STAT 206 Homework 3

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.

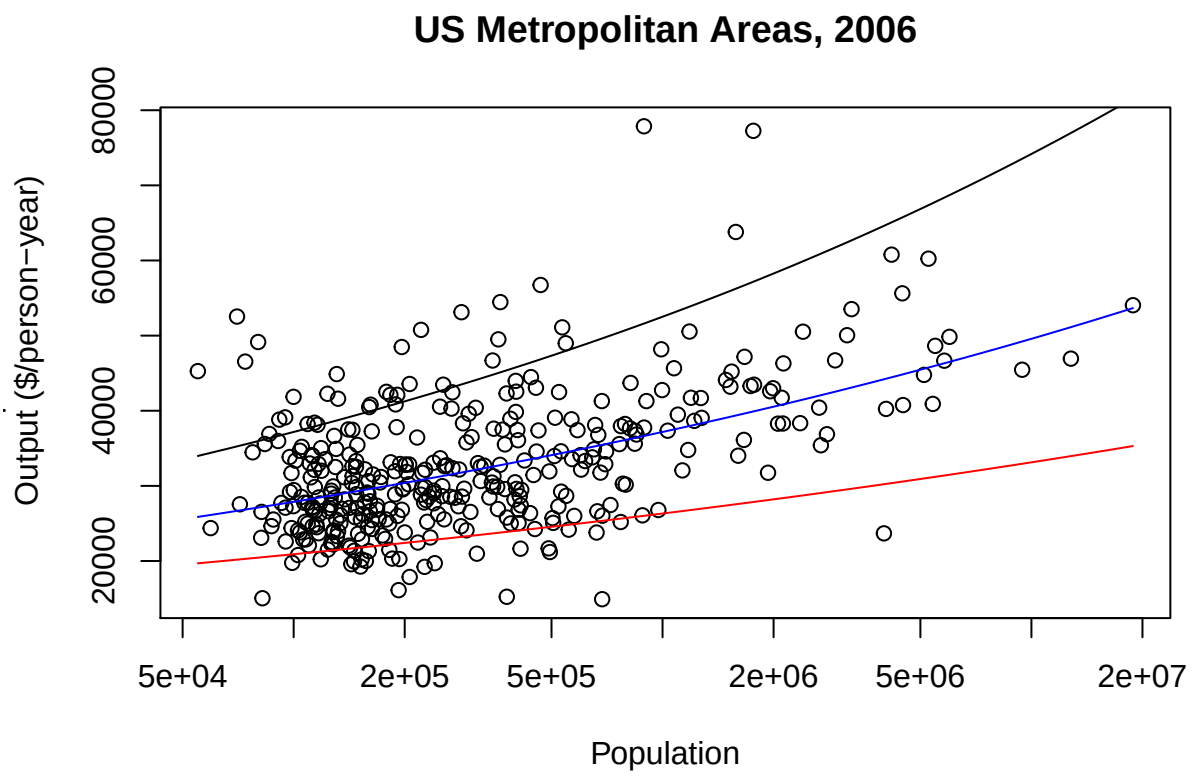
```
gmp <- read.table("gmp.dat")
gmp$pop <- round(gmp$gmp/gmp$pcgmp)
```

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).

```
options(warn=-1)

gmp <- read.table("http://faculty.ucr.edu/~jflegal/206/gmp.dat", header=TRUE)
gmp$pop <- round(gmp$gmp/gmp$pcgmp)
plot(pcgmp~pop, data=gmp, log="x", xlab="Population", ylab="Per-Capita Economic
```

```
    Output ($/person-year)", main="US Metropolitan Areas, 2006")
curve(6611*x^(1/8),add=TRUE,col="blue")
curve(6611*x^(1/10),add=TRUE,col="red")
curve(6611*x^(.15),add=TRUE,col="black")
```




```
#For thie question we read in the table using the website
#we make the gpm pop attribute but dividing the gmp and per capita gmp
#we plot the fuction as in lecture using the log="x" argument to scale the X axis correctly
#this also makes the graph more readable (less squished together)
#finally we add 3 curves with the given 6611 value and use different colors and a values
```

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 (input,pop = gmp$pop, pcgmp = gmp$pcgmp) {
  #b = gmp$pop
  #c = gmp$pcgmp
```

```
  #y_0 = input[1]
  #a_ = input[2]
  return(mean((pcgmp - input[1]*pop^input[2])^2))

}

mse(c(6611, 0.15))
```

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

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

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

```
#class(gmp$pop)

#This function takes the gmp$pop vector as an argument. From the fuction it knows to iterate
# through each value gmp table for the specified columns. I leveraged the mean function within
#our function as shown in lecture. alternatively we could get rid of that and multiply the
#expression by 1/nrow(gmp)
#our function matches the give outputs 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?

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

```
## $minimum
## [1] 61857060
##
## $estimate
## [1] 6611.0000003    0.1263177
##
## $gradient
## [1]   50.04683 -166.46832
##
## $code
## [1] 2
```

```
##
## $iterations
## [1] 6

## $minimum
## [1] 61856513
##
## $estimate
## [1] 6600.0000000    0.1264451
##
## $gradient
## [1] 45.282016 -9.469688
##
## $code
## [1] 2
##
## $iterations
## [1] 3

## $minimum
## [1] 61908051
##
## $estimate
## [1] 7000.0000000    0.1219422
##
## $gradient
## [1] 203.5102844  -0.7301569
##
## $code
## [1] 2
##
## $iterations
## [1] 5
```

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.

```
plm = function(y0, a, pop = gmp$pop, pcgmp = gmp$pcgmp) {
  y0_est = nlm(mse, c(y0 , a), pop, pcgmp)$estimate[1]
  a_est = nlm(mse, c(y0, a), pop, pcgmp)$estimate[2]
  minSE = nlm(mse, c(y0, a), pop, pcgmp)$minimum
  return(list(y0_est, a_est,minSE))
}



#The plm function how makes
```

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?

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

```
## [1] 6611
##
## [[2]]
## [1] 0.1263182
##
## [[3]]
## [1] 61857060
```

```
## [[1]]
## [1] 5000
##
## [[2]]
## [1] 0.1475913
##
## [[3]]
## [1] 62521484
```

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...).

```
#the formula for the standard error is the standard deviation divided by the square root of the populat
#the mean, sd, and standard error are saved to variable objects

avg = mean(gmp$pcgmp)
avg
```

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

```
std = sd(gmp$pcgmp)
std
```

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

```
n = length(gmp$pop)
std_e = std / sqrt(n)
std_e
```

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

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

```
#the function below calls on the cpgmp data and iterates through the data by index
#at each index mean is calculated on all values but the current index value
#we do this by excluding that index value with the -i

remove_one = function (data) {
  for(i in 1:length(data))
    avg = mean(data[-i])
    return(avg)


}

remove_one(gmp$pcgmp)
```

```
## [1] 32957.06
```

c. Using this function, create a vector, `jackknifed.means`, which has the mean per-capita GMP where ev

```
#the function above almost accomplished this however it did not create a vector at the same time.
#the above function was modified such that in each iteration it added the jack knife mean to a
#vector that grew with each iteration
#the most important part of this function is that it return the jackknifed.means globally

jackknifed.means = c()
remove_one = function (data) {
  for(i in 1:length(data))
    jackknifed.means[i] <<- mean(data[-i])

}

remove_one(gmp$pcgmp)
```

d. Using the vector `jackknifed.means`, calculate the jack-knife approximation to the standard error of

```
avg_jk = mean(jackknifed.means)
avg_jk
```

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

```
std_jk = sd(jackknifed.means)
std_jk
```

```
## [1] 25.25935
```

```
std_e = std_jk / sqrt(nrow(gmp))
std_e
```

```
## [1] 1.320327
```

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?

```
## [[1]]
## [1] 8.330748e-06
##
## [[2]]
## [1] 0.7112657
```

```
## [[1]]
## [1] 5.604046
##
## [[2]]
## [1] 0.0009650552
```

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?

```
## [[1]]
## [1] 6611
##
## [[2]]
## [1] 0.1433688
```

```
##
## [[3]]
## [1] 135210524
```

```
## [[1]]
## [1] 5000
##
## [[2]]
## [1] 0.164427
##
## [[3]]
## [1] 139208731
```

```
## [[1]]
## [1] 0.001112349
##
## [[2]]
## [1] 0.390033
```

```
## [[1]]
## [1] 10.98117
##
## [[2]]
## [1] 0.001136817
```