Machine Intelligence 2

Exercise 05

Projection Methods: Principle Component Analysis

Group Members:
Xugang Zhou
Fangzhou Yang

Tutor:
Timm Lochmann

May 21, 2013

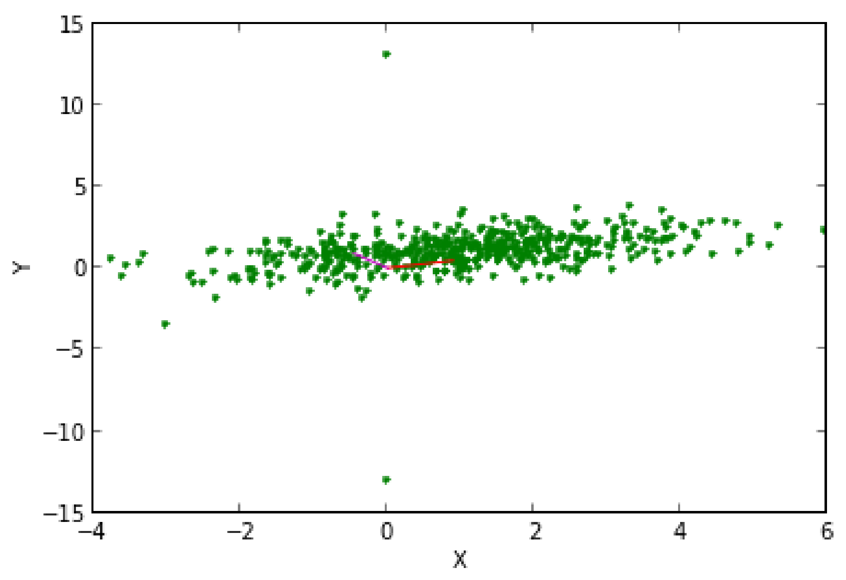
```
#The Followings are some imported package and functions, which will be used
    later..
from numpy import *
import matplotlib
import matplotlib.pyplot as plt
import scipy as sp
from scipy import sparse
from scipy.sparse import linalg
from numpy import matrix
import math

#function to get the centered data
def get_CenteredData(data, dimension):
    m = [0 for i in range(dimension)]
    for i in range(dimension):
        m[i] = sum(data[i][:])/len(data[i][:])
    for i in range(dimension):
        for j in range(dimension):
            data[i][j] = data[i][j] - m[i]
    return data

#function to get the covariance matrix
def get_CoMatrix(data, dimension):
    C = [[0 for i in range(dimension)] for j in range(dimension)]
    p = len(data[1][:])
    m = [0 for i in range(dimension)]
    for i in range(dimension):
        m[i] = sum(data[i][:])/p
    for i in range(dimension):
        for j in range(dimension):
            for a in range(p):
                C[i][j] += ( (data[i][a] - m[i]) * (data[j][a] - m[j]) )/p
    return C

#function to get eigenvalues and eigenvectors
def get_PC(data, dimension, nume):
    C = get_CoMatrix(data, dimension)
    if nume == dimension:
        evals, evecs = np.linalg.eig(asmatrix(C))
    else:
        evals, evecs = sp.sparse.linalg.eigs(asmatrix(C), k = nume)
    return evals, evecs
```

1 5.1. Preprocessing

1.1 a.) PC1 and PC2 from dataset pca2.csv

```
#read data from pca2.csv
data01 = loadtxt('PCAdata2/pca2.csv', delimiter = ',', unpack =True, usecols =
    (0, 1), skiprows = 1 )
#print data01
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(data01[0], data01[1], 'g.')
ax.set_xlabel('X')
ax.set_ylabel('Y')

evals ,evecs = get_PC(data01, 2, 2)
print 'eigenvectors\n', evecs
print 'eigenvalues\n',evals

ax.plot([0, evecs[0,0]],[0,evecs[1,0]],'r-')
ax.plot([0, evecs[0,1]],[0,evecs[1,1]],'m-')
plt.show()
```

```
eigenvectors
[[ 0.88773892 -0.46034728]
 [ 0.46034728  0.88773892]]
eigenvalues
[ 3.15267416  1.20272016]
```

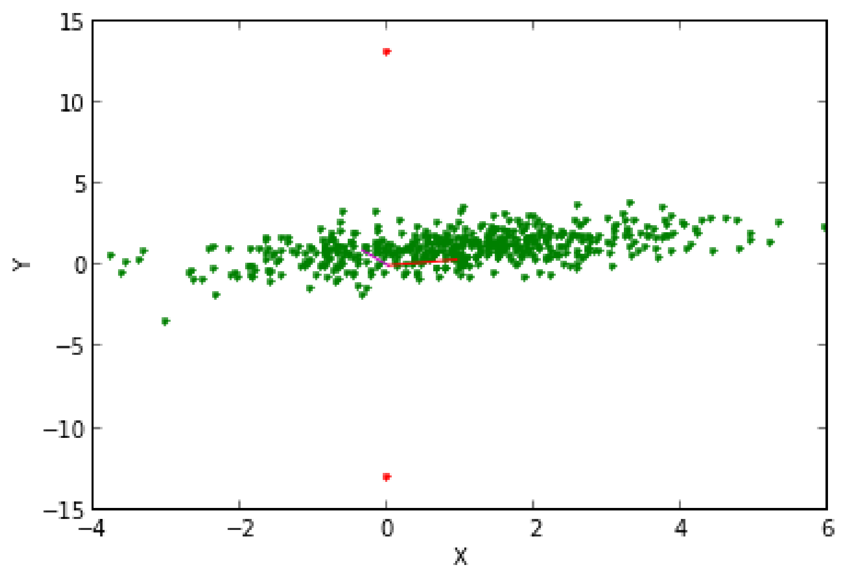


1.2 b.) After removing the observation 17, 157:

```
p = data01.shape[1]
data02 = [[0 for i in range(p-2)] for j in range(2)]
```

```
r = 0
for j in range(p):
   for i in range(2):
        if (j==16):
            r = 1
            break
        if (j==156):
            r = 2
            break
        data02[i][j-r] = data01[i][j]

fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(data02[0], data02[1], 'g.')
ax.set_xlabel('X')
ax.set_ylabel('Y')

#plot the two points , which are removed from the dataset
ax.plot(data01[0][16],data01[1][16], 'r.')
ax.plot(data01[0][156],data01[1][156], 'r.')

evals ,evecs = get_PC(data02, 2, 2)

print 'eigenvectors\n', evecs
print 'eigenvalues\n',evals

ax.plot([0, evecs[0,0]],[0,evecs[1,0]],'r-')
ax.plot([0, evecs[0,1]],[0,evecs[1,1]],'m-')
plt.show()
```

```
eigenvectors
[[ 0.93549122 -0.35334993]
 [ 0.35334993  0.93549122]]
eigenvalues
[ 3.04753257  0.63885724]
```



As we can see from the results, after removing the two noisy points, the eigenvectors changed a lot.

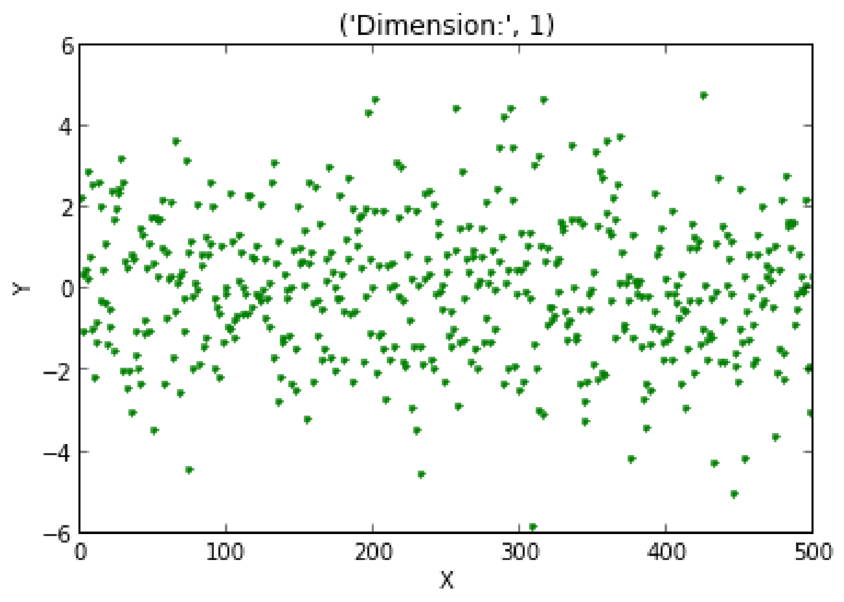
2 5.2. Whitening

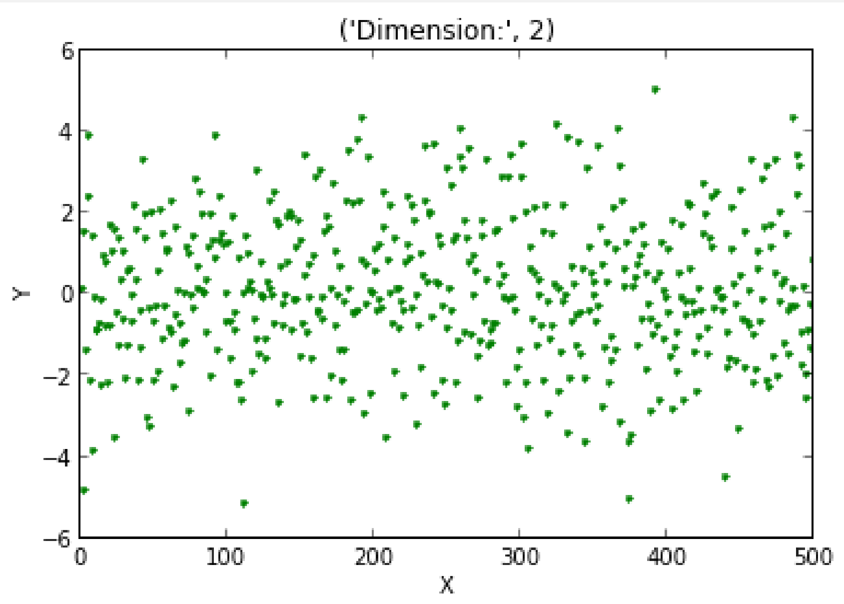
2.1 a.) Load the dataset pca4.csv

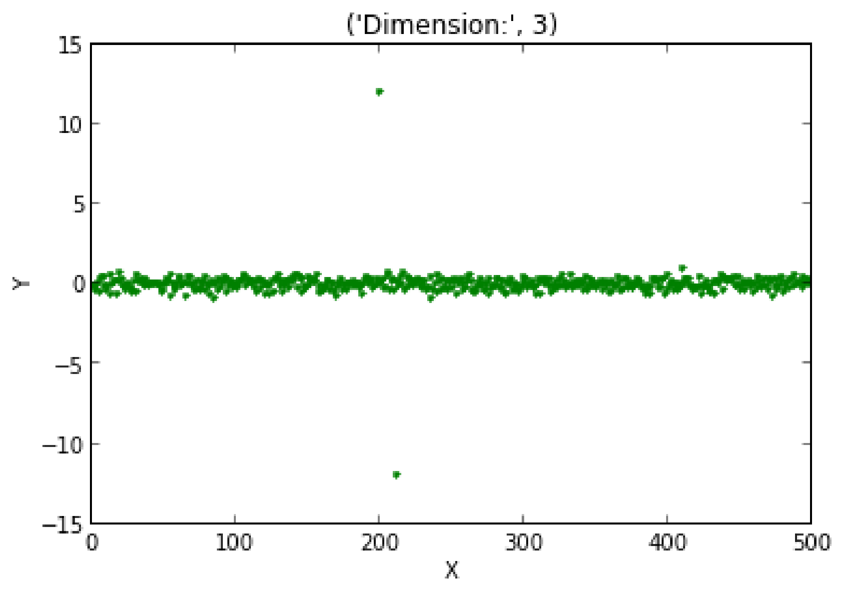
```
data04 = loadtxt('PCAdata2/pca4.csv', delimiter = ',', unpack =True, usecols =
     (0, 1, 2 ,3), skiprows = 1 )
```

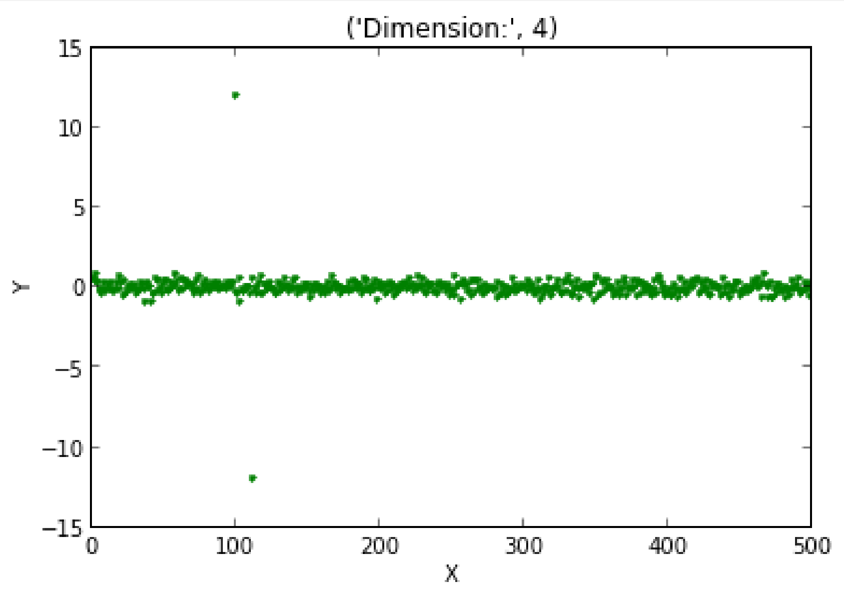
To check outliers in individual variables:

```
X = [ i+1 for i in range (len(data04[0,:]))]

for m in range (4):
    fig = plt.figure(m+1)
    ax = fig.add_subplot(111)
    ax.plot(X, data04[m], 'g.')
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    tmp = "Dimension:",m+1
    plt.title(tmp)
    plt.show()
```


('Dimension:', 1)
Y
X
6
4
2
0
−2
−4
−6
0
100
200
300
400
500


('Dimension:', 2)
Y
X
6
4
2
0
−2
−4
−6
0
100
200
300
400
500



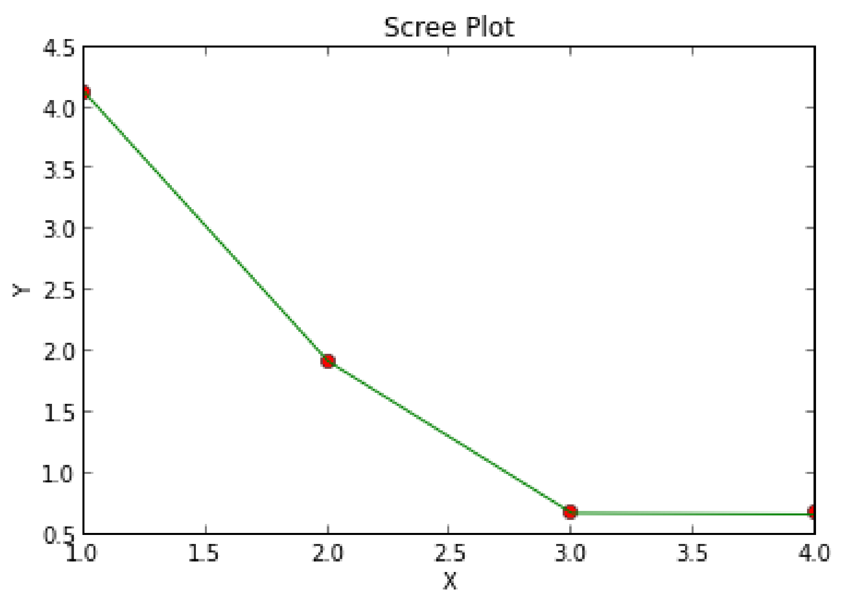


2.2 b.) PCA and Scree plot

```
evals, evecs = get_PC(data04, 4, 4)
print 'eigenvectors\n', evecs
print 'eigenvalues\n',evals
X = [ 0 for i in range (4)]
Y = [ 0 for i in range (4)]
for i in range(4):
    X[i] = i+1
    Y[i] = evals[i]

fig = plt.figure()
ax = fig.add_subplot(111)
```

```
ax.plot(X, Y, 'ro')
ax.plot(X, Y, 'g-')
ax.set_xlabel('X')
ax.set_ylabel('Y')
plt.title('Scree Plot')
plt.show()
```

```
eigenvectors
[[-0.66808317 -0.7440606   0.00612014  0.00111974]
 [-0.74406218  0.66802535 -0.00553581 -0.00910804]
 [ 0.00594509 -0.0054947   0.16150367 -0.98683891]
 [ 0.00100359 -0.00926113 -0.98683761 -0.16144585]]
eigenvalues
[ 4.12502515  1.920516    0.67626699  0.66686614]
```



According to the scree plot, the first two PCs can represent the data well.

2.3 c.) Whiten the Data

```
data_centered = get_CenteredData(data04,4)
evals, evecs = get_PC(data04, 4, 4)
E = matrix(evecs)
Dd = matrix(np.diag([ 1/math.sqrt(evals[i]) for i in range(4)]))
X = matrix(data_centered).T

Z =( X * E) * Dd

data_whiten = [[Z[j,i] for j in range(len(data_centered[0][:]))] for i in
    range(4)]

print 'data whitened done!'
```

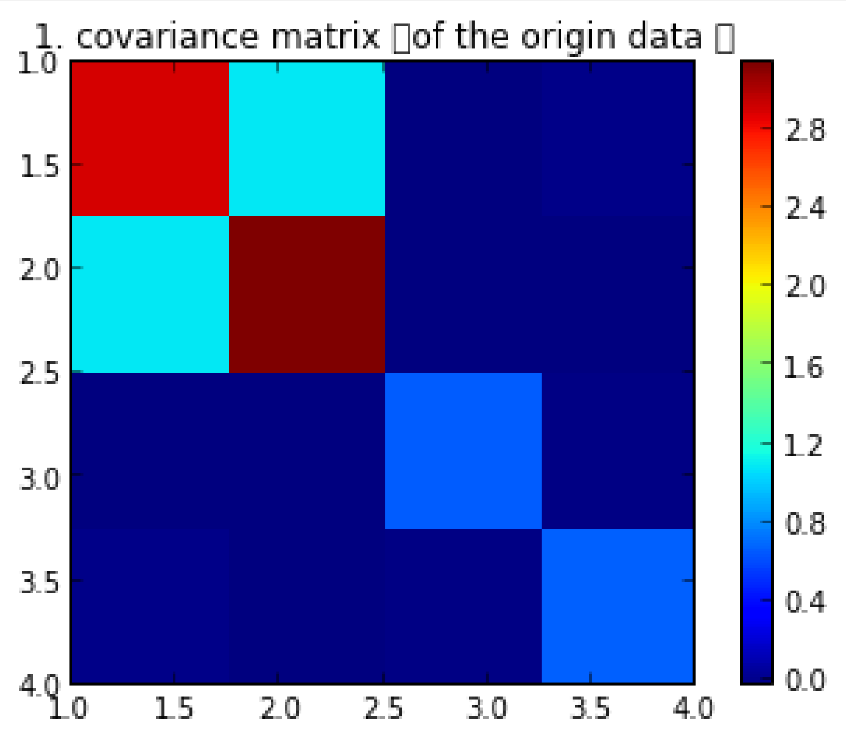
```
data whitened done!
```

2.4 d.) Heat plots

1. covariance matrix of the origin data

```
C = get_CoMatrix(data04,4)

print matrix(C)

plt.title('1. covariance matrix of the origin data ')
plt.imshow(C,interpolation="nearest",extent=[1,4,4,1])
plt.colorbar()
plt.show()
```

```
[[  2.90437328e+00   1.09555553e+00  -8.51729760e-03   6.28739894e-03]
 [  1.09555553e+00   3.14313145e+00  -2.00219487e-02  -1.08781497e-02]
 [ -8.51729760e-03  -2.00219487e-02   6.67233603e-01  -1.33954775e-03]
 [  6.28739894e-03  -1.08781497e-02  -1.33954775e-03   6.76240053e-01]]
```

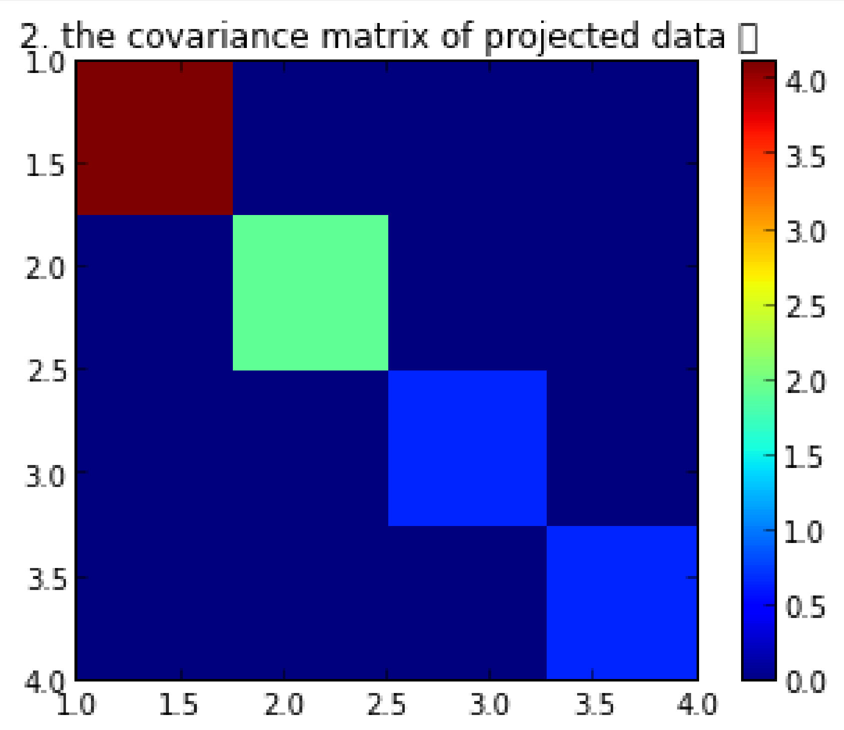


2. the covariance matrix of the data projected onto PC1-PC4

```
evals, evecs = get_PC(data04, 4, 4)
B = matrix(evecs)
A = matrix(data04).T
X = A * B
data_projected = [[ X[i,j] for i in range(len(data04[0][:]))] for j in range
    (4)]
```

```
C = get_CoMatrix(data_projected, 4)
print matrix(C)
plt.title('2. the covariance matrix of projected data ')
plt.imshow(C,interpolation="nearest",extent=[1,4,4,1])
plt.colorbar()
plt.show()
```

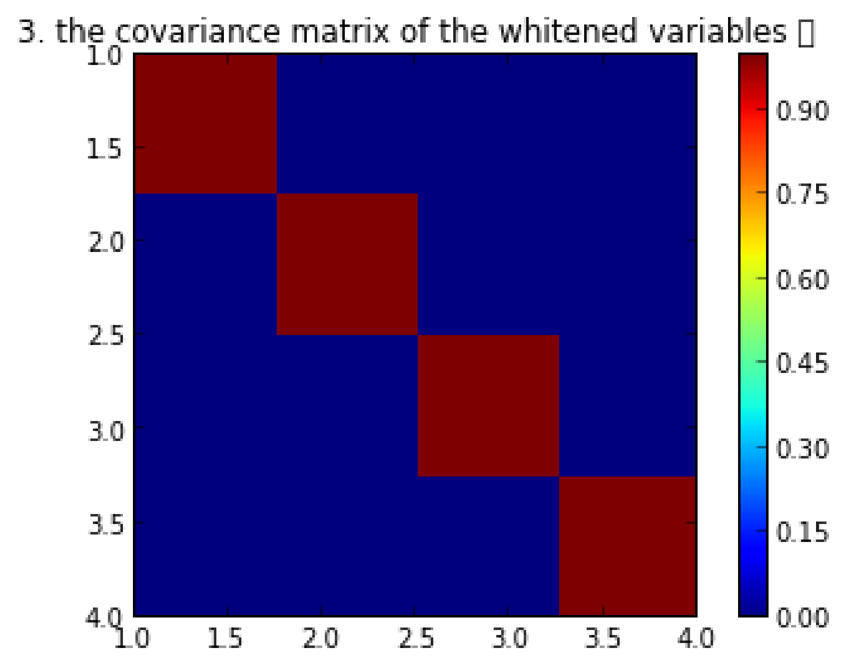
```
[[  4.12591986e+00  -3.26366538e-15   1.37693676e-17   6.74102701e-17]
 [ -3.26366538e-15   1.92186560e+00  -1.46367293e-18   2.33103467e-18]
 [  1.37693676e-17  -1.46367293e-18   6.76341742e-01  -8.57269171e-15]
 [  6.74102701e-17   2.33103467e-18  -8.57269171e-15   6.66851179e-01]]
```



3. the covariance matrix of the whitened variables

```
C = get_CoMatrix(data_whiten,4)
print matrix(C)
plt.title('3. the covariance matrix of the whitened variables ')
plt.imshow(C,interpolation="nearest",extent=[1,4,4,1])
plt.colorbar()
plt.show()
```

```
[[  1.00000000e+00  -1.17606120e-15   1.04083409e-17   2.57023678e-17]
 [ -1.17606120e-15   1.00000000e+00  -9.10729825e-18  -1.25902977e-17]
 [  1.04083409e-17  -9.10729825e-18   1.00000000e+00  -1.27597179e-14]
 [  2.57023678e-17  -1.25902977e-17  -1.27597179e-14   1.00000000e+00]]
```

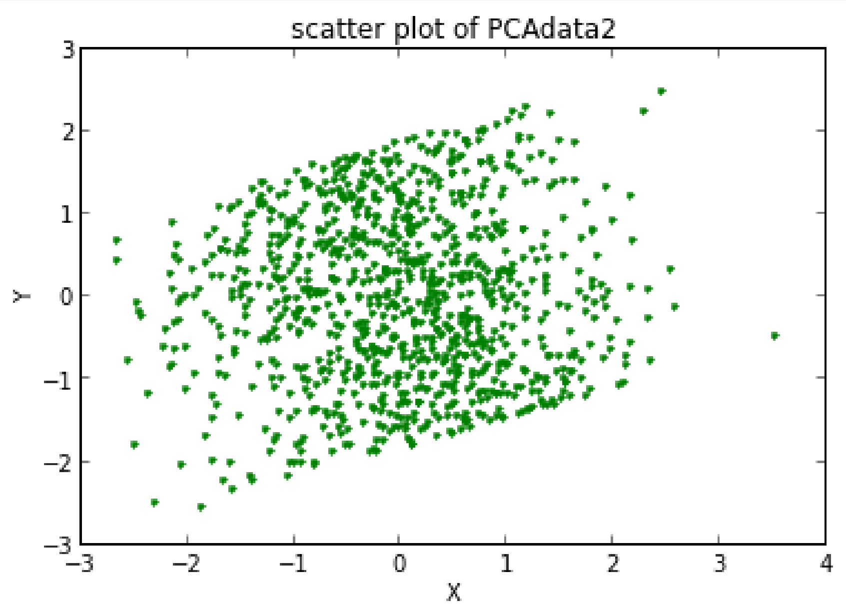


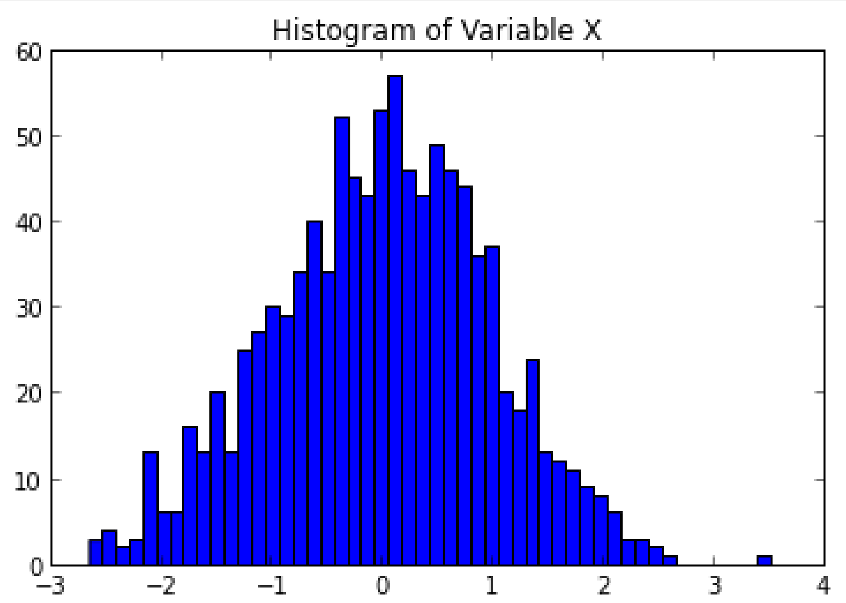
3 5.3 Rotation

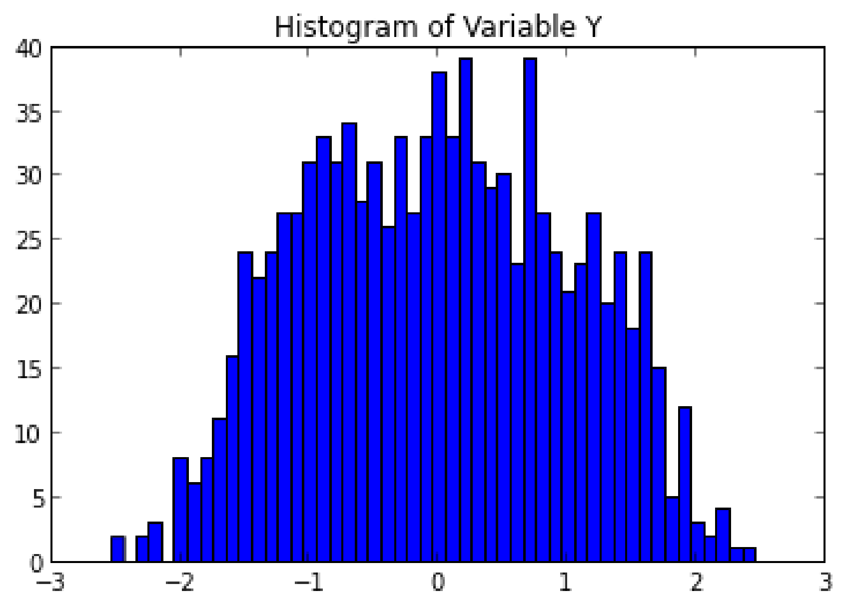
3.1 a.) Dataset Loading and Variables Estimating

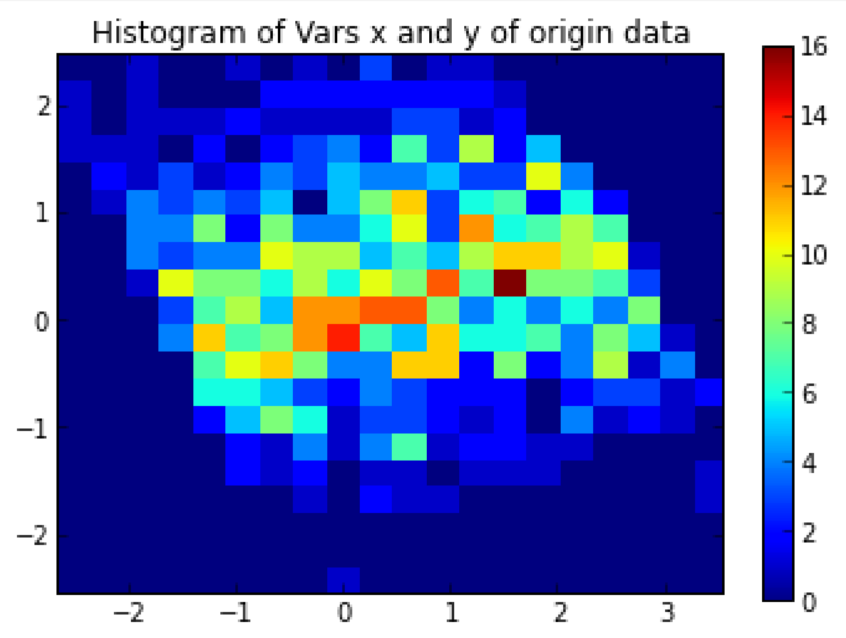
```
data2b = loadtxt('PCAdata2/pca2b.csv', delimiter = ',', unpack =True, usecols
    = (0, 1), skiprows = 1 )

fig = plt.figure(1)
ax = fig.add_subplot(111)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_title('scatter plot of PCAdata2')
ax.plot(data2b[0], data2b[1], 'g.')
fig.show()

plt.clf()
plt.hist(data2b[0], bins=50, color='blue')
plt.title('Histogram of Variable X')
plt.show()

plt.clf()
plt.hist(data2b[1], bins=50, color='blue')
plt.title('Histogram of Variable Y')
plt.show()

#2 vars
heatmap, xedges, yedges = np.histogram2d(data2b[0], data2b[1], bins=20)
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]
```

```
plt.clf()
plt.title('Histogram of Vars x and y of origin data')
plt.imshow(heatmap,interpolation="nearest", extent=extent)
plt.colorbar()
plt.show()
```







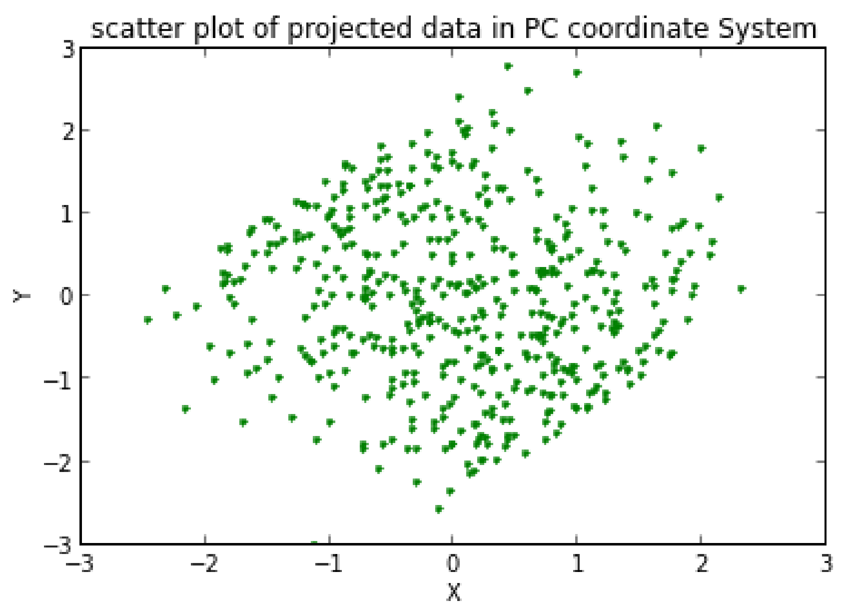


3.2 b.) PCA and Projection

```
evals, evecs = get_PC(data2b, 2, 2)
print 'eigenvectors\n', evecs
print 'eigenvalues\n',evals
B = matrix(evecs)
A = matrix(data2b).T
X = A * B
data2b_projected = [[ X[i,j] for i in range(len(data02[0][:]))] for j in range
    (2)]

fig = plt.figure()
```

```
ax = fig.add_subplot(111)
ax.set_title('scatter plot of projected data in PC coordinate System')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.plot(data2b_projected[0], data2b_projected[1], 'g.')
plt.show()
```

```
eigenvectors
[[-0.90630779 -0.42261826]
 [ 0.42261826 -0.90630779]]
eigenvalues
[ 0.94875444  1.04924556]
```

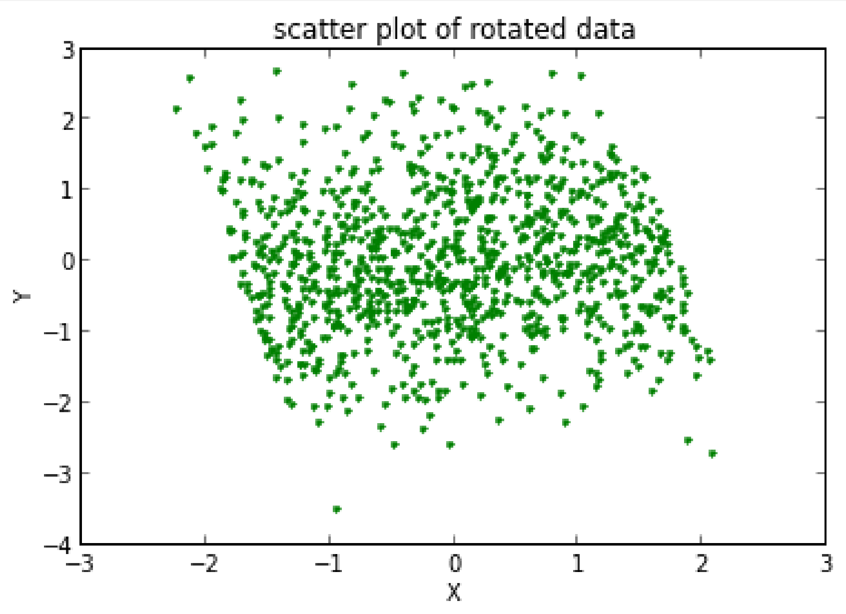


3.3 c.) Whiten the data and do rotation of 45 degree

```
data2b_centered = get_CenteredData(data2b,2)
evals, evecs = get_PC(data2b, 2, 2)
E = evecs
D = evals


data2b_whiten = [[0 for j in range(len(data2b_centered[0][:]))] for i in range
    (2)]


for j in range(len(data2b_centered[0][:])):
    for i in range(2):
        data2b_whiten[i][j] = 0
        for m in range(2):
            data2b_whiten[i][j] = data2b_centered[m][j] * E[m,i] +
                data2b_whiten[i][j]
```

```
        data2b_whiten[i][j] = 1/math.sqrt(evals[i]) * data2b_whiten[i][j]

print 'data whitened done!'


R = matrix ([[math.cos(45), -math.sin(45)],[math.sin(45),math.cos(45)]])
Z = matrix(data2b_whiten).T
Zrot = (R * Z.T).T
data2b_rotate = [[Zrot[j,i] for j in range(len(data2b_centered[0][:]))] for i
    in range(2)]


print 'After Rotation..'
fig = plt.figure()
ax = fig.add_subplot(111)
ax.set_title('scatter plot of rotated data')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.plot(data2b_rotate[0], data2b_rotate[1], 'g.')
plt.show()
```

```
data whitened done!
After Rotation..
```

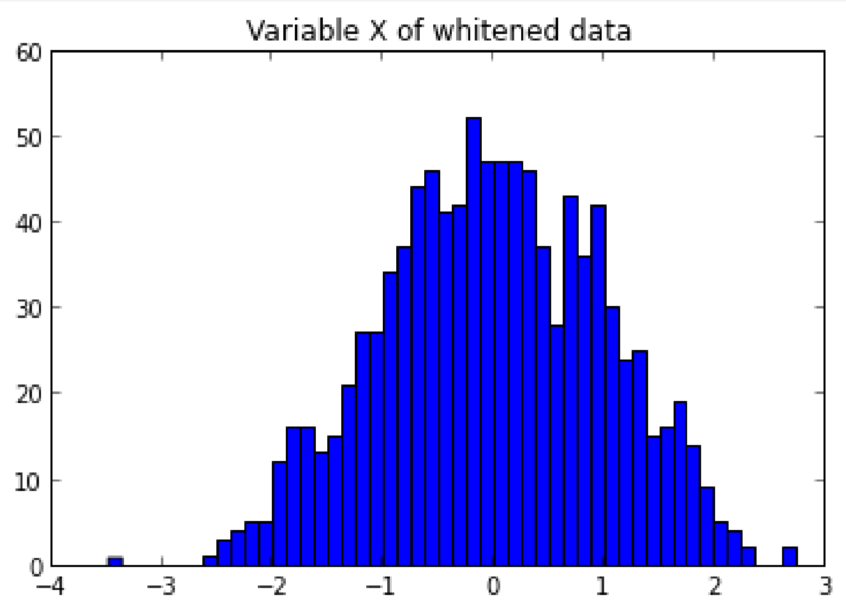


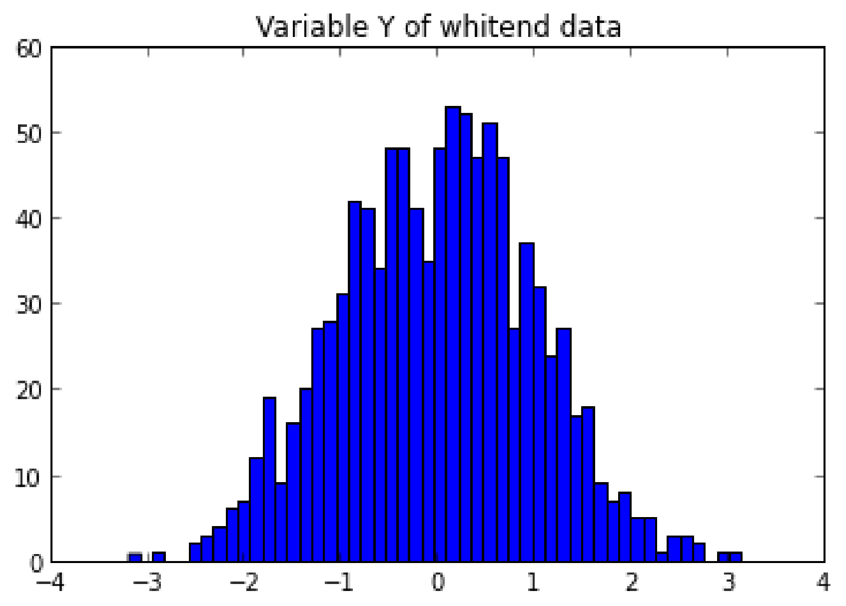
3.4 d. marginal densities of variables

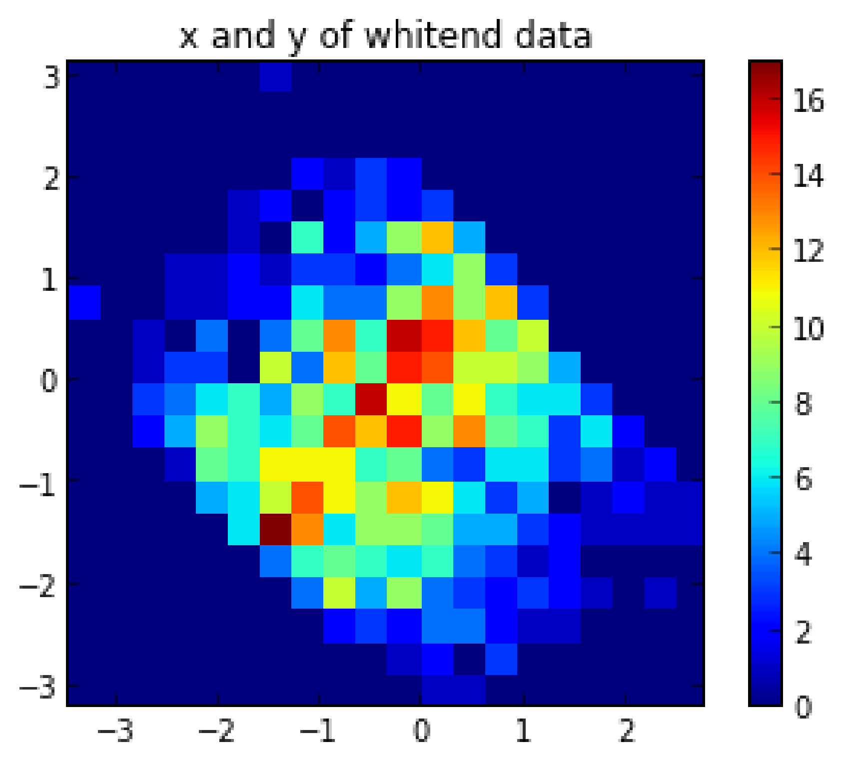
whitened variables zi

```
plt.clf()
plt.hist(Z[:,0], bins=50, color='blue')
plt.title('Variable X of whitened data')
plt.show()
```

```
plt.clf()
plt.hist(Z[:,1], bins=50, color='blue')
plt.title('Variable Y of whitend data')
plt.show()

#2 vars
heatmap, xedges, yedges = np.histogram2d(data2b_whiten[0], data2b_whiten[1],
    bins=20)
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]

plt.clf()
plt.title('x and y of whitend data')
plt.imshow(heatmap,interpolation="nearest", extent=extent)
plt.colorbar()
plt.show()
```



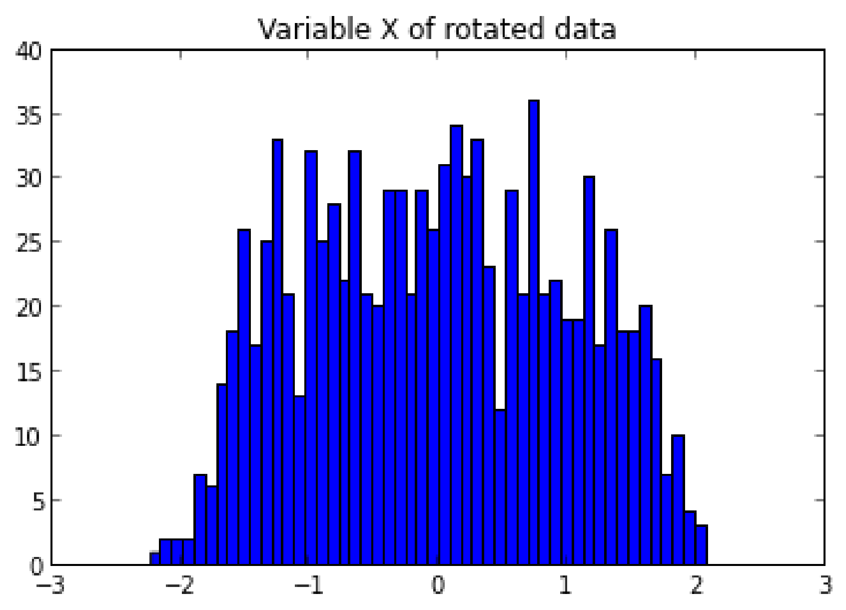


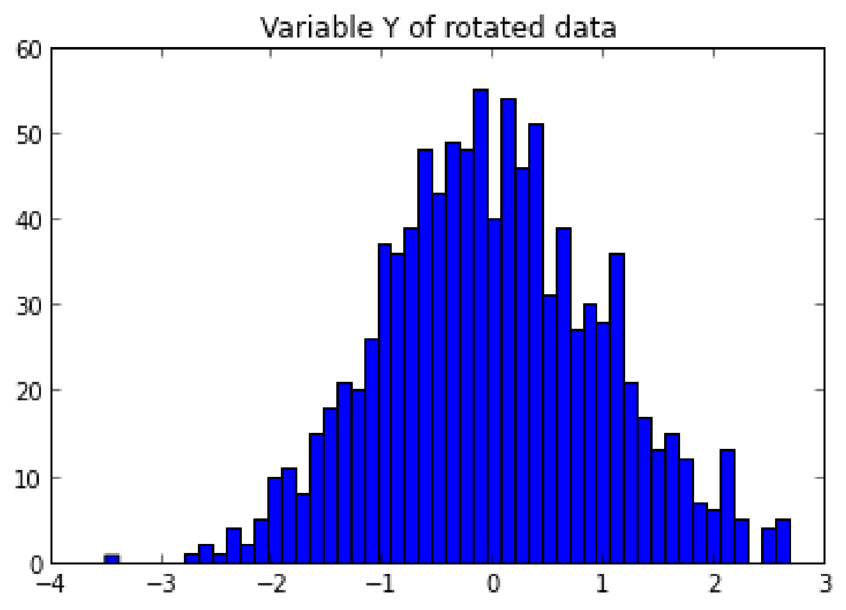


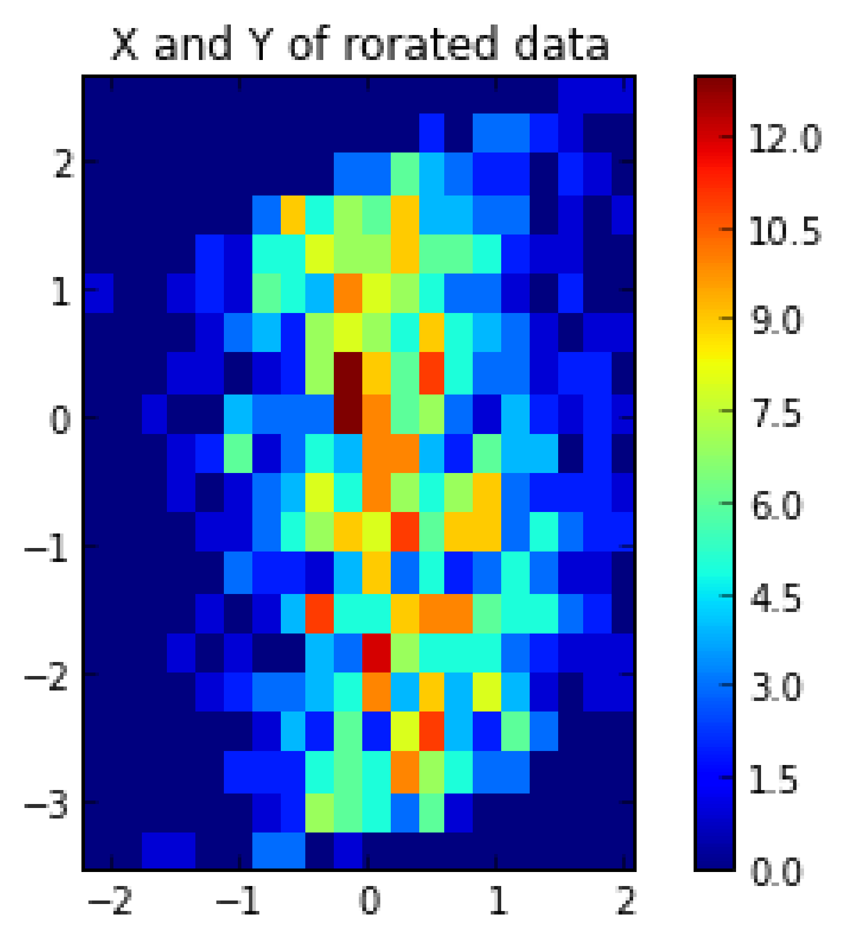
the rotated variables Zrot

```
plt.clf()
plt.hist(Zrot[:,0], bins=50, color='blue')
plt.title('Variable X of rotated data')
plt.show()

plt.clf()
plt.hist(Zrot[:,1], bins=50, color='blue')
```

```
plt.title('Variable Y of rotated data')
plt.show()

#2 vars
heatmap, xedges, yedges = np.histogram2d(data2b_rotate[0], data2b_rotate[1],
    bins=20)
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]
plt.clf()
plt.title('X and Y of rorated data')
plt.imshow(heatmap,interpolation="nearest", extent=extent)
plt.colorbar()
plt.show()
```



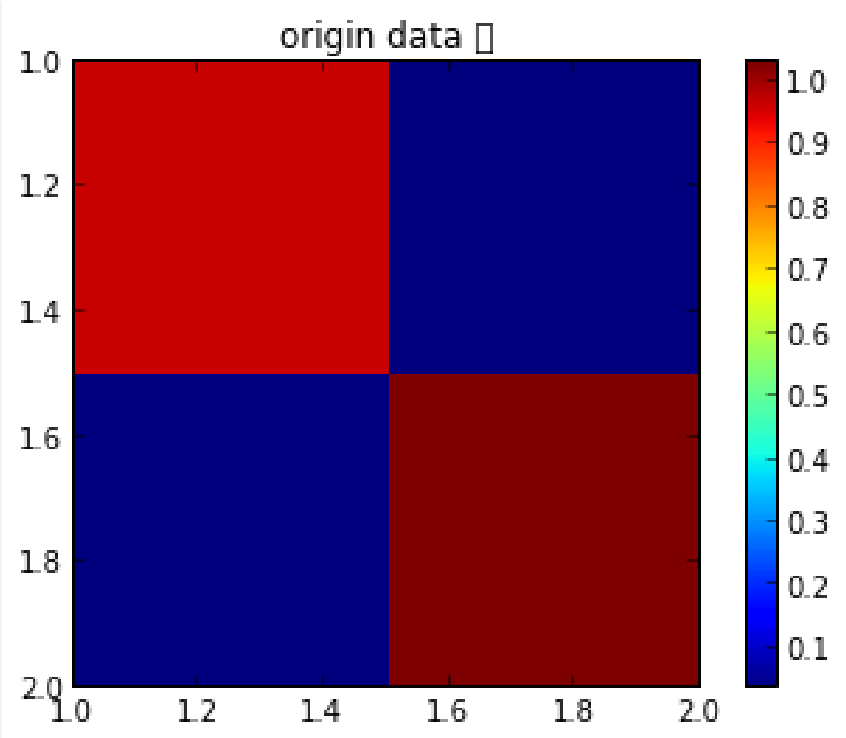




4 Comparation of Covariance Matrix

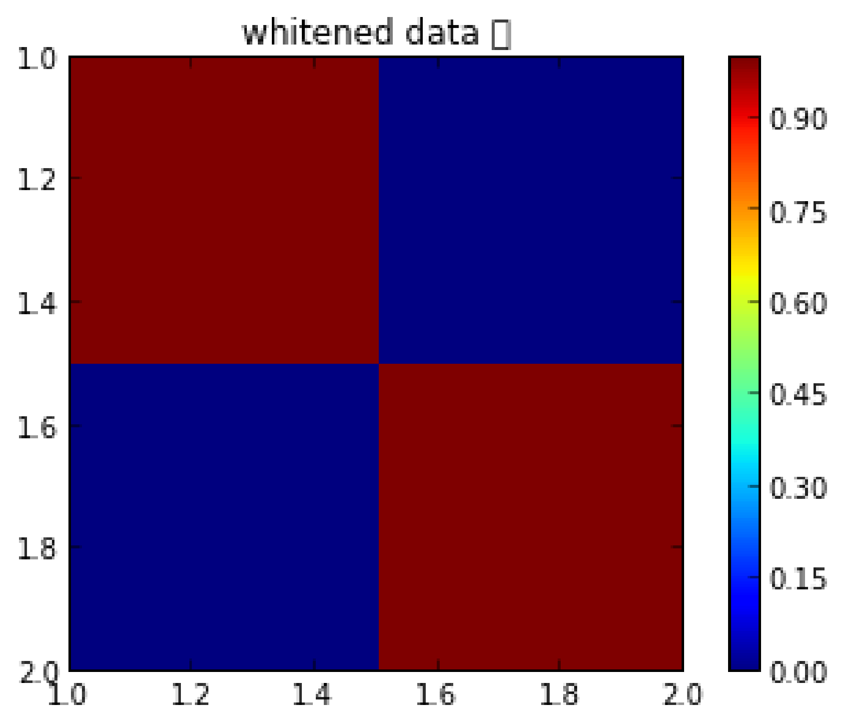
```
C1 = get_CoMatrix(data2b,2)
print matrix(C1)
plt.title('origin data ')
plt.imshow(C1,interpolation="nearest",extent=[1,2,2,1])
plt.colorbar()
plt.show()
```

```
[[ 0.96670278  0.03849033]
 [ 0.03849033  1.03129722]]
```



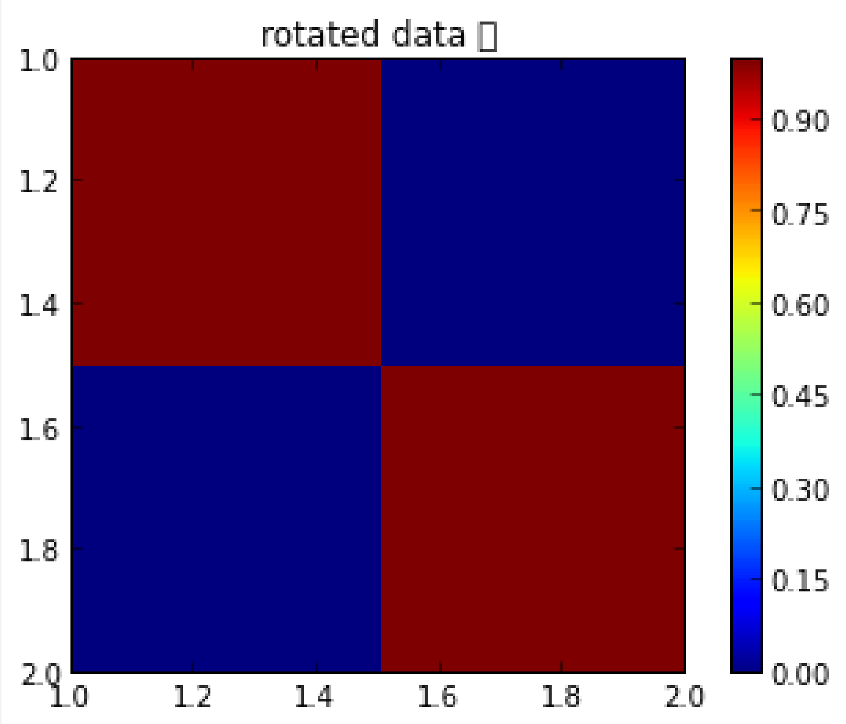

```
C2 = get_CoMatrix(data2b_whiten,2)
print matrix(C2)
plt.title('whitened data ')
plt.imshow(C2,interpolation="nearest",extent=[1,2,2,1])
plt.colorbar()
plt.show()
```

```
[[  1.00000000e+00  -2.29661125e-16]
 [ -2.29661125e-16   1.00000000e+00]]
```




```
C3 = get_CoMatrix(data2b_rotate,2)
print matrix(C3)
plt.title('rotated data ')
plt.imshow(C3,interpolation="nearest",extent=[1,2,2,1])
plt.colorbar()
plt.show()
```

```
[[  1.00000000e+00  -1.77809156e-17]
 [ -1.77809156e-17   1.00000000e+00]]
```



As we can see, the whitened dataset is uncorrelated, even if it is rotated.