数据生成过程

黄嘉平

2025-03-20

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 $X_{1i}, \ldots, X_{ki}$ 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提供的 $(y_i, x_{1i}, \ldots, x_{ki})$ 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。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

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

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

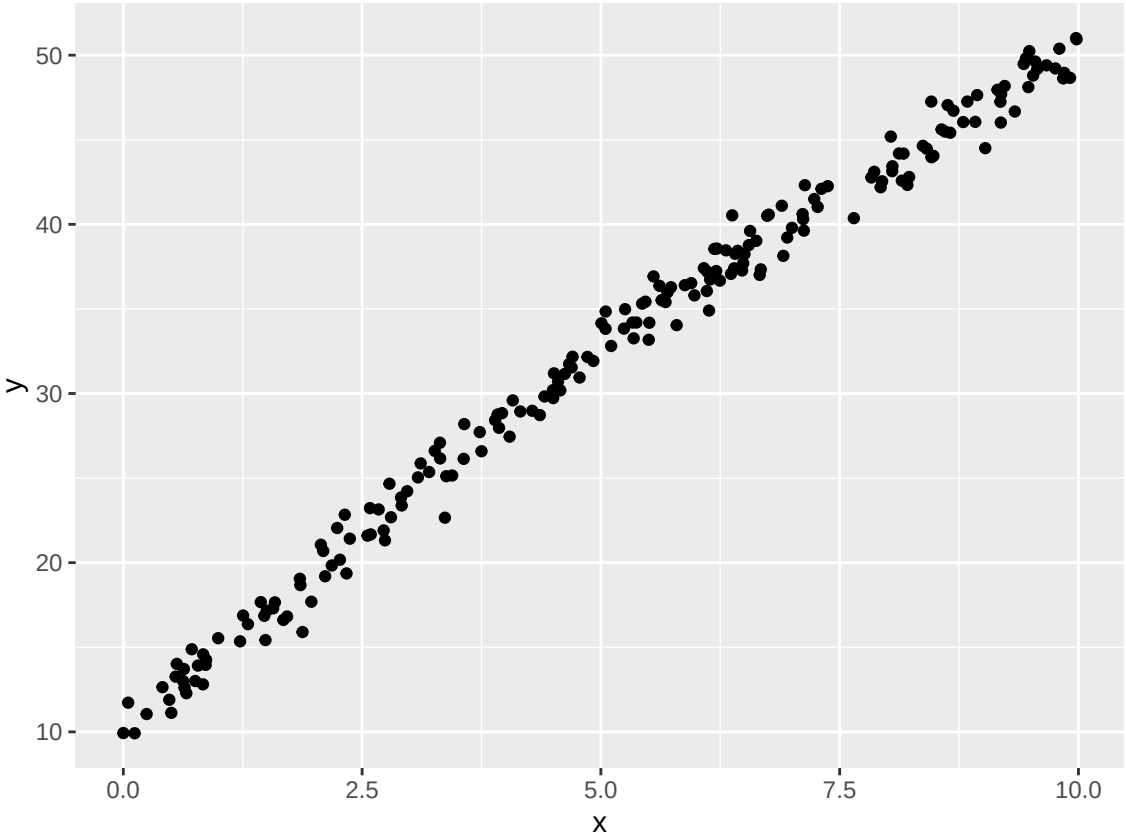
```
# 显示 sim_data 的内容
sim_data
```

```
## # A tibble: 200 x 2
##        y     x
##    <dbl> <dbl>
##  1  22.8  2.32
##  2  44.2  8.12
##  3  19.0  1.85
##  4  15.4  1.22
##  5  40.3  7.12
##  6  49.2  9.57
##  7  34.2  5.51
##  8  32.8  5.11
##  9  30.7  4.55
## 10  19.8  2.18
## # 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> 22.83398, 44.18717, 19.04913, 15.35077, 40.30678, 49.20174, 34.18298~
## $ x <dbl> 2.31893465, 8.12139199, 1.84864238, 1.22276813, 7.11812395, 9.570762~
```

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

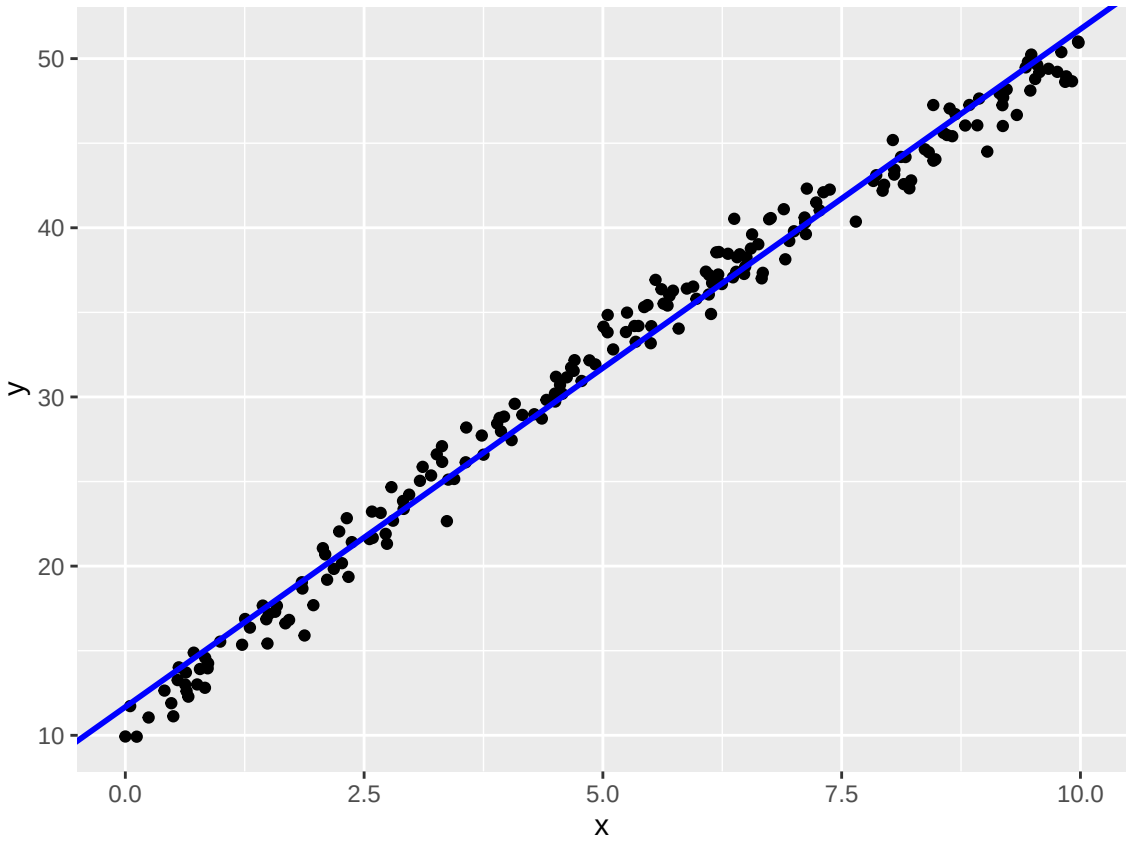


接下来我们用线性回归模型进行拟合（忽略二次项）

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

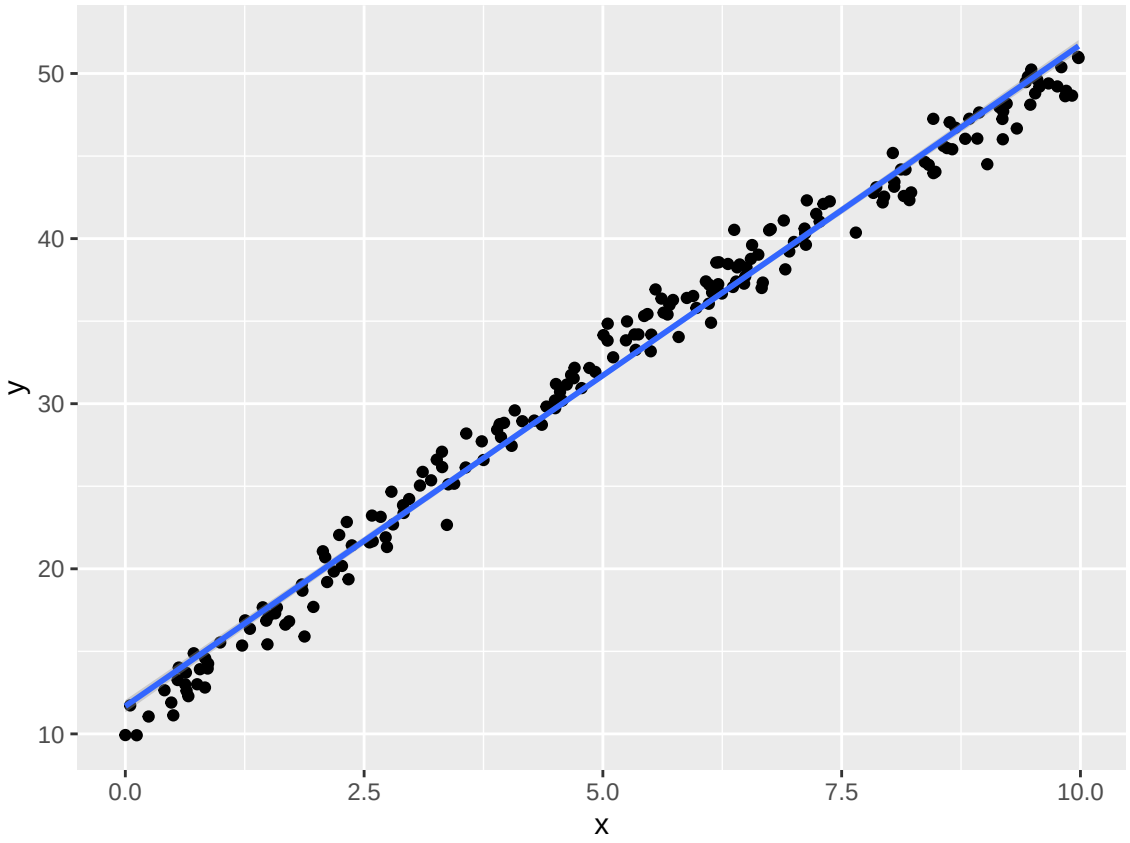
```
##
## Call:
## lm(formula = y ~ x, data = sim_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3320 -0.8522  0.0214  1.0081  3.3102
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.67644    0.19158   60.95   <2e-16 ***
## x            4.00668    0.03248  123.34   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.307 on 198 degrees of freedom
## Multiple R-squared:  0.9872, Adjusted R-squared:  0.9871
## F-statistic: 1.521e+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'
```



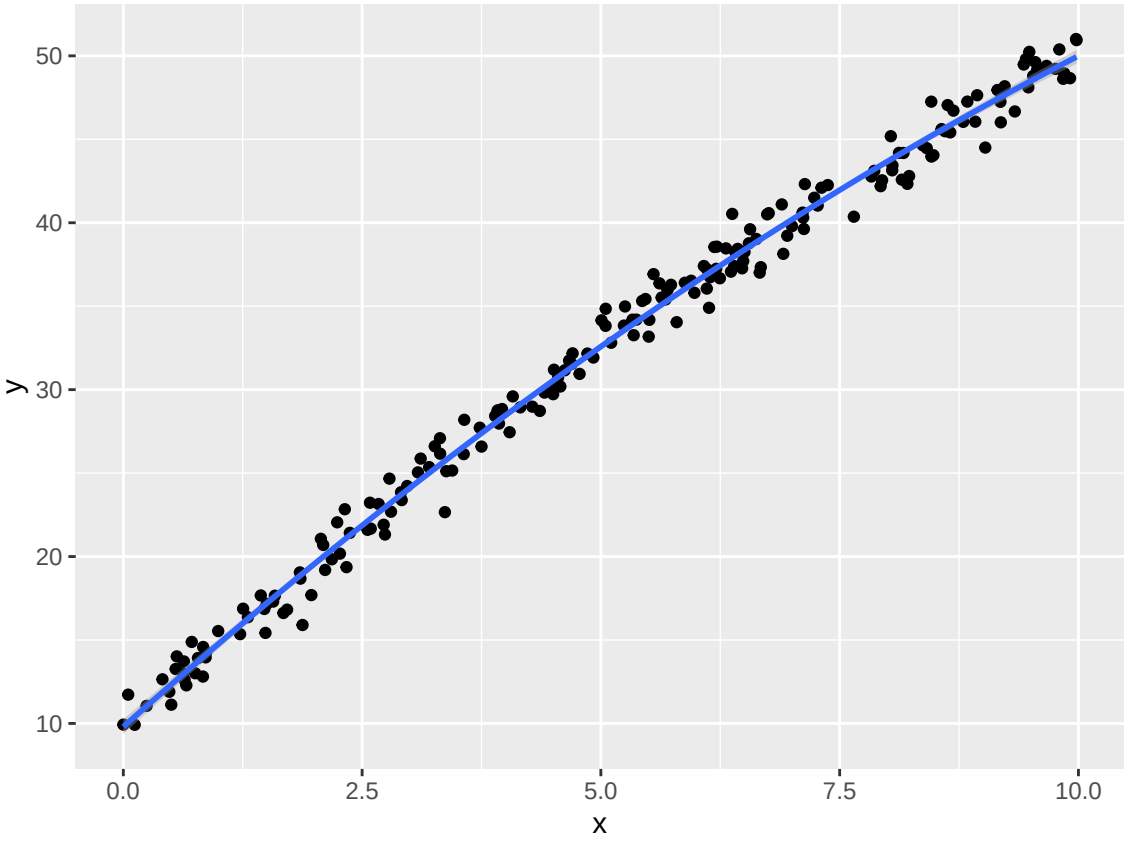
让我们再尝试加入二次项

```
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
## -3.0772 -0.6275  0.0476  0.7405  2.6265
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.800545   0.225427   43.48   <2e-16 ***
## x            5.091714   0.100461   50.68   <2e-16 ***
## I(x^2)      -0.107219   0.009602  -11.17   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.026 on 197 degrees of freedom
## Multiple R-squared:  0.9921, Adjusted R-squared:  0.9921
```

```
## F-statistic: 1.242e+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)")
```



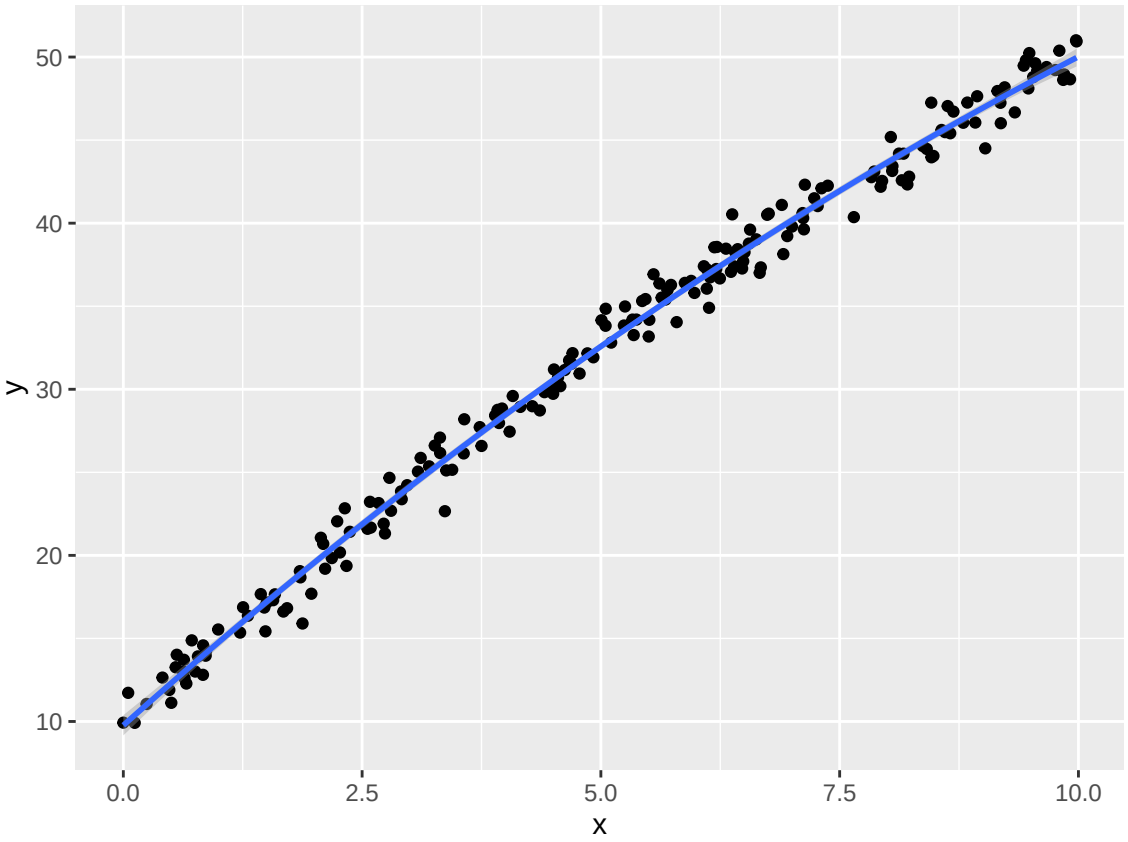
最后尝试加入三次项进行拟合

```
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
## -3.08728 -0.63238  0.05184  0.74926  2.63644
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.7675410  0.3092059  31.589   <2e-16 ***
## x            5.1300475  0.2649990  19.359   <2e-16 ***
## I(x^2)      -0.1165579  0.0604890  -1.927   0.0554 .
```

```
## I(x^3)       0.0006109  0.0039063   0.156   0.8759
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.028 on 196 degrees of freedom
## Multiple R-squared:  0.9921, Adjusted R-squared:  0.992
## F-statistic:  8239 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

```r
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, -0.1)。第二个参数为样本量 n，初始值设定为 200。

调用函数

```r
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
## -2.1187 -0.6687 -0.0726  0.7125  2.2896
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.526961   0.197483  53.306   <2e-16 ***
## x            4.827138   0.092656  52.097   <2e-16 ***
## I(x^2)      -0.084966   0.008937  -9.507   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9176 on 197 degrees of freedom
## Multiple R-squared:  0.994,  Adjusted R-squared:  0.9939
## F-statistic: 1.619e+04 on 2 and 197 DF,  p-value: < 2.2e-16
```

```r
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> 30.551280, 28.078999, 68.187632, 0.208074, 70.765768, 10.959741, 6.3~
## $ x <dbl> 4.6130804, 4.3574142, 7.3185338, 0.2583617, 7.3948667, 2.0689049, 1.~
```

```
ggplot(sim_data3, aes(x, y)) +
  geom_point()
```