NUEN 647
Uncertainty Quantification for Nuclear Engineering
Homework 3

Due on Saturday, December 10, 2016

Dr. McClarren

Paul Mendoza

Contents

Problem 1

Fit the data in Table 1 to a linear model using

(a) Least squares

(b) Ridge Regression

(c) Lasso Regression

Table 1: Data to fit linear model $y = a + bx_1 + cx_2$

	x_1	x_2	y
1	0.99	0.98	6.42
2	-0.75	-0.76	0.20
3	-0.50	-0.48	0.80
4	-1.08	-1.08	-0.57
5	0.09	0.09	4.75
6	-1.28	-1.27	-1.42
7	-0.79	-0.79	1.07
8	-1.17	-1.17	0.20
9	-0.57	-0.57	1.08
10	-1.62	-1.62	-0.15
11	0.34	0.35	2.90
12	0.51	0.51	3.37
13	-0.91	-0.92	0.05
14	1.85	1.86	5.50
15	-1.12	-1.12	0.17
16	-0.70	-0.70	1.72
17	1.19	1.18	3.97
18	1.24	1.23	6.38
19	-0.52	-0.52	3.29
20	-1.41	-1.41	-1.49

Be sure to do cross-validation for each fit, and for each method present your best estimate of the model.

Did this problem in R, and appended the PDF on the following pages.

Problem 1 continued on next page...

Problem1

Paul Mendoza

December 9, 2016

```
require(magrittr)
require(dplyr)
require(ggplot2)
require(glmnet)
```

Fit the data in Table 1 to a linear model using:

```
Table1
```

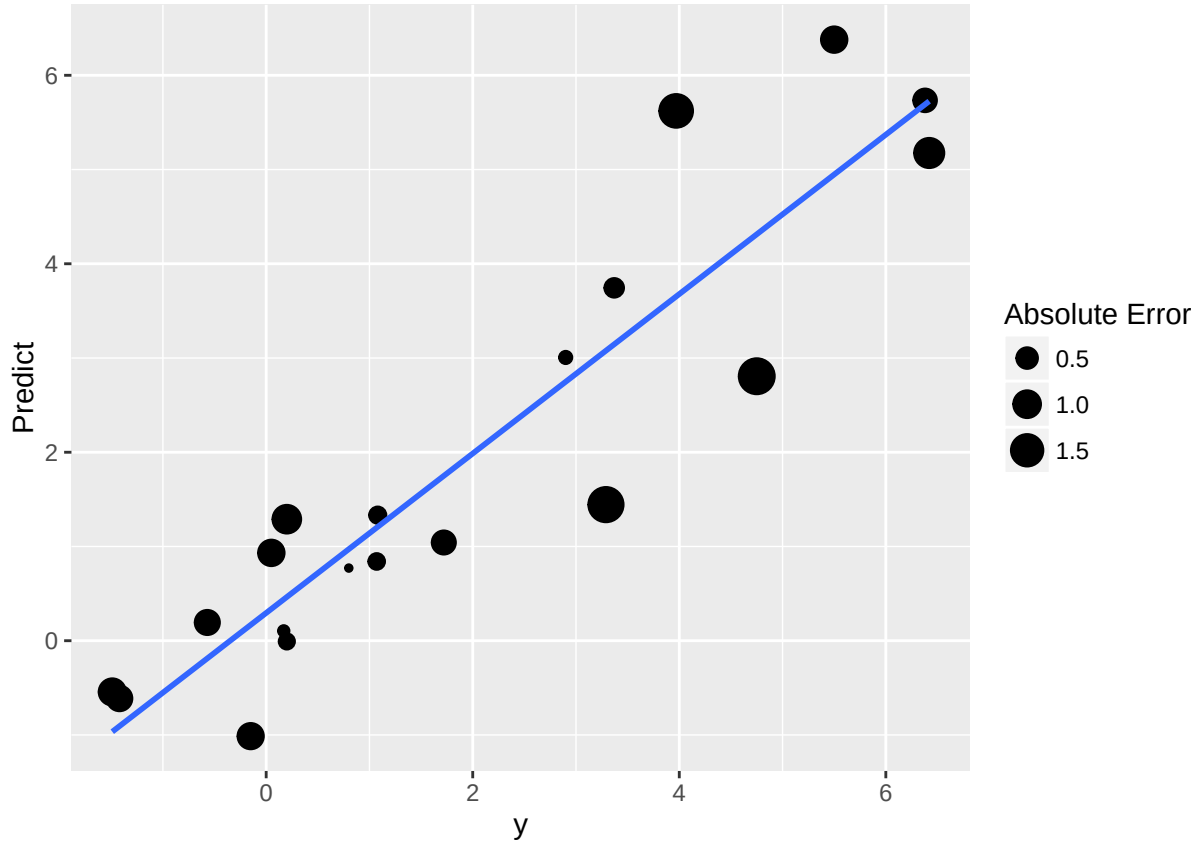
```
##       x1    x2     y
## 1   0.99  0.98  6.42
## 2  -0.75 -0.76  0.20
## 3  -0.50 -0.48  0.80
## 4  -1.08 -1.08 -0.57
## 5   0.09  0.09  4.75
## 6  -1.28 -1.27 -1.42
## 7  -0.79 -0.79  1.07
## 8  -1.17 -1.17  0.20
## 9  -0.57 -0.57  1.08
## 10 -1.62 -1.62 -0.15
## 11  0.34  0.35  2.90
## 12  0.51  0.51  3.37
## 13 -0.91 -0.92  0.05
## 14  1.85  1.86  5.50
## 15 -1.12 -1.12  0.17
## 16 -0.70 -0.70  1.72
## 17  1.19  1.18  3.97
## 18  1.24  1.23  6.38
## 19 -0.52 -0.52  3.29
## 20 -1.41 -1.41 -1.49
```

1. Least Squares

```
LeastFit<-lm(formula = y~x1+x2,data=Table1)
LeastFit
```

```
##
## Call:
## lm(formula = y ~ x1 + x2, data = Table1)
##
## Coefficients:
## (Intercept)           x1           x2
##       2.606       38.133      -35.900
```

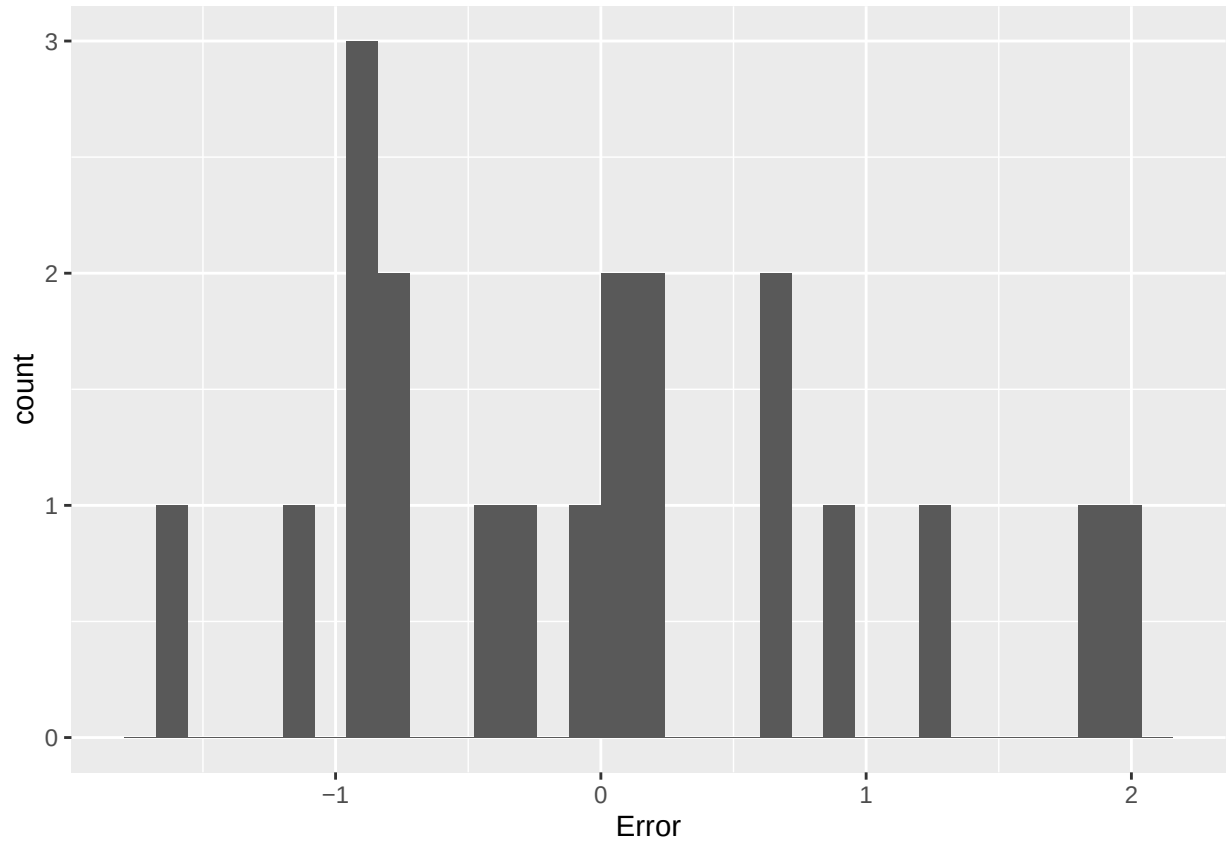
```
plotDF<-Table1
plotDF[,'Type']<-'Train'
plotDF$Predict<-predict(LeastFit,plotDF[,1:2])
plotDF$Error<-plotDF$y-plotDF$Predict
ggplot(plotDF,aes(x=y,y=Predict,size=abs(Error)))+geom_point()+
  scale_size("Absolute Error")+geom_smooth(method="lm",se=F,size=1)
```




```
sqrt(var(data.frame(plotDF %>% filter(Type=="Train") %>% select(Error))))/20
```

```
##            Error
## Error 0.04902635
```

```
ggplot(plotDF,aes(x=Error)) + geom_histogram()
```

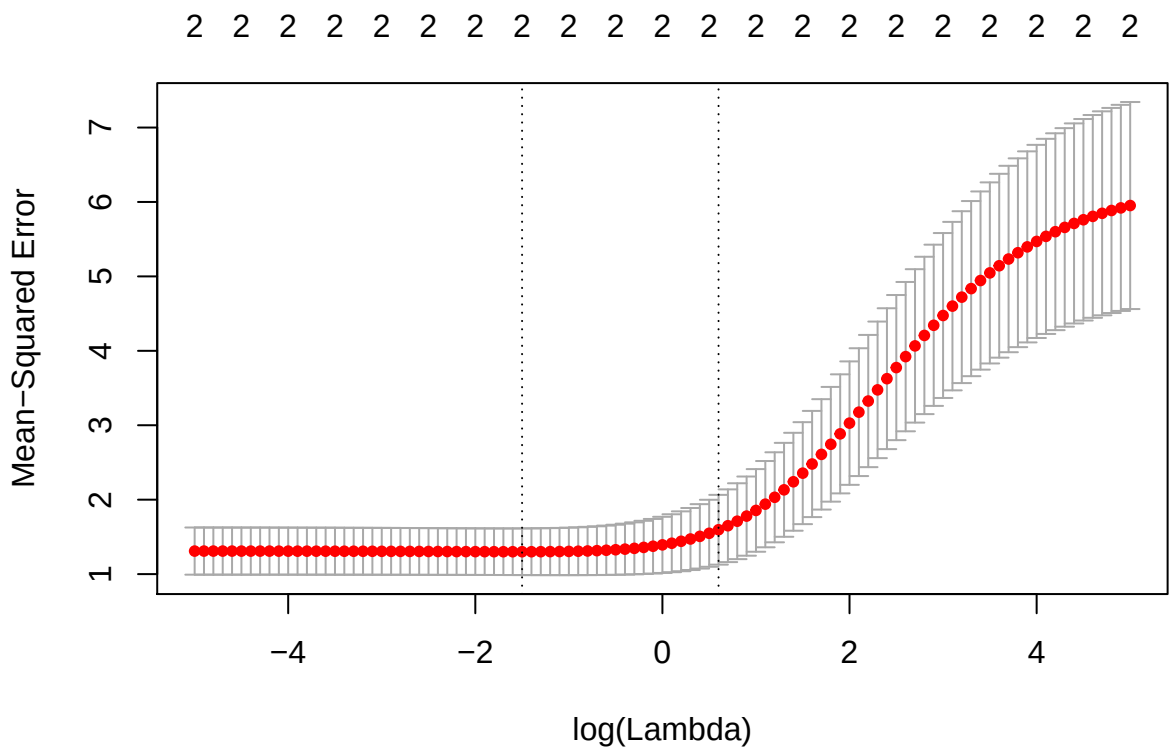



```
sensitivities <- coef(LeastFit)
sensDF <- data.frame(Method = 0, Var = 0, Value=0)
sensDF[1:length(sensitivities),'Method'] <- "Least-Squares"
sensDF[1:length(sensitivities),'Var'] <- names(sensitivities)
sensDF[1:length(sensitivities),'Value'] <- (sensitivities)
rowStart <- length(sensitivities)+1
```

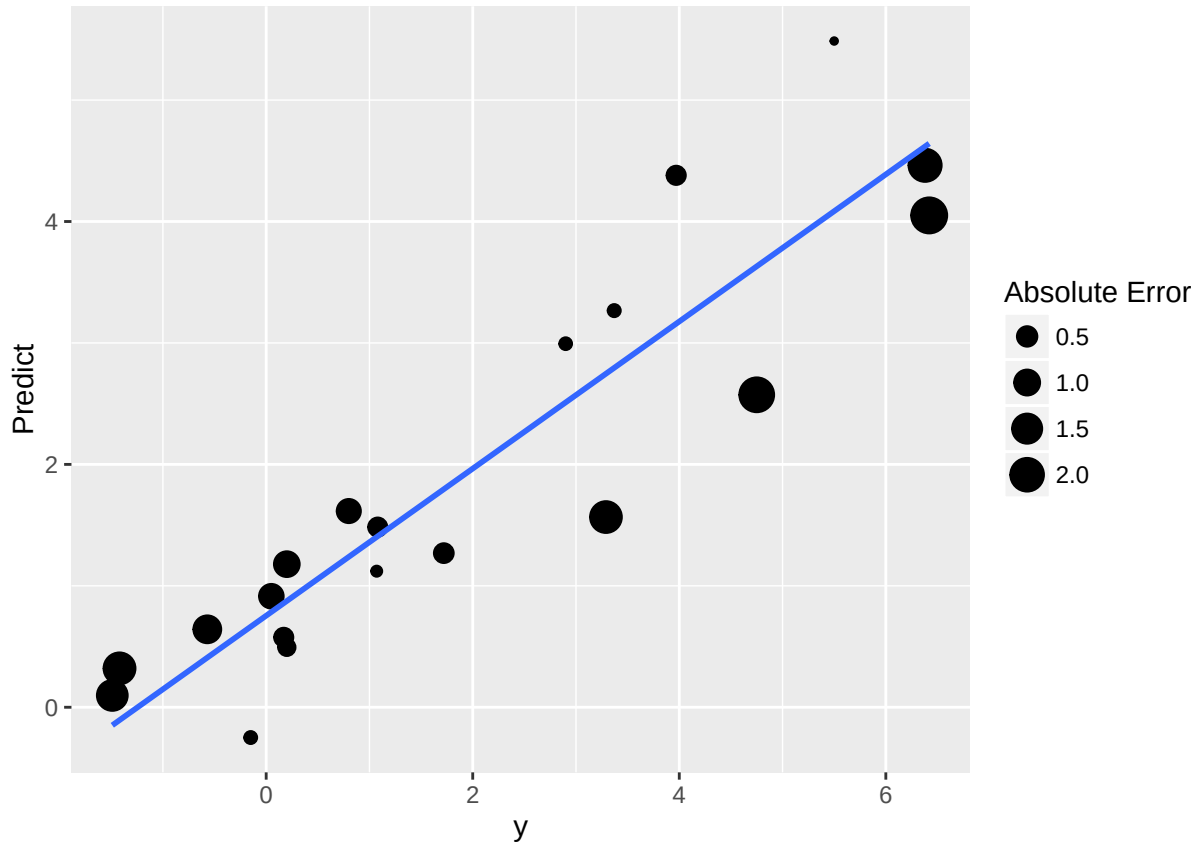
2. Ridge Regression

Ridge regression sets alpha=0, which adds damping to the coefficients

```
crossValid <- cv.glmnet(as.matrix(Table1[,1:2]),
                        as.matrix(Table1$y),alpha = 0,
                        lambda=exp(seq(-5,5,by=0.1)))

plot(crossValid)
```



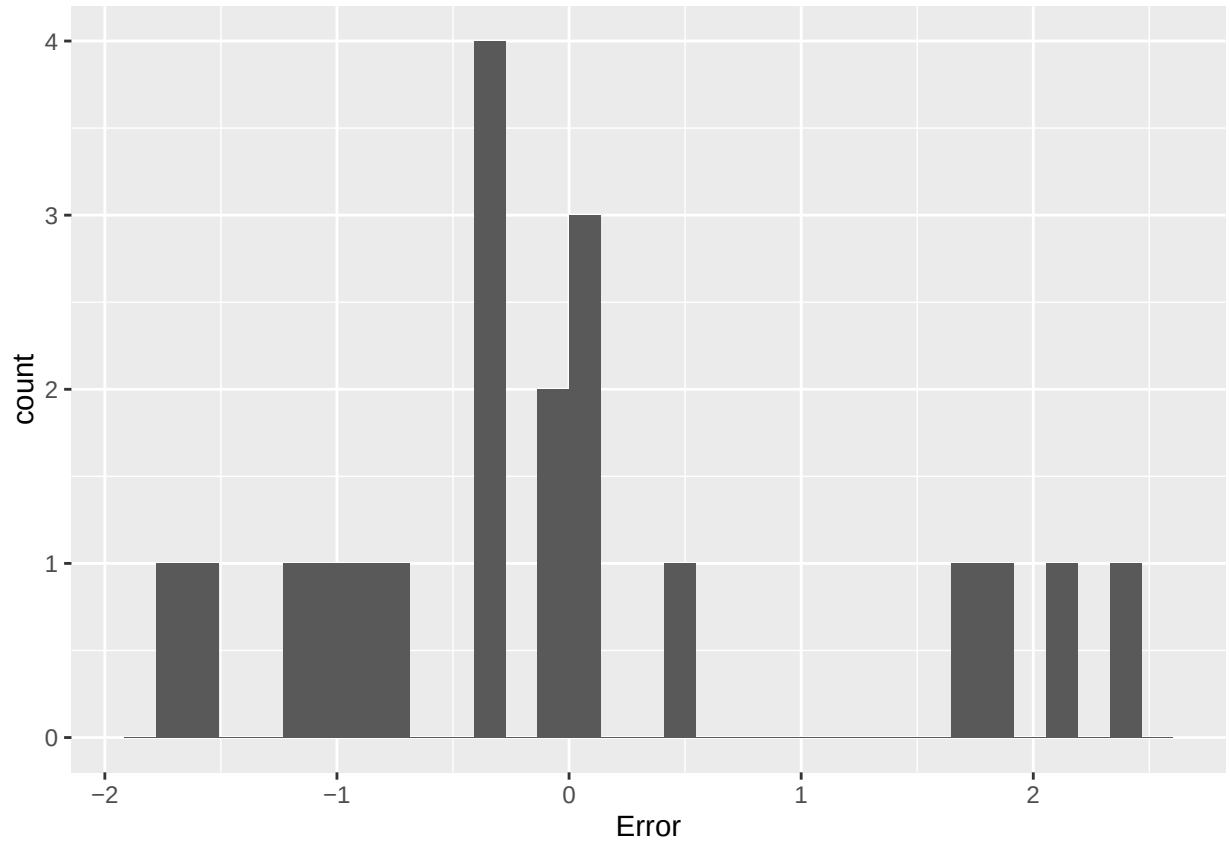

```
lambda <- crossValid$lambda.min
sensitivities <- coef(crossValid)
plotDF <- Table1
plotDF[,'Type'] <- 'Train'
plotDF[,"Predict" ]<- data.frame(Predict=predict(crossValid,as.matrix(plotDF[,1:2]),
                                                 lambda=lambda))
plotDF$Error <- plotDF$y-plotDF$Predict
ggplot(plotDF,aes(x=y,y=Predict,size=abs(Error))) + geom_point() +
scale_size("Absolute Error") + geom_smooth(method="lm",se=F,size=1)
```




```
sqrt(var(data.frame(plotDF %>% filter(Type=="Train") %>% select(Error))))/20
```

```
##             Error
## Error 0.05982956
```

```
ggplot(plotDF,aes(x=Error)) + geom_histogram()
```

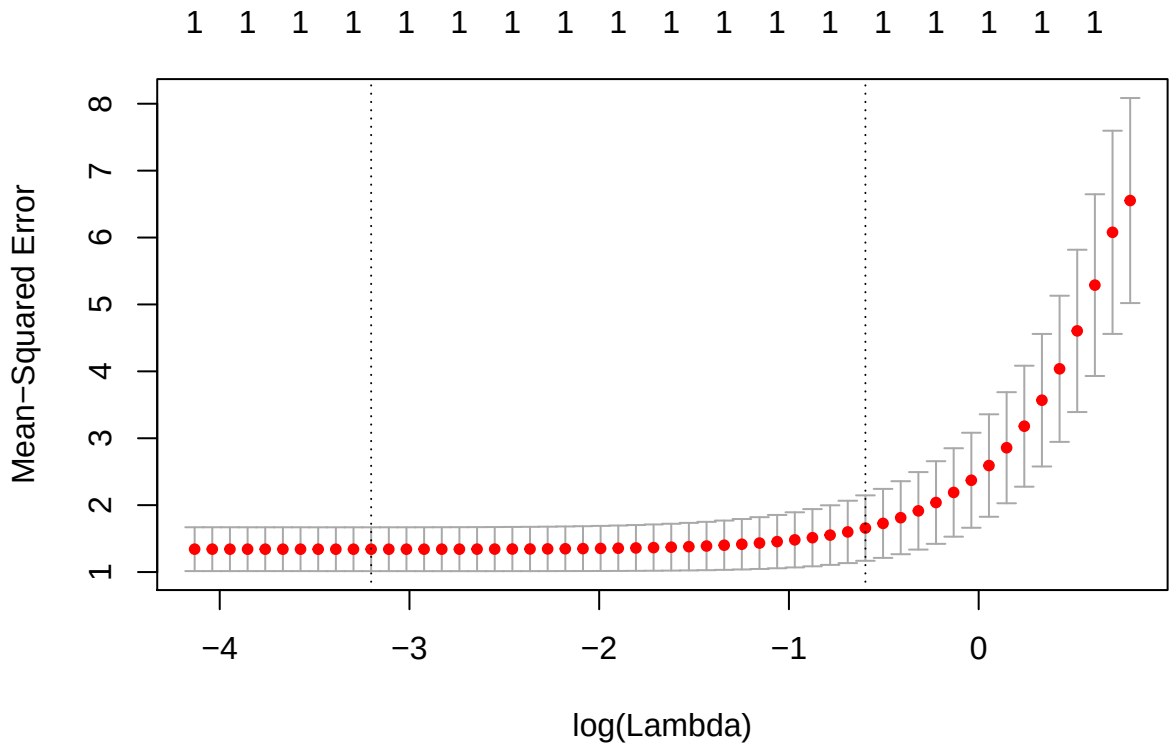



```
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Method'] <- "Ridge"
sensDF[rowStart:(rowStart+length(sensitivities)-1),'Var']<-t(t(rownames(sensitivities)))
sensDF[rowStart:(rowStart +length(sensitivities)-1),'Value']<-as.numeric(sensitivities)
rowStart <- rowStart + length(sensitivities)
```

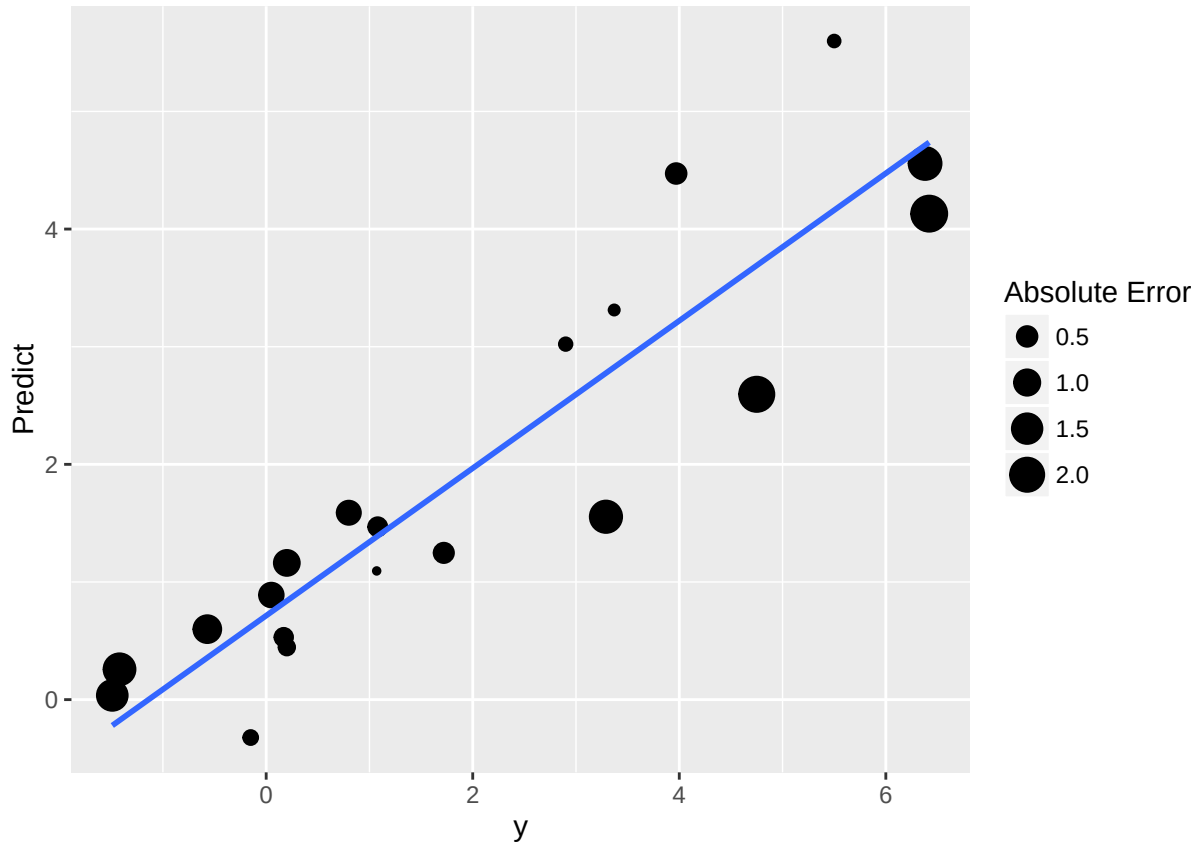
3. Lasso Regression

Lasso regression sets alpha=1

```
crossValid <- cv.glmnet(as.matrix(Table1[,1:2]),as.matrix(Table1$y),alpha = 1)
plot(crossValid)
```



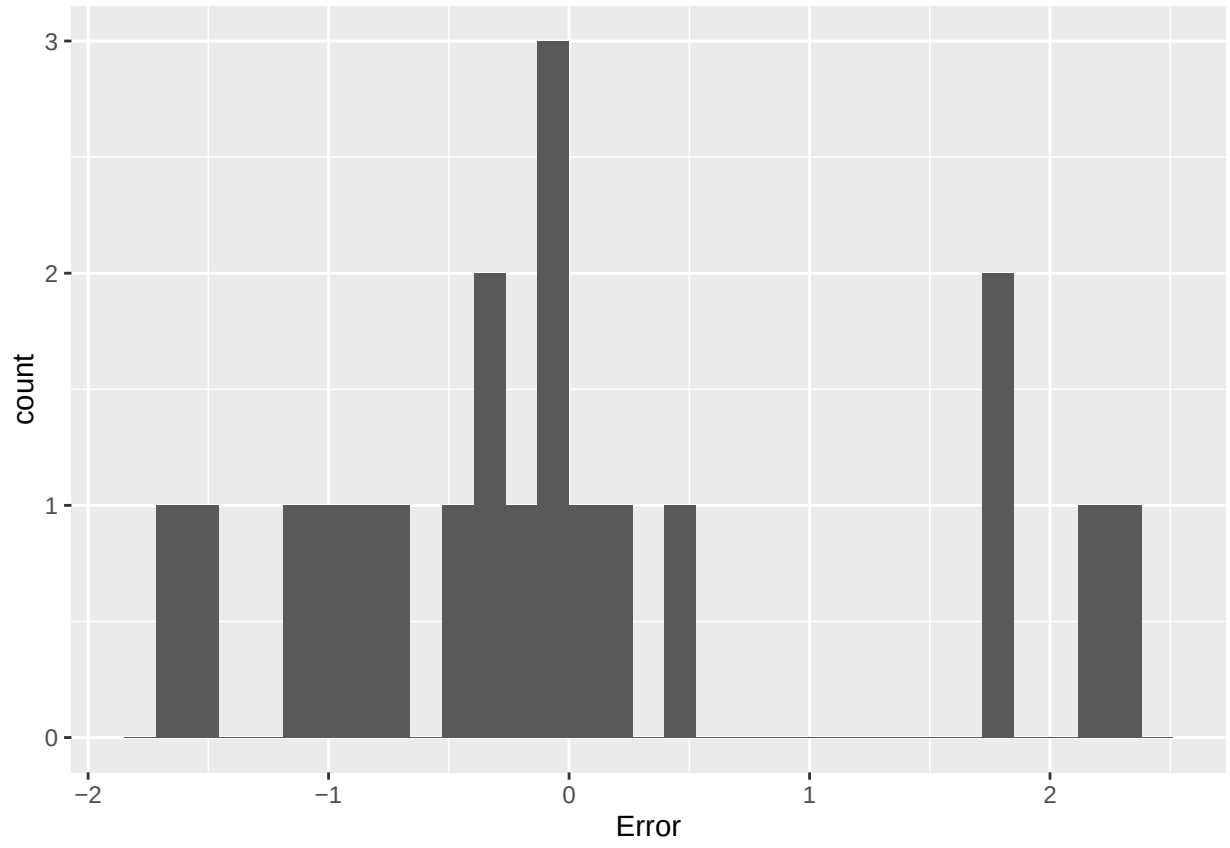

```
lambda <- crossValid$lambda.min
sensitivities <- coef(crossValid)
plotDF <- Table1
plotDF[,'Type'] <- 'Train'
plotDF[,"Predict" ]<- data.frame(Predict=predict(crossValid,as.matrix(Table1[,1:2]),
                                                 lambda=lambda))

plotDF$Error <- plotDF$y-plotDF$Predict
ggplot(plotDF,aes(x=y,y=Predict,size=abs(Error))) + geom_point() +
scale_size("Absolute Error") + geom_smooth(method="lm",se=F,size=1)
```




```
sqrt(var(data.frame(plotDF %>% filter(Type=="Train") %>% select(Error))))/20
```

```
##             Error
## Error 0.05832321
```

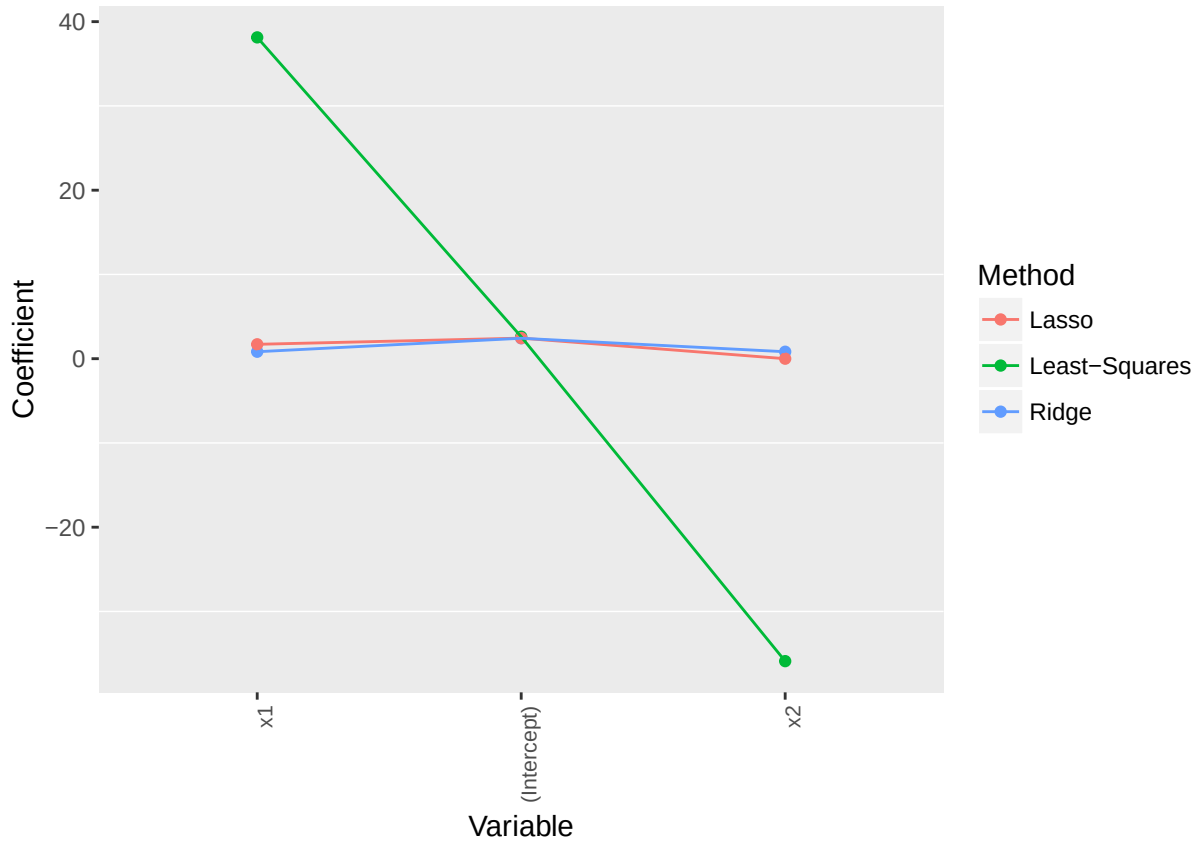
```
ggplot(plotDF,aes(x=Error)) + geom_histogram()
```




```
sensDF[rowStart:(rowStart + length(sensitivities)-1),'Method'] <- "Lasso"
sensDF[rowStart:(rowStart+length(sensitivities)-1),'Var']<-t(t(rownames(sensitivities)))
sensDF[rowStart:(rowStart +length(sensitivities)-1),'Value']<-as.numeric(sensitivities)
rowStart <- rowStart + length(sensitivities)
```

Compare Methods

```
ggplot(sensDF,aes(x=reorder(Var,-Value),y=Value,color=Method,group=Method)) +
  geom_point() + geom_line() +
  theme(panel.grid.major = element_blank(),
        axis.text.x = element_text(angle = 90, hjust = 1, size=8))+
  scale_x_discrete("Variable") + scale_y_continuous("Coefficient")
```



The Lasso and Ridge are both more bounded in their coefficents.

Problem 2

Derive the adjoint operator for the equation

$$-\bigtriangledown^2 \phi(x,y,z) + \frac{1}{L^2}\phi(x,y,z) = \frac{Q}{D}$$

$$\phi(0,y,z) = \phi(x,0,z) = \phi(x,y,0) = \phi(X,y,z) = \phi(x,Y,z) = \phi(x,y,Z) = C$$

Compute the sensitivity to the QOI:

$$QoI = \int_0^X dx \int_0^Y dy \int_0^Z dz \ \frac{D}{L^2}\phi(x,y,z)$$

for X,Y,Z,L,D and Q.

Derive the adjoint operator

See Ch 6 for help more background. Define the operator $\mathcal{L}$ as

$$\mathcal{L} = -D\bigtriangledown^2 + \frac{D}{L^2}$$

and the adjoint $\mathcal{L}^\dagger$ as

$$\mathcal{L}^\dagger = -D\bigtriangledown^2 + \frac{D}{L^2}$$

$$\phi^\dagger(0,y,z) = \phi^\dagger(x,0,z) = \phi^\dagger(x,y,0) = \phi^\dagger(X,y,z) = \phi^\dagger(x,Y,z) = \phi^\dagger(x,y,Z) = C$$

Setting:

$$\left|\frac{\delta\phi^\dagger}{\delta x}\right|_{x=0} = \left|\frac{\delta\phi}{\delta x}\right|_{x=0}$$

and

$$\left|\frac{\delta\phi^\dagger}{\delta x}\right|_{x=X} = \left|\frac{\delta\phi}{\delta x}\right|_{x=X}$$

and similar for the other two dimentions. Also define the inner product as:

$$(u,v) = \int_0^X dx \int_0^Y dy \int_0^Z dz \ uv$$

Proof, in order to prove that this is an adjoint operator for the above equation, it needs to be shown that $(\mathcal{L}\phi, \phi^\dagger) = (\phi, \mathcal{L}^\dagger\phi^\dagger)$.

Equivalent to (all terms have a D in them and cancel):

$$\int_0^X dx \int_0^Y dy \int_0^Z dz \left(-\phi^\dagger \bigtriangledown^2 \phi + \phi^\dagger \frac{\phi}{L^2}\right) = \int_0^X dx \int_0^Y dy \int_0^Z dz \left(-\phi \bigtriangledown^2 \phi^\dagger + \phi\frac{\phi^\dagger}{L^2}\right) \quad (1)$$

The terms

$$\int_0^X dx \int_0^Y dy \int_0^Z dz \left(\phi^\dagger \frac{\phi}{L^2}\right) = \int_0^X dx \int_0^Y dy \int_0^Z dz \left(\phi\frac{\phi^\dagger}{L^2}\right)$$

are equal. For the other term with, $\bigtriangledown^2$, we can expand to (removing the negative sign for simplicity):

$$\int_0^X dx \int_0^Y dy \int_0^Z dz \left(\phi^\dagger \left[\frac{\delta^2\phi}{\delta x^2} + \frac{\delta^2\phi}{\delta y^2} + \frac{\delta^2\phi}{\delta z^2}\right]\right)$$

Problem 2 continued on next page...

Focusing on the x terms, noting that y and z will have the same derivation. Integration by parts, with $u = \phi^\dagger$, $du = \frac{\delta\phi^\dagger}{\delta x}dx$, and $v = \frac{\delta\phi}{dx}$, $dv = \frac{\delta^2\phi}{\delta x^2}dx$ yields.

$$\int_0^Y dy \int_0^Z dz \left(\int_0^X dx\ \phi^\dagger \frac{\delta^2\phi}{\delta x^2} \right) = \int_0^Y dy \int_0^Z dz \left(\left| \phi^\dagger \frac{\delta\phi}{\delta x} \right|_{x=0}^{x=X} - \int_0^X dx\ \frac{\delta\phi}{\delta x}\frac{\delta\phi^\dagger}{\delta x}dx \right)$$

Performing another integration by parts, with $u = \frac{\delta\phi^\dagger}{\delta x}$, $du = \frac{\delta^2\phi^d ag}{\delta x}dx$ and, $v = \phi$, $dv = \frac{\delta\phi}{\delta x}dx$.

$$= \int_0^Y dy \int_0^Z dz \left(\left| \phi^\dagger \frac{\delta\phi}{\delta x} \right|_{x=0}^{x=X} - \left| \phi \frac{\delta\phi^\dagger}{\delta x} \right|_{x=0}^{x=X} + \int_0^X dx\ \phi \frac{\delta^2\phi^\dagger}{\delta x^2}dx \right)$$

At the boundaries, both ϕ and $\phi^\dagger$ are a constant, and the derivatives of both at the boundaries are equal, and therefore those terms cancel, leaving

$$\int_0^Y dy \int_0^Z dz \left(\int_0^X dx\ \phi \frac{\delta^2\phi^\dagger}{\delta x^2}dx \right)$$

which is equal to the x component of the $\bigtriangledown^2$ term of the RHS of equation 1 above.

Compute the sensitivity to the QoI:

The QoI has been defined as,

$$QoI = \int_0^X dx \int_0^Y dy \int_0^Z dz\ \frac{D}{L^2}\phi(x,y,z) = (\phi, p),$$

with $p = \frac{D}{L^2}$. Recall the original system being,

$$\mathcal{L}\phi = Q.$$

If we define an adjoint system as,

$$\mathcal{L}^\dagger\phi^\dagger = p,$$

then the QoI can be represented as,

$$QoI = (\phi, p) = (Q, \phi^\dagger)$$

This is because,

$$(\mathcal{L}\phi, \phi^\dagger) = (\phi, \mathcal{L}^\dagger\phi^\dagger)$$
$$(Q, \phi^\dagger) = (\phi, p)$$

The first line was proved in the first part of the problem, and the second line substitutes q for $\mathcal{L}\phi$ on the LHS of the equation and p for $\mathcal{L}^\dagger\phi^\dagger$ on the RHS.
If the solution for $\phi^\dagger$ were known, then this problem would be a lot easier. Lets see if we can find it.

$$\mathcal{L}^\dagger\phi^\dagger = p$$
$$-D \bigtriangledown^2 \phi^\dagger + \frac{D}{L^2}\phi^\dagger = \frac{D}{L^2}$$
$$\phi^\dagger = 1 + L^2 \bigtriangledown^2 \phi^\dagger$$

Where the solution is $\phi^\dagger = 1 + exp(\frac{x}{L}) + exp(\frac{y}{L}) + exp(\frac{z}{L})$

Problem 2 continued on next page...

Now the QoI is,

$$\begin{aligned} QoI &= (Q, \phi^\dagger) \\ &= \int_0^X dx \int_0^Y dy \int_0^Z dz\ Q\phi^\dagger \\ &= \int_0^X dx \int_0^Y dy \int_0^Z dz\ Q\left(1 + e^{\frac{x}{L}} + e^{\frac{y}{L}} + e^{\frac{z}{L}}\right) \\ &= QXYZ + QYZL(e^{X/L} - 1) + QXZL(e^{Y/L} - 1) + QXYL(e^{Z/L} - 1) \end{aligned}$$

The sensitivity to the QoI will be determined with a simple derivative of the QoI with respect to particular variables.

$$\frac{\delta QoI}{\delta X} = \boxed{QYZ + QYZe^{X/L} + QZL(e^{Y/L} - 1) + QYL(e^{Z/L} - 1)}$$

$$\frac{\delta QoI}{\delta Y} = \boxed{QXZ + QZL(e^{X/L} - 1) + QXZe^{Y/L} + QXL(e^{Z/L} - 1)}$$

$$\frac{\delta QoI}{\delta Z} = \boxed{QXY + QYL(e^{X/L} - 1) + QXL(e^{Y/L} - 1) + QXYe^{Z/L}}$$

$$\frac{\delta QoI}{\delta L} = \boxed{Q(YZe^{X/L}(1 - e^{-X/L} - \frac{X}{L}) + XZe^{Y/L}(1 - e^{-Y/L} - \frac{Y}{L}) + XYe^{Z/L}(1 - e^{-Z/L} - \frac{Z}{L}))}$$

$$\frac{\delta QoI}{\delta D} = \boxed{0}$$

$$\frac{\delta QoI}{\delta Q} = \boxed{XYZ + YZL(e^{X/L} - 1) + XZL(e^{Y/L} - 1) + XYL(e^{Z/L} - 1)}$$

Problem 3

For the random variable X $\sim$ N(0,1) draw fifty samples and generate histograms using the following sampling techniques

(a) Simple random sampling

(b) Stratifid sampling

(c) A van der Corput sequence of base 2

(d) A van der Corput sequence of base 3

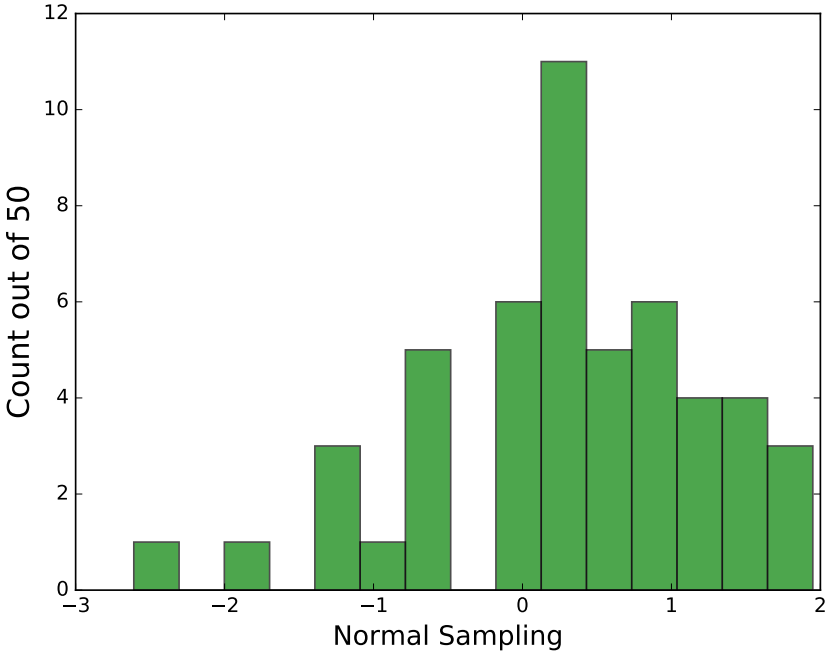
Simple random sampling samples U(0,1) and plugs this value into the inverse CDF. Stratified sampling separates U(0,1) into equal bins and samples "Randomly" in each bin. Van der Corput sequences divides an interval into a a number of equal subintervals.

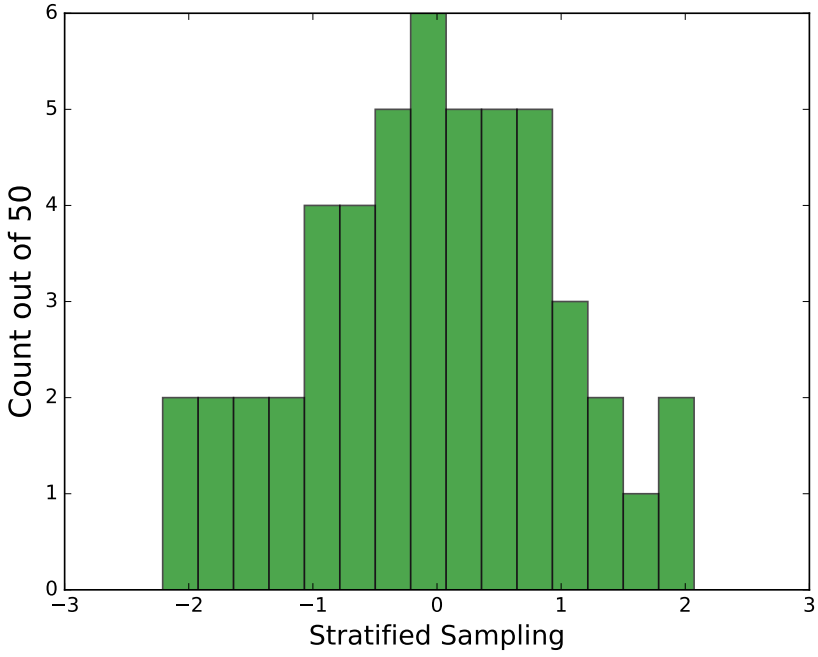
For example, the ordinary van der Corput sequence in base 3 is given by 1/3, 2/3, 1/9, 4/9, 7/9, 2/9, 5/9, 8/9, 1/27.

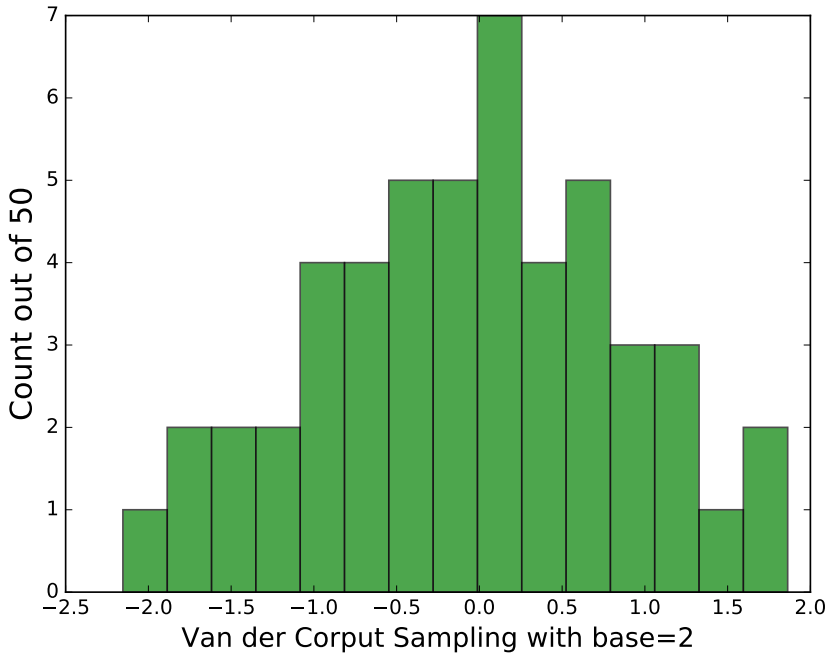
Listing 1: Script for Problem

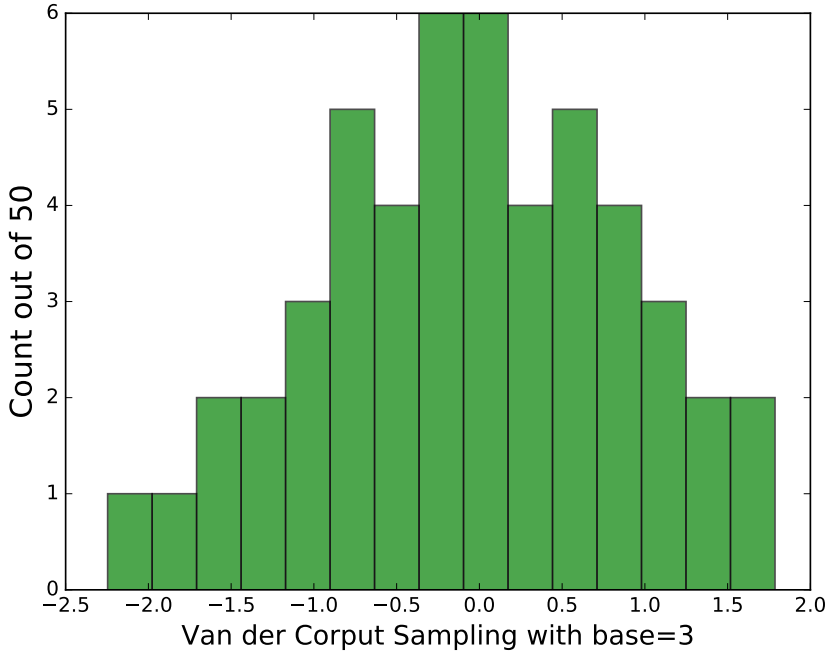
```
#!/usr/bin/env python3

#################################################################
#################### Import packages ############################
#################################################################

import numpy as np
import matplotlib.pyplot as plt
import time
start_time = time.time()
from scipy.stats import norm

#################################################################
####################### Functions ###############################
#################################################################

#Van der Corput sequence function found online
def vdc(n, base=2):
    vdc, denom = 0,1
    while n:
        denom *= base
        n, remainder = divmod(n, base)
        vdc += remainder / denom
    return vdc

#################################################################
#################### Calculations ###############################
#################################################################

#Make sure Nstrata <= N

N=50  #samples
```

```
Nbins=15  #hist plot
Nstrata=49
filename="V2Norm.pdf"
#Stratified or Normal or Van der Corput
Xlabel="Van der Corput Sampling with base=2"
RandomNumbers=[]
vanBase=2;van=True

#Sampling for normal and stratified
if not van:
    Nloop=int(N/Nstrata)*Nstrata
    for i in range(0,int(N/Nstrata)):
        for j in range(0,Nstrata):
            RandomNumbers.append(np.random.uniform(low=j/Nstrata,
                                 high=(j+1)/Nstrata,size=1))
    #If N/Nstrata doesn't divide evenly
    if Nloop<N:
        for j in range(0,N-Nloop):
            RandomNumbers.append(np.random.uniform(low=j/Nstrata,
                                 high=(j+1)/Nstrata,size=1))
#Sampling for van
if van:
    for i in range(0,N):
        RandomNumbers.append(vdc(i+1,vanBase))

#Sample the inverse of the CDF of the standard normal
#distribution
Samples=norm.ppf(RandomNumbers)

#Generate histogram
fig=plt.figure()
ax=fig.add_subplot(111)
ax.set_xlabel(Xlabel,fontsize=16)
ax.set_ylabel('Count out of '+str(N),fontsize=18)
ax.hist(Samples,Nbins,color='green',alpha=0.7,edgecolor='black')
#ax.set_xlim(-500,500)
plt.savefig(filename)

###################### Time To execute ################

print("--- %s seconds ---" % (time.time() - start_time))
```


Count out of 50
Normal Sampling
0
2
4
6
8
10
12
−3
−2
−1
0
1
2


Count out of 50
Stratified Sampling
0
1
2
3
4
5
6
−3
−2
−1
0
1
2
3


Count out of 50
Van der Corput Sampling with base=2


Count out of 50
Van der Corput Sampling with base=3

Problem 4

Consider the Rosenbrock function $f(x,y) = (1-x)^2 + 100(y-x^2)^2$. Assume that $x = 2t - 1$, where T $\sim$ $B(3,2)$ and $y = 2s - 1$, where S $\sim$ $B(1.1, 2)$. Estimate the probability that $f(x,y)$ is less than 10 using:

(a) a first-order second-moment reliability method

(b) Latin hypercube sampling using 50 points

(c) A Halton sequence using 50 points

Compare this with the probability you calculate using 10^5 random samples. (*Hint: Matlab has a built-in function for sampling beta R.V.'s "betard"*).

A first-order second-moment reliability method

Will use a gaussian approximation as shown in section 7.3 in the course notes (FORM).

In this method the performance function ($Z(x,y)$) is a function of the random variables x and y, and is defined such that the failure surface is the location where $Z = 0$ (top of page 119 Ch 7) such that $Z < 0$ represents failure and $Z > 0$ represents success. For the case above, this would mean that our performance function is:

$$\begin{aligned} Z &= 10 - f(x,y) \\ &= 10 - (1-x)^2 - 100(y-x^2)^2 \end{aligned}$$

Next, the probability of failure is defined as:

$$p_{fail} = 1 - \Phi\left(\frac{\mu_Z}{\sigma_Z}\right)$$

Where Φ is the CDF for a standard normal, μ_Z and σ_Z are the mean and standard deviation of Z. As a reminder, if Z were a standard normal, defined such that any part of Z that is less than 0 is failure. Then because $\mu_Z = 0$ there would be a 50% chance of failure. If Z were a non standard normal, then the distance from zero would be normalized (by dividing by σ_Z) to units of σ, and as μ_Z increases, the chance of failure would continue to decrease, which make sense, as the mean of the performance function moves further and further away from the failure point ($Z = 0$), the probability of failure decreases.

I am trying to spell this out for myself, because I was really confused about this. There are two things I would like to point out to future self. First, Z may not be normal, which is why this method is only exact if Z is normal, otherwise its an approximation. Second, if μ_Z were less than 0, then the equation for the probability of failure should (I think) change to

$$p_{fail} = 0.5 + \Phi\left(\frac{|\mu_Z|}{\sigma_Z}\right)$$

Also third, I am not sure if Z has to be a typical PDF or CDF, will let you know after I do some of the math McClarren gave.

Problem 4 continued on next page...

The mean for Z was defined as

$$\mu_Z \approx g(\mu_{x,y})$$

where g is the function we defined as Z (first part of the taylor expansion of Z) evaluated at the mean values of x and y. The standard deviation of Z was defined as the the second part of the taylor expansion of Z (without covariances because we are going to assume that all random variables are independent), namely

$$\sigma_Z^2 = \left(\left| \frac{\delta g}{\delta x} \right|_{\mu_x} \sigma_x \right)^2 + \left(\left| \frac{\delta g}{\delta y} \right|_{\mu_y} \sigma_y \right)^2$$

For what I defined as Z above,

$$\frac{\delta g}{\delta x} = -2(x-1) - 400x(x^2 - y)$$
$$\frac{\delta g}{\delta y} = -200(y - x^2)$$

Also for a beta R.V

$$\mu = \frac{a}{a+b}$$
$$\sigma^2 = \frac{ab}{(a+b)^2(a+b+1)}$$

Calculations

For the random variable t used in $x = 2t - 1$

mean

$$\mu_t = \frac{3}{3+2} = \mathbf{0.6}$$
$$\mu_x = 2 * \mathbf{0.6} - 1 = \boxed{0.2}$$

standard deviation

$$\sigma_t^2 = \frac{3 \cdot 2}{(3+2)^2(3+2+1)} = \mathbf{0.04}$$
$$\sigma_x^2 = \left| \frac{\delta x}{\delta t} \right|_{\mu_T}^2 \sigma_t^2$$
$$= 2^2 \cdot \mathbf{0.04} = \boxed{0.16}$$

Problem 4 continued on next page...

For the random variable s used in $y = 2s - 1$

mean

$$\mu_s = \frac{1.1}{1.1+2} = \mathbf{0.354839}$$
$$\mu_y = 2 * \mathbf{0.354839} - 1 = \boxed{-0.290323}$$

standard deviation

$$\sigma_s^2 = \frac{1.1 \cdot 2}{(1.1+2)^2(1.1+2+1)} = \mathbf{0.055836}$$
$$\sigma_y^2 = \left|\frac{\delta y}{\delta s}\right|_{\mu_s}^2 \sigma_S s^2$$
$$= 2^2 \cdot \mathbf{0.055836} = \boxed{0.223345}$$

For the partial derivative terms

$$\left|\frac{\delta g}{\delta x}\right|_{\mu_X} = -2(x-1) - 400x(x^2 - y) = -2(0.2-1) - 400 \cdot 2(0.2^2 - (-0.290323)) = \boxed{-24.8258}$$

$$\left|\frac{\delta g}{\delta y}\right|_{\mu_y} = -200(y - x^2) = -200(-0.290323 - 0.2^2) = \boxed{66.0646}$$

For Z and the probability of failure

$$\mu_Z = 10 - (1-0.2)^2 + 100(-0.290323 - 0.2^2)^2 = \boxed{20.2713}$$

$$\sigma_Z^2 = 24.8^2 \cdot 0.16 + 66.0646^2 \cdot 0.223345 = 1073.41$$
$$\sigma_Z = \boxed{32.7629}$$

$$p_{fail} = 1 - \Phi\left(\frac{20.27}{32.7629}\right)$$
$$= \boxed{\mathbf{0.268}}$$

Listing 2: Script for Hypercube and Halton

```
#!/usr/bin/env python3

#############################################################
#################### Import packages ########################
#############################################################

import time
start_time = time.time()
import Functions as fun
import lhsmdu
```

Problem 4 continued on next page...

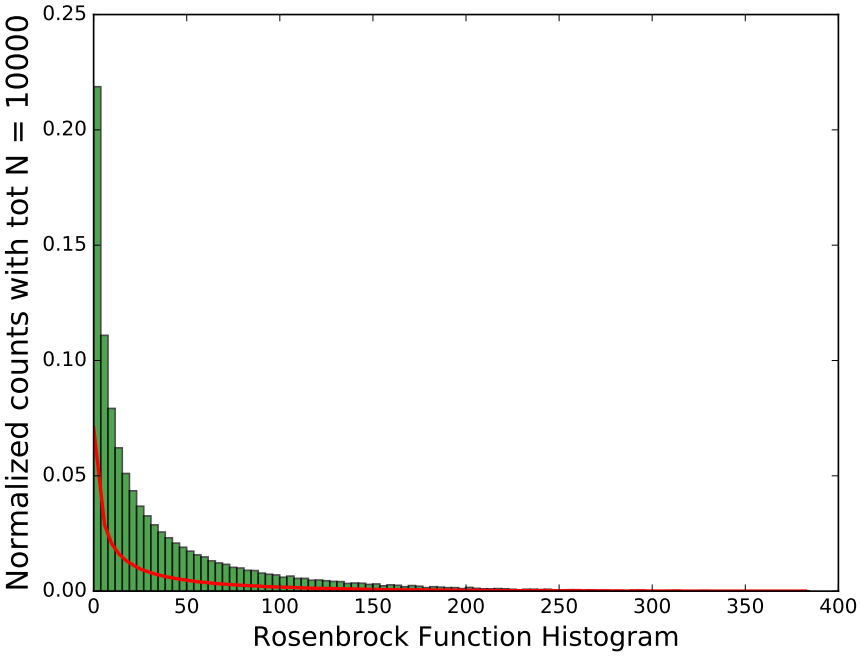
```
from random import shuffle

################################################################
###################### Calculations ############################
################################################################

N=100  #Samples
Nbins=30 #Hist Plot
Nstrata=10
filename="HaltonStrat.pdf"

Xlabel="Halton Sampling"
#RandomNumbersX=fun.np.random.uniform(0,1,N)
#RandomNumbersY=fun.np.random.uniform(0,1,N)
#RandomNumbersX=fun.Rstrat(N,Nstrata)  #strat sampling
#RandomNumbersY=fun.Rstrat(N,Nstrata)
#l=lhsmdu.sample(2,N)  #Hyper cube sampling
#RandomNumbersX=l[0].A1
#RandomNumbersY=l[1].A1
RandomNumbersX=fun.Rvdc(N,2)     #Halton sequence
RandomNumbersY=fun.Rvdc(N,3)     #shuffled the list (tried notto)
#shuffle(RandomNumbersX)
#shuffle(RandomNumbersY)

Samplest=fun.beta.ppf(RandomNumbersX,3,2)
Sampless=fun.beta.ppf(RandomNumbersY,1.1,2)

X=fun.X(Samplest)
Y=fun.Y(Sampless)

f=fun.Rosen(X,Y)

#Plot the data for Rosenbrok, and plot fitted PDF
Xlabel="Rosenbrock Function Histogram"
(n,bins,ax,fig)=fun.HIST(Xlabel,f,Nbins,N)
(ax,fig)=fun.HISTDataToPDF(n,bins,ax,fig)
fun.plt.savefig(filename)

#Find the probability of f being less than 10
PGreater=sum(i<10 for i in f)/N

print("The probability of being less than 10 is: "+str(PGreater))


###################### Time To execute ################

print("--- %s seconds ---" % (time.time() - start_time))
```

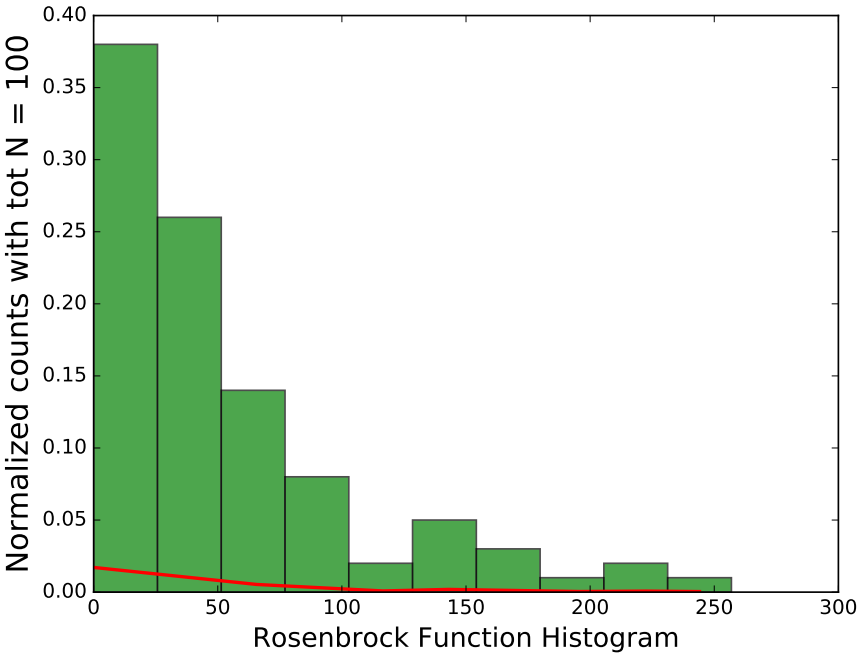
My PDF integrates to 1.01, which I'm okay with. The table below summarizes my results.

Table 2: Different Sampling Techniques

Method	p_{fail} 50 points	p_{fail} 100 points
Stratified	0.30	0.28
First Order	0.268	
Latin Hyper	0.36	0.26
Halton	0.24	0.25
Normal (10^5)	0.26017	

It should be noted that I shuffled the stratified samples, because otherwise there would be some correlation between the numbers. The first order listing in the table is not from the code, but the calculation in the first part of the problem above (did not use 50 points). I realized I could have used a more complicated first-order second moment reliability method, but this is A reliability method.




Normalized counts with tot N = 100
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
50
100
150
200
250
300
Rosenbrock Function Histogram

Problem 5

Consider the exponential integral function, $E_n(x)$,

$$E_n(x) = \int_1^\infty \frac{e^{-xt}}{t^n} dt$$

This function is involved in the solution to many pure-absorbing transport problems. Use this function to solve the transport problem,

$$\mu \frac{\delta \psi}{\delta x} + \sigma \psi = 0,$$

$$\psi(0, \mu > 0) = 1, \psi(10, \mu < 0) = 0,$$

for the scaler flux $\phi(x) = \int_{-1}^{1} \psi(x, \mu) d\mu$. Assume that $\sigma \sim \ GAM(10, 0.1)$. Use a PCE expansion to estimate the distribution, mean, and variance of $\phi(x)$ at $x = 1, 1.5, 3, 5$. Also, plot the mean value of ϕ as a function of x.

Using the integrating factor approach we proceed as follows:

$$\frac{\delta \psi}{\delta x} + \frac{\sigma \psi}{\mu} = 0$$
$$\frac{\delta}{\delta x}\left[e^{\frac{\sigma x}{\mu}} \psi\right] = 0$$

Integrating from 0 to x for $\mu > 0$, we obtain (yes I am copying some old notes)

$$\begin{aligned} \psi(x, \mu > 0) e^{\frac{\sigma x}{\mu > 0}} - \psi(0, \mu > 0) e^{\frac{\sigma \cdot 0}{\mu > 0}} &= 0 \\ \psi(x, \mu > 0) e^{\frac{\sigma x}{\mu > 0}} - 1 &= 0 \\ \psi(x, \mu > 0) e^{\frac{\sigma x}{\mu > 0}} &= 1 \\ \psi(x, \mu > 0) &= e^{\frac{-\sigma x}{\mu > 0}} \end{aligned}$$

Integrating from x to 10 for $\mu < 0$, we obtain

$$\begin{aligned} \psi(10, \mu < 0) e^{\frac{\sigma 10}{\mu < 0}} - \psi(x, \mu < 0) e^{\frac{\sigma \cdot x}{\mu < 0}} &= 0 \\ -\psi(x, \mu < 0) e^{\frac{\sigma \cdot x}{\mu < 0}} &= 0 \\ \psi(x, \mu < 0) &= 0 \end{aligned}$$

Here we can note that there is no flux traveling to the left, and will focus on the flux traveling to the right. Integrating over all $\mu > 0$, and using a substitution of $z = 1/\mu$ and $d\mu = -z^{-2}$:

Problem 5 continued on next page...

$$\begin{aligned}\phi^+(x) &= 2\pi \int_0^1 \psi(x, \mu > 0) d\mu \\ &= 2\pi \int_0^1 e^{-\frac{\sigma x}{\mu}} d\mu \\ &= 2\pi \int_\infty^1 -\frac{e^{-\sigma x z}}{z^2} dz \\ &= 2\pi \int_1^\infty \frac{e^{-\sigma x z}}{z^2} dz \\ &= 2\pi E_2(\sigma x)\end{aligned}$$

I think this might help later on, if we define ϕ like:

$$\phi^+(x') = 2\pi E_2(x')$$

where $x' = \sigma x$.

Now to use PCE expansion to estimate the distribution, mean, and variance of $\phi(x)$. First off, I want to say that I have no idea whats going on. Next, because we have a gamma distribution, I suppose we should use Laguerre Polynomials. Where:

$$\phi(x') = \sum_{n=0}^{\infty} c_n L_n^{(\alpha)}(\beta x')$$

and

$$c_n = \frac{n!}{\Gamma(n+\alpha+1)} \int_0^\infty 2\pi \ \phi\left(\frac{x \cdot z}{\beta}\right) z^\alpha e^{-z} L_n^{(\alpha)}(z) dz$$

It should be noted that z is the standardized gamma distribution, with $z = \beta\sigma$, σ being our original gamma distribution. Also, I am grouping 2π with the flux for all the calculations, not the fastest, but I am not trying to break any records (except the one of taking the longest to do homeworks).

In order to estimate the coefficients in the expansion we have to evaluate a wonderful integral. In order to estimate the integral, will use Gauss-Laguerre quadrature (like I have any idea what that is) to have as few evaluations of the integrand as possible.

The quadrature rule has the form

$$\int_0^\infty f(z) z^\alpha e^{-z} dz \approx \sum_{i=1}^{n} w_i f(z_i)$$

Where z_i are the n roots of $L_n^{(\alpha)}(z)$, and the weights are given by:

$$w_i = \frac{\Gamma(n+\alpha) z_i}{n!(n+\alpha)(L_{n-1}^{\alpha}(z_i))^2}$$

Looking at the quadrature rule, I think $f(z)$, in our instance would have to be

$$f(z) = 2\pi \ \phi\left(\frac{x \cdot z}{\beta}\right) L_n^{(\alpha)}(z)$$

This is potentially confusing about the two n indice, so I'll put it all on one line, and hope its correct, if not then the only person I can blame is myself. The notes aren't very clear on what $f(z)$ is.

Problem 5 continued on next page...

$$c_n \approx \frac{n!}{\Gamma(n+\alpha+1)} \sum_{i=1}^{n'} w_i f(z_i)$$

$$c_n \approx \frac{n!}{\Gamma(n+\alpha+1)} \sum_{i=1}^{n'} \frac{\Gamma(n'+\alpha) z_i}{n'!(n'+\alpha)(L^{\alpha}_{n'-1}(z_i))^2} 2\pi \ \phi\left(\frac{x \cdot z_i}{\beta}\right) L_n^{(\alpha)}(z_i)$$

Where n' will increase until the summation is not changing, this is potentially confusing (for me). n' will start at 1. At which point the Laguerre polynomial $L_1^{\alpha}(x)$ will have a single root. The 'summation' will be a single term. Then n' will increase to 2, where there are two roots. The 'summation' will sum results from those two roots, **and not use the previous summation** (except to compare - not to add onto). The summation from $n' = 1$ and $n' = 2$ will be compared, if there is no difference (note there could be some zero terms) then the n' stops increasing, and the most recent summation is what we use moving forward.

Also n is the constant we are solving for. This way, the $L_n^{\alpha}(z_i)$ term on the right side will only be zero for when $n' = n$ (I hope this is correct).

According to the notes, the variance is:

$$\text{Var}(G) = \sum_{n=1}^{\infty} \frac{\Gamma(n+\alpha+1)}{\Gamma(\alpha+1)n!} c_n^2$$

And c_0 is:

$$c_0 = \int_0^{\infty} 2\pi \ \phi\left(\frac{x \cdot z}{\beta}\right) \frac{z^{\alpha} e^{-z}}{\Gamma(\alpha+1)} dz = E[g(X)]$$
$$\approx \sum_{i=1}^{n'} \frac{\Gamma(n'+\alpha) z_i}{n'!(n'+\alpha)(L^{\alpha}_{n'-1}(z_i))^2} 2\pi \ \phi\left(\frac{x \cdot z_i}{\beta}\right)$$

To check if this is correct, we can change the ϕ term for $cos(z/\beta)$ with Z $\sim$ G(1,2) and check to see if c_n converge to what Dr. McClarren has in his notes...which I'm hoping are correct.

Listing 3: Code for Comparison

```
#!/usr/bin/env python3

############################################################
#################### Import packages ########################
############################################################

import time
start_time = time.time()
import Functions as fun
from mpmath import expint

############################################################
####################### Function ###########################
############################################################
```

```
15
def Determine_cn(n=0,alpha=1,beta=2,x=0):
    """
    This function will determin the cn constant
    dont forget to change the first function dude
20  to your function
    """
    diff=5;tol=0.000001;nprime=1;Sum=1000
    Cprefix=fun.fact(n)/(fun.gamma(n+alpha+1))
    while diff>tol:
25      roots=fun.Lroots(nprime,alpha)
        weights=fun.weight(nprime,alpha,roots)
        function=fun.np.cos(roots/beta) #Change me
        function=function*fun.LagEval(n,alpha,roots)
        WF=weights*function
30      Sumhold=sum(WF)
        cn=Cprefix*Sumhold
        diff=abs(Sum-Sumhold)
        Sum=fun.copy.copy(Sumhold)
        #print(np,cn)
35      nprime=nprime+1
        if nprime==100:
            print("Did not converge on quadrature")
            quit()
    return(cn)
40

#################################################################
#################### Calculations ###############################
#################################################################


45
for i in range(0,6):
    cn=Determine_cn(n=i)
    print(i,cn)


50


#################### Time To execute ################

print("--- %s seconds ---" % (time.time() - start_time))
```

Table 3: Compare to the Dr. MC

C_n	MC notes	Code
0	0.48	0.48
1	0.35	0.35
2	0.04	0.04
3	-0.05	-0.05
4	-0.03	-0.03
5	-0.00	-0.00

Problem 5 continued on next page...

I am glad that works, now results

Listing 4: Code for Problem

```
#!/usr/bin/env python3

################################################################
#################### Import packages ###########################
################################################################

import time
start_time = time.time()
import Functions as fun

x=0.5;alpha=10;beta=0.1;NumofC=10
print("For x = "+str(x))
################################################################
############### Deterministic Calculations #####################
################################################################

cn=[]
for n in range(0,NumofC):
    cn.append(fun.Determine_cn(n,alpha,beta,x))

Dphi=0 #Deterministic phi
for n in range(0,NumofC):
    Dphi=Dphi+cn[n]*fun.LagEval(n,alpha,beta*x)
print("Deterministic Calc mean is "+str(Dphi))

################################################################
#################### Monte Calculations ########################
################################################################

N=10000  #Samples
Nbins=60 #Hist Plot
#Nstrata=10
filename="mean2.pdf"

#RandomNumbers=fun.np.random.uniform(0,1,N)
#RandomNumbers=fun.Rstrat(N,Nstrata)  #strat sampling
#l=lhsmdu.sample(1,N)  #Hyper cube sampling
#RandomNumbers=l[0].A1
#RandomNumbers=fun.Rvdc(N,7)    #Halton sequence
#SamplesSigma=fun.gamma.ppf(RandomNumbers,0.1,10)
SamplesSigma=fun.np.random.gamma(shape=10,scale=0.1,size=N)

x=1
Mphi=[] #Monte Phi
for i in range(0,len(SamplesSigma)):
    Mphi.append(fun.PhiEval(x*SamplesSigma[i]))

print("Monte Calc mean is "+str(fun.mean(Mphi)))
print("")
#Plot the data, and plot fitted PDF
```

Problem 5 continued on next page...

```
Xlabel="Distribution of flux"
(n,bins,ax,fig)=fun.HIST(Xlabel,Mphi,Nbins,N)
(ax,fig)=fun.HISTDataToPDF(n,bins,ax,fig)
fun.plt.savefig(filename)

print("")
##################### Time To execute ################

print("--- %s seconds ---" % (time.time() - start_time))
```

Table 4: Different Sampling Techniques

Location	Distribution	Mean	Variance
1	Dist	mean	Variance
1.5	Dist	mean	Variance
3	Dist	mean	Variance
5	Dist	mean	Variance