Homework 3: The Death and Life of Great American City Scaling Laws

options(repos = c(CRAN = "https://mirrors.tuna.tsinghua.edu.cn/CRAN/"))

**Background**: In the previous lectures and lab, we fitted the following model

by minimizing the mean squared error

We did this by approximating the derivative of the MSE, and adjusting by an amount proportional to that, stopping when the derivative became small. Our procedure assumed we knew . 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 data points and get an estimate for the parameter of interest .
* For each data point , remove from the data set, and get an estimate from the remaining data points. The are sometimes called the “jackknife estimates”.
* Find the mean of the values of
* The jackknife variance of is
* where stands for the sample variance. (*Challenge*: can you explain the factor of ? *Hint*: think about what happens when is large so .)

它代表了每个估计值的权重。这里的权重表示每个子样本的相对大小，也就是被删除的观测值占原始数据集的比例。当数据量很大时，每个子样本的大小几乎都等于整个数据集的大小， 因此每个估计值的权重都相当接近。

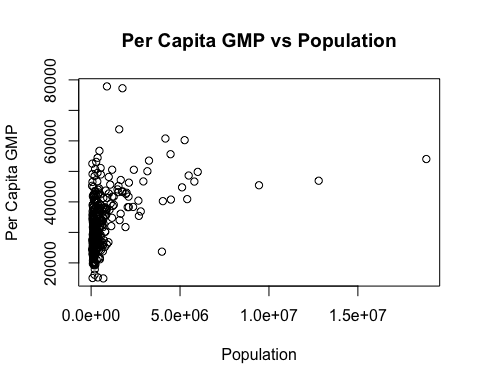
样本方差的公式是，其中使样本方差成为总体方差的无偏估计，后面 多除了一个前面要多乘一个，故出现了

* The jackknife standard error of is the square root of the jackknife variance.

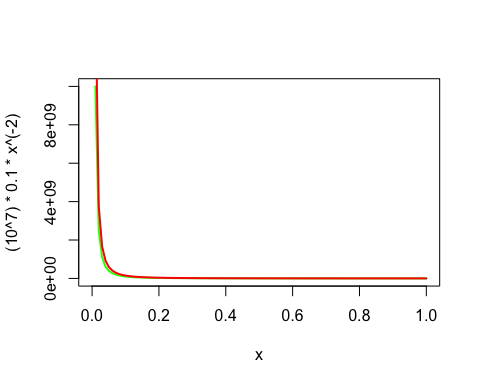
You will estimate the power-law scaling model, and its uncertainty, using the data alluded to in lecture, available in the file gmp.dat from lecture, which contains data for 2006.

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 and ; use the col option to give each curve a different color (of your choice).

# 读取数据  
gmp <- read.table("./mynotes/data/gmp.dat",header=T)  
gmp$pop <- round(gmp$gmp/gmp$pcgmp)  
  
# 绘制散点图  
plot(gmp$pop, gmp$pcgmp, xlab = "Population", ylab = "Per Capita GMP", main = "Per Capita GMP vs Population")



# 添加a = 0.1的曲线  
curve(expr = (10^7) \* 0.1 \* x^(-2), col = "green", lwd = 2)  
  
# 添加a = 0.15的曲线  
curve(expr = (10^7) \* 0.15 \* x^(-2), col = "red", lwd = 2, add = TRUE)

 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 and the second for ; a numerical vector containing the values of ; and a numerical vector containing the values of . 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.

#计算均方误差  
mse <- function(params, N = gmp$pop, Y = gmp$pcgmp) {  
 y0 <- params[1]  
 a <- params[2]  
  
 predicted <- y0 \* N^a  
 mse\_value <- mean((predicted - Y)^2)  
  
 return(mse\_value)  
}  
 mse(c(6611,0.15))

## [1] 207057513

mse(c(5000,0.10))

## [1] 298459914

1. 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 and as in

# 定义数据集中的人口数量  
N <- gmp$pop  
  
# 调用 nlm() 函数三次，分别传入不同的起始值  
res1 <- nlm(mse, c(y0 = 7000, a = 0.15))  
res2 <- nlm(mse, c(y0 = 10000, a = 0.12))  
res3 <- nlm(mse, c(y0 = 4000, a = 0.18))  
  
# 输出每次最小化得到的结果  
res1

## $minimum  
## [1] 61908051  
##   
## $estimate  
## [1] 6999.9999997 0.1219422  
##   
## $gradient  
## [1] 203.510313 1.832843  
##   
## $code  
## [1] 2  
##   
## $iterations  
## [1] 7

res2

## $minimum  
## [1] 63606178  
##   
## $estimate  
## [1] 1.00000e+04 9.44909e-02  
##   
## $gradient  
## [1] 799.60521 20.16127  
##   
## $code  
## [1] 2  
##   
## $iterations  
## [1] 6

res3

## $minimum  
## [1] 64171247  
##   
## $estimate  
## [1] 3999.9999997 0.1644647  
##   
## $gradient  
## [1] -2419.98483 -14.59569  
##   
## $code  
## [1] 2  
##   
## $iterations  
## [1] 5

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

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

在R语言的nlm()函数中，minimum和estimate分别表示目标函数的最小值和使目标函数最小化的参数值。

minimum代表目标函数在优化过程中达到的最小值。对于回归问题中的均方误差函数而言，minimum表示预测值与真实值之间的平均误差的平方和。较小的minimum值表示模型的预测结果较接近真实值。

estimate代表使目标函数最小化的参数值。对于回归问题而言，它是一个包含参数估计值的向量。例如，在简单线性回归中，estimate通常包含截距和斜率两个参数的值。通过优化过程，nlm()函数调整参数值直到找到使目标函数最小化的最佳参数估计值。

1. Using nlm(), and the mse() function you wrote, write a function, plm(), which estimates the parameters and of the model by minimizing the mean squared error. It should take the following arguments: an initial guess for ; an initial guess for ; a vector containing the values; a vector containing the 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 ; the final guess for ; 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 and ? From and ? If these are not the same, why do they differ? Which estimate has the lower MSE?

#最小化均方误差来估算模型的参数  
plm <- function(y0\_init, a\_init, N, Y) {  
 minimize\_mse <- function(parameters) {  
 y0 <- parameters[1]  
 a <- parameters[2]  
 mse\_val <- mse(params = c(y0, a), N = N, Y = Y)  
 return(mse\_val)  
 }  
   
 initial\_guess <- c(y0\_init, a\_init)  
 result <- nlm(minimize\_mse, p = initial\_guess)  
   
 y\_final <- result$estimate[1]  
 a\_final <- result$estimate[2]  
 mse\_final <- result$minimum  
   
 return(list(y0\_final = y\_final, a\_final = a\_final, MSE = mse\_final))  
}

result1 <- plm(y0\_init = 6611, a\_init = 0.15, N = gmp$pop, Y = gmp$pcgmp)  
result1$y0\_final # Final estimate for y0

## [1] 6611

result1$a\_final # Final estimate for a

## [1] 0.1263182

result1$MSE # Final MSE value

## [1] 61857060

result2 <- plm(y0\_init = 5000, a\_init = 0.10, N = gmp$pop, Y = gmp$pcgmp)

## Warning in nlm(minimize\_mse, p = initial\_guess): NA/Inf replaced by maximum  
## positive value  
  
## Warning in nlm(minimize\_mse, p = initial\_guess): NA/Inf replaced by maximum  
## positive value  
  
## Warning in nlm(minimize\_mse, p = initial\_guess): NA/Inf replaced by maximum  
## positive value  
  
## Warning in nlm(minimize\_mse, p = initial\_guess): NA/Inf replaced by maximum  
## positive value  
  
## Warning in nlm(minimize\_mse, p = initial\_guess): NA/Inf replaced by maximum  
## positive value  
  
## Warning in nlm(minimize\_mse, p = initial\_guess): NA/Inf replaced by maximum  
## positive value

result2$y0\_final # Final estimate for y0

## [1] 5000

result2$a\_final # Final estimate for a

## [1] 0.1475913

result2$MSE # Final MSE value

## [1] 62521484

不同初始值导致估计值不同的原因在于优化过程可能会收敛到不同的局部极小值，而初始值的选择将影响最终的结果。具有较低均方误差的估计值被认为更好地拟合了数据。第一组数据的均方误差较低。

1. *Convince yourself the jackknife can work*.
   1. 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…).

#使用mean()函数计算人均GMP的均值：  
per\_capita\_gmp <- gmp$pcgmp  
mean\_gmp <- mean(per\_capita\_gmp)  
  
#使用sd()函数计算人均GMP的标准差：  
sd\_gmp <- sd(per\_capita\_gmp)  
  
#计算样本大小（城市数量）：  
n <- length(per\_capita\_gmp)  
  
#使用公式计算均值的标准误差：标准差除以样本大小的平方根：  
se\_mean <- sd\_gmp / sqrt(n)

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

calculate\_mean\_except\_i <- function(i) {  
 mean(per\_capita\_gmp[-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()`.)

#使用for循环：  
jackknifed.means <- numeric(length(per\_capita\_gmp))  
for (i in 1:length(per\_capita\_gmp)) {  
 jackknifed.means[i] <- calculate\_mean\_except\_i(i)  
}  
  
#使用sapply()函数：  
jackknifed.means <- sapply(1:length(per\_capita\_gmp), calculate\_mean\_except\_i)

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)?

jackknife\_se <- sqrt((n - 1) / n \* sum((jackknifed.means - mean\_gmp)^2))  
print(jackknife\_se)

## [1] 481.9195

print(se\_mean)

## [1] 481.9195

#完全吻合

1. Write a function, plm.jackknife(), to calculate jackknife standard errors for the parameters and . 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?

#最小化均方误差来估算模型的参数  
plm <- function(y0\_init, a\_init, N, Y) {  
 minimize\_mse <- function(parameters) {  
 y0 <- parameters[1]  
 a <- parameters[2]  
 mse\_val <- mse(params = c(y0, a), N = N, Y = Y)  
 return(mse\_val)  
 }  
   
 initial\_guess <- c(y0\_init, a\_init)  
 result <- nlm(minimize\_mse, p = initial\_guess)  
   
 y\_final <- result$estimate[1]  
 a\_final <- result$estimate[2]  
 mse\_final <- result$minimum  
   
 return(list(y0\_final = y\_final, a\_final = a\_final, MSE = mse\_final))  
}

plm.jackknife <- function(y0\_init, a\_init, N, Y) {  
 n <- length(N) # 观测数量  
  
 # 初始化用于存储参数估计的向量  
 y0\_estimates <- numeric(n)  
 a\_estimates <- numeric(n)  
   
 y0 <- y0\_init  
 a <- a\_init  
 # jackknife过程  
 for (i in 1:n) {  
 # 略去第 i 个观测  
   
 # 使用 plm() 估计参数  
 plm\_result <- plm(y0, a, N[-i], Y[-i])  
  
 # 存储参数估计  
 y0\_estimates[i] <- plm\_result$y0\_final  
 a\_estimates[i] <- plm\_result$a\_final  
 }  
  
 # 计算jackknife标准误差  
 y0\_se <- sqrt((n - 1) / n \* sum((y0\_estimates - mean(y0\_estimates))^2))  
 a\_se <- sqrt((n - 1) / n \* sum((a\_estimates - mean(a\_estimates))^2))  
  
 return(list(y0\_se = y0\_se, a\_se = a\_se))  
}

1. 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?

#Step 1: 从文件 "gmp-2013.dat" 中加载数据集：  
data\_2013 <- read.table("./mynotes/data/gmp-2013.dat", header = TRUE)  
#Step 2: 从加载的数据集中提取所需变量：  
N <- data\_2013$gmp  
Y <- data\_2013$pcgmp  
  
  
#Step 3: 初始化模型参数的初始值：  
  
y0\_init <- 5000 # 用您需要的初始值替换  
a\_init <- 0.15 # 用您需要的初始值替换  
#请为参数提供适当的初始值。  
  
#Step 4: 使用 plm() 函数估计2013年模型的参数：  
  
plm\_result <- plm(y0\_init, a\_init, N, Y)  
#这将估计参数并返回它们的值和均方误差（MSE）。  
  
#Step 5: 使用 plm.jackknife() 函数估计2013年参数的标准误差：  
  
plm\_jack\_result <- plm.jackknife(y0\_init, a\_init, N, Y)  
#使用删除样本方法估计参数的标准误差，并返回它们的值。  
  
#Step 6: 访问2013年估计的参数值及其标准误差：  
  
  
y0\_2013 <- plm\_result$y0\_final  
a\_2013 <- plm\_result$a\_final  
y0\_se\_2013 <- plm\_jack\_result$y0\_se  
a\_se\_2013 <- plm\_jack\_result$a\_se  
  
print(y0\_2013)

## [1] 4999.999

print(a\_2013)

## [1] -109.7309

print(y0\_se\_2013)

## [1] 4.03446e-06

print(a\_se\_2013)

## [1] 0.0001279537