

数据生成过程

黄嘉平

2025-03-03

1 什么是数据生成过程

用来生成仿真数据的模型称为数据生成过程 (data generating process)。在计量经济学中，我们最熟悉的数据生成过程是线性回归模型，即

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} + u_i$$

其中误差项 u_i 是随机变量，解释变量 X_{1i}, \dots, X_{ki} 通常也是随机变量，系数 $\beta_0, \beta_1, \dots, \beta_k$ 是未知参数。回归的目的是利用样本中提供的 $(y_i, x_{1i}, \dots, x_{ki})$ 数据拟合出适合的参数估计值 $\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_k$ 并以此进行统计推断。但是在生成仿真数据时，我们要做相反的操作，即指定一组参数值，然后确定解释变量和误差项所服从的概率分布，再利用伪随机数生成器生成一组样本。

生成仿真数据的主要目的是验证计量方法。当我们假设了一个模型（数据生成过程）时，它就是“真实模型”。在生成了仿真数据后，我们假装不了解真实模型，而是用计量方法进行拟合和推断，并将结果和真实模型中进行对比，以达到验证计量方法效果的目的。

2 一个例子

下面我们考虑一个简单的非线性回归模型

$$Y_i = 10 + 5X_i - 0.1X_i^2 + u_i, \quad u_i \sim N(0, 1), \quad X_i \sim \text{Unif}(0, 10)$$

并依此模型生成一组样本量为 200 的随机样本。

首先不要忘记调用 tidyverse 程序包。

```
library(tidyverse)
```

生成数据

```
# 设定样本量
```

```
n <- 200
```

```
# 生成  $X$  和  $u$ 
```

```
x <- runif(n, 0, 10)
u <- rnorm(n)

# 生成 Y
y <- 10 + 5 * x - 0.1 * x^2 + u

# 保存数据
sim_data <- tibble(y, x)
```

检查数据是否正确生成

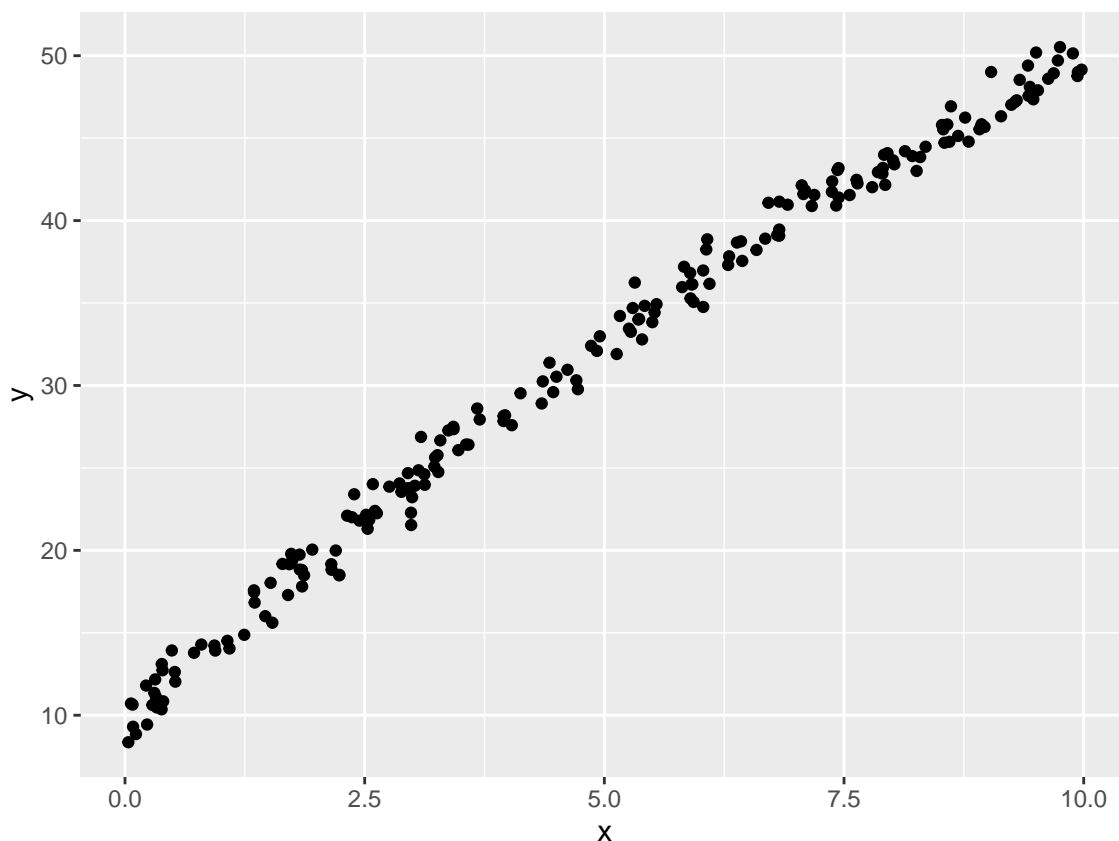
```
# 显示 sim_data 的内容
sim_data
```

```
## # A tibble: 200 x 2
##       y       x
##   <dbl> <dbl>
## 1 45.5  8.91
## 2 48.5  9.33
## 3 34.8  6.03
## 4 23.9  3.02
## 5 19.2  1.73
## 6 36.1  5.92
## 7 23.8  2.95
## 8  8.36 0.0350
## 9 41.1  6.71
## 10 17.3  1.70
## # i 190 more rows
```

```
# 用 glimpse() 函数了解 sim_data 中的变量
glimpse(sim_data)
```

```
## Rows: 200
## Columns: 2
## $ y <dbl> 45.539477, 48.534336, 34.763633, 23.916716, 19.236902, 36.133699, 23~
## $ x <dbl> 8.9128141, 9.3325130, 6.0315952, 3.0230846, 1.7328862, 5.9175867, 2.~
```

```
# 绘制散点图
ggplot(sim_data, aes(x, y)) +
  geom_point()
```

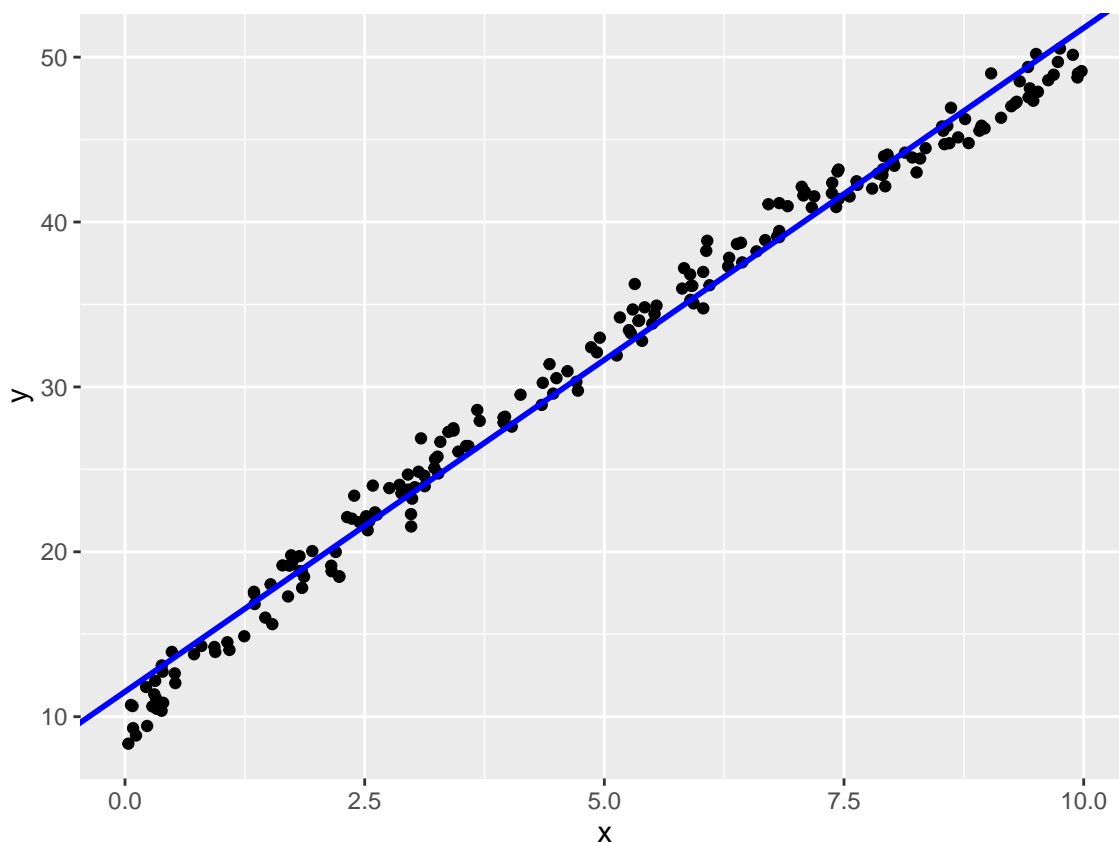


接下来我们用线性回归模型进行拟合（忽略二次项）

```
linearfit <- lm(y ~ x, data = sim_data)
summary(linearfit) # 查看拟合结果
```

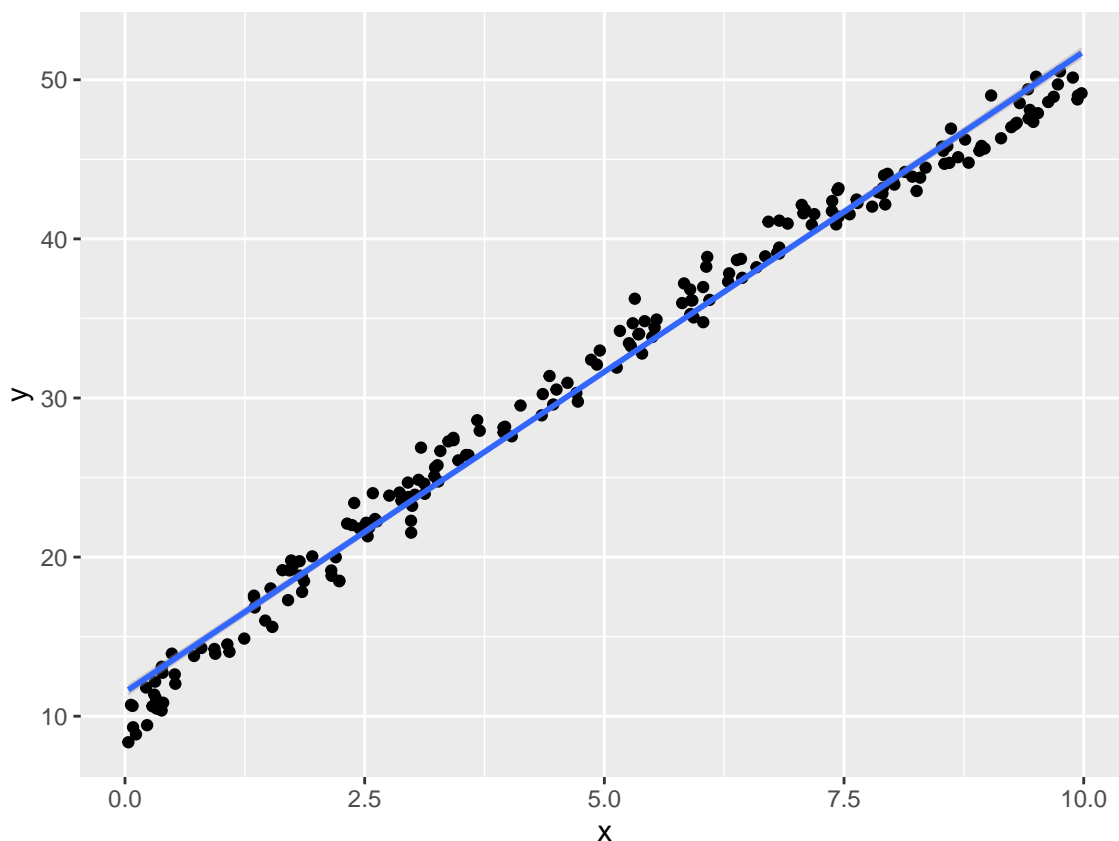
```
##
## Call:
## lm(formula = y ~ x, data = sim_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.2936 -1.0459  0.0995  0.9275  3.3197
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  11.51520    0.18361   62.71  <2e-16 ***
## x             4.02533    0.03221  124.95  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.353 on 198 degrees of freedom
## Multiple R-squared:  0.9875, Adjusted R-squared:  0.9874
## F-statistic: 1.561e+04 on 1 and 198 DF, p-value: < 2.2e-16
```

```
# 在散点图中添加拟合曲线
ggplot(sim_data, aes(x, y)) +
  geom_point() +
  geom_abline(
    intercept = coef(linearfit)[1], slope = coef(linearfit)[2],
    color = "blue", linewidth = 1
  )
```



```
# 或者利用 geom_smooth() 添加相同的拟合曲线
ggplot(sim_data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm")
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



让我们再尝试加入二次项

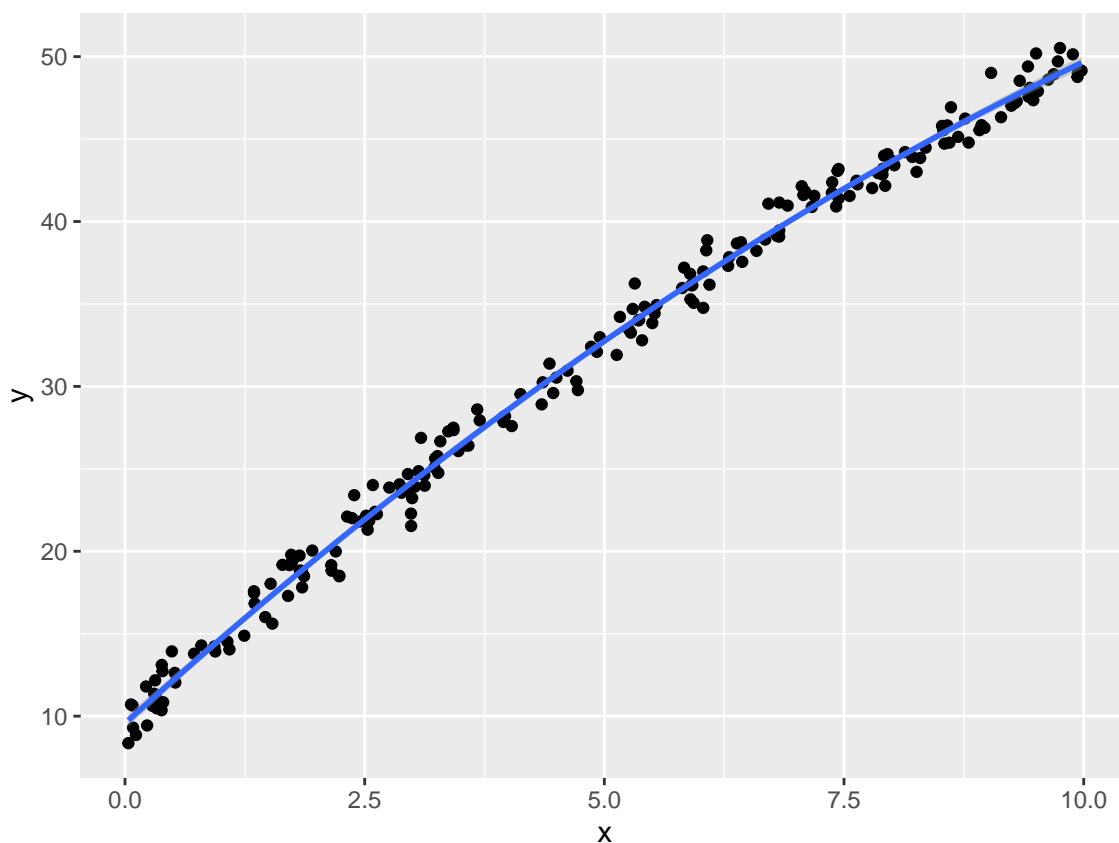
```
linearfit2 <- lm(y ~ x + I(x^2), data = sim_data) # 注意如何添加二次项
summary(linearfit2) # 查看拟合结果
```

```
##
## Call:
## lm(formula = y ~ x + I(x^2), data = sim_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.6163 -0.6723 -0.1496  0.6864  2.2653
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.563182   0.191013   50.07  <2e-16 ***
## x            5.259514   0.091167   57.69  <2e-16 ***
## I(x^2)       -0.124716   0.008918  -13.98  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9611 on 197 degrees of freedom
## Multiple R-squared:  0.9937, Adjusted R-squared:  0.9937
```

```
## F-statistic: 1.558e+04 on 2 and 197 DF, p-value: < 2.2e-16
```

```
# 这次直接利用 geom_smooth() 添加拟合曲线
```

```
ggplot(sim_data, aes(x, y)) +  
  geom_point() +  
  geom_smooth(method = "lm", formula = "y ~ x + I(x^2)")
```



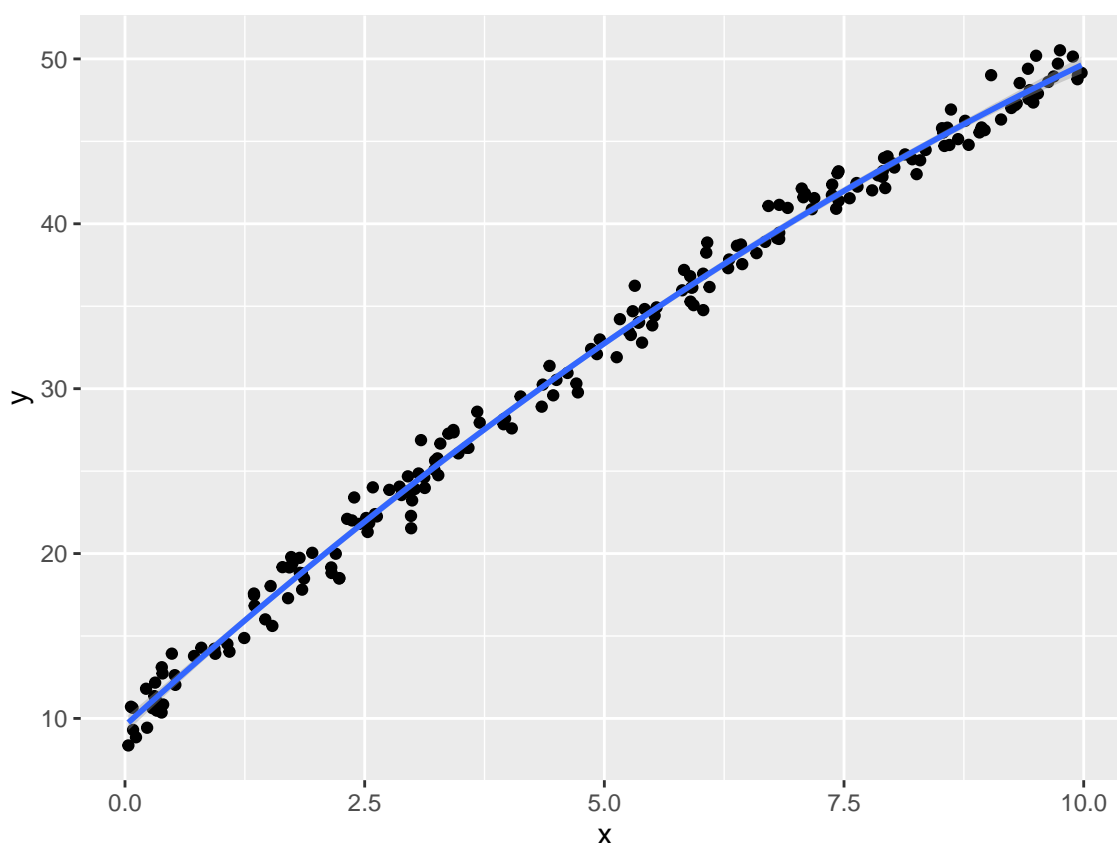
最后尝试加入三次项进行拟合

```
linearfit3 <- lm(y ~ x + I(x^2) + I(x^3), data = sim_data)  
summary(linearfit3)
```

```
##  
## Call:  
## lm(formula = y ~ x + I(x^2) + I(x^3), data = sim_data)  
##  
## Residuals:  
##      Min       1Q   Median       3Q      Max   
## -2.6160 -0.6722 -0.1498  0.6867  2.2656   
##  
## Coefficients:  
##              Estimate Std. Error t value Pr(>|t|)      
## (Intercept)  9.564e+00  2.428e-01  39.382  <2e-16 ***  
## x            5.259e+00  2.135e-01  24.630  <2e-16 ***  
## I(x^2)       -1.245e-01  5.048e-02  -2.467   0.0145 *  
```

```
## I(x^3)      -1.343e-05  3.362e-03  -0.004   0.9968
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9635 on 196 degrees of freedom
## Multiple R-squared:  0.9937, Adjusted R-squared:  0.9936
## F-statistic: 1.033e+04 on 3 and 196 DF,  p-value: < 2.2e-16
```

```
ggplot(sim_data, aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", formula = "y ~ x + I(x^2) + I(x^3)")
```



3 函数化

如果需要反复生成多个样本或者调整参数，就可以将数据生成的操作函数化。R 中函数的基本格式是

```
function_name <- function(arguments) {
  body
}
```

针对上一节中的数据生成过程，如果我们固定 X_i 和 u_i 的分布，但想要调整回归函数的系数和样本量，则函数化后的程序如下

```
sim_lm <- function(parameters = c(10, 5, -0.1), n = 200) {
  # 生成 X 和 u
  x <- runif(n, 0, 10)
  u <- rnorm(n)

  # 生成 Y
  y <- parameters[1] + parameters[2] * x + parameters[3] * x^2 + u

  # 保存数据
  return(tibble(y, x))
}
```

上面这一段是定义函数 `sim_lm()`。它有两个参数，第一个参数是 `parameters`，初始值设定为 `c(10, 5, -1)`。第二个参数为样本量 `n`，初始值设定为 200。

调用函数

```
sim_data2 <- sim_lm() # 不更改参数的初始值
lm(y ~ x + I(x^2), data = sim_data2) |> summary()

##
## Call:
## lm(formula = y ~ x + I(x^2), data = sim_data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.3857 -0.7270 -0.0458  0.7107  3.3619
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.12623    0.21660  46.751  <2e-16 ***
## x           4.99589    0.10265  48.671  <2e-16 ***
## I(x^2)      -0.10013    0.01008  -9.935  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.104 on 197 degrees of freedom
## Multiple R-squared:  0.9918, Adjusted R-squared:  0.9917
## F-statistic: 1.184e+04 on 2 and 197 DF, p-value: < 2.2e-16

sim_data3 <- sim_lm(parameters = c(1, 2, 1), n = 500) # 指定新参数
glimpse(sim_data3)

## Rows: 500
## Columns: 2
```



```
## $ y <dbl> 30.295259, 6.082521, 24.488183, 34.469157, 5.167573, 22.312440, 3.33~  
## $ x <dbl> 4.4066031, 1.4383132, 4.0359105, 4.6259790, 0.8396141, 3.6760356, 0.~  
ggplot(sim_data3, aes(x, y)) +  
  geom_point()
```

