Homework 3

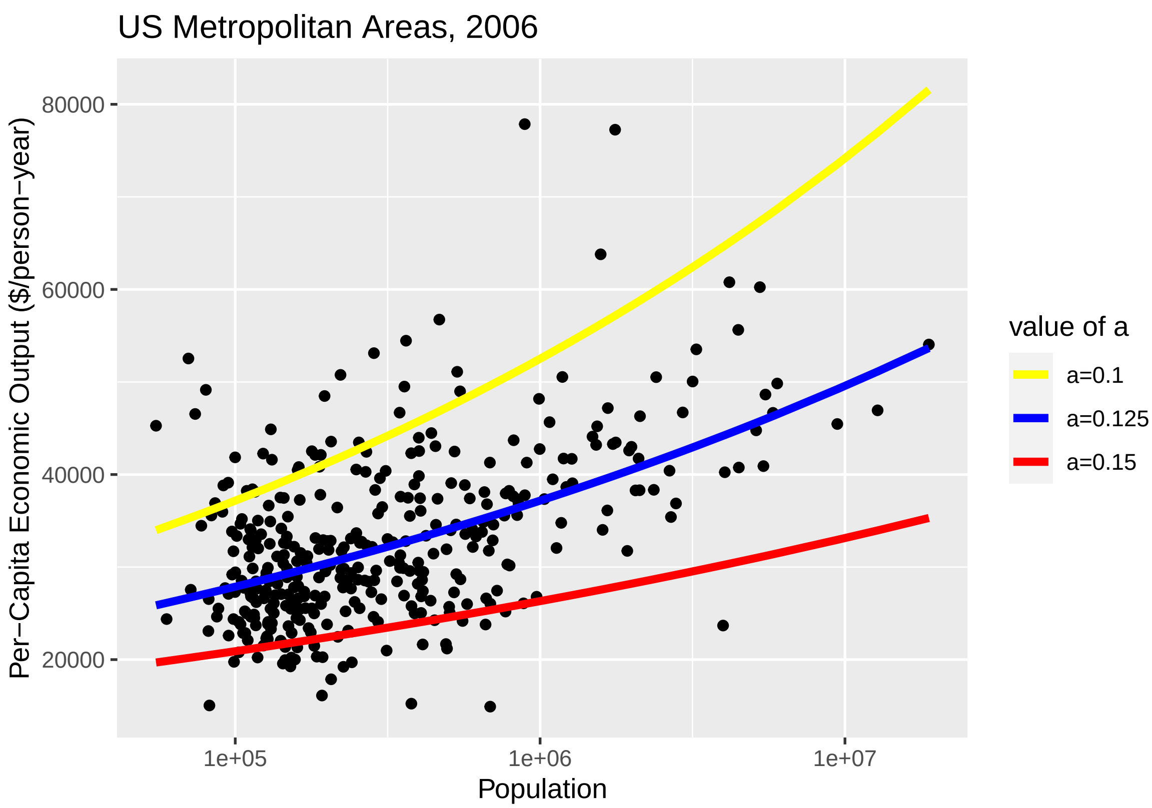
zhang zhuohan

Background: ……

```
library(ggplot2)
library(dplyr)
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).

```
gmp %>% ggplot() + geom_point(aes(pop, pcgmp)) +
    labs(x = "Population", y = "Per-Capita Economic Output ($/person-year)",
        title = "US Metropolitan Areas, 2006") +
    scale_x_log10() +
    geom_line(aes(pop, 6611*pop^(0.125), color="blue"), size=1.5) +
    geom_line(aes(pop, 6611*pop^(0.1), color="red"), size=1.5) +
    geom_line(aes(pop, 6611*pop^(0.15), color="yellow"), size=1.5) +
    scale_colour_manual(name = "value of a",
                        values=c("yellow"="yellow", "blue"="blue", "red"="red"),
                        breaks=c("yellow", "blue", "red"),
                        labels=c("a=0.1", "a=0.125", "a=0.15"))
```



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

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

```
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?

```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
```

```r
nlm(mse, c(y0=6611,a=0.125))
```

```
## $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=0.15))
```

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

- **minimum** represents the value of the estimated minimum of the function.

- **estimate** represents the the point at which the minimum value of the function is obtained.

4. Using `nlm()`, and the `mse()` function you wrote, write a function, `plm()`, which estimates the parameters y0 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.

```
plm <- function(coeff, N = gmp$pop, Y = gmp$pcgmp){
    res <- nlm(mse, c(coeff[1],coeff[2]), N, Y)
    return (list(final_guess=c(res$estimate[1],res$estimate[2]),
                 MSE=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?

```
plm(c(6611,0.15))
```

```
## $final_guess
## [1] 6610.9999997    0.1263182
##
## $MSE
## [1] 61857060
```

```
plm(c(5000,0.10))
```

```
## $final_guess
## [1] 5000.0000008    0.1475913
##
## $MSE
## [1] 62521484
```

The difference may be caused because there are multiple local optimal values.

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…).

  ```
  mean(gmp$pcgmp) # mean
  ```

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

  ```
  sd(gmp$pcgmp)/sqrt(nrow(gmp)) # standard error
  ```

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

  The standard error is calculated as $\sqrt{\frac{1}{n(n-1)} \sum_{i=1}^{n} (x_i - \bar{x})^2}$.

- 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
mean.jackknife <- function(i, Y=gmp$pcgmp){
    return (mean(Y[-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
n <- nrow(gmp)
jackknifed.means <- sapply(1:n, mean.jackknife)
```

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

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

It's actually the same as the answer in part (a).

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(coeff, N=gmp$pop, Y=gmp$pcgmp){
    n <- length(N)
    pre.jackknife <- function(i){
        return(plm(coeff, N[-i], Y[-i])$final_guess)
    }
    jackknife.means <- sapply(1:n, pre.jackknife)
    result <- data.frame(
        y0.se = sqrt(((n-1)^2/n)*var(jackknife.means[1,])),
        a.se  = sqrt(((n-1)^2/n)*var(jackknife.means[2,]))
    )
    return(result)
}
plm.jackknife(c(6611, 0.125))
```

```
##          y0.se         a.se
## 1 1.136653e-08 0.0009901003
```

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
gmp2013 <- read.table("../data/gmp-2013.dat", header=T)
gmp2013$pop <- round(gmp2013$gmp/gmp2013$pcgmp)
plm(c(6611, 0.125), gmp2013$pop, gmp2013$pcgmp)
```

```
## $final_guess
## [1] 6611.0000002    0.1433688
##
## $MSE
## [1] 135210524
```

```r
plm.jackknife(c(6611, 0.125), gmp2013$pop, gmp2013$pcgmp)
```

```
##          y0.se        a.se
## 1 2.692652e-08 0.001098548
```

The parameters of the model didn't change significantly.