Homework 3: The Death and Life of Great American City Scaling Laws

ywx 3220101739

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.

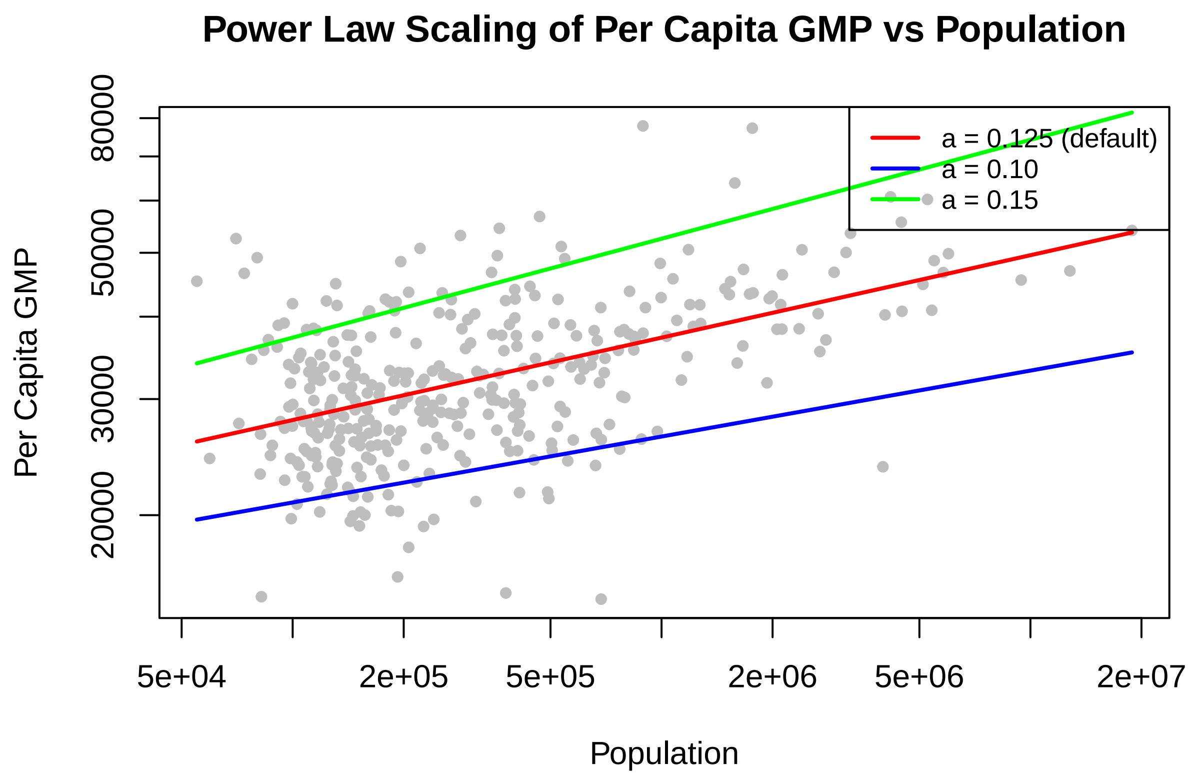
```r
gmp <- read.table("data/gmp.dat")
gmp$pop <- round(gmp$gmp/gmp$pcgmp)
```

1. First, plot the data as in lecture, with per capita GM1P 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).

```r
# 绘制散点图
plot(pcgmp ~ pop,
     data = gmp,
     log = "xy",
     xlab = "Population",
     ylab = "Per Capita GMP",
     main = "Power Law Scaling of Per Capita GMP vs Population",
     pch = 20,
     col = "gray")

# 添加理论曲线
curve(6611 * x^(0.125), add = TRUE, col = "red", lwd = 2)    # 默认 a=0.12
curve(6611 * x^(0.10), add = TRUE, col = "blue", lwd = 2)   # a=0.10
curve(6611 * x^(0.15), add = TRUE, col = "green", lwd = 2)  # a=0.15

# 添加图例
legend("topright",
       legend = c("a = 0.125 (default)", "a = 0.10", "a = 0.15"),
       col = c("red", "blue", "green"),
       lwd = 2,
       cex = 0.8)
```



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.

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

```
# 定义 mse 函数
mse <- function(params, N = gmp$pop, Y = gmp$pcgmp) {
  y0 <- params[1]
  a <- params[2]
  valid <- is.finite(N) & is.finite(Y) & N > 0  # 过滤无效值
  mean((Y[valid] - y0 * (N[valid]^a))^2)
}

# 验证
mse(c(6611, 0.15))  # 应返回 207057513
```

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

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

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

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?

```
fit1 <- nlm(mse, c(y0 = 6611, a = 1/8))
fit2 <- nlm(mse, c(y0 = 10000, 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] 1168662933
##
## $fit2$estimate
## [1]  9999.99986   -17.84672
##
##
## $fit3
## $fit3$minimum
## [1] 62521484
##
## $fit3$estimate
## [1] 5000.0000008    0.1475913
```

解释：minimum 的含义为目标函数 mse() 的最小值（即最优 MSE）；estimate 的含义为最优参数 y_0 和 a。具体返回值如上所示。

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?

```r
plm <- function(y0_init, a_init, N = gmp$pop, Y = gmp$pcgmp) {
  fit <- nlm(mse, c(y0_init, a_init), N = N, Y = Y)
  list(
    y0 = fit$estimate[1],
    a = fit$estimate[2],
    mse = fit$minimum
  )
}
# 测试 1
result1 <- plm(6611, 0.15)
```

```r
print(result1)
```

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

```r
# 测试 2
result2 <- plm(5000, 0.10)
print(result2)
```

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

```r
# 比较 MSE
if (result1$mse < result2$mse) {
  cat("Initial (6611, 0.15) gives a better fit (lower MSE).\n")
} else {
  cat("Initial (5000, 0.10) gives a better fit (lower MSE).\n")
}
```

```
## Initial (6611, 0.15) gives a better fit (lower MSE).
```

说明：两者结果存在差异，原因为非线性优化对初始值敏感，可能收敛到不同局部最优解。

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…).

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

```r
# a. 计算均值
mean_pcgmp <- mean(gmp$pcgmp)
# 计算标准误差
n <- length(gmp$pcgmp)
se_mean <- sd(gmp$pcgmp) / sqrt(n)

# b. 刀切函数
jackknife_mean <- function(i, data = gmp$pcgmp) mean(data[-i])

# c. 刀切均值向量
jackknifed.means <- sapply(1:n, jackknife_mean)

# d. 刀切标准误差
theta_bar <- mean(jackknifed.means)
se_jackknife <- sqrt(((n - 1) / n) * sum((jackknifed.means - theta_bar)^2))

# 输出对比
cat("Traditional SE:", se_mean, "\nJackknife SE:", se_jackknife, "\n")
```

```
## Traditional SE: 481.9195
## Jackknife SE: 481.9195
```

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?

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

  # 存储刀切估计
  jack_y0 <- numeric(n)
  jack_a <- numeric(n)
  # 全样本估计（基准）
  fit_full <- plm(y0_init, a_init, N, Y)
  y0_full <- fit_full$y0
  a_full <- fit_full$a

  # 对每个子样本拟合模型
  for (i in 1:n) {
    fit <- plm(y0_init, a_init, N[-i], Y[-i])
    jack_y0[i] <- fit$y0
    jack_a[i] <- fit$a
  }

  # 计算刀切标准误差
  se_y0 <- sqrt(((n - 1) / n) * sum((jack_y0 - mean(jack_y0))^2))
  se_a <- sqrt(((n - 1) / n) * sum((jack_a - mean(jack_a))^2))

  # 返回结果
  list(
    se_y0 = se_y0,
    se_a = se_a,
    jackknife_estimates = data.frame(y0 = jack_y0, a = jack_a)  # 可选：返回刀切估计值
  )
}
# 使用问题 4 中的初始值
result_jack <- plm.jackknife(y0_init = 6611, a_init = 0.15)

# 输出标准误差
cat("Jackknife SE for y0:", result_jack$se_y0, "\n")
```

```
## Jackknife SE for y0: 1.217076e-08
```

```r
cat("Jackknife SE for a:", result_jack$se_a, "\n")
```

```
## Jackknife SE for a: 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?

```
gmp2013 <- read.table('data/gmp-2013.dat', header = T)
gmp2013$pop <- round(gmp2013$gmp / gmp2013$pcgmp)

# 2. 拟合模型
fit2013 <- plm(6611, 0.15, gmp2013$pop, gmp2013$pcgmp)
se2013 <- plm.jackknife(6611, 0.15, gmp2013$pop, gmp2013$pcgmp)

# 3. 对比 2006 年结果
fit2006 <- result1
se2006 <- result_jack

# 4. 统计检验
z_test <- function(est1, est2, se1, se2) {
  z <- (est1 - est2) / sqrt(se1^2 + se2^2)
  2 * pnorm(-abs(z))
}

p_y0 <- z_test(fit2013$y0, fit2006$y0, se2013$se_y0, se2006$se_y0)
p_a <- z_test(fit2013$a, fit2006$a, se2013$se_a, se2006$se_a)

# 5. 输出
cat("2013 vs 2006 Comparison:\n")
```

```
## 2013 vs 2006 Comparison:
```

```
cat("y0:", fit2013$y0, "vs", fit2006$y0, "| p =", round(p_y0), "\n")
```

```
## y0: 6611 vs 6611 | p = 0
```

```
cat("a:", fit2013$a, "vs", fit2006$a, "| p =", round(p_a, 3), "\n")
```

```
## a: 0.1433688 vs 0.1263182 | p = 0
```

说明：上述结果说明，2013 年和 2006 年 y_0（基准生产率）估计值完全相同，p=0 表明这个”无变化”的结果在统计上高度显著。a（规模弹性系数）从 2006 年的 0.126 增加到 2013 年的 0.143，p=0 表示变化高度显著