Indian Institute of Technology, Kanpur

Department of Mathematics and Statistics

Non-Linear Regression

Project-1

Author:
Deepak Singh
Roll No: 151091
sdeepak@iitk.ac.in

Supervisor:
Prof. Debasis Kundu
kundu@iitk.ac.in

February 24, 2017

GitHub Repository:
https://github.com/sdeepak09/MTH686A-Non-linear-Regression-Projects

1 Answer 1

Given Details: Given Model $y(t) = a + b * t + \epsilon(t), \quad t = 1, 2, ..., n$
$\epsilon(t)'s$ are sequence of i.i.d. Normal Random variable with mean 0 and variance 1.

Answer 1(a)

In this we have to generate the samples using the above model and compute the least squares estimators of a = 1.5 and b = 2.0 and repeat the experiment 1000 times and compute the average biases and mean squared errors. We also have to report the (5-th, 10-th, 90-th, 95-th) percentile points of the least squares estimators and we have to repeat this experiment for n = 10; 20; 30; 50.

Table 1: Average Baise, MSE and Percentile points for ‘a’

n	Average bias and MSE		Percentile Points				Length
	Average Bias	M.S.E.	5^{th}	10^{th}	90^{th}	95^{th}	of 90% C.I.
10	-0.003847626	0.44771017	0.3773969	0.6635978	2.3714075	2.6221263	2.244729
20	0.0001476761	0.201508630	0.7521162	0.9049263	2.0627360	2.2603127	1.508196
30	-0.0131966523	0.1332617241	0.8740121	1.0237945	1.9505963	2.0739810	1.199969
50	0.0007727076	0.08380423	1.029913	1.107389	1.867226	1.973537	0.943624

In the above table we can see that in all the cases average bias is almost 0 which shows that LSE is unbiased estimator in this case which confirms the fact that LSE is unbiased estimator which we have seen in the class.
We can also see that MSE is decreasing and tending to 0 w.r.t. increase in ‘n’ which shows that MSE is consistent.
In the last column we can see that the length of 90% confidence interval is decreasing w.r.t increase in ‘n’.

Table 2: Average Baise, MSE and Percentile points for ‘b’

n	Average bias and MSE		Percentile Points				Length
	Average Bias	M.S.E.	5^{th}	10^{th}	90^{th}	95^{th}	of 90% C.I.
10	0.002119260	0.01153002	1.823718	1.856493	2.140396	2.183469	0.359751
20	0.00001998868	0.001374488	1.941939	1.953543	2.047429	2.061280	0.119341
30	0.0005108111	0.0004137607	1.965252	1.975761	2.026706	2.033285	0.068033
50	-0.00002136948	0.00009340254	1.984056	1.987536	2.012102	2.015074	0.031018

In the above table we can see that in all the cases average bias is almost 0 which shows that LSE is unbiased estimator in this case which confirms the fact that LSE is unbiased estimator.
We can also see that MSE is decreasing and tending to 0 w.r.t. increase in ‘n’ which shows that MSE is consistent.We can also observe that MSE is very small w.r.t MSE of ‘a’
In the last column we can see that the length of 90% confidence interval is decreasing w.r.t increase in ‘n’ implies that as we are increasing n (i.e. we are getting more information about Population through samples) we are getting better confidence intervals.

Answer 1(b)

In this part we have to take a particular data and analyse that using R
For this exercise I have taken sample of size 100 from the above model and used R to analyse that. I have used Linear regression to analyse.
Summary of analysis is given in the following figure.

```
> summary(reslt)

Call:
lm(formula = rspnc_var ~ vc_2)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.41850 -0.63073 -0.01723  0.68983  2.16847 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.620032   0.186031   8.708 7.64e-14 ***
vc_2        1.997096   0.003198 624.447  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9232 on 98 degrees of freedom
Multiple R-squared:  0.9997,	Adjusted R-squared:  0.9997 
F-statistic: 3.899e+05 on 1 and 98 DF,  p-value: < 2.2e-16

> |
```

Figure 1: Summary of Analysis

From the above we can see that estimates of a and b are 1.620032 and 1.997096 respectively.
Both the estimates has low standard error 0.186031 and 0.003198
Both the coefficients are significant indicating that our regression is significant.
Adjusted R-squared: 0.9997 which is showing that we have captured the 99.97% of the total variance in the data.

Answer 1(c)

In this part we have to introduce the heteroscedasticity as here we have to take the variance of $\epsilon_t = t^2$ t=1,2,...,n and we have to use OLSE and GLSE on the same data and report all the outputs as in case of 1(a).

Results for OLSE:

Table 3: Average Baise, MSE and Percentile points for 'a'

n	Average bias and MSE		Percentile Points				Length
	Average Bias	M.S.E.	5^{th}	10^{th}	90^{th}	95^{th}	of 90% C.I.
10	0.03534344	8.4387428	-3.349833	-2.124721	5.295359	6.413930	9.763763
20	0.075935423	14.2201743	-4.504818	-3.011329	6.147884	7.252584	11.7574
30	-0.340180	19.4066730	-5.919434	-4.402138	6.581940	8.110079	14.02951
50	0.005359826	30.86047034	-7.236970	-5.342723	8.437100	10.913800	18.15077

In the above table we can see that in all the cases average bias is almost 0 which shows that LSE is unbiased estimator in this case .
But here we can see that MSE is increasing w.r.t. increase in 'n' which is not a good behaviour for a estimator this shows that LSE is not good in case of heteroscedasticity. This is happening as when we are increasing our n we are increasing the variablity in tha data. We can also observe in the last column that length of 90% confidence interval is increasing.

Table 4: Average Baise, MSE and Percentile points for 'b'

n	Average bias and MSE		Percentile Points				Length
	Average Bias	M.S.E.	5^{th}	10^{th}	90^{th}	95^{th}	of 90% C.I.
10	-0.01010716	0.5810276	0.7116805	1.0249862	2.9327551	3.2037209	2.49204
20	-0.008496085	0.2546161	1.190845	1.337808	2.653833	2.835903	1.645058
30	0.037608	0.164947	1.386410	1.513342	2.557068	2.721178	1.334768
50	0.010039300	0.09916675	1.469259	1.583138	2.403300	2.521293	1.052034

In case of Average Baise, MSE and Percentile points for 'b' things are good but not very much good.

Results for GLSE:

Table 5: Average Baise, MSE and Percentile points for 'a'

n	Average bias and MSE		Percentile Points				Length
	Average Bias	M.S.E.	5^{th}	10^{th}	90^{th}	95^{th}	of 90% C.I.
10	-0.04244133	1.534274	-0.59418887	-0.08114361	3.06012722	3.42111651	4.015305
20	0.01026293	1.05163757	-0.2083764	0.2106003	2.8573841	3.2627716	3.471148
30	0.017592241	0.9685703	-0.0387301	0.2430633	2.7085891	3.1637341	3.202464
50	0.0335792	0.81907818	-0.009852836	0.351264098	2.679699185	2.958678919	2.968532

In the above table we can see that in all the cases average bias is almost 0 which shows that LSE is unbiased estimator in this case.
We can also see that MSE is decreasing and tending to 0 w.r.t. increase in 'n' which shows that MSE is consistent.
In the last column we can see that the length of 90% confidence interval is decreasing w.r.t increase in 'n' implies that as we are increasing n (i.e. we are getting more information about Population through samples) we are getting better confidence intervals.

Table 6: Average Baise, MSE and Percentile points for 'b'

n	Average bias and MSE		Percentile Points				Length
	Average Bias	M.S.E.	5^{th}	10^{th}	90^{th}	95^{th}	of 90% C.I.
10	0.01250046	0.235284	1.189430	1.381211	2.633946	2.778978	1.589548
20	-0.00551923	0.08313936	1.509556	1.607253	2.354280	2.464194	0.954638
30	0.004195452	0.0476553	1.641772	1.725114	2.285871	2.362217	0.720445
50	0.00006069	0.02673401	1.721452	1.794532	2.208315	2.275536	0.554084

In the above table we can see that in all the cases average bias is almost 0 which shows that LSE is unbiased estimator in this case.
We can also see that MSE is decreasing and tending to 0 w.r.t. increase in 'n' which shows that MSE is consistent.
In the last column we can see that the length of 90% confidence interval is decreasing w.r.t increase in 'n' implies that as we are increasing n (i.e. we are getting more information about Population through samples) we are getting better confidence intervals.

2 Answer 2

Given details:
Given Model $y(t) = a + b * t + \epsilon(t), \quad t = 1, 2, ..., n$
$\epsilon(t)'s$ are sequence of i.i.d. Laplace Random variable with mean 0 and variance 5.

Answer 2(a)

Problem:
We have to generate a sample of size 10 using the above model and compute the maximum likelihood estimators of a = 1.5 and b = 2.0.
Solution:
As it is given that $\epsilon_t's$ are from Laplace Distribution which has the following form:

$$f(\epsilon|\mu, d) = \frac{1}{2d} e^{\left(-\frac{|\epsilon-\mu|}{d}\right)}$$

Please note that the variance of this distribution is $2 * d^2$ So the Likelihood function will be:

$$L(\mu, d|\epsilon_1, \epsilon_2, .., \epsilon_n) = \left(\frac{1}{2d}\right)^n e^{\left(\sum_{i=1}^{n} -\frac{|\epsilon_i-\mu|}{d}\right)}$$

As $\epsilon_i = y_i - a - b * i$ and $\mu = 0$ and $d = \sqrt(5/2)$ and n=10
So Likelihood function will now be

$$L(a, b|y_1, y_2, ..., y_{10}) = \left(\frac{1}{\sqrt{10}}\right)^{10} e^{\left(\sum_{i=1}^{10} \sqrt{2/5} * |y_i - a - b * i|\right)}$$

Now we have to maximize the above Likelihood Function. As we have only 2 variables so we can plot the Likelihood function in 3D. So for that we have to find the values of Likelihood at different combinations of 'a' and 'b'. As we know the true values of 'a' and 'b' so we can generate different random combinations of 'a' and 'b' near 'a' , 'b' and find out the value of Likelihood function at these combinations and see the behaviour of the Likelihood (In case of real problem where we don't know true values of 'a' and 'b' firstly we have to check the behaviour of the over on whole space of 'a' and 'b')
I have generated the 5000 random combinations of 'a' and 'b' and plotted the Likelihood for those. See the plot in *Figure:1* in the next page.
From *Figure:1* we can see that this Likelihood Function is Uni-Model function, So the combination which will maximise the Likelihood will be MLE.
To find out the MLE we have to find out that which combination of 'a' and 'b' is maximising the Likelihood.
I have found that $a = 2.028089, b = 1.886935$ is maximising the Likelihood function. So $a = 2.028089, b = 1.886935$ will be MLE.
MLE for $a = 1.5, b = 2.0$ **is** $a = 2.028089, b = 1.886935$

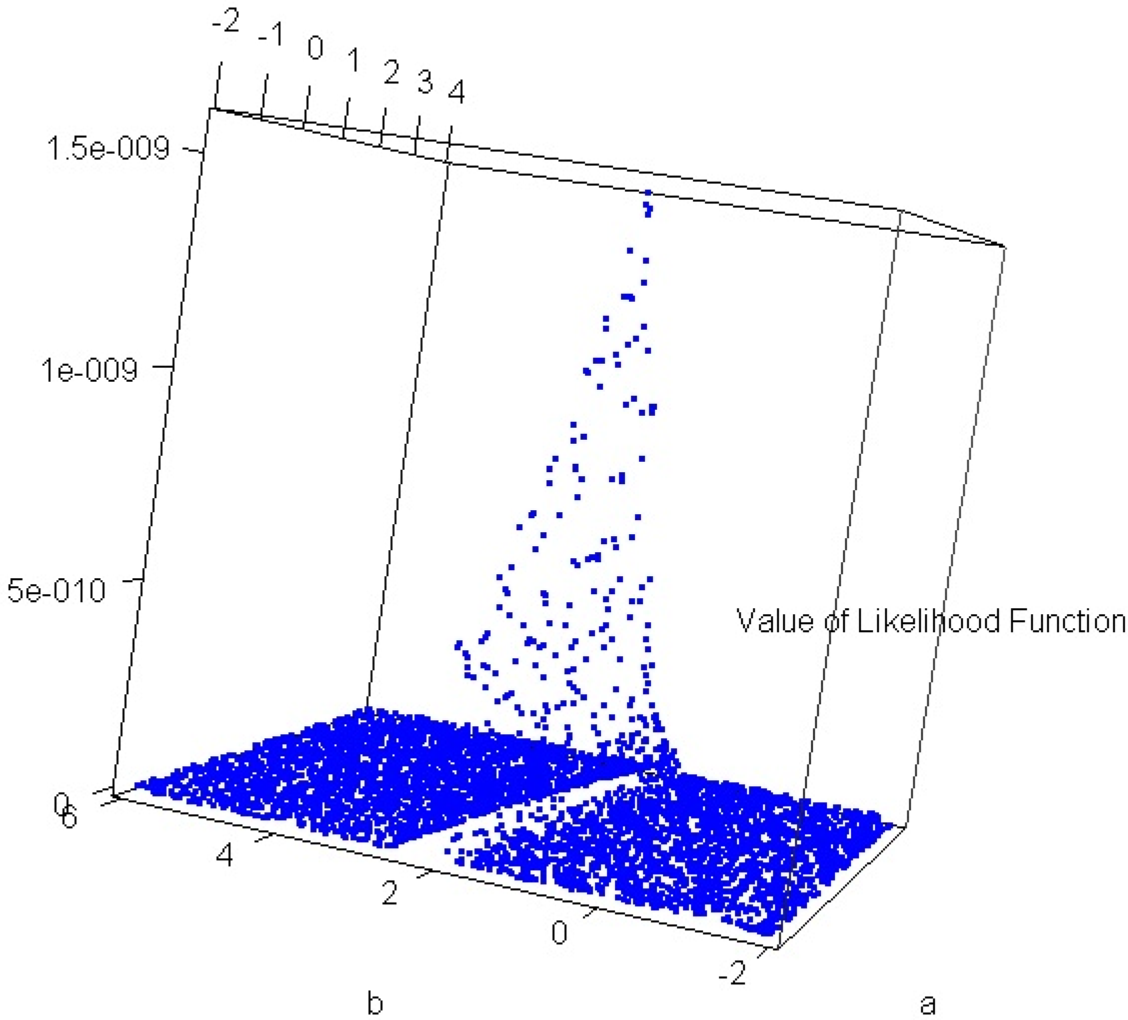

Figure 2: 3D Plot of Likelihood Function

Plotted the Value of Likelihood function on Z-axis, ‘a’ on X-axis and ‘b’ on Y-axis

Answer 2(b)

In this part we have to use some optimization routine available in R or MATLAB to solve the above problem. I have used R.
As we can see that it is problem of maximization of absolute values, So, As discussed in the class, we can use Simplex method to solve this maximization problem.
I have converted the above maximization problem as Linear Programming Problem (using the technique discussed in the class) and solve this LPP using Simplex function in R and found the following results.

```
> simplex(a=L,A3=A,b3=response_vec)

Linear Programming Results

Call : simplex(a = L, A3 = A, b3 = response_vec)

Minimization Problem with Objective Function Coefficients
 x1  x2  x3  x4  x5  x6  x7  x8  x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x20 x21 x22
  1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   0   0
x23 x24
  0   0


Optimal solution has the following values
        x1         x2         x3         x4         x5         x6         x7         x8
0.00000000 0.00000000 1.07575212 0.00000000 0.00000000 2.58826990 0.00000000 0.08042627
        x9        x10        x11        x12        x13        x14        x15        x16
1.87277573 0.00000000 0.00000000 0.49009231 0.00000000 1.96777390 0.00000000 0.00000000
       x17        x18        x19        x20        x21        x22        x23        x24
0.67844551 0.00000000 0.59022680 0.00000000 2.12700540 0.00000000 1.85514186 0.00000000
The optimal value of the objective  function is 9.34376253996461.
>
```

Figure 3: Result of LPP

In the above result x1,x2,...,x20 are variables corresponding to ϵ^+ and ϵ^-; x21, x22 are for a^+ and a^-; x23, x24 are for b^+ and b^-
we know that the estimates for 'a' will be 'estimated a^+ + estimated a^-'
So we can infer from the above result that MLE for 'a' is 2.12700540 and MLE for 'b' is 1.85514186

Answer 2(c)

In this case we have to find out the MLE in case we know the value of 'a'=1.5
In this case likelihood function will be

$$L(b|y_1, y_2, ..., y_{10}) = \left(\frac{1}{\sqrt{10}}\right)^{10} e^{\left(\sum_{i=1}^{10} \sqrt{2/5} * |y_i - 1.5 - b*i|\right)}$$

As this is the function of only one variable 'b' so we can plot the Likelihood in 2D.

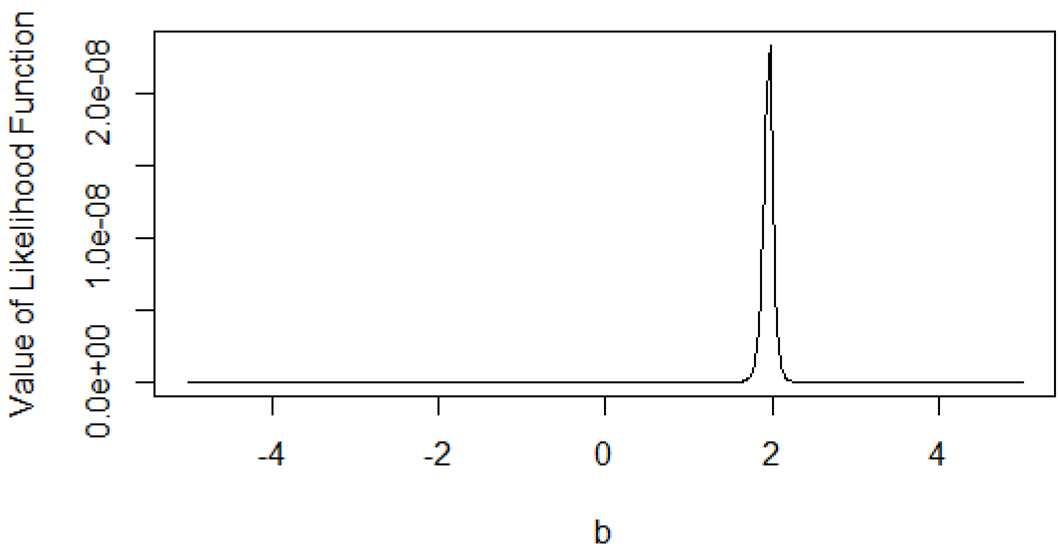

Figure 4: Plot of Likelihood Function

From the above we can infer that this Likelihood function is Uni-Model. So the value maximizing the Likelihood will be global Maximum and it will be MLE for 'b' too.
So now we have to find out the value of MLE. For that we have to plot the Likelihood near b=2
please refer to next page to see the plot of Likelihood Function near b=2

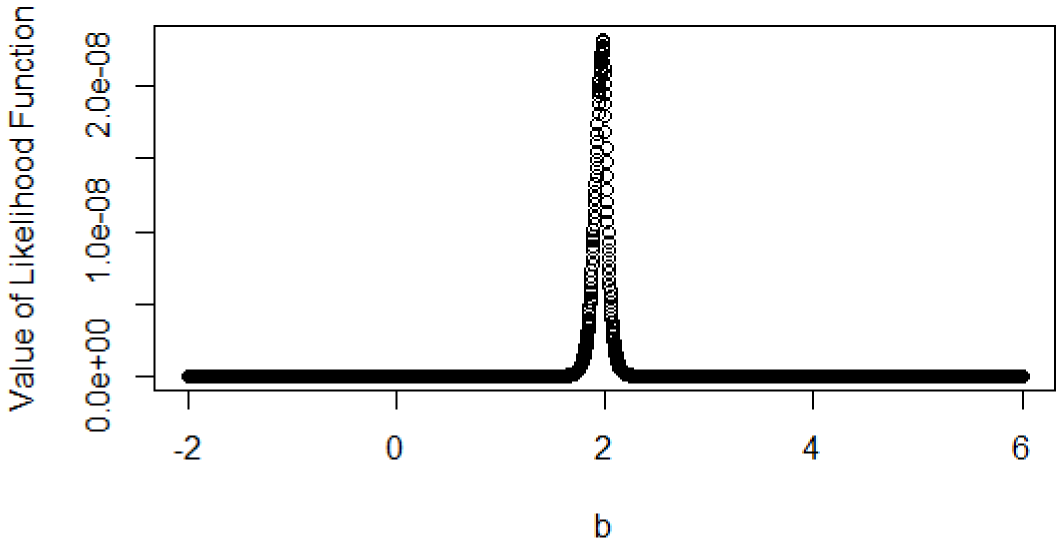

Figure 5: Plot of Likelihood Function near b=2

Here we have to find out the value at which the above plot is attaining its maximum. I have found that the above plot is taking its maximum at $b = 1.977326$
So **MLE of b is 1.977326** which looks like good as it is close to 2.

3 Appendix

R Code

```
## Starting 04-02-2017
## Model we have y(t) = 1.5 + 2.0t + epsilon_(t);

## Function for LSE will return least Sqaure Estimates
lse=function(dsgn_mat, resp_var){
  if(!is.matrix(dsgn_mat)){
    dsgn_mat=as.matrix(dsgn_mat)
  }
  if(!is.vector(resp_var)){
    resp_var=as.vector(resp_var)
  }
  est_coef=solve(t(dsgn_mat)%*%dsgn_mat)%*%t(dsgn_mat)%*%resp_var
  return(as.numeric(est_coef))
}

gen_lse=function(dsgn_mat, resp_var, sigma){
  if(!is.matrix(dsgn_mat)){
    dsgn_mat=as.matrix(dsgn_mat)
  }
  if(!is.vector(resp_var)){
    resp_var=as.vector(resp_var)
  }
  sigma_inv=solve(sigma)
  est_coef=solve(t(dsgn_mat)%*%sigma_inv%*%dsgn_mat)%*%t(dsgn_mat)%*%sigma_inv
    %*%resp_var
  return(as.numeric(est_coef))
}


q1=function(n, r, true_a, true_b, prntile){
  results=list()
  est_coefs=list()
  mean_bias=list()
  mean_sqrd_error=list()
  percentile_points_for_a=list()
  percentile_points_for_b=list()
  true_coef_mat=data.frame(true_a=c(rep(true_a, r)), true_b=c(rep(true_b, r)))
  true_coef_mat=as.matrix(true_coef_mat)
  for(i in 1:length(n)){
    est_coef=matrix(,nrow = r, ncol = 2)
    est_coef_names=paste0("est_coef_for_n=",n[i])
    mean_bias_names=paste0("mean_bias_for_n=",n[i])
    mean_sqr_names=paste0("mean_sqrd_error_for_n=",n[i])
    percnt_names_a=paste0("percentile_points_for_a_for_n=",n[i])
    percnt_names_b=paste0("percentile_points_for_b_for_n=",n[i])
    for(j in 1:r){
      dsgn_mat_1=c(rep(1,n[i]))
      dsgn_mat_2=c(1:n[i])
      dsgn_mat=data.frame(dsgn_mat_1, dsgn_mat_2)
      dsgn_mat=as.matrix(dsgn_mat)
      epsn=rnorm(n[i],0,1)
      resp_vec=true_a*dsgn_mat[,1]+true_b*dsgn_mat[,2]+epsn
      est_coef[j,]=lse(dsgn_mat, resp_vec)
    }
    est_coefs[[est_coef_names]]=est_coef
    bias_mat=est_coef-true_coef_mat
    bias_2_mat=bias_mat^2
    mean_bias_vec=colMeans(bias_mat)
```

```
    mean_sqrd_vec=colMeans(bias_2_mat)
    mean_bias[[mean_bias_names]]=mean_bias_vec
    mean_sqrd_error[[mean_sqr_names]]=mean_sqrd_vec
    vec_for_a=est_coef[,1]
    ordrd_vec_for_a=sort(vec_for_a)
    perctl_vec_for_a=ordrd_vec_for_a[c(r*prntile)]
    percentile_points_for_a[[percnt_names_a]]=perctl_vec_for_a
    vec_for_b=est_coef[,2]
    ordrd_vec_for_b=sort(vec_for_b)
    perctl_vec_for_b=ordrd_vec_for_b[c(r*prntile)]
    percentile_points_for_b[[percnt_names_b]]=perctl_vec_for_b
  }
  results=list(est_coefs=est_coefs,mean_bias=mean_bias,mean_sqrd_error=mean_
      sqrd_error,percentile_points_for_a=percentile_points_for_a,percentile_
      points_for_b=percentile_points_for_b)
  return(results)
}
x1=q1(c(10,20,30,50),1000,1.5,2,c(0.05,0.1,0.9,0.95))
x1$mean_bias
x1$mean_sqrd_error
x1$percentile_points_for_a
x1$percentile_points_for_b


# Answer 1(b)
rand_vc=rnorm(100,0,1)
vc_1=rep(1,100)
vc_2=c(1:100)
rspnc_var=1.5*vc_1+2.0*vc_2+rand_vc
dsn_mat=as.matrix(data.frame(vc_1,vc_2))
reslt=lm(rspnc_var~vc_2)
reslt
summary(reslt)

q1_c=function(n,r,true_a,true_b,prntile){
  results=list()
  est_coefs=list()
  mean_bias=list()
  mean_sqrd_error=list()
  percentile_points_for_a=list()
  percentile_points_for_b=list()
  true_coef_mat=data.frame(true_a=c(rep(true_a,r)),true_b=c(rep(true_b,r)))
  true_coef_mat=as.matrix(true_coef_mat)
  for(i in 1:length(n)){
    est_coef=matrix(,nrow = r,ncol = 2)
    est_coef_names=paste0("est_coef_for_n=",n[i])
    mean_bias_names=paste0("mean_bias_for_n=",n[i])
    mean_sqr_names=paste0("mean_sqrd_error_for_n=",n[i])
    percnt_names_a=paste0("percentile_points_for_a_for_n=",n[i])
    percnt_names_b=paste0("percentile_points_for_b_for_n=",n[i])
    for(j in 1:r){
      dsgn_mat_1=c(rep(1,n[i]))
      dsgn_mat_2=c(1:n[i])
      dsgn_mat=data.frame(dsgn_mat_1,dsgn_mat_2)
      dsgn_mat=as.matrix(dsgn_mat)
      epsn=c()
      for(k in 1:n[i]){
        epsn[k]=rnorm(1,0,k)

      }
      resp_vec=true_a*dsgn_mat[,1]+true_b*dsgn_mat[,2]+epsn
      est_coef[j,]=lse(dsgn_mat,resp_vec)
```

```
      }
      est_coefs[[est_coef_names]]=est_coef
      bias_mat=est_coef-true_coef_mat
      bias_2_mat=bias_mat^2
      mean_bias_vec=colMeans(bias_mat)
      mean_sqrd_vec=colMeans(bias_2_mat)
      mean_bias[[mean_bias_names]]=mean_bias_vec
      mean_sqrd_error[[mean_sqr_names]]=mean_sqrd_vec
      vec_for_a=est_coef[,1]
      ordrd_vec_for_a=sort(vec_for_a)
      perctl_vec_for_a=ordrd_vec_for_a[c(r*prntile)]
      percentile_points_for_a[[percnt_names_a]]=perctl_vec_for_a
      vec_for_b=est_coef[,2]
      ordrd_vec_for_b=sort(vec_for_b)
      perctl_vec_for_b=ordrd_vec_for_b[c(r*prntile)]
      percentile_points_for_b[[percnt_names_b]]=perctl_vec_for_b
   }
   results=list(est_coefs=est_coefs,mean_bias=mean_bias,mean_sqrd_error=mean_sqrd_error,percentile_points_for_a=percentile_points_for_a,percentile_points_for_b=percentile_points_for_b)
   return(results)
}
x_c=q1_c(c(10,20,30,50),1000,1.5,2,c(0.05,0.1,0.9,0.95))
x_c$mean_bias
x_c$mean_sqrd_error
x_c$percentile_points_for_a
x_c$percentile_points_for_b


q1_c_2=function(n,r,true_a,true_b,prntile){
   results=list()
   est_coefs=list()
   mean_bias=list()
   mean_sqrd_error=list()
   percentile_points_for_a=list()
   percentile_points_for_b=list()
   true_coef_mat=data.frame(true_a=c(rep(true_a,r)),true_b=c(rep(true_b,r)))
   true_coef_mat=as.matrix(true_coef_mat)
   for(i in 1:length(n)){
      est_coef=matrix(,nrow = r,ncol = 2)
      est_coef_names=paste0("est_coef_for_n=",n[i])
      mean_bias_names=paste0("mean_bias_for_n=",n[i])
      mean_sqr_names=paste0("mean_sqrd_error_for_n=",n[i])
      percnt_names_a=paste0("percentile_points_for_a_for_n=",n[i])
      percnt_names_b=paste0("percentile_points_for_b_for_n=",n[i])
      for(j in 1:r){
         dsgn_mat_1=c(rep(1,n[i]))
         dsgn_mat_2=c(1:n[i])
         dsgn_mat=data.frame(dsgn_mat_1,dsgn_mat_2)
         dsgn_mat=as.matrix(dsgn_mat)
         epsn=c()
         for(k in 1:n[i]){
            epsn[k]=rnorm(1,0,k)

         }
         resp_vec=true_a*dsgn_mat[,1]+true_b*dsgn_mat[,2]+epsn
         sigma=diag(c(1:n[i])^2)
         est_coef[j,]=gen_lse(dsgn_mat,resp_vec,sigma)
      }
      est_coefs[[est_coef_names]]=est_coef
      bias_mat=est_coef-true_coef_mat
      bias_2_mat=bias_mat^2
```

```
    mean_bias_vec=colMeans(bias_mat)
    mean_sqrd_vec=colMeans(bias_2_mat)
    mean_bias[[mean_bias_names]]=mean_bias_vec
    mean_sqrd_error[[mean_sqr_names]]=mean_sqrd_vec
    vec_for_a=est_coef[,1]
    ordrd_vec_for_a=sort(vec_for_a)
    perctl_vec_for_a=ordrd_vec_for_a[c(r*prntile)]
    percentile_points_for_a[[percnt_names_a]]=perctl_vec_for_a
    vec_for_b=est_coef[,2]
    ordrd_vec_for_b=sort(vec_for_b)
    perctl_vec_for_b=ordrd_vec_for_b[c(r*prntile)]
    percentile_points_for_b[[percnt_names_b]]=perctl_vec_for_b
  }
  results=list(est_coefs=est_coefs,mean_bias=mean_bias,mean_sqrd_error=mean_
      sqrd_error,percentile_points_for_a=percentile_points_for_a,percentile_
      points_for_b=percentile_points_for_b)
  return(results)
}
x_c_2_1=q1_c_2(c(10,20,30,50),1000,1.5,2,c(0.05,0.1,0.9,0.95))
x_c_2_1$mean_bias
x_c_2_1$mean_sqrd_error
x_c_2_1$percentile_points_for_a
x_c_2_1$percentile_points_for_b


########## Answer 2
install.packages("boot")
library(boot)
install.packages("smoothmest")
library(smoothmest)

vec_1=1.5*rep(1,10)
vec_2=2*c(1:10)
rand_vec=rlaplace(10, m=0, s=sqrt(5/2))
## I am giving s=sqrt(5/2) as variance in the case of laplace distribution is
    2*s^2
response_vec=vec_1+vec_2+rand_vec

a1=runif(5000,-2,4)
b1=runif(5000,-2,6)
a_b=as.matrix(data.frame(a1,b1))
Liklhd_fun1=function(b){
  sum1=0
  for(i in 1:10){
    sum1=sum1+abs(response_vec[i]-b[1]-b[2]*i)
  }
  lik=(1/sqrt(10))^10*exp(-sqrt(2/5)*sum1)
  return(lik)
}
fun_value=function(value){
  values=c(0,nrow(value))
  for(j in 1:nrow(value)){
    values[j]=Liklhd_fun1(as.vector(value[j,]))
  }
  return(values)

}
y1=fun_value(a_b)
library(rgl)
plot3d(x=a1,y=b1,z=y1,col="blue",xlab = "a",ylab = "b",zlab = "Value of
    Likelihood Function")
a_b[which(y1==max(y1)),] ##2.028089 1.886935
```

```r

L=c(rep(1,20),0,0,0,0)
A_1=rep(c(1,-1,rep(0,20)),10)
A_2=matrix(A_1,ncol = 20,byrow = T)
A_2=A_2[-11,]
A=data.frame(A_2,rep(1,10),-rep(1,10),c(1:10),-c(1:10))
A=as.matrix(A)
simplex(a=L,A3=A,b3=response_vec)

Liklhd_fun=function(b){
  sum1=0
  for(i in 1:10){
    sum1=sum1+abs(response_vec[i]-1.5-b*i)
  }
  lik=(1/sqrt(10))^10*exp(-sqrt(2/5)*sum1)
  return(lik)
}
curve(Liklhd_fun, from=-5, to=5, n=1000,xlab = "b",ylab = "Value of Likelihood
    Function")
fun_value1=function(value){
  values=c(0,length(value))
  for(j in 1:length(value)){
    values[j]=Liklhd_fun(as.vector(value[j]))
  }
  return(values)

}
b_2=seq(-2,6,length.out = 3000)
lik_value=fun_value1(b_2)
plot(y=lik_value,x=b_2, xlab = "b",ylab = "Value of Likelihood Function")
b_2[which(lik_value==max(lik_value))] ##1.977326
```

Project_1_codes.R