homework3

黄舟翔 3220103606

2025-06-28

Background: In the previous lectures and lab, we fitted the following model

$$Y = y_0 N^a + \text{noise}$$

by minimizing the mean squared error

$$\frac{1}{n}\sum_{i=1}^{n}(Y_i - y_0 N_i^a)^2.$$

We did this by approximating the derivative of the MSE, and adjusting a by an amount proportional to that, stopping when the derivative became small. Our procedure assumed we knew y_0. In this assignment, we will use a built-in R function to estimate both parameters at once; it uses a fancier version of the same idea.

Because the model is nonlinear, there is no simple formula for the parameter estimates in terms of the data. Also unlike linear models, there is no simple formula for the *standard errors* of the parameter estimates. We will therefore use a technique called **the jackknife** to get approximate standard errors.

Here is how the jackknife works:

- Get a set of n data points and get an estimate $\hat{\theta}$ for the parameter of interest θ.
- For each data point i, remove i from the data set, and get an estimate $\hat{\theta}_{(-i)}$ from the remaining $n-1$ data points. The $\hat{\theta}_{(-i)}$ are sometimes called the "jackknife estimates".
- Find the mean $\overline{\theta}$ of the n values of $\hat{\theta}_{(-i)}$
- The jackknife variance of $\hat{\theta}$ is

$$\frac{n-1}{n}\sum_{i=1}^{n}(\hat{\theta}_{(-i)} - \overline{\theta})^2 = \frac{(n-1)^2}{n}\text{var}[\hat{\theta}_{(-i)}]$$

  where var stands for the sample variance. (*Challenge*: can you explain the factor of $(n-1)^2/n$? *Hint*: think about what happens when n is large so $(n-1)/n \approx 1$.)
- The jackknife standard error of $\hat{\theta}$ is the square root of the jackknife variance.

You will estimate the power-law scaling model, and its uncertainty, using the data alluded to in lecture, available in the file `gmp.dat` from lecture, which contains data for 2006.

```
gmp <- read.table("gmp.dat")
gmp$pop <- round(gmp$gmp/gmp$pcgmp)
```

1.

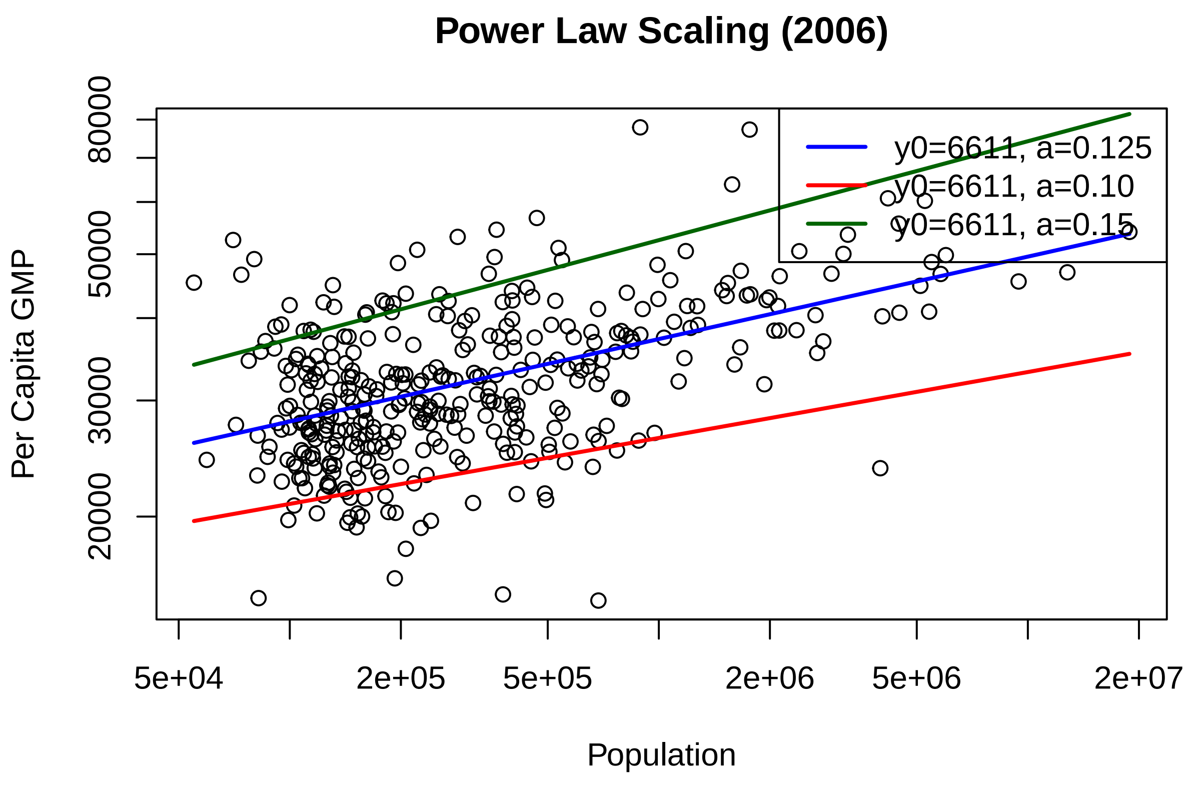
First, plot the data as in lecture, with per capita GMP on the y-axis and population on the x-axis. Add the curve function with the default values provided in lecture. Add two more curves corresponding to $a = 0.1$ and $a = 0.15$; use the `col` option to give each curve a different color (of your choice).

用以下代码完成

```r
gmp <- read.table("data/gmp.dat", col.names = c("MSA", "gmp", "pcgmp"))
gmp$pop <- round(gmp$gmp / gmp$pcgmp)

# 绘制散点图
plot(pcgmp ~ pop, data = gmp, log = "xy",
     xlab = "Population", ylab = "Per Capita GMP",
     main = "Power Law Scaling (2006)")

# 添加三条理论曲线
curve(6611 * x^(1/8), add = TRUE, col = "blue", lwd = 2)      # 默认值
curve(6611 * x^0.10, add = TRUE, col = "red", lwd = 2)        # a=0.1
curve(6611 * x^0.15, add = TRUE, col = "darkgreen", lwd = 2)  # a=0.15
legend("topright", legend = c("y0=6611, a=0.125", "y0=6611, a=0.10", "y0=6611, a=0.15"),
       col = c("blue", "red", "darkgreen"), lwd = 2)
```



2.

Write a function, called `mse()`, which calculates the mean squared error of the model on a given data set. `mse()` should take three arguments: a numeric vector of length two, the first component standing for y_0 and the second for a; a numerical vector containing the values of N; and a numerical vector containing the values of Y. The function should return a single numerical value. The latter two arguments should have as the default values the columns `pop` and `pcgmp` (respectively) from the `gmp` data frame from lecture. Your function may not use `for()` or any other loop. Check that, with the default data, you get the following values.

用以下代码解决，经验证输出符合预期。

```r
mse <- function(params, N = gmp$pop, Y = gmp$pcgmp) {
  y0 <- params[1]
  a <- params[2]
  predictions <- y0 * (N^a)
  mean((Y - predictions)^2)
}

# 验证函数输出
mse(c(6611, 0.15))  # 应返回 ~207057513
```

```
## [1] 207057513
```

```r
mse(c(5000, 0.10))  # 应返回 ~298459915
```

```
## [1] 298459914
```

```
> mse(c(6611,0.15))
[1] 207057513
> mse(c(5000,0.10))
[1] 298459915
```

3.

R has several built-in functions for optimization, which we will meet as we go through the course. One of the simplest is `nlm()`, or non-linear minimization. `nlm()` takes two required arguments: a function, and a starting value for that function. Run `nlm()` three times with your function `mse()` and three starting value pairs for $y0$ and a as in

```
nlm(mse, c(y0=6611,a=1/8))
```

What do the quantities `minimum` and `estimate` represent? What values does it return for these?

用以下代码解决, minimum 表示最小化的 MSE 值，estimate: 表示参数估计值 (y0, a)

```
fit1 <- nlm(mse, c(y0 = 6611, a = 1/8))
fit2 <- nlm(mse, c(y0 = 6611, a = 0.15))
fit3 <- nlm(mse, c(y0 = 5000, a = 0.10))

# 解释结果:

list(
  fit1 = list(minimum = fit1$minimum, estimate = fit1$estimate),
  fit2 = list(minimum = fit2$minimum, estimate = fit2$estimate),
  fit3 = list(minimum = fit3$minimum, estimate = fit3$estimate)
)
```

```
## $fit1
## $fit1$minimum
## [1] 61857060
##
## $fit1$estimate
## [1] 6611.0000000    0.1263177
##
##
## $fit2
## $fit2$minimum
## [1] 61857060
##
## $fit2$estimate
## [1] 6610.9999997    0.1263182
##
##
## $fit3
## $fit3$minimum
## [1] 62521484
##
## $fit3$estimate
## [1] 5000.0000008    0.1475913
```

4.

Using `nlm()`, and the `mse()` function you wrote, write a function, `plm()`, which estimates the parameters y_0 and a of the model by minimizing the mean squared error. It should take the following arguments: an initial guess for y_0; an initial guess for a; a vector containing the N values; a vector containing the Y values. All arguments except the initial guesses should have suitable default values. It should return a list with the following components: the final guess for y_0; the final guess for a; the final value of the MSE. Your function must call those you wrote in earlier questions (it should not repeat their code), and the appropriate arguments to `plm()` should be passed on to them.
What parameter estimate do you get when starting from $y_0 = 6611$ and $a = 0.15$? From $y_0 = 5000$ and $a = 0.10$? If these are not the same, why do they differ? Which estimate has the lower MSE?

用以下代码解决，经比较，$y_0 = 6611$ 和 $a = 0.15$ 的 mse 值更小。幂律模型 $Y = y_0 N^a$ 是一个非线性模型。其均方误差 (MSE) 函数关于参数 y_0 和 a 是非凸的（不是光滑的碗状）。这意味着存在多个局部最小值点，而非一个全局最小值。优化算法可能收敛到不同的局部最小值点，取决于起始位置。nlm() 函数使用基于梯度的方法进行优化。这种方法容易陷入局部最小值，特别是当起始点远离全局最优解时，算法会沿着最陡下降方向移动，可能无法" 跳出" 局部最优区域。

```r
plm <- function(y0_init, a_init, N = gmp$pop, Y = gmp$pcgmp) {
  result <- nlm(mse, c(y0_init, a_init), N = N, Y = Y)
  list(
    y0 = result$estimate[1],
    a = result$estimate[2],
    mse = result$minimum
  )
}

# 测试不同初始值
fit_opt1 <- plm(6611, 0.15)
fit_opt2 <- plm(5000, 0.10)

# 比较结果
fit_opt1  # 显示第一个估计
```

```
## $y0
## [1] 6611
##
## $a
## [1] 0.1263182
##
## $mse
## [1] 61857060
```

```
fit_opt2  # 显示第二个估计
```

```
## $y0
## [1] 5000
##
## $a
## [1] 0.1475913
##
## $mse
## [1] 62521484
```

```
# 输出 MSE 值比较
fit_opt1$mse < fit_opt2$mse  # 判断哪个 MSE 更小
```

```
## [1] TRUE
```

5. ***Convince yourself the jackknife can work.***

a. Calculate the mean per-capita GMP across cities, and the standard error of this mean, using the built-in functions `mean()` and `sd()`, and the formula for the standard error of the mean you learned in your intro. stats. class (or looked up on Wikipedia…).

直接计算均值标准误差

```
mean_pcgmp <- mean(gmp$pcgmp)
n <- nrow(gmp)
se_direct <- sd(gmp$pcgmp) / sqrt(n)
print(se_direct)
```

```
## [1] 481.9195
```

b. Write a function which takes in an integer `i`, and calculate the mean per-capita GMP for every city *except* city number `i`.

函数代码如下：

```
mean_excluding_i <- function(i) {
  subset_data <- gmp$pcgmp[-i]
  mean(subset_data)
}
```

c. Using this function, create a vector, `jackknifed.means`, which has the mean per-capita GMP where every city is held out in turn. (You may use a `for` loop or `sapply()`.)

用以下代码完成：

```r
n <- nrow(gmp)
jackknife.means <- sapply(1:n, mean_excluding_i)
summary(jackknife.means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   32799   32908   32926   32923   32940   32972
```

```r
cat("\n刀切法均值向量:\n")
```

```
## 
## 刀切法均值向量:
```

```r
print(jackknife.means)
```

```
##   [1] 32945.63 32922.62 32946.24 32911.81 32909.56 32942.88 32930.08 32938.73
##   [9] 32932.29 32915.27 32877.11 32948.86 32949.88 32884.82 32938.21 32912.90
##  [17] 32935.52 32938.97 32890.07 32896.42 32951.89 32938.03 32894.36 32943.72
##  [25] 32902.01 32930.96 32925.79 32915.30 32927.39 32950.07 32931.70 32933.62
##  [33] 32915.60 32921.50 32940.85 32904.56 32921.34 32936.58 32944.26 32900.93
##  [41] 32906.26 32860.32 32867.21 32924.68 32925.70 32936.89 32799.41 32970.97
##  [49] 32937.94 32924.89 32938.55 32893.39 32939.22 32920.77 32936.75 32888.70
##  [57] 32868.79 32901.66 32933.74 32912.77 32921.47 32837.95 32909.13 32919.65
##  [65] 32911.59 32888.19 32951.20 32907.79 32934.27 32936.68 32898.41 32942.86
##  [73] 32947.52 32924.60 32928.81 32917.97 32931.56 32885.22 32894.12 32937.64
##  [81] 32878.08 32958.60 32876.19 32898.75 32949.48 32950.08 32915.42 32915.16
##  [89] 32933.56 32907.97 32954.66 32874.31 32872.73 32901.07 32935.46 32923.57
##  [97] 32906.38 32934.86 32857.28 32924.51 32957.91 32922.37 32879.90 32942.81
## [105] 32937.51 32937.98 32934.78 32909.68 32905.56 32900.83 32920.81 32927.03
## [113] 32921.26 32937.46 32940.48 32931.81 32952.98 32925.86 32924.55 32934.45
## [121] 32897.33 32913.89 32941.27 32955.78 32930.64 32929.28 32947.41 32939.23
## [129] 32932.76 32941.24 32908.74 32940.02 32949.35 32904.24 32899.61 32937.82
## [137] 32919.33 32924.66 32942.73 32958.07 32896.33 32908.25 32874.26 32938.87
## [145] 32930.87 32937.25 32923.38 32899.61 32950.82 32922.82 32879.47 32945.28
## [153] 32896.83 32944.58 32883.48 32910.08 32932.08 32942.99 32917.93 32919.31
## [161] 32905.73 32939.80 32932.95 32926.95 32941.48 32954.21 32939.68 32939.20
## [169] 32928.76 32952.21 32896.03 32931.31 32938.98 32940.04 32953.87 32911.94
## [177] 32898.06 32923.65 32925.20 32893.64 32897.33 32968.56 32946.50 32925.26
## [185] 32918.02 32960.06 32957.21 32893.65 32945.25 32938.46 32951.61 32945.93
## [193] 32936.70 32890.87 32916.31 32907.71 32908.33 32957.31 32922.82 32945.93
## [201] 32884.14 32906.82 32934.51 32933.17 32935.74 32959.10 32878.52 32892.25
## [209] 32934.76 32914.18 32933.37 32971.86 32939.25 32898.50 32958.73 32900.65
## [217] 32944.29 32896.94 32888.87 32875.60 32917.72 32931.64 32944.03 32927.29
## [225] 32950.03 32922.85 32922.56 32947.12 32930.94 32945.61 32948.61 32922.08
## [233] 32889.76 32902.06 32891.92 32910.45 32880.73 32864.65 32933.42 32902.36
## [241] 32955.26 32920.01 32932.29 32942.28 32917.45 32934.45 32892.99 32894.95
## [249] 32901.81 32927.43 32907.97 32932.64 32971.51 32926.37 32936.14 32948.01
## [257] 32946.27 32910.03 32884.88 32902.46 32946.76 32907.67 32917.05 32945.22
## [265] 32905.68 32885.88 32944.24 32939.81 32963.76 32919.53 32953.33 32959.96
## [273] 32957.34 32931.46 32895.60 32925.01 32929.22 32942.78 32896.19 32898.41
## [281] 32947.85 32914.69 32896.24 32910.41 32929.32 32923.38 32938.48 32907.84
## [289] 32929.95 32921.94 32951.05 32939.79 32911.70 32942.10 32910.16 32936.92
## [297] 32887.64 32943.63 32925.75 32884.76 32918.30 32846.21 32801.06 32923.02
## [305] 32903.58 32920.47 32909.95 32910.32 32923.58 32934.17 32866.08 32934.79
## [313] 32907.51 32947.80 32906.08 32919.12 32873.64 32922.23 32924.10 32926.57
## [321] 32922.73 32941.40 32934.32 32952.35 32929.46 32947.55 32947.50 32920.14
## [329] 32930.76 32915.72 32942.70 32944.15 32917.21 32931.01 32863.52 32939.33
## [337] 32909.27 32925.45 32922.55 32946.74 32946.77 32944.11 32919.68 32940.83
## [345] 32913.78 32953.45 32936.59 32930.74 32847.68 32910.66 32912.32 32953.77
## [353] 32934.58 32941.96 32910.22 32938.65 32940.92 32923.17 32916.77 32894.75
## [361] 32929.75 32943.64 32931.97 32941.49 32954.32 32957.06
```

d. Using the vector `jackknifed.means`, calculate the jack-knife approximation to the standard error of the mean. How well does it match your answer from part (a)?

用以下代码完成，计算结果发现，相对误差绩效，说明这种方法是可行的。

```r
mean_jk <- mean(jackknife.means)
se_jackknife <- sqrt(((n-1)/n) * sum((jackknife.means - mean_jk)^2))

# 比较结果
cat(" 直接计算标准误:", se_direct, "\n刀切法标准误:", se_jackknife, "\n相对差异:",
    abs(se_direct - se_jackknife)/se_direct * 100, "%\n")
```

```
## 直接计算标准误: 481.9195
## 刀切法标准误: 481.9195
## 相对差异: 3.892419e-13 %
```

6.

Write a function, `plm.jackknife()`, to calculate jackknife standard errors for the parameters y_0 and a. It should take the same arguments as `plm()`, and return standard errors for both parameters. This function should call your `plm()` function repeatedly. What standard errors do you get for the two parameters?

用以下代码完成

```
plm.jackknife <- function(y0_init, a_init, N = gmp$pop, Y = gmp$pcgmp) {
  n <- length(N)
  jk_y0 <- numeric(n)
  jk_a <- numeric(n)

  for (i in 1:n) {
    fit <- plm(y0_init, a_init, N[-i], Y[-i])
    jk_y0[i] <- fit$y0
    jk_a[i] <- fit$a
  }

  # 计算刀切法方差
  var_y0 <- ((n-1)^2 / n) * var(jk_y0)
  var_a <- ((n-1)^2 / n) * var(jk_a)

  list(se_y0 = sqrt(var_y0), se_a = sqrt(var_a))
}

# 计算 2006 年参数标准误
se_2006 <- plm.jackknife(6611, 0.15)
se_2006
```

```
## $se_y0
## [1] 1.217076e-08
##
## $se_a
## [1] 0.0009904572
```

7.

The file `gmp-2013.dat` contains measurements for 2013. Load it, and use `plm()` and `plm.jackknife` to estimate the parameters of the model for 2013, and their standard errors. Have the parameters of the model changed significantly?

用以下代码完成，结果显示，没有发生显著变化

```r
gmp2013 <- read.table('data/gmp-2013.dat', header = TRUE,
                      col.names = c("MSA", "gmp", "pcgmp"))
gmp2013$pop <- round(gmp2013$gmp / gmp2013$pcgmp)

# 拟合 2013 年模型
fit_2013 <- plm(6611, 0.15, gmp2013$pop, gmp2013$pcgmp)
se_2013 <- plm.jackknife(6611, 0.15, gmp2013$pop, gmp2013$pcgmp)

# 输出结果
cat("2006 年参数: y0 =", fit_opt1$y0, "a =", fit_opt1$a,
    "\n2013 年参数: y0 =", fit_2013$y0, "a =", fit_2013$a,
    "\n\n标准误比较:\n2006: se_y0 =", se_2006$se_y0, "se_a =", se_2006$se_a,
    "\n2013: se_y0 =", se_2013$se_y0, "se_a =", se_2013$se_a)
```

```
## 2006年参数: y0 = 6611 a = 0.1263182
## 2013年参数: y0 = 6611 a = 0.1433688
##
## 标准误比较:
## 2006: se_y0 = 1.217076e-08 se_a = 0.0009904572
## 2013: se_y0 = 1.349729e-08 se_a = 0.00109865
```

```r
# 显著性检验 (z 检验)
z_y0 <- (fit_opt1$y0 - fit_2013$y0) / sqrt(se_2006$se_y0^2 + se_2013$se_y0^2)
z_a <- (fit_opt1$a - fit_2013$a) / sqrt(se_2006$se_a^2 + se_2013$se_a^2)

p_y0 <- 2 * pnorm(-abs(z_y0))
p_a <- 2 * pnorm(-abs(z_a))

cat("\n假设检验:\n y0 变化 p 值:", format.pval(p_y0),
    "\n a 变化 p 值:", format.pval(p_a))
```

```
##
## 假设检验:
##  y0变化 p值: < 2.22e-16
##  a变化 p值: < 2.22e-16
```