# CompStatsHW3

Henrique Cheng

2/17/2021

# Problem 1

```
p = 5
rho = .8
Sigma = matrix(rep(rho,p^2),ncol=p)
diag(Sigma) = 1.0
Sigma
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  1.0  0.8  0.8  0.8  0.8
## [2,]  0.8  1.0  0.8  0.8  0.8
## [3,]  0.8  0.8  1.0  0.8  0.8
## [4,]  0.8  0.8  0.8  1.0  0.8
## [5,]  0.8  0.8  0.8  0.8  1.0
```

a. The marginal distribution of $(Y_1, Y_2)$ is

$$\text{MVN}(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1 & 0.8\\0.8 & 1\end{pmatrix})$$

b. From the notes, we have the formula

$$Y|X = x \sim \text{NORM}(\mu_y + \Sigma_{yx}\Sigma_{xx}^{-1}(x - \mu_x), \Sigma_{yy} - \Sigma_{yx}\Sigma_{xx}^{-1}\Sigma_{xy})$$

Observe that

$$\Sigma_{yx} = E([\begin{pmatrix}Y_1\\Y_2\end{pmatrix} - \begin{pmatrix}\mu_1\\\mu_2\end{pmatrix}][\begin{pmatrix}Y_3\\Y_4\\Y_5\end{pmatrix} - \begin{pmatrix}\mu_3\\\mu_4\\\mu_5\end{pmatrix}]^T = \begin{pmatrix}E(Y_1-\mu_1)(Y_3-\mu_3) & E(Y_1-\mu_1)(Y_4-\mu_4) & E(Y_1-\mu_1)(Y_5-\mu_5)\\E(Y_2-\mu_2)(Y_3-\mu_3) & E(Y_2-\mu_2)(Y_4-\mu_4) & E(Y_2-\mu_2)(Y_5-\mu_5)\end{pmatrix}$$
$$= \begin{pmatrix}0.8 & 0.8 & 0.8\\0.8 & 0.8 & 0.8\end{pmatrix}$$

Thus by filling in the appropriate parts we have

$$\mu_{Y|X} = \begin{pmatrix}0\\0\end{pmatrix} + \begin{pmatrix}0.8 & 0.8 & 0.8\\0.8 & 0.8 & 0.8\end{pmatrix}\begin{pmatrix}1 & 0.8 & 0.8\\0.8 & 1 & 0.8\\0.8 & 0.8 & 1\end{pmatrix}^{-1}\begin{pmatrix}0.23\\-0.65\\-0.30\end{pmatrix}$$
$$\Sigma_{Y|X} = \begin{pmatrix}1 & 0.8\\0.8 & 1\end{pmatrix} - \begin{pmatrix}0.8 & 0.8 & 0.8\\0.8 & 0.8 & 0.8\end{pmatrix}\begin{pmatrix}1 & 0.8 & 0.8\\0.8 & 1 & 0.8\\0.8 & 0.8 & 1\end{pmatrix}^{-1}\begin{pmatrix}0.8 & 0.8\\0.8 & 0.8\\0.8 & 0.8\end{pmatrix}$$

Computing this in R we have

```
muy=matrix(0,ncol=1,nrow=2)
Sigyx=matrix(0.8,ncol=3,nrow=2)
Sigxx=matrix(0.8,ncol=3,nrow=3)
diag(Sigxx)=1
Sigyy=matrix(0.8,ncol=2,nrow=2)
diag(Sigyy)=1
x=matrix(c(0.23,-0.65,-0.30),ncol=1,nrow=3)

muYX=muy+Sigyx%*%solve(Sigxx)%*%x
SigYX=Sigyy-Sigyx%*%solve(Sigxx)%*%t(Sigyx)
muYX
```

```
##            [,1]
## [1,] -0.2215385
## [2,] -0.2215385
```

```
SigYX
```

```
##            [,1]       [,2]
## [1,] 0.26153846 0.06153846
## [2,] 0.06153846 0.26153846
```

c.

```
L=t(chol(Sigma))
L
```

```
##      [,1]      [,2]      [,3]      [,4]      [,5]
## [1,]  1.0 0.0000000 0.0000000 0.0000000 0.0000000
## [2,]  0.8 0.6000000 0.0000000 0.0000000 0.0000000
## [3,]  0.8 0.2666667 0.5374838 0.0000000 0.0000000
## [4,]  0.8 0.2666667 0.1653796 0.5114083 0.0000000
## [5,]  0.8 0.2666667 0.1653796 0.1203314 0.4970501
```

d.

```
LInv=solve(L)
LInv
```

```
##            [,1]       [,2]       [,3]       [,4]    [,5]
## [1,]  1.0000000  0.0000000  0.0000000  0.0000000 0.00000
## [2,] -1.3333333  1.6666667  0.0000000  0.0000000 0.00000
## [3,] -0.8268982 -0.8268982  1.8605210  0.0000000 0.00000
## [4,] -0.6016568 -0.6016568 -0.6016568  1.9553847 0.00000
## [5,] -0.4733811 -0.4733811 -0.4733811 -0.4733811 2.01187
```

e.

```
svDec=svd(Sigma)
P=svDec$u
DHalf=matrix(0,nrow=5,ncol=5)
diag(DHalf)=sqrt(svDec$d)
A=P%*%DHalf
A
```

```
##           [,1]         [,2]        [,3]         [,4] [,5]
## [1,] -0.9165151 -1.899137e-17  0.00000000  1.139482e-16  0.4
## [2,] -0.9165151 -4.767313e-02  0.04767313  3.813850e-01 -0.1
## [3,] -0.9165151  2.701477e-01 -0.27014773 -6.356417e-02 -0.1
## [4,] -0.9165151 -3.348441e-01 -0.11236950 -1.589104e-01 -0.1
## [5,] -0.9165151  1.123695e-01  0.33484410 -1.589104e-01 -0.1
```

f.

```
library(MASS)
n=10000
mu=rep(0,5)
set.seed(19)
sample=mvrnorm(n,mu,Sigma)
#MLE for mean
muEst=colMeans(sample)
muEst
```

```
## [1] 0.007197000 0.004376895 0.007126411 0.004410131 0.005175354
```

```
#MLE for covariance matrix
sigEst=cov(sample)
sigEst
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.9772871 0.7732417 0.7823810 0.7950968 0.7852651
## [2,] 0.7732417 0.9706822 0.7803051 0.7887291 0.7799930
## [3,] 0.7823810 0.7803051 0.9884352 0.7980218 0.7867896
## [4,] 0.7950968 0.7887291 0.7980218 1.0038474 0.7967481
## [5,] 0.7852651 0.7799930 0.7867896 0.7967481 0.9884991
```

g.

```
n2=50
set.seed(2)
sample2=mvrnorm(n2,mu,Sigma)
#MLE for mean
muEst2=colMeans(sample2)
muEst2
```

```
## [1] -0.02252608 -0.15224481 -0.09121910 -0.08188604  0.03104630
```

```
#MLE for covariance matrix
sigEst2=cov(sample2)
sigEst2
```

```
##           [,1]      [,2]      [,3]     [,4]     [,5]
## [1,] 1.2442130 1.0556385 0.9455597 1.029563 1.035010
## [2,] 1.0556385 1.3477810 0.9599075 1.032719 1.112125
## [3,] 0.9455597 0.9599075 1.0931326 0.962343 1.056209
## [4,] 1.0295632 1.0327193 0.9623430 1.237229 1.078766
## [5,] 1.0350101 1.1121251 1.0562086 1.078766 1.341789
```

For a sample size of 10K the MLE is fairly close to the true values, but for a sample size of 50 the results are a bit dodgy

h.

Since the MLE for $\Sigma$ is the sample covariance $\frac{A}{n}$, to get the MLE for L, one needs to solve for L in the expression

$$LL' = \frac{A}{n}$$

This entails computing the Cholesky decomposition for the sample covariance matrix. The MLE's for the 10K and 50 samples are as follows

```
LEst10K=t(chol(sigEst))
LEst10K
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 0.9885783 0.0000000 0.0000000 0.0000000 0.0000000
## [2,] 0.7821754 0.5990691 0.0000000 0.0000000 0.0000000
## [3,] 0.7914204 0.2692102 0.5381587 0.0000000 0.0000000
## [4,] 0.8042831 0.2664778 0.1667859 0.5080829 0.0000000
## [5,] 0.7943378 0.2648801 0.1613391 0.1188426 0.4972033
```

```
LEst50=t(chol(sigEst2))
LEst50
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.1154430 0.0000000 0.0000000 0.0000000 0.0000000
## [2,] 0.9463850 0.6724108 0.0000000 0.0000000 0.0000000
## [3,] 0.8476988 0.2344667 0.5653005 0.0000000 0.0000000
## [4,] 0.9230084 0.2367569 0.2200568 0.5299109 0.0000000
## [5,] 0.9278916 0.3479753 0.3326503 0.1259209 0.4829156
```

# Problem 2

The algorithm seems correct. Here is the original function followed by a vectorized version

```
logL=function(Y,mu,Sigma)
{
   n = nrow(Y); p=ncol(Y)
   dS = det(Sigma)
   retval = -.5*n*log(dS)
   Si = solve(Sigma)
   for(i in 1:n) {
      yi = matrix(Y[i,]-mu,ncol=1)
      retval = retval - .5 * t(yi) %*% Si %*% yi
   }
   return(retval)
}

logLVec=function(Y,mu,Sigma)
{
  n = nrow(Y); p=ncol(Y)
  dS = det(Sigma)
  retval = -.5*n*log(dS)
  Si = solve(Sigma)
  Ybar=colMeans(Y)
  Yminmu=(t(Y)-mu)
  retval=retval-.5*(sum(diag(t(Yminmu)%*%Si%*%Yminmu)))
  return(retval)
}

#Check the speed with "sample2" from problem 1
library(microbenchmark)
```

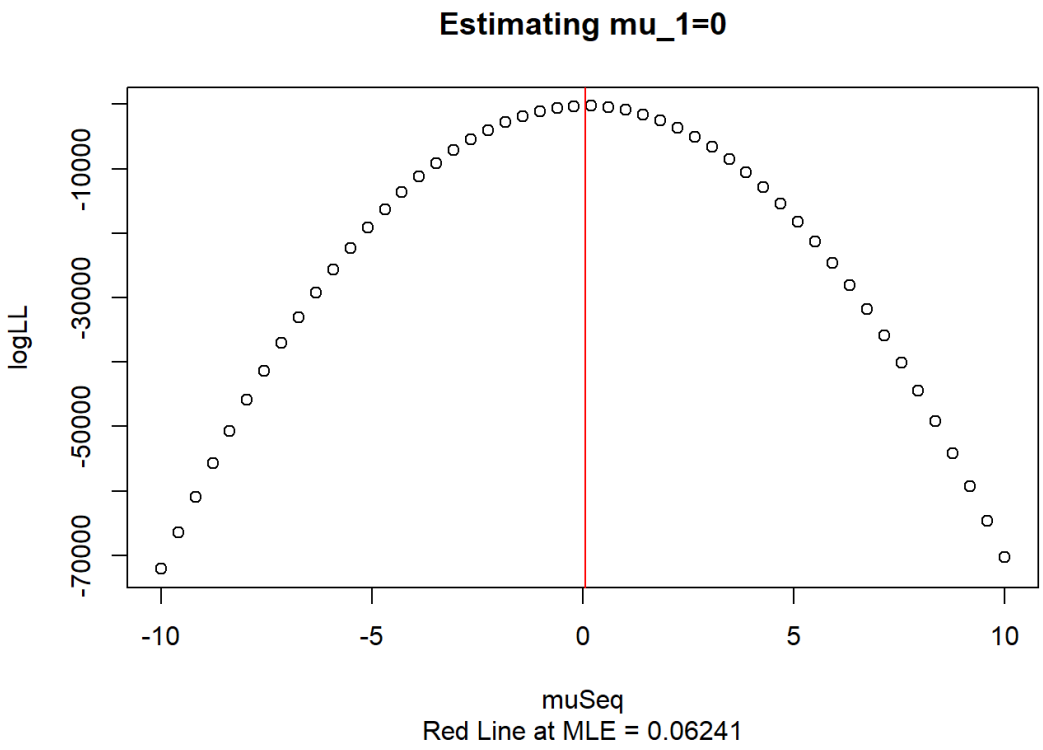
```
## Warning: package 'microbenchmark' was built under R version 4.0.3
```

```
microbenchmark(logL(sample2,mu,Sigma),logLVec(sample2,mu,Sigma))
```

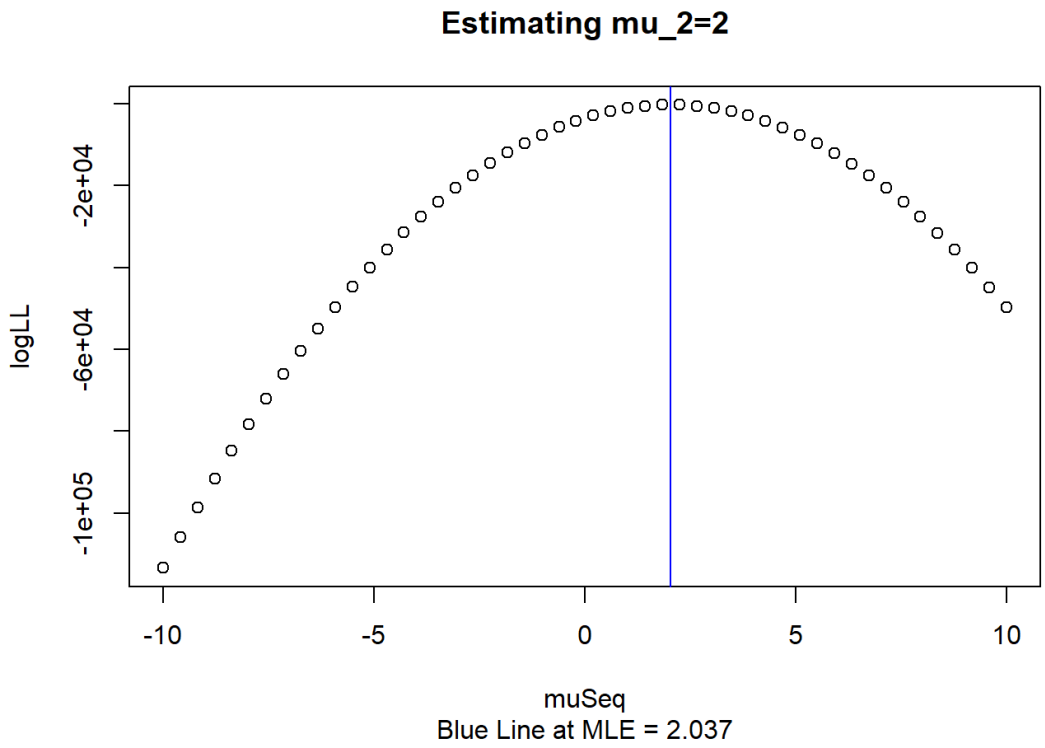
```
## Unit: microseconds
##                         expr   min    lq     mean  median      uq     max neval
##     logL(sample2, mu, Sigma) 697.7 910.9 1400.821 1088.85 1167.70 32255.5   100
##  logLVec(sample2, mu, Sigma) 126.4 170.6 1105.052  198.70  227.45 82991.8   100
```

The stats on the vectorized version of the code are much better. Now on to some plotting!

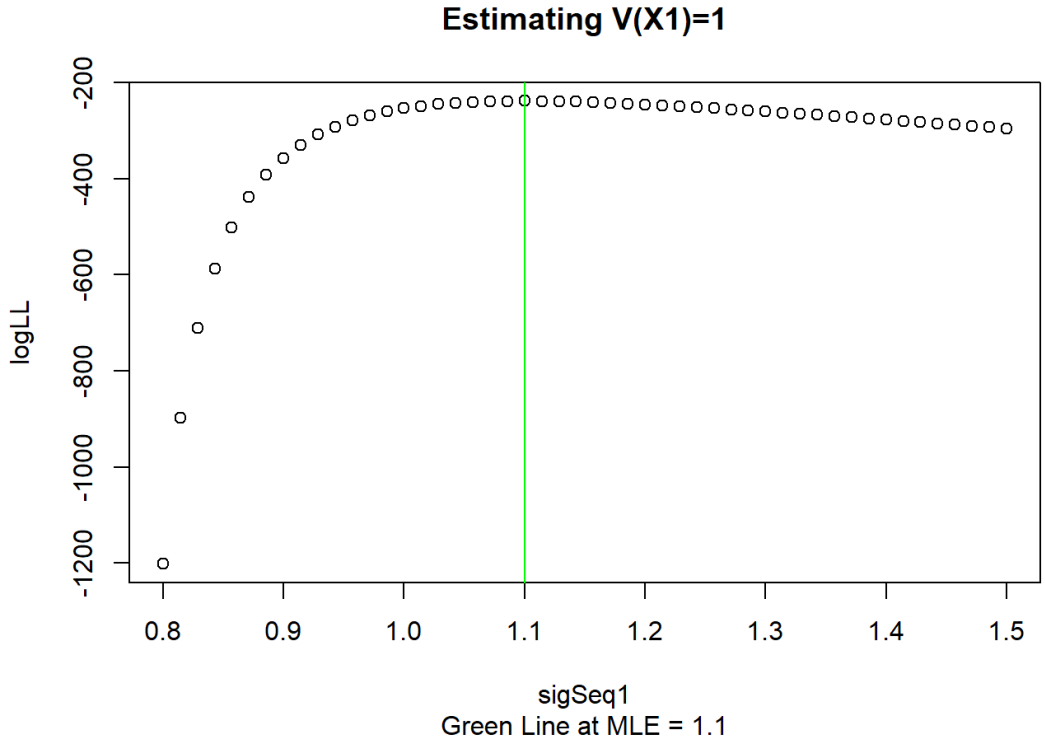
```
mu2=c(0,2)
Sigma2=Sigma[1:2,1:2]
n3=500
set.seed(3)
sample3=mvrnorm(n3,mu2,Sigma2)
mu2Est=colMeans(sample3)
sig2Est=cov(sample3)
muSeq=seq(from=-10,to=10,length.out=50)
logLLmu1=rep(NA,50)
logLLmu2=rep(NA,50)
for(i in 1:length(muSeq))
{
  logLLmu1[i]=logLVec(sample3,c(muSeq[i],mu2Est[2]),sig2Est)
  logLLmu2[i]=logLVec(sample3,c(mu2Est[1],muSeq[i]),sig2Est)
}

plot(muSeq,logLLmu1,main="Estimating mu_1=0",ylab="logLL",
     sub=paste("Red Line at MLE =",signif(mu2Est[1],4)))
abline(v=mu2Est[1],col="red")
```



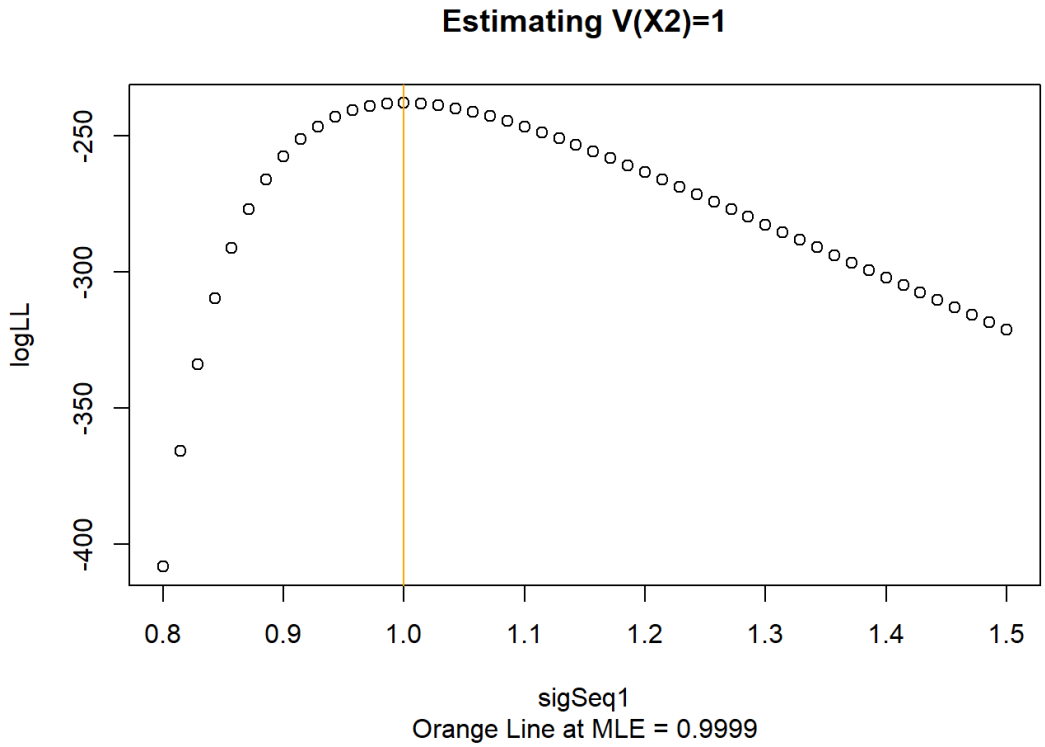

```
plot(muSeq,logLLmu2,main="Estimating mu_2=2",ylab="logLL",
     sub=paste("Blue Line at MLE =",signif(mu2Est[2],4)))
abline(v=mu2Est[2],col="blue")
```



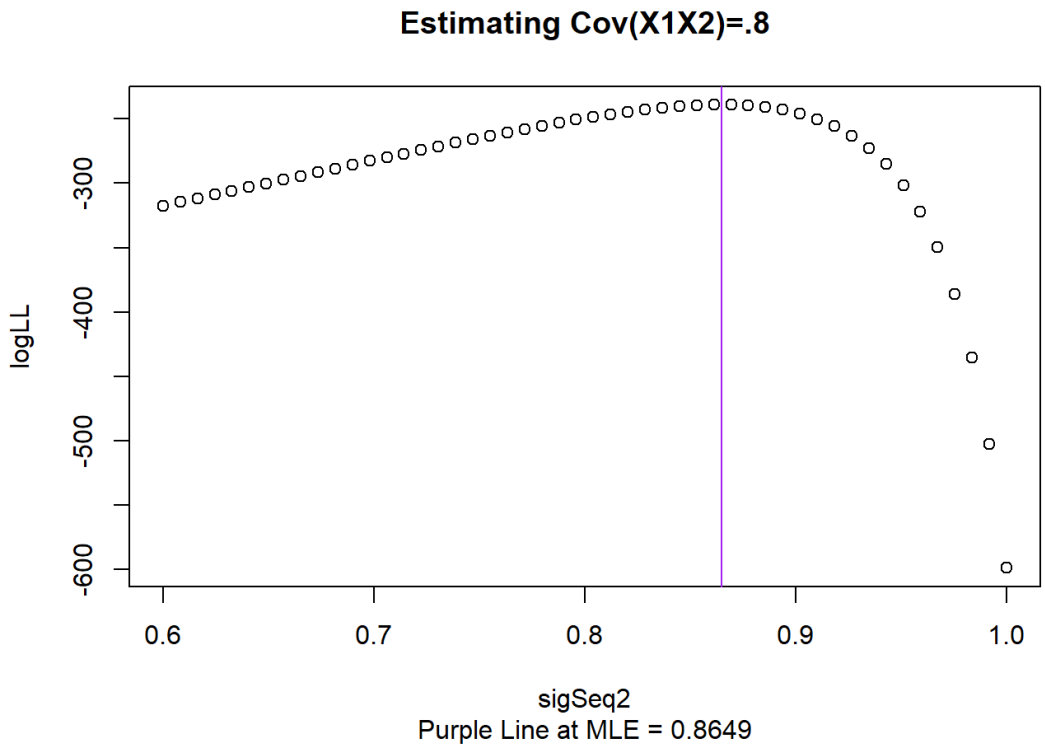

```
sigSeq1=seq(from=.8,to=1.5,length.out=50)
sigSeq2=seq(from=.6,to=1,length.out=50)
logLLVX1=rep(NA,50)
logLLVX2=rep(NA,50)
logLLCovX1X2=rep(NA,50)
Sigma2Ed1=Sigma2Ed2=Sigma2Ed3=sig2Est
for(i in 1:length(sigSeq1))
{
  Sigma2Ed1[1,1]=sigSeq1[i]
  logLLVX1[i]=logLVec(sample3,mu2Est,Sigma2Ed1)
  Sigma2Ed2[2,2]=sigSeq1[i]
  logLLVX2[i]=logLVec(sample3,mu2Est,Sigma2Ed2)
  Sigma2Ed3[1,2]=Sigma2Ed3[2,1]=sigSeq2[i]
  logLLCovX1X2[i]=logLVec(sample3,mu2,Sigma2Ed3)
}

plot(sigSeq1,logLLVX1,main="Estimating V(X1)=1",ylab="logLL",
     sub=paste("Green Line at MLE =",signif(sig2Est[1,1],4)))
abline(v=sig2Est[1,1],col="green")
```




```
plot(sigSeq1,logLLVX2,main="Estimating V(X2)=1",ylab="logLL",
     sub=paste("Orange Line at MLE =",signif(sig2Est[2,2],4)))
abline(v=sig2Est[2,2],col="orange")
```

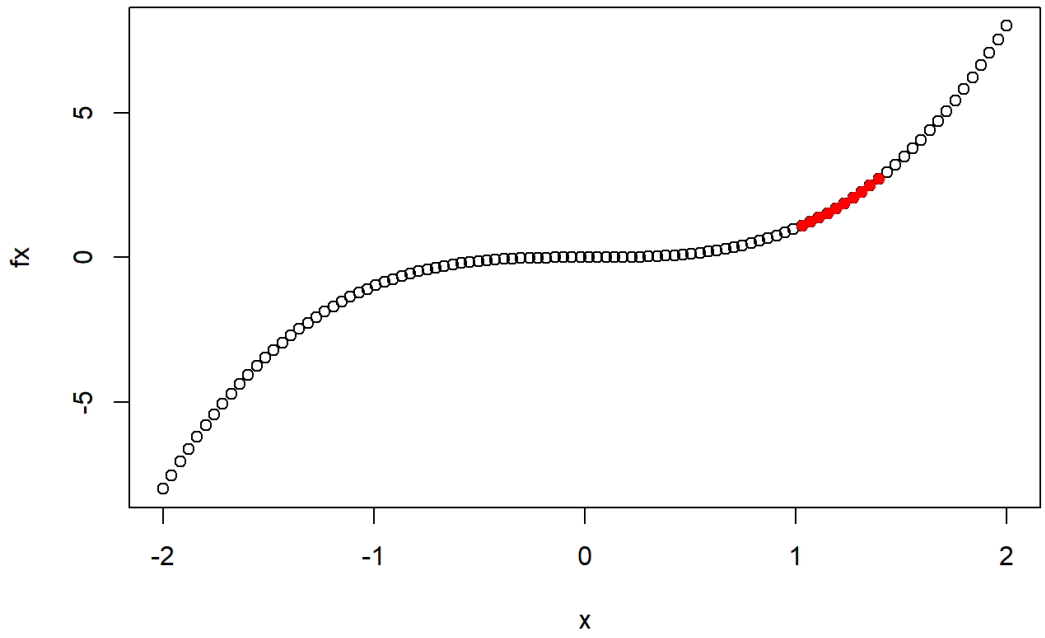



```
plot(sigSeq2,logLLCovX1X2,main="Estimating Cov(X1X2)=.8",ylab="logLL",
     sub=paste("Purple Line at MLE =",signif(sig2Est[1,2],4)))
abline(v=sig2Est[1,2],col="purple")
```




# Problem 3

```
n=100
x = seq(from=-2,to=2,length.out=100)
fx = x^3
plot(x,fx)
ii = 76:85
points(x[ii],fx[ii],col="red",pch=16)
```



```
kernel=function(sigmaf,l,xi,xj)
{
  sigVal=(sigmaf**2)*exp(-1/(2*(l^2))*(xi-xj)^2)
}
muy=matrix(0,nrow=length(ii),ncol=1)
X=matrix(fx[-(76:85)],ncol=1)
#The values for l and sigmaf were obtained from optimizing
#on a grid in a previous piece of code
l=0.5
sigmaf=0.1547692
Sigyy=matrix(NA,nrow=length(ii),ncol=length(ii))
diag(Sigyy)=sigmaf
iiVals=x[ii]
for(i in 1:(ncol(Sigyy)-1))
{
  for(j in (i+1):ncol(Sigyy))
  {
    Sigyy[i,j]=Sigyy[j,i]=kernel(sigmaf,l,iiVals[i],iiVals[j])
  }
}
Sigxx=matrix(NA,nrow=(n-length(ii)),ncol=(n-length(ii)))
diag(Sigxx)=sigmaf
xVals=x[-(76:85)]
for(i in 1:(ncol(Sigxx)-1))
{
  for(j in (i+1):ncol(Sigxx))
  {
    Sigxx[i,j]=Sigxx[j,i]=kernel(sigmaf,l,xVals[i],xVals[j])
  }
}
Sigyx=matrix(NA,nrow=10,ncol=90)
for(i in 1:nrow(Sigyx))
{
  for(j in 1:ncol(Sigyx))
  {
    Sigyx[i,j]=kernel(sigmaf,l,iiVals[i],xVals[j])
  }
}

muYX=muy+Sigyx%*%solve(Sigxx)%*%X
SigYX=Sigyy-Sigyx%*%solve(Sigxx)%*%t(Sigyx)
```

So $Y|X$ is an MVN with mean and covariance as follows

```
muYX
```

```
##           [,1]
##  [1,] 1.120639
##  [2,] 1.265361
##  [3,] 1.420213
##  [4,] 1.584472
##  [5,] 1.757146
##  [6,] 1.936969
##  [7,] 2.122400
##  [8,] 2.311636
##  [9,] 2.502629
## [10,] 2.693118
```

```
SigYX
```

```
##              [,1]        [,2]        [,3]        [,4]        [,5]        [,6]
##  [1,] 0.139587728 0.008871817 0.008872267 0.008773385 0.008577752 0.008290426
##  [2,] 0.008871817 0.139863271 0.009121769 0.009092372 0.008960090 0.008728312
##  [3,] 0.008872267 0.009121769 0.140085391 0.009311998 0.009247405 0.009077474
##  [4,] 0.008773385 0.009092372 0.009311998 0.140241945 0.009431876 0.009328500
##  [5,] 0.008577752 0.008960090 0.009247405 0.009431876 0.140324024 0.009474437
##  [6,] 0.008290426 0.008728312 0.009077474 0.009328500 0.009474437 0.140326759
##  [7,] 0.007918735 0.008402975 0.008806515 0.009118628 0.009330799 0.009437045
##  [8,] 0.007471996 0.007992332 0.008441420 0.008807549 0.009080914 0.009253978
##  [9,] 0.006961166 0.007506631 0.007991392 0.008403122 0.008731042 0.008966330
## [10,] 0.006398436 0.006957730 0.007467598 0.007915469 0.008289956 0.008581258
##              [,7]        [,8]        [,9]       [,10]
##  [1,] 0.007918735 0.007471996 0.006961166 0.006398436
##  [2,] 0.008402975 0.007992332 0.007506631 0.006957730
##  [3,] 0.008806515 0.008441420 0.007991392 0.007467598
##  [4,] 0.009118628 0.008807549 0.008403122 0.007915469
##  [5,] 0.009330799 0.009080914 0.008731042 0.008289956
##  [6,] 0.009437045 0.009253978 0.008966330 0.008581258
##  [7,] 0.140249840 0.009321759 0.009102440 0.008781530
##  [8,] 0.009321759 0.140097717 0.009135359 0.008885169
##  [9,] 0.009102440 0.009135359 0.139879450 0.008889034
## [10,] 0.008781530 0.008885169 0.008889034 0.139608254
```

and the true values for Y are

```
fx[ii]
```

```
##  [1] 1.093692 1.227473 1.371742 1.526894 1.693325 1.871431 2.061608 2.264251
##  [9] 2.479755 2.708518
```

It seems like the average value for $Y|X$ and true value for $Y$ match up pretty well, but it must be noted that the results here are very sensitive to the parameter values chosen for the kernel.