Monte Carlo Simulation

AECN 896-002

Outline

Monte Carlo Simulation: Introduction

Monte Carlo Simulation

A way to test econometric theories via simulation

How is it used in econometrics?

- confirm ecoometric theory numerically
  - OLS estimators are unbiased if $E[u|x] = 0$ along with other conditions (theory)
  - I know the above theory is right, but let's check if it is true numerically
- You kind of sense that something in your data may cause problems, but there is no proven econometric theory about what's gonna happen (I used MC simulation for this purpose a lot)
- assist students in understanding econometric theories by providing actual numbers instead of a series of Greek letters

Question

Suppose you are interested in checking what happens to OLS estimators if $E[u|x] = 0$ (the error term and x are not correlated) is violated.

Can you use the real data to do this?

Key part of MC simulation

You generate data (you have control over how data are generated)

- You know the true parameter unlike the real data generating process
- You can change only the part that you want to change about data generating process and econometric methods with everything else fixed

Generating data

Pseudo random number generators

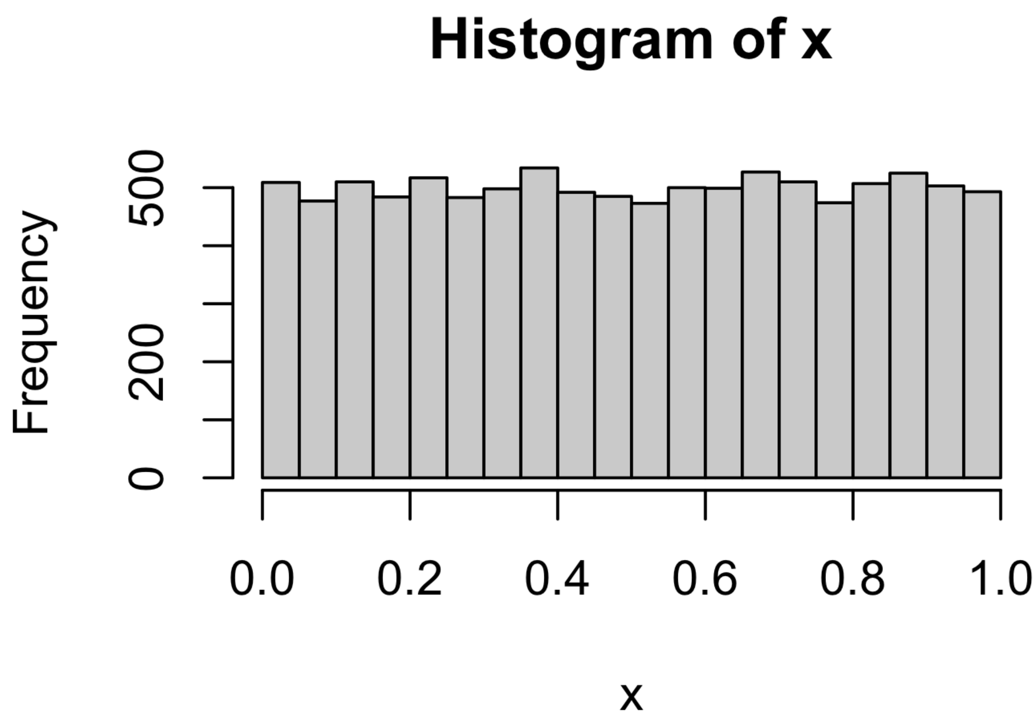
Algorithms for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers

Examples in R: Uniform Distribution

```r
runif(5) # default is min=0 and max=1
```

```
## [1] 0.7276097 0.5903365 0.6655839 0.9031736 0.5203449
```

```r
x <- runif(10000)
hist(x)
```



Pseudo random number generator

- Pseudo random number generators are not really random number generators
- What numbers you will get are pre-determined
- What numbers you will get can be determined by setting a seed

An example

```
set.seed(2387438)
runif(5)
```

```
## [1] 0.0474233 0.7116970 0.4066674 0.2422949 0.3567480
```

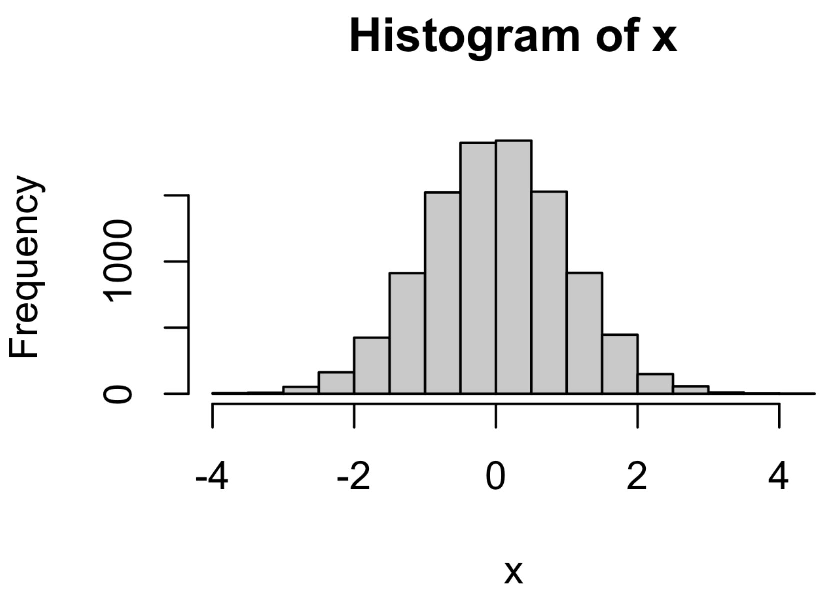
Question

What benefits does setting a seed have?

Examples in R: Normal Distribution

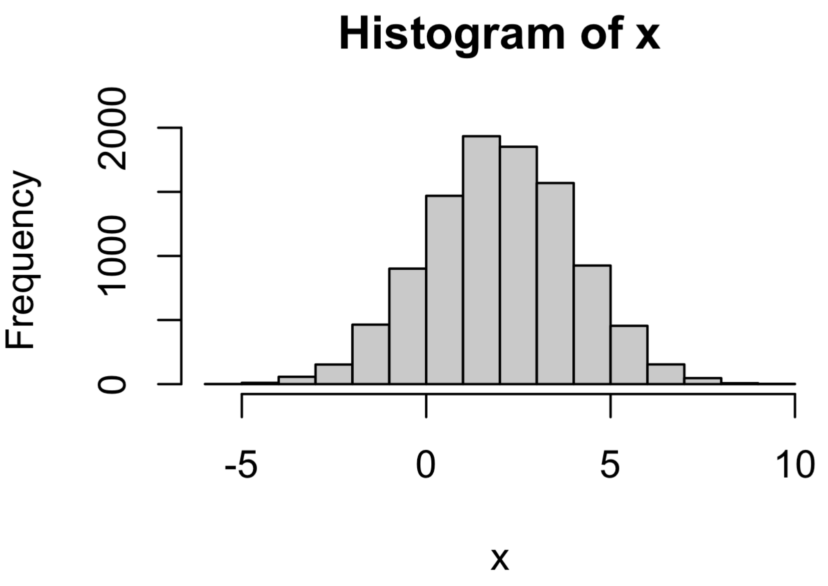
$x \sim N(0, 1)$

```r
# default is mean = 0,sd = 1
x <- rnorm(10000)
hist(x)
```



$x \sim N(2, 2)$

```r
# mean = 2, sd = 2
x <- rnorm(10000, mean = 2, sd = 2)
hist(x)
```



Other distributions

- Beta
- Chi-square
- F
- Logistic
- Log-normal
- many others

d, p, q, r

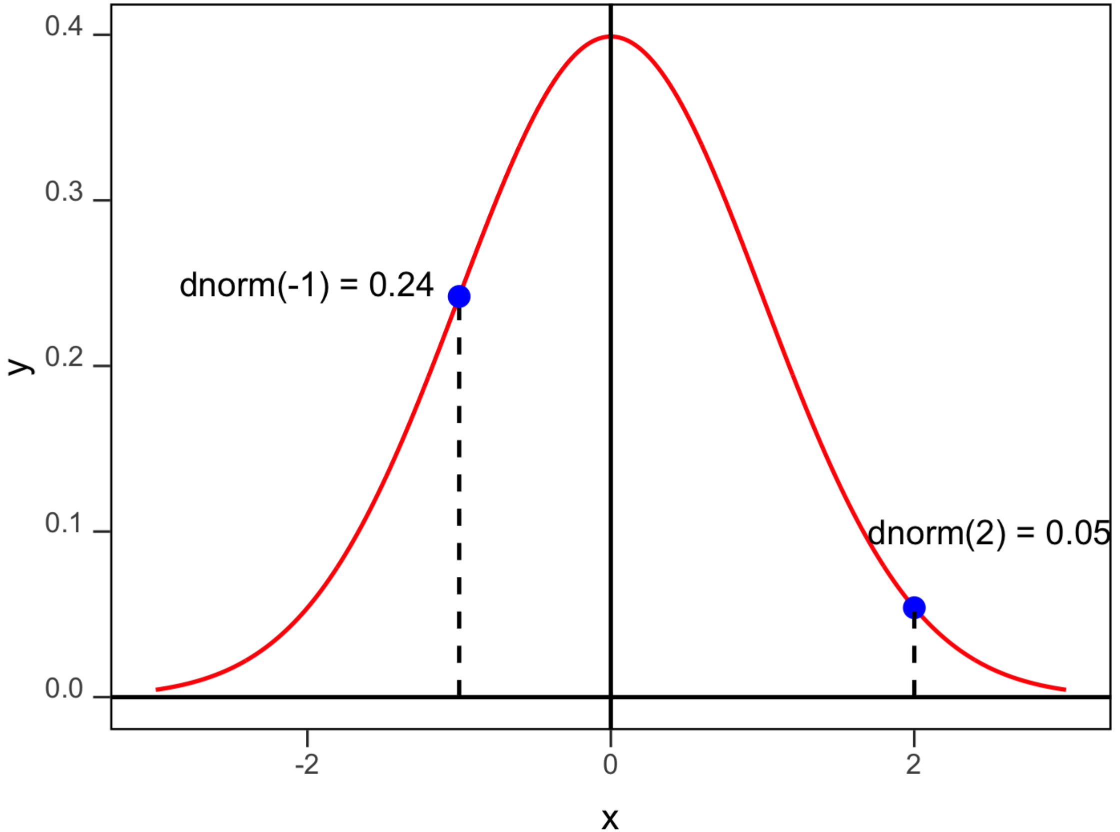
For each distribution, you have four different kinds of functions:

- `dnorm`: density function
- `pnorm`: distribution function
- `qnorm`: quantile function
- `rnorm`: random draw

dnorm

`dnorm(x)` gives you the height of the density function at x.

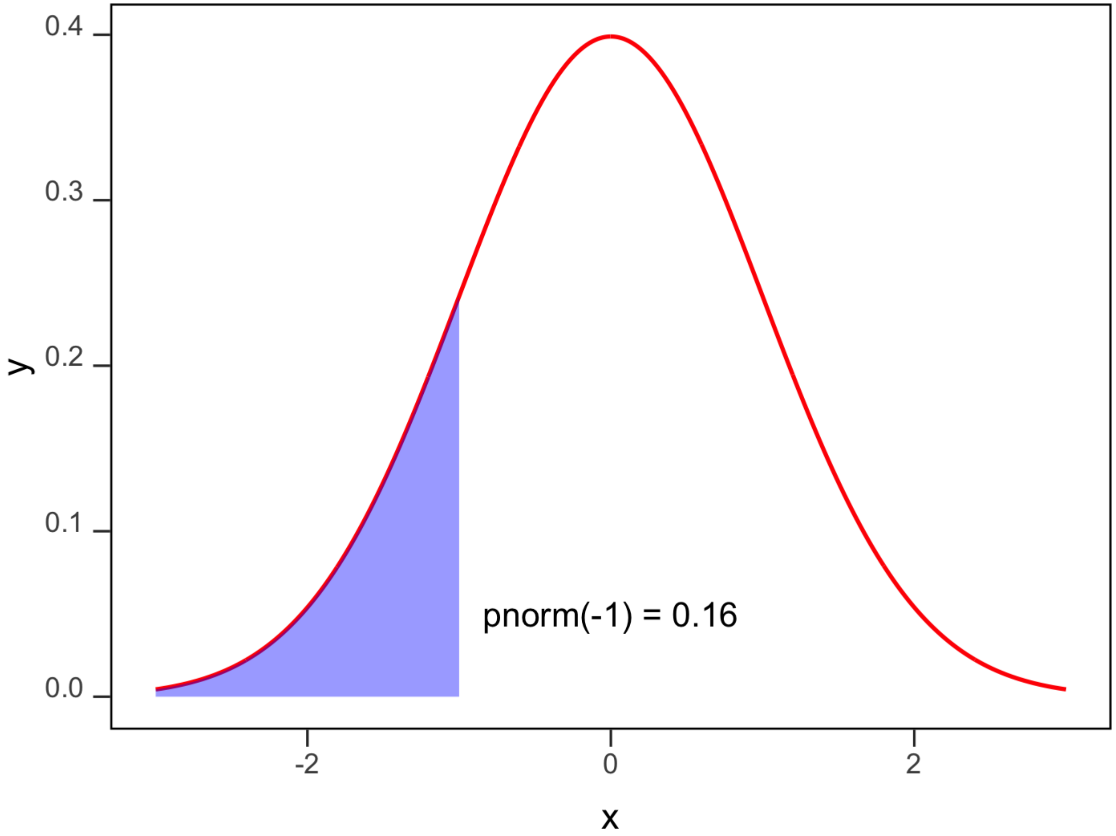
`dnorm(-1)` and `dnorm(2)`



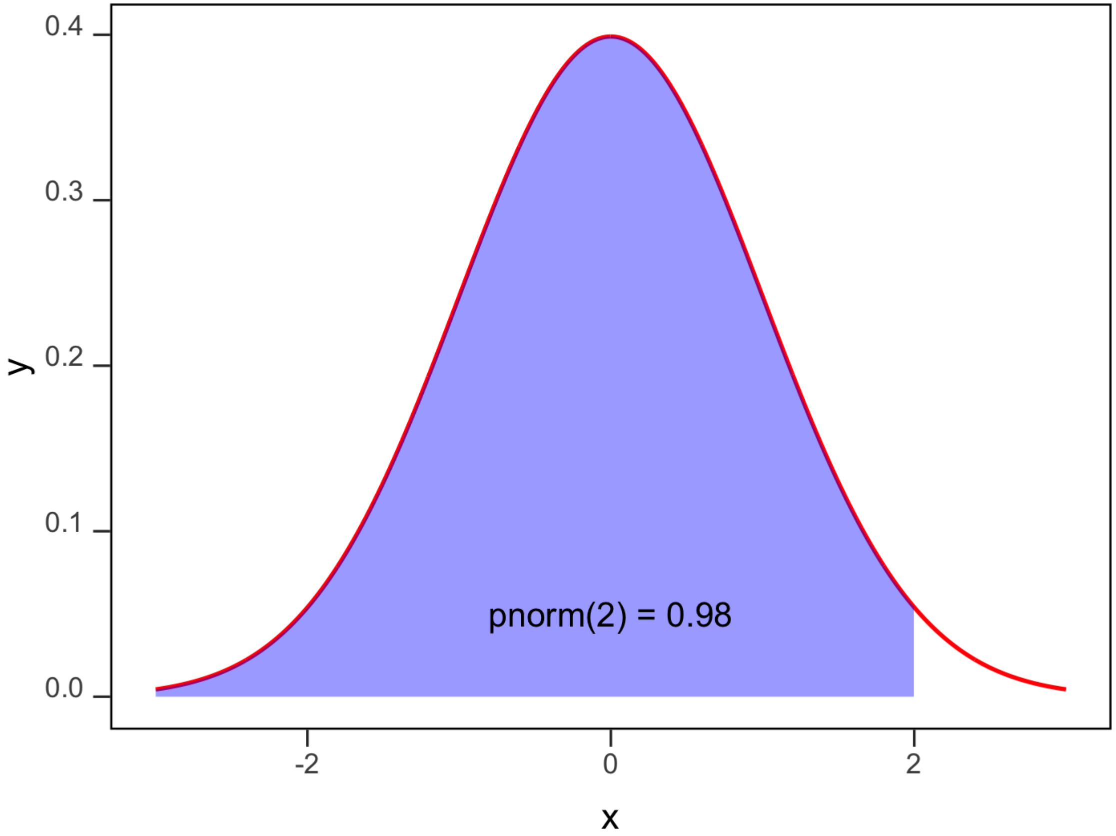
pnorm

`pnorm(x)` gives you the probability that a single random draw is less than x.

`pnorm(-1)`



`pnorm(2)`



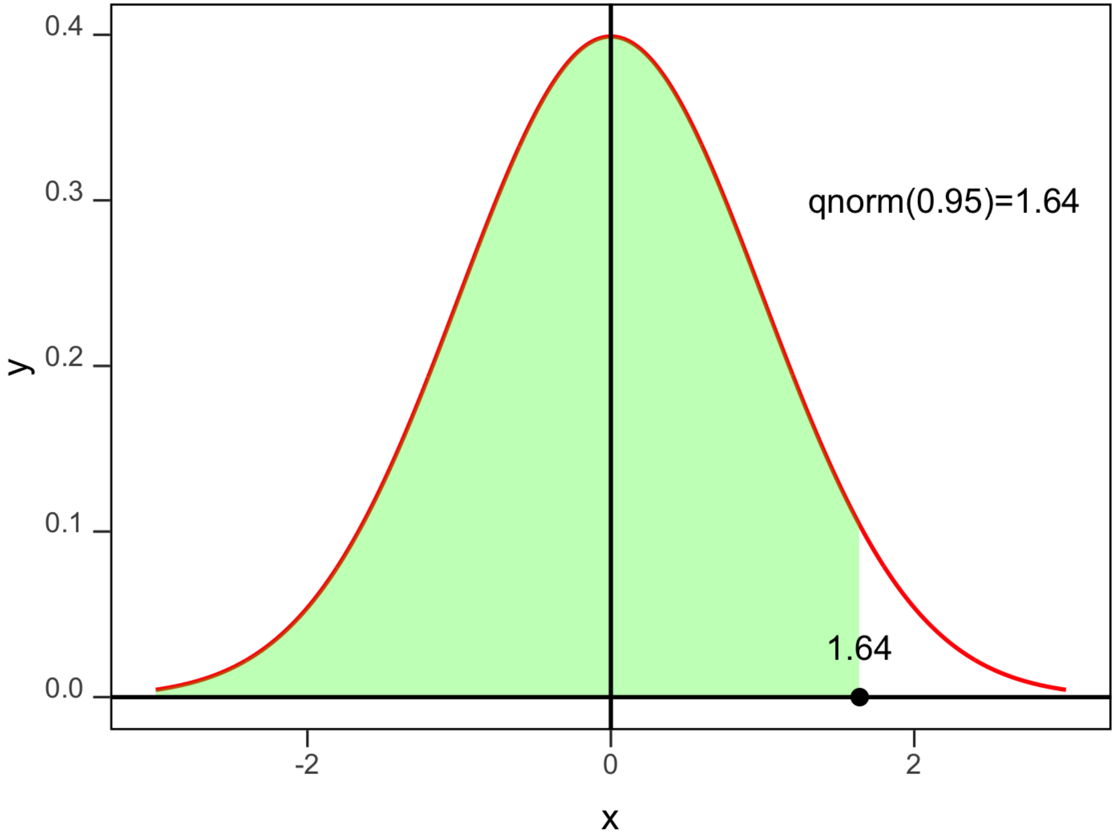
Practice

What is the probability that a single random draw from a Normal distribution with `mean = 1` and `sd = 2` is less than 1?

`qnorm(x)`, where $0 < x < 1$, gives you a number π, where the probability of observing a number from a single random draw is less than π with probability of x.

We call the output of `qnorm(x)`, x quantile of the standard Normal distribution (because the default is `mean = 0` and `sd = 1` for `rnorm()`).

`qnorm(0.95)`



Practice

What is the 88% quantile of Normal distribution with `mean = 0` and `sd = 9`?

Monte Carlo Simulation: Introduction

Monte Carlo Simulation: Steps

- specify the data generating process
- generate data based on the data generating process
- get an estimate based on the generated data (e.g. OLS, mean)
- repeat the above steps many many times
- compare your estimates with the true parameter

Question

Why do the steps $1-3$ many many times?

Monte Carlo Simulation: Example 1

Is sample mean really an unbiased estimator of the expected value?

That is, is $E[\frac{1}{n}\sum_{i=1}^{n} x_i] = E[x]$, where x_i is an independent random draw from the same distribution,

Sample Mean: Steps 1-3

```r
#--- steps 1 and 2:  ---#
# specify the data generating process and generate data
x <- runif(100) # Here, E[x]=0.5

#--- step 3 ---#
# calculate sample mean
mean_x <- mean(x)
mean_x
```

```
## [1] 0.507078
```

Sample Mean: Step 4

- repeat the above steps many times
- We use a loop to do the same (similar) thing over and over again

Loop: for loop

```r
#--- the number of iterations ---#
B <- 1000

#--- repeat steps 1-3 B times ---#
for (i in 1:B) {
  print(i) # print i
}
```

Verbally

For each of i in $1 : B$ $(1, 2, \dots, 1000)$, do `print(i)`.

- `i` takes the value of 1, and then `print(1)`
- `i` takes the value of 2, and then `print(2)`
- ...
- `i` takes the value of 999, and then `print(999)`
- `i` takes the value of 1000, and then `print(1000)`

Step 4

```r
#--- the number of iterations ---#
B <- 1000

#--- create a storage that stores estimates ---#
estimate_storage_mean <- rep(0, B)

#--- repeat steps 1-3 B times ---#
for (i in 1:B) {
  #--- steps 1 and 2:  ---#
  # specify the data generating process and generate data
  x <- runif(100) # Here, E[x]=0.5

  #--- step 3 ---#
  # calculate sample mean
  mean_x <- mean(x)
  estimate_storage_mean[i] <- mean_x
}
```

Compare your estimates with the true parameter

```
mean(estimate_storage_mean)
```

```
## [1] 0.500199
```

```
hist(estimate_storage_mean)
```

Monte Carlo Simulation: Example 2

Question

What happens to β_1 if $E[u|x] \neq 0$ when estimating $y = \beta_0 + \beta_1 x + u$?

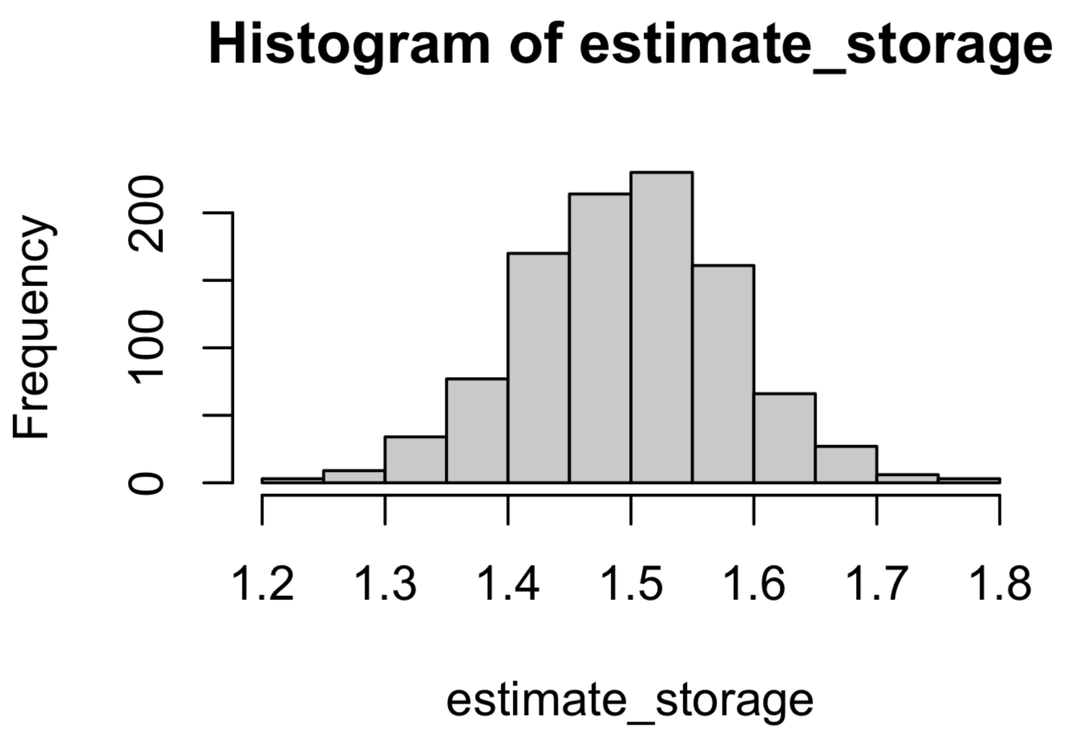
```r
#--- load the fixest pacakge for feols() ---#
library(fixest)

#--- Preparation ---#
B <- 1000 # the number of iterations
N <- 100 # sample size
estimate_storage <- rep(0, B) # estimates storage

#--- repeat steps 1-3 B times ---#
for (i in 1:B) {
  #--- steps 1 and 2:  ---#
  mu <- rnorm(N) # the common term shared by both x and u
  x <- rnorm(N) + mu # independent variable
  u <- rnorm(N) + mu # error
  y <- 1 + x + u # dependent variable
  data <- data.frame(y = y, x = x)

  #--- OLS ---#
  reg <- feols(y ~ x, data = data) # OLS
  estimate_storage[i] <- reg$coefficient["x"]
}
```

```r
hist(estimate_storage)
```



Examle 3: Variance of OLS Estimators

Model

$$y = \beta_0 + \beta_1 x + u$$

- $x \sim N(0, 1)$
- $u \sim N(0, 1)$
- $E[u|x] = 0$

Variance of the OLS estimator

True Variance of $\hat{\beta}_1$: $V(\hat{\beta}_1) = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{\sigma^2}{SST_X}$

Its estimator: $\widehat{V(\hat{\beta}_1)} = \frac{\hat{\sigma}^2}{SST_X} = \frac{\sum_{i=1}^{n} \hat{u}_i^2}{n-2} \times \frac{1}{SST_X}$

Question

Does the estimator really work? (Is it unbiased?)

```r
set.seed(903478)

#--- Preparation ---#
B <- 10000 # the number of iterations
N <- 100 # sample size
beta_storage <- rep(0, B) # estimates storage for beta
V_beta_storage <- rep(0, B) # estimates storage for V(beta)
x <- rnorm(N) # x values are the same for every iteration
SST_X <- sum((x - mean(x))^2)

#--- repeat steps 1-3 B times ---#
for (i in 1:B) {
  #--- steps 1 and 2:  ---#
  u <- 2 * rnorm(N) # error
  y <- 1 + x + u # dependent variable
  data <- data.frame(y = y, x = x)

  #--- OLS ---#
  reg <- feols(y ~ x, data = data) # OLS
  beta_storage[i] <- reg$coefficient["x"]
  #* store estimated variance of beta_1_hat
  V_beta_storage[i] <- vcov(reg)["x", "x"]
}
```

True Variance

- $SST_X = 112.07$
- $\sigma^2 = 4$

$$V(\hat{\beta}) = 4/112.07 = 0.0357$$

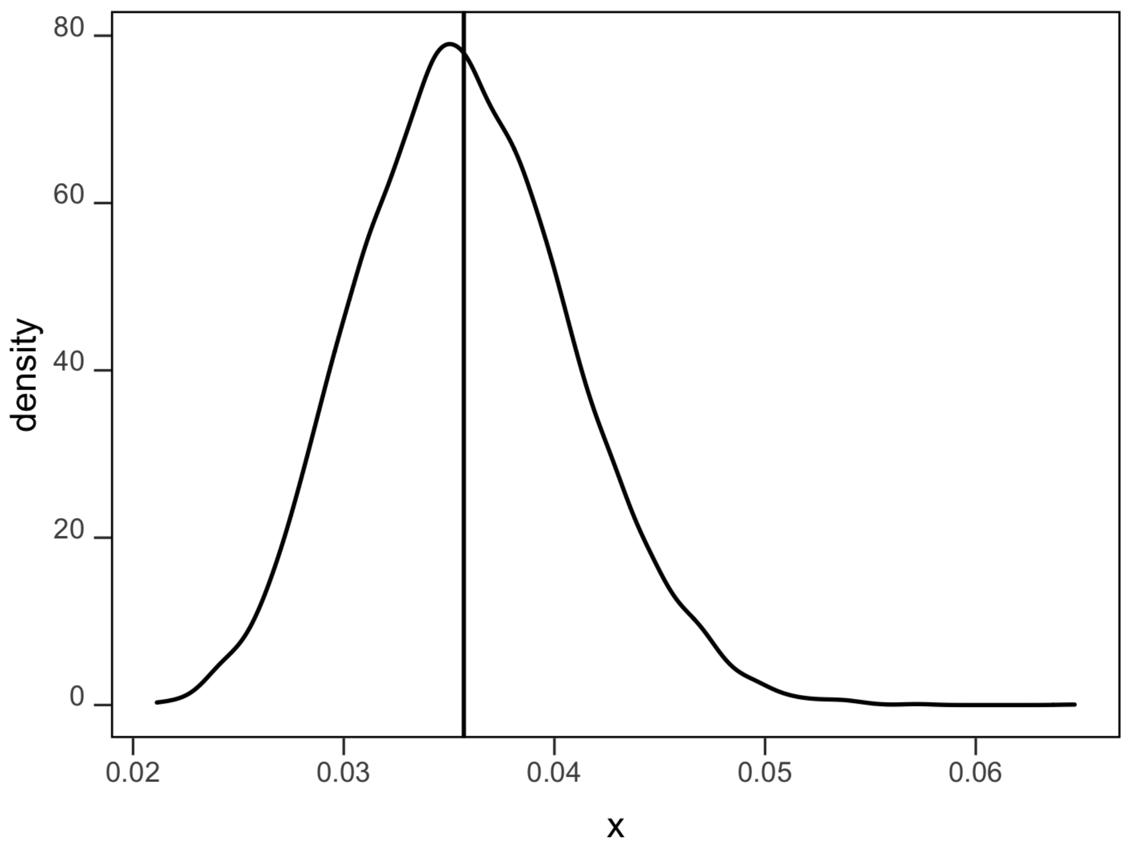
Check

Your Estimates of Variance of $\hat{\beta}_1$?

```r
# === mean ===#
mean(V_beta_storage)
```

```
## [1] 0.03579118
```

```r
ggplot(data = data.frame(x = V_beta_storage)) +
  geom_density(aes(x = x)) +
  geom_vline(xintercept = round(4 / SST_X, digits = 4))
```



Exercise

Problem

Using MC simulations, find out how the variation in x affects the OLS estimators

Model setup

$$y = \beta_0 + \beta_1 x_1 + u$$
$$y = \beta_0 + \beta_1 x_2 + u$$

- $x_1 \sim N(0,1)$ and $x_2 \sim N(0,9)$
- $u \sim N(0,1)$
- $E[u_1|x] = 0$ and $E[u_2|x] = 0$

Solution

```
#--- Preparation ---#
B <- 1000 # the number of iterations
N <- 100 # sample size
estimate_storage <- matrix(0, B, 2) # estimates storage

for (i in 1:B) {
  #--- generate data ---#
  x_1 <- rnorm(N, sd = 1) # indep var 1
  x_2 <- rnorm(N, sd = 3) # indep var 2
  u <- rnorm(N) # error
  y_1 <- 1 + x_1 + u # dependent variable 1
  y_2 <- 1 + x_2 + u # dependent variable 2
  data <- data.table(y_1 = y_1, y_2 = y_2, x_1 = x_1, x_2 = x_2)

  #--- OLS ---#
  reg_1 <- feols(y_1 ~ x_1, data = data) # OLS
  reg_2 <- feols(y_2 ~ x_2, data = data) # OLS

  #--- store coef estimates ---#
  estimate_storage[i, 1] <- reg_1$coefficient["x_1"] # equation 1
  estimate_storage[i, 2] <- reg_2$coefficient["x_2"] # equation 2
}
```

```
#--- assign new names ---#
beta_1s <- estimate_storage[, 1]
beta_2s <- estimate_storage[, 2]

#--- mean ---#
mean(beta_1s)
```

Visualization

```
plot_data_1 <- data.table(x = beta_1s, type = "Equation 1")
plot_data_2 <- data.table(x = beta_2s, type = "Equation 2")
plot_data <- rbind(plot_data_1, plot_data_2)
ggplot(data = plot_data) +
  geom_density(aes(x = x, fill = type), alpha = 0.5) +
  scale_fill_discrete(name = "") +
  xlab("Coefficient Estimate") +
  theme(
    legend.position = "bottom"
  )
```