2)

```
In [154]: n=100000
     ...: X1=np.random.randn(n)
     ...: X2=np.random.randn(n)
     ...: Y=np.zeros(n)
     ...: for i in range(n):
     ...:     Y[i]=min(X1[i],X2[i])
     ...:
     ...: mean_Y=np.mean(Y)
     ...: mean_Y
Out[154]: -0.567442457616232
```

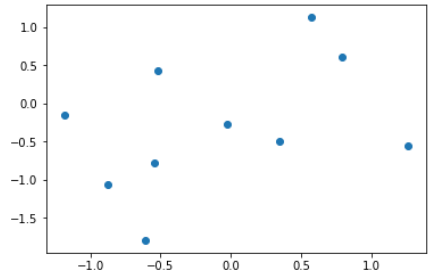
5)

```
> f<-function(x){x^(-0.5)*exp(x)}
> integrate(f,0,1)
2.925303 with absolute error < 9.4e-06
```

7)

a)

```
In [94]: n=10
    ...: mu = np.array([[0, 0]]).T
    ...: Sigma = np.array([[1, 0.5], [0.5, 1]])
    ...: R = cholesky(Sigma)
    ...: s = np.dot(R,np.random.randn(2,n)) + mu
    ...: plt.scatter(s[0,:],s[1,:])
Out[94]: <matplotlib.collections.PathCollection at 0x1198b690cf8>
```




```
In [95]: s
Out[95]:
array([[-0.54432308, -0.51744262,  0.34963533,  1.26343555,  0.79325468,
         0.5720538 , -1.1850183 , -0.02806256, -0.87507593, -0.6075421 ],
       [-0.77457526,  0.42486786, -0.49585424, -0.5587643 ,  0.61373061,
         1.12671067, -0.14531576, -0.27307146, -1.05408875, -1.7905015 ]])
```

Observation

```
In [96]: sample_sigma=np.cov(s[0,:],s[1,:])
    ...: sample_mean=np.reshape(np.mean(s,1),(2,1))
```

```
In [97]: sample_mean
Out[97]:
array([[-0.07790852],
       [-0.29268621]])
```

Mean

```
In [98]: sample_sigma
Out[98]:
array([[0.63536345, 0.26843084],
       [0.26843084, 0.7267491 ]])
```

Covariance

```
In [99]: def md(t):
    ...:     return(np.linalg.inv(1+np.dot(np.dot((t-sample_mean).T,np.linalg.inv(sample_sigma)),(t-mu))))
```

```
In [100]: mds=np.zeros((1,n))
     ...: maxmd=0
     ...: maxi=0
     ...: for i in range(n):
     ...:     t=np.reshape(s[:,i],(2,1))
     ...:     mds[0,i]=md(t)
     ...:     if md(t)>maxmd:
     ...:         maxmd=md(t)
     ...:         maxi=i
```

```
In [101]: mds
Out[101]:
array([[0.6051441 , 0.4337282 , 0.61050523, 0.19414774, 0.42878172,
        0.30891884, 0.29229238, 1.00158896, 0.38884484, 0.21310353]])
```

Depth

```
In [102]: maxmd
Out[102]: array([[1.00158896]])
```

```
In [103]: maxi
Out[103]: 7
```

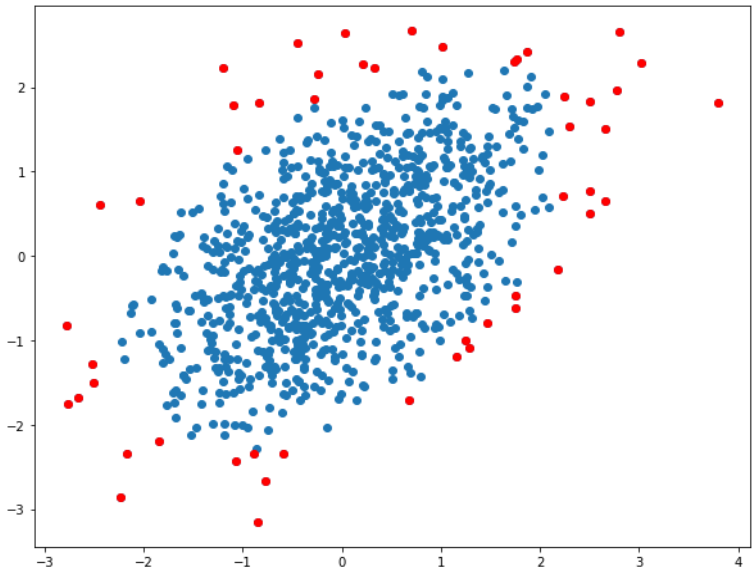
```
In [104]: s[:,7]
Out[104]: array([-0.02806256, -0.27307146])
```

The bivariate vector that maximizes the depth function

b)

```
In [146]: n=1000
     ...: mu = np.array([[0, 0]]).T
     ...: Sigma = np.array([[1, 0.5], [0.5, 1]])
     ...: R = cholesky(Sigma)
     ...: s = np.dot(R,np.random.randn(2,n)) + mu
     ...: #plt.scatter(s[0,:],s[1,:])
     ...:
     ...:
     ...: sample_sigma=np.cov(s[0,:],s[1,:])
     ...: sample_mean=np.reshape(np.mean(s,1),(2,1))
     ...:
     ...: def md(t):
     ...:     return(np.linalg.inv(1+np.dot(np.dot((t-sample_mean).T,np.linalg.inv(sample_sigma)),(t-mu))))
     ...:
     ...:
     ...: mds=np.zeros((1,n))
     ...: maxmd=0
     ...: maxi=0
     ...: for i in range(n):
     ...:     t=np.reshape(s[:,i],(2,1))
     ...:     mds[0,i]=md(t)
     ...:     if md(t)>maxmd:
     ...:         maxmd=md(t)
     ...:         maxi=i
     ...:
     ...:
     ...: md_5=np.percentile(mds,5)
     ...: j=0
     ...: flag=np.zeros((2,50))
     ...: for i in range(n):
     ...:     if mds[0,i]<=md_5:
     ...:         flag[:,j]=s[:,i]
     ...:         j=j+1

     ...: plt.figure(figsize=(10,8))
     ...: plt.plot(s[0,:],s[1,:],'o',flag[0,:],flag[1,:],'ro')
Out[146]:
[<matplotlib.lines.Line2D at 0x1198bccb668>,
 <matplotlib.lines.Line2D at 0x1198bccb780>]
```



3)

```
data systems;
input x1 x2 x3 y;
cards;
1 0 0 1.333333333
0 1 0 0
0 0 1 0.26666
;
run;

proc nlin;
parameters lam0=1 lam1=0 lam2=0;
model y=(lam0+lam1+lam2)*x1+
        (lam0*0.654654-lam2*0.654654)*x2+
        (lam0*0.654654**2+lam2*0.654654**2)*x3;
run;
```

Parameter	Estimate	Approx Std Error	Approximate 95% Confidence Limits	
lam0	0.3111	.	.	.
lam1	0.7111	.	.	.
lam2	0.3111	.	.	.