LINEAR REGRESSION

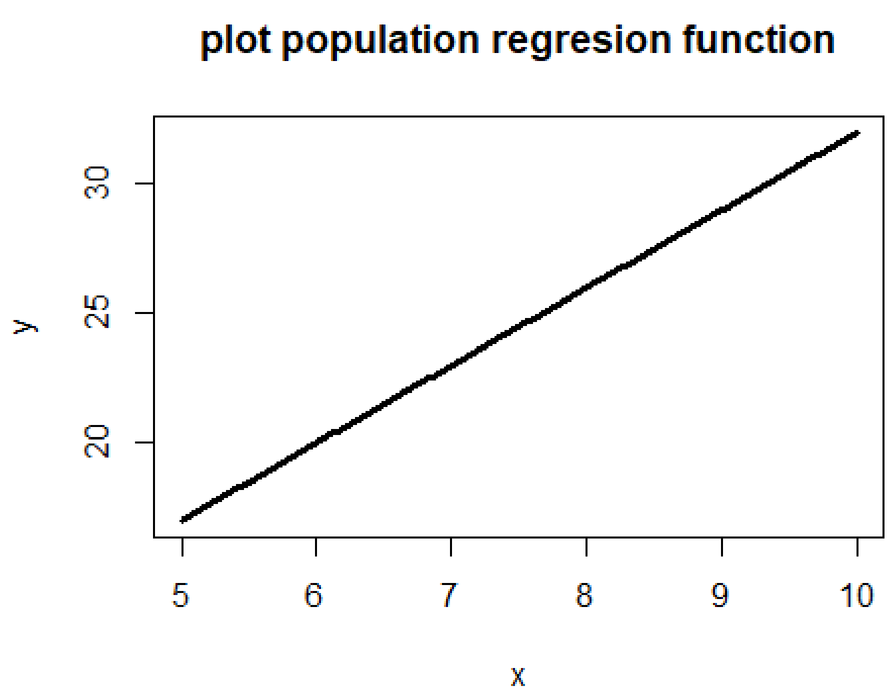
DEBKANTA GHOSH

2026-02-04

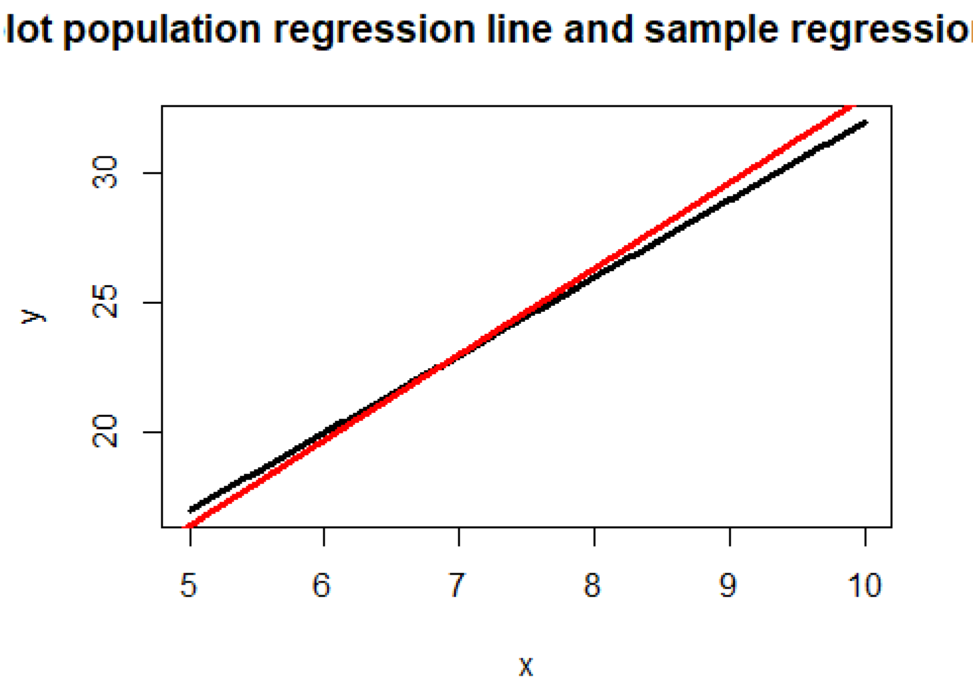
1.Problem to demonstrate that the population regression line is fixed, but least square regression line varies.

```
rm(list=ls())

#STEP1:TAKE EQUIDISTANT POINT FROM 5 TO 10

x=seq(5,10,length.out=200)
x

##   [1]  5.000000  5.025126  5.050251  5.075377  5.100503  5.125628  5.15075
4
##   [8]  5.175879  5.201005  5.226131  5.251256  5.276382  5.301508  5.32663
3
##  [15]  5.351759  5.376884  5.402010  5.427136  5.452261  5.477387  5.50251
3
##  [22]  5.527638  5.552764  5.577889  5.603015  5.628141  5.653266  5.67839
2
##  [29]  5.703518  5.728643  5.753769  5.778894  5.804020  5.829146  5.85427
1
##  [36]  5.879397  5.904523  5.929648  5.954774  5.979899  6.005025  6.03015
1
##  [43]  6.055276  6.080402  6.105528  6.130653  6.155779  6.180905  6.20603
0
##  [50]  6.231156  6.256281  6.281407  6.306533  6.331658  6.356784  6.38191
0
##  [57]  6.407035  6.432161  6.457286  6.482412  6.507538  6.532663  6.55778
9
##  [64]  6.582915  6.608040  6.633166  6.658291  6.683417  6.708543  6.73366
8
##  [71]  6.758794  6.783920  6.809045  6.834171  6.859296  6.884422  6.90954
8
##  [78]  6.934673  6.959799  6.984925  7.010050  7.035176  7.060302  7.08542
7
##  [85]  7.110553  7.135678  7.160804  7.185930  7.211055  7.236181  7.26130
7
##  [92]  7.286432  7.311558  7.336683  7.361809  7.386935  7.412060  7.43718
6
##  [99]  7.462312  7.487437  7.512563  7.537688  7.562814  7.587940  7.61306
5
## [106]  7.638191  7.663317  7.688442  7.713568  7.738693  7.763819  7.78894
5
## [113]  7.814070  7.839196  7.864322  7.889447  7.914573  7.939698  7.96482
4
```

```
## [120]  7.989950  8.015075  8.040201  8.065327  8.090452  8.115578  8.14070
4
## [127]  8.165829  8.190955  8.216080  8.241206  8.266332  8.291457  8.31658
3
## [134]  8.341709  8.366834  8.391960  8.417085  8.442211  8.467337  8.49246
2
## [141]  8.517588  8.542714  8.567839  8.592965  8.618090  8.643216  8.66834
2
## [148]  8.693467  8.718593  8.743719  8.768844  8.793970  8.819095  8.84422
1
## [155]  8.869347  8.894472  8.919598  8.944724  8.969849  8.994975  9.02010
1
## [162]  9.045226  9.070352  9.095477  9.120603  9.145729  9.170854  9.19598
0
## [169]  9.221106  9.246231  9.271357  9.296482  9.321608  9.346734  9.37185
9
## [176]  9.396985  9.422111  9.447236  9.472362  9.497487  9.522613  9.54773
9
## [183]  9.572864  9.597990  9.623116  9.648241  9.673367  9.698492  9.72361
8
## [190]  9.748744  9.773869  9.798995  9.824121  9.849246  9.874372  9.89949
7
## [197]  9.924623  9.949749  9.974874 10.000000

y=2+3*x
plot(x,y,type='l',lwd=3,main="plot population regresion function")
```



```
#step 2:Generate xi(i = 1, 2, .., n) from Uniform(5, 10) and ϵi(i = 1, 2, ..,
n)from N(0, 42). Hence, compute y1, y2, .., yn.

set.seed(123)
n=50
xi=runif(n,5,10)
ei=rnorm(n,0,4)
yi=2+3*xi+ei

#step3: Least square regression line
#two plot in same graph
plot(x,y,type='l',lwd=3,main="plot population regression line and sample regr
ession line")
lin.reg=lm(yi~xi)
coef(lin.reg)

## (Intercept)          xi
## -0.09638929  3.30539569

abline(lin.reg,col="red",lwd=3)
```

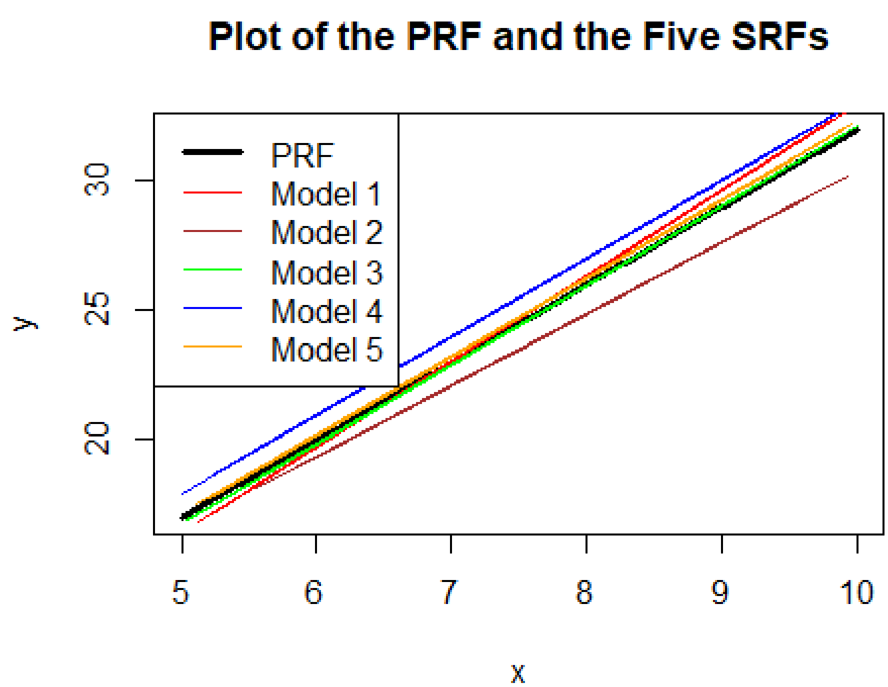



```
#Step 4: Repeat steps 2-3 five times. Graph the 5 least squares regression li
nes over the population regression line obtained in Step 1.
set.seed(123)
x1=runif(n,5,10)
e1=rnorm(n,0,4)
y1=2+3*x1+e1
```

```r
fit1=lm(y1~x1)
summary(fit1)

##
## Call:
## lm(formula = y1 ~ x1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0231 -2.2314 -0.2627  2.1970  8.7445
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.09639    2.82610  -0.034    0.973
## x1           3.30540    0.36519   9.051 5.96e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.761 on 48 degrees of freedom
## Multiple R-squared:  0.6306, Adjusted R-squared:  0.6229
## F-statistic: 81.93 on 1 and 48 DF,  p-value: 5.962e-12

x2=runif(n,5,10)
e2=rnorm(n,0,4)
y2=2+3*x2+e2
fit2=lm(y2~x2)

x3=runif(n,5,10)
e3=rnorm(n,0,4)
y3=2+3*x3+e3
fit3=lm(y3~x3)

x4=runif(n,5,10)
e4=rnorm(n,0,4)
y4=2+3*x4+e4
fit4=lm(y4~x4)

x5=runif(n,5,10)
e5=rnorm(n,0,4)
y5=2+3*x5+e5
fit5=lm(y5~x5)

#Data frame containing the five models' coefficients
coeff=data.frame(coef(fit1),
                 coef(fit2),
                 coef(fit3),
                 coef(fit4),
                 coef(fit5))
coeff
```

```
##                 coef.fit1. coef.fit2. coef.fit3. coef.fit4. coef.fit5.
## (Intercept) -0.09638929   2.792188   1.392997   2.823089   2.032506
## x1           3.30539569   2.761042   3.073267   3.023608   3.028097
```

```r
#Plot of the PRF and the Five SRFs

plot(x,y,type='l',lwd=3,main="Plot of the PRF and the Five SRFs")
lines(x1,predict(fit1),type='l',col="red")
lines(x2,predict(fit2),type='l',col="brown")
lines(x3,predict(fit3),type='l',col="green")
lines(x4,predict(fit4),type='l',col="blue")
lines(x5,predict(fit5),type='l',col="orange")
legend("topleft",legend=c("PRF","Model 1","Model 2","Model 3",
                          "Model 4","Model 5"),
       col=c("black","red","brown","green","blue","orange"),
       lwd=c(3,1,1,1,1,1))
```




```r
coeff
```

```
##                 coef.fit1. coef.fit2. coef.fit3. coef.fit4. coef.fit5.
## (Intercept) -0.09638929   2.792188   1.392997   2.823089   2.032506
## x1           3.30539569   2.761042   3.073267   3.023608   3.028097
```

```r
#Interpretation: PRF is fixed but SRF varies
```

Problem 2 Demonstrate that the estimators $\hat{\beta}_0$ and $\hat{\beta}_1$ minimize the Residual Sum of Squares (RSS).

```r
rm(list=ls())

#step 1: Generate xi~U(5,10) of size 50, and ei~N(0,1)
#y=2+3x+e
n=50
set.seed(123)
x=runif(n,5,10)
xm=x-mean(x)
e=rnorm(n)
y=2+3*xm+e
fit=lm(y~xm)
summary(fit)

##
## Call:
## lm(formula = y ~ xm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.25578 -0.55786 -0.06567  0.54926  2.18613
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.0562     0.1330   15.46   <2e-16 ***
## xm            3.0764     0.0913   33.70   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9404 on 48 degrees of freedom
## Multiple R-squared:  0.9594, Adjusted R-squared:  0.9586
## F-statistic:  1135 on 1 and 48 DF,  p-value: < 2.2e-16

beta0=2.0562
beta=3.0764

#Creating the beta and beta0 grid values
beta0.grid=seq(-1,1,length.out=101)+beta0
beta.grid=seq(-1,1,length.out=101)+beta
#Computing RSS
RSS=c()
for(i in 1:length(beta.grid))
{
  RSS[i]=sum((y-beta0.grid[i]-beta.grid[i]*xm)^2)
}
RSS

##   [1] 198.52541 192.34441 186.28828 180.35703 174.55065 168.86915 163.3125
2
##   [8] 157.88076 152.57388 147.39187 142.33473 137.40247 132.59508 127.9125
6
##  [15] 123.35492 118.92215 114.61425 110.43123 106.37308 102.43981  98.6314
```

```
0
##  [22]  94.94788  91.38922  87.95544  84.64653  81.46249  78.40333  75.4690
5
##  [29]  72.65963  69.97509  67.41542  64.98063  62.67071  60.48566  58.4254
9
##  [36]  56.49019  54.67976  52.99421  51.43353  49.99772  48.68679  47.5007
3
##  [43]  46.43954  45.50323  44.69179  44.00522  43.44353  43.00671  42.6947
7
##  [50]  42.50769  42.44550  42.50817  42.69572  43.00814  43.44544  44.0076
1
##  [57]  44.69465  45.50656  46.44335  47.50502  48.69155  50.00296  51.4392
5
##  [64]  53.00040  54.68643  56.49734  58.43311  60.49376  62.67929  64.9896
8
##  [71]  67.42495  69.98510  72.67012  75.48001  78.41477  81.47441  84.6589
2
##  [78]  87.96831  91.40257  94.96170  98.64570 102.45458 106.38833 110.4469
6
##  [85] 114.63046 118.93883 123.37208 127.93020 132.61319 137.42106 142.3538
0
##  [92] 147.41141 152.59390 157.90126 163.33349 168.89060 174.57258 180.3794
3
##  [99] 186.31116 192.36776 198.54924
```

```r
df=data.frame(beta0.grid,beta.grid,RSS)
df
```

```
##     beta0.grid beta.grid       RSS
## 1       1.0562    2.0764 198.52541
## 2       1.0762    2.0964 192.34441
## 3       1.0962    2.1164 186.28828
## 4       1.1162    2.1364 180.35703
## 5       1.1362    2.1564 174.55065
## 6       1.1562    2.1764 168.86915
## 7       1.1762    2.1964 163.31252
## 8       1.1962    2.2164 157.88076
## 9       1.2162    2.2364 152.57388
## 10      1.2362    2.2564 147.39187
## 11      1.2562    2.2764 142.33473
## 12      1.2762    2.2964 137.40247
## 13      1.2962    2.3164 132.59508
## 14      1.3162    2.3364 127.91256
## 15      1.3362    2.3564 123.35492
## 16      1.3562    2.3764 118.92215
## 17      1.3762    2.3964 114.61425
## 18      1.3962    2.4164 110.43123
## 19      1.4162    2.4364 106.37308
## 20      1.4362    2.4564 102.43981
## 21      1.4562    2.4764  98.63140
```

```
## 22       1.4762    2.4964  94.94788
## 23       1.4962    2.5164  91.38922
## 24       1.5162    2.5364  87.95544
## 25       1.5362    2.5564  84.64653
## 26       1.5562    2.5764  81.46249
## 27       1.5762    2.5964  78.40333
## 28       1.5962    2.6164  75.46905
## 29       1.6162    2.6364  72.65963
## 30       1.6362    2.6564  69.97509
## 31       1.6562    2.6764  67.41542
## 32       1.6762    2.6964  64.98063
## 33       1.6962    2.7164  62.67071
## 34       1.7162    2.7364  60.48566
## 35       1.7362    2.7564  58.42549
## 36       1.7562    2.7764  56.49019
## 37       1.7762    2.7964  54.67976
## 38       1.7962    2.8164  52.99421
## 39       1.8162    2.8364  51.43353
## 40       1.8362    2.8564  49.99772
## 41       1.8562    2.8764  48.68679
## 42       1.8762    2.8964  47.50073
## 43       1.8962    2.9164  46.43954
## 44       1.9162    2.9364  45.50323
## 45       1.9362    2.9564  44.69179
## 46       1.9562    2.9764  44.00522
## 47       1.9762    2.9964  43.44353
## 48       1.9962    3.0164  43.00671
## 49       2.0162    3.0364  42.69477
## 50       2.0362    3.0564  42.50769
## 51       2.0562    3.0764  42.44550
## 52       2.0762    3.0964  42.50817
## 53       2.0962    3.1164  42.69572
## 54       2.1162    3.1364  43.00814
## 55       2.1362    3.1564  43.44544
## 56       2.1562    3.1764  44.00761
## 57       2.1762    3.1964  44.69465
## 58       2.1962    3.2164  45.50656
## 59       2.2162    3.2364  46.44335
## 60       2.2362    3.2564  47.50502
## 61       2.2562    3.2764  48.69155
## 62       2.2762    3.2964  50.00296
## 63       2.2962    3.3164  51.43925
## 64       2.3162    3.3364  53.00040
## 65       2.3362    3.3564  54.68643
## 66       2.3562    3.3764  56.49734
## 67       2.3762    3.3964  58.43311
## 68       2.3962    3.4164  60.49376
## 69       2.4162    3.4364  62.67929
## 70       2.4362    3.4564  64.98968
## 71       2.4562    3.4764  67.42495
```

```
## 72       2.4762    3.4964  69.98510
## 73       2.4962    3.5164  72.67012
## 74       2.5162    3.5364  75.48001
## 75       2.5362    3.5564  78.41477
## 76       2.5562    3.5764  81.47441
## 77       2.5762    3.5964  84.65892
## 78       2.5962    3.6164  87.96831
## 79       2.6162    3.6364  91.40257
## 80       2.6362    3.6564  94.96170
## 81       2.6562    3.6764  98.64570
## 82       2.6762    3.6964 102.45458
## 83       2.6962    3.7164 106.38833
## 84       2.7162    3.7364 110.44696
## 85       2.7362    3.7564 114.63046
## 86       2.7562    3.7764 118.93883
## 87       2.7762    3.7964 123.37208
## 88       2.7962    3.8164 127.93020
## 89       2.8162    3.8364 132.61319
## 90       2.8362    3.8564 137.42106
## 91       2.8562    3.8764 142.35380
## 92       2.8762    3.8964 147.41141
## 93       2.8962    3.9164 152.59390
## 94       2.9162    3.9364 157.90126
## 95       2.9362    3.9564 163.33349
## 96       2.9562    3.9764 168.89060
## 97       2.9762    3.9964 174.57258
## 98       2.9962    4.0164 180.37943
## 99       3.0162    4.0364 186.31116
## 100      3.0362    4.0564 192.36776
## 101      3.0562    4.0764 198.54924

df[which.min(RSS),]

##    beta0.grid beta.grid     RSS
## 51     2.0562    3.0764 42.4455

#We can verify the LSE minimises the RSS
```

3.Problem to demonstrate that least square estimators are unbiased

```
rm(list=ls())

#Step 1: Generate xi~U(5,10) of size 50, and ei~N(0,1)
n=50
set.seed(123)
x=runif(n,0,1)
e=rnorm(n)
y=2+3*x+e
```

```r
#Step 2: obtain LSE
fit=lm(y~x)
summary(fit)

##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.25578 -0.55786 -0.06567  0.54926  2.18613
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.8576     0.2721   6.827 1.36e-08 ***
## x             3.3817     0.4565   7.408 1.75e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9404 on 48 degrees of freedom
## Multiple R-squared:  0.5334, Adjusted R-squared:  0.5237
## F-statistic: 54.88 on 1 and 48 DF,  p-value: 1.745e-09

beta0.hat=1.8576
beta.hat=3.3817
#Repeat R=1000 times
set.seed(123)
R=10000
beta0=numeric(R)
beta=numeric(R)
n=50
for(i in 1:R) {
  x=runif(n,0,1)
  e=rnorm(n,0,1)
  y=2+3*x+e
  fit=lm(y~x)
  beta0[i]=coef(fit)[1]
  beta[i]=coef(fit)[2]
}
mean(beta0)

## [1] 2.005203

mean(beta)

## [1] 2.991338

#comment: the long run means of beta0.hat and beta hat converges to true beta
and beta0

var(beta0)
```

```
## [1] 0.08292927

var(beta)

## [1] 0.2495031

#Comment: var approaches 0 as R goes up. Hence LSE is consistent.
```

4.Comparing several simple linear regressions Attach “Boston” data from MASS library in R. Select median value of owneroccupied

```
rm(list=ls())

# Load required library and data
library(MASS)
data(Boston)
# (a) Run separate simple linear regressions
model_crim =lm(medv ~ crim, data = Boston)
model_nox =lm(medv ~ nox, data = Boston)
model_black=lm(medv ~ black, data = Boston)
model_lstat=lm(medv ~ lstat, data = Boston)

# Present output in a single table
summary_table=data.frame(
  Predictor = c("crim", "nox", "black", "lstat"),
  Coefficient = c(
    coef(model_crim)[2],
    coef(model_nox)[2],
    coef(model_black)[2],
    coef(model_lstat)[2]
  ),
  R_squared = c(
    summary(model_crim)$r.squared,
    summary(model_nox)$r.squared,
    summary(model_black)$r.squared,
    summary(model_lstat)$r.squared
  )
)

summary_table

##        Predictor  Coefficient R_squared
## crim        crim  -0.41519028 0.1507805
## nox          nox -33.91605501 0.1826030
## black      black   0.03359306 0.1111961
## lstat      lstat  -0.95004935 0.5441463

# (b) Identify the best model based on R-squared
best_model=summary_table[which.max(summary_table$R_squared), ]
best_model
```

```
##       Predictor Coefficient R_squared
## lstat     lstat  -0.9500494 0.5441463
```

```r
# (c) Compare coefficients and usefulness of predictors
summary(model_crim)
```

```
## 
## Call:
## lm(formula = medv ~ crim, data = Boston)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -16.957  -5.449  -2.007   2.512  29.800 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 24.03311    0.40914   58.74   <2e-16 ***
## crim        -0.41519    0.04389   -9.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 8.484 on 504 degrees of freedom
## Multiple R-squared:  0.1508, Adjusted R-squared:  0.1491 
## F-statistic: 89.49 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(model_nox)
```

```
## 
## Call:
## lm(formula = medv ~ nox, data = Boston)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -13.691  -5.121  -2.161   2.959  31.310 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)   41.346      1.811   22.83   <2e-16 ***
## nox          -33.916      3.196  -10.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 8.323 on 504 degrees of freedom
## Multiple R-squared:  0.1826, Adjusted R-squared:  0.181 
## F-statistic: 112.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

```r
summary(model_black)
```

```
## 
## Call:
## lm(formula = medv ~ black, data = Boston)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.884  -4.862  -1.684   2.932  27.763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.551034   1.557463   6.775 3.49e-11 ***
## black        0.033593   0.004231   7.941 1.32e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.679 on 504 degrees of freedom
## Multiple R-squared:  0.1112, Adjusted R-squared:  0.1094
## F-statistic: 63.05 on 1 and 504 DF,  p-value: 1.318e-14
```

```r
summary(model_lstat)
```

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```