Assignment on predictive analysis

Chandrani Sengupta

2026-02-04

Problem Set 2: Linear Regression

Question 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. 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. Step 3: On the basis of the data (xi

, yi)(i = 1, 2, .., 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 seed=123.

```
#step 1
x=seq(5,10,length.out=200)
x
```

```
##   [1]  5.000000  5.025126  5.050251  5.075377  5.100503  5.125628
5.150754
##   [8]  5.175879  5.201005  5.226131  5.251256  5.276382  5.301508
5.326633
##  [15]  5.351759  5.376884  5.402010  5.427136  5.452261  5.477387
5.502513
##  [22]  5.527638  5.552764  5.577889  5.603015  5.628141  5.653266
5.678392
##  [29]  5.703518  5.728643  5.753769  5.778894  5.804020  5.829146
5.854271
##  [36]  5.879397  5.904523  5.929648  5.954774  5.979899  6.005025
6.030151
##  [43]  6.055276  6.080402  6.105528  6.130653  6.155779  6.180905
6.206030
##  [50]  6.231156  6.256281  6.281407  6.306533  6.331658  6.356784
6.381910
##  [57]  6.407035  6.432161  6.457286  6.482412  6.507538  6.532663
6.557789
##  [64]  6.582915  6.608040  6.633166  6.658291  6.683417  6.708543
6.733668
##  [71]  6.758794  6.783920  6.809045  6.834171  6.859296  6.884422
6.909548
```

```
##  [78]  6.934673  6.959799  6.984925  7.010050  7.035176  7.060302
7.085427
##  [85]  7.110553  7.135678  7.160804  7.185930  7.211055  7.236181
7.261307
##  [92]  7.286432  7.311558  7.336683  7.361809  7.386935  7.412060
7.437186
##  [99]  7.462312  7.487437  7.512563  7.537688  7.562814  7.587940
7.613065
## [106]  7.638191  7.663317  7.688442  7.713568  7.738693  7.763819
7.788945
## [113]  7.814070  7.839196  7.864322  7.889447  7.914573  7.939698
7.964824
## [120]  7.989950  8.015075  8.040201  8.065327  8.090452  8.115578
8.140704
## [127]  8.165829  8.190955  8.216080  8.241206  8.266332  8.291457
8.316583
## [134]  8.341709  8.366834  8.391960  8.417085  8.442211  8.467337
8.492462
## [141]  8.517588  8.542714  8.567839  8.592965  8.618090  8.643216
8.668342
## [148]  8.693467  8.718593  8.743719  8.768844  8.793970  8.819095
8.844221
## [155]  8.869347  8.894472  8.919598  8.944724  8.969849  8.994975
9.020101
## [162]  9.045226  9.070352  9.095477  9.120603  9.145729  9.170854
9.195980
## [169]  9.221106  9.246231  9.271357  9.296482  9.321608  9.346734
9.371859
## [176]  9.396985  9.422111  9.447236  9.472362  9.497487  9.522613
9.547739
## [183]  9.572864  9.597990  9.623116  9.648241  9.673367  9.698492
9.723618
## [190]  9.748744  9.773869  9.798995  9.824121  9.849246  9.874372
9.899497
## [197]  9.924623  9.949749  9.974874 10.000000
```

```
y=2+3*x
y
```

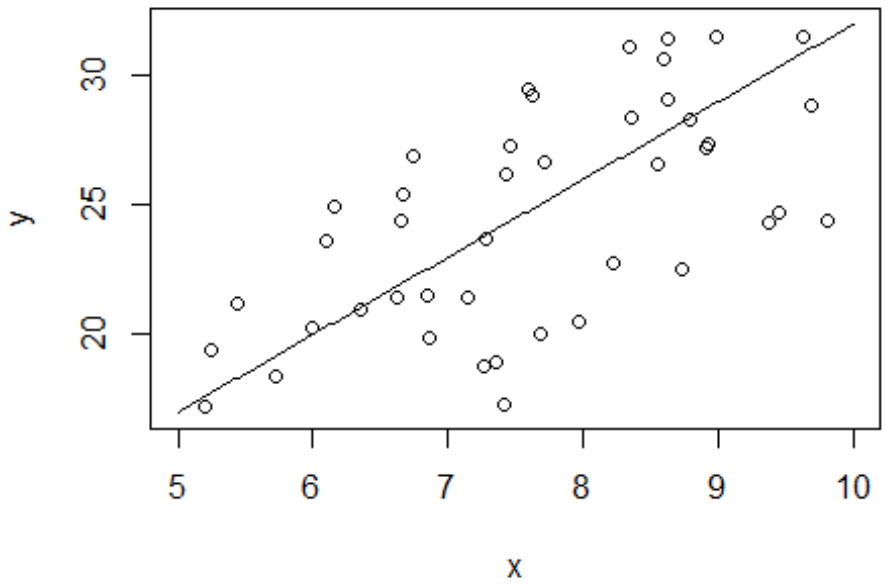
```
##   [1] 17.00000 17.07538 17.15075 17.22613 17.30151 17.37688 17.45226
17.52764
##   [9] 17.60302 17.67839 17.75377 17.82915 17.90452 17.97990 18.05528
18.13065
##  [17] 18.20603 18.28141 18.35678 18.43216 18.50754 18.58291 18.65829
18.73367
##  [25] 18.80905 18.88442 18.95980 19.03518 19.11055 19.18593 19.26131
19.33668
##  [33] 19.41206 19.48744 19.56281 19.63819 19.71357 19.78894 19.86432
19.93970
##  [41] 20.01508 20.09045 20.16583 20.24121 20.31658 20.39196 20.46734
```

```
20.54271
##  [49] 20.61809 20.69347 20.76884 20.84422 20.91960 20.99497 21.07035
21.14573
##  [57] 21.22111 21.29648 21.37186 21.44724 21.52261 21.59799 21.67337
21.74874
##  [65] 21.82412 21.89950 21.97487 22.05025 22.12563 22.20101 22.27638
22.35176
##  [73] 22.42714 22.50251 22.57789 22.65327 22.72864 22.80402 22.87940
22.95477
##  [81] 23.03015 23.10553 23.18090 23.25628 23.33166 23.40704 23.48241
23.55779
##  [89] 23.63317 23.70854 23.78392 23.85930 23.93467 24.01005 24.08543
24.16080
##  [97] 24.23618 24.31156 24.38693 24.46231 24.53769 24.61307 24.68844
24.76382
## [105] 24.83920 24.91457 24.98995 25.06533 25.14070 25.21608 25.29146
25.36683
## [113] 25.44221 25.51759 25.59296 25.66834 25.74372 25.81910 25.89447
25.96985
## [121] 26.04523 26.12060 26.19598 26.27136 26.34673 26.42211 26.49749
26.57286
## [129] 26.64824 26.72362 26.79899 26.87437 26.94975 27.02513 27.10050
27.17588
## [137] 27.25126 27.32663 27.40201 27.47739 27.55276 27.62814 27.70352
27.77889
## [145] 27.85427 27.92965 28.00503 28.08040 28.15578 28.23116 28.30653
28.38191
## [153] 28.45729 28.53266 28.60804 28.68342 28.75879 28.83417 28.90955
28.98492
## [161] 29.06030 29.13568 29.21106 29.28643 29.36181 29.43719 29.51256
29.58794
## [169] 29.66332 29.73869 29.81407 29.88945 29.96482 30.04020 30.11558
30.19095
## [177] 30.26633 30.34171 30.41709 30.49246 30.56784 30.64322 30.71859
30.79397
## [185] 30.86935 30.94472 31.02010 31.09548 31.17085 31.24623 31.32161
31.39698
## [193] 31.47236 31.54774 31.62312 31.69849 31.77387 31.84925 31.92462
32.00000
```

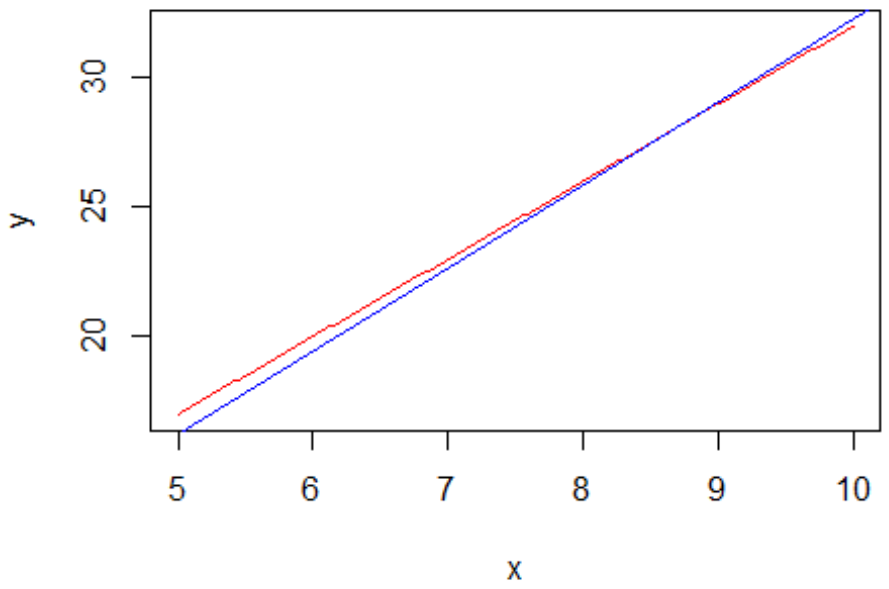
```r
plot(x,y,type='l')
#step2
n=50
x1=runif(50,5,10)
x1
```

```
##  [1] 7.969716 9.802671 8.558071 5.444186 7.459862 9.687883 8.216514
6.351221
##  [9] 5.515075 8.908464 5.989649 9.376409 5.204452 6.854462 9.635217
7.681841
```

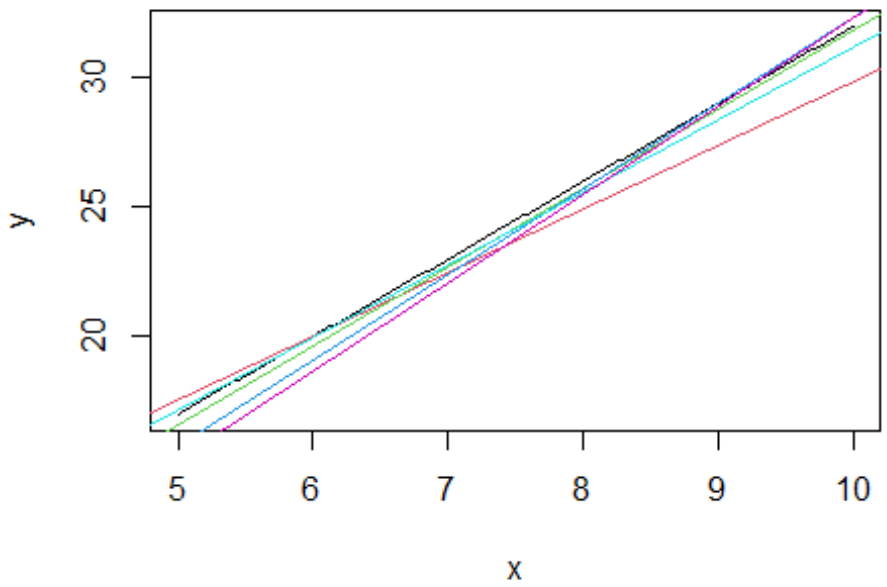
```
## [17] 9.250357 8.926526 6.652887 9.659516 8.623565 8.794105 5.243854
6.672255
## [25] 8.730931 8.354203 7.598708 5.726385 8.590139 7.360572 6.846349
7.262759
## [33] 9.451626 8.625439 6.165179 6.734055 6.619490 7.273549 5.021913
7.152649
## [41] 7.409838 7.627102 7.708101 5.634432 6.094051 9.386364 7.430531
8.982201
## [49] 5.325925 8.343973

e=rnorm(50,0,4)
e

##  [1] -5.39343803 -7.01921557 -1.07544142  2.85633781  2.96057313 -
2.16072507
##  [7] -3.85190811 -0.07078827 -9.55620984 -1.53661675  0.27875533 -
5.81085605
## [13] -0.41751804 -2.65413106  0.65266939 -5.00654445  5.63594574 -
1.39750081
## [19]  2.41738215  9.10290410  3.56500631 -0.04186448  1.70791937
3.43054471
## [25] -5.65629232  1.33070910  4.68242828 -0.78183322  2.93033457 -
5.11923843
## [31] -1.03321432 -4.98856727 -5.60022368  1.26181681  4.42633547
4.74308947
## [37] -0.43367473 -0.14537730 -1.84958336 -2.04136103 -6.91170270
4.39029483
## [43]  1.55226501 -3.08342295  3.35827158  6.71350113  1.88409998
2.60088200
## [49] -5.26477367  4.11314241

y1=2+3*x1+e
y1

##  [1] 20.515709 24.388799 26.598771 21.188896 27.340159 28.902924 22.797633
##  [8] 20.982874  8.989014 27.188775 20.247701 24.318371 17.195837 19.909255
## [15] 31.558320 20.038980 35.387017 27.382078 24.376044 40.081452 31.435701
## [22] 28.340451 19.439482 25.447308 22.536502 28.393318 29.478552 18.397322
## [29] 30.700752 18.962478 21.505831 18.799710 24.754654 29.138134 24.921873
## [36] 26.945256 21.424797 23.675270 15.216157 21.416587 17.317810 29.271602
## [43] 26.676568 15.819872 23.640424 36.872594 26.175694 31.547484 12.713001
## [50] 31.145061

points(x1,y1)
```




```r
#step3
lin.reg=lm(y1~x1)
lin.reg#lm object

##
## Call:
## lm(formula = y1 ~ x1)
##
## Coefficients:
## (Intercept)           x1
##      0.1032       3.2184

coef(lin.reg)

## (Intercept)          x1
##   0.1031788   3.2183820

plot(x,y,type='l',col="red")#population regression line
abline(lin.reg,col="blue")#sample regression line
```




```r
#step4
plot(x,y,type='l')
n=50
beta.hat=c()
for(i in 1:5){

  x2=runif(n,5,10)
  eps=rnorm(n,0,4)
  y2=2+3*x2+eps
  lin.reg=lm(y2~x2)
  beta.hat=rbind(beta.hat,coef(lin.reg))
  abline(lin.reg,col=i+1)
  }
```




```
beta.hat=as.data.frame(beta.hat,colnames=c("beta_0_hat","beta_hat"))
head(beta.hat)
```

```
##   (Intercept)       x2
## 1   5.2033775 2.466752
## 2   1.3701508 3.046174
## 3  -0.8241796 3.314432
## 4   3.0977081 2.812735
## 5  -1.7479371 3.403991
```

Question 2 : Problem to demonstrate that β^0 and β^ minimises RSS

Step 1: Generate xi from Uniform(5, 10) and mean centre the values. Generate εi from N(0, 1). Calculate yi = 2 + 3xi + εi

, i = 1,2,.., n. Take n=50 and

seed=123. 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 yi = β0 + βxi + εi

, and based on these

data (xi , yi), i = 1, 2, .., n, obtain the least squares estimates of β0 and β. 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.

```
#step1

x=runif(50,5,10)
x
```

```
##  [1] 6.184547 7.960892 5.030061 9.148818 5.543152 7.283953 6.409737
7.404770
##  [9] 7.092128 5.455692 7.403749 9.214539 6.065383 5.783260 9.529368
6.295444
## [17] 8.863508 8.554613 6.830856 6.494904 5.041425 6.720073 8.862320
5.999880
## [25] 5.281977 6.293306 6.956423 7.254962 7.309258 6.291363 8.837801
6.221546
## [33] 9.814047 5.456501 7.103373 9.967378 5.713610 8.135989 6.959268
8.956002
## [41] 5.514825 9.899958 7.269552 9.570969 9.998060 9.763062 7.861456
6.379462
## [49] 6.006276 6.140459
```

```
eps=rnorm(50,0,1)
eps
```

```
##  [1] -0.985702853 -0.766779668 -0.373632231 -0.960611817  0.435824622
##  [6] -0.545872976 -1.681042607  0.077959405 -0.718442729 -2.106568039
## [11] -0.655608394  0.032740067 -1.017633524 -1.878365874 -0.006234435
## [16] -1.022627806 -0.789217131 -0.737715214 -0.005511238  1.267501570
## [21]  0.494603528 -0.136741054  2.518204651 -0.028904409  0.114965346
## [26] -0.715999897 -2.126166656  0.829429548  0.417751703 -0.214898704
## [31]  0.309045497  0.561123822  0.019602620 -0.072032936 -1.677541983
## [36]  0.652685945  0.555004334  0.077208314 -1.509826736 -0.369157003
## [41] -1.983371022  0.329924145  1.270275459 -1.434768413 -0.425525920
## [46]  1.100934387 -0.242491378  1.152271054  1.150912154  0.207809105
```

```
#step2
x_mean=mean(x)
x_mean_centered=x-x_mean
x_mean_centered
```

```
##  [1] -1.098052103  0.678293105 -2.252537773  1.866218815 -1.739447426
##  [6]  0.001354336 -0.872862049  0.122171194 -0.190471496 -1.826907045
## [11]  0.121149635  1.931939975 -1.217215893 -1.499339117  2.246768529
## [16] -0.987154808  1.580909426  1.272014087 -0.451743384 -0.787694831
```

```
## [21] -2.241174189 -0.562526459  1.579720858 -1.282718660 -2.000622164
## [26] -0.989292902 -0.326176230 -0.027637217  0.026659231 -0.991236189
## [31]  1.555201745 -1.061053181  2.531448137 -1.826098537 -0.179226244
## [36]  2.684779043 -1.568988974  0.853389481 -0.323331365  1.673402443
## [41] -1.767773729  2.617358581 -0.013047532  2.288369926  2.715461142
## [46]  2.480463183  0.578857024 -0.903136649 -1.276323283 -1.142140469
```

```
y=2+3*x_mean_centered+eps
y
```

```
##  [1] -2.2798592  3.2680996 -5.1312455  6.6380446 -2.7825177  1.4581900
##  [7] -2.2996288  2.4444730  0.7101428 -5.5872892  1.7078405  7.8285600
## [13] -2.6692812 -4.3763832  8.7340712 -1.9840922  5.9535111  5.0783270
## [19]  0.6392586  0.9044171 -4.2289190  0.1756796  9.2573672 -1.8770604
## [25] -3.8869011 -1.6838786 -1.1046953  2.7465179  2.4977294 -1.1886073
## [31]  6.9746507 -0.6220357  9.6139470 -3.5503285 -0.2152207 10.7070231
## [37] -2.1519626  4.6373768 -0.4798208  6.6510503 -5.2866922 10.1819999
## [43]  3.2311329  7.4303414  9.7208575 10.5423239  3.4940797  0.4428611
## [49] -0.6780577 -1.2186123
```

```
lin.reg=lm(y~x)
lin.reg
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##     -20.833        3.103
```

```
beta0_hat=coef(lin.reg) [1]
beta_hat=coef(lin.reg)[2]
```

```
beta0_hat
```

```
## (Intercept)
##   -20.83345
```

```
beta_hat
```

```
##        x
## 3.103451
```

```
beta0_grid= seq(beta0_hat-2, beta0_hat + 2, length.out = 100)
beta_grid = seq(beta_hat-2, beta_hat + 2, length.out = 100)

RSS <- matrix(NA, nrow = length(beta0_grid), ncol = length(beta_grid))

# Compute RSS for each (beta0, beta)
for (i in 1:length(beta0_grid)) {
```

```
  for (j in 1:length(beta_grid)) {
    RSS[i, j] <- sum((y1 - beta0_grid[i] - beta_grid[j] * x1)^2)
  }
}
which(RSS == min(RSS), arr.ind=TRUE)
```

```
##      row col
## [1,] 100 100
```

```
# Find minimum RSS
min_RSS <- min(RSS)
index <- which(RSS == min_RSS, arr.ind = TRUE)
index
```

```
##      row col
## [1,] 100 100
```

##Question 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).

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. Repeat Steps 1-2, R = 1000 times. In each simulation obtain βˆ

0 and βˆ. 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.

```
x=runif(50,0,1)
x
```

```
##  [1] 0.91954030 0.43609056 0.55652145 0.63723548 0.49624580 0.78449462
##  [7] 0.44903243 0.66848769 0.29980540 0.88313706 0.03624314 0.49694234
## [13] 0.85768868 0.08043533 0.13068987 0.94042848 0.08242313 0.95761017
## [19] 0.37303280 0.43664158 0.59222158 0.55077948 0.37940802 0.67552168
## [25] 0.96426716 0.41476347 0.59516219 0.64023404 0.55318780 0.19619461
## [31] 0.29200771 0.41083248 0.79893475 0.41551168 0.77732370 0.50439703
## [37] 0.75579330 0.18213116 0.19591294 0.47069061 0.53980904 0.38129912
## [43] 0.07102295 0.41714622 0.05595624 0.19683692 0.71674888 0.89386611
## [49] 0.78749603 0.34370396
```

```
eps=rnorm(50,0,1)
eps
```

```
##  [1] -0.069234486 -0.469155673 -1.281806623  1.550826371  1.306605699
##  [6]  1.617492140 -1.011527050  0.736505744 -0.724979760 -0.419388147
## [11]  2.482719838 -0.502607181  1.021796486  0.468301839 -0.951487135
## [16] -1.024428388 -0.272726842 -0.936856090  0.683643418 -0.531612523
## [21]  1.296531873  1.522019019 -0.490273153  1.939867799 -0.372639001
## [26]  0.678443614  0.001943419  0.454984054  0.374740448 -1.024112288
## [31]  0.490215698  1.542885311  1.487311429  1.578516412  0.392557275
## [36]  1.368257224  0.707285334  0.596229524  0.799482749  0.545883459
## [41] -0.443146164 -0.704946297  0.516925883 -1.154832531  0.099309907
## [46] -1.151124845  2.221067283  0.637563328 -1.269752883 -0.929544778
```

```
y=2+3*x+eps
y
```

```
##  [1] 4.689386 2.839116 2.387758 5.462533 4.795343 5.970976 2.335570
4.741969
##  [9] 2.174436 4.230023 4.591449 2.988220 5.594863 2.709608 1.440582
3.796857
## [17] 1.974543 3.935974 3.802742 2.778312 5.073197 5.174357 2.647951
5.966433
## [25] 4.520162 3.922734 3.787430 4.375686 4.034304 1.564472 3.366239
4.775383
## [33] 5.884116 4.825051 4.724528 4.881448 4.974665 3.142623 3.387222
3.957955
## [41] 3.176281 2.438951 2.729995 2.096606 2.267179 1.439386 6.371314
5.319162
## [49] 3.092735 2.101567
```

```
lin.reg=lm(y~x)
lin.reg
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##       2.170        3.194
```

```
beta.hat=c()
for(i in 1:1000)
{
  x=runif(50,0,1)
  eps=rnorm(50,0,1)
  y=2+3*x+eps
  lin.reg=lm(y~x)
  beta.hat=rbind(beta.hat,coef(lin.reg))
}
head(beta.hat)
```

```
##      (Intercept)        x
## [1,]    1.516681 3.443819
## [2,]    1.931161 3.107286
## [3,]    2.008123 2.982416
## [4,]    2.423518 2.835003
## [5,]    1.670347 3.660578
## [6,]    2.094118 2.563321
```

```
avg_beta0=mean(beta.hat[,1])
avg_beta0
```

```
## [1] 1.988911
```

```
avg_beta=mean(beta.hat[,2])
avg_beta
```

```
## [1] 3.001523
```

Question 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 popu- lation 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.

```
library(MASS)
```

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

```
x1=Boston
y=Boston$medv#response
x=data.frame(x1$crim,x1$nox,x1$black,x1$lstat)#predictor
y
```

```
##   [1] 24.0 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 15.0 18.9 21.7
20.4 18.2
##  [16] 19.9 23.1 17.5 20.2 18.2 13.6 19.6 15.2 14.5 15.6 13.9 16.6 14.8
18.4 21.0
##  [31] 12.7 14.5 13.2 13.1 13.5 18.9 20.0 21.0 24.7 30.8 34.9 26.6 25.3
24.7 21.2
##  [46] 19.3 20.0 16.6 14.4 19.4 19.7 20.5 25.0 23.4 18.9 35.4 24.7 31.6
23.3 19.6
##  [61] 18.7 16.0 22.2 25.0 33.0 23.5 19.4 22.0 17.4 20.9 24.2 21.7 22.8
23.4 24.1
```

```
##  [76] 21.4 20.0 20.8 21.2 20.3 28.0 23.9 24.8 22.9 23.9 26.6 22.5 22.2
23.6 28.7
##  [91] 22.6 22.0 22.9 25.0 20.6 28.4 21.4 38.7 43.8 33.2 27.5 26.5 18.6
19.3 20.1
## [106] 19.5 19.5 20.4 19.8 19.4 21.7 22.8 18.8 18.7 18.5 18.3 21.2 19.2
20.4 19.3
## [121] 22.0 20.3 20.5 17.3 18.8 21.4 15.7 16.2 18.0 14.3 19.2 19.6 23.0
18.4 15.6
## [136] 18.1 17.4 17.1 13.3 17.8 14.0 14.4 13.4 15.6 11.8 13.8 15.6 14.6
17.8 15.4
## [151] 21.5 19.6 15.3 19.4 17.0 15.6 13.1 41.3 24.3 23.3 27.0 50.0 50.0
50.0 22.7
## [166] 25.0 50.0 23.8 23.8 22.3 17.4 19.1 23.1 23.6 22.6 29.4 23.2 24.6
29.9 37.2
## [181] 39.8 36.2 37.9 32.5 26.4 29.6 50.0 32.0 29.8 34.9 37.0 30.5 36.4
31.1 29.1
## [196] 50.0 33.3 30.3 34.6 34.9 32.9 24.1 42.3 48.5 50.0 22.6 24.4 22.5
24.4 20.0
## [211] 21.7 19.3 22.4 28.1 23.7 25.0 23.3 28.7 21.5 23.0 26.7 21.7 27.5
30.1 44.8
## [226] 50.0 37.6 31.6 46.7 31.5 24.3 31.7 41.7 48.3 29.0 24.0 25.1 31.5
23.7 23.3
## [241] 22.0 20.1 22.2 23.7 17.6 18.5 24.3 20.5 24.5 26.2 24.4 24.8 29.6
42.8 21.9
## [256] 20.9 44.0 50.0 36.0 30.1 33.8 43.1 48.8 31.0 36.5 22.8 30.7 50.0
43.5 20.7
## [271] 21.1 25.2 24.4 35.2 32.4 32.0 33.2 33.1 29.1 35.1 45.4 35.4 46.0
50.0 32.2
## [286] 22.0 20.1 23.2 22.3 24.8 28.5 37.3 27.9 23.9 21.7 28.6 27.1 20.3
22.5 29.0
## [301] 24.8 22.0 26.4 33.1 36.1 28.4 33.4 28.2 22.8 20.3 16.1 22.1 19.4
21.6 23.8
## [316] 16.2 17.8 19.8 23.1 21.0 23.8 23.1 20.4 18.5 25.0 24.6 23.0 22.2
19.3 22.6
## [331] 19.8 17.1 19.4 22.2 20.7 21.1 19.5 18.5 20.6 19.0 18.7 32.7 16.5
23.9 31.2
## [346] 17.5 17.2 23.1 24.5 26.6 22.9 24.1 18.6 30.1 18.2 20.6 17.8 21.7
22.7 22.6
## [361] 25.0 19.9 20.8 16.8 21.9 27.5 21.9 23.1 50.0 50.0 50.0 50.0 50.0
13.8 13.8
## [376] 15.0 13.9 13.3 13.1 10.2 10.4 10.9 11.3 12.3  8.8  7.2 10.5  7.4
10.2 11.5
## [391] 15.1 23.2  9.7 13.8 12.7 13.1 12.5  8.5  5.0  6.3  5.6  7.2 12.1
8.3  8.5
## [406]  5.0 11.9 27.9 17.2 27.5 15.0 17.2 17.9 16.3  7.0  7.2  7.5 10.4
8.8  8.4
## [421] 16.7 14.2 20.8 13.4 11.7  8.3 10.2 10.9 11.0  9.5 14.5 14.1 16.1
14.3 11.7
## [436] 13.4  9.6  8.7  8.4 12.8 10.5 17.1 18.4 15.4 10.8 11.8 14.9 12.6
14.1 13.0
```

```
## [451] 13.4 15.2 16.1 17.8 14.9 14.1 12.7 13.5 14.9 20.0 16.4 17.7 19.5
20.2 21.4
## [466] 19.9 19.0 19.1 19.1 20.1 19.9 19.6 23.2 29.8 13.8 13.3 16.7 12.0
14.6 21.4
## [481] 23.0 23.7 25.0 21.8 20.6 21.2 19.1 20.6 15.2  7.0  8.1 13.6 20.1
21.8 24.5
## [496] 23.1 19.7 18.3 21.2 17.5 16.8 22.4 20.6 23.9 22.0 11.9
```

```
model1=lm(y~x1$crim)
model1
```

```
##
## Call:
## lm(formula = y ~ x1$crim)
##
## Coefficients:
## (Intercept)      x1$crim
##     24.0331      -0.4152
```

```
summary(model1)
```

```
##
## Call:
## lm(formula = y ~ x1$crim)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.957  -5.449  -2.007   2.512  29.800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.03311    0.40914   58.74   <2e-16 ***
## x1$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
```

```
model2=lm(y~x1$nox)
model2
```

```
##
## Call:
## lm(formula = y ~ x1$nox)
##
## Coefficients:
## (Intercept)       x1$nox
##       41.35       -33.92
```

```r
model3=lm(y~x1$black)
model3
```

```
##
## Call:
## lm(formula = y ~ x1$black)
##
## Coefficients:
## (Intercept)     x1$black
##    10.55103      0.03359
```

```r
model4=lm(y~x1$crim)
model4
```

```
##
## Call:
## lm(formula = y ~ x1$crim)
##
## Coefficients:
## (Intercept)      x1$crim
##     24.0331      -0.4152
```

```r
models=c("model of y vs crim","model of y vs nox","model of y vs
black","model of y vs lstat")
beta_hat_0=c(coef(model1)[1],coef(model2)[1],coef(model3)[1],coef(model4)[1])
beta_hat=c(coef(model1)[2],coef(model2)[2],coef(model3)[2],coef(model4)[2])
Data=data.frame(models,beta_hat_0,beta_hat)
Data
```

```
##                models beta_hat_0     beta_hat
## 1  model of y vs crim   24.03311  -0.41519028
## 2   model of y vs nox   41.34587 -33.91605501
## 3 model of y vs black   10.55103   0.03359306
## 4 model of y vs lstat   24.03311  -0.41519028
```

```r
summary(model1)
```

```
##
## Call:
## lm(formula = y ~ x1$crim)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.957  -5.449  -2.007   2.512  29.800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.03311    0.40914   58.74   <2e-16 ***
## x1$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
```

```
summary(model2)
```

```
##
## Call:
## lm(formula = y ~ x1$nox)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.691  -5.121  -2.161   2.959  31.310
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   41.346      1.811   22.83   <2e-16 ***
## x1$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
```

```
summary(model3)
```

```
##
## Call:
## lm(formula = y ~ x1$black)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.884  -4.862  -1.684   2.932  27.763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.551034   1.557463   6.775 3.49e-11 ***
## x1$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
```

```
summary(model4)
```

```
##
## Call:
```

```
## lm(formula = y ~ x1$crim)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.957  -5.449  -2.007   2.512  29.800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.03311    0.40914   58.74   <2e-16 ***
## x1$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
```

None of them are a good fit but model 1 works better compared to the other models since the R square values are very less

Comparing predictors from different models

```
summary(model1)

##
## Call:
## lm(formula = y ~ x1$crim)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.957  -5.449  -2.007   2.512  29.800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.03311    0.40914   58.74   <2e-16 ***
## x1$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

summary(model2)

##
## Call:
## lm(formula = y ~ x1$nox)
##
## 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 ***
## x1$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
```

```
summary(model3)
```

```
##
## Call:
## lm(formula = y ~ x1$black)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.884  -4.862  -1.684   2.932  27.763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.551034   1.557463   6.775 3.49e-11 ***
## x1$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
```

```
summary(model4)
```

```
##
## Call:
## lm(formula = y ~ x1$crim)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.957  -5.449  -2.007   2.512  29.800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.03311    0.40914   58.74   <2e-16 ***
## x1$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
```