Problem Set - 2

Arijit Das, Roll - 713

2026-02-02

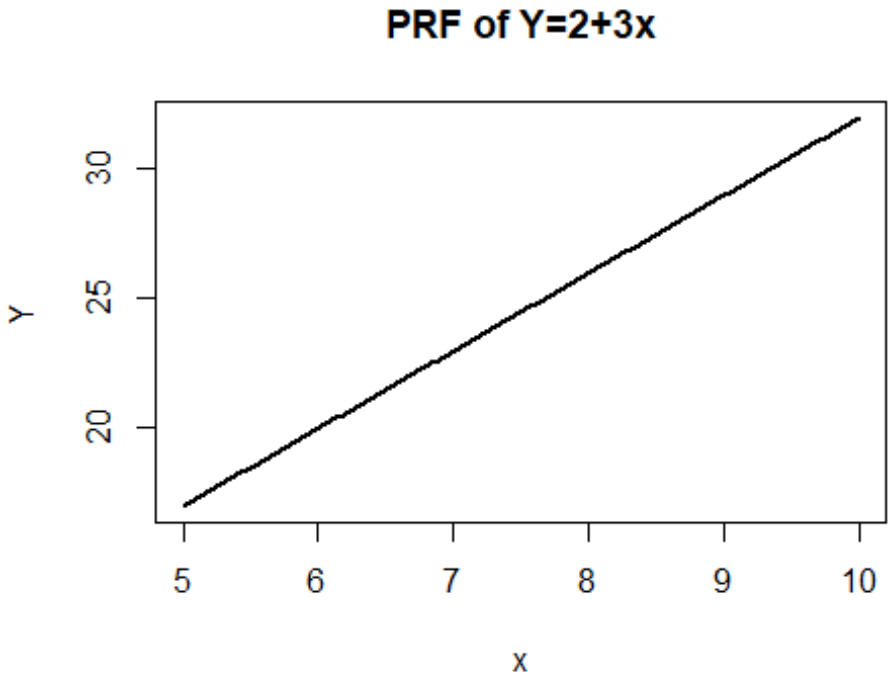
Linear Regression

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

Suppose the population regression line is given by Y = 2 + 3x, while the data comes from the model y = 2 + 3x + ϵ.

Step 1: *For x in the range [5,10] graph the population regression line.*

```
rm(list=ls())
x=seq(5,10,length.out=200)
y=2+3*x
plot(x,y,type='l',lwd=2,main="PRF of Y=2+3x",xlab = "x",ylab = "Y")
```



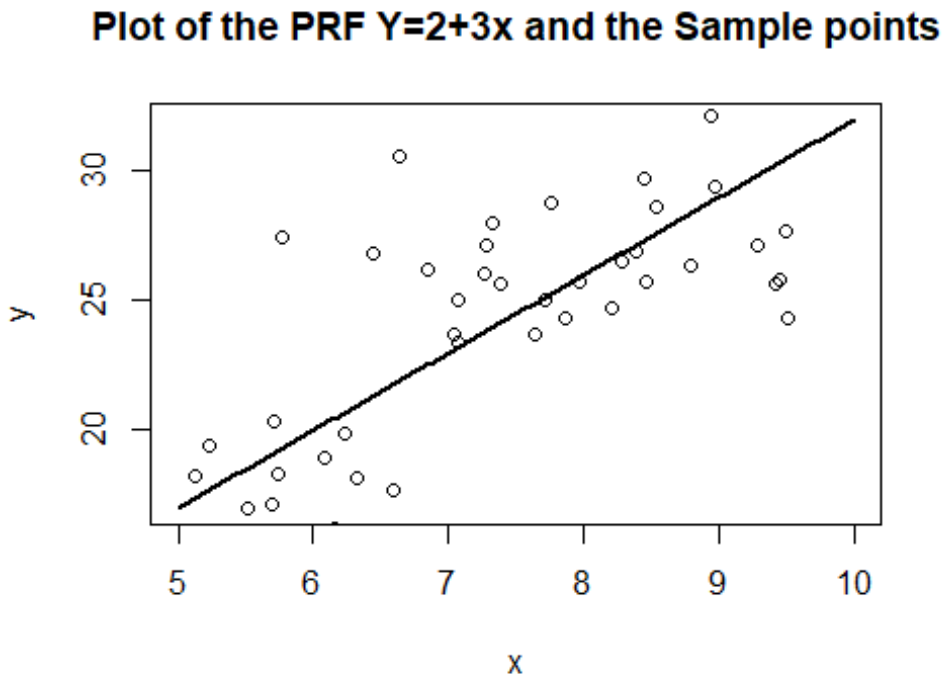
Step 2: *Generate xi(i = 1, 2, .., n) from Uniform(5, 10) and ϵi(i = 1, 2, .., n) from N(0, 4^2). Hence, compute y1, y2, .., yn.*

```
set.seed(123)
n=50
xi=runif(n,5,10)
```

```r
ei=rnorm(n,0,4)
yi=2+3*xi+ei
head(data.frame(xi,ei,yi))
```

```
##         xi         ei       yi
## 1 6.437888 -6.7467732 14.56689
## 2 8.941526  3.3511482 32.17573
## 3 7.044885  0.6134925 23.74815
## 4 9.415087 -4.5525477 25.69271
## 5 9.702336  5.0152597 36.12227
## 6 5.227782  1.7058569 19.38920
```

```r
plot(x,y,type='l',lwd=2,main="Plot of the PRF Y=2+3x and the Sample points")
points(xi,yi)
```



Step 3: *On the basis of the data (xi, yi)(i = 1, 2, .., n) generated in Step 2, report the least squares regression line.*

```r
model=lm(yi~xi)
summary(model)
```

```
##
## Call:
## lm(formula = yi ~ xi)
##
## 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
## xi           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
```

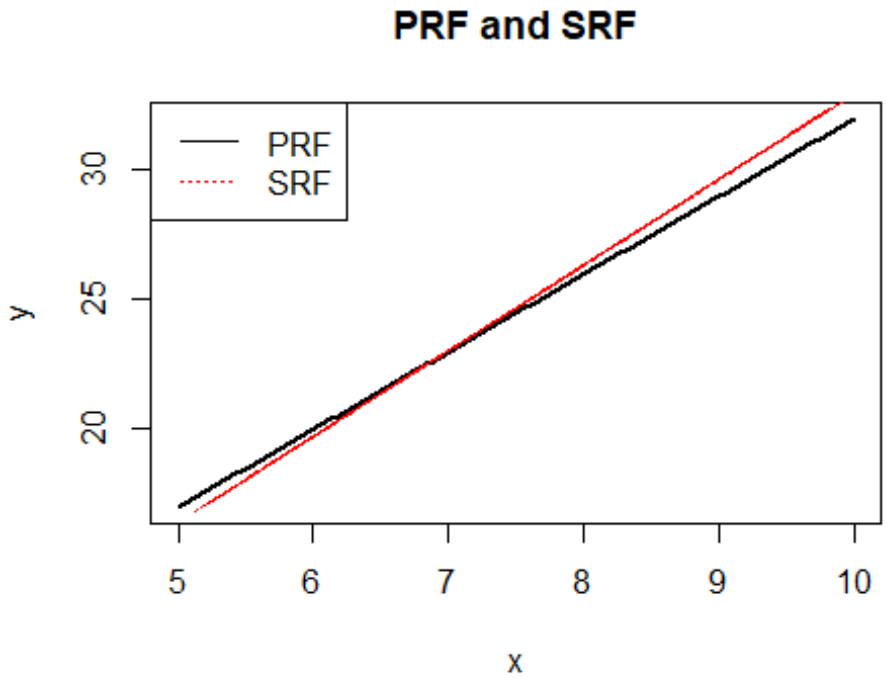
```
b0=coef(model)[1];b0
```

```
## (Intercept)
## -0.09638929
```

```
b1=coef(model)[2];b1
```

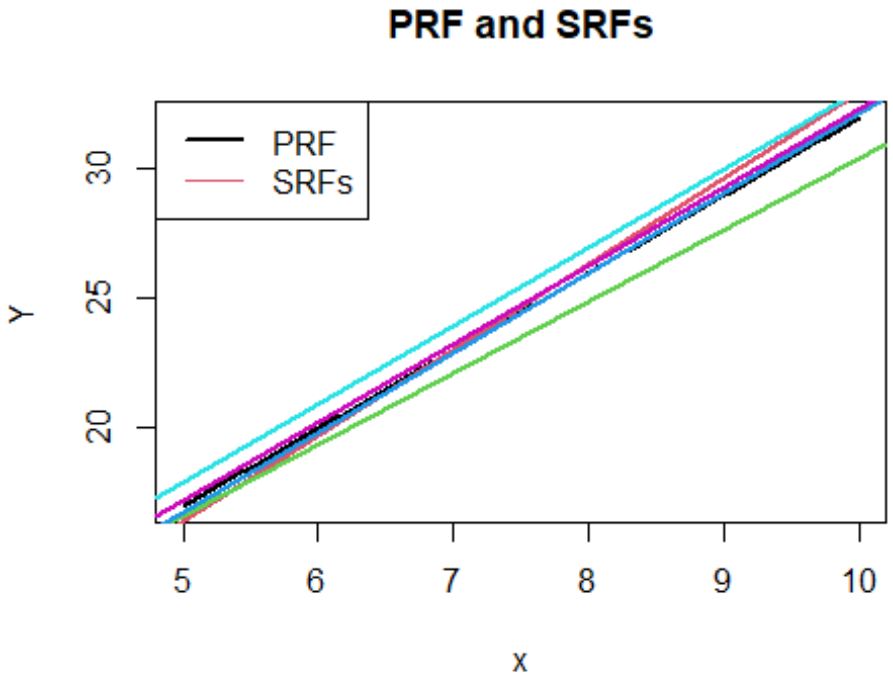
```
##       xi
## 3.305396
```

```
y.hat=b0+b1*xi
plot(x,y,type='l',lwd=2,main="PRF and SRF")
lines(xi,y.hat,col="red",lty="dotted")
legend("topleft",col=c("black","red"),legend=c("PRF","SRF"),
       lty=c("solid","dotted"))
```



Step 4: *Repeat steps 2-3 five times. Graph the 5 least squares regression lines over the population regression line obtained in Step 1.*

```
rm(list=ls())
set.seed(123)
n = 50
x_grid = seq(5,10,length.out=200)
y_grid = 2 + 3*x_grid

coefs = matrix(0, 5, 2)

plot(x_grid, y_grid, type='l', lwd=2, col="black",main="PRF and SRFs",
xlab="x", ylab="Y")

for(i in 1:5){

  x_sim = runif(n,5,10)
  e = rnorm(n,0,4)
  y_sim = 2 + 3*x_sim + e
  model = lm(y_sim ~ x_sim)
  coefs[i,] = coef(model)
  abline(model, col=i+1, lwd=2)
}

legend("topleft",legend=c("PRF","SRFs"),col=c("black",2),lty=c(1,1),lwd=c(2,1
))
```



```
coefs
```

```
##             [,1]     [,2]
## [1,] -0.09638929 3.305396
## [2,]  2.79218839 2.761042
## [3,]  1.39299737 3.073267
## [4,]  2.82308856 3.023608
## [5,]  2.03250638 3.028097
```

The PRF remains same. But the SRF is not fixed

2 Problem to demonstrate that $\hat{\beta}_0$ and $\hat{\beta}$ minimises RSS

Step 1: *Generate xi from Uniform(5, 10) and mean centre the values. Generate ϵ_i from N(0, 1). Calculate $y_i = 2 + 3x_i + \epsilon_i$, i = 1,2,.., n. Take n=50 and seed=123*

```
rm(list=ls())
n=50
set.seed(123)
x=runif(n,5,10)
xm=x-mean(x)
e=rnorm(n)
y=2+3*xm+e
```

Step 2: *Now imagine that you only have the data on (xi, yi), i = 1, 2, .., n, without knowing the mechanism that was used to generate the data in step 1. Assuming a linear regression of the type $y_i = \beta_0 + \beta x_i + \epsilon_i$, and based on these data (xi, yi), i = 1, 2, .., n, obtain the least squares estimates of β_0 and β.*

```
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
```

```r
beta0=coef(fit)[1];beta0
```

```
## (Intercept)
##    2.056189
```

```r
beta=coef(fit)[2];beta
```

```
##       xm
## 3.076349
```

Step 3: *Take a large number of grid values of (β0, β) that also include the least squares estimates obtained from step 2. Compute the RSS for each parametric choice of (β0, β), where RSS = (y1 − β0 − βx1)^2 + (y2 − β0 − βx2)^2 + ….(yn −β0 − βxn)^2. Find out for which combination of (β0, β), RSS is minimum.*

```r
beta0.grid=seq(-1,1,length.out=101)+beta0
beta.grid=seq(-1,1,length.out=101)+beta

RSS=c()
for(i in 1:length(beta.grid))
{
  RSS[i]=sum((y-beta0.grid[i]-beta.grid[i]*xm)^2)
}
RSS
```

```
##   [1] 198.53732 192.35609 186.29972 180.36823 174.56162 168.87987 163.32301
##   [8] 157.89101 152.58389 147.40164 142.34426 137.41176 132.60413 127.92138
##  [15] 123.36350 118.93049 114.62236 110.43910 106.38071 102.44719  98.63855
##  [22]  94.95479  91.39589  87.96187  84.65273  81.46845  78.40905  75.47453
##  [29]  72.66487  69.98009  67.42019  64.98516  62.67500  60.48971  58.42930
##  [36]  56.49376  54.68310  52.99730  51.43639  50.00034  48.68917  47.50287
##  [43]  46.44145  45.50490  44.69322  44.00641  43.44448  43.00743  42.69524
##  [50]  42.50793  42.44550  42.50793  42.69524  43.00743  43.44448  44.00641
##  [57]  44.69322  45.50490  46.44145  47.50287  48.68917  50.00034  51.43639
##  [64]  52.99730  54.68310  56.49376  58.42930  60.48971  62.67500  64.98516
##  [71]  67.42019  69.98009  72.66487  75.47453  78.40905  81.46845  84.65273
##  [78]  87.96187  91.39589  94.95479  98.63855 102.44719 106.38071 110.43910
##  [85] 114.62236 118.93049 123.36350 127.92138 132.60413 137.41176 142.34426
##  [92] 147.40164 152.58389 157.89101 163.32301 168.87987 174.56162 180.36823
##  [99] 186.29972 192.35609 198.53732
```

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

```
##     beta0.grid beta.grid       RSS
## 1     1.056189  2.076349 198.53732
## 2     1.076189  2.096349 192.35609
## 3     1.096189  2.116349 186.29972
## 4     1.116189  2.136349 180.36823
## 5     1.136189  2.156349 174.56162
## 6     1.156189  2.176349 168.87987
## 7     1.176189  2.196349 163.32301
```

```
## 8     1.196189  2.216349 157.89101
## 9     1.216189  2.236349 152.58389
## 10    1.236189  2.256349 147.40164
## 11    1.256189  2.276349 142.34426
## 12    1.276189  2.296349 137.41176
## 13    1.296189  2.316349 132.60413
## 14    1.316189  2.336349 127.92138
## 15    1.336189  2.356349 123.36350
## 16    1.356189  2.376349 118.93049
## 17    1.376189  2.396349 114.62236
## 18    1.396189  2.416349 110.43910
## 19    1.416189  2.436349 106.38071
## 20    1.436189  2.456349 102.44719
## 21    1.456189  2.476349  98.63855
## 22    1.476189  2.496349  94.95479
## 23    1.496189  2.516349  91.39589
## 24    1.516189  2.536349  87.96187
## 25    1.536189  2.556349  84.65273
## 26    1.556189  2.576349  81.46845
## 27    1.576189  2.596349  78.40905
## 28    1.596189  2.616349  75.47453
## 29    1.616189  2.636349  72.66487
## 30    1.636189  2.656349  69.98009
## 31    1.656189  2.676349  67.42019
## 32    1.676189  2.696349  64.98516
## 33    1.696189  2.716349  62.67500
## 34    1.716189  2.736349  60.48971
## 35    1.736189  2.756349  58.42930
## 36    1.756189  2.776349  56.49376
## 37    1.776189  2.796349  54.68310
## 38    1.796189  2.816349  52.99730
## 39    1.816189  2.836349  51.43639
## 40    1.836189  2.856349  50.00034
## 41    1.856189  2.876349  48.68917
## 42    1.876189  2.896349  47.50287
## 43    1.896189  2.916349  46.44145
## 44    1.916189  2.936349  45.50490
## 45    1.936189  2.956349  44.69322
## 46    1.956189  2.976349  44.00641
## 47    1.976189  2.996349  43.44448
## 48    1.996189  3.016349  43.00743
## 49    2.016189  3.036349  42.69524
## 50    2.036189  3.056349  42.50793
## 51    2.056189  3.076349  42.44550
## 52    2.076189  3.096349  42.50793
## 53    2.096189  3.116349  42.69524
## 54    2.116189  3.136349  43.00743
## 55    2.136189  3.156349  43.44448
## 56    2.156189  3.176349  44.00641
## 57    2.176189  3.196349  44.69322
```

```
## 58    2.196189  3.216349  45.50490
## 59    2.216189  3.236349  46.44145
## 60    2.236189  3.256349  47.50287
## 61    2.256189  3.276349  48.68917
## 62    2.276189  3.296349  50.00034
## 63    2.296189  3.316349  51.43639
## 64    2.316189  3.336349  52.99730
## 65    2.336189  3.356349  54.68310
## 66    2.356189  3.376349  56.49376
## 67    2.376189  3.396349  58.42930
## 68    2.396189  3.416349  60.48971
## 69    2.416189  3.436349  62.67500
## 70    2.436189  3.456349  64.98516
## 71    2.456189  3.476349  67.42019
## 72    2.476189  3.496349  69.98009
## 73    2.496189  3.516349  72.66487
## 74    2.516189  3.536349  75.47453
## 75    2.536189  3.556349  78.40905
## 76    2.556189  3.576349  81.46845
## 77    2.576189  3.596349  84.65273
## 78    2.596189  3.616349  87.96187
## 79    2.616189  3.636349  91.39589
## 80    2.636189  3.656349  94.95479
## 81    2.656189  3.676349  98.63855
## 82    2.676189  3.696349 102.44719
## 83    2.696189  3.716349 106.38071
## 84    2.716189  3.736349 110.43910
## 85    2.736189  3.756349 114.62236
## 86    2.756189  3.776349 118.93049
## 87    2.776189  3.796349 123.36350
## 88    2.796189  3.816349 127.92138
## 89    2.816189  3.836349 132.60413
## 90    2.836189  3.856349 137.41176
## 91    2.856189  3.876349 142.34426
## 92    2.876189  3.896349 147.40164
## 93    2.896189  3.916349 152.58389
## 94    2.916189  3.936349 157.89101
## 95    2.936189  3.956349 163.32301
## 96    2.956189  3.976349 168.87987
## 97    2.976189  3.996349 174.56162
## 98    2.996189  4.016349 180.36823
## 99    3.016189  4.036349 186.29972
## 100   3.036189  4.056349 192.35609
## 101   3.056189  4.076349 198.53732
```

```
df[which.min(RSS),]
```

```
##    beta0.grid beta.grid     RSS
## 51   2.056189  3.076349 42.4455
```

3 Problem to demonstrate that least square estimators are unbiased

Step 1: *Generate xi(i = 1, 2, .., n) from Uniform(0, 1), ϵi(i = 1, 2, .., n) from N(0, 1) and hence generate y using yi = β0 + βxi + ϵi. (Take β0 = 2, β = 3).*

```
rm(list=ls())
n=50
set.seed(123)
x=runif(n,0,1)
e=rnorm(n,0,1)
beta_not=2
beta=3
y=beta_not+beta*x+e
```

Step 2: *On the basis of the data (xi, yi)(i = 1, 2, .., n) generated in Step 1, obtain the least square estimates of β0 and β.*

```
model=lm(y~x)
b0=coef(model)[1];b0
```

```
## (Intercept)
##    1.857647
```

```
b1=coef(model)[2];b1
```

```
##        x
## 3.381745
```

Repeat Steps 1-2, R = 1000 times. In each simulation obtain β^0 and β^. Finally, the least-square estimates will be given by the average of these estimated values.Compare these with the true β0 and β and comment.

Take n = 50 and seed=123.

```
R = 1000
b0_store = c()
b1_store = c()

for(i in 1:R){
  x = runif(n,0,1)
  e = rnorm(n,0,1)
  y = beta_not + beta*x + e
  model = lm(y~x)
  b0_store[i] = coef(model)[1]
  b1_store[i] = coef(model)[2]
}

mean_b0 = mean(b0_store)
mean_b1 = mean(b1_store)
mean_b0
```

```
## [1] 2.013052

mean_b1

## [1] 2.981907
```

4 Comparing several simple linear regressions

Attach "Boston" data from MASS library in R. Select median value of owneroccupied homes, as the response and per capita crime rate, nitrogen oxides concentration, proportion of blacks and percentage of lower status of the population as predictors.

(a) Selecting the predictors one by one, run four separate linear regressions to the data. Present the output in a single table.

(b) Which model gives the best fit?

(c) Compare the coefficients of the predictors from each model and comment on the usefulness of the predictors.

```
rm(list=ls())
library(MASS)
head(Boston)

##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7

attach(Boston)
model1=lm(medv~crim)
model2=lm(medv~nox)
model3=lm(medv~black)
model4=lm(medv~lstat)
library(stargazer)

stargazer(model1,model2,model3,model4,type="html",out="f1.html")
```

	Dependent variable:			
	medv			
	(1)	(2)	(3)	(4)
crim	-0.415***			
	(0.044)			
nox		-33.916***		
		(3.196)		
black			0.034***	
			(0.004)	
lstat				-0.950***
				(0.039)
Constant	24.033***	41.346***	10.551***	34.554***
	(0.409)	(1.811)	(1.557)	(0.563)
Observations	506	506	506	506
R^2	0.151	0.183	0.111	0.544
Adjusted R^2	0.149	0.181	0.109	0.543
Residual Std. Error (df = 504)	8.484	8.323	8.679	6.216
F Statistic (df = 1; 504)	89.486***	112.591***	63.054***	601.618***
Note:				*p<0.1; **p<0.05; ***p<0.01