HW 8 q2 answers

December 10, 2018

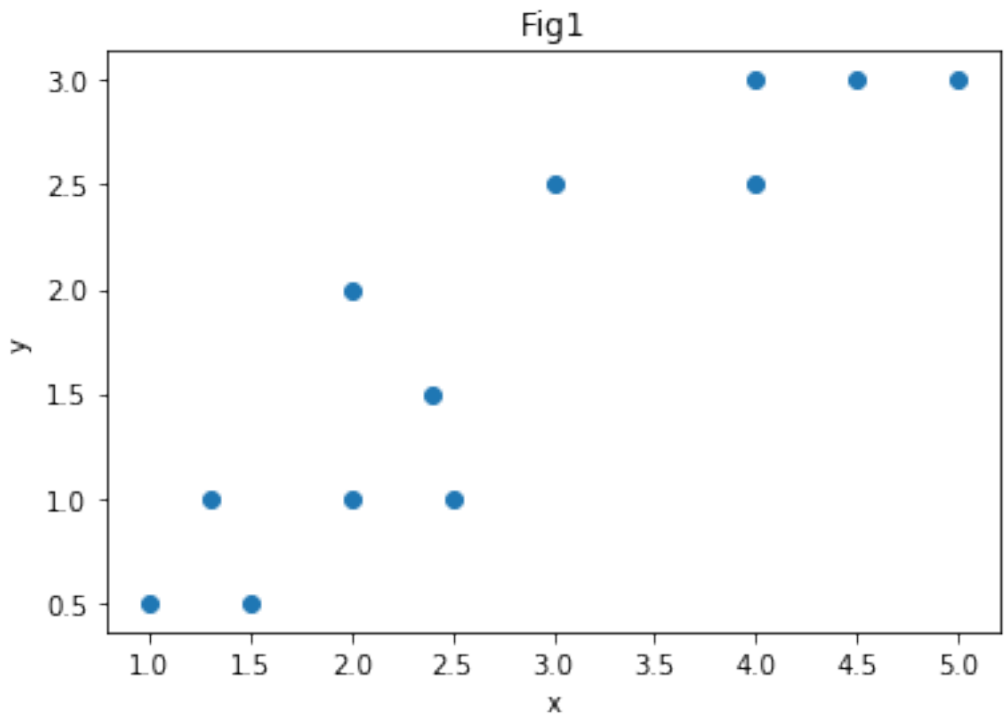
In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv("Data.csv")

#  https://pandas.pydata.org/pandas-docs/stable/10min.html
# https://matplotlib.org/index.html

x1 = df['x'].values
y1 = df['y'].values
df
```

Out[1]:

	x	y
0	1.0	0.5
1	1.5	0.5
2	1.3	1.0
3	2.0	1.0
4	2.5	1.0
5	2.4	1.5
6	2.0	2.0
7	3.0	2.5
8	4.0	2.5
9	4.0	3.0
10	4.5	3.0
11	5.0	3.0

In [2]:
```
# plot
plt.scatter(x1,y1)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Fig1')
plt.legend()
plt.show()
```

```
/anaconda/lib/python2.7/site-packages/matplotlib/axes/_axes.py:545: UserWarning: No labelled ob
  warnings.warn("No labelled objects found. "
```



1 1. Finish it without PCA packages (by hand)

In [3]:
```
# center the points

import numpy as np
x1_mean = np.mean(x1)
print('x1_mean = ', x1_mean)
y1_mean = np.mean(y1)
print('y1_mean = ', y1_mean)

x1_new = x1 - x1_mean
y1_new = y1 - y1_mean
df_new = pd.DataFrame({'x1_new':x1_new,'y1_new':y1_new})
df_new
```

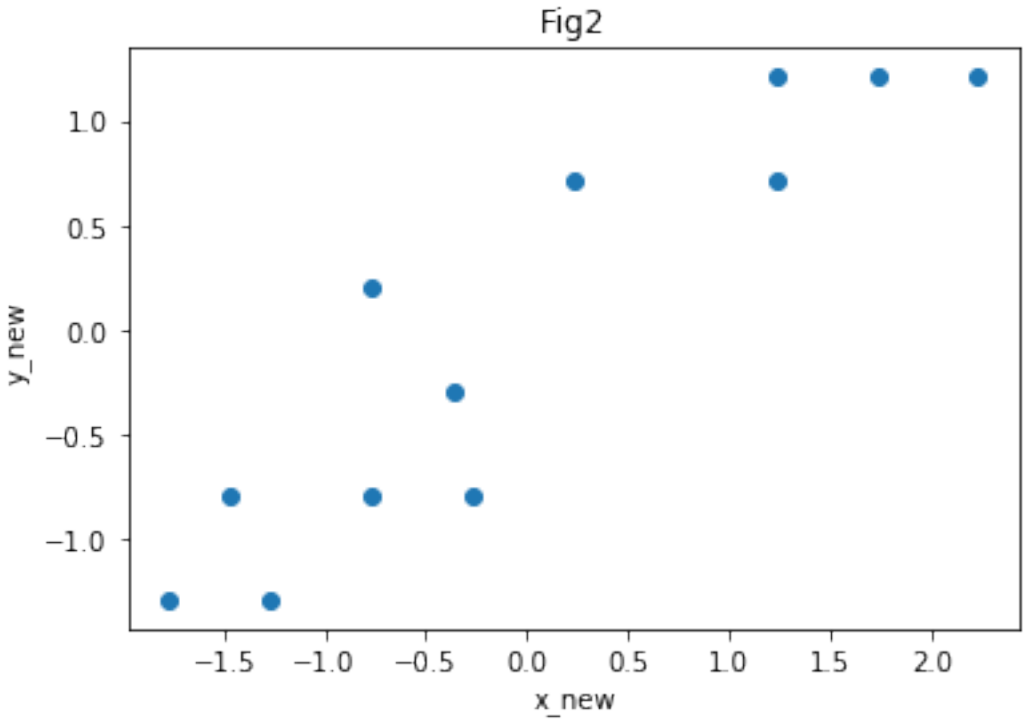
```
('x1_mean = ', 2.7666666666666671)
('y1_mean = ', 1.7916666666666667)
```

Out[3]:

	x1_new	y1_new
0	-1.766667	-1.291667
1	-1.266667	-1.291667
2	-1.466667	-0.791667

```
3  -0.766667 -0.791667
4  -0.266667 -0.791667
5  -0.366667 -0.291667
6  -0.766667  0.208333
7   0.233333  0.708333
8   1.233333  0.708333
9   1.233333  1.208333
10  1.733333  1.208333
11  2.233333  1.208333
```

In [4]:
```
plt.scatter(x1_new,y1_new)
plt.xlabel('x_new')
plt.ylabel('y_new')
plt.title('Fig2')
plt.legend()
plt.show()
```



In [5]:
```
# cov matrix
covmat = np.cov([x1_new,y1_new])
covmat
```

Out[5]:
```
array([[ 1.75878788,  1.1969697 ],
       [ 1.1969697 ,  0.97537879]])
```

```
In [6]: # The eigenvalues and eigenvectors
        eigenvalues, eigenvectors = np.linalg.eig(covmat)
        print(eigenvalues)
        print(eigenvectors)
```

```
[ 2.62651531  0.10765136]
[[ 0.80963474 -0.58693405]
 [ 0.58693405  0.80963474]]
```

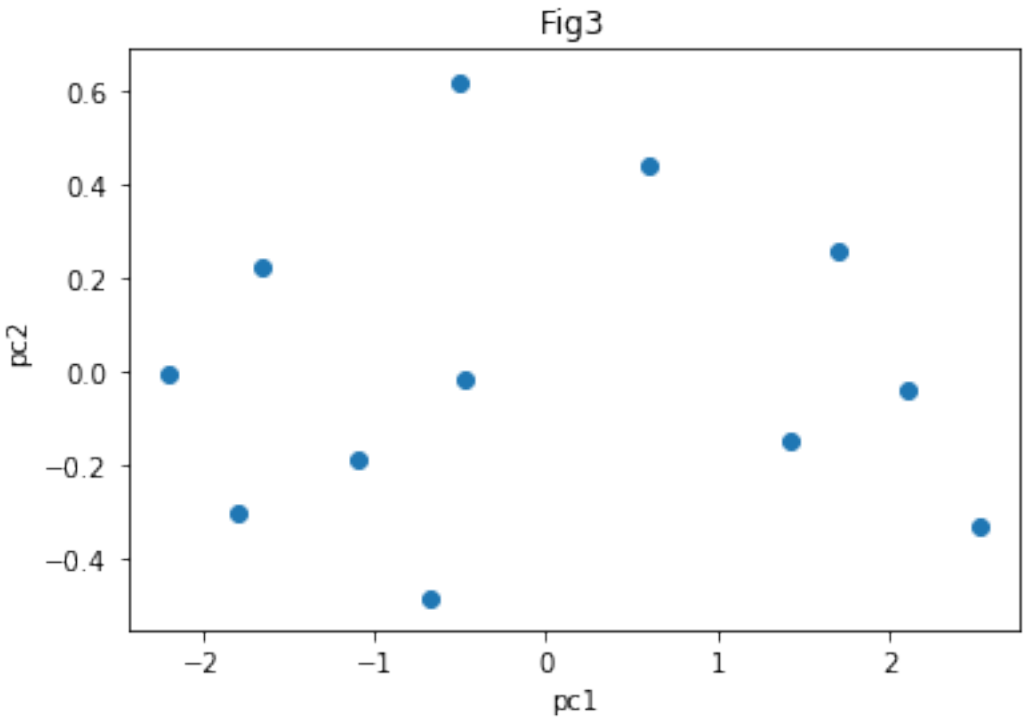
```
In [7]: W = eigenvectors
        df_new_matrix = np.transpose([x1_new,y1_new])
        np.mat(df_new_matrix)* np.mat(W)
```

```
Out[7]: matrix([[-2.18847786, -0.00886138],
                [-1.78366049, -0.30232841],
                [-1.65212042,  0.21987577],
                [-1.0853761 , -0.19097806],
                [-0.68055872, -0.48444509],
                [-0.46805517, -0.02093431],
                [-0.49844204,  0.61865668],
                [ 0.60465973,  0.43654   ],
                [ 1.41429447, -0.15039406],
                [ 1.7077615 ,  0.25442331],
                [ 2.11257887, -0.03904371],
                [ 2.51739624, -0.33251074]])
```

```
In [8]: # Loadings, here you have to use x1_new and Y1_new
        pc1 = x1_new * 0.80963474 + y1_new * (0.58693405)
        pc2 = x1_new * (-0.58693405) + y1_new * (0.80963474)
        print (pc1)
        print (pc2)
```

```
[-2.18847786 -1.78366049 -1.65212041 -1.08537609 -0.68055872 -0.46805517
 -0.49844204  0.60465972  1.41429446  1.70776149  2.11257886  2.51739623]
[-0.00886138 -0.30232841  0.21987577 -0.19097806 -0.48444509 -0.02093431
  0.61865668  0.43654    -0.15039405  0.25442332 -0.03904371 -0.33251073]
```

```
In [9]: plt.scatter(pc1,pc2)
        plt.xlabel('pc1')
        plt.ylabel('pc2')
        plt.title('Fig3')
        plt.legend()
        plt.show()
```



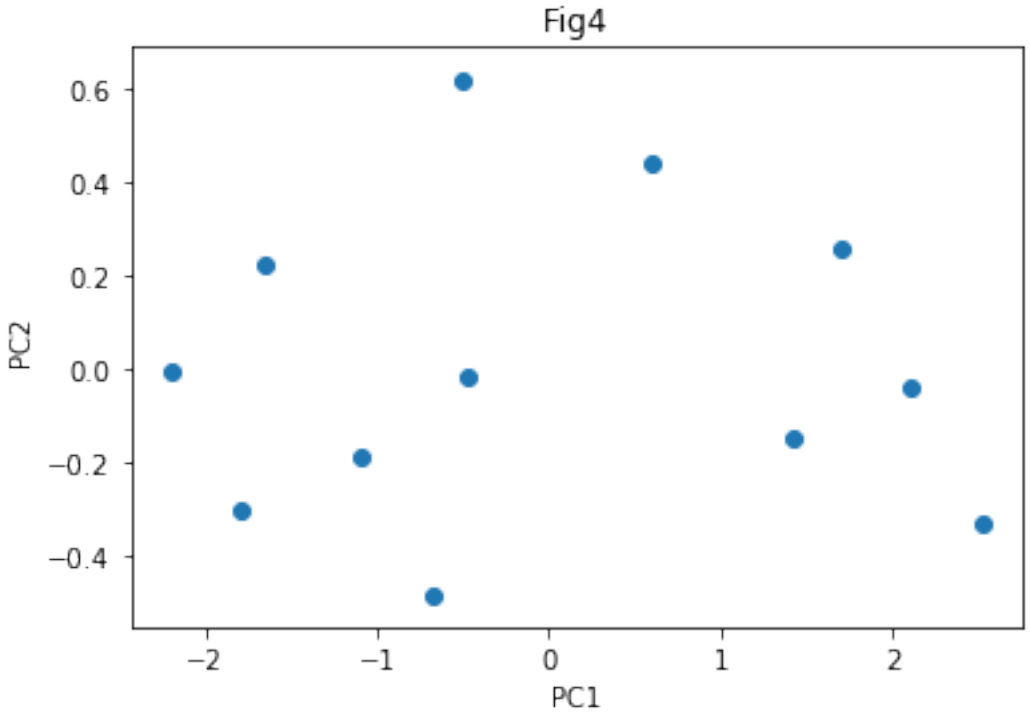

```
In [10]: # A*W = B, then A*W*W-1 = B*W-1, A = B*W-1
         # Thus,W-1 is the score
         from numpy.linalg import inv
         inv(W)

Out[10]: array([[ 0.80963474,  0.58693405],
                [-0.58693405,  0.80963474]])
```

2 Finish it by using package provided

```
In [11]: from sklearn.decomposition import PCA

         pca = PCA(n_components=2)
         principalComponents = pca.fit_transform(df)
         principalDf = pd.DataFrame(data = principalComponents
                     , columns = ['principal component 1', 'principal component 2'])
         x2 = principalDf['principal component 1'].values
         y2 = principalDf['principal component 2'].values
         principalDf

Out[11]:     principal component 1  principal component 2
         0              -2.188478              -0.008861
         1              -1.783660              -0.302328
```

2	-1.652120	0.219876
3	-1.085376	-0.190978
4	-0.680559	-0.484445
5	-0.468055	-0.020934
6	-0.498442	0.618657
7	0.604660	0.436540
8	1.414294	-0.150394
9	1.707761	0.254423
10	2.112579	-0.039044
11	2.517396	-0.332511

In [12]:
```
plt.scatter(x2,y2)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Fig4')
plt.legend()
plt.show()
```



In [13]:
```
# Percentage of variance explained by each of the selected components.
pca.explained_variance_ratio_
```

Out[13]:
```
array([ 0.96062736,  0.03937264])
```

In [14]:
```
# Principal axes in feature space, representing the directions of maximum variance in
# The components are sorted by explained_variance_.
```

```
Loadings = pca.components_
Loadings_table = pd.DataFrame(data = np.transpose(Loadings)
             , columns = ['PC1', 'PC2'], index = ['X','Y'])
Loadings_table
```

Out[14]:

	PC1	PC2
X	0.809635	-0.586934
Y	0.586934	0.809635

```
In [16]: # score
         from numpy.linalg import inv
         inv(np.transpose(Loadings))
```

```
Out[16]: array([[ 0.80963474,  0.58693405],
                [-0.58693405,  0.80963474]])
```

```
In [19]: # The amount of variance explained by each of the selected components.
         pca.explained_variance_
```

```
Out[19]: array([ 2.40763904,  0.09868041])
```

3 You can also use the score to calculate the PC1 and PC1 based on the x1 and y1, instead of x1_new and y1_new

```
In [22]: pc1_ori = pc1 = x1 * 0.80963474 + y1 * (0.58693405)
         pc2_ori = x1 * (-0.58693405) + y1 * (0.80963474)
```

```
In [23]: plt.scatter(pc1_ori,pc2_ori)
         plt.xlabel('pc1')
         plt.ylabel('pc2')
         plt.title('Fig5')
         plt.legend()
         plt.show()
```

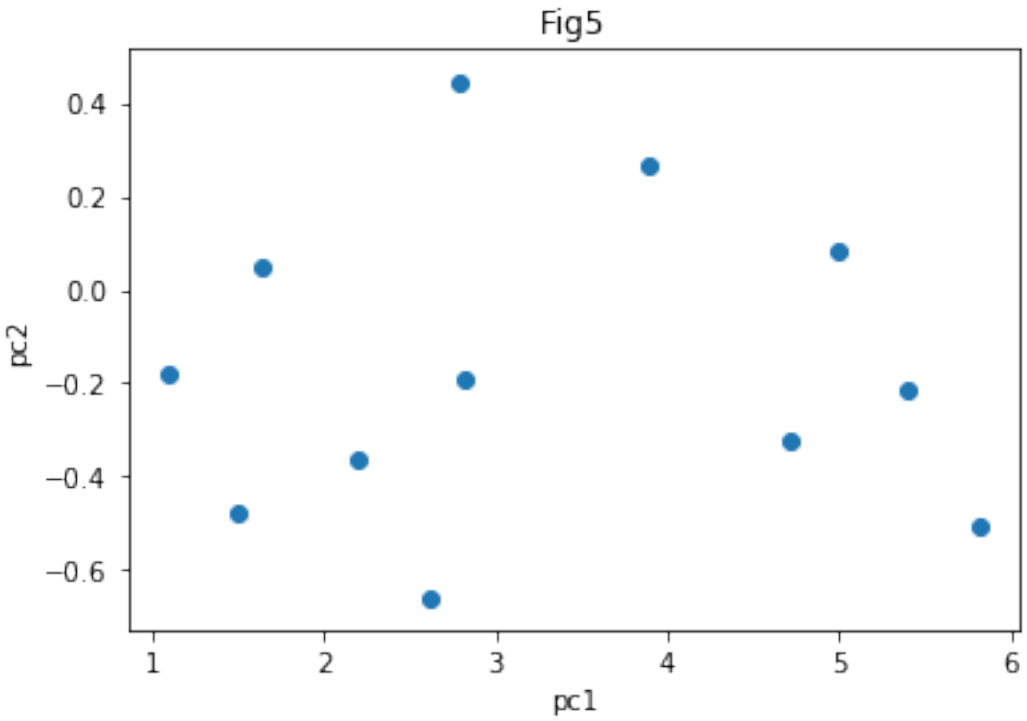


Fig5
0.4
0.2
0.0
−0.2
−0.4
−0.6
pc2
1
2
3
4
5
6
pc1