STAT 206 Homework 3

Xin Feng(Vanessa)

Due Monday, October 21, 5:00 PM

General instructions for homework: Homework must be submitted as pdf file, and 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 $(n-1)/n \approx 1$.)

```
# Explain the factor of (n-1)^2/n
# lim(n->infinity)(n-1)^2/n*var(hat_theta(-i)) = var(hat_theta(-i))
# When the data set is large enough, the jackknife varience of hat_theta is
# approximately equal to the varience of hat_theta of raw data.
```

- The jackknife standard error of $\hat{\theta}$ is the square root of the jackknife variance.

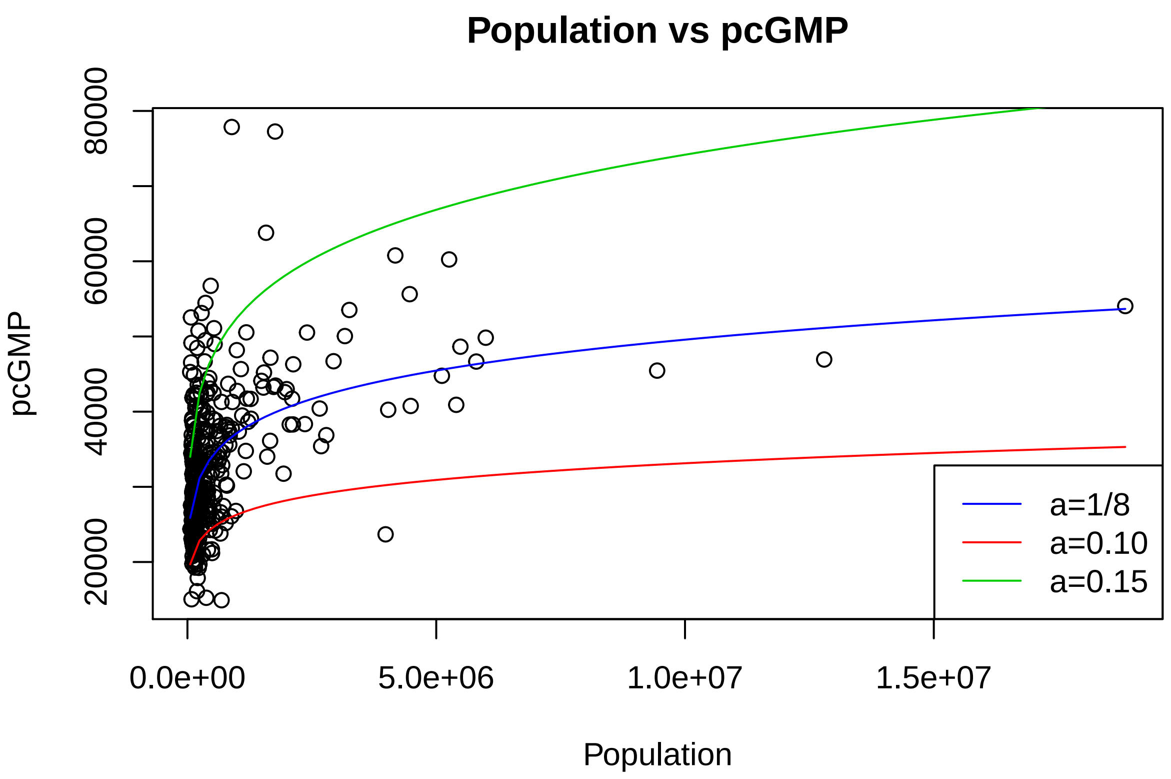
You will estimate the power-law scaling model, and its uncertainty, using the data alluded to in lecture, available in the file `gmp.dat` from lecture, which contains data for 2006.

```
gmp <- read.table("http://faculty.ucr.edu/~jflegal/206/gmp.dat")
gmp$pop <- round(gmp$gmp/gmp$pcgmp)
# View(gmp)
```

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
plot(gmp$pop, gmp$pcgmp, xlab="Population", ylab="pcGMP", main="Population vs pcGMP")
f <- function(x, y0, a){
  return(y0*x^a)
}
# Add curves
curve(f(x, 6611, 1/8), col=4, add=T)

curve(f(x, 6611, 0.10), col=2, add=T)
curve(f(x, 6611, 0.15), col=3, add=T)
legend("bottomright", legend=c("a=1/8", "a=0.10", "a=0.15"), col=c(4,2,3), lty=c(1,1,1))
```



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(value_default, value_x=gmp$pop, value_y=gmp$pcgmp){
  error_minus <- (value_y-value_default[1]*(value_x^value_default[2]))^2
  error_sum <- sum(error_minus)
  error_ms <- error_sum/length(value_y)
  return (error_ms)
}
cat("Mean squared error(y0=6611, a=0.15): ", mse(c(6611, 0.15)), "\n")
```

```
## Mean squared error(y0=6611, a=0.15):  207057513
```

```
cat("Mean squared error(y0=5000, a=0.10): ", mse(c(5000, 0.10)), "\n")
```

```
## Mean squared error(y0=5000, a=0.10):  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?

```
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 -9.976327
##
## $code
## [1] 2
##
## $iterations
## [1] 3
```

```
nlm(mse, c(y0=6611, a=1/2))
```

```
## $minimum
## [1] 1168662933
##
## $estimate
## [1] 6610.9965 -334.0494
##
## $gradient
## [1] 0 0
##
## $code
## [1] 1
##
## $iterations
## [1] 1
```

```
nlm(mse, c(y0=10000, a=1/4))
```

```
## $minimum
## [1] 1168662933
##
## $estimate
## [1] 9999.9984 -219.5155
##
## $gradient
## [1] 0 0
##
## $code
## [1] 1
##
## $iterations
## [1] 1
```

```
# `minimum` means the value of the estimated minimum of mse.
# `estimate` means the point at which the minimum value of mse is obtained.
# (point means c(y0, a))
# a1: when y0=6611, a=0.1263177, the minumum value=61857060, but there is warning
# about the result.
# I consider that the value has not reached the minimum due to the initial value
# setting.
# When (a2)y0=6611, a=1/2 or (a3)y0=10000, a=1/4, the estimated minimum of y both
# equal to 1168662933,
# but the corresponding points(y0, a) which can get the minimum value are different.
# a2: (6610.9965, -334.0494)  a3:(9999.9984, -219.5155)
```

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(y0, a, value_x=gmp$pop, value_y=gmp$pcgmp){
  mini_info <- nlm(mse, c(y0, a), value_x, value_y)
  mini_plm <- c(mini_y0=mini_info$estimate[1], mini_a=mini_info$estimate[2],
                mini_mse=mini_info$minimum)
  return (mini_plm)
}
plm(y0=6611, a=0.15)
```

```
##      mini_y0       mini_a     mini_mse
## 6.611000e+03 1.263182e-01 6.185706e+07
```

```
plm(y0=5000, a=0.10)
```

```
## Warning in nlm(mse, c(y0, a), value_x, value_y): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0, a), value_x, value_y): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0, a), value_x, value_y): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0, a), value_x, value_y): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0, a), value_x, value_y): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0, a), value_x, value_y): NA/Inf replaced by maximum
## positive value
```

```
##      mini_y0       mini_a     mini_mse
## 5.000000e+03 1.475913e-01 6.252148e+07
```

```
# According to the result of plm, with different input, the estimate is different.
# The difference, I consider, is come from the selection of the initial parameters.
# According to the plm, we get the minimal mse. During the fitting process,
# we constantly fine-tune the parameters to make mse smaller. Thus, the initial
# parameters can affect the final fine-tuning result.
# a1:y0=6611, a=0.15 mse=61857060
# a2:y0=5000, a=0.10 mse=62521480
# Situation a1 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…).
    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)?

```
# a) Mean of pcgmp, standard error of the mean
cat("Mean of pcgmp: ", mean(gmp$pcgmp), "\n")
```

```
## Mean of pcgmp:  32922.53
```

```r
cat("Standard error of mean of pcgmp: ", sd(gmp$pcgmp)/(length(gmp$pcgmp)^(1/2)), "\n")
```

```
## Standard error of mean of pcgmp:  481.9195
```

```r
# b) Mean of pcgmp except i-th number.
jk_mean_fun <- function(i=1, value_x=gmp$pcgmp){
  jk_item <- mean(value_x[-i])
  return (jk_item)
}
# c) Vector save
jackknifed.means <- c()
for(i in 1:length(gmp$pcgmp)){
  jk_i <- jk_mean_fun(i)
  jackknifed.means <- c(jackknifed.means, jk_i)
}
# Or b+c) Get
jk_mean_fun2 <- function(value_x=gmp$pcgmp){
  length_i <- length(value_x)
  value_sum <- sum(value_x)
  jk_item <- lapply(value_x, function(c) (value_sum-c)/(length_i-1))
  return (jk_item)
}
jk_mean <- jk_mean_fun2()
# d) Standard error of the mean
cat("Jackknifed-standard error of the mean: ", sd(jackknifed.means)/
      (length(gmp$pcgmp)^(1/2)), "\n")
```

```
## Jackknifed-standard error of the mean:  1.320327
```

```r
# After the jackknife process, the standard varience of the mean is smaller than
# that obtained from the raw data.
```

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(y0, a, value_x=gmp$pop, value_y=gmp$pcgmp){
  length_n <- length(value_x)
  plm_y0 = c()
  plm_a = c()
  for(i in 1:length(value_x)){
    plm_item <- plm(y0, a, value_x[-i], value_y[-i])
    plm_y0 <- c(plm_y0, plm_item[["mini_y0"]])
    plm_a <- c(plm_a, plm_item[["mini_a"]])
  }
  plm_y0_sd <- ((length_n-1)^2/length_n*var(plm_y0))^(1/2)
  plm_a_sd <- ((length_n-1)^2/length_n*var(plm_a))^(1/2)
  return(c(plm_y0_sd, plm_a_sd))
}
plm_result <- plm.jackknife(6611, 0.15)
cat("standard errors of y0: ", plm_result[1], "\n")
```

```
## standard errors of y0:  1.217076e-08
```

```r
cat("standard errors of a: ", plm_result[2], "\n")
```

```
## standard errors of a:  0.0009904572
```

9. The file `gmp-2013.dat` contains measurements 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("http://faculty.ucr.edu/~jflegal/206/gmp-2013.dat")
# View(gmp_2013)
gmp_2013$pop <- gmp_2013$gmp/gmp_2013$pcgmp
plm_item <- plm(y0=6611, a=0.15, value_x=gmp_2013$pop, value_y=gmp_2013$pcgmp)
plm_sd <- plm.jackknife(y0=6611, a=0.15, value_x=gmp_2013$pop, value_y=gmp_2013$pcgmp)
cat("Estimation of the y0: ", plm_item[["mini_y0"]], "\nAnd its standard errors: "
    , plm_sd[1], "\n")
```

```
## Estimation of the y0:  6611
## And its standard errors:  1.349723e-08
```

```r
cat("Estimation of the a: ", plm_item[["mini_a"]], "\nAnd its standard errors: "
    , plm_sd[2], "\n")
```

```
## Estimation of the a:  0.1433688
## And its standard errors:  0.00109865
```

```r
# According to the result, the parameters of the model don't have great change.
```