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

Xinran Ren, 3220103492

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.

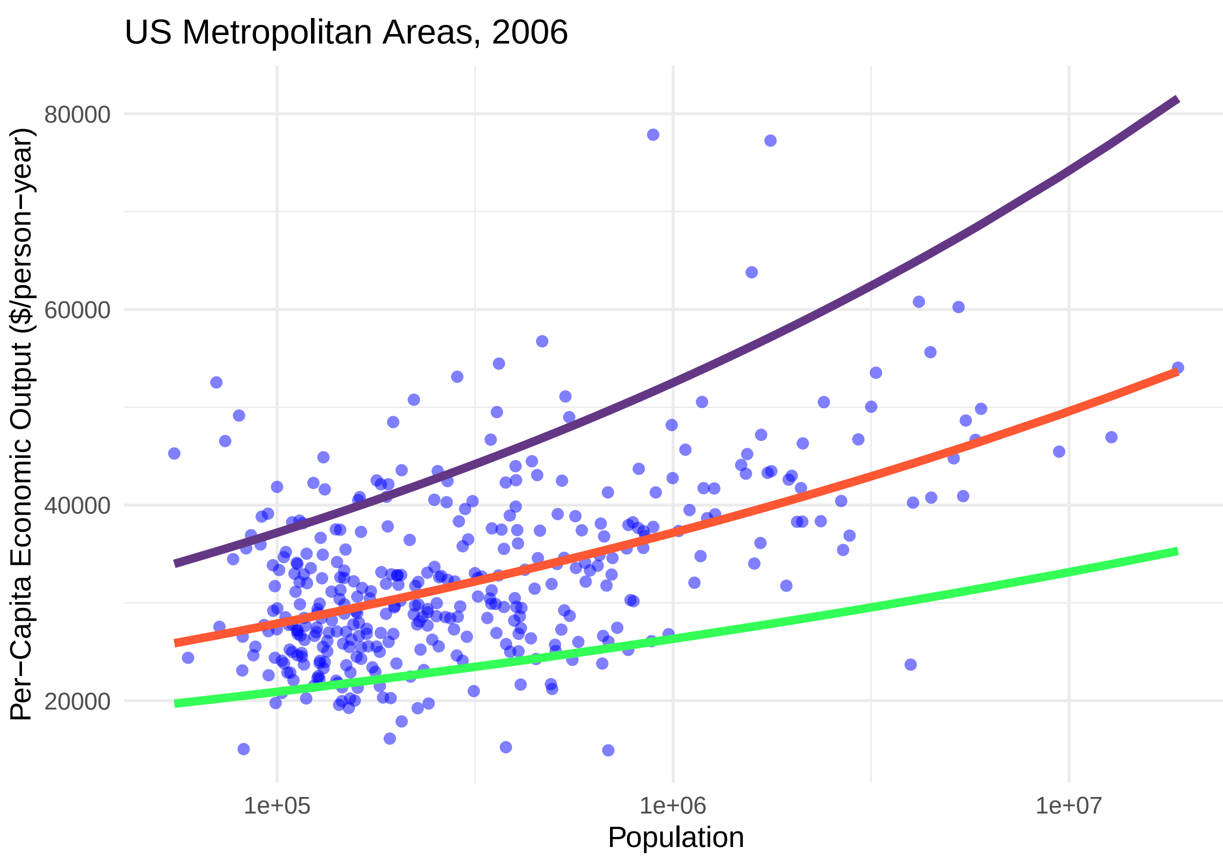
```
gmp <- read.table("gmp.dat")
```

```
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).

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

gmp <- gmp |> mutate(pop = gmp/pcgmp, nlmfit_125 = 6611*pop^(1/8),
                     nlmfit_100 = 6611*pop^(0.1), nlmfit_150 = 6611*pop^(0.15))
gmp |> ggplot() + geom_point(aes(x = pop, y = pcgmp), color = "#0000FF80")+
  labs(x = "Population", y = "Per-Capita Economic Output ($/person-year)",
  title = "US Metropolitan Areas, 2006")+
  scale_x_log10()+
  geom_line(aes(x = pop, y = nlmfit_125), col = '#FF5733', linewidth = 1.5) +
  geom_line(aes(x = pop, y = nlmfit_100), col = '#33FF57', linewidth = 1.5) +
  geom_line(aes(x = pop, y = nlmfit_150), col = '#643785', linewidth = 1.5) +
  theme_minimal()
```



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
# Define the MSE function
mse <- function(params, N = gmp$pop, Y = gmp$pcgmp) {
  y0 <- params[1]
  a  <- params[2]
  mean((Y - y0 * N^a)^2)
}

# Test cases
mse(c(6611, 0.15))  # Should output: 207057513
```

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

```r
mse(c(5000, 0.10))  # Should output: 298459915
```

```
## [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 y_0 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?

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

```
## $minimum
## [1] 61857061
##
## $estimate
## [1] 6611.0000000    0.1263177
##
## $gradient
## [1]  50.048517 -9.976327
##
## $code
## [1] 2
##
```

```
## $iterations
## [1] 3
```

```r
# Run optimization using nlm
result <- nlm(mse, c(y0 = 6611, a = 1/8))$estimate
result <- nlm(mse, result)$estimate
result <- nlm(mse, result)$estimate
result
```

```
## [1] 6610.9999950    0.1263177
```

- `minimum`: the value of the estimated minimum of f.
- `estimate`: the point at which the minimum value of f is obtained.
- It returns values: `minimum`, `estimate`, `gradient`, `code` and `iterations`. Note that the choice of initial values may affect the results of `nlm()`.

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) {
  # Use nlm to minimize mse()
  result <- nlm(mse, c(y0 = y0_init, a = a_init), N = N, Y = Y)

  # Return a list with estimated parameters and minimum MSE
  list(
    y0 = result$estimate[1],
    a = result$estimate[2],
    mse = result$minimum
  )
}
# Starting from y0=6611, a=0.15
fit1 <- plm(6611, 0.15)

# Starting from y0=5000, a=0.10
fit2 <- plm(5000, 0.10)

fit1
```

```
## $y0
```

```
## [1] 6611
##
## $a
## [1] 0.1263182
##
## $mse
## [1] 61857061
```

```r
fit2
```

```
## $y0
## [1] 5000
##
## $a
## [1] 0.1475913
##
## $mse
## [1] 62521485
```

- Different initial values may lead to different parameter estimates due to local minima.
- `fit1` has the lower MSE.

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…).

```r
mean_pcgmp <- mean(gmp$pcgmp)
se_pcgmp <- sd(gmp$pcgmp) / sqrt(length(gmp$pcgmp))
cat("Mean (all cities):", mean_pcgmp, "\n")
```

```
## Mean (all cities): 32922.53
```

```r
cat("Standard error (normal):", se_pcgmp, "\n")
```

```
## Standard error (normal): 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`.

```r
leave_one_out_mean <- function(i) {
  mean(gmp$pcgmp[-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()`.)

```r
jackknifed.means <- sapply(1:nrow(gmp), leave_one_out_mean)
```

   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
n <- length(jackknifed.means)
mean_jack <- mean(jackknifed.means)
jackknife_variance <- ((n - 1) / n) * sum((jackknifed.means - mean_jack)^2)
jackknife_se <- sqrt(jackknife_variance)

# Print results
cat("Mean (all cities):", mean_pcgmp, "\n")
```

```
## Mean (all cities): 32922.53
```

```r
cat("Standard error (normal):", se_pcgmp, "\n")
```

```
## Standard error (normal): 481.9195
```

```r
cat("Jackknife standard error:", jackknife_se, "\n")
```

```
## Jackknife standard error: 481.9195
```

- It match our answer from part (a) very well!

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)

  # Compute leave-one-out estimates for each observation
  jack_estimates <- sapply(1:n, function(i) {
    result <- plm(y0_init, a_init, N = N[-i], Y = Y[-i])
    c(result$y0, result$a)
  })

  jack_estimates <- t(jack_estimates)  # Now each row is an estimate

  # Mean of jackknife estimates
  jack_means <- colMeans(jack_estimates)

  # Jackknife variance formula
  jack_var <- (n - 1) / n * colSums((jack_estimates - matrix(jack_means, n, 2, byrow = TRUE))^2)
  jack_se <- sqrt(jack_var)

  names(jack_se) <- c("SE_y0", "SE_a")
  return(jack_se)
}
```

```r
jackknife_se <- plm.jackknife(6611, 0.15)
cat("Jackknife standard error for y0:", jackknife_se[1], "\n")
```

```
## Jackknife standard error for y0: 1.217122e-08
```

```r
cat("Jackknife standard error for a:", jackknife_se[2], "\n")
```

```
## Jackknife standard error 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?

```r
gmp2013 <- read.table('data/gmp-2013.dat', header = T)
```

```r
# Load 2013 data
gmp2013 <- read.table("data/gmp-2013.dat", header = TRUE)

# Compute population column
gmp2013$pop <- round(gmp2013$gmp / gmp2013$pcgmp)

# Estimate parameters using plm() on 2013 data
fit2013 <- plm(6611, 0.15, N = gmp2013$pop, Y = gmp2013$pcgmp)

# Estimate standard errors using jackknife
se2013 <- plm.jackknife(6611, 0.15, N = gmp2013$pop, Y = gmp2013$pcgmp)

# Display results
cat("Parameter estimates for 2013: y0 =", fit2013$y0, ", a =", fit2013$a, "\n")
```

```
## Parameter estimates for 2013: y0 = 6611 , a = 0.1433688
```

```r
cat("Jackknife standard errors for 2013: se_y0 =", se2013[1], ", se_a =", se2013[2], "\n")
```

```
## Jackknife standard errors for 2013: se_y0 = 1.349727e-08 , se_a = 0.00109865
```

- The parameters of the model have not changed significantly.