Homework 3 - Report

3170105743 李政达

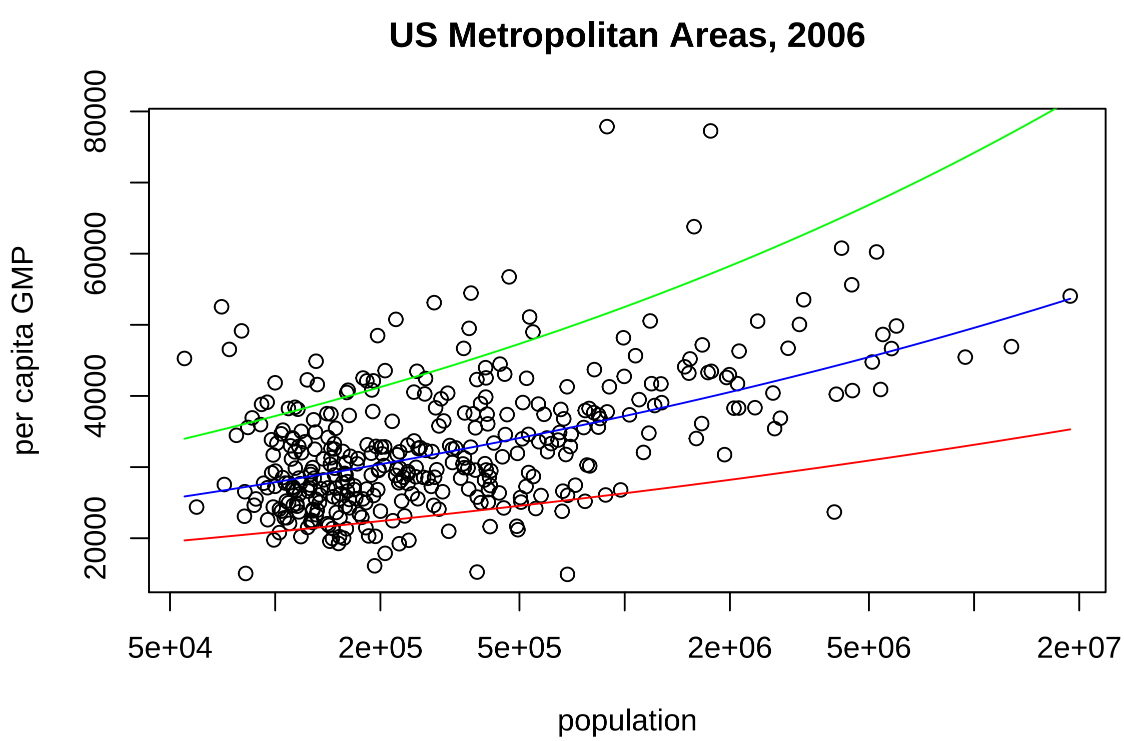
2020/7/8

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(gmp$pcgmp ~ gmp$pop, log = "x",
     xlab = "population", ylab = "per capita GMP",
     main = "US Metropolitan Areas, 2006")
curve(6611*x^(1/8), add = TRUE, col = "blue")
curve(6611*x^(0.1), add = TRUE, col = "red")
curve(6611*x^(0.15), add = TRUE, 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(c(6611,0.15))
[1] 207057513
> mse(c(5000,0.10))
[1] 298459915
```

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

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

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

```
## [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 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?

Ans: The `minimum` is the minimum of `mse(c(y0, a))` which the function can find, and the `estimate` is a vector with the optimal solution.

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

```
## $minimum
## [1] 61857060
##
## $estimate
## [1] 6611.0000000    0.1263177
##
## $gradient
## [1] 50.048639 -9.976327
##
## $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 -166.46087
##
## $code
## [1] 2
##
## $iterations
## [1] 6
```

```r
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
```

We can find that all the three command return the same `minimum`, which is 61857060, and `estimate`, which is 6610.9999997 and 0.1263182.

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.

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

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(6611, 0.15)
```

```
## [1] 6.611000e+03 1.263182e-01 6.185706e+07
```

```r
plm(5000, 0.10)
```

```
## [1] 5.000000e+03 1.475913e-01 6.252148e+07
```

The parameter estimates are different. Because different initial value may lead to different solution, which may be local optimal solution, rather than global optimal solution.

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…).

Ans: Given i.i.d. sample $X_1, \cdots, X_n$, the variance of the mean $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ can be computed as below:

$$Var(\bar{X}) = \frac{1}{n^2}Var(\sum_{i=1}^{n} X_i) = \frac{1}{n^2}\sum_{i=1}^{n} Var(X_i) = \frac{1}{n}Var(X)$$

Then the standard error of the mean is $\sqrt{\frac{1}{n}Var(X)} = \frac{SD(X)}{\sqrt{n}}$.

```
n <- length(gmp$pcgmp)
mean(gmp$pcgmp)
```

```
## [1] 32922.53
```

```
sd(gmp$pcgmp) / sqrt(n)
```

```
## [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`.

```
mean_except <- function(i) {
  return(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()`.)

```
jackknifed.means <- c()
for (city in 1:n) {
  jackknifed.means <- c(jackknifed.means, mean_except(city))
}
```

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)?

```
sqrt(((n-1)^2/n) * var(jackknifed.means))
```

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

We can find that the jack-knife approximation to the standard error of the mean matches the real standard error 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?

```
plm.jackknife <- function(y0, a, N = gmp$pop, Y = gmp$pcgmp) {
  y0.estimate <- c()
  a.estimate <- c()
  n <- length(N)
  for (city in 1:n) {
    temp_res <- plm(y0, a, N[-city], Y[-city])
    y0.estimate <- c(y0.estimate, temp_res[1])
    a.estimate <- c(y0.estimate, temp_res[2])
  }
  y0.std_err <- sqrt(((n-1)^2/n) * var(y0.estimate))
```

```r
  a.std_err <- sqrt(((n-1)^2/n) * var(a.estimate))
  return(c(y0.std_err, a.std_err))
}
plm.jackknife(6611, 1/8)
```

```
## [1] 1.136653e-08 6.583823e+03
```

7. 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
# the file "gmp-2013.dat" is downloaded in
# https://www.stat.cmu.edu/~cshalizi/statcomp/14/hw/04/gmp-2013.dat
gmp.2013 <- read.table("../data/gmp-2013.dat")
gmp.2013$pop <- round(gmp.2013$gmp / gmp.2013$pcgmp)
# plm(6611, 0.15)
# ## [1] 6.611000e+03 1.263182e-01 6.185706e+07
plm(6611, 1/8, gmp.2013$pop, gmp.2013$pcgmp)
```

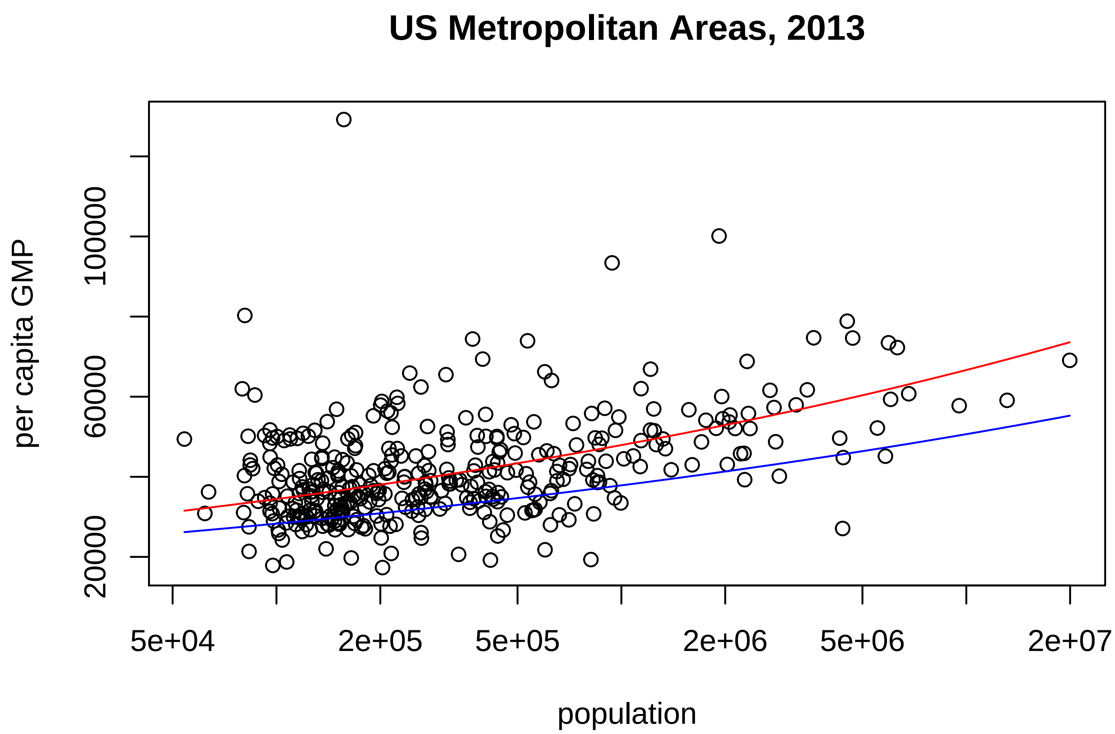
```
## [1] 6.611000e+03 1.433688e-01 1.352105e+08
```

```r
# plm.jackknife(6611, 1/8)
# ## [1] 1.136653e-08 6.583823e+03
plm.jackknife(6611, 1/8, gmp.2013$pop, gmp.2013$pcgmp)
```

```
## [1] 2.692652e-08 6.584869e+03
```

Ans: Obviously, the parameter y_0 has not changed significantly, and it is still 6611 in 2013. The parameter has changed from 0.1263 to 0.1434. We can also plot the data in 2013 and can figure out it more clearly.

```r
plot(gmp.2013$pcgmp ~ gmp.2013$pop, log = "x",
     xlab = "population", ylab = "per capita GMP",
     main = "US Metropolitan Areas, 2013")
curve(6.611000e+03*x^(1.263182e-01), add = TRUE, col = "blue")
curve(6.611000e+03*x^(1.433688e-01), add = TRUE, col = "red")
```


US Metropolitan Areas, 2013
per capita GMP
20000
60000
100000
5e+04
2e+05
5e+05
2e+06
5e+06
2e+07
population