Stat243 PS8

Meng Wang, SID: 21706745

December 5, 2015

1 Problem 1

According to what we has described in class, there are typically six basic steps of a simulation study. We go over the six steps one by one for this particular experiment.

Step 1: At the first beginning, we need to specify what makes up an individual experiment. The basic components for this particular experiment includes at least the following essential components.

(1). We need a large sets of data to run simulations on. Let us assume those data are denoted as $(x_1, y_1), ..., (x_n, y_n)$, where the number of data points are represented as n, which is a fairly large number. The requirement for the data is, they are generated based on or close to the linear model, with a fair amount of data are outlying values, which mean they are far away from the standard fitted line. Based on the requirement for the data, here is the method we could use to generate the data. We firstly generated the data which satisfies the linear models, i.e. $y_i = \beta_1 * x_i + \beta_0 + \epsilon_i, /i = 1, ..., n$, where ϵ_i can be generated from the Gaussian distribution generator with the same variance σ. Those are the data for the standard linear regression. Then we can generator some other data with big variance, i.e. the outlying data. Those data can be generated also from the Gaussian distribution, but with big variance, 3σ for example. Mixing those two kinds of data together, we have the data set to test our models.

(2). We need the two models (methodologies) to predict how well the data suites the model. There are two methods in this problem, the standard linear regression and the new method dealing with the outlying data. The LES model can be directly called from the function in R.

(3). We need some standards or criteria to evaluate the performance of two methods. The two parameters of our interests are the absolute prediction error and coverage of prediction intervals. The absolute error can be calculated by $err = \sum_{i=1}^{n} |y_{i,new} - y_{i,pred}|$, where $y_{i,new}$ are new data for testing and $y_{i,pred}$ are the results calculated from our prediction model. Comparing err for two methods can provide us the evaluation of the model. For the coverage intervals, we can pick up a specific bands and calculate the ratio of the points fall into these bands for different model.

(4). There are some other parameters we need to specify in this experiment, for example, the number of the Boostrap sampling m.

Step 2: After we specified all the parameters we need to use in this experiment, we should pick up those numbers and generate the data during this step. For example, set the sample size $n = 10000$, the linear model $y = 2 \times x + 1$, the variance $\sigma = 0.02$, the number of experiments $m = 1000$.

Step 3: After we quantify our experiment in Step 2. We should write the code for this step to numerically implement our experiment design from Step 1 with the numerical value in Step 2. The code should give us the two results for one testing cases, the absolute error for two methods for this set of data err_{new}, err_{LES}, as well as the the coverage ratio α_{new}, α_{LES}.

Step 4: When the code is ready, we just need to run the simulations lots of times(m times). This efficiency of this procedure can be greatly improved by using the parallel computing technique. After this step, we shall have the results as $err_{new,i}$, $err_{LES,i}$ and $\alpha_{new,i}$, $\alpha_{LES,i}$ for total m experiments.

Step 5: Since we have the data for err and α for two methods, we can compute the mean value of the absolute error for new method and LES as $\bar{err} = \frac{\sum_{i=1}^{m} err_i}{m}$ and the mean value of the coverage ratio $\bar{\alpha} = \frac{\sum_{i=1}^{m} \alpha_i}{m}$. The uncertainty can be reported based on the variance of the err and α.

Step 6: After we post-processing the results in Step 5, we need to think about how to visualize the results in plots. The standard way is to plot the two results with bar plot, for two methods and put them together. This could provide people a straight forward way to compare two methods which one is better.

2 Problem 2

2.1 (a)

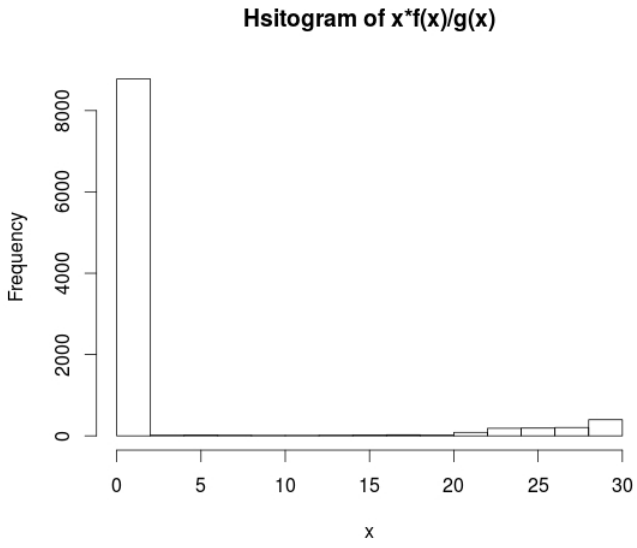
Pareto distribution is one of common heavily-tailed distributions. Its tails is not exponentially bounded thus it decays slower than the exponential distribution

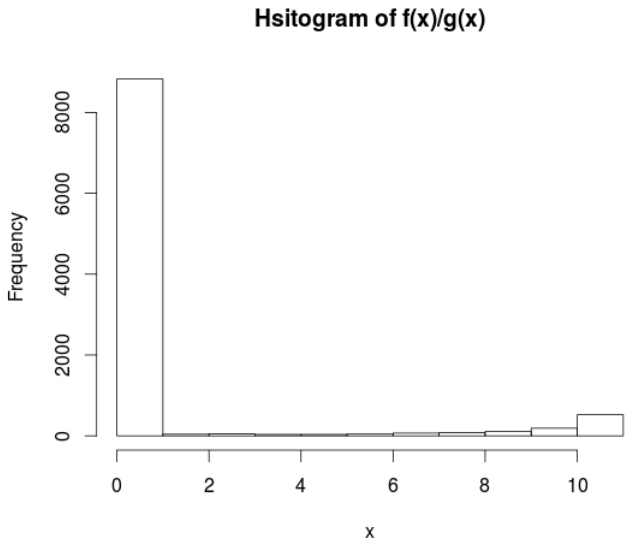
2.2 (b)

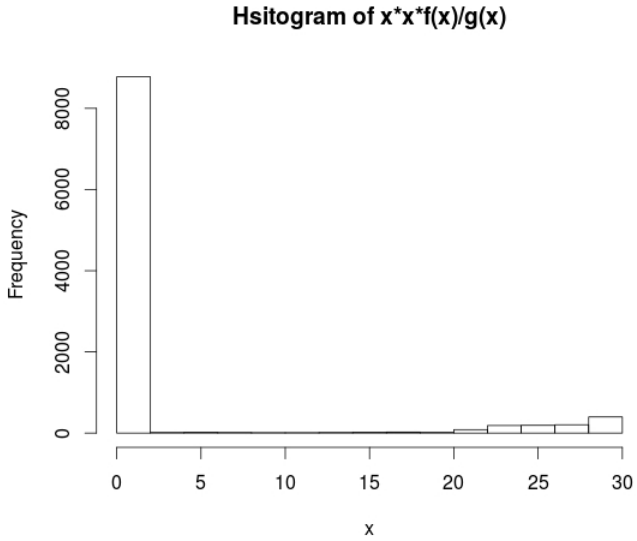
When x is from shifted exponential distribution, theoretically $E(x) = 1 + 2 = 3$, $E(x^2) = E(x) - Var(x) = 3 - 2 = 2$. The R code is straight forward,
Modify the code in part(b), we have the results

```
library(actuar)
m <- 1e4
alpha <- 2; beta <- 3;
########## (b) ##########
x <- rpareto(m, shape = beta, scale = alpha)

## h(x) = x ####
hist1 <- x * dexp(x-2, rate = 1) / dpareto(x, shape = beta, scale = alpha)
hist(hist1,main="Hsitogram of x*f(x)/g(x)",xlab="x")
hist2 <- dexp(x-2, rate = 1) / dpareto(x, shape = beta, scale = alpha)
hist(hist2,main="Hsitogram of f(x)/g(x)",xlab="x")
Ex <- 1/m * sum(x * dexp(x-2, rate = 1) / dpareto(x, shape = beta, scale = alpha))
## h(x) = x*x ####
hist3 <- x * x * dexp(x-2, rate = 1) / dpareto(x, shape = beta, scale = alpha)
hist(hist1,main="Hsitogram of x*x*f(x)/g(x)",xlab="x")
Ex2 <- 1/m * sum(x^2 * dexp(x, rate = 1) / dpareto(x, shape = beta, scale = alpha))
```







The results are correctly $Ex = 2.09923$ and $Ex2 = 1.9976$. This is right since the sampling distribution Pareto distribution has heavier tail than the shifted exponential distribution.

2.3 (c)

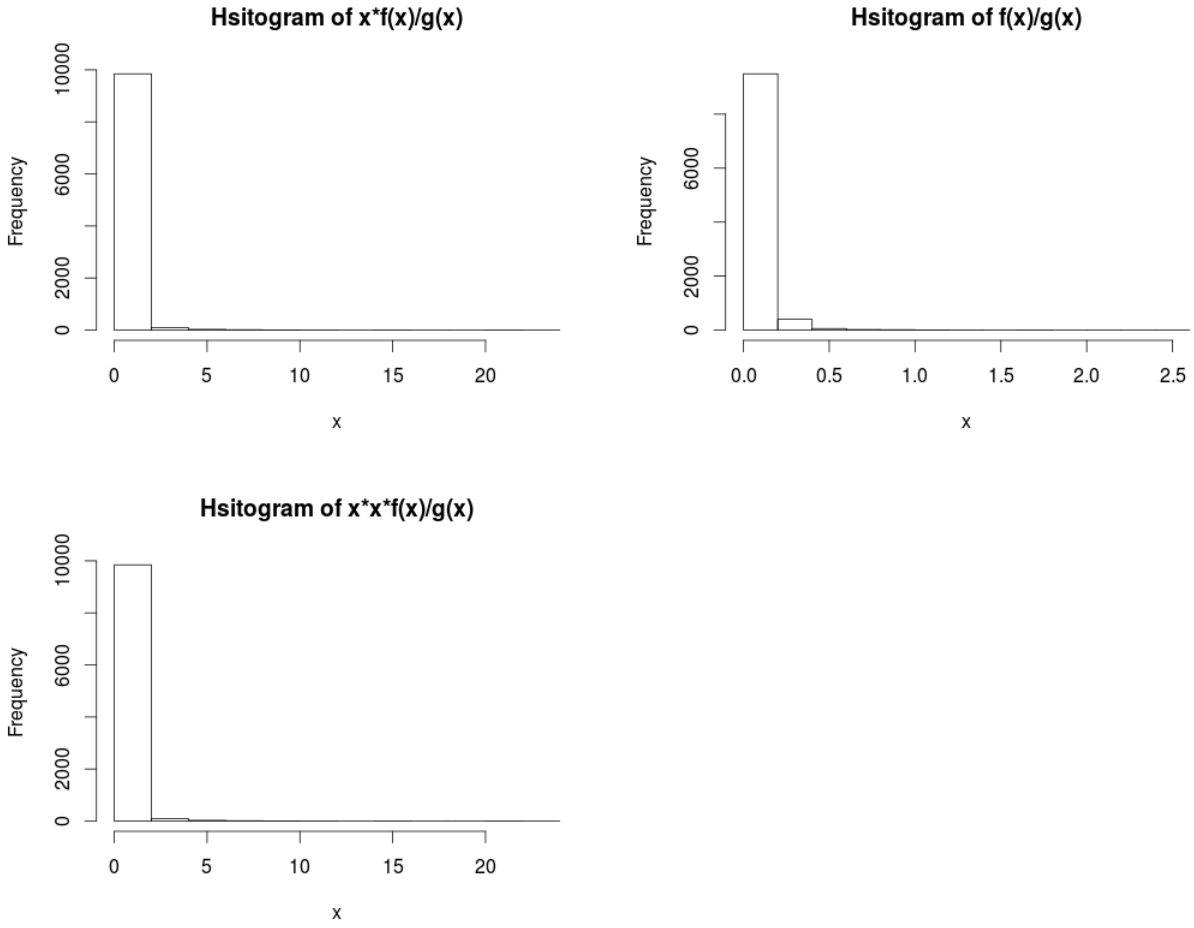
Now we switch the distribution. The code is similar

```
########## (c) ##########
x <- rexp(m, rate = 1) + 2

## h(x) = x ####
hist1 <- x * dpareto(x, shape = beta, scale = alpha)/ dexp(x-2, rate = 1)
hist(hist1,main="Hsitogram of x*f(x)/g(x)",xlab="x")
hist2 <- dpareto(x, shape = beta, scale = alpha) / dexp(x-2, rate = 1)
hist(hist2,main="Hsitogram of f(x)/g(x)",xlab="x")
Ex <- 1/m * sum(x * dpareto(x, shape = beta, scale = alpha) / dexp(x-2, rate = 1) )
```

```
## h(x) = x*x ####
hist3 <- x * x * dpareto(x, shape = beta, scale = alpha)/ dexp(x-2, rate = 1)
hist(hist1,main="Hsitogram of x*x*f(x)/g(x)",xlab="x")
Ex2 <- 1/m * sum(x^2 * dpareto(x, shape = beta, scale = alpha)/ dexp(x-2, rate = 1))

print(Ex)
print(Ex2)
```



The results are not correct. $Ex = 0.499$ and $Ex2 = 1.9435$ which are different from the theoretical value. THE reason is because the sampling distribution, the shifted exponential has lighter tails than the distribution of interests.

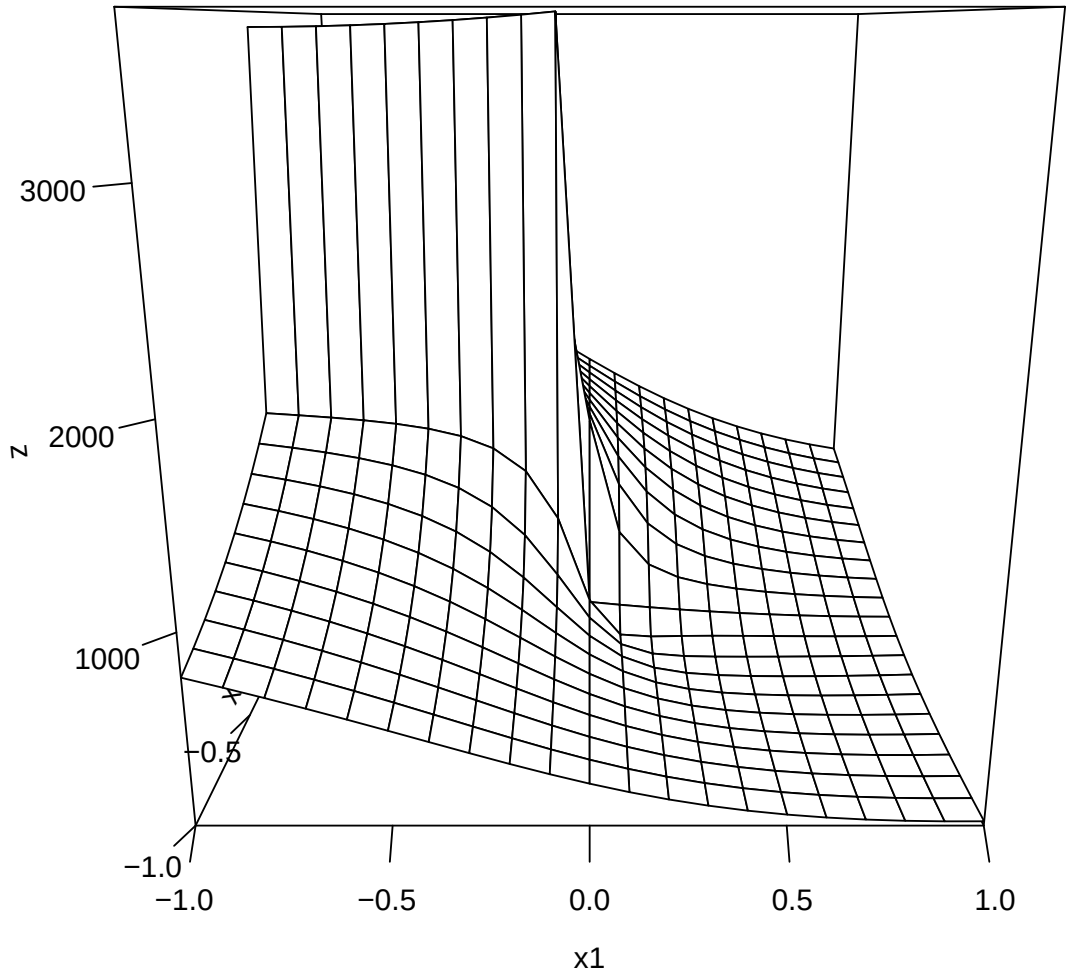
3 Problem 4

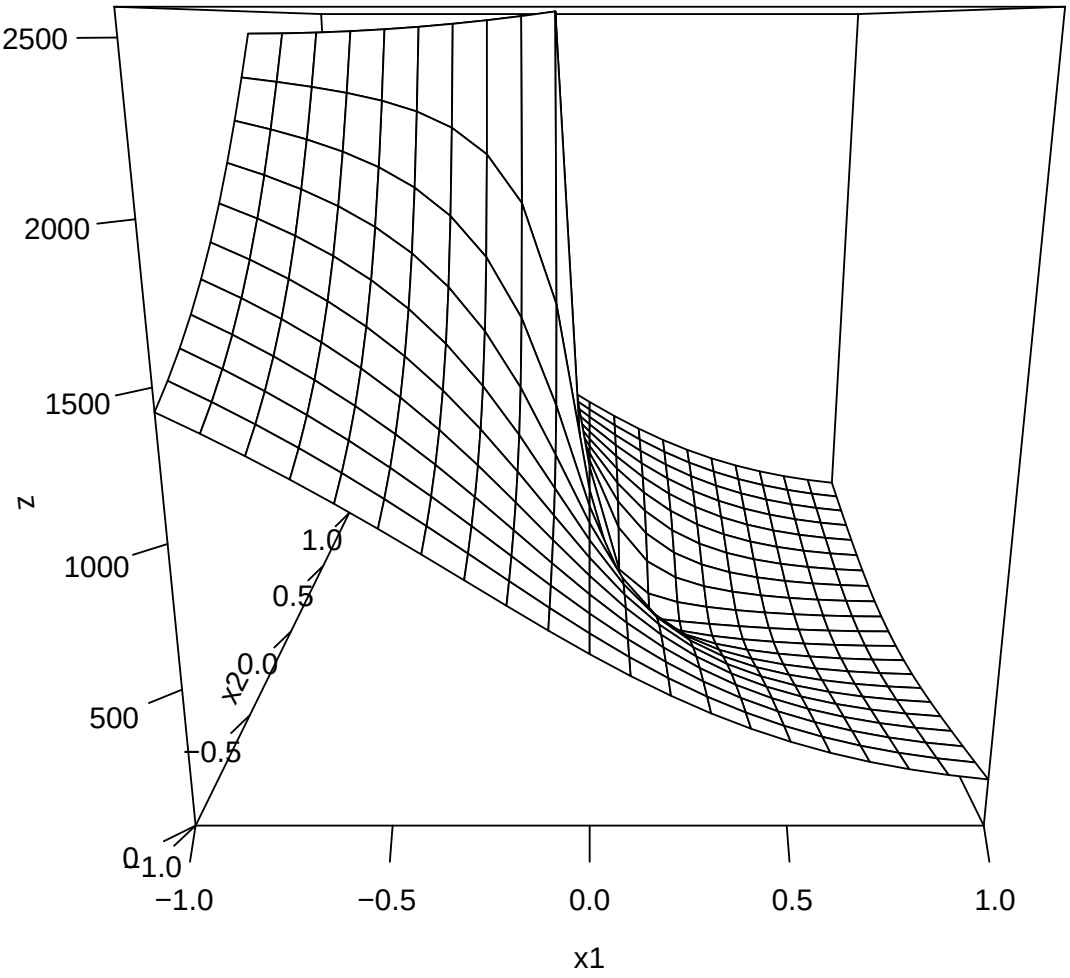
Firstly, let us plot some 2-d slices of such function. Then we can see that the optimal solution is around the origin. Then we can try several different initial conditions, the best I can get is $1.343098e-07$. Here are the code and plots

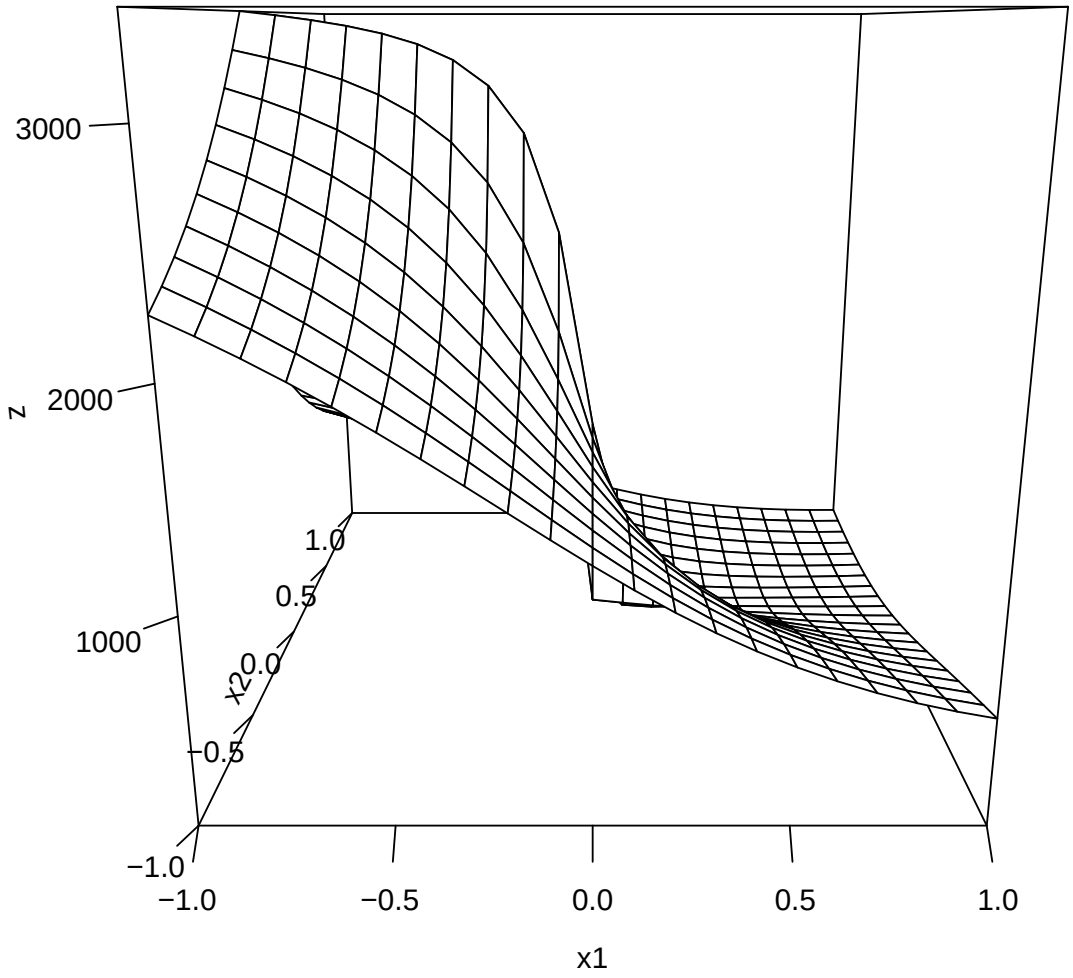
```
library(reshape)
############# Porblem 4 ########################
theta <- function(x1,x2) atan2(x2, x1)/(2*pi)
f <- function(x) {
```

```
  f1 <- 10*(x[3] - 10*theta(x[1],x[2]))
  f2 <- 10*(sqrt(x[1]^2+x[2]^2)-1)
  f3 <- x[3]
  return(f1^2+f2^2+f3^2)
}

require(reshape, quietly = TRUE)
library(lattice)
x3 <- c(-1, 0, 1)
for(i in 1:length(x3)){
  x1 <- seq(-1,1,0.1)
  x2 <- seq(-1,1,0.1)
  model = function(a,b){
    f1 <- 10*(x3[i] - 10*theta(a,b))
    f2 <- 10*(sqrt(a^2+b^2)-1)
    f3 <- x3[i]
    return(f1^2+f2^2+f3^2)
  }
  z <- outer(x1, x2, model)
  persp(x1,x2,z,ticktype="detailed")
}
```


3000
2000
1000
z
x
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
x1


2500
2000
1500
1000
500
0
z
1.0
0.5
0.0
−0.5
−1.0
x2
−1.0
−0.5
0.0
0.5
1.0
x1




```
optim(c(0,0,0), f)$value
## [1] 1.876851e-05
optim(c(1,-1,0), f)$value
## [1] 1.393654e-06
optim(c(1,1,1), f)$value
## [1] 1.343098e-07
optim(c(1,1,1), f)$value
## [1] 1.343098e-07
optim(c(1,1,0), f)$value
## [1] 9.678658e-07
```

** R code saved in HW8.R; all codes have been run and tested*