# 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 $\theta$.
- 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

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

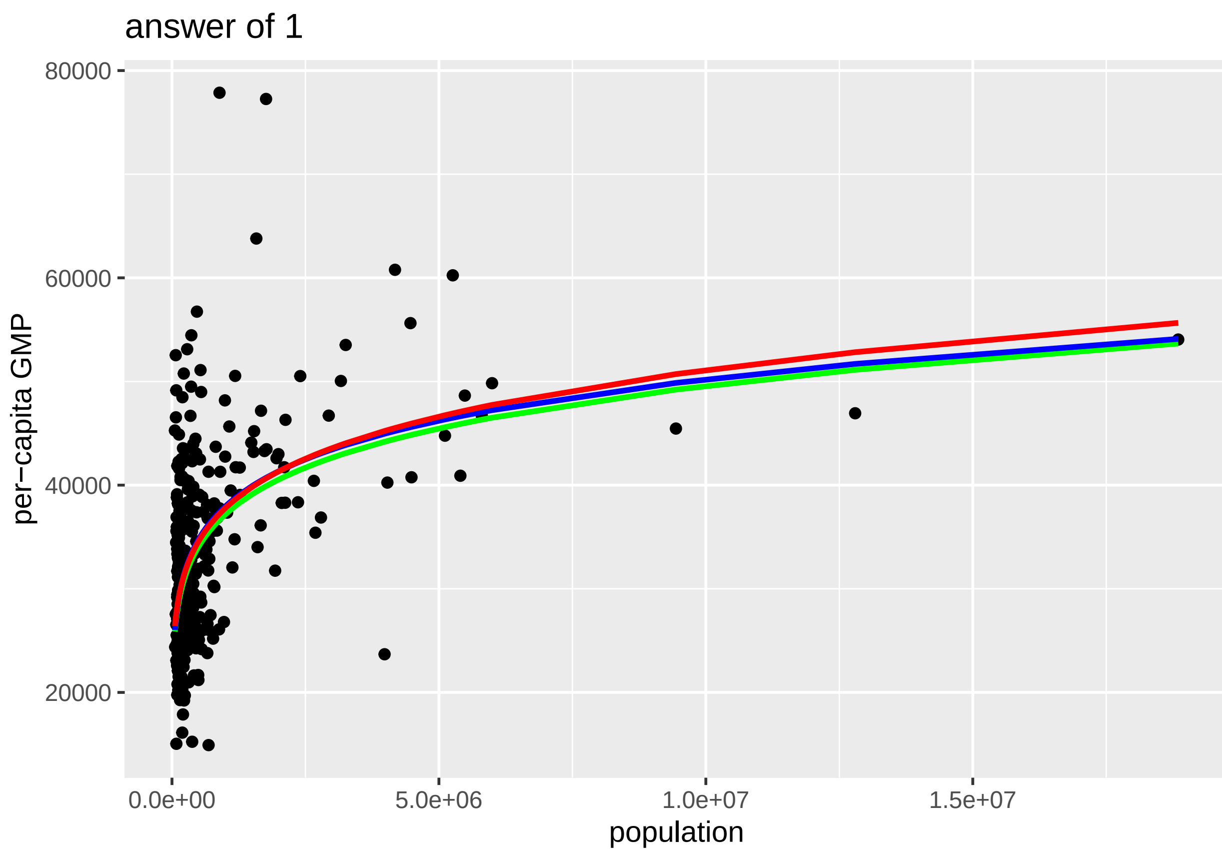
  where var stands for the sample variance. (*Challenge*: can you explain the factor of $(n-1)^2/n$? *Hint*: think about what happens when $n$ is large so $(n-1)/n \approx 1$.)
- The jackknife standard error of $\hat{\theta}$ is the square root of the jackknife variance.

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).

```r
gmp <- read.table("data/gmp.dat")
gmp$pop <- round(gmp$gmp/gmp$pcgmp)
defaults<-c(y0=6611,a=0.125) # the green line
a1<-0.1 # the blue line
a2<-0.15 # the red line
gmp$popa1 <-gmp$pop^a1
gmp$popa2 <-gmp$pop^a2
plm1 <- coefficients(lm(pcgmp~popa1, data=gmp))
plm2 <- coefficients(lm(pcgmp~popa2, data=gmp))
gmp %>% ggplot() + geom_point(aes(x = pop, y = pcgmp))+
  labs(x = "population", y = "per-capita GMP",title = "answer of 1")+
  geom_line(aes(x = pop, y = defaults[1]*pop^defaults[2]), col = ’green’, size = 1.0)+
  geom_line(aes(x = pop, y = plm1[1]+plm1[2]*pop^a1), col = ’blue’, size = 1.0)+
  geom_line(aes(x = pop, y = plm2[1]+plm2[2]*pop^a2), col = ’red’, size = 1.0)
```




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(v,N=gmp$pop,Y=gmp$pcgmp) { mean((Y - v[1]*N^v[2])^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

```
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=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=0.15))
```

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

```r
nlm(mse, c(y0=5000,a=0.1))
```

```
## $minimum
## [1] 62521484
##
## $estimate
## [1] 5000.0000008    0.1475913
##
## $gradient
## [1] -1028.22544    11.38762
##
## $code
## [1] 2
##
## $iterations
## [1] 5
```

```r
# minimum : the value of the estimated minimum of mse.It returns a numerical value.
# estimate : the point at which the minimum value of mse is obtained.It returns a numerical vector.
```

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?

```r
plm<-function(v,N=gmp$pop,Y=gmp$pcgmp) {
mse1<-function(v,n=N,y=Y){mse(v,n,y)}
x0<-nlm(mse1,v)
x<-data.frame(value=x0$minimum[1],y0=x0$estimate[1],a=x0$estimate[2])
return(x)
}
plm(c(6611,0.15))
```

```
##      value   y0         a
## 1 61857060 6611 0.1263182
```

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

```
##      value   y0         a
## 1 62521484 5000 0.1475913
```

```r
# The results are different because y0 of them differ ,and the former one has 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…).
b. Write a function which takes in an integer `i`, and calculate the mean per-capita GMP for every city *except* city number `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()`.)
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)?

```
x<-mean(gmp$pcgmp)
x
```

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

```
n<-length(gmp$pcgmp)
e<-sd(gmp$pcgmp)/sqrt(n)
# via sd() function
e
```

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

```
e1<-sqrt(mean((gmp$pcgmp-x)^2)/n)
# via the formula for the standard error of the mean
e1
```

```
## [1] 481.2607
```

```
CalpcGMP <- function(i) {
  x<-mean(gmp$pcgmp[-i])
  return(x)
}
jackknifed.means<-rep(0,n)
for (i in c(1:n)) {
  jackknifed.means[i]<-CalpcGMP(i)
}
e2<-sqrt((n-1)*(n-1)/n*var(jackknifed.means))
# via  the jack-knife approximation
# using jackknife variance equation's RHS
# It's very close to the  answer e from part (a)
e2
```

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

```
all.equal(e2,e)
```

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

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?

```r
plm.jackknife<-function(v,N=gmp$pop,Y=gmp$pcgmp) {
  n=length(N)
  y<-rep(0,n)
  a<-rep(0,n)
  jackknifed.meansA<-rep(0,n)
  jackknifed.meansY<-rep(0,n)
  for (i in c(1:n)) {
    x<-plm(v,N[-i],Y[-i])
    y[i]<-x[2]
    a[i]<-x[3]
  }
  a<-unlist(a)
  y<-unlist(y)
  for (i in c(1:n)) {
  jackknifed.meansA[i]<-mean(a[-i])
  jackknifed.meansY[i]<-mean(y[-i])
  }
  ea<-sqrt((n-1)*(n-1)/n*var(jackknifed.meansA))
  ey<-sqrt((n-1)*(n-1)/n*var(jackknifed.meansY))
  x<-data.frame(sdey=ey,sdea=ea)
  return(x)
  }
```

```r
plm.jackknife(c(6611,0.15))
```

```
##           sdey         sdea
## 1 3.439384e-11 2.713581e-06
```

```r
plm.jackknife(c(5000,0.10))
```

```
##           sdey         sdea
## 1 5.610612e-11 2.734198e-06
```

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$pcgmp)
N<-gmp.2013$pop
Y<-gmp.2013$pcgmp
# the parameters of the model didn't change significantly
#  parameter-y0 almost didn't change and parameter-a changed a little , about 0.02 larger.
```

```r
plm(c(6611,0.15),N,Y)
```

```
##       value   y0         a
## 1 135210524 6611 0.1433688
```

```r
plm.jackknife(c(6611,0.15),N,Y)
```

```
##           sdey         sdea
## 1 3.587499e-11 2.891184e-06
```

```r
plm(c(5000,0.10),N,Y)
```

```
##       value   y0        a
## 1 139208731 5000 0.164427
```

```r
plm.jackknife(c(5000,0.10),N,Y)
```

```
##           sdey         sdea
## 1 1.646606e-10 2.953983e-06
```