STAT 206 Homework 3_Lihua_Xu

Due Monday, October 23, 5:00 PM

General instructions for homework: Homework must be completed as an R Markdown file. Be sure to include your name in the file. Give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Each answer must be supported by written statements as well as any code used. (Examining your various objects in the "Environment" section of RStudio is insufficient – you must use scripted commands.)

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

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

  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.

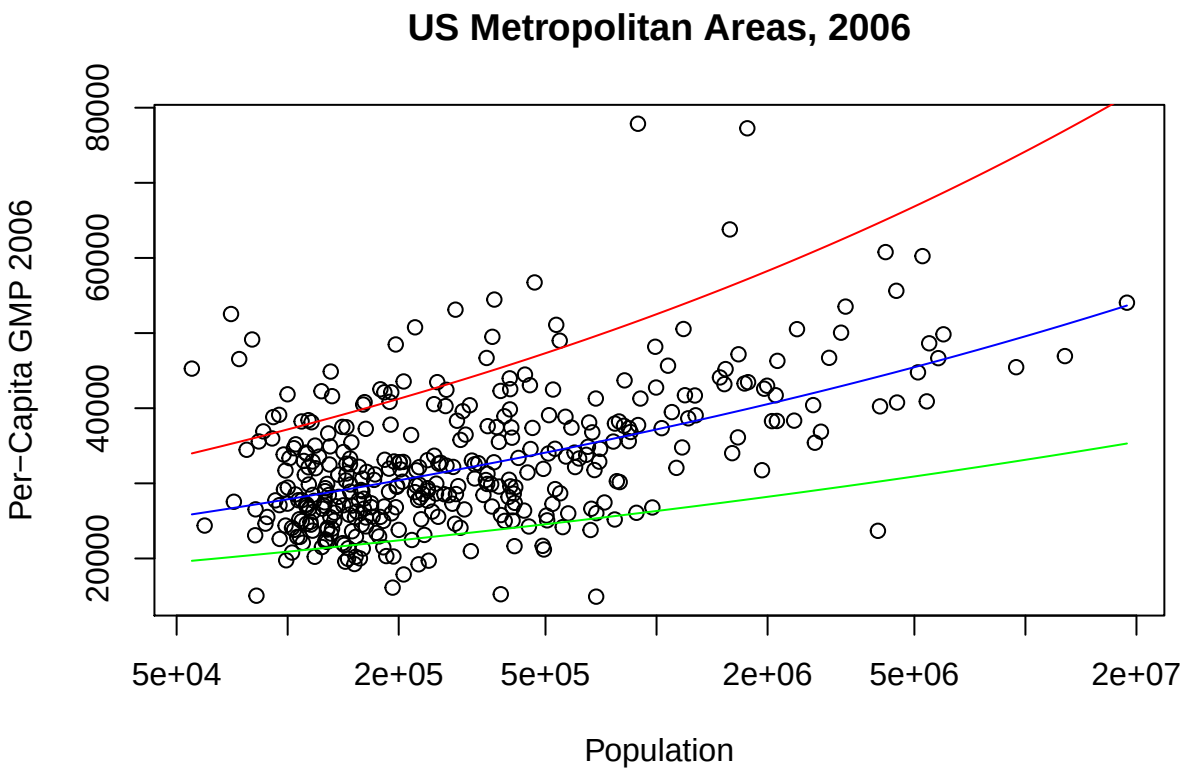
You will estimate the power-law scaling model, and its uncertainty, using the data alluded to in lecture, available in the file `gmp.dat` from lecture, which contains data for 2006.

```
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).

```
gmp <- read.table("gmp.dat")
gmp$pop <- round(gmp$gmp/gmp$pcgmp)
plot(pcgmp~pop, data=gmp,xlab="Population", ylab="Per-Capita GMP 2006",
     main="US Metropolitan Areas, 2006",log="x")
curve(6611*x^(1/8),add=TRUE,col="blue")
```

```
a1 <- 0.1
curve(6611*x^(a1),add=TRUE,col="green")
a2 <- 0.15
curve(6611*x^(a2),add=TRUE,col="red")
```




```
#The blue curve is related to a equal to 1/8.
#The green curve is related to a equal to 0.1.
#The red curve is related to a equal to 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(c(6611,0.15))
[1] 207057513
> mse(c(5000,0.10))
[1] 298459915
```

```
mse <- function(y_0_a,N=gmp$pop,Y=gmp$pcgmp)
{MSE_value <- mean((Y-y_0_a[1]*N^y_0_a[2])^2)
 return(MSE_value)}
mse(c(6611,0.15))
```

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

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

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

```
#I got the same answer as the value provided above.
```

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

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

   What do the quantities `minimum` and `estimate` represent? What values does it return for these?

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

```
## Warning in nlm(mse, c(y0 = 6611, a = 1/8)): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0 = 6611, a = 1/8)): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0 = 6611, a = 1/8)): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0 = 6611, a = 1/8)): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0 = 6611, a = 1/8)): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0 = 6611, a = 1/8)): NA/Inf replaced by maximum
## positive value
```

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

```
#The minimum stand for the value of the estimated minimum of mse.
#The minimum is 61857060.
#The estimate stand for the point at which the minimum value of mse is obtained.
#The estimate is 6611.0000000 0.1263177.
```

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(y_0,a,N=gmp$pop,Y=gmp$pcgmp)
{ it <-0
  max.it <-100000
  first.MSE.value <- mse(c(y_0,a,N,Y))
  while (it < max.it)
  {y_0 <- y_0 - mean(N^a)
   a <- a - mean(a*N^(a-1))
   if (mse(c(y_0,a,N,Y)) < first.MSE.value)
       {first.MSE.value <- mse(c(y_0,a,N,Y))}
       else break()
  }
  fit <- list(y0=y_0,a=a,minimum.mse=first.MSE.value)
  return(fit)}

plm(6611,0.15)
```

```
## $y0
## [1] 4916.279
##
## $a
## [1] 0.148906
##
## $minimum.mse
## [1] 62607483
```

```
#The above three values is related to y_0=6611 and a=0.15.

plm(5000,0.1)
```

```
## $y0
## [1] 4996.441
##
## $a
## [1] 0.0999984
##
## $minimum.mse
## [1] 298459914
```

```
#The above three values is related to y_0=5000 and a=0.1.

#They are not same because of the following several reasons:
#1)In the function there are two variables are changing at the same time.
#2)Onece it chose a direction to minimize, it will go along that direction.
#So there may be some overestimation.
#3)There is no tolerance here, which may attribute to the difference.
#Above all,as every time the input is different, the final estimating values should be different.

#The first estimate y_0=6611 and a=0.15 has the lower mse.
```

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...).

```
n <- dim(gmp)[1]
mean_pcgmp <- mean(gmp$pcgmp)
mean_pcgmp
```

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

```
sd_pcgmp <- sd(gmp$pcgmp)
#The standard deviation for these data is:
sd_pcgmp
```

```
## [1] 9219.663
```

```
standard_error_mean <- sd_pcgmp/(sqrt(n))
#The formula is showing below:
standard_error_mean
```

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

$$u_a = SE = \frac{s}{\sqrt{n}} = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{n(n-1)}}$$

b. Write a function which takes in an integer `i`, and calculate the mean per-capita GMP for every city *except* city number `i`.

```
except_i <- function(i)
{mean_pcgmp <- mean(gmp$pcgmp[-i])
return(mean_pcgmp)}
#The above is the needed function.
except_i(1)
```

```
## [1] 32945.63
```

```
#Trying i=1.And the result is showing above.
```

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_c <- dim(gmp)
Dim_N <- r_c[1]
jackknifed.means <- array(0,c(Dim_N,1))
for (i in 1:Dim_N)
{jackknifed.means[i] <- except_i(i)}
#As the vector has a length 366,here I will not show the result for jackknifed.means.
#jackknifed.means
```

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)?

```
mean_theata <- mean(jackknifed.means)
Jack_stand_error <- sqrt(((Dim_N-1)^2/n)*var(jackknifed.means))
#The jack-knife approximation to the standard error of the mean is:
Jack_stand_error
```

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

```
#The result matches well with the result from (a).
```

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(y_0,a,N=gmp$pop,Y=gmp$pcgmp)
{   y0 <- c()
    a_col <- c()
    n_value <- length(N)
    for (i in 1:n_value) {
        y0 <- c(y0, plm(y_0,a,N[-i],Y[-i])$y0)
        a_col <- c(a_col, plm(y_0,a,N[-i],Y[-i])$a)
    }
    result.y0 <- sqrt(((n-1)^2/n)*var(y0))
    result.a <- sqrt(((n-1)^2/n)*var(a_col))
  fit <- list(standard_error_y0=result.y0,standard_error_a=result.a)
    return(fit)
}
#The standard errors I get for the two parameters are:
#Assuming the initial guess for Y0 and a are 6611 and 0.15 seperately.
plm.jackknife(6611,0.15)
```

```
## $standard_error_y0
## [1] 15.78215
##
## $standard_error_a
## [1] 3.576669e-05
```

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?

```
gmp2013 <- read.table("gmp-2013.dat")
gmp2013$pop <- round(gmp2013$gmp/gmp2013$pcgmp)

#Changing the initial parameters for plm function significantly (3 times)
#y0=1, and a=0.1
plm(1,0.1,N=gmp2013$pop,Y=gmp2013$pcgmp)
```

```
## $y0
## [1] -2.568635
##
## $a
## [1] 0.09999843
##
## $minimum.mse
## [1] 1168424766
```

```
#y0=500, and a=0.15
plm(500,0.15,N=gmp2013$pop,Y=gmp2013$pcgmp)
```

```
## $y0
```

```
## [1] 493.2264
##
## $a
## [1] 0.1499957
##
## $minimum.mse
## [1] 952650412
```

```
#y0=6000, and a=0.12
plm(6000,0.12,N=gmp2013$pop,Y=gmp2013$pcgmp)
```

```
## $y0
## [1] 5995.39
##
## $a
## [1] 0.1199976
##
## $minimum.mse
## [1] 91568594
```

```
#The parameters changes significantly with different initial guess for y0 and a.

#Changing the initial parameters for plm.jackknife function significantly (3 times)
#y0=1, and a=0.1
plm.jackknife(1,0.1,N=gmp2013$pop,Y=gmp2013$pcgmp)
```

```
## $standard_error_y0
## [1] 0.02047418
##
## $standard_error_a
## [1] 5.260068e-08
```

```
#y0=500, and a=0.15
plm.jackknife(500,0.15,N=gmp2013$pop,Y=gmp2013$pcgmp)
```

```
## $standard_error_y0
## [1] 0.0602357
##
## $standard_error_a
## [1] 1.378925e-07
```

```
#y0=6000, and a=0.12
plm.jackknife(6000,0.12,N=gmp2013$pop,Y=gmp2013$pcgmp)
```

```
## $standard_error_y0
## [1] 0.0321496
##
## $standard_error_a
## [1] 7.892895e-08
```

```
#The standard error for the parameters changes little with different initial guess for y0 and a.
```