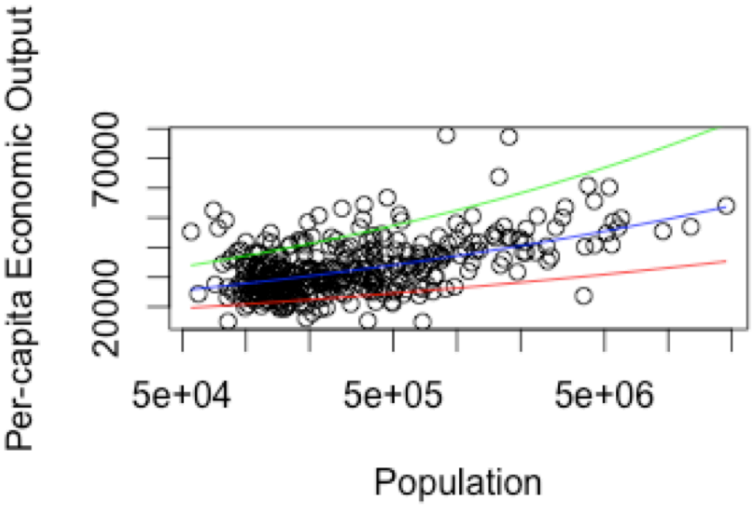
---
title: "HW 4"
author: "Autumn Li and UNI ql2280"
date: "Oct 25, 2016"
output: html_document

```{r}
gmp <- read.table("~/Desktop/2016 fall/5206/hw4/gmp.txt", header=TRUE, quote="\"")
gmp$pop <- round(gmp$gmp/gmp$pcgmp)
names(gmp)
plot(gmp$pop, gmp$pcgmp, log = "x", xlab = "Population",ylab = "Per-capita Economic Output")
beta_0 = 6611
beta_1 = 1/8
curve(6611*x^{1/8}, add = TRUE, col = "blue")
curve(6611*x^{0.1},add = TRUE,col = "red")
curve(6611*x^{0.15},add = TRUE,col = "green")
```



ii. Write a function called mse() which calculates the mean squared error if the model on a given dataset.

```{r}
v = c(6611,0.15)
x = gmp$pop
y = gmp$pcgmp
mse1 <- function(v,x=gmp$pop,y=gmp$pcgmp) {
  value = sum((y-x^v[2]*v[1])^2)/length(y)
  return(value)
  }

mse1(c(6611,0.15))
```

mse1(c(5000, 0.10))

```
> mse1(c(6611,0.15))
[1] 207057513
> mse1(c(5000, 0.10))
[1] 298459914
```

iii. R has several built-in functions for optimization which we'll talk about later on in the course.

```{r, warning=FALSE}
nlm(mse1, c(beta0 = 6611, beta1 = 1/8))
nlm(mse1, c(beta0 = 6611, beta1 = 0.1))
nlm(mse1, c(beta0 = 6611, beta1 = 0.15))
nlm(mse1, c(beta0 = 6611, beta1 = 1/8))$estimate
nlm(mse1, c(beta0 = 6611, beta1 = 0.1))$estimate
nlm(mse1, c(beta0 = 6611, beta1 = 0.15))$estimate
```

```
> nlm(mse1, c(beta0 = 6611, beta1 = 0.15))$estimate
[1] 6610.9999997    0.1263182
```

```
# minimum represents the value of the estimated minimum of function.
# estimate represents the point at which the minimum value of function is obtained.
#
```

iv. Using nlm() and the mse() function you wrote, write a function plm() which estimates the parameters β0 and β1 of the model by minimizing the mean squared error.

```{r}
beta0 = 6611
beta1 = 0.15
plm = function(beta0,beta1,x=gmp$pop,y=gmp$pcgmp){
  v1 = c(beta0,beta1)
  l3 = mse1(v1,x,y)
  l1 = nlm(mse1, c(beta0, beta1))$estimate[1]
  l2 = nlm(mse1, c(beta0, beta1))$estimate[2]
  value1 = list(l1,l2,l3)
  return(value1)
  }

plm(6611,0.15)
plm(5000,0.1)
```

```
> plm(6611,0.15)
[[1]]
[1] 6611

[[2]]
[1] 0.1263182

[[3]]
```

[1] 207057513

> plm(5000,0.1)
[[1]]
[1] 5000

[[2]]
[1] 0.1475913

[[3]]
[1] 298459914

plm(6611,0.15) has smaller MSE
```
#v. Let's practice the bootstrap in a simple example to convince ourselves, again, that it will work.
(a)
```{r}
mean(y)
sd(y)
```
> mean(y)
[1] 32922.53
> sd(y)
[1] 9219.663

(b)
```{r}
v3 = c(1:length(y))
index.mean = function(v3){
  value3 = mean(y[v3])
  return(value3)
  }
index.mean(c(1,5,1,3))
```
(c) Using the function in (b) create a vector bootstrap.means, which has the mean per-capita GMP for one hundred bootstrap samples.
```{r}
dat = y[sample(100)]
nboot = 100
bootstrap.mean <- function(dat, nboot) {
  bootstat <- NULL
  for(i in 1:nboot) {
    truemean <- mean(dat)
    samp <- sample(nboot, replace = T)
    samp.mean = index.mean(samp)
    bootstat[i] = samp.mean
    }
  return(bootstat)
```
```

```
  }

bootstrap.mean(dat,nboot)
```
(d) Calculate the standard deviation of the bootstrap.means to approximate the standard error from part (a).
```{r}
sd(bootstrap.mean(dat,nboot))

> sd(bootstrap.mean(dat,nboot))
[1] 1052.133

#Compares to the part a, part d has a lager standard deviation.
```

```{r}
B = 100
plm.bootstrap = function (beta0,beta1,B,x = gmp$pop,y = gmp$pcgmp) {
  est.beta0 = matrix(NA,nrow = B, ncol = 1)
  est.beta1 = matrix(NA,nrow = B, ncol = 1)
  for( b in 1:B)
    {
     resamp <- sample(1:length(y),size = length(y),replace = T)
     y1=y[resamp]
     x1=x[resamp]
    l1 = plm(beta0, beta1, x1, y1)[[1]]
    l2 = plm(beta0, beta1, x1, y1)[[2]]
    est.beta0[b] = l1
    est.beta1[b] = l2

  }
   est.beta  = list(c(sd(est.beta0)/sqrt(B),sd(est.beta1)/sqrt(B)))
   return(est.beta)
}
plm.bootstrap(beta0=6611,beta1=0.15,B=100)
plm.bootstrap(5000,0.1,100)

> plm.bootstrap(beta0=6611,beta1=0.15,B=100)
[[1]]
[1] 0 0

> plm.bootstrap(5000,0.1,100)
[[1]]
[1] 0 0

# To whom it may concern:
#I know the answer should not be 0. The reason is nlm function generates same value dedpite the different sample size. I think my bootstrap function is right as long as my nlm function could fix above problem.
```

```
vii. The file gmp-2013.txt contains measurements for 2013 (in contrast to measurements from 2006 in gmp.txt).
```{r}
gmp.2013 <- read.table("~/Desktop/2016 fall/5206/hw4/gmp-2013.txt", header=TRUE, quote="\"")
x2 = gmp$pop
y2 = gmp.2013$pcgmp
plm(6611,0.15,x2,y2)
plm(5000,0.1,x2,y2)
plm.bootstrap(611,0.15,100)
plm.bootstrap(5000,0.1,100)

> plm(6611,0.15,x2,y2)
[[1]]
[1] 6611

[[2]]
[1] 0.1263182

[[3]]
[1] 217878913

> plm(5000,0.1,x2,y2)
[[1]]
[1] 5000

[[2]]
[1] 0.1475913

[[3]]
[1] 725717454

[[1]]
[1] 0 0

[[1]]
[1] 0 0



# To whom it may concern:
#The answer above is wrong, because the plm function should return different numbers for different sample size from gmp 2013.
# The reason is nlm function generates same value dedpite the different sample size. I think my plm function and plm.boostrap function are right as long as  nlm function could fix above problem.
```
```