Sheet#1 PCA

(1)

ID	a1	a2	a3	a4
1	10	60	10	90
2	20	50	40	70
3	30	50	30	40
4	20	50	20	60
5	10	60	30	10

(a) 4 Dimensions.

(b) Integers

(c) $\sqrt{(30-10)^2+(50-60)^2+(30-10)^2+(40-90)^2}$ = 58.31

(d) $\sqrt{(20)^2+(50)^2+(40)^2+(70)^2}$ = 96.95

(e) $\frac{d^T b}{||a||\,||b||}$ = $8300 / (\sqrt{(20)^2+(50)^2+(40)^2+(70)^2} * \sqrt{(20)^2+(50)^2+(20)^2+(60)^2})$ = 1.03071442

(f) Yes

(g)

ID :1	$\frac{10-10}{90-10}$ = 0	$\frac{60-10}{90-10}$ = 0.625	$\frac{10-10}{90-10}$ = 0	$\frac{90-10}{90-10}$ = 1
ID :2	$\frac{20-10}{90-10}$ = 0.125	$\frac{50-10}{90-10}$ = 0.5	$\frac{40-10}{90-10}$ = 0.375	$\frac{70-10}{90-10}$ = 0.75
ID :3	$\frac{30-10}{90-10}$ = 0.25	$\frac{50-10}{90-10}$ = 0.5	$\frac{30-10}{90-10}$ = 0.25	$\frac{40-10}{90-10}$ = 0.375
ID :4	$\frac{20-10}{90-10}$ = 0.125	$\frac{50-10}{90-10}$ = 0.5	$\frac{20-10}{90-10}$ = 0.125	$\frac{60-10}{90-10}$ = 0.625
ID :5	$\frac{10-10}{90-10}$ = 0	$\frac{60-10}{90-10}$ = 0.625	$\frac{30-10}{90-10}$ = 0.25	$\frac{10-10}{90-10}$ = 0

(h)

* $\sqrt{(0.25-0)^2+(0.5-0.625)^2+(0.25-0)^2+(0.375-1)^2}$ = 0.72886
* $\sqrt{(0.125)^2+(0.5)^2+(0.375)^2+(0.75)^2}$ = 0.98425
* $0.78125 / (\sqrt{(0.125)^2+(0.5)^2+(0.375)^2+(0.75)^2} * \sqrt{(0.125)^2+(0.5)^2+(0.125)^2+(0.625)^2})$ = 0.968

Sheet#1

February 21, 2019

```
In [1]: import numpy as np
        import pandas as pd
        from scipy.spatial import distance
        import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D
```

```
In [2]: #-------------------------------2.A--------------------------------------#
        a = np.array([[10,60,10,90],[20,50,40,70],[30,50,30,40],[20,50,20,60],[10,60,30,10]])
        print(str(np.linalg.norm(a,axis = 1)))
```

```
[109.08712115  96.95359715  76.81145748  83.06623863  68.556546  ]
```

```
In [3]: #-------------------------------2.B--------------------------------------#
        result = np.empty([5,5],dtype=float)
        for i in range (0,len(a)):
            for j in range(0,len(a)):
                result[i][j] = np.round(np.dot(a[i],a[j]) / (np.linalg.norm(a[i]) * np.linalg.r
        print(result)
```

```
[[1.   0.94 0.86 0.97 0.66]
 [0.94 1.   0.95 0.98 0.77]
 [0.86 0.95 1.   0.96 0.87]
 [0.97 0.98 0.96 1.   0.77]
 [0.66 0.77 0.87 0.77 1.  ]]
```

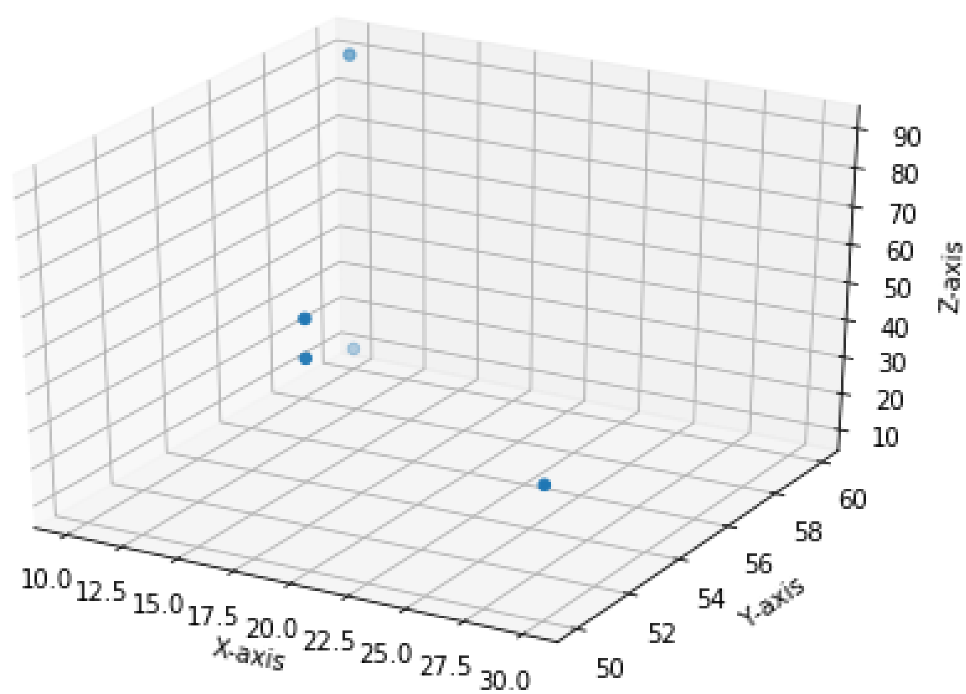
```
In [4]: #-------------------------------2.C--------------------------------------#
        result = np.empty([5,5],dtype = float)
        for i in range(0,len(a)):
            for j in range(0,len(a)):
                result[i][j] = np.round(distance.euclidean(a[i],a[j]),2)
        print(result)
```

```
[[ 0.   38.73 58.31 34.64 82.46]
 [38.73  0.   33.17 22.36 62.45]
 [58.31 33.17  0.   24.49 37.42]
 [34.64 22.36 24.49  0.   52.92]
 [82.46 62.45 37.42 52.92  0.  ]]
```

```
In [5]: #-------------------------------3.A--------------------------------------#
        a = np.array([[10,60,90],[20,50,70],[30,50,40],[20,50,60],[10,60,10]])
        print(a)
```

```
[[10 60 90]
 [20 50 70]
 [30 50 40]
 [20 50 60]
 [10 60 10]]
```

```
In [6]: #-------------------------------3.B--------------------------------------#
        fig = plt.figure()
        ax = Axes3D(fig)
        ax.scatter([a[i][0] for i in range(0,len(a))], [a[i][1] for i in range(0,len(a))], [a[i
        ax.set_xlabel('X-axis')
        ax.set_ylabel('Y-axis')
        ax.set_zlabel('Z-axis')
        plt.show()
```




```
In [7]: #-------------------------------3.C--------------------------------------#
        mean = np.mean(a,axis=0)
        print(mean.reshape(3,1))
```

```
[[18.]
 [54.]
 [54.]]
```

```
In [8]: #------------------------------3.D-------------------------------------#
        centered = a-mean
        print (centered)
```

```
[[ -8.   6.  36.]
 [  2.  -4.  16.]
 [ 12.  -4. -14.]
 [  2.  -4.   6.]
 [ -8.   6. -44.]]
```

```
In [9]: #------------------------------3.E-------------------------------------#
        covariance = np.cov(centered.T)
        print(covariance)
```

```
[[ 70. -40. -15.]
 [-40.  30. -20.]
 [-15. -20. 930.]]
```

```
In [10]: #------------------------------3.F-------------------------------------#
         values, vectors = np.linalg.eigh(covariance)
         print(np.round(vectors,2))
         values = np.diag(values)
         print(np.round(values,2))
```

```
[[ 0.53  0.85 -0.02]
 [ 0.85 -0.53 -0.02]
 [ 0.03  0.    1.  ]]
[[  4.61   0.     0.  ]
 [  0.    94.72   0.  ]
 [  0.     0.   930.68]]
```

```
In [11]: #------------------------------3.G-------------------------------------#
```

```
In [12]: #------------------------------3.H-------------------------------------#
         var = values[2][2] / np.trace(values)
         print("Yes: " + str(np.round(var,3)))
```

```
Yes: 0.904
```

```
In [13]: #------------------------------3.I-------------------------------------#
         proj = vectors[:,1:]
         print(proj)
```

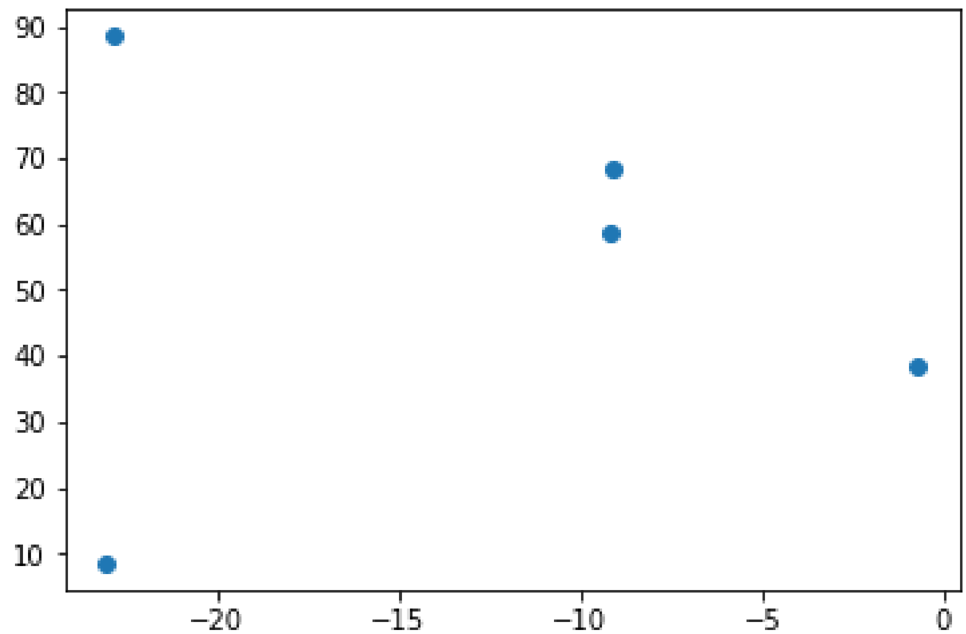
```
[[ 0.85025739 -0.01642407]
 [-0.5263604  -0.02146802]
 [ 0.00266574  0.99963462]]
```

In [14]:
```
#------------------------------3.J---------------------------------------#
result = np.dot(a,proj)
print(result)
```

```
[[-22.83913329  88.51479397]
 [ -9.12627026  68.57254111]
 [ -0.70366863  38.41926184]
 [ -9.15292768  58.57619491]
 [-23.05239261   8.54402434]]
```

In [15]:
```
#------------------------------3.K---------------------------------------#
plt.scatter(result[:,0],result[:,1])
```

Out[15]: <matplotlib.collections.PathCollection at 0x7faa36f2bf60>



In [16]:
```
#------------------------------4.A---------------------------------------#
a = np.array([[0.5,4.5,2.5],[2.2,1.5,0.1], [3.9,3.5,1.1],[2.1,1.9,4.9],[0.5,3.2,1.2]])
b = a - np.mean(a, axis = 0)
cov = np.cov(b.T)
print(cov)
```

```
[[ 2.008  -0.5185 -0.3455]
 [-0.5185  1.492  -0.054 ]
 [-0.3455 -0.054   3.428 ]]
```

```
In [17]: #------------------------------4.B--------------------------------------#
         tot = np.trace(cov)
         #tot = (b * 0.505) + b + b/0.647
         b = tot / (0.505 + 1 + 1.545)
         a = b * 0.505
         c = b / 0.647
         print([a,b,c])
```

```
[1.1470950819672134, 2.271475409836066, 3.510781158942915]
```

```
In [18]: #------------------------------4.B--------------------------------------#
         b_var = b / sum([a,b,c])
         c_var = c / sum([a,b,c])
         print('explained_variance b:',b_var)
         print('explained_variance c:',c_var)
```

```
explained_variance b: 0.3278048978206294
explained_variance c: 0.5066536287799527
```