计算统计作业 1

朱强强 17064001

2020 年 3 月 10 日

1. 习题 1（一元二次方程）

一元二次方程 $ax^2 + bx + c = 0$ 的根为 2

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}; \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

1) 自定义一个名为 quaroot 的函数求解一元二次方程，其中，a，b 和 c 为函数的输入。

```
> quaroot <- function(a, b, c) {
+   x1 <- (-b + sqrt(b * b - 4 * a * c)) / (2 * a)
+   x2 <- (-b - sqrt(b * b - 4 * a * c)) / (2 * a)
+   return(c(x1, x2))
+ }
```

2) 使用以下方程验证你的函数

$$x^2 + 4x - 1 = 0$$
$$-2x^2 + 2 = 0$$
$$3x^2 - 9x + 1 = 0$$
$$x^2 - 4 = 0$$

```
> quaroot(1, 4, -1)
```

```
[1]  0.236068 -4.236068
```

```
> quaroot(-2, 0, 2)
```

```
[1] -1  1
```

```
> quaroot(3, -9, 1)
```

```
[1] 2.8844373 0.1155627
```

```
> quaroot(1, 0, -4)
```

```
[1]  2 -2
```

3) 当使用 $5x^2+2x+1=0$ 验证时，结果返回什么？为什么没有报错？

```
> quaroot(5, 2, 1)
```

```
Warning in sqrt(b * b - 4 * a * c): 产生了NaNs

Warning in sqrt(b * b - 4 * a * c): 产生了NaNs

[1] NaN NaN
```

$b^2-4ac=4-20=-16<0$，所以该公式无解。

4) 修正你的函数，确保 $b^2-4ac<0$ 是函数返回 NA。

```
> quaroot <- function(a, b, c) {
+   if (b * b - 4 * a * c < 0) {
+     return("NA")
+   } else {
+     x1 <- (-b + sqrt(b * b - 4 * a * c)) / (2 * a)
+     x2 <- (-b - sqrt(b * b - 4 * a * c)) / (2 * a)
+     return(c(x1, x2))
+   }
+ }
```

5) 在数值运算中，两个很大的相近数相减会使有效数字严重损失。衡量 $a=1, b=10^{-8}+10^{8}, c=1$ 时，quaroot 函数的相对误差。

```
> x1 <- quaroot(1, 1e-08 + 1e+08, 1)[1]
> x2 <- quaroot(1, 1e-08 + 1e+08, 1)[2]
> quaroot(1, 1e-08 + 1e+08, 1)
> x3 <- -1e-08
> x4 <- -1e+08
> re1 <- (x1 - x3) / x3
> re2 <- (x2 - x4) / x4
> cat("第一个根对应的相对误差: ", re1, "\n")
> cat("第二个根对应的相对误差: ", re2, "\n")
```

```
[1] -1.490116e-08 -1.000000e+08
第一个根对应的相对误差:  0.4901161
第二个根对应的相对误差:  0
```

第一个根对应的相对误差大。

6) 基于第 (5) 题的观察，能否使用韦达定理 $x_1x_2 = c/a$ 改进流程？

```
> quaroot.new <- function(a, b, c) {
+   if (b * b - 4 * a * c < 0) {
+     return("NA")
+   } else {
+     x1 <- (-b + sqrt(b * b - 4 * a * c)) / (2 * a)
+     x2 <- (-b - sqrt(b * b - 4 * a * c)) / (2 * a)
+     x2 <- min(x1, x2)
+     x1 <- (c / a) / x2
+     return(c(x1, x2))
+   }
+ }
>
> x1 <- quaroot.new(1, 1e-08 + 1e+08, 1)[1]
> x2 <- quaroot.new(1, 1e-08 + 1e+08, 1)[2]
> quaroot.new(1, 1e-08 + 1e+08, 1)
```

```
[1] -1e-08 -1e+08
```

```
> x3 <- -1e-08
> x4 <- -1e+08
> re1 <- (x1 - x3) / x3
> re2 <- (x2 - x4) / x4
> re1
```

```
[1] 0
```

```
> re2
```

```
[1] 0
```

2. 习题 2（简单线性回归）

假设一个不带截距项的简单线性回归模型

$$y_i = \beta_1 x_i + e_i$$

其中，$e_i \sim NID\,(0, \sigma^2)$，　$i = 1, \cdots, n$。

1) 未知参数有哪些？y_i 服从什么分布？整个样本的对数似然函数的数学函数是什么？

未知参数有 β_1。

y_i 服从正态分布：

$$y_i \sim NID\left(\beta_1 x_i, \sigma^2\right), \quad i = 1, \cdots, np(y_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(y_i - \beta_1 x_i)^2}$$

对数似然函数：

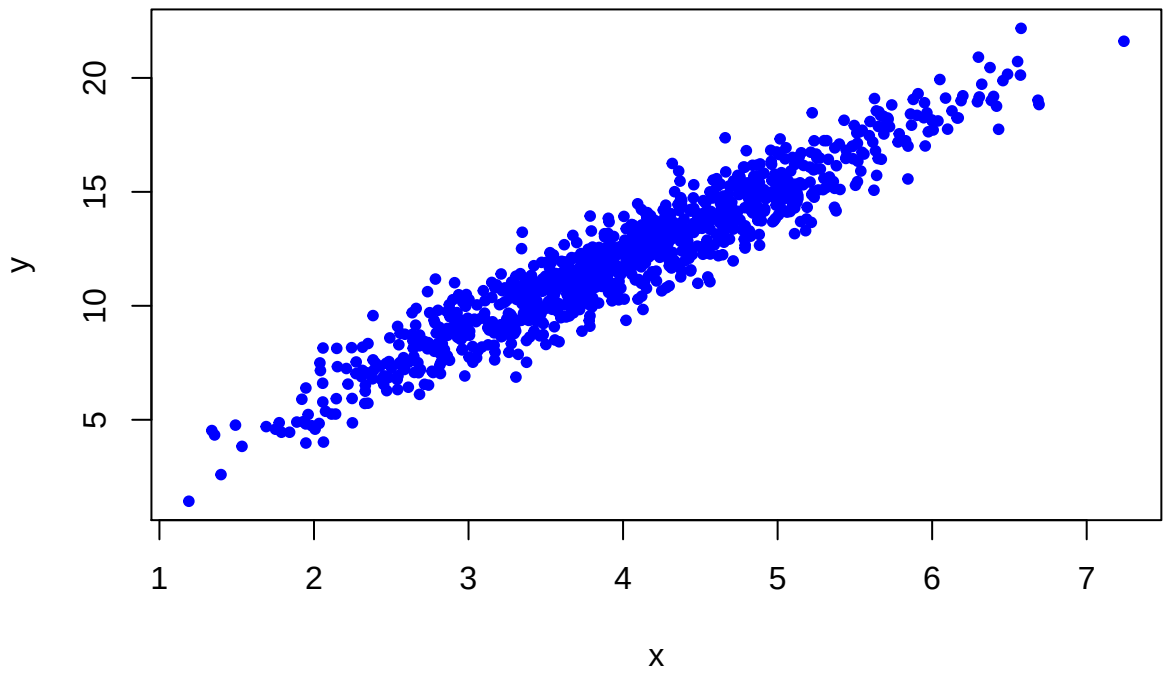
$$\begin{aligned} L(\beta_1, \sigma^2) &= p(y_1)p(y_2)\ldots p(y_n) \\ &= \frac{1}{(2\pi)^{n/2}\sigma^n} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_1 x_i)^2} \end{aligned}$$

$$\begin{aligned} \log L(\beta_1, \sigma^2) &= -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_1 x_i)^2 - \log(2\pi)^{n/2}\sigma^n \\ &= -\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_1 x_i)^2 - \frac{n}{2}\log(2\pi) - n\log\sigma \end{aligned}$$

2) 在 R 中编写一个自定义函数，函数输入为上述所有未知参数、自变量数据向量 ($\boldsymbol{x}$) 和因变量数据向量 ($\boldsymbol{y}$)，函数输出为对数似然函数值。

```
> log.likelihood <- function(beta, sigma, x, y) {
+   result <- -1 / (2 * sigma^2) * sum((y - beta * x)^2) -
+     n / 2 * log(2 * pi) - n * log(sigma)
+   return(result)
+ }
```

```
> set.seed(123)
> beta <- 3
> sigma <- 1
> n <- 1000
> x <- rnorm(n, 4, 1)
> y <- x * beta + rnorm(n, mean=0, sd=sigma)
> plot(x, y, col="blue", pch=20)
```




```
> cat("对数似然函数值: ", log.likelihood(beta, sigma, x, y), "\n")
```

```
对数似然函数值:  -1429.051
```

4) 使用 which.max 找对数似然函数近似最大值对应的参数值。

$$\frac{\partial \log L(\beta_1, \sigma^2)}{\partial \beta_1} = -\frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - \beta_1 x_i) = 0$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i}$$

$$\frac{\partial \log L(\beta_1, \sigma^2)}{\partial \sigma^2} = \frac{1}{2\sigma^4}\sum_{i=1}^{n}(y_i - \beta_1 x_i)^2 - \frac{n}{2\sigma^2} = 0$$

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n}(y_i - \beta_1 x_i)^2}{n}$$

先利用上述公式计算参数值。

```
> beta <- sum(y) / sum(x)
> sigma.sq <- sum((y - beta * x)^2) / n
> print(beta)
```

```
[1] 3.010574
```

```
> print(sigma.sq)
```

```
[1] 1.016703
```

```
> beta.list <- seq(0, 20, by=0.1)
> sigma.list <- seq(0, 20, by=0.1)
> paraAll <- expand.grid(beta.list, sigma.list)
> res.likelihood <- c()
> for (i in 1:nrow(paraAll)) {
+   res.likelihood[i] <- log.likelihood(paraAll[i, 1], paraAll[i, 2], x, y)
+ }
> paraAll[which.max(res.likelihood), ]
```

```
     Var1 Var2
2041    3    1
```

两种计算方式得出的结果是一样的。

5) 用等高线图图示似然函数

```
> contour(x=beta.list, y=sigma.list, z=matrix(res.likelihood, length(beta.list)))
> points(paraAll[which.max(res.likelihood), ], col="red", pch=19)
```