PS-4 Partial Soln

ECO204@EWU, Fall 2023

Instructor : Tanvir Hossain

2023-12-04

Contents

§. Problem 1 Soln

Solution a.

Here $\hat{\beta}_1 = 10$, means holding all other variables constant, if x_1 increases by 1 unit, Y is predicted to increase by 10 units. Similarly, $\hat{\beta}_2 = 8$ means holding all other variables constant, if x_2 increases by 1 unit, Y is predicted to increase by 8 units. Finally, $\hat{\beta}_3 = 9$ means holding all other variables constant, if x_3 increases by 1 unit, Y is predicted to increase by 9 units.

Solution b.

```
# define the function
y_hat <- function(x1, x2, x3){
  output <- 25 + 10*x1 + 8*x2 + 9*x3
  return(output)
}

# test the function
y_hat(2, 1, 5)
```

```
## [1] 98
```

Solution c.

```
# predict Y
y_hat(15, 10, 5)
```

```
## [1] 300
```

We can also give the names of the arguments, the benefit is that we don't have to remember the order of the arguments.

```
# predict Y
y_hat(x2 = 10, x1 = 15, x3 = 5)
```

```
## [1] 300
```

§. Problem 3 Soln

Let's calculate SSE first

```
# SSR
SST <- 6724.125
SSR <- 6216.375
SSE <- SST - SSR
SSE
```

```
## [1] 507.75
```

We can calculate MSR with the following formula

$$\text{MSR} = \frac{\text{SSR}}{p}$$

```
# MSR
p <- 2 # already this is defined before
MSR <- SSR / p
MSR
```

```
## [1] 3108.188
```

Now we can also calculate MSE with the following formula

$$\text{MSE} = \frac{\text{SSE}}{n - p - 1}$$

```
n <- 10
p <- 2
MSE <- SSE / (n - p - 1)
MSE
```

```
## [1] 72.53571
```

Finally we can calculate F with the following formula

$$F = \frac{\text{MSR}}{\text{MSE}}$$

```
Fcalc <- MSR / MSE
Fcalc
```

```
## [1] 42.85044
```

Now to do the F-test we need to find the critical value from the F distribution with $p = 2$ and $n - p - 1 = 7$ degrees of freedom. We can use the `qf` function to find the critical value for the F-test. Here $n = 10$.

```
# find the critical value of F
alpha <- 0.05
n <- 10
p <- 2

# F test is always one tail test, this is F(1-alpha)
Fcrit <- qf(p = 1 - alpha, df1 = p, df2 = n - p - 1)

Fcalc > Fcrit
```

```
## [1] TRUE
```

Since $F_{calc} > F_{crit}$, we reject the null hypothesis. Recall the Null hypothesis here is

$$H_0 : \beta_1 = \beta_2 = 0$$

So this means at least one of the coefficients is non-zero. Now similarly we can do the t-test for β_1 and β_2. The t-test for β_1 is

```
# t-test for beta_1
tcalc_beta1 <- (0.5906 - 0) / 0.0813


# it's a two tail test, so let's calculate the p value directly
pvalue_beta1 <- 2 * (1 - pt( abs(tcalc_beta1) , df = n - p - 1) )

pvalue_beta1
```

```
## [1] 0.0001678217
```

```
pvalue_beta1 < alpha
```

```
## [1] TRUE
```

So we reject the Null hypothesis that $\beta_1 = 0$. Similarly, we can do the t-test for β_2.

```r
# t-test for beta_2
tcalc_beta2 <- (0.4980 - 0) / 0.0567

pvalue_beta2 <- 2 * (1 - pt( abs(tcalc_beta2) , df = n - p - 1) )

pvalue_beta2
```

```
## [1] 4.997936e-05
```

```r
pvalue_beta2 < alpha
```

```
## [1] TRUE
```

So we reject the Null hypothesis that $\beta_2 = 0$.

§. Problem 6 Soln

Solution a.

In answer a. we need to load the data, fit the simple linear regression model with `tv`, and then write the estimated regression equation.

```r
# load the library
library(readxl)

# directly load the data
Showtime <- read_excel("/home/tanvir/Documents/ownCloud/Git_Repos/EWU_repos/3_Fall_2023/eco_204/ewu-eco2
```

We can view the data

```r
Showtime
```

```
## # A tibble: 8 x 5
##   revenue    tv newspaper magazines leaflets
##     <dbl> <dbl>     <dbl>     <dbl>    <dbl>
## 1      96   5         1.5       4.3     3.01
## 2      90   2         2         2.3     2
## 3      95   4         1.5       4.4     2.5
## 4      92   2.5       2.5       3.3     2.3
## 5      95   3         3.3       2.4     3.5
## 6      94   3.5       2.3       3.5     1.7
## 7      94   2.5       4.2       2.4     2.5
## 8      94   3         2.5       5.3     2.7
```

It's also good to see some summary stats

```r
summary(Showtime)
```

```
##     revenue           tv          newspaper       magazines
##  Min.   :90.00   Min.   :2.000   Min.   :1.500   Min.   :2.300
##  1st Qu.:93.50   1st Qu.:2.500   1st Qu.:1.875   1st Qu.:2.400
##  Median :94.00   Median :3.000   Median :2.400   Median :3.400
##  Mean   :93.75   Mean   :3.188   Mean   :2.475   Mean   :3.487
##  3rd Qu.:95.00   3rd Qu.:3.625   3rd Qu.:2.700   3rd Qu.:4.325
##  Max.   :96.00   Max.   :5.000   Max.   :4.200   Max.   :5.300
##     leaflets
##  Min.   :1.700
##  1st Qu.:2.225
```

```
##   Median :2.500
##   Mean   :2.526
##   3rd Qu.:2.777
##   Max.   :3.500
```

Looks okm Now let's regress `revenue` on `tv`

```r
# fit the model
model_slr <- lm(revenue ~ tv, data = Showtime)

# view the results with stargazer package
library(stargazer)
stargazer(model_slr, type = "text")
```

```
##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                               revenue
## -----------------------------------------------
## tv                            1.604**
##                               (0.478)
##
## Constant                     88.638***
##                               (1.582)
##
## -----------------------------------------------
## Observations                     8
## R2                             0.653
## Adjusted R2                    0.595
## Residual Std. Error        1.215 (df = 6)
## F Statistic             11.269** (df = 1; 6)
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

So the estimated regression equation is

$$\widehat{revenue} = 88.63 + 1.604\text{tv}$$

Solution b.

We need to develop an estimated equation using both tv and newspaper advertising. So let's fit MLR (notice the dot after the tilde sign, this means all the other variables in the data set except the dependent variable revenue will be used in the model)

```r
# fit the model
model_mlr <- lm(revenue ~ ., data = Showtime)

# view the results with stargazer package
library(stargazer)
stargazer(model_mlr, type = "text")
```

```
##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
```

```
##                                 revenue
## -----------------------------------------------
## tv                              1.952**
##                                 (0.360)
##
## newspaper                       1.243**
##                                 (0.360)
##
## magazines                        0.311
##                                 (0.272)
##
## leaflets                         0.537
##                                 (0.487)
##
## Constant                       82.010***
##                                 (1.714)
##
## -----------------------------------------------
## Observations                       8
## R2                               0.956
## Adjusted R2                      0.898
## Residual Std. Error         0.610 (df = 3)
## F Statistic              16.397** (df = 4; 3)
## ===============================================
## Note:                 *p<0.1; **p<0.05; ***p<0.01
```

So the estimated regression equation is

$$\widehat{revenue} = 82.010 + 1.952\text{tv} + 1.243\text{newspaper} + 0.311\text{magazines} + 0.537\text{leaflets}$$

Solution c.

The two estimated coefficients of tv are different. I am not writing the interpretation but you should be able to do this (see the slides)

Solution d.

For multiple linear regression getting anova table in R requires a bit more work. If we run simple `anova(model_mlr)` function, this will not work (you can try!). We need to put two regression model in `anova()`

```r
# fit null model
model_null <- lm(revenue ~ 1, data = Showtime)

# we already have the complete model

# anova function will compare between two models
anova(model_null, model_mlr)
```

```
## Analysis of Variance Table
##
## Model 1: revenue ~ 1
## Model 2: revenue ~ tv + newspaper + magazines + leaflets
##   Res.Df     RSS Df Sum of Sq      F   Pr(>F)
## 1      7 25.5000
```

```
## 2      3  1.1153  4    24.385 16.397 0.02227 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Careful here RSS (Residual sum of squares) is actually SSE, so the anova table is showing SSE. For null model the SSE = 25.50, the Df = 7. Recall for the Null model SSE is same as SST for the full model. So for the full model the SST = 25.50, the Df = $n - 1 = 7$. Now the second row shows

- SSE = 1.1153
- Df for SSE = 3
- SSR = 24.385
- Df for SSR = 4

With this it is possible to calculate other things asked in the question (do it!) Important is here SSR = SSE_R − SSE, because SSE_R = SST.

Solution e.

The R^2 in SLR is 0.653 and the R^2 in MLR is 0.956. So the R^2 is slightly higher in MLR, this is because we have added another variable in the model, so the R^2 will increase. The adjusted R^2 in MLR model is 0.898.

Solution f.

answer of i.. Clearly in the MLR results, the `tv` and `newspaper` are significant both at 5% level of significance. But magazines and leaflets are not significant, not even at 10% level of significance.

answer of ii. For overall significance testing we need to do the F-test. Again looking the summary of the regression output, it seems we can reject the joint hypothesis that all coefficients are 0 can he rejected at 5% significance level. We can also see the value of the F-statistic and the p-value of this calculated F-statistics in the anova table (this is in answer d).

answer of iii. In this case we have two restrictions, so we need to use restricted / unrestricted F-test. The null hypothesis is

$$H_0 : \beta_3 = \beta_4 = 0$$

The alternative is same as before. Again we need to use the anova table to do the F-test. First fit the model with the restrictions, this means we need to drop the `magazines` and `leaflets` from the model.

```
model_restrict <- lm(revenue ~ tv + newspaper, data = Showtime)
```

Now compare this with the full model

```
anova( model_restrict, model_mlr)
```

```
## Analysis of Variance Table
##
## Model 1: revenue ~ tv + newspaper
## Model 2: revenue ~ tv + newspaper + magazines + leaflets
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1      5 2.0646
## 2      3 1.1153  2   0.94926 1.2766 0.3971
```

Here $SSE_R - SSE = 0.94926 $ and $Df = 2$. So the F-statistic is 1.2766 and the p-value is 0.312. So we fail to reject the null hypothesis that $\beta_3 = \beta_4 = 0$ at 5% level of significance. So we can say that magazines and leaflets are not significant at 5% level of significance. Both coefficients are zero. So perhaps we can drop them.

Solution g.

This is the same prediction problem, but careful we need to divide the numbers by 1000 because the model is in thousands of dollars.

```
test_data <- data.frame(tv = 3500/1000, newspaper = 2300/1000, magazines = 1000/1000, leaflets = 500/100

predict(model_mlr, newdata = test_data)
```

```
##        1
## 92.28007
```

Solution h.

The question is asking to provide a confidence interval for the mean of revenues when we have a value for the `tv` and `newspaper` advertisement. So we need to use the restricted model to predict (not the full model).

```
predict(model_restrict, newdata = data.frame(tv = 3500/1000, newspaper = 2300/1000), interval = "confide
```

```
##        fit      lwr      upr
## 1 94.23801 93.61968 94.85633
```

solution i.

Same question, but this time prediction interval

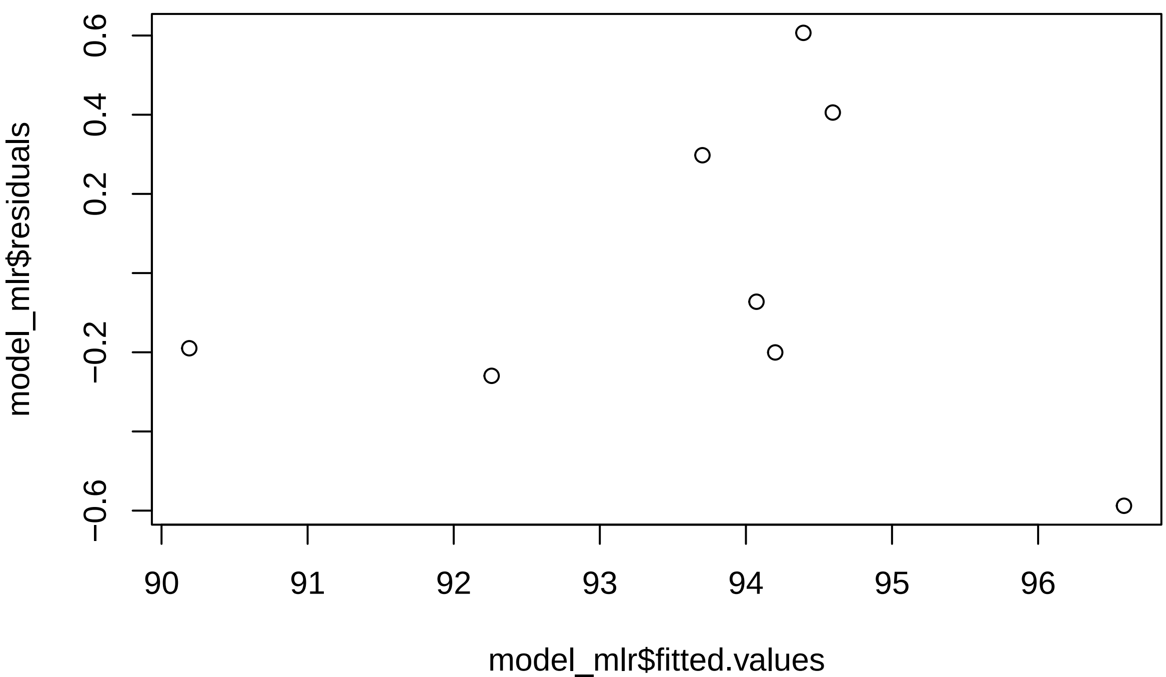
```
predict(model_restrict, newdata = data.frame(tv = 3500/1000, newspaper = 2300/1000), interval = "predict
```

```
##        fit      lwr      upr
## 1 94.23801 92.47425 96.00177
```

solution j.

Let's see the residual plot, here we need to plot the residuals against the fitted values.

```
plot(model_mlr$fitted.values, model_mlr$residuals)
```



§. Problem 7 Soln

This is a simulation exercise, simulation means the data world is in your control, you can generate the data. In this case we will generate a data with a given **Data Generating Process** (DGP). Here is the DGP

$$Y = 3 + 5X_1 + 2X_2 + \epsilon$$

We call it DGP, because if we know the how ϵ, X_1 and X_2 are generated, then we can generate the Y variable. In this case we will generate the X_1, X_2 and ϵ from given distributions. Note that here $\beta_0 = 3$, $\beta_1 = 5$ and $\beta_2 = 2$.

```r
# set the seed for reproducibility
set.seed(1238818)

# fix number of observations
n <- 50

# generate epsilon
epsilon <- rnorm(n, mean = 0, sd = sqrt(.25))
x2 <- rnorm(n, mean = 0, sd = sqrt(.35))
x1 <- runif(n, min = 0, max = 1)

# generate y
beta0 <- 3
beta1 <- 5
beta2 <- 2
y <- beta0 + beta1*x1 + beta2*x2 + epsilon

# create a data frame
sim_data <- data.frame(sales = y, tv = x1, newspaper = x2)

# view the data
sim_data
```

```
##       sales         tv   newspaper
## 1  4.488870 0.67475860 -0.88409171
## 2  4.622145 0.23868809 -0.07856125
## 3  6.758594 0.84156399 -0.53307615
## 4  4.672921 0.32841212 -0.09015324
## 5  7.531821 0.66415995  0.73722200
## 6  6.888550 0.97950962 -0.23011098
## 7  7.013012 0.54656473  0.54823801
## 8  4.179907 0.14094410  0.70350506
## 9  4.109454 0.26633477 -0.10927090
## 10 3.911798 0.07834296 -0.24842969
## 11 8.682285 0.96228672  0.17763126
## 12 4.129723 0.48018464 -0.66100598
## 13 5.294358 0.51607777 -0.07072806
## 14 3.754744 0.34645097 -0.51126184
## 15 4.108006 0.03651132  0.27954052
## 16 6.159030 0.73164819 -0.01979241
## 17 4.857479 0.36765320  0.30927280
## 18 5.363396 0.03544635  1.07404090
## 19 5.402776 0.58979273 -0.23550138
## 20 2.668866 0.32410347 -0.74674305
```

```
## 21 6.462291 0.58577540  0.29084035
## 22 4.192903 0.64075905 -1.08463652
## 23 6.139809 0.68333499 -0.20025186
## 24 7.022200 0.82843651  0.26080553
## 25 4.639332 0.13464558  0.40706327
## 26 5.577750 0.53171211 -0.07781594
## 27 6.952884 0.07929914  1.18350233
## 28 6.397108 0.64287492  0.55759626
## 29 8.057331 0.74319998  0.18020091
## 30 4.627889 0.63376057 -0.79549043
## 31 7.705306 0.51871720  0.43074176
## 32 9.096185 0.67693402  1.27519794
## 33 5.896129 0.85302258 -0.31919664
## 34 5.200500 0.41885106  0.52612226
## 35 4.927863 0.32940194  0.48290405
## 36 7.426120 0.84504846  0.10357934
## 37 5.658618 0.80828391 -0.54442856
## 38 1.883194 0.11722543 -0.96721930
## 39 5.397571 0.58166589 -0.32375446
## 40 8.388477 0.90007700  0.46233869
## 41 3.371092 0.19324682 -0.25581953
## 42 5.907344 0.51609563  0.37455582
## 43 7.224768 0.62993903  0.53770501
## 44 5.412905 0.43644514 -0.03660434
## 45 6.359813 0.52871944  0.28302049
## 46 6.956927 0.75649477 -0.14057255
## 47 6.887140 0.02564032  1.36090442
## 48 5.003667 0.23858327  0.13021873
## 49 5.259506 0.91529906 -0.94328946
## 50 2.863081 0.04504938 -0.39662903
```

Since we need to generate the data for $n = 50$, then $n = 1000$, then $n = 300$ and $n = 500$, rather than copying and pasting we will write a function first and generate the data just by changing the value of n

```r
# function to generate the data

generate_data <- function(n){
  # generate epsilon
  epsilon <- rnorm(n, mean = 0, sd = sqrt(.25))
  x2 <- rnorm(n, mean = 0, sd = sqrt(.35))
  x1 <- runif(n, min = 0, max = 1)

  # generate y
  y <- 3 + 5*x1 + 2*x2 + epsilon

  # create a data frame
  sim_data <- data.frame(y, x1, x2)

  # return the data
  return(sim_data)
}
```

So every time we call this function it will return a data frame for us.

```r
# generate the data for n = 50
# give the data a name
```

```r
data_50 <- generate_data(50)

# see the data
data_50
```

```
##           y           x1          x2
## 1  7.160021 0.8643315108 -0.19719144
## 2  9.734567 0.9969846567  0.58983081
## 3  7.991663 0.6065868419  1.00988904
## 4  6.169271 0.6150654876  0.07433327
## 5  4.942380 0.3112173709 -0.07363607
## 6  3.040282 0.2318498383 -0.55072124
## 7  6.053363 0.5054107767  0.09598662
## 8  4.010350 0.1884302876  0.02694795
## 9  7.921904 0.6750195345  0.68835958
## 10 7.405118 0.5041511960  0.77197236
## 11 7.389173 0.5789139955  0.78967562
## 12 4.347276 0.4874288528 -0.35773069
## 13 5.709370 0.3924997861  0.30472539
## 14 2.000691 0.4026356814 -1.61490029
## 15 6.494485 0.3747238233  0.81481434
## 16 5.466603 0.1296375915  1.25576296
## 17 6.228939 0.6780066260 -0.08746064
## 18 6.505348 0.4280622276  0.63755284
## 19 8.548070 0.9217563251  0.08847848
## 20 6.046780 0.6598597541 -0.18164946
## 21 7.414852 0.6765351992  0.40062186
## 22 4.802390 0.2228303547  0.39143820
## 23 3.772313 0.2202511090  0.05416289
## 24 6.289815 0.5026718231  0.39166579
## 25 6.870056 0.6968779189 -0.04820738
## 26 3.700308 0.0770797962  0.15239624
## 27 3.064961 0.0514335553  0.07420946
## 28 3.549345 0.2654457928 -0.24022771
## 29 8.432483 0.6817292878  0.82744826
## 30 4.365981 0.1752574323  0.18817417
## 31 4.801454 0.0008933423  0.76074385
## 32 4.425263 0.1407824385  0.10744378
## 33 2.435871 0.1311974372 -0.42757508
## 34 4.767181 0.5122128390  0.14078620
## 35 8.822670 0.9512334329  0.55533630
## 36 3.847990 0.3305822532 -0.22609786
## 37 5.568301 0.5961034803 -0.23829744
## 38 8.049991 0.9791409415  0.04746287
## 39 5.075841 0.7545945263 -0.74532186
## 40 6.015720 0.6723935523  0.09361944
## 41 8.098380 0.9895856185 -0.06795263
## 42 3.034823 0.0246400333 -0.05446336
## 43 3.431211 0.5167486565 -0.78645788
## 44 8.071740 0.8496184449  0.80016113
## 45 2.962801 0.1994201830 -0.52511673
## 46 5.591551 0.6358970397 -0.42060792
## 47 5.024179 0.0855034578  0.79432019
## 48 8.304139 0.8475595561  0.26167367
```

```
## 49 5.239059 0.3890425721 -0.07001134
## 50 8.412478 0.7185553380  0.72129103
```

Let's do it for other values of n.

```r
# generate the data for n = 1000
data_100 <- generate_data(100)

# generate the data for n = 300
data_300 <- generate_data(300)

# generate the data for n = 500
data_500 <- generate_data(500)
```

Now finally we will fit the model for each of the data set and see the results.

```r
# fit the model for n = 50
model_50 <- lm(y ~ x1 + x2, data = data_50)
summary(model_50)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data_50)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10598 -0.19676  0.08032  0.23143  0.60772
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.87250    0.10523   27.30   <2e-16 ***
## x1           5.30474    0.18596   28.53   <2e-16 ***
## x2           2.01375    0.09929   20.28   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3706 on 47 degrees of freedom
## Multiple R-squared:  0.9648, Adjusted R-squared:  0.9633
## F-statistic: 644.8 on 2 and 47 DF,  p-value: < 2.2e-16
```

```r
# fit the model for n = 1000
model_100 <- lm(y ~ x1 + x2, data = data_100)
summary(model_100)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data_100)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.12078 -0.38419  0.01002  0.36810  1.13314
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.99315    0.11044   27.10   <2e-16 ***
## x1           4.93396    0.18659   26.44   <2e-16 ***
## x2           2.04248    0.08212   24.87   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5164 on 97 degrees of freedom
## Multiple R-squared:  0.9264, Adjusted R-squared:  0.9249
## F-statistic: 610.4 on 2 and 97 DF,  p-value: < 2.2e-16
```

```r
# fit the model for n = 300
model_300 <- lm(y ~ x1 + x2, data = data_300)
summary(model_300)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data_300)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.62477 -0.31072  0.00712  0.31896  1.17742
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.91282    0.05883   49.51   <2e-16 ***
## x1           5.07259    0.09549   53.12   <2e-16 ***
## x2           2.05007    0.04601   44.56   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4787 on 297 degrees of freedom
## Multiple R-squared:  0.9429, Adjusted R-squared:  0.9425
## F-statistic:  2451 on 2 and 297 DF,  p-value: < 2.2e-16
```

```r
# fit the model for n = 500
model_500 <- lm(y ~ x1 + x2, data = data_500)
summary(model_500)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = data_500)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.31164 -0.32046 -0.01006  0.31560  1.27403
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.05733    0.04150   73.67   <2e-16 ***
## x1           4.92437    0.07255   67.87   <2e-16 ***
## x2           1.99314    0.03658   54.48   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4784 on 497 degrees of freedom
## Multiple R-squared:  0.9373, Adjusted R-squared:  0.9371
## F-statistic:  3715 on 2 and 497 DF,  p-value: < 2.2e-16
```

For any fitted results we can also get the estimated coefficients by using the following command,

```r
# get the estimated coefficients for n = 50
model_50$coefficients
```

```
## (Intercept)          x1          x2
##    2.872500    5.304738    2.013745
```

Now we will generate the data for sample size $n = 50$, 1000 times, fit the model and save $\hat{\beta}_1$ and plto the histogram.

```r
# set the seed for reproducibility
set.seed(1238818)

# create an empty vector to save the results
beta_1_hat_50 <- c()

# generate the data for n = 50, 1000 times
for(i in 1:1000){
  # generate the data
  data_50 <- generate_data(50)

  # fit the model
  model_50 <- lm(y ~ x1 + x2, data = data_50)

  # save the results
  beta_1_hat_50[i] <- model_50$coefficients[2]
}
```

Now we have all the estimated coefficients for $\hat{\beta}_1$ saved in the vector `beta_1_hat`, there will be 1000 of them, since we did repeated sampling 1000 times. Also note, sample size was 50 each times. Let's plot the histogram.

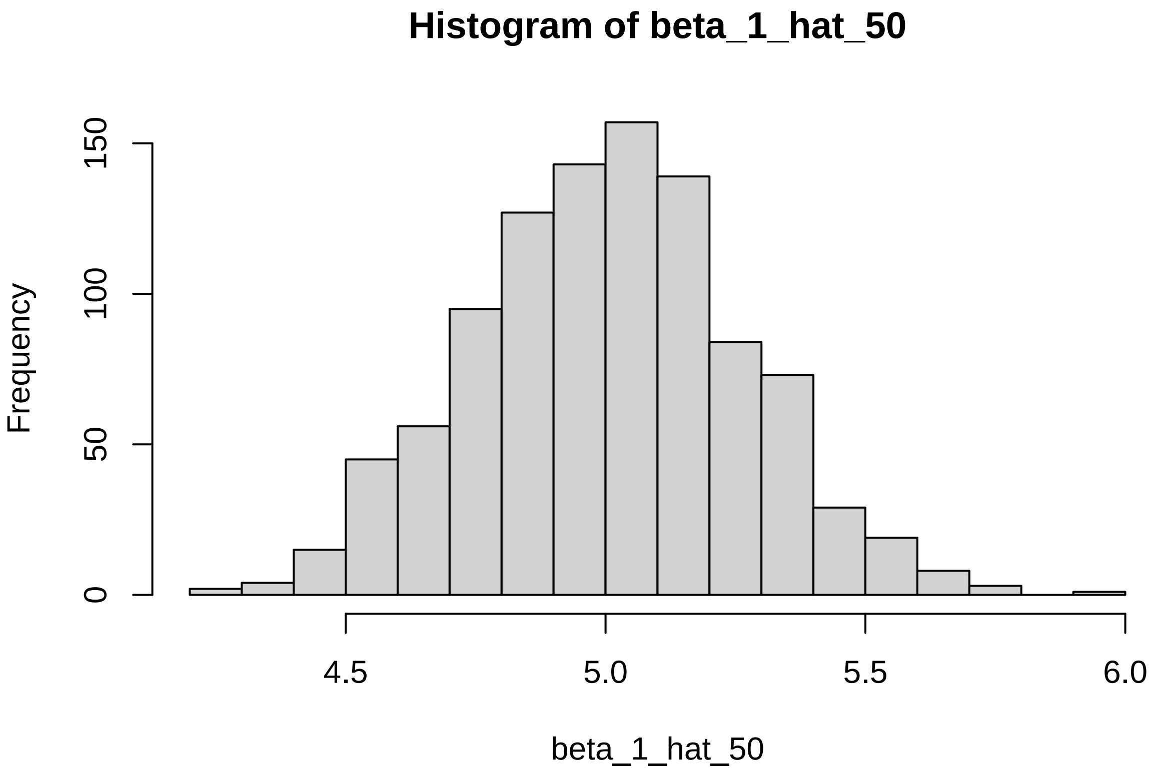
```r
beta_1_hat_50 # there will be 1000 of the beta1hat from 1000 times repeated sampling
```

```
##    [1] 4.485478 5.304738 4.959747 5.251674 4.895635 5.101398 4.754430 5.054136
##    [9] 5.306305 5.236520 4.901062 4.991120 5.166903 5.254400 5.007742 5.168038
##   [17] 4.743377 4.666466 5.135399 4.790026 4.898384 5.093273 5.101810 4.836331
##   [25] 4.437754 4.828734 5.081656 5.162919 4.781128 4.649088 5.181499 5.069309
##   [33] 4.987362 5.620613 5.013005 5.116759 4.665046 4.862916 4.757111 4.997230
##   [41] 5.093257 4.670814 5.486029 5.231187 5.063979 5.250513 5.085916 4.924652
##   [49] 5.347803 5.268144 4.563011 4.908670 5.452769 4.518165 4.598946 5.317104
##   [57] 4.928645 5.285162 4.812861 5.078343 4.714498 4.790625 4.832180 4.642192
##   [65] 4.696364 4.898946 4.822505 4.919054 5.172281 5.204671 4.976395 4.921758
##   [73] 4.866372 4.761117 4.646155 5.493639 5.326390 5.067666 4.902544 4.786867
##   [81] 4.850499 5.553196 4.865611 4.733506 5.062203 4.855864 5.367243 5.187651
##   [89] 4.834076 5.056856 4.929961 5.397157 5.027832 4.582912 5.038255 5.386291
##   [97] 5.088605 4.954704 4.737006 5.112566 5.399548 5.163330 4.889125 5.071413
##  [105] 5.252471 5.158623 5.345738 5.141599 4.672340 4.980025 4.994469 4.905299
##  [113] 5.374069 4.629335 4.885723 4.806755 4.962482 4.887558 5.065308 4.597303
##  [121] 4.717842 5.102193 4.959834 5.132458 5.556404 4.746247 5.381661 4.892779
##  [129] 5.153609 5.583849 4.508429 4.925393 5.031795 5.537385 5.143640 5.098437
##  [137] 4.840591 4.568493 4.848978 5.260674 4.993063 5.111483 4.756304 5.084543
##  [145] 5.085331 4.621612 5.126513 4.868906 4.927173 4.802615 4.816373 4.769819
##  [153] 4.698057 4.757790 5.071649 4.994194 5.018386 5.094399 5.291332 5.261723
##  [161] 4.833200 5.337813 5.318991 4.708214 4.643773 5.368306 5.053251 4.843891
##  [169] 5.207936 4.872206 5.185777 5.130650 4.880765 4.874002 4.908826 5.069372
##  [177] 5.236570 4.901159 4.869120 4.724201 4.769751 4.979136 5.051217 5.076313
```

```
## [185] 5.249442 5.141515 4.580691 4.836354 5.752185 4.668689 5.183913 5.374048
## [193] 5.364288 4.776408 5.179440 4.548132 5.070780 5.092629 5.098910 4.991300
## [201] 5.240991 5.132287 5.300620 5.163879 4.844649 5.018457 5.201190 5.206082
## [209] 5.228265 5.262440 4.857969 4.934296 4.772525 4.906537 4.461187 5.227305
## [217] 5.109712 5.055504 5.141304 5.015812 4.920844 5.186784 5.040651 5.442981
## [225] 5.175959 4.455601 4.873602 4.952239 5.105042 4.525350 4.904833 5.179180
## [233] 5.067308 4.930890 4.699570 4.762481 4.844586 4.823260 4.833469 4.826104
## [241] 5.232760 4.966485 4.608799 4.623180 4.920504 4.648176 4.802458 4.531290
## [249] 5.251741 5.069689 4.269026 4.532997 5.285251 5.350430 5.170922 4.898040
## [257] 5.135269 4.949178 5.197332 4.889503 4.949457 4.842371 5.420308 5.032928
## [265] 5.061658 4.849542 4.824760 5.026706 4.857946 4.993099 4.726667 4.824758
## [273] 4.749239 5.039691 5.057484 5.132294 5.256295 5.253414 4.773786 4.836962
## [281] 4.381831 4.381285 4.937885 5.007333 5.460112 5.143479 4.810318 4.918331
## [289] 4.741700 4.801868 4.903011 5.052805 5.025423 4.658228 5.264722 5.010361
## [297] 5.199477 4.689741 5.159770 5.613585 4.999266 5.374725 5.448959 5.130768
## [305] 5.172403 5.165188 5.245571 4.684331 5.185454 4.944623 4.798659 4.537141
## [313] 4.979890 4.845987 5.566050 5.370217 4.866928 5.132236 4.766464 4.934627
## [321] 5.277240 5.212898 5.181595 4.728540 5.645782 4.894996 5.025202 4.986876
## [329] 5.429623 4.773761 4.710793 5.233495 5.184020 4.902078 5.181114 5.487995
## [337] 4.993412 5.373381 4.988172 5.287826 4.917209 4.995167 4.938579 4.871048
## [345] 5.301386 4.826808 4.554810 4.970244 5.156826 5.358136 4.855491 4.855747
## [353] 4.979132 5.477457 4.872498 5.194213 4.834971 5.065981 5.079478 4.957526
## [361] 4.519404 5.053165 4.679644 5.308541 4.829895 5.303690 5.397950 5.037411
## [369] 4.920102 4.881469 5.054026 4.503253 5.088003 5.470697 5.068837 4.582705
## [377] 5.255538 4.450715 4.785105 5.080632 5.148690 4.562165 5.195823 4.754681
## [385] 4.777834 5.038562 4.935939 5.198320 5.160737 5.100383 4.542710 4.938678
## [393] 5.357185 5.097343 4.832971 4.855398 4.917219 5.174243 4.586963 5.118714
## [401] 4.996982 5.145911 5.278410 5.333532 5.031475 5.017665 5.022780 4.684846
## [409] 4.932596 5.510468 4.642632 4.565029 4.682752 5.381128 4.837027 5.051457
## [417] 5.298639 4.766839 5.361065 5.097074 5.287559 4.910771 5.360722 4.740435
## [425] 5.036110 5.133384 4.777498 5.435280 4.907954 4.824628 5.035225 4.654535
## [433] 5.240856 4.824306 5.078145 5.155043 4.901482 5.046447 5.109161 5.125047
## [441] 5.125059 4.829513 5.292919 5.296713 5.044832 5.300289 5.116186 5.459428
## [449] 4.930736 5.012760 5.715140 4.970881 4.885448 4.681220 4.971130 5.176769
## [457] 5.123961 5.576833 4.815896 5.151657 4.892834 5.045430 5.225182 4.945834
## [465] 4.594349 5.049462 5.134431 4.734553 4.798456 5.264664 5.054971 5.058237
## [473] 4.703205 4.563755 5.124378 4.901899 4.720079 4.974602 4.857145 4.640791
## [481] 5.718206 4.975588 5.302425 4.577156 4.667167 4.971344 4.662398 4.747257
## [489] 5.061702 4.596287 5.074772 5.185422 5.037521 4.992930 4.982168 4.746163
## [497] 5.011922 5.009470 4.873124 4.911402 4.995777 5.012149 5.056535 4.735705
## [505] 4.910249 4.882824 4.676207 5.132500 5.116744 4.809766 5.058672 4.988212
## [513] 4.895384 4.424062 4.593222 4.959165 4.972550 4.997310 5.160204 5.126934
## [521] 4.747962 4.832251 4.817506 5.213387 4.937497 5.051790 4.727891 5.046995
## [529] 5.001643 5.090624 4.587650 5.140236 4.853415 4.881284 5.210002 5.131030
## [537] 5.011962 5.159495 4.496885 4.895233 5.134878 4.659649 5.094232 4.944533
## [545] 5.007720 5.204960 4.808499 4.858506 4.500230 4.986009 4.856290 5.134735
## [553] 5.207438 4.573706 5.121951 4.970375 4.953576 5.475451 5.319224 5.060667
## [561] 4.614044 5.205308 5.000751 5.240097 4.621917 5.371268 5.302424 4.773791
## [569] 5.366924 5.129211 5.029783 5.141469 5.300557 5.366798 4.828968 5.283011
## [577] 5.129549 5.080294 4.920634 4.973983 5.371933 5.515251 4.922464 4.940064
## [585] 5.292185 5.188624 5.074337 4.934006 5.320070 4.991679 4.952876 5.083044
## [593] 5.341746 5.168061 4.789659 4.571647 5.581421 4.871725 5.239086 4.727561
## [601] 5.055505 4.736164 5.286184 5.255170 4.600747 4.809100 5.199217 5.082497
## [609] 4.847345 4.809383 5.113691 4.708357 5.008076 4.846232 5.088397 4.855276
```

```
## [617] 4.884021 4.856612 4.997421 5.515449 5.126846 4.792163 4.961163 5.094427
## [625] 4.811079 4.885611 4.537607 5.002173 4.715811 4.727964 5.547906 5.517590
## [633] 5.103533 5.169705 4.615584 4.771699 5.556764 4.695611 5.068552 4.270519
## [641] 5.192190 5.084283 5.008164 4.972422 4.856107 5.387147 4.613194 4.897642
## [649] 4.402707 5.152640 4.985800 4.789159 5.062928 5.135107 5.166197 4.980036
## [657] 5.047420 5.156798 5.389144 5.244218 4.947095 5.454278 5.465370 5.342624
## [665] 5.600024 5.268523 4.757993 5.104025 4.977780 5.396636 5.316958 4.887372
## [673] 5.025701 4.668983 4.848392 4.796946 4.908208 5.187225 4.749204 5.142773
## [681] 4.890062 4.622630 5.025661 4.824953 4.553671 5.214686 5.267459 5.336312
## [689] 5.258944 4.871090 4.610731 5.192959 5.148209 4.910966 4.521401 4.921669
## [697] 4.973159 4.785430 5.217412 5.102504 4.805551 4.746484 5.326392 4.840600
## [705] 5.225079 5.099077 4.535507 4.845539 4.742769 5.017666 4.741661 4.827594
## [713] 5.051057 4.896021 4.963346 4.822602 5.016697 5.317157 4.723515 4.816076
## [721] 4.754528 4.657783 4.722018 5.229558 5.039458 4.757617 4.709079 5.003111
## [729] 5.187133 5.385466 5.516635 5.195196 5.058186 5.013071 5.209867 5.248617
## [737] 5.198047 4.748710 5.111081 5.287853 5.077555 4.858578 4.891701 5.575714
## [745] 5.455877 5.012015 5.024091 4.911574 4.784502 4.928050 5.154982 5.331623
## [753] 5.428720 5.006266 4.992775 5.002264 5.653871 5.353800 5.068550 5.201087
## [761] 5.573610 4.931420 5.210005 5.084679 5.064386 5.274057 4.733860 5.251118
## [769] 4.936122 4.482423 5.069234 4.906082 5.061590 5.659901 4.460655 5.434459
## [777] 4.957917 5.598068 5.113459 5.073720 4.726930 5.071080 5.483408 5.497302
## [785] 5.373398 4.633976 5.033450 5.263387 4.300644 4.557961 5.546296 4.790253
## [793] 4.586537 5.213404 5.330883 4.601047 4.972378 5.042581 5.134534 4.607962
## [801] 5.136604 5.987094 5.172229 5.293352 4.975573 4.886609 4.931155 4.488223
## [809] 5.119127 4.825748 5.245991 5.147783 5.515031 5.111311 4.988581 4.540124
## [817] 4.500462 5.055904 5.003116 4.670038 4.460839 5.086395 4.733326 5.372004
## [825] 4.786599 5.067048 4.675092 5.122879 5.248668 5.126275 4.823130 4.536572
## [833] 4.728632 4.726170 5.385931 5.005362 4.996158 4.997981 4.865040 4.939332
## [841] 4.966626 4.975067 5.304325 5.008067 5.205826 4.885080 5.043048 5.379153
## [849] 4.637116 4.704769 5.097440 5.382212 4.732754 5.195657 4.850238 5.086946
## [857] 5.026773 4.609905 5.210246 5.175134 5.157766 5.155974 4.768472 4.558817
## [865] 5.353287 5.105809 5.130621 5.066071 4.701898 5.025615 4.994343 5.198548
## [873] 5.322911 5.472932 4.887698 5.179708 4.853338 5.013853 5.139585 4.818475
## [881] 5.472540 4.931023 5.398423 4.849495 4.718906 5.172996 4.933374 5.299378
## [889] 5.488836 4.823664 5.230227 4.664390 4.923887 4.989167 5.026607 5.358095
## [897] 4.971740 4.836640 4.585121 5.324230 5.091373 4.799231 4.490879 5.077438
## [905] 5.060028 5.109613 5.431736 5.053831 5.440969 4.879381 4.967715 4.821120
## [913] 4.743480 4.861706 5.380455 4.639657 5.102443 4.726643 5.362494 4.857883
## [921] 4.979018 4.952605 4.662066 5.118392 4.425176 4.546790 4.497086 5.286347
## [929] 4.753523 5.198947 5.033935 5.233569 5.095337 5.288266 4.774368 5.072913
## [937] 4.717581 5.472029 5.074954 4.944193 4.745364 5.250638 4.628851 5.256483
## [945] 4.616283 5.362248 4.971833 5.102638 4.963357 5.228972 4.582668 5.309639
## [953] 4.975145 4.970081 5.311372 4.773843 5.347762 4.379886 5.120061 5.407706
## [961] 5.170946 4.949518 5.198010 4.951369 5.072183 4.944847 5.332221 5.025111
## [969] 4.612138 5.023289 5.131394 5.127251 4.577943 5.043374 4.789041 5.215886
## [977] 5.075715 5.123780 4.937445 4.922138 5.637467 5.239811 5.665027 4.801432
## [985] 4.973035 4.954486 4.864773 4.742852 4.771484 5.163935 4.677989 4.762401
## [993] 5.156840 5.480446 4.666277 5.178436 4.933347 5.091997 4.787239 4.548936
```

```r
# plot the histogram of 1000s beta1hats
hist(beta_1_hat_50, breaks = 20)
```



This is the sampling distribution of $\hat{\beta}_1$ for $n = 50$. Notice it looks like Normal distribution. We can also calculate the standadrd error from this simulation, which is the standard deviation of the sampling distribution.

```
# calculate the standard error
sd(beta_1_hat_50)
```

```
## [1] 0.2604082
```

In this way you can also get the sampling distribution for $\hat{\beta}_0$ and $\hat{\beta}_2$. Note you need to use `model_50$coefficients[0]` command for $\hat{\beta}_0$ and `model_50$coefficients[3]` command for $\hat{\beta}_2$.

Now will do the same for $n = 1000$ and compare the sampling distribution of $\hat{\beta}_1$ for $n = 50$ and $n = 1000$.

```
# set the seed for reproducibility
set.seed(1238818)

# create an empty vector to save the results
beta_1_hat_1000 <- c()

# generate the data for n = 500, 1000 times
for(i in 1:1000){
  # generate the data
  data_1000 <- generate_data(1000)

  # fit the model
  model_1000 <- lm(y ~ x1 + x2, data = data_1000)

  # save the results
  beta_1_hat_1000[i] <- model_1000$coefficients[2]
}
```

let's calculate the standard error.

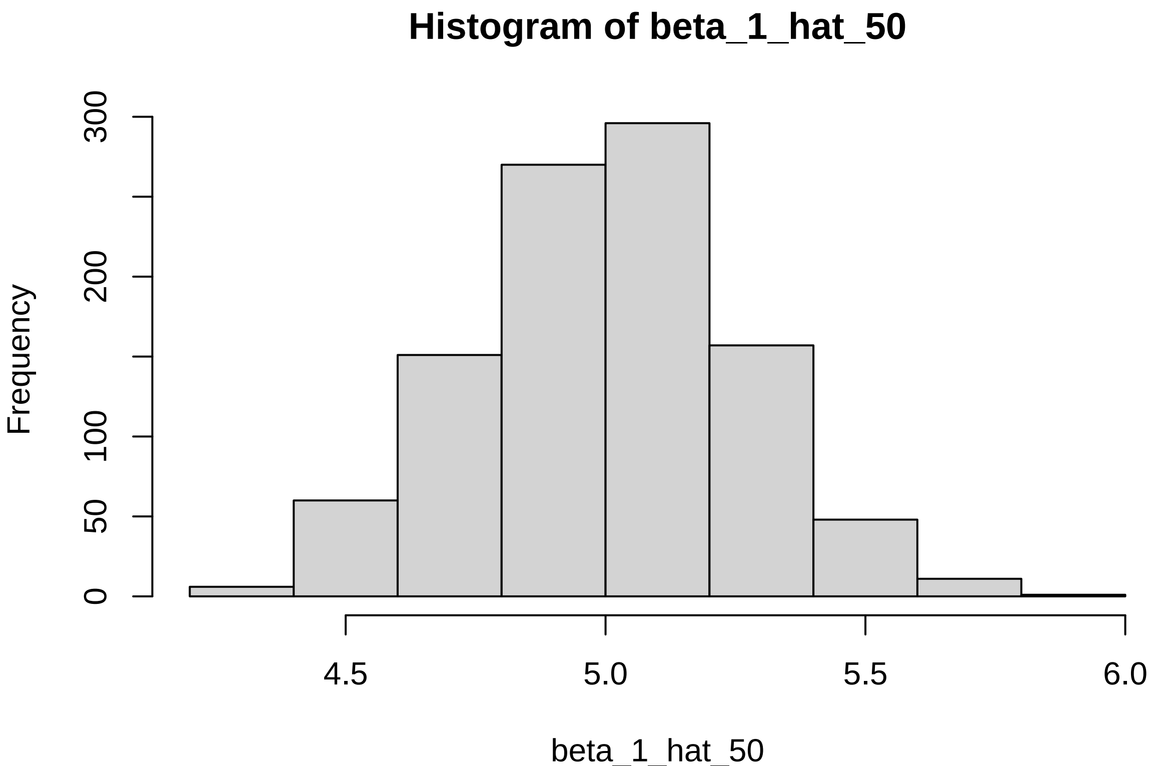
```
# calculate the standard error
sd(beta_1_hat_1000)
```

```
## [1] 0.05418711
```

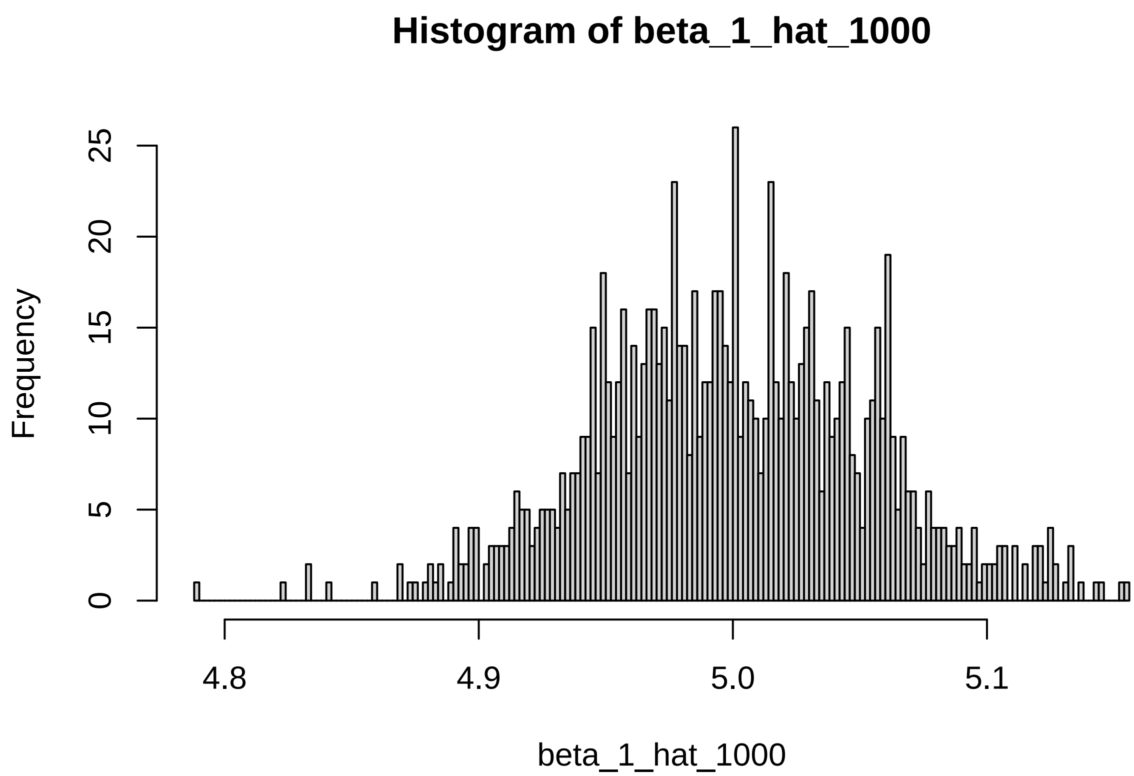
Notice the standard error significantly reduced. This is the similar story like sample means, as we have more and more data, we have more accurate estimate.

Now let's compare the two histograms

```
# plot the histogram for n = 50
hist(beta_1_hat_50, breaks = 10)
```




```
# plot the histogram for n = 1000
hist(beta_1_hat_1000, breaks = 200)
```



Notice the histogram for $n = 1000$ is much more concentrated around the true value of $\beta_1 = 5$.

§. Problem 8 Soln

We can include the interaction term by running following command,

```
# fit the model
model_interaction <- lm(revenue ~ tv*newspaper, data = Showtime)

# view the results with stargazer package
library(stargazer)
stargazer(model_interaction, type = "text")
```

```
##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                               revenue
## -----------------------------------------------
## tv                             1.089
##                               (1.020)
##
## newspaper                     -0.667
##                               (1.631)
##
## tv:newspaper                   0.724
##                               (0.589)
##
## Constant                     86.531***
##                               (3.077)
```

```
##
## -----------------------------------------------
## Observations                     8
## R2                             0.941
## Adjusted R2                    0.897
## Residual Std. Error       0.612 (df = 4)
## F Statistic            21.349*** (df = 3; 4)
## ===============================================
## Note:               *p<0.1; **p<0.05; ***p<0.01
```

Again we can write the estimated model as,

$$\widehat{revenue} = 86.531 - 1.089 \times tv - 0.667 \times newspaper + 0.724 \times (tv \times newspaper)$$

Now let's interpret the results. This is an interaction model, so we need to be careful. First let's slightly rewrite estimated model as

$$\widehat{revenue} = 86.531 + (-1.089 + .724\, newspaper) \times tv - (0.667 \times newspaper)$$

This means if we increase the TV advertisement by 1 unit, the revenue is predicted to increase by $-1.089 + .724\, newspaper$ units. Notice this is a function of newspaper advertisement. So the effect of TV advertisement on revenue depends on the level of newspaper advertisement.

§. Problem 9 Soln

This is for the auto data set that you already saw before. In this task we will do some non-linear regression, in particular polynomial regression with given degree. Interestingly we will see that the polynomial regression is actually a special case of multiple linear regression.

Solution a.

The first task is same, load the data.

```r
# load the library, load the data
library(readxl)
Auto <- read_excel("/home/tanvir/Documents/ownCloud/Git_Repos/EWU_repos/3_Fall_2023/eco_204/ewu-eco204.g
```

```r
summary(Auto)
```

```
##       mpg          cylinders      displacement    horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Length:397
##  1st Qu.:17.50   1st Qu.:4.000   1st Qu.:104.0   Class :character
##  Median :23.00   Median :4.000   Median :146.0   Mode  :character
##  Mean   :23.52   Mean   :5.458   Mean   :193.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:262.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0
##      weight      acceleration        year           origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
##  1st Qu.:2223   1st Qu.:13.80   1st Qu.:73.00   1st Qu.:1.000
##  Median :2800   Median :15.50   Median :76.00   Median :1.000
##  Mean   :2970   Mean   :15.56   Mean   :75.99   Mean   :1.574
##  3rd Qu.:3609   3rd Qu.:17.10   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
##      name
```

```
##  Length:397
##  Class :character
##  Mode  :character
##
##
##
```

surprisingly horsepower is showing some character, we need to fix it.

Little bit of data cleaning (You can ignore this part)

The summary stats show there is a `char` which means character or text in horepower. This is a problem. let's see the data for horsepower,

```
sort(Auto$horsepower)
```

```
##   [1] "?"   "?"   "?"   "?"   "?"   "100" "100" "100" "100" "100" "100" "100"
##  [13] "100" "100" "100" "100" "100" "100" "100" "100" "100" "100" "102" "103"
##  [25] "105" "105" "105" "105" "105" "105" "105" "105" "105" "105" "105" "105"
##  [37] "107" "108" "110" "110" "110" "110" "110" "110" "110" "110" "110" "110"
##  [49] "110" "110" "110" "110" "110" "110" "110" "110" "112" "112" "112" "113"
##  [61] "115" "115" "115" "115" "115" "116" "120" "120" "120" "120" "122" "125"
##  [73] "125" "125" "129" "129" "130" "130" "130" "130" "130" "132" "133" "135"
##  [85] "137" "138" "139" "139" "140" "140" "140" "140" "140" "140" "140" "142"
##  [97] "145" "145" "145" "145" "145" "145" "145" "148" "149" "150" "150" "150"
## [109] "150" "150" "150" "150" "150" "150" "150" "150" "150" "150" "150" "150"
## [121] "150" "150" "150" "150" "150" "150" "150" "152" "153" "153" "155" "155"
## [133] "158" "160" "160" "165" "165" "165" "165" "167" "170" "170" "170" "170"
## [145] "170" "175" "175" "175" "175" "175" "180" "180" "180" "180" "180" "190"
## [157] "190" "190" "193" "198" "198" "200" "208" "210" "215" "215" "215" "220"
## [169] "225" "225" "225" "230" "46"  "46"  "48"  "48"  "48"  "49"  "52"  "52"
## [181] "52"  "52"  "53"  "53"  "54"  "58"  "58"  "60"  "60"  "60"  "60"  "60"
## [193] "61"  "62"  "62"  "63"  "63"  "63"  "64"  "65"  "65"  "65"  "65"  "65"
## [205] "65"  "65"  "65"  "65"  "65"  "66"  "67"  "67"  "67"  "67"  "67"  "67"
## [217] "67"  "67"  "67"  "67"  "67"  "67"  "68"  "68"  "68"  "68"  "68"  "68"
## [229] "69"  "69"  "69"  "70"  "70"  "70"  "70"  "70"  "70"  "70"  "70"  "70"
## [241] "70"  "70"  "70"  "71"  "71"  "71"  "71"  "71"  "72"  "72"  "72"  "72"
## [253] "72"  "72"  "74"  "74"  "74"  "75"  "75"  "75"  "75"  "75"  "75"  "75"
## [265] "75"  "75"  "75"  "75"  "75"  "75"  "75"  "76"  "76"  "76"  "76"  "77"
## [277] "78"  "78"  "78"  "78"  "78"  "78"  "79"  "79"  "80"  "80"  "80"  "80"
## [289] "80"  "80"  "80"  "81"  "81"  "82"  "83"  "83"  "83"  "83"  "84"  "84"
## [301] "84"  "84"  "84"  "84"  "85"  "85"  "85"  "85"  "85"  "85"  "85"  "85"
## [313] "85"  "86"  "86"  "86"  "86"  "86"  "87"  "87"  "88"  "88"  "88"  "88"
## [325] "88"  "88"  "88"  "88"  "88"  "88"  "88"  "88"  "88"  "88"  "88"  "88"
## [337] "88"  "88"  "88"  "89"  "90"  "90"  "90"  "90"  "90"  "90"  "90"  "90"
## [349] "90"  "90"  "90"  "90"  "90"  "90"  "90"  "90"  "90"  "90"  "90"  "90"
## [361] "91"  "92"  "92"  "92"  "92"  "92"  "92"  "93"  "94"  "95"  "95"  "95"
## [373] "95"  "95"  "95"  "95"  "95"  "95"  "95"  "95"  "95"  "95"  "95"  "96"
## [385] "96"  "96"  "97"  "97"  "97"  "97"  "97"  "97"  "97"  "97"  "97"  "98"
## [397] "98"
```

there is a actually a question mark, this is actually missing data. We need to fix it. Following command will remove the question marks,

```
# there are many ways also with a package dplyr, but here is a simple way to do this
Auto <-  Auto[Auto$horsepower != "?", ]
```

However still it's character, we need to convert it to numeric, the function in R is `as.numeric()`, for example

```r
"134"
```

```
## [1] "134"
```

```r
as.numeric("134")
```

```
## [1] 134
```

Let's now do it for horsepower

```r
Auto$horsepower <- as.numeric(Auto$horsepower)
```

Now let's see the summary statistics again

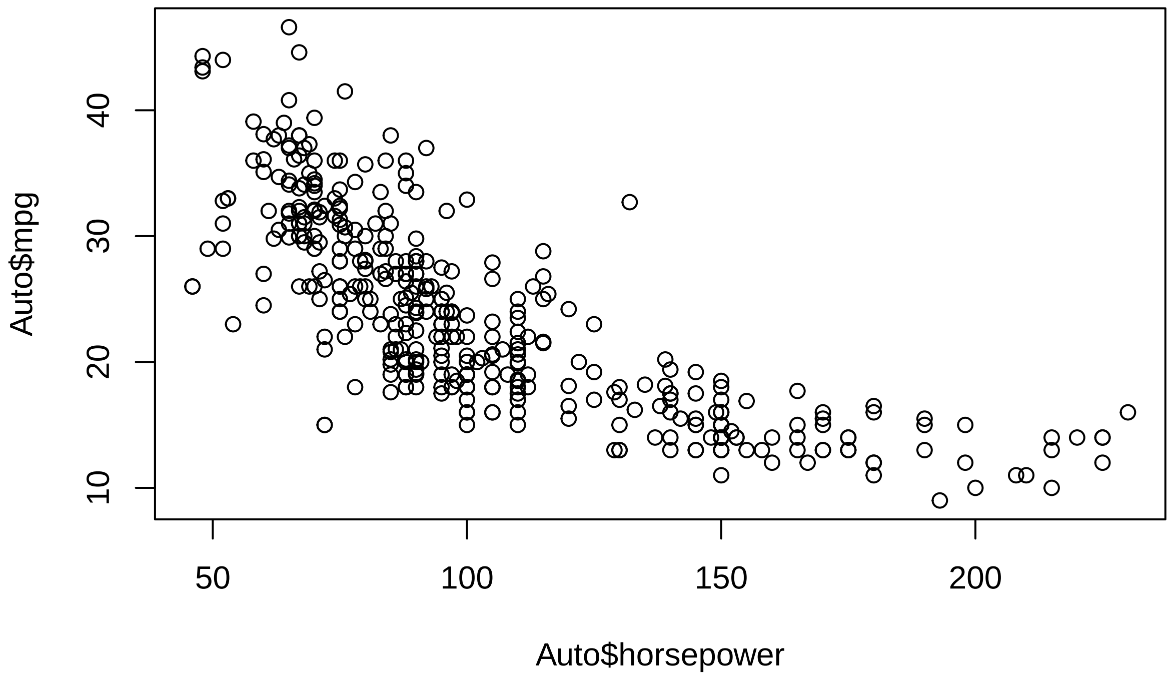
```r
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##   acceleration        year           origin          name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   Length:392
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   Class :character
##  Median :15.50   Median :76.00   Median :1.000   Mode  :character
##  Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :24.80   Max.   :82.00   Max.   :3.000
```

Looks good

Let's plot the data,

```r
plot(Auto$horsepower, Auto$mpg)
```



So the pattern shows some non-linearities

Running Polynomial Regression

Now let's perfoem polynomial regression. We will use the `I()` to incorporate polynomials in `lm()`

```
quad_fit <- lm(mpg ~ horsepower + I(horsepower^2), data = Auto)

library(stargazer)
stargazer(quad_fit, type = "text")
```

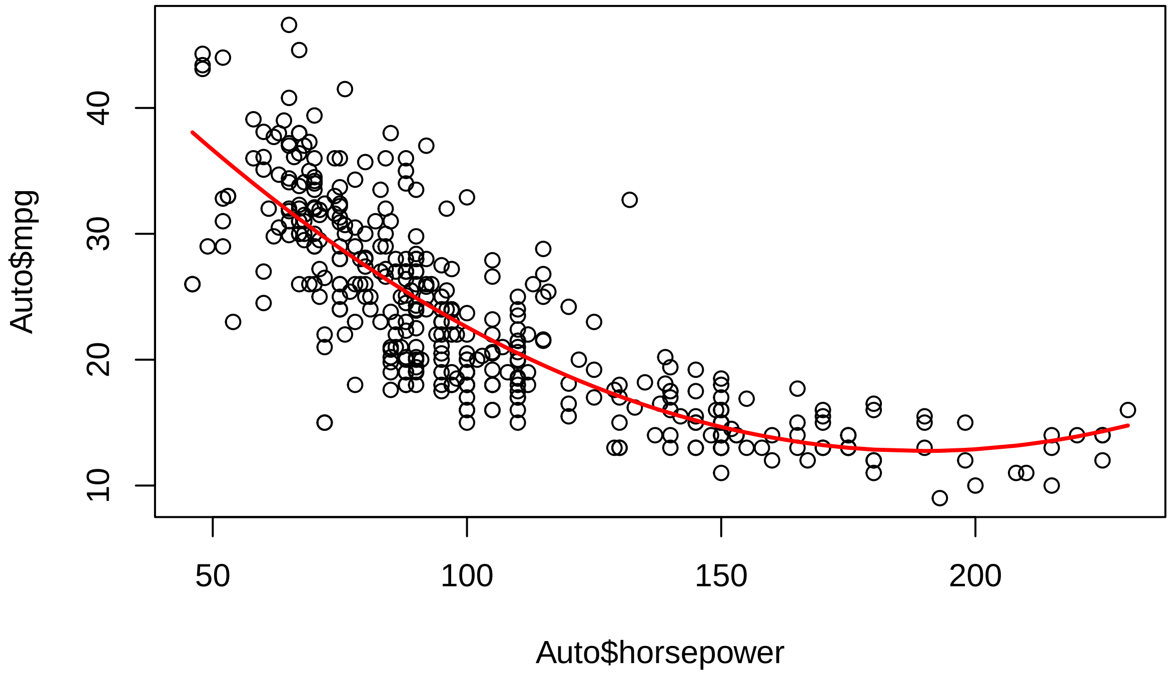
```
##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                                 mpg
## -----------------------------------------------
## horsepower                   -0.466***
##                               (0.031)
##
## I(horsepower2)                0.001***
##                              (0.0001)
##
## Constant                     56.900***
##                               (1.800)
##
## -----------------------------------------------
## Observations                    392
## R2                             0.688
## Adjusted R2                    0.686
## Residual Std. Error      4.374 (df = 389)
## F Statistic          428.018*** (df = 2; 389)
## ===============================================
```

```
## Note:                  *p<0.1; **p<0.05; ***p<0.01
```

Plot the fitted line

```r
plot(Auto$horsepower, Auto$mpg)

lines(sort(Auto$horsepower), fitted(quad_fit)[order(Auto$horsepower)], col = "red", lwd = 2)
```



What if we want to perform cubic regression? We can do that too,

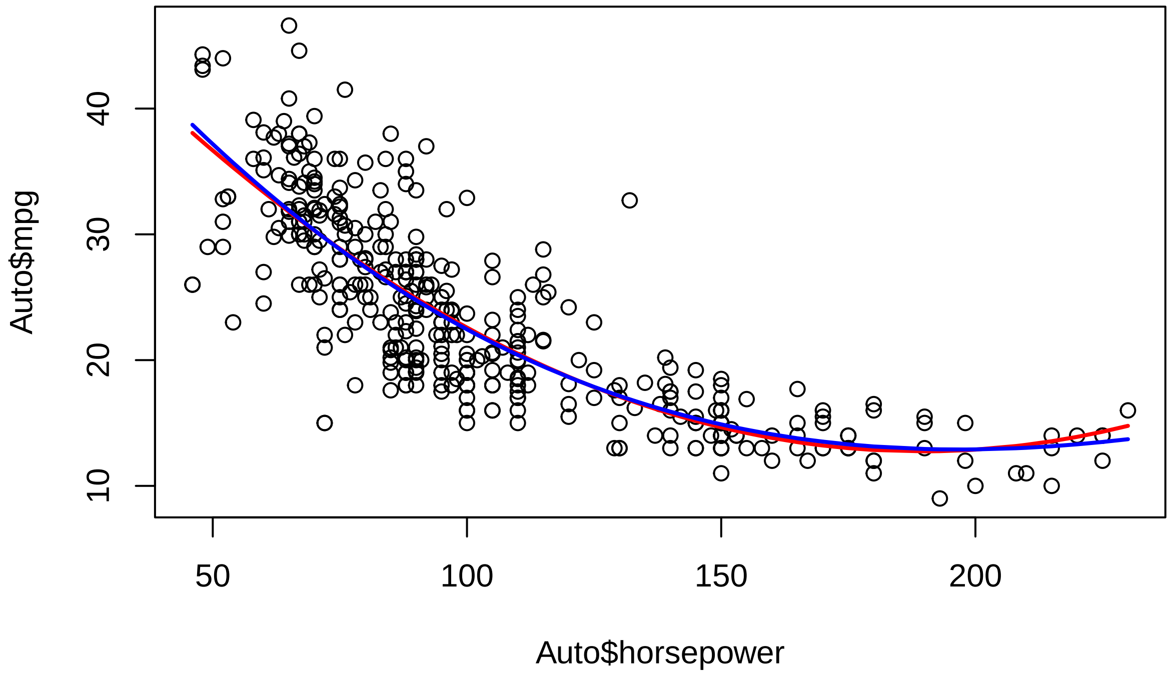
```r
cubic_fit <- lm(mpg ~ horsepower + I(horsepower^2) + I(horsepower^3), data = Auto)
stargazer(cubic_fit, type = "text")
```

```
##
## ===============================================
##                         Dependent variable:
##                     ---------------------------
##                                mpg
## -----------------------------------------------
## horsepower                  -0.569***
##                              (0.118)
##
## I(horsepower2)               0.002**
##                              (0.001)
##
## I(horsepower3)              -0.00000
##                             (0.00000)
##
## Constant                    60.685***
##                              (4.563)
##
## -----------------------------------------------
## Observations                   392
```

```
## R2                              0.688
## Adjusted R2                     0.686
## Residual Std. Error       4.375 (df = 388)
## F Statistic           285.481*** (df = 3; 388)
## ================================================
## Note:                 *p<0.1; **p<0.05; ***p<0.01
```

Plot all the lines

```r
plot(Auto$horsepower, Auto$mpg)
lines(sort(Auto$horsepower), fitted(quad_fit)[order(Auto$horsepower)], col = "red", lwd = 2)
lines(sort(Auto$horsepower), fitted(cubic_fit)[order(Auto$horsepower)], col = "blue", lwd = 2)
```



The improvement is not so much, we can also do it for higher order polynomia, in this way with `I()` in `lm()`. But there is another function in R called `poly()` that can do it for us.

Polynomial Regression with `poly()`

```r
poly_fit <- lm(mpg ~ poly(horsepower, 3, raw = TRUE ), data = Auto)

stargazer(poly_fit, type = "text")
```

```
##
## =============================================================
##                                      Dependent variable:
##                                  ---------------------------
##                                              mpg
## -------------------------------------------------------------
## poly(horsepower, 3, raw = TRUE)1          -0.569***
##                                            (0.118)
##
## poly(horsepower, 3, raw = TRUE)2           0.002**
##                                            (0.001)
##
```

```
## poly(horsepower, 3, raw = TRUE)3          -0.00000
##                                           (0.00000)
##
## Constant                                  60.685***
##                                            (4.563)
##
## -------------------------------------------------------------
## Observations                                 392
## R2                                          0.688
## Adjusted R2                                 0.686
## Residual Std. Error                   4.375 (df = 388)
## F Statistic                       285.481*** (df = 3; 388)
## =============================================================
## Note:                            *p<0.1; **p<0.05; ***p<0.01
```

§. Problem 11 Soln

Load the data

The first task is same, load the data.

```
Auto <- read_excel("/home/tanvir/Documents/ownCloud/Git_Repos/EWU_repos/3_Fall_2023/eco_204/ewu-eco204.g
```

```
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##   acceleration        year           origin          name
##  Min.   : 8.00   Min.   :70.00   Min.   :1.000   Length:392
##  1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000   Class :character
##  Median :15.50   Median :76.00   Median :1.000   Mode  :character
##  Mean   :15.54   Mean   :75.98   Mean   :1.577
##  3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
##  Max.   :24.80   Max.   :82.00   Max.   :3.000
```

Origin variable should be a factor, let's fix it

```
Auto$origin <- factor(Auto$origin, labels = c("USA", "Europe", "Japan"))
summary(Auto)
```

```
##       mpg          cylinders      displacement     horsepower        weight
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0   Min.   :1613
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0   1st Qu.:2225
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5   Median :2804
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5   Mean   :2978
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0   3rd Qu.:3615
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0   Max.   :5140
##   acceleration        year         origin        name
##  Min.   : 8.00   Min.   :70.00   USA   :245   Length:392
##  1st Qu.:13.78   1st Qu.:73.00   Europe: 68   Class :character
##  Median :15.50   Median :76.00   Japan : 79   Mode  :character
```

```
##  Mean   :15.54   Mean   :75.98
##  3rd Qu.:17.02   3rd Qu.:79.00
##  Max.   :24.80   Max.   :82.00
```

```r
# without interaction
model1 <- lm(mpg ~ horsepower + origin, data = Auto)
summary(model1)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower + origin, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.322  -3.292  -0.584   2.526  14.167
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.944129   0.867303  41.444  < 2e-16 ***
## horsepower   -0.133648   0.006863 -19.474  < 2e-16 ***
## originEurope  2.425339   0.677860   3.578  0.00039 ***
## originJapan   5.176352   0.647817   7.990 1.55e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.554 on 388 degrees of freedom
## Multiple R-squared:  0.6621, Adjusted R-squared:  0.6595
## F-statistic: 253.4 on 3 and 388 DF,  p-value: < 2.2e-16
```

```r
# with interaction
model2 <- lm(mpg ~ horsepower + origin + origin*horsepower, data = Auto)
summary(model2)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower + origin + origin * horsepower,
##     data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.7415  -2.9547  -0.6389   2.3978  14.2495
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             34.476496   0.890665  38.709  < 2e-16 ***
## horsepower              -0.121320   0.007095 -17.099  < 2e-16 ***
## originEurope            10.997230   2.396209   4.589 6.02e-06 ***
## originJapan             14.339718   2.464293   5.819 1.24e-08 ***
## horsepower:originEurope -0.100515   0.027723  -3.626 0.000327 ***
## horsepower:originJapan  -0.108723   0.028980  -3.752 0.000203 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.422 on 386 degrees of freedom
## Multiple R-squared:  0.6831, Adjusted R-squared:  0.679
## F-statistic: 166.4 on 5 and 386 DF,  p-value: < 2.2e-16
```