Tutorial 3 Solution

1. Read the data from the file `Colleges.txt`. Consider a simple linear regression of percentage of applicants accepted (`Acceptance`) on the median combined math and verbal SAT score of students (`SAT`), called Model M1.

   (a) Write your own function in R to derive the equation of Model M1.

   (b) Use function `lm()` in R to derive the equation of Model M1. Compare with your answer in part (a).

   Solution:

```
> ##### (a)
> simple <- function(x , y) {
+ beta_1 <- (sum(x*y)- mean (y)* sum (x ))/( sum(x^2)- mean(x)* sum(x));
+ beta_0 <- mean(y)- beta_1* mean(x) ;
+ return(c( beta_0 , beta_1)) ;
+ }
> ### Import data set into R:
> dat= read.table("C:/Data/Colleges.txt",header =TRUE,sep= "\t")
> names(dat)

[1] "School"      "School_Type" "SAT"         "Acceptance"  "DPerStudent"
[6] "Top.10p"     "PerPhD"      "GradPer"

> head(dat)

      School School_Type  SAT Acceptance DPerStudent Top.10p PerPhD GradPer
1    Amherst    Lib Arts 1315         22       26636      85     81      93
2 Swarthmore    Lib Arts 1310         24       27487      78     93      88
3   Williams    Lib Arts 1336         28       23772      86     90      93
4    Bowdoin    Lib Arts 1300         24       25703      78     95      90
5  Wellesley    Lib Arts 1250         49       27879      76     91      86
6     Pomona    Lib Arts 1320         33       26668      79     98      80

> # Run the function to obtain the coefficients:
> simple(dat$SAT, dat$Acceptance )

[1] 202.2677440  -0.1300894

> ##### (b) - Compare outputs
> lm(Acceptance ~ SAT , data =dat )

Call:
lm(formula = Acceptance ~ SAT, data = dat)

Coefficients:
(Intercept)          SAT
   202.2677      -0.1301
```

   CONCLUDE: the results are the same.

2. Consider the question given in Tutorial 1.

(a) For the first question in Tutorial 1, use the code to define a function, called F1, where the argument of F1 is **salary**. Run function F1 for the two cases mentioned.

(b) For the second question in Tutorial 1, use the code to define a function, called F2, where F2 has two arguments: **salary** and **rate**. Run function F2 for the two cases mentioned to obtain the results.

(c) From question the settings given in Tutorial 1, we know that both the percentage of your salary that you save each month and the rate of raising salary every 4 months affects how long it takes you to save for a down payment.

Now, suppose the raise in salary every 4 months is fixed at 0.01 and you want to set a particular goal, e.g. to be able to afford the down payment in five years for a house with the price is of your choice, **price**. How much should you save each month instead of 40% to achieve the goal? In this problem, you are going to write a function, called F3, which helps to answer that question.

You are now going to find the **best propotion of savings monthly** from your salary to achieve a down payment in five years. Since hitting this exactly is a challenge, we simply want your total savings to be at least as the same as the required down payment. The proportion of saving should be of 2 decimal places.

Run function F3 and report the answers obtained for two cases: (**salary** = \$7,000 and **price** = \$1,200,000) and (**salary** = \$4,000, **price** = \$800,000).

Solution:

(a) Define function F1:

```
> price = 1200000 # House's price
> cost = price*0.25 # down payment amount
> r= 0.02 #monthly rate return from investment
> portion_save = 0.4 # portion of salary for saving, every month
> ### salary = monthly salary is the argument of F1
> F1 = function(salary){
+ saved <- 10000
+ month <- 0
+ while(saved < cost){
+     month = month +1
+     saved = saved+ portion_save *salary + saved*r
+   }
+   return(month)
+ }
> ###  Test F1 for the two cases:
> F1(7000) # answer should be 55 months
[1] 55
> F1(10000) # answer should be 44 months
[1] 44
```

(b) Define function F2:

```
> F2 <- function(salary, price = 1200000, rate = 0.01, portion_save = 0.4) {
+ r = 0.02 #monthly rate return from investment
+ saved <- 10000 # savings given by parents initially
+ month <- 0
+ cost = 0.25*price
+ while(saved < cost){
+     month = month +1
```

```
+     saved = saved+ portion_save *salary + saved*r
+     if (month%%4 ==0){salary = salary*(1+rate)}
+   }
+   return(month)
+ }
> ### Test function F2
> F2(salary = 7000, rate = 0.02) # answer: 52
[1] 52
> F2(salary = 10000, rate = 0.01) # answer: 43
[1] 43
```

Note: It requires F2 to have only two arguments, however we can add more arguments for F2 and set the default values for the arguments. This will help to set function F3 below be easier.

(c) Define function F3 where function F2 defined above is used within F3.

```
> F3 = function(price,salary){
+     rate = 0.01 # raise in salary per 4 months
+     portion = seq(0.01,1, by = 0.01) # possible percentage of salary for saving monthly
+     i = 1
+     month = 80
+     while((month > 60)&(i <=100)){
+       portion_save = portion[i]
+       month = F2(price = price, salary = salary, rate = rate, portion_save)
+       if (month>60){i = i+1}
+     }
+
+ return(portion_save)
+ }
> ### Test function F3:
> F3(price = 1200000, salary = 7000) # answer: 0.32
[1] 0.32
> F3(price = 800000, salary = 4000) # answer: 0.35
[1] 0.35
```