnp,ncol=6)
colnames(sum.stat)=c("MAF", "MAF.hat", "beta.hat", "se", "Z.value", "p.value")
for(j in 1:nsnp){
  fit=lm(Y~G[,j]); sum.stat[j,]=c(maf[j],maf.hat[j],summary(fit)$coefficients[2,])
}
```

Digesting the results

```
hist(sum.stat[,"p.value"],freq=F,breaks=seq(0,1,0.05));abline(h=1,lwd=3)
```



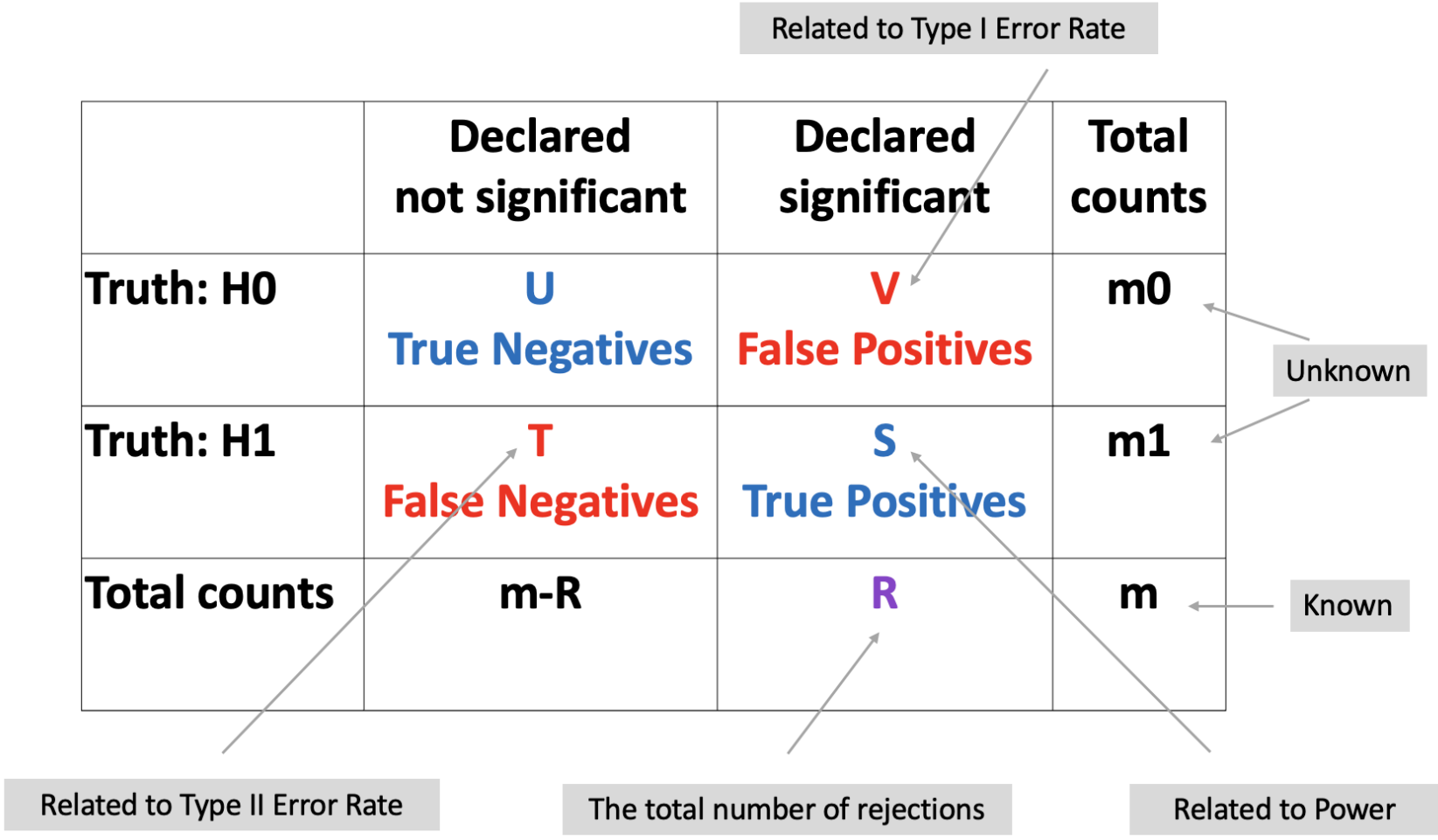
Again, there are MANY small p-values even though $Y_{simulated}$ and X_j's (i.e. G_j, genotypes of all the SNPs) NOT associated!

```
summary(sum.stat[,"p.value"])
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0004707 0.2394612 0.4795711 0.4937862 0.7320732 0.9989480
```

```
c(sum(sum.stat[,"p.value"]<=0.05),sum(sum.stat[,"p.value"]<=0.05)/n.rep)
```

```
## [1] 21.0000  0.0021
```



Question: How to measure false positive rate or type I error rate: family-wise error rate (FWER) vs. false discovery rate (FDR)?

$$FWER = Pr(V \geq 1) \quad \text{vs.} \quad FDR = E(\frac{V}{R})$$