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

Background: In the previous lectures and lab, we began to look at user-written functions. For this assignment we will continue with a look at fitting models by optimizing error functions, and making user-written functions parts of larger pieces of code.

In lecture, we saw how to estimate the parameter a in a nonlinear 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

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

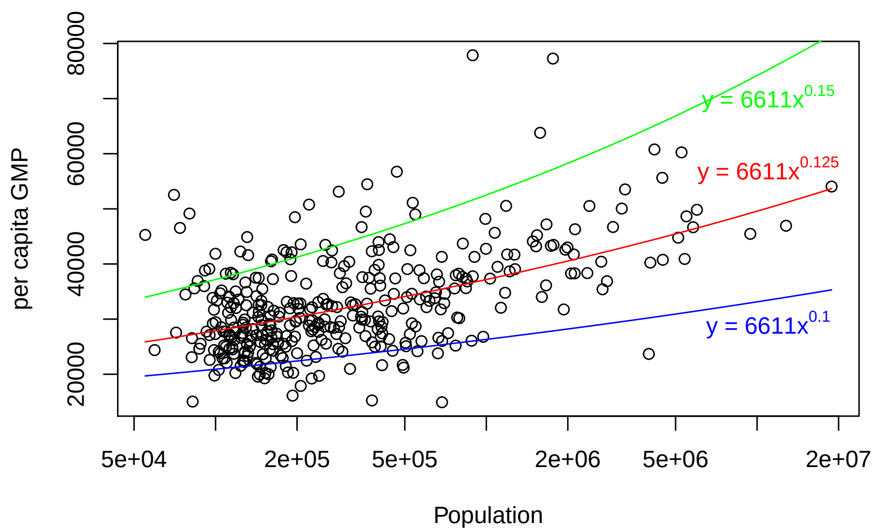
  where var stands for the sample variance. (*Challenge*: can you explain the factor of $(n-1)^2/n$? *Hint*: think about what happens when n is large so $(n-1)/n \approx 1$.)
- The jackknife standard error of $\hat{\theta}$ is the square root of the jackknife variance.

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("data/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).

```
plot(pcgmp~pop, data = gmp, log = "x", xlab = "Population", ylab= "per capita GMP")
curve(6611*x^(1/8), add = TRUE, col = "red")
text(1.1e+07, 57000, expression(paste("y = 6611", x^0.125)), col = "red")
curve(6611*x^(0.1),add=TRUE,col="blue")
text(1.1e+07, 29000, expression(paste("y = 6611", x^0.1)), col = "blue")
curve(6611*x^(0.15),add=TRUE,col="green")
text(1.1e+07, 70000, expression(paste("y = 6611", x^0.15)), col = "green")
```



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 <- function(x,N = gmp$pop,Y = gmp$pcgmp){
  y0 <- x[1]
  a <- x[2]
  return(mean((Y - y0*N^a)^2))
}
 mse(c(6611,0.15))
```

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

```r
mse(c(5000,0.10))
```

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

4. 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

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

```
## $minimum
## [1] 61857060
##
## $estimate
## [1] 6611.0000000    0.1263177
##
## $gradient
## [1] 50.048639 -9.983778
##
## $code
## [1] 2
##
## $iterations
## [1] 3
```

```r
nlm(mse, c(y0=6611,a=0.1))
```

```
## $minimum
## [1] 61857060
##
## $estimate
## [1] 6611.0000003    0.1263177
##
## $gradient
```

```
## [1]   50.04683 -166.46832
##
## $code
## [1] 2
##
## $iterations
## [1] 6
```

```
nlm(mse, c(y0=6611,a=0.15))
```

```
## $minimum
## [1] 61857060
##
## $estimate
## [1] 6610.9999997    0.1263182
##
## $gradient
## [1]   51.76354 -210.18952
##
## $code
## [1] 2
##
## $iterations
## [1] 7
```

What do the quantities `minimum` and `estimate` represent? What values does it return for these?
“minimum”代表着函数估计值的最小值。“estimate”代表着函数取最小值时的点坐标

5. 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(x, N = gmp$pop, Y = gmp$pcgmp){
  result = nlm(mse, x)
  result_1 = list("final_y0" = result$estimate[1], "final_a" = result$estimate[2], "the
  return(result_1)
}
plm(c(y0 = 6611,a = 0.15))
```

```
## $final_y0
## [1] 6611
##
## $final_a
## [1] 0.1263182
##
## $`the final value of the MSE`
## [1] 61857060
```

```
plm(c(y0 = 5000,a = 0,10))
```

```
## $final_y0
## [1] 6494.493
##
## $final_a
## [1] 0.1276774
##
## $`the final value of the MSE`
## [1] 61853982
```

(y0 = 6611,a = 0.15)的结果是(final_y0 = 6611,final_a = 0.1263182,the final value of the MSE = 61857060)。(y0 = 5000,a = 0,10)的结果是(final_y0 = 6494.493,final_a = 0.1276774,the final value of the MSE = 61853982)。因为nlm函数使用的是牛顿迭代法，牛顿迭代法对初值比较敏感，不同的初值会算出不同的结果。MSE较低的是(y0 = 5000,a = 0,10)。

7. *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_GMP <- mean(gmp$pcgmp)
sd_GMP <- sd(gmp$pcgmp)/sqrt(length(gmp$pcgmp) - 1)
sprintf("mean_GMP是 %f,sd_GMP是 %f",mean_GMP,sd_GMP)
```

```
## [1] "mean_GMP是 32922.530055,sd_GMP是 482.579242"
```
b. Write a function which takes in an integer `i`, and calculate the mean per-capita GM

```r
mean_except_pcgmp <- function(i){
  return(mean(gmp$pcgmp[i]))
}
```
c. Using this function, create a vector, `jackknifed.means`, which has the mean per-cap
~~~

```r
jackknifed.means = vector(length = length(gmp$pcgmp))
for ( i in 1:length(gmp$pcgmp)){
jackknifed.means[i] = mean_except_pcgmp(i)
}
```

~~~
d. Using the vector `jackknifed.means`, calculate the jack-knife approximation to the s
~~~

```
jack_knife_approximation <- sd(jackknifed.means)/sqrt(length(gmp$pcgmp) - 1)
print(jack_knife_approximation)
```

```
## [1] 482.5792
```

答案和(a)得出的答案一样。

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(x, N = gmp$pop, Y = gmp$pcgmp){
  n = length(N)
  va = vector(length = n)
  vy = vector(length = n)
  for (i in 1:n) {
    mt = plm(x, N[-i], Y[-i])
    va[i] = mt$final_a
    vy[i] = mt$final_y0
  }
  result = list("sd_y0" = sd(vy) * (n-1) / sqrt(n), "sd_a" = sd(va) * (n-1) / sqrt(n))
  return(result)
}
plm.jackknife(c(6611,0.15))
```

```
## $sd_y0
## [1] 0
##
## $sd_a
## [1] 0
```

仅去掉一个样本对结果的影响是很小的 9. The file `gmp-2013.dat` contains measurements for 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
gmp_2013 <- read.table("data/gmp-2013.dat")
gmp_2013$pop = round(gmp_2013$gmp/gmp_2013$pcgmp)
plm(c(6611,0.15), gmp_2013$pop, gmp_2013$pcgmp)
```

```
## $final_y0
## [1] 6611
##
## $final_a
## [1] 0.1263182
##
## $`the final value of the MSE`
## [1] 61857060
```

```r
plm.jackknife(c(6611,0.15), gmp_2013$pop, gmp_2013$pcgmp)
```

```
## $sd_y0
## [1] 0
##
## $sd_a
## [1] 0
```

参数没有变化或变化很小