Bienvenido Hiyas Jr. 13/0902021
13824819

MA5832-Assessment 1

1 Linear algebra and probabilities

Question 1
A survey of holiday accommodation in New Zealand shows that in Auckland, excluding caravan parks and camping grounds, 40% were full (i.e., had no vacancies). A random sample of 25 holiday accommodation facilities (of the same capacity) in Auckland is selected at random. Assume the probability that a particular holiday accommodation facility is full is 0.40.

Ans:
The events in the question above are independent with each other therefore consider them as binomial distribution.

From the formula:

$$f(x) = \binom{n}{k} p^k (1-p)^{n-k}$$

Where:
k is the number of successes (no. of holiday accommodation as full),
n = 25 (total random sample)
p=0.4 (probability of success – when accommodation is full)

(a) What is the probability that at most 10 of these are full?

$$p(\mathrm{k} \leq 10) = \sum_{k=0}^{10} \binom{n}{k} p^k (1-p)^{n-k}$$

$$p(\mathrm{k} \leq 10) = \binom{25}{0}(0.4)^0(0.6)^{25} + \cdots + \binom{25}{10}(0.4)^{10}(0.6)^{15} = 0.59$$

(b) What is the probability that no more than 15 of these are full?

$$p(\mathrm{k} \leq 15) = \sum_{k=0}^{15} \binom{n}{k} p^k (1-p)^{n-k}$$

$$p(\mathrm{k} \leq 15) = \binom{25}{0}(0.4)^0(0.6)^{25} + \cdots + \binom{25}{15}(0.4)^{15}(0.6)^{10} = 0.99$$

(c) What is the probability that less than one quarter of them are full?

$$p(\mathrm{k} \leq 6.25) = \sum_{k=0}^{n6} \binom{n}{k} p^k (1-p)^{n-k}$$

$p(\text{k} \leq 6) = \binom{25}{0}(0.4)^0(0.6)^{25} + \cdots + \binom{25}{6}(0.4)^6(0.6)^{19} = 0.073$

Based on the answers above, we can see that as the number of samples increases, the probability of the holiday accommodations being full also increases.

Question 2

1. (a) Find the eigenvalues and eigenvectors of the 3 × 3 matrix

$$A = \begin{bmatrix} 5 & -3 & 3 \\ 4 & -2 & 3 \\ 4 & -4 & 5 \end{bmatrix}$$

Ans:

First, we need to find eigenvalues using the formula $|A - \lambda I| = 0$.

Where; $|\lambda I| = \begin{vmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{vmatrix}$

$$|A - \lambda I| = \begin{bmatrix} 5 & -3 & 3 \\ 4 & -2 & 3 \\ 4 & -4 & 5 \end{bmatrix} - \begin{vmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{vmatrix} = \begin{vmatrix} 5-\lambda & -30 & 3 \\ 4 & -2-\lambda & 30 \\ 4 & -4 & 5-\lambda \end{vmatrix}$$

Finding the determinant of the Matrix.

(5-λ) [(-2- λ)(5- λ)+12] +3[4(5- λ)-12] +3[-16-4(-2- λ)] =0

(5-λ)($\lambda^2 - 3\lambda\_2$)-$12\,\lambda + 24 + 12\lambda - 24$ =0

(5-λ)($\lambda^2 - 3\lambda\_2$)=0

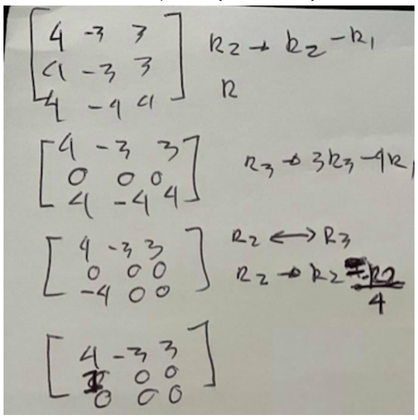
(5-λ)($\lambda - \mathbf{2}$)($\lambda - \mathbf{1}$)=0

Therefore, the eigenvalues are 1,2 and 5.

(a.1) Find the eigenvectors associated with $\lambda = 1$

$$A - \lambda I = \begin{bmatrix} 5 & -3 & 3 \\ 4 & -2 & 3 \\ 4 & -4 & 5 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & -3 & 3 \\ 4 & -3 & 3 \\ 4 & -4 & 4 \end{bmatrix}$$

Find RREF (Reduced Row-Echelon Form) of $(A - \lambda I)$:



$\begin{bmatrix} 4 & -3 & 3 \\ 4 & -3 & 3 \\ 4 & -4 & 4 \end{bmatrix}$ $R_2 \to R_2 - R_1$

R

$\begin{bmatrix} 4 & -3 & 3 \\ 0 & 0 & 0 \\ 4 & -4 & 4 \end{bmatrix}$ $R_3 \to 3R_3 - 4R_1$

$\begin{bmatrix} 4 & -3 & 3 \\ 0 & 0 & 0 \\ -4 & 0 & 0 \end{bmatrix}$ $R_2 \leftrightarrow R_3$

$R_2 \to R_2 \frac{-R_2}{4}$

$\begin{bmatrix} 4 & -3 & 3 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

RREF $(A-\lambda I)$ = $\begin{bmatrix} 4 & -3 & 3 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Find eigenvectors using RREF$(A-\lambda I)x$ = 0.

$$\begin{bmatrix} 4 & -3 & 3 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$4x-3y+3z=0$ Eq1

$x=0$ Eq2

Substitute Eq2 to Eq1;

$0-3y+3z=0$

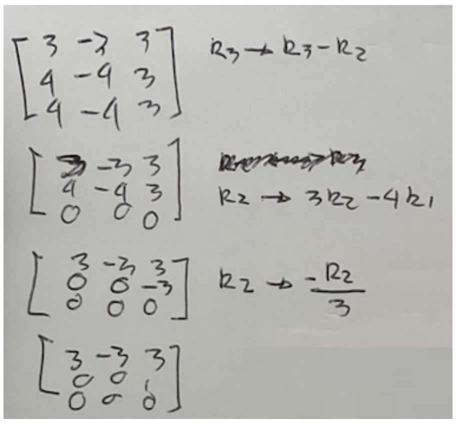
$z=y$

Let z = y = 1, therefore, the eigenvectors for eigenvalue 1 are $\begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}$

(a.2) Find the eigenvectors associated with $\lambda = 2$,

A- λI = $\begin{bmatrix} 5 & -3 & 3 \\ 4 & -2 & 3 \\ 4 & -4 & 5 \end{bmatrix}$ - $\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$ $=\begin{bmatrix} 3 & -3 & 3 \\ 4 & -4 & 3 \\ 4 & -4 & 3 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Find RREF of $(A-\lambda I)$ and solve eigenvectors using RREF$(A-\lambda I)x$ = 0.



$\begin{bmatrix} 3 & -3 & 3 \\ 4 & -4 & 3 \\ 4 & -4 & 3 \end{bmatrix}$ $R_3 \to R_3 - R_2$

$\begin{bmatrix} 3 & -3 & 3 \\ 4 & -4 & 3 \\ 0 & 0 & 0 \end{bmatrix}$ ~~[illegible]~~ $R_2 \to 3R_2 - 4R_1$

$\begin{bmatrix} 3 & -3 & 3 \\ 0 & 0 & -3 \\ 0 & 0 & 0 \end{bmatrix}$ $R_2 \to \frac{-R_2}{3}$

$\begin{bmatrix} 3 & -3 & 3 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

RREF $(A-\lambda I)$ = $\begin{bmatrix} 3 & -3 & 3 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

Find eigenvectors using RREF$(A-\lambda I)x$ = 0.

$$\begin{bmatrix} 3 & -3 & 3 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix}=\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$-3x-3y+3z=0$ Eq3

$z=0$ Eq4

Substitute Eq4 to Eq3;

$$-3x - 3y + 0 = 0$$
$$x = y$$

Let x = y =1. Therefore, the eigenvectors for eigenvalue 2 are $\begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix}$

(a.3) Find the eigenvectors associated with $\lambda = 5$;

A- λI = $\begin{bmatrix} 5 & -3 & 3 \\ 4 & -2 & 3 \\ 4 & -4 & 5 \end{bmatrix}$ - $\begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 0 & -3 & 3 \\ 4 & -7 & 3 \\ 4 & -4 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

Find RREF of $(A - \lambda I)$ and solve eigenvectors using RREF$(A - \lambda I)x$ = 0.

$$\begin{bmatrix} 0 & -3 & 3 \\ 4 & -7 & 3 \\ 4 & -4 & 0 \end{bmatrix} \quad R_3 \to R_3 - R_2$$

$$\begin{bmatrix} 0 & -3 & 3 \\ 4 & -7 & 3 \\ 0 & 3 & -3 \end{bmatrix} \quad R_3 \to R_3 + R_1$$

$$\begin{bmatrix} 0 & -3 & 3 \\ 4 & -7 & 3 \\ 0 & 0 & 0 \end{bmatrix} \quad R_1 \to \frac{R_1}{3}$$

$$\begin{bmatrix} 0 & -1 & 1 \\ 4 & -7 & 3 \\ 0 & 0 & 0 \end{bmatrix}$$

RREF $(A - \lambda I)$ = $\begin{bmatrix} 0 & -1 & 1 \\ 4 & -7 & 3 \\ 0 & 0 & 0 \end{bmatrix}$

Find eigenvectors using RREF$(A - \lambda I)x$ = 0.

$$\begin{bmatrix} 0 & -1 & 1 \\ 4 & -7 & 3 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$-y + z = 0;\ z = y \qquad \text{Eq5}$$
$$4x - 7y + 3z = 0 \qquad \text{Eq6}$$

Substitute Eq5 to Eq6 and assume z =y =1;

$$4x - 7(1) + 3(1) = 0$$
$$4x = 4$$
$$x = 1$$

Therefore, the eigenvectors for eigenvalue 5 are $\begin{bmatrix} \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}$

2. (b) Provide an application of eigenvalues and eigenvectors in Data Science or Statistics. Elaborate their usefulness in the application.

Ans: The decomposition of matrix into eigenvectors and eigenvalues can give important information about the properties of the matrix. In Machine Learning, the concept of eigenvalues and eigenvectors are used in in Principal Component Analysis (PCA). It is used as a dimensionality reduction technique that address the of curse of dimensionality which is the error that increases when the number of features increases. This problem is seen in computer vision that deals with images and even in Machine Learning features with high dimensionality increase model capacity which in turn requires a large amount of data to train. The eigenvalue and eigenvectors are used to reduce the dimensionality of the data by projecting it in fewer principal directions than its original dimensionality. The eigenvalue decomposition is used in the covariance matrix of Z since the covariance matrix is symmetrical and the eigenvectors as the principal axes or principal directions. Eigenvalues and eigenvectors are also used in Spectral Clustering to find K clusters that handles the issues related to K-Means such as dependence in cluster initialization and dimensionality features or when clusters are not spherical. Spectral Clustering using eigenvector and eigenvalues of the graph Laplacian matrix to find clusters (or partitions) of the graph. This method easily outperforms other algorithms for clustering. Eigenvectors and eigenvalues are also used in other fields. In communications systems, they are used to determine how much data can be transmitted through a communication medium. Furthermore, eigenvectors and eigenvalues are used to reduce the noise from the data. Finally, covariance matrix can also be used for computing correlation and risk tolerance.

Question 3

In this question, we consider the marketing dataset from datarium package in R.

```
install.packages("datarium")
library(datarium)
data(marketing)
```

The data contains 200 observations and 4 variables. The response variable is sales, denoted as Y. The explanatory variables—measured in thousands of dollars—are advertising budget spent on youtube, newspapers and facebook, respectively, which are denoted as X_1, X_2 and X_3, respectively. Provided equations below for calculating Beta (β)

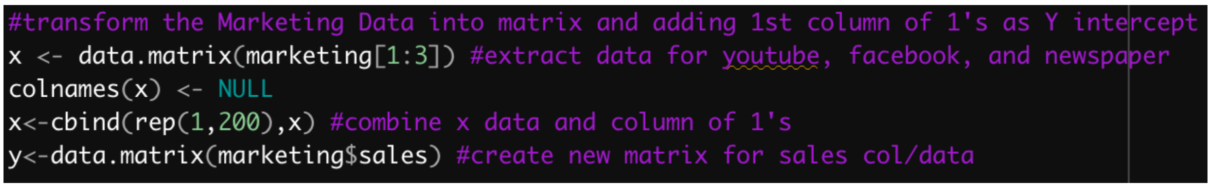
$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'Y \qquad (4)$$

And its standard error.

$$s.e(\hat{\boldsymbol{\beta}}) = \sqrt{s^2(\mathbf{X}'\mathbf{X})^{-1}} \text{ where } s^2 = \frac{1}{n-4}\sum_{i=1}^{n}(Y_i - \mathbf{X}_i\hat{\boldsymbol{\beta}})^2, i = 1, 2, \ldots, n. \qquad (5)$$

Ans:

First, the data (marketing) is split into the two; the response variable (sales) which represents our “y” and the explanatory variables (youtube, newspaper and facbook) which represents “x”. Then, we add a column of 1’s in the “y” variable that represents Beta or Y-Intercept. Finally, we transform both data into matrices.

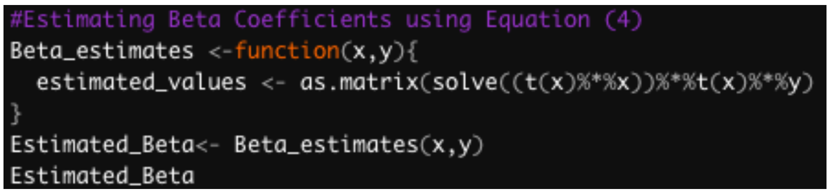


```
#transform the Marketing Data into matrix and adding 1st column of 1's as Y intercept
x <- data.matrix(marketing[1:3]) #extract data for youtube, facebook, and newspaper
colnames(x) <- NULL
x<-cbind(rep(1,200),x) #combine x data and column of 1's
y<-data.matrix(marketing$sales) #create new matrix for sales col/data
```

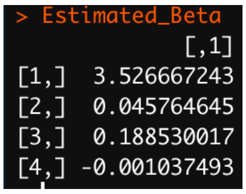
Task (a)

Use equations (4) and (5) to estimate β and its standard error in R.

In this task, we used the Equation 4 provided above to estimate the Beta coefficients.



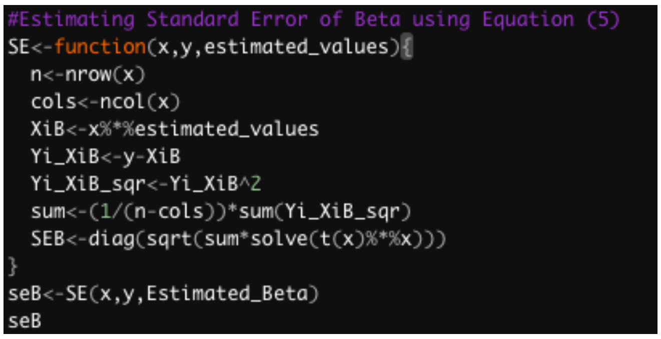
```
#Estimating Beta Coefficients using Equation (4)
Beta_estimates <-function(x,y){
  estimated_values <- as.matrix(solve((t(x)%*%x))%*%t(x)%*%y)
}
Estimated_Beta<- Beta_estimates(x,y)
Estimated_Beta
```



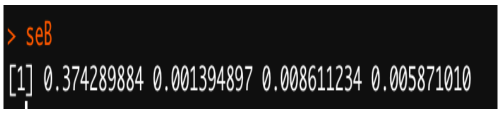
```
> Estimated_Beta
             [,1]
[1,]  3.526667243
[2,]  0.045764645
[3,]  0.188530017
[4,] -0.001037493
```

As shown above, the estimated coefficients are 3.526667 for the Beta (β) or Y-Intercept, 0.045765 for youtube, 0.188530 for facebook and -0.001037 for newspaper.

Then, the β Standard Error is computed using the Equation 5.



```
#Estimating Standard Error of Beta using Equation (5)
SE<-function(x,y,estimated_values){
  n<-nrow(x)
  cols<-ncol(x)
  XiB<-x%*%estimated_values
  Yi_XiB<-y-XiB
  Yi_XiB_sqr<-Yi_XiB^2
  sum<-(1/(n-cols))*sum(Yi_XiB_sqr)
  SEB<-diag(sqrt(sum*solve(t(x)%*%x)))
}
seB<-SE(x,y,Estimated_Beta)
seB
```



```
> seB
[1] 0.374289884 0.001394897 0.008611234 0.005871010
```

As shown above, the standard errors are 0.37429, 0.001395, 0.008611 and 0.005871. for the Y-Intercept, youtube, facebook, and newspaper respectively.

Task (b)

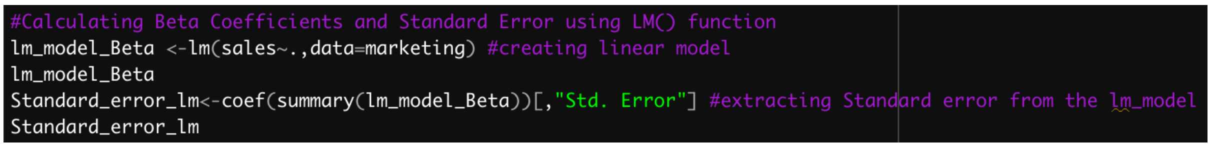
Compare the results obtained in Question 3(a) with those obtained from the function lm() in R.

Ans:

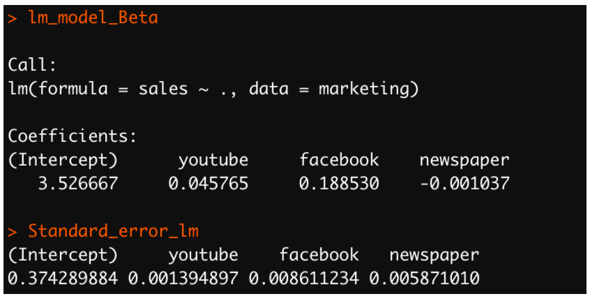
Using the lm() function in R. We have the results as shown in Table 1 below.

Table 1: Results from using Equations Provided and lm() function in R

	Using Equation Provided		Using lm() function in R	
	Estimated Coefficient	Standard Error	Estimated Beta	Standard Error
Y-Intercept	3.526667	0.37429	3.526667	0.374289884
youtube	0.045765	0.001395	0.045765	0.001394897
facebook	0.18853	0.008611	0.18853	0.008611234
newspaper	-0.001037	0.005871	-0.001037	0.00587101



```
#Calculating Beta Coefficients and Standard Error using LM() function
lm_model_Beta <-lm(sales~.,data=marketing) #creating linear model
lm_model_Beta
Standard_error_lm<-coef(summary(lm_model_Beta))[,"Std. Error"] #extracting Standard error from the lm_model
Standard_error_lm
```



```
> lm_model_Beta

Call:
lm(formula = sales ~ ., data = marketing)

Coefficients:
(Intercept)      youtube     facebook    newspaper
   3.526667     0.045765     0.188530    -0.001037

> Standard_error_lm
(Intercept)     youtube    facebook   newspaper
0.374289884 0.001394897 0.008611234 0.005871010
```

Table I: Results using Equation provided and **lm()** function in R

	Using Equation Provided		**Using lm() function in R**	
	Estimated Coefficient	**Standard Error**	**Estimated Beta**	**Standard Error**
Y-Intercept	3.526667	0.37429	3.526667	0.374289884
youtube	0.045765	0.001395	0.045765	0.001394897
facebook	0.18853	0.008611	0.18853	0.008611234
newspaper	-0.001037	0.005871	-0.001037	0.00587101

A shown in table I, the results are *exactly the same* using Equation provided and lm() function in R. Please note that the results are rounded

2 Optimisation

Question 4

Classical Gradient Descent.

(a) Write down a step-by-step procedure of Classical Gradient Descent to estimate β in Equation (3).

$$\mathcal{L} = \frac{1}{n}(Y - \mathbf{X}\boldsymbol{\beta})'(Y - \mathbf{X}\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \mathbf{X}_i\boldsymbol{\beta})^2 \tag{3}$$

Ans:

The Gradient Descent method attempts to estimate the coefficients of the marketing data by minimising the loss function or the mean square error. It starts by a random position of the weight and then calculates the loss function each time we move to the next steps and adjusting the estimated coefficients to a more optimum coefficient. Initial steps is usually bigger but as the steps going to the global minima, it becomes smaller and smaller until the gradients are as close to 0 as possible to optimise coefficients.

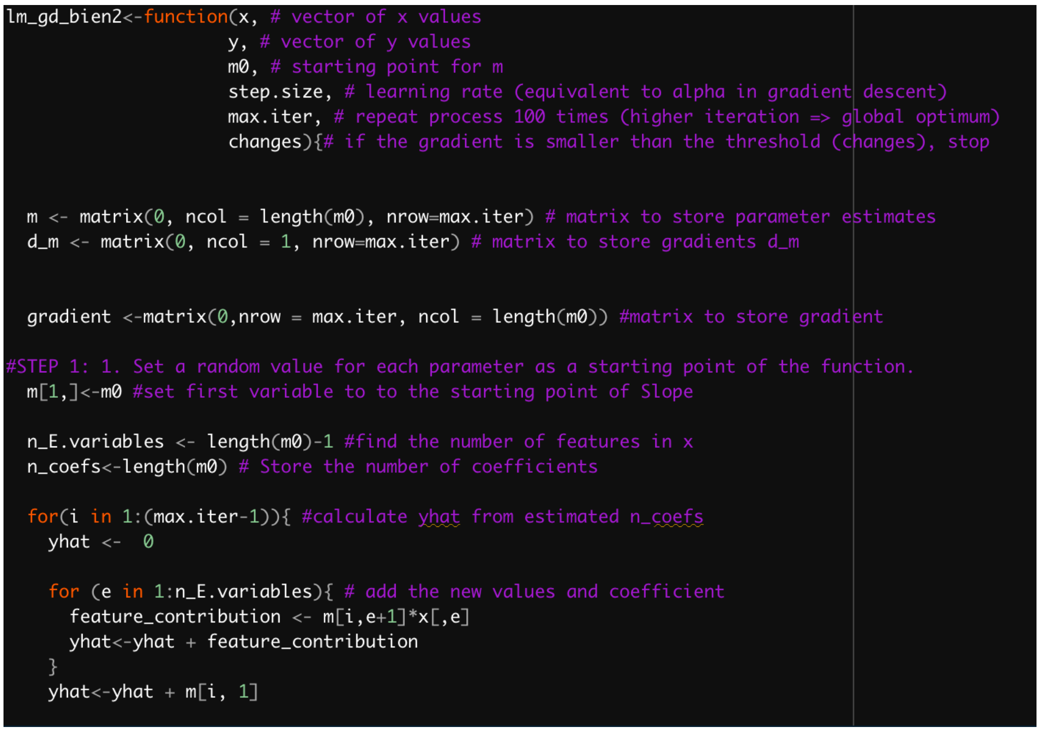
Gradient Descent Step-by-Step Procedure

1. Set a random value for each parameter as a starting point of the function.
2. Calculate the Loss Function and Store the value.
3. Calculate the Gradient by taking the derivative of Loss Function.
4. Update m according to M= m-l*d_m to determine the next step sizes.
5. Calculate new parameters by subtracting the step size from each of the previous parameters
6. Repeat steps 2-5 until the minimum step size is very small or the maximum number of Steps is achieved

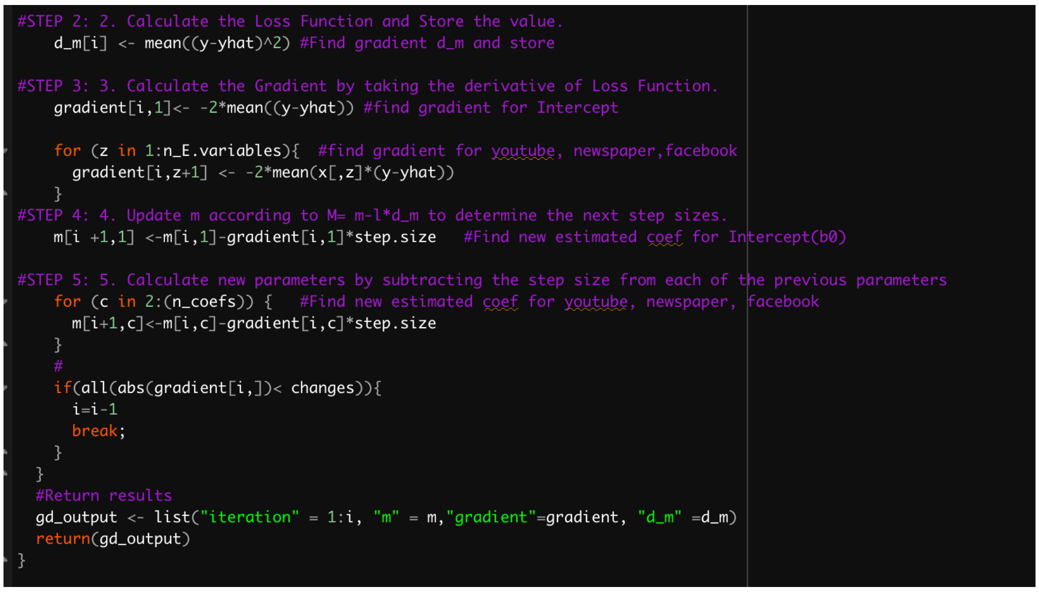
(b) Write an R code to implement the Classical Gradient Descent procedure provided in Question 4(a).

Ans.

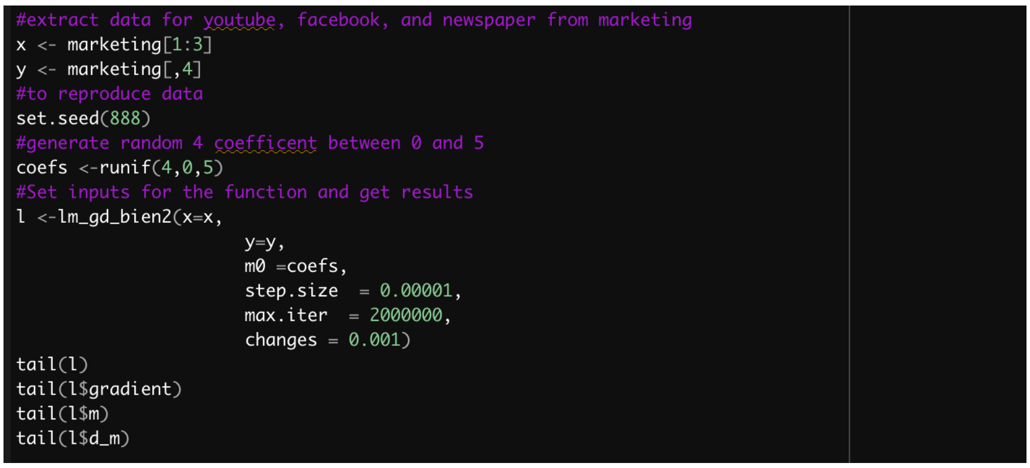
The Gradient Descent algorithm below takes dependent to the Y-Intercept values in a data frame and the second matrix with the values of the variables in the marketing dataset (YouTube, Facebook, newspaper). The parameters are *m0* or the starting point of m, the *step.size* or the learning rate, the *max.iter* or the iterations, and the *change*s or the minimum step size. The algorithm will calculate the results that includes the coefficients of gradient=m, loss functions d_m and maximum iterations.



```r
lm_gd_bien2<-function(x, # vector of x values
                     y, # vector of y values
                     m0, # starting point for m
                     step.size, # learning rate (equivalent to alpha in gradient descent)
                     max.iter, # repeat process 100 times (higher iteration => global optimum)
                     changes){# if the gradient is smaller than the threshold (changes), stop


  m <- matrix(0, ncol = length(m0), nrow=max.iter) # matrix to store parameter estimates
  d_m <- matrix(0, ncol = 1, nrow=max.iter) # matrix to store gradients d_m


  gradient <-matrix(0,nrow = max.iter, ncol = length(m0)) #matrix to store gradient

#STEP 1: 1. Set a random value for each parameter as a starting point of the function.
  m[1,]<-m0 #set first variable to to the starting point of Slope

  n_E.variables <- length(m0)-1 #find the number of features in x
  n_coefs<-length(m0) # Store the number of coefficients

  for(i in 1:(max.iter-1)){ #calculate yhat from estimated n_coefs
    yhat <-  0

    for (e in 1:n_E.variables){ # add the new values and coefficient
      feature_contribution <- m[i,e+1]*x[,e]
      yhat<-yhat + feature_contribution
    }
    yhat<-yhat + m[i, 1]
```



```r
 #STEP 2: 2. Calculate the Loss Function and Store the value.
    d_m[i] <- mean((y-yhat)^2) #Find gradient d_m and store

 #STEP 3: 3. Calculate the Gradient by taking the derivative of Loss Function.
    gradient[i,1]<- -2*mean((y-yhat)) #find gradient for Intercept

    for (z in 1:n_E.variables){  #find gradient for youtube, newspaper,facebook
      gradient[i,z+1] <- -2*mean(x[,z]*(y-yhat))
    }
 #STEP 4: 4. Update m according to M= m-l*d_m to determine the next step sizes.
    m[i +1,1] <-m[i,1]-gradient[i,1]*step.size   #Find new estimated coef for Intercept(b0)

 #STEP 5: 5. Calculate new parameters by subtracting the step size from each of the previous parameters
    for (c in 2:(n_coefs)) {   #Find new estimated coef for youtube, newspaper, facebook
      m[i+1,c]<-m[i,c]-gradient[i,c]*step.size
    }
    #
    if(all(abs(gradient[i,])< changes)){
      i=i-1
      break;
    }
  }
  #Return results
  gd_output <- list("iteration" = 1:i, "m" = m,"gradient"=gradient, "d_m" =d_m)
  return(gd_output)
}
```



```r
#extract data for youtube, facebook, and newspaper from marketing
x <- marketing[1:3]
y <- marketing[,4]
#to reproduce data
set.seed(888)
#generate random 4 coefficent between 0 and 5
coefs <-runif(4,0,5)
#Set inputs for the function and get results
l <-lm_gd_bien2(x=x,
                     y=y,
                     m0 =coefs,
                     step.size  = 0.00001,
                     max.iter  = 2000000,
                     changes = 0.001)
tail(l)
tail(l$gradient)
tail(l$m)
tail(l$d_m)
```

(c) Discuss the results obtained from Question 4(b) and compare it with that obtained from Question 3(a).

Ans:

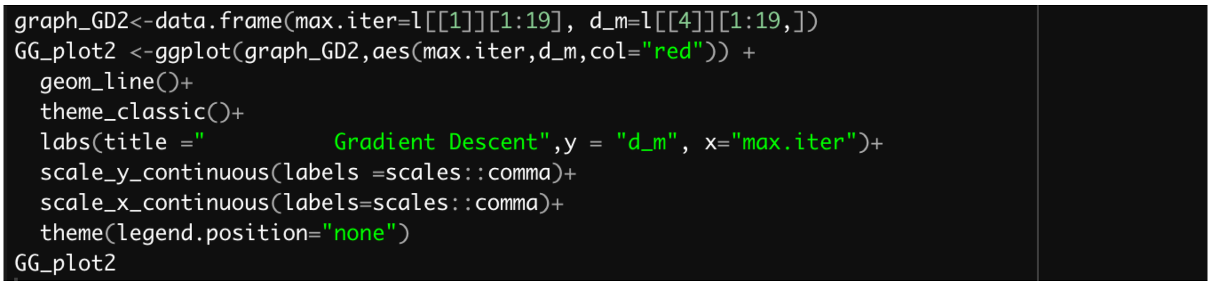
The final step size and max iterations have been achieved through trial and error. It has been determined to be the step.size = 0.00001, max.iter = 2300000 and changes = 0.001. The gradients are -0.001210496 2.754924e-06 1.157021e-05 6.127463e-06 for the intercept, YouTube, Facebook and newspaper respectively which are as close to 0 when the algorithm reached the final max iteration value. Increasing the max.iter value will result in 0 0 0 0 values. It was observed that changing the values in changes or the threshold doesn't affect the result. However, changing the values of step size will affect the result but has a little effect. The best way is to change the values for max.iter, however, the higher the value of max.iter will result in a longer waiting time for the results of the algorithm.

The d_m or mean squared error was calculated to be 4.009144 when it reach the max.iter of 2300000. Which is more or less than the lm() function in R. The also results shows that the m coefficients are [2299999,] 3.522521 0.04577408 0.1885696 -0.001016507 for the Intercept, youtube, Facebook and newspaper respectively which are as close as the results from lm() function in R as shown in Table 2. The slight difference between the results could be because that the gradient descent algorithm is just an estimation which could result in ineffective optimization of the coefficients. Another potential cause is that you need to find the correct step size and maximum iteration which was done manually. Finally, gradient descent tends to find for local minima instead of global minima of the loss function which depends on the random number of coefficients from the start of the algorithm.

Tabel 2. Lm() function in R and Gradient Descent Algorithm results.

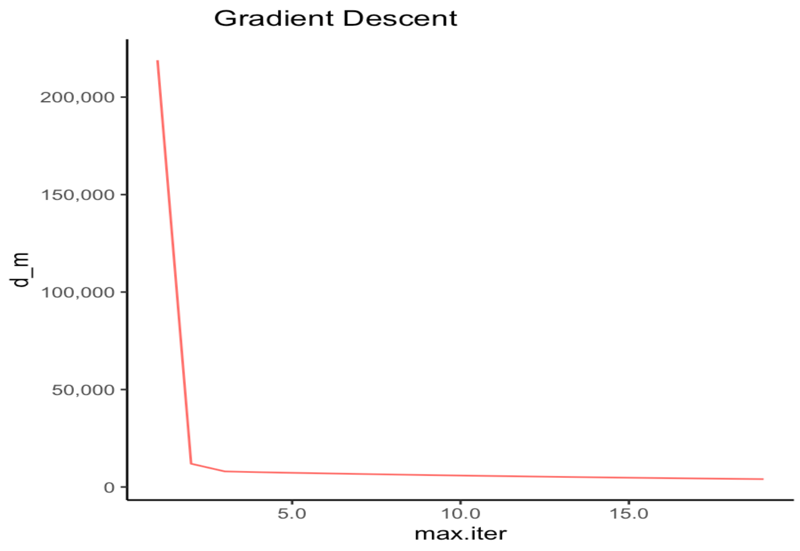
	m (slope) coefficents	
	lm() in R	Gradient Descent Algorithm
Y-Intercept	3.526667	3.522521
youtube	0.045765	n0.04577408
facebook	0.18853	0.1885696
newspaper	-0.001037	0.001016507

The figure below shows how gradient descent algorithm perform. It can be seen that the d_m was quickly minimizing the mean squared error prior to closing in the optimal solution.



```
graph_GD2<-data.frame(max.iter=l[[1]][1:19], d_m=l[[4]][1:19,])
GG_plot2 <-ggplot(graph_GD2,aes(max.iter,d_m,col="red")) +
  geom_line()+
  theme_classic()+
  labs(title ="          Gradient Descent",y = "d_m", x="max.iter")+
  scale_y_continuous(labels =scales::comma)+
  scale_x_continuous(labels=scales::comma)+
  theme(legend.position="none")
GG_plot2
```

Figure 1.



Question 5

Compare the optimisation algorithms of Classical Gradient Descent, Stochastic Gradient Descent and Newton's methods to see advantages and disadvantages of each algorithm.

Ans:

Classical Gradient Descent

The formula for the Gradient Descent is below:

$$x^{(k+1)} = x^{(k)} - \alpha_k \nabla f(x^{(k)})$$

It starts with an arbitrary starting point x(0) and the next point x(1) is moving in the direction −α1∇f(x) from the starting point x(0). The parameter α1 is the step size. The complicated part is the choice of the step size. The gradient descent converges if the step size is small enough else, it the algorithm zigzags through the space when the step size is large.

In the standard gradient descent algorithm, the parameters are updated on the entire data set iteratively. On a large data set, the standard gradient descent algorithm is not very efficient to find the global minimum of parameters (weights). A smaller learning rate is recommended so that we don't overshoot the global minimum while calculating the parameter.
It is evident that on every step, the entire training set is used and that is the reason it is also called batch gradient descent. Such calculation is expensive on a huge dataset as it has to store the intermediate values in memory or read from disk on each iteration.

Stochastic Gradient Descent

Stochastic Gradient Descent is the same as the classical gradient descent however, its main difference is that it considers a single training observation at a single iteration or epoch. Hence, it is computationally very fast than the latter.

In stochastic gradient descent, the algorithm uses a single or a few training examples to calculate the parameters. The algorithm starts by randomly shuffling the training set. Then iteratively calculate the parameters. The learning rate is much smaller than the classical gradient descent due to higher variance. Additionally, it is also memory efficient because it considers one observation at a time from the complete dataset. The only disadvantage of this is that it does not converge straight because of the noise since it only uses a sample of the data et.

Newton's Method

The main difference between Newton's Method from Gradient and Stochastic Descent is that it uses a second derivative instead of the first derivative. However, computing the second derivative is often complicated or tractable, requiring a lot of computations. Another difference between Newton's method and the Gradient Descent is that it uses the inverse of the Hessian matrix instead of the step size or learning rate and then uses the product of the inverse of the Hessian and the gradient to optimize the system.

Newton step

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boxed{\nabla^2 f(\boldsymbol{x}_k)^{-1}}\,\nabla f(\boldsymbol{x}_k)$$

Gradient Descent

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boxed{\alpha}\,\nabla f(\boldsymbol{x}_k)$$

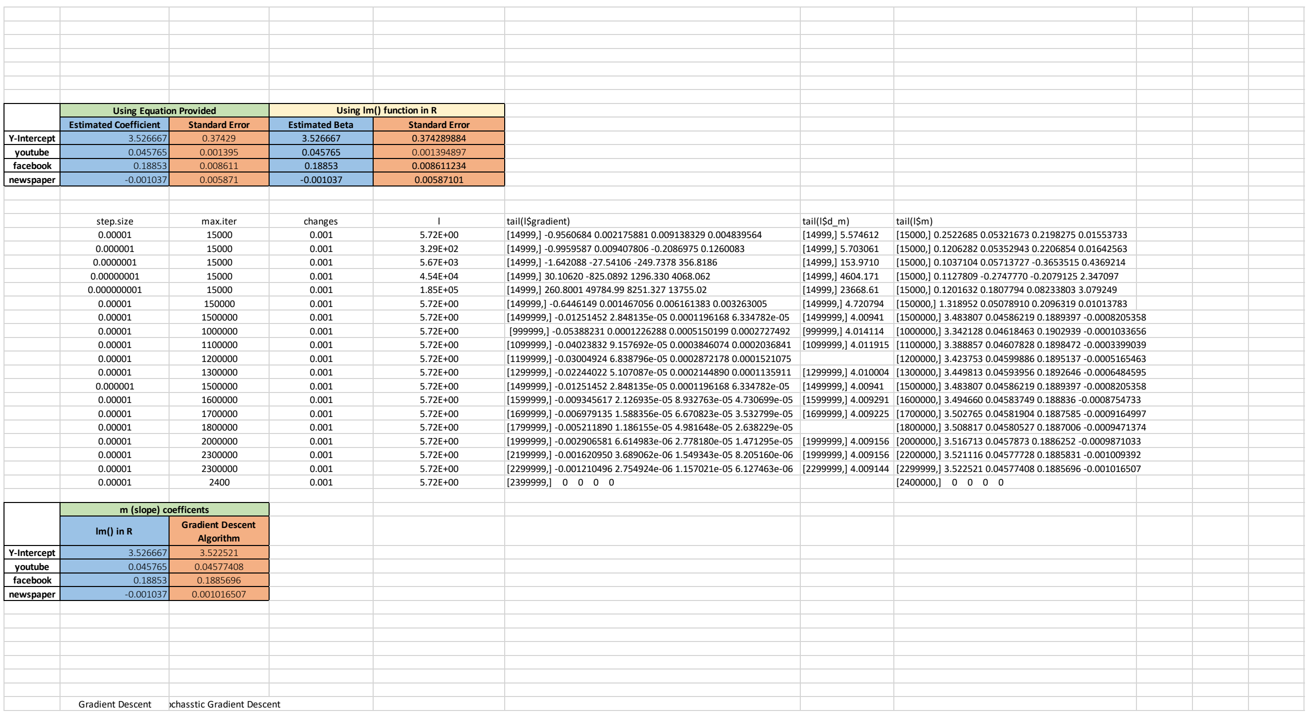
The Hessian matrix allows us to capture the curvature of the problem and help to converge very fast to the optimum point. But the one drawback is that direction in which to move is not known. Further computation is needed to find it. Furthermore, it stores the second derivatives in a Hessian Matrix after each iteration, but the downside is that the Hessian computation is expensive and demanding. Moreover, the Hessian might not be invertible in nonconvex problems. However, Newton's method is great for high precision and is used in practice.

Appendix:

```
install.packages("datarium")
library(ggplot2)
library(datarium)
data(marketing)
#transform the Marketing Data into matrix and adding 1st column of 1's as Y intercept
x <- data.matrix(marketing[1:3]) #extract data for youtube, facebook, and newspaper
colnames(x) <- NULL
x<-cbind(rep(1,200),x) #combine x data and column of 1's
y<-data.matrix(marketing$sales) #create new matrix for sales col/data

#Estimating Beta Coefficients using Equation (4)
```

```
Beta_estimates <-function(x,y){
  estimated_values <- as.matrix(solve((t(x)%*%x))%*%t(x)%*%y)
}
Estimated_Beta<- Beta_estimates(x,y)
Estimated_Beta

#Estimating Standard Error of Beta using Equation (5)
SE<-function(x,y,estimated_values){
  n<-nrow(x)
  cols<-ncol(x)
  XiB<-x%*%estimated_values
  Yi_XiB<-y-XiB
  Yi_XiB_sqr<-Yi_XiB^2
  sum<-(1/(n-cols))*sum(Yi_XiB_sqr)
  SEB<-diag(sqrt(sum*solve(t(x)%*%x)))
}
seB<-SE(x,y,Estimated_Beta)
seB

#Calculating Beta Coefficients and Standard Error using LM() function in R
lm_model_Beta <-lm(sales~.,data=marketing) #creating linear model
lm_model_Beta #display Beta Coefficients
Standard_error_lm<-coef(summary(lm_model_Beta))[,"Std.   Error"]   #extracting   Standard
error from the lm_model
Standard_error_lm #display Standard error

###########
#Question 4b: Gradient Descent
lm_gd_bien2<-function(x, # vector of x values
                y, # vector of y values
                m0, # starting point for m
                step.size, # learning rate (equivalent to alpha in gradient descent)
                max.iter, # repeat process 100 times (higher iteration => global optimum)
                changes){# if the gradient is smaller than the threshold (changes), stop


  m <- matrix(0, ncol = length(m0), nrow=max.iter) # matrix to store parameter estimates
  d_m <- matrix(0, ncol = 1, nrow=max.iter) # matrix to store gradients d_m


  gradient <-matrix(0,nrow = max.iter, ncol = length(m0)) #matrix to store gradient

#STEP 1: 1.      Set a random value for each parameter as a starting point of the function.
  m[1,]<-m0 #set first variable to to the starting point of Slope

  n_parameters <- length(m0)-1 #find the number of features in x
  n_coefs<-length(m0) # Store the number of coefficients
  for(i in 1:(max.iter-1)){ #calculate yhat from estimated n_coefs
    yhat <-  0

    for (e in 1:n_parameters){ # add the new values and coefficient
      feature_contribution <- m[i,e+1]*x[,e]
      yhat<-yhat + feature_contribution
    }
    yhat<-yhat + m[i, 1]

#STEP 2: 2.      Calculate the Loss Function and Store the value.
    d_m[i] <- mean((y-yhat)^2) #Find gradient d_m and store

#STEP 3: 3.      Calculate the Gradient by taking the derivative of Loss Function.
    gradient[i,1]<- -2*mean((y-yhat)) #find gradient for Intercept

    for (z in 1:n_parameters){  #find gradient for youtube, newspaper,facebook
      gradient[i,z+1] <- -2*mean(x[,z]*(y-yhat))
    }
#STEP 4: 4.      Update m according to M= m-l*d_m to determine the next step sizes.
    m[i +1,1] <-m[i,1]-gradient[i,1]*step.size   #Find new estimated coef for Intercept(b0)

#STEP 5: 5.      Calculate new parameters by subtracting the step size from each of the
previous parameters
    for (c in 2:(n_coefs)) {   #Find new estimated coef for youtube, newspaper, facebook
      m[i+1,c]<-m[i,c]-gradient[i,c]*step.size
    }
    #
    if(all(abs(gradient[i,])< changes)){  #i>1 & will result to Nan
      i=i-1
      break;
    }
  }
  #Return results
  gd_output <- list("iteration" = 1:i, "m" = m,"gradient"=gradient, "d_m" =d_m)
  return(gd_output)
}

#extract data for youtube, facebook, and newspaper from marketing
x <- marketing[1:3]
y <- marketing[,4]
#to reproduce data
set.seed(888)
#generate random 4 coefficent between 0 and 5
coefs <-runif(4,0,5)
#Set inputs for the function and get results
l <-lm_gd_bien2(x=x,
                    y=y,
                    m0 =coefs,
                    step.size  = 0.00001,
                    max.iter  = 2300000 ,
                    changes = 0.001)
tail(l)
tail(l$gradient)
tail(l$m)
tail(l$d_m)

# Plot result (l) for interpretation
graph_GD2<-data.frame(max.iter=l[[1]][1:100], d_m=l[[4]][1:100,])
GG_plot2 <-ggplot(graph_GD2,aes(max.iter,d_m,col="red")) +
  geom_line()+
  theme_classic()+
  labs(title ="          Gradient Descent",y = "d_m", x="max.iter")+
  scale_y_continuous(labels =scales::comma)+
  scale_x_continuous(labels=scales::comma)+
  theme(legend.position="none")
GG_plot2
```

Results in Excel File.



	Using Equation Provided		Using lm() function in R	
	Estimated Coefficient	Standard Error	Estimated Beta	Standard Error
Y-Intercept	3.526667	0.37429	3.526667	0.374289884
youtube	0.045765	0.001395	0.045765	0.001394897
facebook	0.18853	0.008611	0.18853	0.008611234
newspaper	-0.001037	0.005871	-0.001037	0.00587101

step.size	max.iter	changes	l	tail(l$gradient)	tail(l$d_m)	tail(l$m)
0.00001	15000	0.001	5.72E+00	[14999,] -0.9560684 0.002175881 0.009138329 0.004839564	[14999,] 5.574612	[15000,] 0.2522685 0.05321673 0.2198275 0.01553733
0.000001	15000	0.001	3.29E+02	[14999,] -0.9959587 0.009407806 -0.2086975 0.1260083	[14999,] 5.703061	[15000,] 0.1206282 0.05352943 0.2206854 0.01642563
0.0000001	15000	0.001	5.67E+03	[14999,] -1.642088 -27.54106 -249.7378 356.8186	[14999,] 153.9710	[15000,] 0.1037104 0.05713727 -0.3653515 0.4369214
0.00000001	15000	0.001	4.54E+04	[14999,] 30.10620 -825.0892 1296.330 4068.062	[14999,] 4604.171	[15000,] 0.1127809 -0.2747770 -0.2079125 2.347097
0.000000001	15000	0.001	1.85E+05	[14999,] 260.8001 49784.99 8251.327 13755.02	[14999,] 23668.61	[15000,] 0.1201632 0.1807794 0.08233803 3.079249
0.00001	150000	0.001	5.72E+00	[149999,] -0.6446149 0.001467056 0.006161383 0.003263005	[149999,] 4.720794	[150000,] 1.318952 0.05078910 0.2096319 0.01013783
0.00001	1500000	0.001	5.72E+00	[1499999,] -0.01251452 2.848135e-05 0.0001196168 6.334782e-05	[1499999,] 4.00941	[1500000,] 3.483807 0.04586219 0.1889397 -0.0008205358
0.00001	1000000	0.001	5.72E+00	[999999,] -0.05388231 0.0001226288 0.0005150199 0.0002727492	[999999,] 4.014114	[1000000,] 3.342128 0.04618463 0.1902939 -0.0001033656
0.00001	1100000	0.001	5.72E+00	[1099999,] -0.04023832 9.157692e-05 0.0003846074 0.0002036841	[1099999,] 4.011915	[1100000,] 3.388857 0.04607828 0.1898472 -0.0003399039
0.00001	1200000	0.001	5.72E+00	[1199999,] -0.03004924 6.838796e-05 0.0002872178 0.0001521075		[1200000,] 3.423753 0.04599886 0.1895137 -0.0005165463
0.00001	1300000	0.001	5.72E+00	[1299999,] -0.02244022 5.107087e-05 0.0002144890 0.0001135911	[1299999,] 4.010004	[1300000,] 3.449813 0.04593956 0.1892646 -0.0006484595
0.000001	1500000	0.001	5.72E+00	[1499999,] -0.01251452 2.848135e-05 0.0001196168 6.334782e-05	[1499999,] 4.00941	[1500000,] 3.483807 0.04586219 0.1889397 -0.0008205358
0.00001	1600000	0.001	5.72E+00	[1599999,] -0.009345617 2.126935e-05 8.932763e-05 4.730699e-05	[1599999,] 4.009291	[1600000,] 3.494660 0.04583749 0.188836 -0.0008754733
0.00001	1700000	0.001	5.72E+00	[1699999,] -0.006979135 1.588356e-05 6.670823e-05 3.532799e-05	[1699999,] 4.009225	[1700000,] 3.502765 0.04581904 0.1887585 -0.0009164997
0.00001	1800000	0.001	5.72E+00	[1799999,] -0.005211890 1.186155e-05 4.981648e-05 2.638229e-05		[1800000,] 3.508817 0.04580527 0.1887006 -0.0009471374
0.00001	2000000	0.001	5.72E+00	[1999999,] -0.002906581 6.614983e-06 2.778180e-05 1.471295e-05	[1999999,] 4.009156	[2000000,] 3.516713 0.0457873 0.1886252 -0.0009871033
0.00001	2300000	0.001	5.72E+00	[2199999,] -0.001620950 3.689062e-06 1.549343e-05 8.205160e-06	[1999999,] 4.009156	[2200000,] 3.521116 0.04577728 0.1885831 -0.001009392
0.00001	2300000	0.001	5.72E+00	[2299999,] -0.001210496 2.754924e-06 1.157021e-05 6.127463e-06	[2299999,] 4.009144	[2299999,] 3.522521 0.04577408 0.1885696 -0.001016507
0.00001	2400	0.001	5.72E+00	[2399999,] 0 0 0 0		[2400000,] 0 0 0 0

	m (slope) coefficents	
	lm() in R	Gradient Descent Algorithm
Y-Intercept	3.526667	3.522521
youtube	0.045765	0.04577408
facebook	0.18853	0.1885696
newspaper	-0.001037	0.001016507

Gradient Descent	ochasstic Gradient Descent