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

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.
- **对 challenge 的回答：**
- 因子 $(n-1)^2/n$ 是为了调整样本方差的偏差，使得在有限样本情况下，估计的方差更接近真实的总体方差。具体来说：首先，在其他条件不变的情况下，样本量越大，基于该样本的统计估计量通常越稳定，其方差也越小。
- 我们的目标是估计 $\text{Var}(\hat{\theta})$，其中 $\hat{\theta}$ 是基于 n 个数据点得到的。
- 我们用来估计的工具是 $\hat{\theta}_{(-i)}$ 的集合，其中每一个 $\hat{\theta}_{(-i)}$ 都是基于 $n-1$ 个数据点得到的。因为 $\hat{\theta}_{(-i)}$ 的计算样本量更小，我们可以预期 $\text{Var}(\hat{\theta}_{(-i)})$ 会比 $\text{Var}(\hat{\theta})$ 系统性地偏大。因此，直接使用 $\text{var}(\hat{\theta}_{(-i)})$ 会高估我们真正想知道的 $\text{Var}(\hat{\theta})$。Jackknife 方法需要对此进行修正。
- $\text{Var}(\hat{\theta})$ 对应样本量 n。
- $\text{Var}(\hat{\theta}_{(-i)})$ 对应样本量 $n-1$。
- 我们可以建立如下的近似比例关系：

$$\frac{\text{Var}(\hat{\theta})}{\text{Var}(\hat{\theta}_{(-i)})}\approx\frac{1/n}{1/(n-1)}=\frac{n-1}{n}$$

  通过移项，我们得到了我们想要的 $\text{Var}(\hat{\theta})$ 的一个近似估计：

$$\text{Var}(\hat{\theta})\approx\frac{n-1}{n}\text{Var}(\hat{\theta}_{(-i)})$$

这个公式清晰地告诉我们，应该将 Jackknife 估计量的方差乘以一个因子 $\frac{n-1}{n}$，以校正因样本量减少而带来的方差增加。标准的样本方差公式为：

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

而 Jackknife 方差公式为：

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

我们从样本方差公式中可以得到：

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

将这个关系代入 Jackknife 方差公式中：

$$\text{Var}_{\text{jack}}(\hat{\theta}) = \frac{n-1}{n} \times \left[(n-1)\cdot\text{var}(\hat{\theta}_{(-i)})\right] = \frac{(n-1)^2}{n}\text{var}(\hat{\theta}_{(-i)})$$

这便完美地解释了 $\frac{(n-1)^2}{n}$ 因子的来源。它由两部分相乘得到：

- 一个 $(n-1)$ 因子：它来自于将离差平方和 $\sum(\cdot)^2$ 转换为样本方差 $\text{var}(\cdot)$ 的代数关系。
- 一个 $\frac{n-1}{n}$ 因子：这是我们上面推导出的、用于**校正因样本量变化所引起的方差尺度变化**的关键缩放因子。

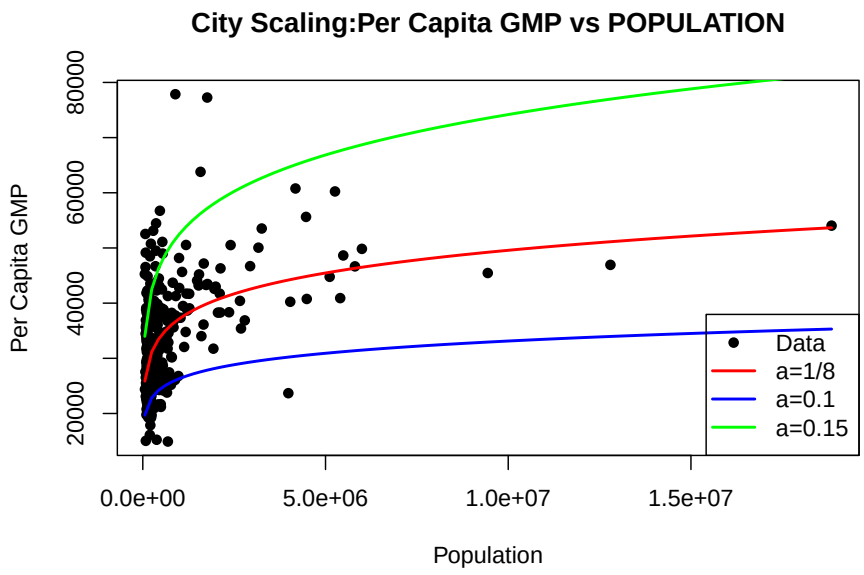
You will estimate the power-law scaling model, and its uncertainty, using the data alluded to in lecture, available in the file `gmp.dat` from lecture, which contains data for 2006.

```
gmp <- read.table("data/gmp.dat")
gmp$pop <- round(gmp$gmp / gmp$pcgmp)
library(ggplot2)
```

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).

```r
plot_data_with_curves <- function() {
  plot(gmp$pop, gmp$pcgmp, xlab = "Population", ylab = "Per Capita GMP",
       main = "City Scaling:Per Capita GMP vs POPULATION",
       pch = 16, col = "black")

  curve(6611 * x^(1 / 8), add = TRUE, col = "red", lwd = 2)
  curve(6611 * x^(0.1), add = TRUE, col = "blue", lwd = 2)
  curve(6611 * x^(0.15), add = TRUE, col = "green", lwd = 2)
  legend("bottomright",
         legend = c("Data", "a=1/8", "a=0.1", "a=0.15"),
         col = c("black", "red", "blue", "green"),
         pch = c(16, NA, NA, NA),
         lty = c(NA, 1, 1, 1),
         lwd = c(NA, 2, 2, 2))
}
plot_data_with_curves()
```



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
```

```r
mse <- function(params, N = gmp$pop, Y = gmp$pcgmp) {
  y0 <- params[1]
  a <- params[2]
  # 添加参数边界检查，防止数值溢出
  if (y0 <= 0 || a < -1 || a > 1) {
    return(1e10)  # 返回一个很大的值表示不合理的参数
  }
  # 计算预测值，使用 try-catch 防止数值溢出
  predicted <- tryCatch({
    y0 * N^a
  }, error = function(e) {
    return(rep(1e10, length(N)))  # 如果出错，返回大数值
  })
  # 检查是否有无限值或 NaN
  if (any(!is.finite(predicted))) {
    return(1e10)
  }
  # 计算 MSE
  mse_value <- mean((Y - predicted)^2)
```

```
  # 确保返回有限值
  if (!is.finite(mse_value)) {
    return(1e10)
  }
  return(mse_value)
}
# 测试 mse 函数
cat("mse(c(6611,0.15)):", mse(c(6611, 0.15)), "\n")
```

```
## mse(c(6611,0.15)): 207057513
```

```
cat("mse(c(5000,0.10)):", mse(c(5000, 0.10)), "\n")
```

```
## mse(c(5000,0.10)): 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?

```
nlm_results_1 <- nlm(mse, c(y0 = 6611, a = 1/8), gradtol = 1e-6, steptol = 1e-6)
nlm_results_2 <- nlm(mse, c(y0 = 5000, a = 0.10),
                     gradtol = 1e-6, steptol = 1e-6)
nlm_results_3 <- nlm(mse, c(y0 = 7000, a = 0.12),
                     gradtol = 1e-6, steptol = 1e-6)
print(nlm_results_1)
```

```
## $minimum
## [1] 61857060
##
## $estimate
```

```
## [1] 6611.0000000    0.1263177
##
## $gradient
## [1]  50.04837 -33.05823
##
## $code
## [1] 2
##
## $iterations
## [1] 3
```

```
print(nlm_results_2)
```

```
## $minimum
## [1] 62521484
##
## $estimate
## [1] 5000.0000008    0.1475909
##
## $gradient
## [1] -1030.49472   -38.86223
##
## $code
## [1] 2
##
## $iterations
## [1] 9
```

```
print(nlm_results_3)
```

```
## $minimum
## [1] 61908051
##
## $estimate
## [1] 7000.0000000    0.1219422
```

```
##
## $gradient
## [1] 203.510306   1.162291
##
## $code
## [1] 2
##
## $iterations
## [1] 4
```

```
cat("The 'minimum' represents the final MSE value at convergence.\n")
```

```
## The 'minimum' represents the final MSE value at convergence.
```

```
cat("The 'estimate' represents the optimized parameter values (y0, a).\n\n")
```

```
## The 'estimate' represents the optimized parameter values (y0, a).
```

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?

```
plm <- function(y0_init, a_init, N = gmp$pop, Y = gmp$pcgmp) {
  # 创建一个包装函数，将 N 和 Y 传递给 mse
  mse_wrapper <- function(params) {
    mse(params, N = N, Y = Y)
```

```
  }
  result <- nlm(mse_wrapper, c(y0 = y0_init, a = a_init),
                gradtol = 1e-6, steptol = 1e-6)
  list(y0 = result$estimate[1],
       a = result$estimate[2],
       mse = result$minimum)
}
# 测试 plm 函数
plm_result_1 <- plm(6611, 0.15)
plm_result_2 <- plm(5000, 0.10)
print(plm_result_1)
```

```
## $y0
## [1] 6611
##
## $a
## [1] 0.1263177
##
## $mse
## [1] 61857060
```

```
print(plm_result_2)
```

```
## $y0
## [1] 5000
##
## $a
## [1] 0.1475909
##
## $mse
## [1] 62521484
```

```
if (abs(plm_result_1$mse - plm_result_2$mse) < 1e-6) {
  cat("Both starting points converge to the same solution.\n")
} else {
```

```
  cat("The estimates differ because the optimization algorithm
  converged to different solutions.\n")
  if (plm_result_1$mse < plm_result_2$mse) {
    cat("The estimate starting from y0=6611, a=0.15 has the lower MSE.\n")
  } else {
    cat("The estimate starting from y0=5000, a=0.10 has the lower MSE.\n")
  }
}
```

```
## The estimates differ because the optimization algorithm
##   converged to different solutions.
## The estimate starting from y0=6611, a=0.15 has the lower MSE.
```

```
cat("\n")
```

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)?

```
# a. 计算平均人均 GMP 和标准误差
mean_gmp <- mean(gmp$pcgmp)
se_gmp <- sd(gmp$pcgmp) / sqrt(nrow(gmp))
cat("5a. Mean per-capita GMP:", mean_gmp, "\n")
```

```
## 5a. Mean per-capita GMP: 32922.53
```

```
cat("    Standard error of mean per-capita GMP:", se_gmp, "\n")
```

```
##     Standard error of mean per-capita GMP: 481.9195
```

```
# b. 定义函数计算除第 i 个城市外的平均人均 GMP
mean_without_i <- function(i) {
  mean(gmp$pcgmp[-i])
}
# c. 创建 jackknifed.means 向量
n <- nrow(gmp)
jackknifed.means <- sapply(1:n, mean_without_i)
# d. 计算 jackknife 标准误差
jackknife_variance <- ((n - 1)^2 / n) * var(jackknifed.means)
jackknife_se <- sqrt(jackknife_variance)
cat("5d. Jackknife standard error:", jackknife_se, "\n")
```

```
## 5d. Jackknife standard error: 481.9195
```

```
cat("    Difference from built-in SE:", abs(jackknife_se - se_gmp), "\n")
```

```
##     Difference from built-in SE: 1.932676e-12
```

```
cat("    Match closely?", abs(jackknife_se - se_gmp) < 1e-10, "\n\n")
```

```
##     Match closely? TRUE
```

8. 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?

```
plm.jackknife <- function(y0_init, a_init, N = gmp$pop, Y = gmp$pcgmp) {
  n <- length(N)
  jackknife_estimates <- matrix(0, nrow = n, ncol = 2)
  colnames(jackknife_estimates) <- c("y0", "a")
  for (i in 1:n) {
```

```
    N_excluded <- N[-i]
    Y_excluded <- Y[-i]
    estimates <- plm(y0_init, a_init, N_excluded, Y_excluded)
    jackknife_estimates[i, ] <- c(estimates$y0, estimates$a)
  }
  mean_y0 <- mean(jackknife_estimates[, "y0"])
  mean_a <- mean(jackknife_estimates[, "a"])
  # 计算 jackknife 方差
  jackknife_variance <- ((n - 1)^2 / n) * apply(jackknife_estimates, 2, var)
  # 返回 y0 和 a 的标准误差
  list(y0_se = sqrt(jackknife_variance[1]), a_se = sqrt(jackknife_variance[2]))
}
best_plm <- plm(6611, 0.15)  # 从第 4 题我们知道这个起始点更好

cat("--- Question 8: Jackknife Standard Errors ---\n")
```

```
## --- Question 8: Jackknife Standard Errors ---
```

```
cat("Using optimal parameters as starting point:\n")
```

```
## Using optimal parameters as starting point:
```

```
cat("y0 =", best_plm$y0, ", a =", best_plm$a, "\n")
```

```
## y0 = 6611 , a = 0.1263177
```

```
# 计算 jackknife 标准误差
se_results <- plm.jackknife(best_plm$y0, best_plm$a)
cat("Standard errors: y0_se =", se_results$y0_se, ", a_se =",
    se_results$a_se, "\n\n")
```

```
## Standard errors: y0_se = 1.137052e-08 , a_se = 0.0009904774
```

6. 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?

```r
# 分析 2006 年数据：寻找 2006 的最优参数
gmp2006 <- read.table("data/gmp.dat", header = TRUE)
gmp2006$pop <- round(gmp2006$gmp / gmp2006$pcgmp)

start_point_1 <- plm(6611, 0.15, N = gmp2006$pop, Y = gmp2006$pcgmp)
start_point_2 <- plm(5000, 0.10, N = gmp2006$pop, Y = gmp2006$pcgmp)

if (start_point_1$mse < start_point_2$mse) {
  best_result_2006 <- start_point_1
  cat("2006: Starting point (6611, 0.15) gives better fit\n")
} else {
  best_result_2006 <- start_point_2
  cat("2006: Starting point (5000, 0.10) gives better fit\n")
}
```

```
## 2006: Starting point (6611, 0.15) gives better fit
```

```r
best_y0_2006 <- best_result_2006$y0
best_a_2006 <- best_result_2006$a

# 计算 2006 年的标准误差
se_2006 <- plm.jackknife(best_y0_2006, best_a_2006,
                         N = gmp2006$pop, Y = gmp2006$pcgmp)

# 分析 2013 年数据
gmp2013 <- read.table('data/gmp-2013.dat', header = TRUE)
gmp2013$pop <- round(gmp2013$gmp / gmp2013$pcgmp)

# 估计 2013 年的参数：使用 2006 年找到的最优参数作为初始猜测值
plm_2013 <- plm(best_y0_2006, best_a_2006, N = gmp2013$pop, Y = gmp2013$pcgmp)

# 计算 2013 年参数的标准误
se_2013 <- plm.jackknife(plm_2013$y0, plm_2013$a,
                         N = gmp2013$pop, Y = gmp2013$pcgmp)
```

```r
# 输出结果
cat("--- Analysis Results ---\n")
```

```
## --- Analysis Results ---
```

```r
cat("2006 Parameter Estimates: y0 =", best_y0_2006, ", a =", best_a_2006, "\n")
```

```
## 2006 Parameter Estimates: y0 = 6611 , a = 0.1263177
```

```r
cat("2006 Standard Errors: y0_se =", se_2006$y0_se, ", a_se =",
    se_2006$a_se, "\n\n")
```

```
## 2006 Standard Errors: y0_se = 1.137052e-08 , a_se = 0.0009904774
```

```r
cat("2013 Parameter Estimates: y0 =", plm_2013$y0, ", a =", plm_2013$a, "\n")
```

```
## 2013 Parameter Estimates: y0 = 6611 , a = 0.1433688
```

```r
cat("2013 Standard Errors: y0_se =", se_2013$y0_se, ", a_se =",
se_2013$a_se, "\n\n")
```

```
## 2013 Standard Errors: y0_se = 3.46066e-08 , a_se = 0.00109852
```

```r
# 统计显著性检验
cat("--- Significance Testing ---\n")
```

```
## --- Significance Testing ---
```

```r
a_diff <- abs(plm_2013$a - best_a_2006)
y0_diff <- abs(plm_2013$y0 - best_y0_2006)

# 使用合并标准误差进行更严格的检验
pooled_se_a <- sqrt((se_2006$a_se^2 + se_2013$a_se^2) / 2)
pooled_se_y0 <- sqrt((se_2006$y0_se^2 + se_2013$y0_se^2) / 2)

cat("Difference in parameter 'a':", a_diff, "\n")
```

```
## Difference in parameter 'a': 0.01705106
```

```r
cat("Pooled SE for 'a':", pooled_se_a, "\n")
```

```
## Pooled SE for 'a': 0.001045895
```

```r
cat("Threshold for significance (2 * pooled SE):", 2 * pooled_se_a, "\n")
```

```
## Threshold for significance (2 * pooled SE): 0.002091789
```

```r
if (a_diff > 2 * pooled_se_a) {
  cat("Conclusion: The change in parameter 'a' is
  statistically significant.\n")
} else {
  cat("Conclusion: The change in parameter 'a' is
  NOT statistically significant.\n")
}
```

```
## Conclusion: The change in parameter 'a' is
##   statistically significant.
```

```r
cat("\nDifference in parameter 'y0':", y0_diff, "\n")
```

```
##
## Difference in parameter 'y0': 2.140023e-07
```

```r
cat("Pooled SE for 'y0':", pooled_se_y0, "\n")
```

```
## Pooled SE for 'y0': 2.575758e-08
```

```r
cat("Threshold for significance (2 * pooled SE):", 2 * pooled_se_y0, "\n")
```

```
## Threshold for significance (2 * pooled SE): 5.151515e-08
```

```r
if (y0_diff > 2 * pooled_se_y0) {
  cat("Conclusion: The change in parameter 'y0' is
  statistically significant.\n")
} else {
  cat("Conclusion: The change in parameter 'y0' is
  NOT statistically significant.\n")
```

```
}
```

```
## Conclusion: The change in parameter 'y0' is
##   statistically significant.
```