

# ASSIGNMENT 2

SRIJITA DEY\_720

2026-02-05

## 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  $y = 2 + 3x + \epsilon$ .

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

Step 2: Generate  $x_i (i = 1, 2, \dots, n)$  from  $\text{Uniform}(5, 10)$  and  $\epsilon_i (i = 1, 2, \dots, n)$  from  $N(0, 4^2)$ . Hence, compute  $y_1, y_2, \dots, y_n$ .

Step 3: On the basis of the data  $(x_i, y_i) (i = 1, 2, \dots, n)$  generated in Step 2, report the least squares regression line.

Step 4: Repeat steps 2-3 five times. Graph the 5 least squares regression lines over the population regression line obtained in Step 1. Interpret the findings. Take  $n = 50$ . Set the seed as  $\text{seed}=123$ .

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

```
#step 1
```

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

```
x
```

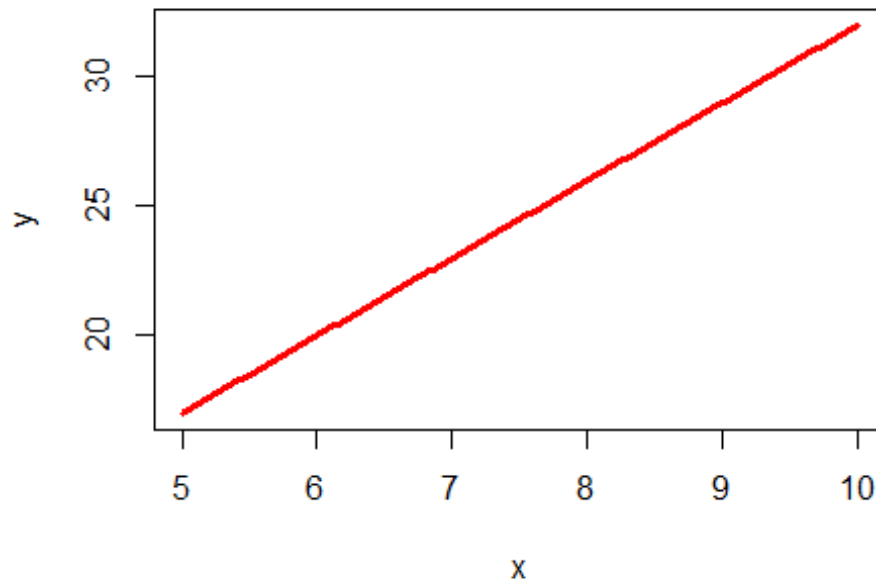
```
## [1] 5.000000 5.025126 5.050251 5.075377 5.100503 5.125628 5.15075
4
## [8] 5.175879 5.201005 5.226131 5.251256 5.276382 5.301508 5.32663
3
## [15] 5.351759 5.376884 5.402010 5.427136 5.452261 5.477387 5.50251
3
## [22] 5.527638 5.552764 5.577889 5.603015 5.628141 5.653266 5.67839
2
## [29] 5.703518 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,col="red",main="Plot of Y=2+3X")
```

**Plot of  $Y=2+3X$**



```
#step 2
set.seed(123)
x_i=runif(50,5,10)
e_i=rnorm(50,0,4)
y_i=2+3*x_i+e_i
y_i

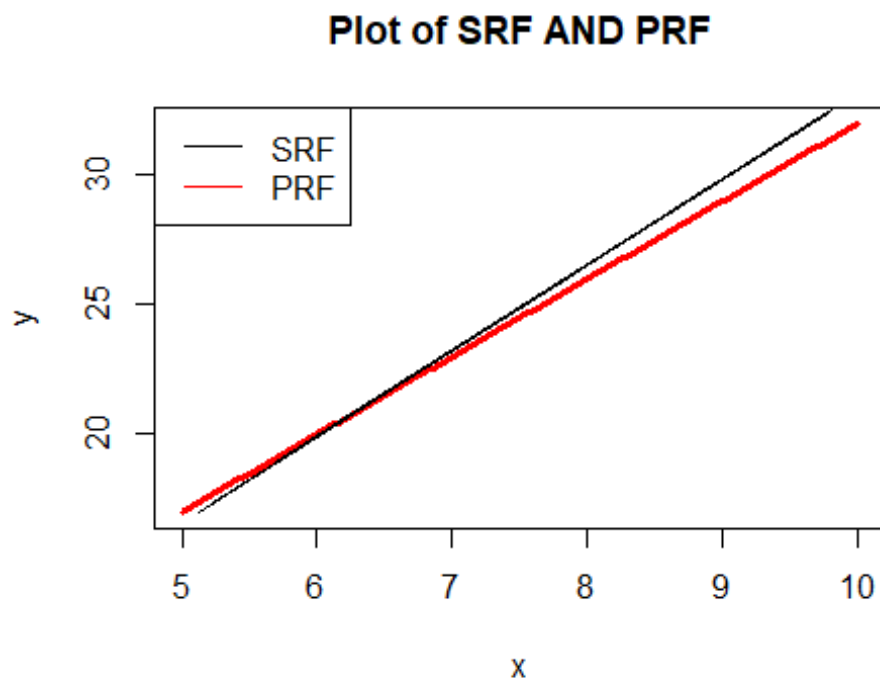
## [1] 14.56689 32.17573 23.74815 25.69271 36.12227 19.38920 23.74130 33.966
79
## [9] 28.78406 27.13555 34.10706 26.01568 26.91591 24.36565 17.02199 27.718
55
## [17] 19.85965 12.56931 30.59463 36.14940 25.85066 25.78051 24.74098 35.033
91
## [25] 26.50211 28.64123 25.04680 25.74065 26.81181 18.30362 37.51125 24.339
47
## [33] 29.69903 29.42743 18.23297 25.68550 26.36760 18.91329 17.69841 16.187
22
## [41] 20.35611 25.01103 23.41788 26.22175 27.48701 17.11797 11.25884 28.012
39
## [49] 18.15279 27.11538

#step 3
model=lm(y_i~x_i)
summary(model)

##
## Call:
```

```
## lm(formula = y_i ~ x_i)
##
## 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
## x_i          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

y_hat=0.09639+3.30540*x_i
plot(x,y,type='l',lwd=3,col="red",main="Plot of SRF AND PRF")
lines(x_i,y_hat)
legend("topleft",legend=c("SRF","PRF"),col=c("black","red"),lty=c("solid","solid"))
```



```
b0=coef(model)[1]
b1=coef(model)[2]
b0
```

```

## (Intercept)
## -0.09638929

b1

##      x_i
## 3.305396

#step 4

set.seed(123)
x_1=runif(50,5,10)
e_1=rnorm(50,0,4)
y_1=2+3*x_1+e_1
model1=lm(y_1~x_1)
summary(model1)

##
## Call:
## lm(formula = y_1 ~ x_1)
##
## 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
## x_1          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

x_2=runif(50,5,10)
e_2=rnorm(50,0,4)
y_2=2+3*x_2+e_2
model2=lm(y_2~x_2)

x_3=runif(50,5,10)
e_3=rnorm(50,0,4)
y_3=2+3*x_3+e_3
model3=lm(y_3~x_3)

x_4=runif(50,5,10)
e_4=rnorm(50,0,4)
y_4=2+3*x_4+e_4
model4=lm(y_4~x_4)

```

```

x_5=runif(50,5,10)
e_5=rnorm(50,0,4)
y_5=2+3*x_5+e_5
model5=lm(y_5~x_5)

coeff_table=data.frame(coef(model1),
                        coef(model2),
                        coef(model3),
                        coef(model4),
                        coef(model5))

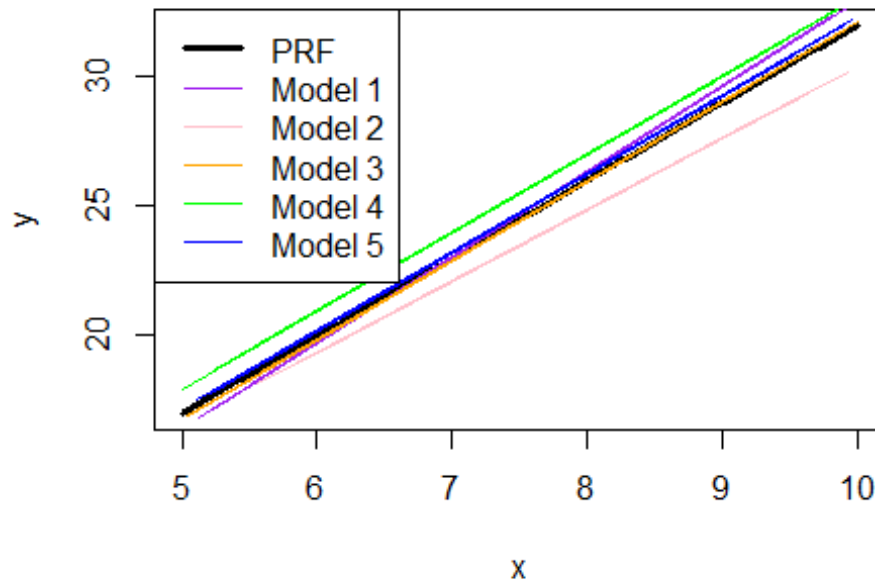
coeff_table

##           coef.model1. coef.model2. coef.model3. coef.model4. coef.model
5.
## (Intercept) -0.09638929    2.792188    1.392997    2.823089    2.0325
06
## x_1          3.30539569    2.761042    3.073267    3.023608    3.0280
97

plot(x,y,type='l',lwd=3,main="Plot of the PRF and the SRFs")
lines(x_1,predict(model1),type='l',col="purple")
lines(x_2,predict(model2),type='l',col="pink")
lines(x_3,predict(model3),type='l',col="orange")
lines(x_4,predict(model4),type='l',col="green")
lines(x_5,predict(model5),type='l',col="blue")
legend("topleft",legend=c("PRF","Model 1","Model 2","Model 3",
                          "Model 4","Model 5"),
      col=c("black","purple","pink","orange","green","blue"),
      lwd=c(3,1,1,1,1,1))

```

**Plot of the PRF and the SRFs**



```
coeff_table
##          coef.model1. coef.model2. coef.model3. coef.model4. coef.model
5.
## (Intercept) -0.09638929    2.792188    1.392997    2.823089    2.0325
06
## x_1         3.30539569    2.761042    3.073267    3.023608    3.0280
97
```

From the above graph we observe that the PRF remains the same but the SRF keeps on changing.

## 2. Problem to demonstrate that $\beta_0$ and $\hat{\beta}$ minimises RSS

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

Step 2: Now imagine that you only have the data on  $(x_i, y_i)$ ,  $i = 1, 2, \dots, n$ , 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  $(x_i, y_i)$ ,  $i = 1, 2, \dots, n$ , obtain the least squares estimates of  $\beta_0$  and  $\beta$ .

Step 3: Take a large number of grid values of  $(\beta_0, \beta)$  that also include the least squares estimates obtained from step 2. Compute the RSS for each parametric choice of  $(\beta_0, \beta)$ , where  $\text{RSS} = (y_1 - \beta_0 - \beta x_1)^2 + (y_2 - \beta_0 - \beta x_2)^2 + \dots + (y_n - \beta_0 - \beta x_n)^2$ . Find out for which combination of  $(\beta_0, \beta)$ , RSS is minimum.

```

rm(list=ls())

#step 1
set.seed(123)
x=runif(50,5,10)
e=rnorm(50,0,1 )
x_centered=x-mean(x)
y=2+3*x_centered+e
head(data.frame(x_centered,y))

##   x_centered      y
## 1 -1.1625673 -3.1743951
## 2  1.3410708  6.8609995
## 3 -0.5555703  0.4866624
## 4  1.8146322  6.3057595
## 5  2.1018816  9.5594596
## 6 -2.3726724 -4.6915529

#step 2
m=lm(y~x_centered)
coef(m)

## (Intercept)  x_centered
##    2.056189    3.076349

#step 3
beta0=coef(m)[1]
beta0

## (Intercept)
##    2.056189

beta=coef(m)[2]
beta

## x_centered
##    3.076349

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]*x_centered)^2)
}
RSS

##   [1] 198.53732 192.35609 186.29972 180.36823 174.56162 168.87987 163.3230
## 1
##   [8] 157.89101 152.58389 147.40164 142.34426 137.41176 132.60413 127.9213
## 8

```

```
## [15] 123.36350 118.93049 114.62236 110.43910 106.38071 102.44719 98.6385
5
## [22] 94.95479 91.39589 87.96187 84.65273 81.46845 78.40905 75.4745
3
## [29] 72.66487 69.98009 67.42019 64.98516 62.67500 60.48971 58.4293
0
## [36] 56.49376 54.68310 52.99730 51.43639 50.00034 48.68917 47.5028
7
## [43] 46.44145 45.50490 44.69322 44.00641 43.44448 43.00743 42.6952
4
## [50] 42.50793 42.44550 42.50793 42.69524 43.00743 43.44448 44.0064
1
## [57] 44.69322 45.50490 46.44145 47.50287 48.68917 50.00034 51.4363
9
## [64] 52.99730 54.68310 56.49376 58.42930 60.48971 62.67500 64.9851
6
## [71] 67.42019 69.98009 72.66487 75.47453 78.40905 81.46845 84.6527
3
## [78] 87.96187 91.39589 94.95479 98.63855 102.44719 106.38071 110.4391
0
## [85] 114.62236 118.93049 123.36350 127.92138 132.60413 137.41176 142.3442
6
## [92] 147.40164 152.58389 157.89101 163.32301 168.87987 174.56162 180.3682
3
## [99] 186.29972 192.35609 198.53732
```

```
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

##	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  $x_i (i = 1, 2, \dots, n)$  from  $\text{Uniform}(0, 1)$ ,  $\epsilon_i (i = 1, 2, \dots, n)$  from  $N(0, 1)$  and hence generate  $y$  using  $y_i = \beta_0 + \beta x_i + \epsilon_i$ . (Take  $\beta_0 = 2$ ,  $\beta = 3$ ). Step 2: On the basis of the data  $(x_i, y_i) (i = 1, 2, \dots, n)$  generated in Step 1, obtain the least square estimates of  $\beta_0$  and  $\beta$ . Repeat Steps 1-2,  $R = 1000$  times. In each simulation obtain  $\hat{\beta}_0$  and  $\hat{\beta}$ . Finally, the least-square estimates will be given by the average of these estimated values. Compare these with the true  $\beta_0$  and  $\beta$  and comment. Take  $n = 50$  and  $\text{seed} = 123$ .

```

rm(list=ls())

#step 1
set.seed(123)
x=runif(50,0,1)
e=rnorm(50,0,1)
beta_0=2
beta=3
y=beta_0+beta*x+e

#step 2
m2=lm(y~x)
b0=coef(m2)[1]
b0

## (Intercept)
##      1.857647

b1=coef(m2)[2]
b1

##           x
##  3.381745

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

for(i in 1:R){
  x = runif(50,0,1)
  e = rnorm(50,0,1)
  y = beta_0 + beta*x + e
  m3 = lm(y~x)
  b0_store[i] = coef(m3)[1]
  b1_store[i] = coef(m3)[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 owner-occupied 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.

- Selecting the predictors one by one, run four separate linear regressions to the data. Present the output in a single table.
- Which model gives the best fit?
- Compare the coefficients of the predictors from each model and comment on the usefulness of the predictors.

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

## Warning: package 'MASS' was built under R version 4.5.2

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.
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671  2 242    17.8 396.90  9.
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671  2 242    17.8 392.83  4.
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622  3 222    18.7 394.63  2.
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622  3 222    18.7 396.90  5.
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622  3 222    18.7 394.12  5.
##      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)

## Warning: package 'stargazer' was built under R version 4.5.2

##
## Please cite as:
## Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.
## R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```
stargazer(model1,model2,model3,model4,type="html",out="f3.html")
```

	<i>Dependent variable:</i>			
	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*** (0.409)	41.346*** (1.811)	10.551*** (1.557)	34.554*** (0.563)
Observations	506	506	506	506
R <sup>2</sup>	0.151	0.183	0.111	0.544
Adjusted R <sup>2</sup>	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***
<i>Note:</i>		* p<0.1; ** p<0.05; *** p<0.01		