专题二：数据生成过程

黄嘉平

2026.3.16

1. 什么是数据生成过程

通常我们不知道观测数据是遵循什么机制产生的。在学术研究中，人们通常假设一种能够产生数据的理论模型，也就是所谓的**数据生成过程**（data generating process, DGP）。根据给定的数据生成过程就可以生成多变量仿真数据。计量经济学中最常用的数据生成过程是线性回归模型，即

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

上式中误差项 u_i 是随机变量，解释变量 $X_{1i}, \ldots, X_{ki}$ 通常也是随机变量，系数 $\beta_0, \beta_1, \ldots, \beta_k$ 是未知参数。回归的目的是利用样本中包含的 $(y_i, x_{1i}, \ldots, x_{ki})$ 数据拟合出适合的参数估计值 $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_k$ 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的效果。当我们假设一个模型（数据生成过程）时，意味着它就是“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进行对比，以达到验证效果的目的。

2. 一个简单的例子

下面我们考虑一个简单的非线性回归模型

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

并依此模型生成一组样本量为 200 的随机样本。

1. 首先不要忘记调用 **tidyverse** 程序包。

```
library(tidyverse)
```

2. 生成数据

```
n <- 200 # 设定样本量

set.seed(111)
x <- runif(n, 0, 10) # 生成 X (服从 [0,10] 间的均匀分布)
u <- rnorm(n) # 生成 u （服从标准正态分布）

y <- 1 + 2 * x - 0.1 * x^2 + u # 生成 Y
```

```r
sim_data <- tibble(y, x) # 保存数据
```

3. 检查生成的数据是否正确

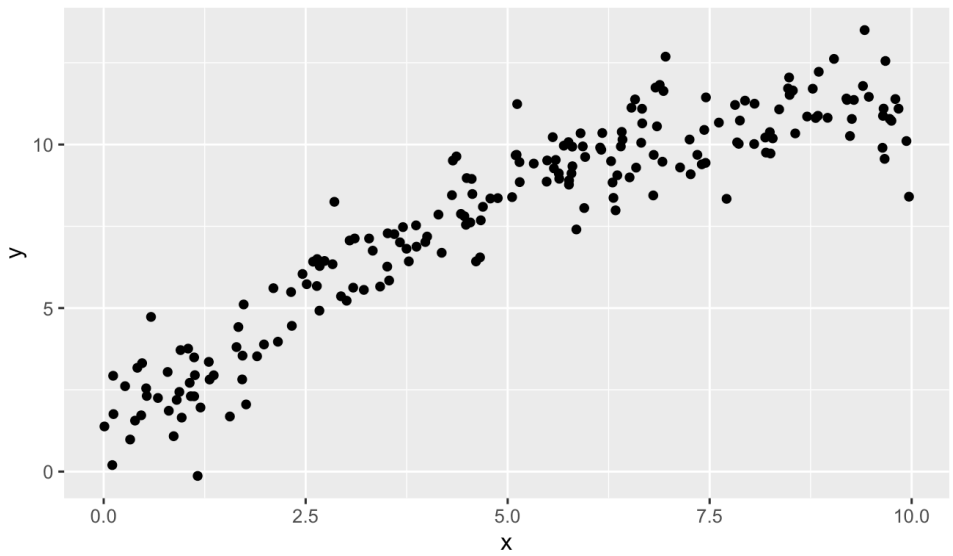
```r
sim_data # 显示 sim_data 的内容
```

```
# A tibble: 200 × 2
       y     x
   <dbl> <dbl>
 1 9.94  5.93
 2 9.09  7.26
 3 7.48  3.70
 4 8.85  5.15
 5 6.43  3.78
 6 6.69  4.18
 7 0.199 0.107
 8 9.42  5.32
 9 9.51  4.32
10 2.44  0.937
# i 190 more rows
# i Use print(n = …) to see more rows
```

```r
glimpse(sim_data) # 也可以用 glimpse() 函数了解 sim_data 中的变量
```

```
Rows: 200
Columns: 2
$ y <dbl> 9.9429774, 9.0915447, 7.4754089, 8.8518648, 6.4277879, 6.6900597…
$ x <dbl> 5.9298128, 7.2648112, 3.7042200, 5.1492383, 3.7766322, 4.1833733…
```

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

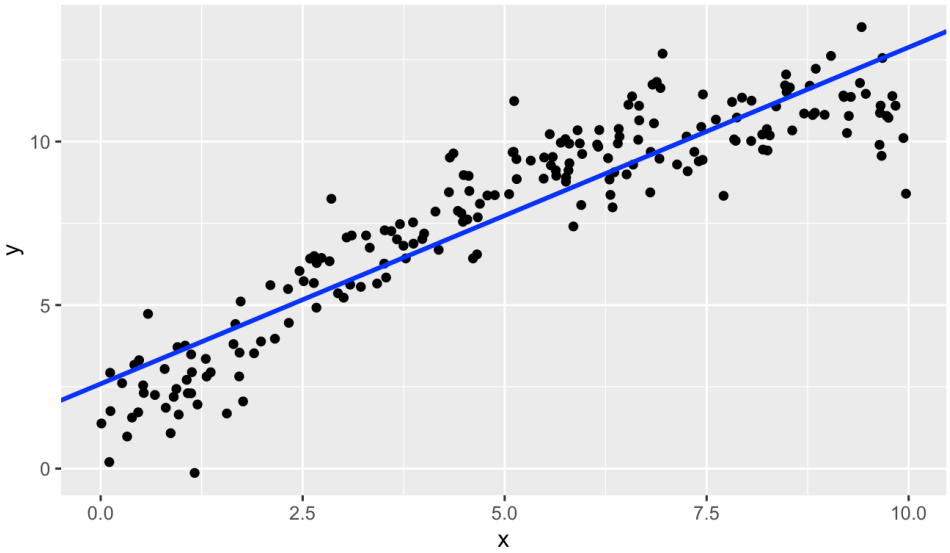


4. 接下来我们用线性回归模型进行拟合（忽略二次项）。从上图中可以看出，虽然数据生成过程是二次函数，但在数据取值范围内用线性函数近似的效果也不会太差。

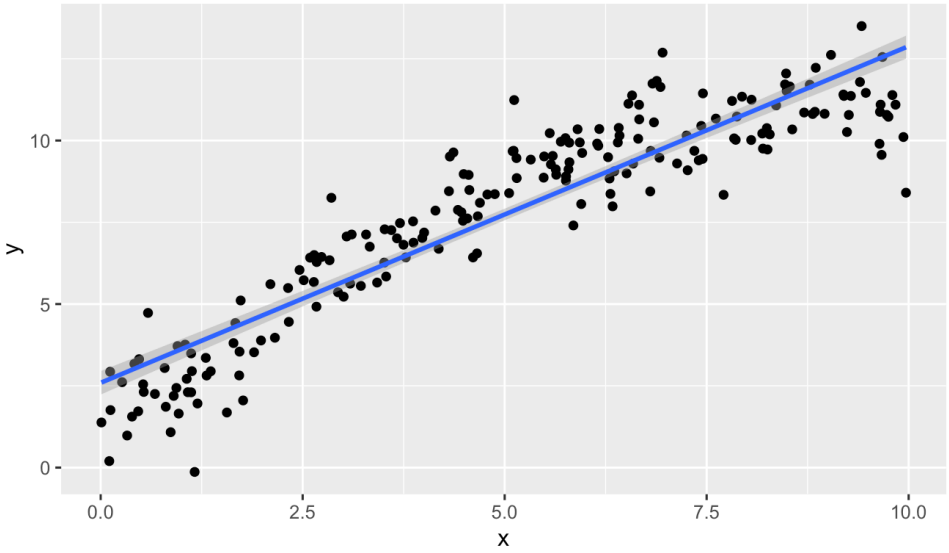
```r
linearfit <- lm(y ~ x, data = sim_data) # 线性回归拟合的命令是 lm()
summary(linearfit)     # 查看拟合结果
```

```
Call:
lm(formula = y ~ x, data = sim_data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.4466 -0.8387  0.1037  0.9157  3.3772

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.59245    0.18178   14.26   <2e-16 ***
x            1.02934    0.03131   32.88   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.267 on 198 degrees of freedom
Multiple R-squared:  0.8452,  Adjusted R-squared:  0.8444
F-statistic:  1081 on 1 and 198 DF,  p-value: < 2.2e-16
```

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




```r
ggplot(sim_data, aes(x, y)) + geom_point() +
  geom_smooth(method = "lm")
  # 利用 geom_smooth() 可以添加相同的拟合曲线，以及95%预测区间
```



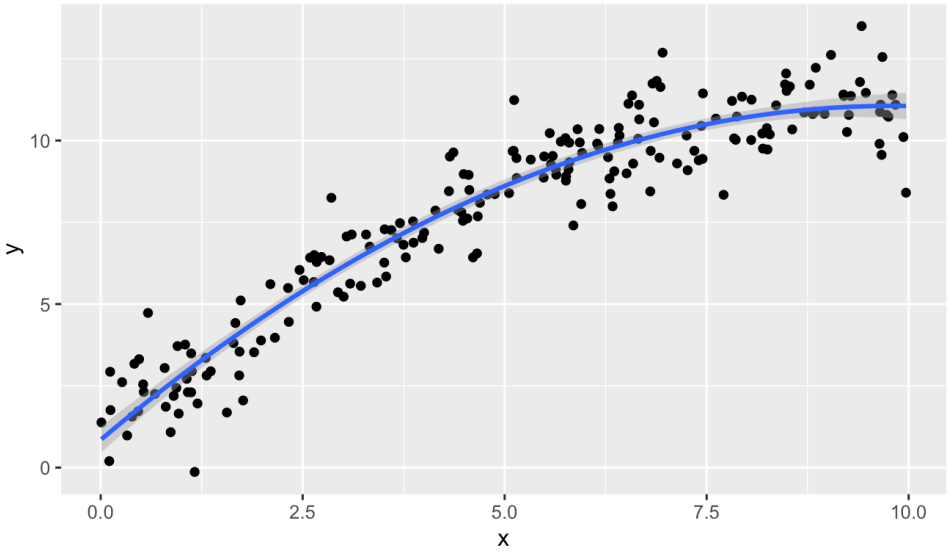
5. 现在再尝试加入二次项

```
linearfit2 <- lm(y ~ x + I(x^2), data = sim_data)
  # 添加二次项时需要用到 I() 函数，这是因为 ^2 既可以表达取平方，也可以表达解释变量间的交互关系，I() 函数的目的是表明我们要进行平方计算
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
-3.2607 -0.6289 -0.0247  0.6317  2.6973

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.847483   0.203070   4.173  4.5e-05 ***
x            2.084677   0.092473  22.544  < 2e-16 ***
I(x^2)      -0.106393   0.009003 -11.818  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9717 on 197 degrees of freedom
Multiple R-squared:  0.9094,  Adjusted R-squared:  0.9085
F-statistic: 988.6 on 2 and 197 DF,  p-value: < 2.2e-16
```

从结果可以看出二次项系数显著，各项系数和理论模型非常相似，拟合效果（R^2）也比线性函数好。

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



6. 在实证研究中通常还会尝试添加三次项或者将 X 进行对数变换，大家可以自行尝试，并和前面的结果进行比较。

3. U 形（倒 U 形）检验

在实证研究中经常会见到对两个变量间的 U 形（或倒 U 形）关系的检验，例如大家熟知的 Kuznets 曲线（收入水平和不平等度之间存在倒 U 形关系）。很多研究利用二次函数拟合后二次项系数是否显著（部分研究会计算拟合后的曲线顶点并观察它是否在数据区间内）来判断是否存在 U 形关系。但以 Lind & Mehlum (2010) 为代表的研究者认为该方法有可能把非线性但是单调的关系误判为 U 形关系。在确认回归函数为非线性且在数据区间内只存在一个极值的前提下，检验 U 形（或倒 U 形）关系需要检验回归函数在 X 的最大值和最小值附近的斜率是否同时显著不为零且符号相反。Lind & Mehlum (2010) 提出了 U 形检验的正确方法，该方法可以用 **utest** 程序包实现。

下面我们针对前文中生成的仿真数据进行 Lind-Mehlum U 形检验。

1. 首先需要安装 **utest** 包并调用。

```
install.packages("utest")
library(utest)
```

2. **utest** 包中包含两个命令，一个是计算并检验 X 最大值和最小值附近斜率的 **uslopes()**，另一个是计算整体检验统计量和 p-值 的 **utest()**。二者都需要利用线性回归（二次函数或对数函数）的拟合结果。前文中该结果保存在 **linearfit** 变量中，但直接调用会报错，需要先将非线性项（这里是二次项）单独保存后再进行拟合。

```
xsq <- x^2 # 必须先将二次项保存为一个变量
mod <- lm(y ~ x + xsq) # 二次函数的拟合
uslopes(mod, c("x","xsq")) # 检验 x 的最大值和最小值附近的斜率
```

```
 Slopes at the extremes of the interval
-------------------------------------------------
                |   Lower bound    Upper bound
-------------------------------------------------
Interval        |   0.009253        9.967167
Slope           |   2.082708       -0.036192
t-value         |   22.561483      -0.387881
P>|t|           |   0.000000        0.349261
-------------------------------------------------

Extreme point: 9.797080
```

上面的表格中，第一行为 X 的最小值和最大值，第二行为最小值和最大值附近的斜率估计值，第三行为单个斜率的 t-统计量，第四行为单个斜率的单边检验 p-值。我们看到在 X 最大值处，样本数据不支持斜率显著为负。表格下面一行给出了拟合后的二次函数顶点的位置，它非常靠近 X 的最大值。

```
utest(mod, c("x","xsq")) # 检验 U 或倒 U 形状
```

```
  Test of inverted U shape

data:  mod
t-value = 0.38788, p-value = 0.3493
alternative hypothesis: Monotone or U shape
```

这是检验 U 形关系的结果。p-值为 0.3493，无法拒绝不存在倒 U 形关系（由拟合系数可知）的原假设。这个结论更加符合我们从散点图中得到的直观判断，因为我们没有 $X > 10$ 时的数据，无法判断是否真的存在倒 U 形关系。

4. 函数化

如果需要反复生成多个样本或者调整参数，那么将生成数据的操作函数化可以极大提升代码的效率。R 中定义函数的基本格式是

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

其中需要自己设定的包括函数名称 **function_name**，函数的输入变量 **arguments**，以及函数进行的操作（即 **body** 部分）。针对前文中的数据生成过程，如果我们固定 X_i 和 u_i 的分布，但想要自由调整回归函数的系数和样本量，则函数化后的代码如下。

```
sim_lm <- function(parameters = c(1, 2, -0.1), n = 200) {

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

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

  return(tibble(y, x)) # 输出数据
}
```

这里函数名是 **sim_lm**。共有两个输入变量，一个是 **parameters**，这是一个三维向量，分别保存二次回归函数的三个系数，另一个是样本量 **n**。我们给这两个输入变量都赋予了预设值，这意味着在使用时如果不另外指定，它们的取值就是预设值。函数的操作分为生成 X 和 u，生成 Y，以 tibble 形式输出数据。如果一个函数需要输出数据，那么一定要用 **return()** 命令执行。

在执行定义函数的代码后，该函数就会被暂时记忆在内存中，直到退出当前的 R session 或者手动清理内存。执行后就可以调用该函数。

```
set.seed(111)
sim_data2 <- sim_lm()    # 不更改输入变量的预设值
```

通过函数化，仿真数据的生成就简化为一行命令，非常方便。

```
set.seed(111)
sim_data3 <- sim_lm(parameters = c(1, 0.1, 0.1), n = 500)
  # 将系数向量改为 (1, 0.1, 0.1)，样本量改为 500
```

这是更改回归系数和样本量后再次生成数据的代码，依旧只需要一行即可实现。在此基础上，我们可以根据需要反复生成样本数据，用于检验计量方法在不同条件下的表现。

参考文献

Lind, J. T., and Mehlum, H. (2010). With or without U? The appropriate test for a U shaped relationship. *Oxford Bulletin of Economics and Statistics*, 72(1): 109-18. doi:10.1111/j.1468-0084.2009.00569.x.