# Lecture 6: Covariate adjustment in logistic regression

## HIV example

This is an example of no classical confounding but operational confounding. We can create two versions of this data set to make the analysis easier.

```
noclassic.data <- data.frame(city=c(rep('San Francisco',2),rep('New
York',2)),
                             known.hiv=c(0,1,0,1),
                             risky=c(10,25,50,75),
                             total=rep(100,4)
                             )

noclassic.counts <- data.frame(city=c(rep('San Francisco',4),rep('New
York',4)),
                               known.hiv=rep(0:1,4),
                               risky=rep(c(0,0,1,1),2),
                               count=c(90,75,10,25,50,25,50,75)
                               )
```

First we want to look at the association between outcome and exposure within each level of city

```
xtabs(count ~ risky+known.hiv+city,data=noclassic.counts)
```

```
## , , city = New York
##
##      known.hiv
## risky  0  1
##     0 50 25
##     1 50 75
##
## , , city = San Francisco
##
##      known.hiv
## risky  0  1
##     0 90 75
##     1 10 25
```

```
# odds ratio calculation for San Fran
mod.sf <- glm(cbind(risky,total-risky) ~ known.hiv, subset=(city=='San
Francisco'),
              family=binomial,data=noclassic.data)
exp(coef(mod.sf)[2])
```

```
## known.hiv
##         3
```

```
# odds ratio calculation for NYC
mod.ny <- glm(cbind(risky,total-risky) ~ known.hiv, subset=(city=='New
York'),
              family=binomial,data=noclassic.data)
exp(coef(mod.ny)[2])

## known.hiv
##         3
```

They have the same odds ratio.

Now we look at the association between city and exposure.

```
cityhiv.tab <- xtabs(count ~ city+known.hiv,data=noclassic.counts)
cityhiv.tab

##                known.hiv
## city              0   1
##   New York      100 100
##   San Francisco 100 100

# OR is
cityhiv.tab[1,1]*cityhiv.tab[2,2]/(cityhiv.tab[1,2]*cityhiv.tab[2,1])

## [1] 1
```

So city is NOT a classical confounder since city and exposure are not associated.

Now we look at the association between city and outcome, limiting to the unexposed.

```
cityriskhiv.tab <- xtabs(count ~ city+risky+known.hiv,data=noclassic.counts)
# (among the unexposed)
cityriskhiv.tab[,,1]

##                risky
## city             0  1
##   New York      50 50
##   San Francisco 90 10

# OR is
cityriskhiv.tab[1,1,1]*cityriskhiv.tab[2,2,1]/(cityriskhiv.tab[1,2,1]*cityris
khiv.tab[2,1,1])

## [1] 0.1111111
```

This odds ratio is the inverse of what is on the lecture slides because we have reversed the rows. The interpretation is still the same: odds of risky behavior among those living in New York are 9 times higher than the odds of risky behavior among those living in San Francisco. This is the same as saying that the odds of risky behavior in San Fran are 1/9 those in NYC.

Since the possible confounder (city) is associated with the outcome (risky behavior) but not the exposure (known HIV+ status), we call this a precision variable. This means that

including it in our regression models should improve our estimate of the exposure-outcome relationship. Let's see if that's the case for this example.

```
mod1 <- glm(cbind(risky,total-risky) ~ known.hiv,
family=binomial,data=noclassic.data)
# crude OR is
exp(coef(mod1)[2])

## known.hiv
##  2.333333

mod2 <- glm(cbind(risky,total-risky) ~ known.hiv+city,
family=binomial,data=noclassic.data)
# adjusted OR is
exp(coef(mod2)[2])

## known.hiv
##         3

# city is an operational confounder since the exposure effect changes when we
adjust for it
(exp(coef(mod1)[2])-exp(coef(mod2)[2]))/exp(coef(mod2)[2])

##  known.hiv
## -0.2222222
```

We can look at the z statistics for each model:

```
# unadjusted
round(summary(mod1)$coef[2,3],3)

## [1] 4.048

# adjusted for city
round(summary(mod2)$coef[2,3],3)

## [1] 4.502
```

The model with city has a higher z score, implying we have more power to detect the association between known HIV+ status and risky behavior when we include it in our models.

## Confounding and mediation in linear regression

For linear models, the classical and operational definitions of confounding coincide. That is to say that the difference between the adjusted and unadjusted estimates of the exposure outcome relationship is equal to the product of the regression coefficient for the confounder in the adjusted model and the regression coefficient for the exposure in the model for the confounder conditional on the exposure.

The difference between confounding and mediation is causal rather than mathematical. The third classical criterion for confounding simply states that the third variable (i.e.,

possible confounder or mediator) is NOT a mediator: the exposure does not "cause" the third variable; rather, the third variable "causes' both the exposure and outcome.

We will look at a mediation example using simulations. First we look at a null case, with no mediation. We can do this either by setting alpha (association between x and z) or beta (association between z and y given x) equal to zero. We start with a small sample.

```
library(DescTools)

gamma1 <- gamma2 <- gamma3 <- 0

tau.prime <- .1

# association strength between x and z
a <- 0
# association strength between z and y given x
b <- .5

tau <- tau.prime+a*b

n <- 20

trials <- 1000
prod.se <- prod.est <- numeric()

system.time(
for(tr in 1:trials) {
  set.seed(tr)
  # generate the data
  x <- rbinom(n,1,.5)
  z <- gamma2 + a*x + rnorm(n)
  y <- gamma1 + tau*x + b*z + rnorm(n)
  # fit the models
  mod.crude <- lm(y ~ x)
  mod.adj <- lm(y ~ x+z)
  mod.conf <- lm(z ~ x)
  # compute the estimate and standard error
  prod.est[tr] <- coef(mod.conf)[2]*coef(mod.adj)[3]
  prod.se[tr] <- sqrt(coef(mod.conf)[2]^2*vcov(mod.adj)[3,3] +
coef(mod.adj)[3]^2*vcov(mod.conf)[2,2])
}
)

##    user  system elapsed
##    2.12    0.00    2.13
```

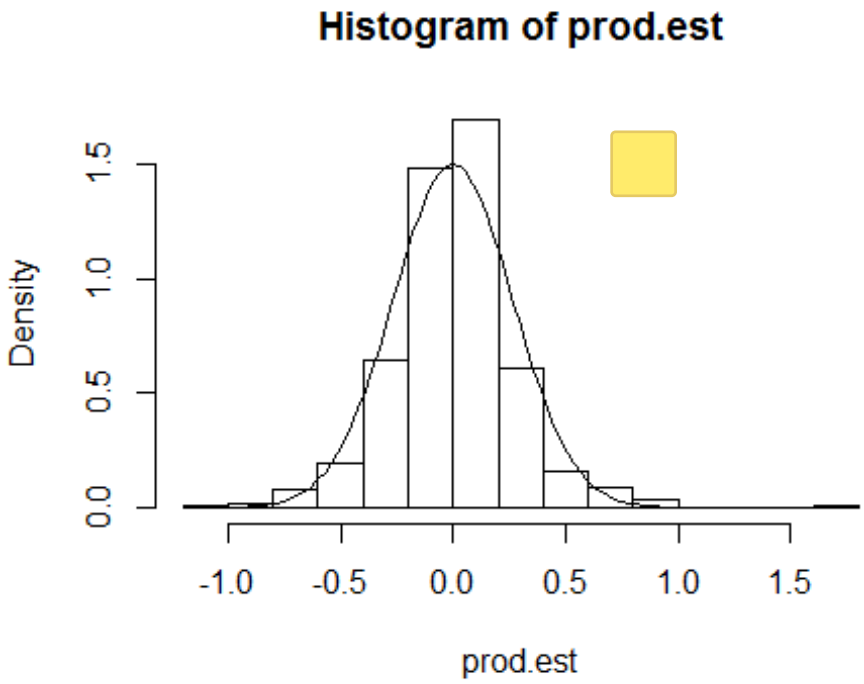
Now we examine the results.

```
# compare mean of estimates
mean(prod.est)
```

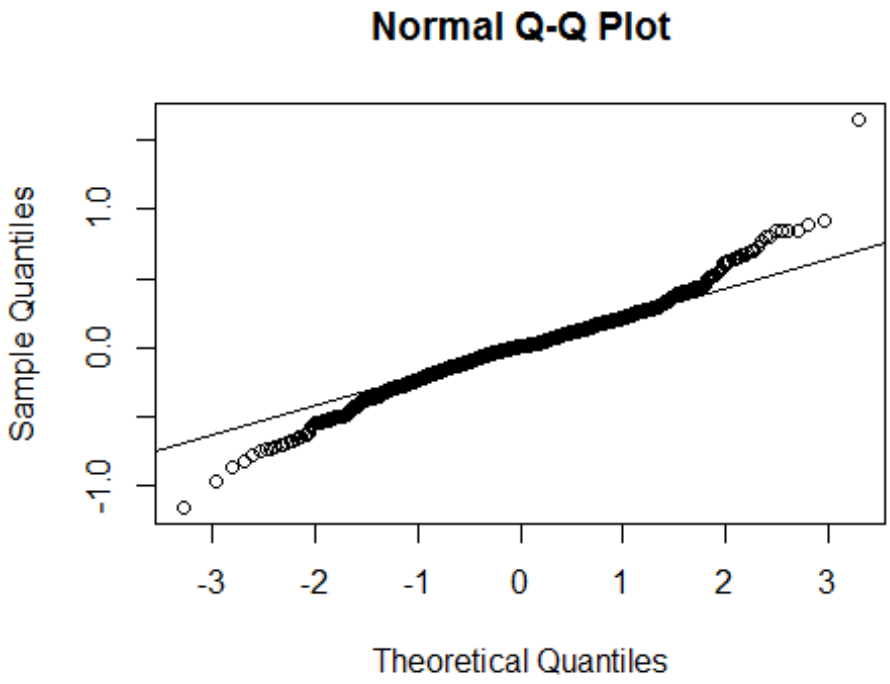
```
## [1] -0.001026413

# with true parameter value
a*b

## [1] 0

# compare standard deviation of estimates
sd(prod.est)

## [1] 0.266052

# with average estimated standard error
mean(prod.se)

## [1] 0.2667171

# look at power
# test statistic is
z <- prod.est/prod.se
# 2-sided p-values are
pvals <- 2*pnorm(-abs(z))
# power is proportion of times the null is rejected
mean(pvals<.05)

## [1] 0.006

# under the null, this should be close to 0.05
```

So there is good agreement between the estimated and true parameter values, and the estimated and true standard errors, but the rejection rate is too low. This implies that the normal approximation for the sampling distribution of a*b is not good for this sample size.

```
# look at distribution of estimates
hist(prod.est,freq=FALSE)
# compared with a normal curve
curve(dnorm(x,mean=mean(prod.est),sd=sd(prod.est)),add=TRUE)
```




```
# this test is not strictly correct since it involves estimated parameters
ks.test(prod.est,pnorm,mean=mean(prod.est),sd=sd(prod.est))

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  prod.est
## D = 0.053816, p-value = 0.006102
## alternative hypothesis: two-sided

# ... this test should be more accurate
LillieTest(prod.est)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  prod.est
## D = 0.053816, p-value = 3.41e-07

# another way to look at normality
qqnorm(prod.est)
# these points should fall on this line if the sampling distribution is
normal
qqline(prod.est)
```




Let's push the sample size up to 1000.

```
##    user  system elapsed
##    4.26    0.02    4.28
```

Now we look at the results again.

```
# compare mean of estimates
mean(prod.est)

## [1] 0.0001846271

# with true parameter value
a*b

## [1] 0

# compare standard deviation of estimates
sd(prod.est)

## [1] 0.03064648

# with average estimated standard error
mean(prod.se)

## [1] 0.03176702

# look at power
# test statistic is
```

```r
z <- prod.est/prod.se
# 2-sided p-values are
pvals <- 2*pnorm(-abs(z))
# power is proportion of times the null is rejected
mean(pvals<.05)
```

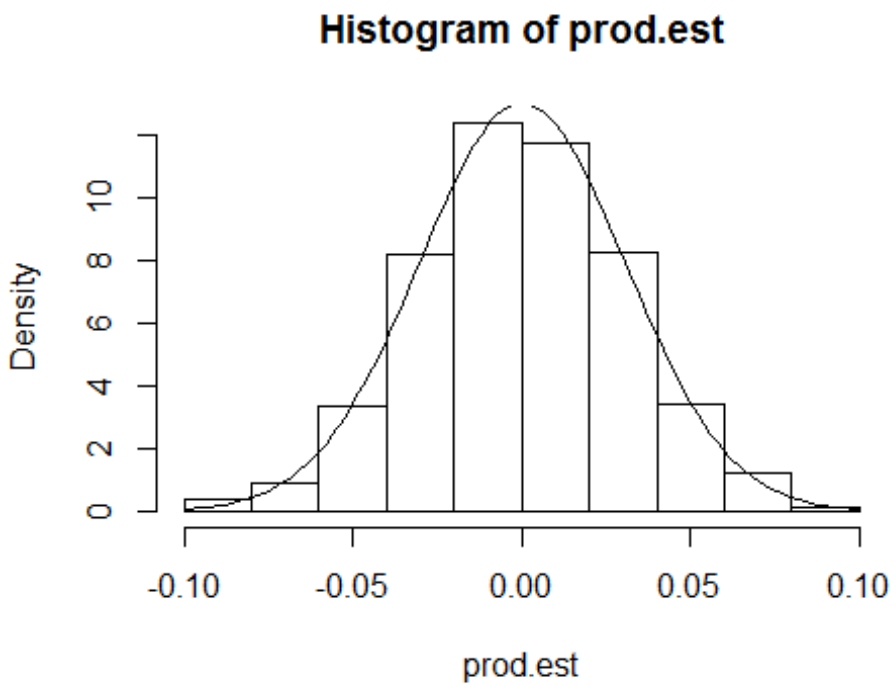
```
## [1] 0.045
```

```r
# under the null, this should be close to 0.05
```

Now the rejection rate is closer to the nominal level.

We can look at how close the sampling distribution is to normaility again too.

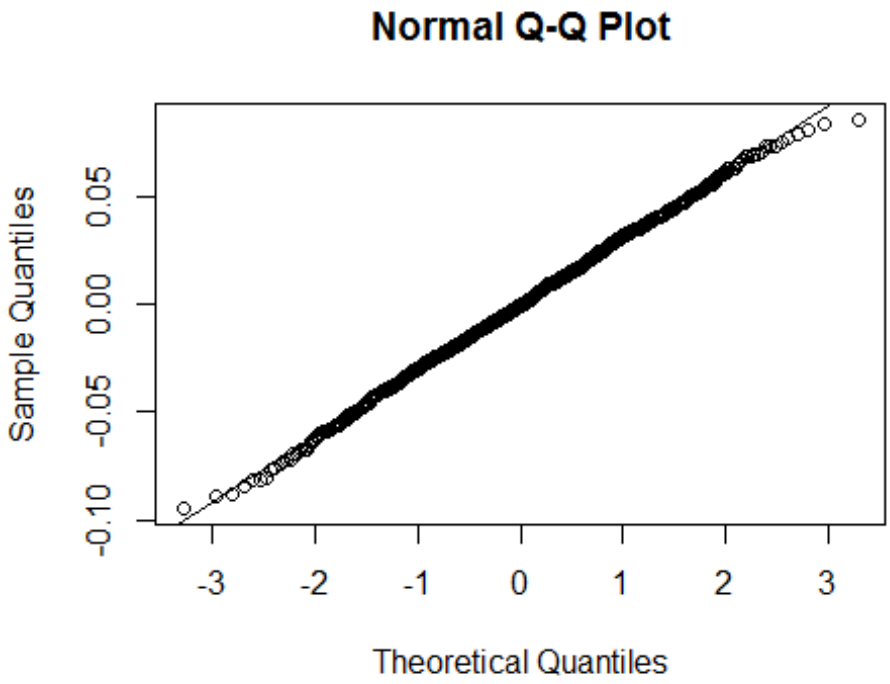
```r
# look at distribution of estimates
hist(prod.est,freq=FALSE)
# compared with a normal curve
curve(dnorm(x,mean=mean(prod.est),sd=sd(prod.est)),add=TRUE)
```




```r
# this test is not strictly correct since it involves estimated parameters
ks.test(prod.est,pnorm,mean=mean(prod.est),sd=sd(prod.est))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  prod.est
## D = 0.01561, p-value = 0.9679
## alternative hypothesis: two-sided
```

```
# ... this test should be more accurate
LillieTest(prod.est)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  prod.est
## D = 0.01561, p-value = 0.8027

# another way to look at normality
qqnorm(prod.est)
# these points should fall on this line if the sampling distribution is
normal
qqline(prod.est)
```




What happens when the alternative hypothesis is true?

```
# association strength between x and z
a <- 1
# association strength between z and y given x
b <- .5

##    user  system elapsed
##    4.08    0.00    4.07
```

Now we look at the results

```r
# compare mean of estimates
mean(prod.est)
```

```
## [1] 0.5017678
```

```r
# with true parameter value
a*b
```

```
## [1] 0.5
```

```r
# compare standard deviation of estimates
sd(prod.est)
```

```
## [1] 0.04308643
```

```r
# with average estimated standard error
mean(prod.se)
```

```
## [1] 0.04486976
```

```r
# look at power
# test statistic is
z <- prod.est/prod.se
# 2-sided p-values are
pvals <- 2*pnorm(-abs(z))
# power is proportion of times the null is rejected
mean(pvals<.05)
```

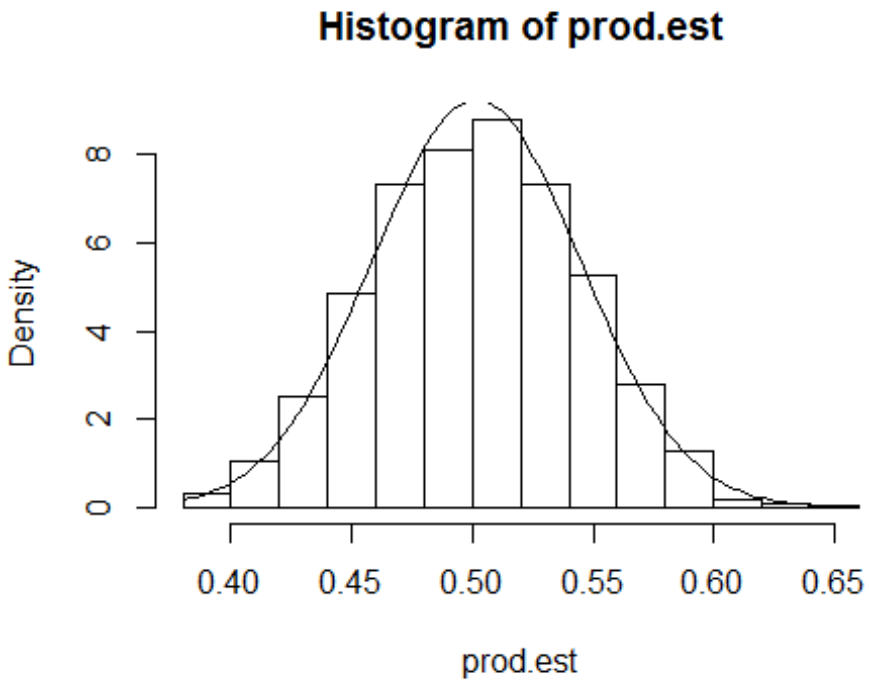
```
## [1] 1
```

```r
# under the null, this should be close to 0.05
```

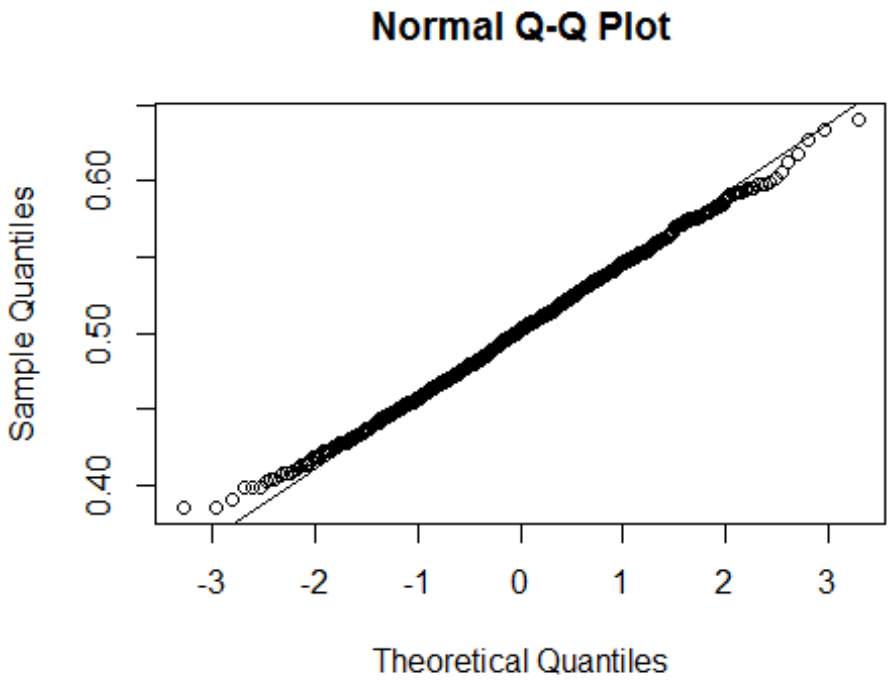
Rejection rate gives power now since we are simulating under the alternative hypothesis.

We can look at how close the sampling distribution is to normality again too.

```r
# look at distribution of estimates
hist(prod.est,freq=FALSE)
# compared with a normal curve
curve(dnorm(x,mean=mean(prod.est),sd=sd(prod.est)),add=TRUE)
```




```
# this test is not strictly correct since it involves estimated parameters
ks.test(prod.est,pnorm,mean=mean(prod.est),sd=sd(prod.est))

##
##  One-sample Kolmogorov-Smirnov test
##
## data:  prod.est
## D = 0.020176, p-value = 0.8103
## alternative hypothesis: two-sided

# ... this test should be more accurate
LillieTest(prod.est)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  prod.est
## D = 0.020176, p-value = 0.4161

# another way to look at normality
qqnorm(prod.est)
# these points should fall on this line if the sampling distribution is
normal
qqline(prod.est)
```




A smaller sample will decrease power.

```
##    user  system elapsed
##    2.15    0.00    2.17
```

Now we look at the results

```
# compare mean of estimates
mean(prod.est)

## [1] 0.513491

# with true parameter value
a*b

## [1] 0.5

# compare standard deviation of estimates
sd(prod.est)

## [1] 0.356482

# with average estimated standard error
mean(prod.se)

## [1] 0.3615026

# look at power
# test statistic is
```

```r
z <- prod.est/prod.se
# 2-sided p-values are
pvals <- 2*pnorm(-abs(z))
# power is proportion of times the null is rejected
mean(pvals<.05)
```

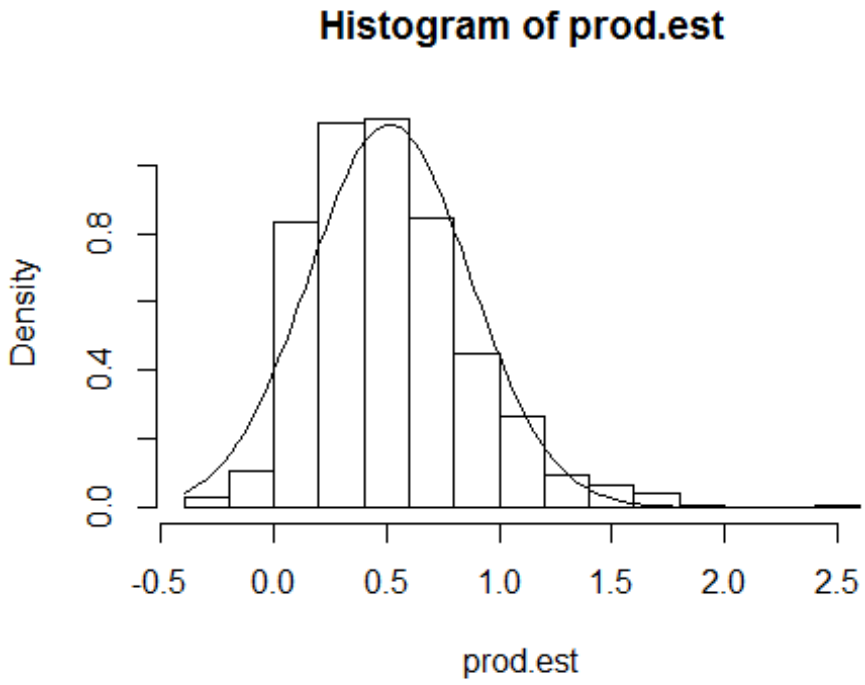
```
## [1] 0.147
```

```r
# under the null, this should be close to 0.05
```

Power is much lower for this small sample.

We can look at how close the sampling distribution is to normality again too.

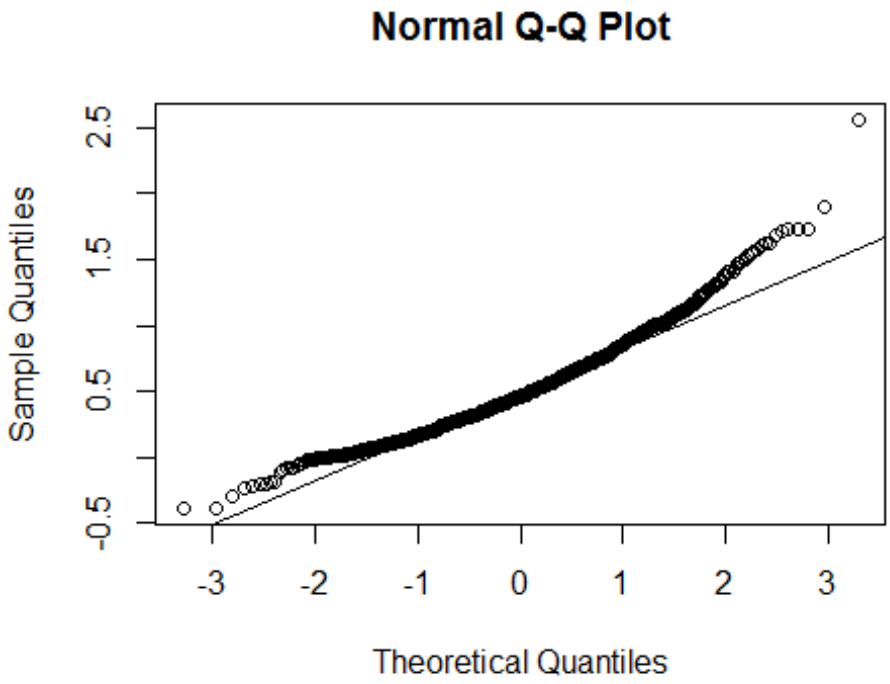
```r
# look at distribution of estimates
hist(prod.est,freq=FALSE)
# compared with a normal curve
curve(dnorm(x,mean=mean(prod.est),sd=sd(prod.est)),add=TRUE)
```




```r
# this test is not strictly correct since it involves estimated parameters
ks.test(prod.est,pnorm,mean=mean(prod.est),sd=sd(prod.est))
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  prod.est
## D = 0.062105, p-value = 0.000893
## alternative hypothesis: two-sided
```

```
# ... this test should be more accurate
LillieTest(prod.est)

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  prod.est
## D = 0.062105, p-value = 9.96e-10

# another way to look at normality
qqnorm(prod.est)
# these points should fall on this line if the sampling distribution is
normal
qqline(prod.est)
```




Normality approximation seems questionable again too.