Report

119010259 沈嘉佑
120090414 费祥

1. Select the cities and get the distance matrix D

We selected 10 cities in range of Asia and label them as follows:

1. Shanghai (SH)
2. Beijing (BJ)
3. Tianjin (TJ)
4. Guangzhou (GZ)
5. Wuhan (WH)
6. Changsha (CS)
7. Chengdu (CD)
8. Singapore (SGP)
9. Tokyo (TKY)
10. Mumbai (MB)

After deciding the cities, we searched for the distances between each two cities and made the distance matrix D:

	SH	BJ	TJ	GZ	WH	CS	CD	SGP	TKY	MB
SH	0	1068	962	1212	690	1089	1658	3809	1755	5040
BJ	1068	0	108	1890	1055	1339	1516	4477	2092	4752
TJ	962	108	0	1820	988	1277	1519	4417	2016	4796
GZ	1212	1890	1820	0	836	566	1238	2631	2900	4205
WH	690	1055	988	836	0	293	976	3439	2421	4352
CS	1089	1339	1277	566	293	0	905	3143	2643	4195
CD	1658	1516	1519	1238	976	905	0	3264	3341	3394
SGP	3809	4477	4417	2631	3439	3143	3264	0	5315	3907
TKY	1755	2092	2016	2900	2421	2643	3341	5315	0	6730
MB	5040	4752	4796	4205	4352	4195	3394	3907	6730	0

2. Double centering and derive the coordinate matrix X

After getting the distance matrix D, we can derive the matrix B by double centering ($B = -0.5JD^{(2)}J$):

```
summation = np.sum(D2,axis=1)/D2.shape[0]
Di = np.repeat(summation[:,np.newaxis],D2.shape[0],axis=1)
Dj = np.repeat(summation[np.newaxis,:],D2.shape[0],axis=0)
Dij = np.sum(D2)/((D2.shape[0])**2)*np.ones([D2.shape[0],D2.shape[0]])
B = (Di+Dj-D2-Dij)/2
B = B.astype(np.float64)

print(B)
```

```
[[ 1.17177761e+06  9.01164810e+05  9.49225610e+05  8.64767600e+04
   4.33445260e+05  4.68481600e+04 -5.49807590e+05 -1.78642804e+06
   2.95905201e+06 -4.21175459e+06]
 [ 9.01164810e+05  1.77117601e+06  1.70581481e+06 -6.65402040e+05
   4.14681960e+05  4.30473600e+04 -2.47543900e+04 -4.25425284e+06
   2.61053171e+06 -2.50200739e+06]
 [ 9.49225610e+05  1.70581481e+06  1.65211761e+06 -5.95081240e+05
   4.23593260e+05  6.46141600e+04 -8.88360900e+04 -4.04696204e+06
   2.70710651e+06 -2.77159259e+06]
 [ 8.64767600e+04 -6.65402040e+05 -5.95081240e+05  4.70119910e+05
  -2.87815900e+04  1.28801810e+05 -2.92476440e+05  1.65590311e+06
  -5.67643400e+04 -7.02795940e+05]
 [ 4.33445260e+05  4.14681960e+05  4.23593260e+05 -2.87815900e+04
   1.71212910e+05  9.66018100e+04 -1.51895940e+05 -9.45830390e+05
   1.06816166e+06 -1.48118894e+06]
 [ 4.68481600e+04  4.30473600e+04  6.46141600e+04  1.28801810e+05
   9.66018100e+04  1.07839710e+05 -1.16807040e+05 -3.38099000e+03
   4.74371060e+05 -8.41936040e+05]
 [-5.49807590e+05 -2.47543900e+04 -8.88360900e+04 -2.92476440e+05
  -1.51895940e+05 -1.16807040e+05  4.77571210e+05 -2.06138740e+05
  -1.42917919e+06  2.38232421e+06]
 [-1.78642804e+06 -4.25425284e+06 -4.04696204e+06  1.65590311e+06
  -9.45830390e+05 -3.38099000e+03 -2.06138740e+05  9.76384731e+06
  -5.32951314e+06  5.15275576e+06]
 [ 2.95905201e+06  2.61053171e+06  2.70710651e+06 -5.67643400e+04
```

show more (open the raw output data in a text editor) ...

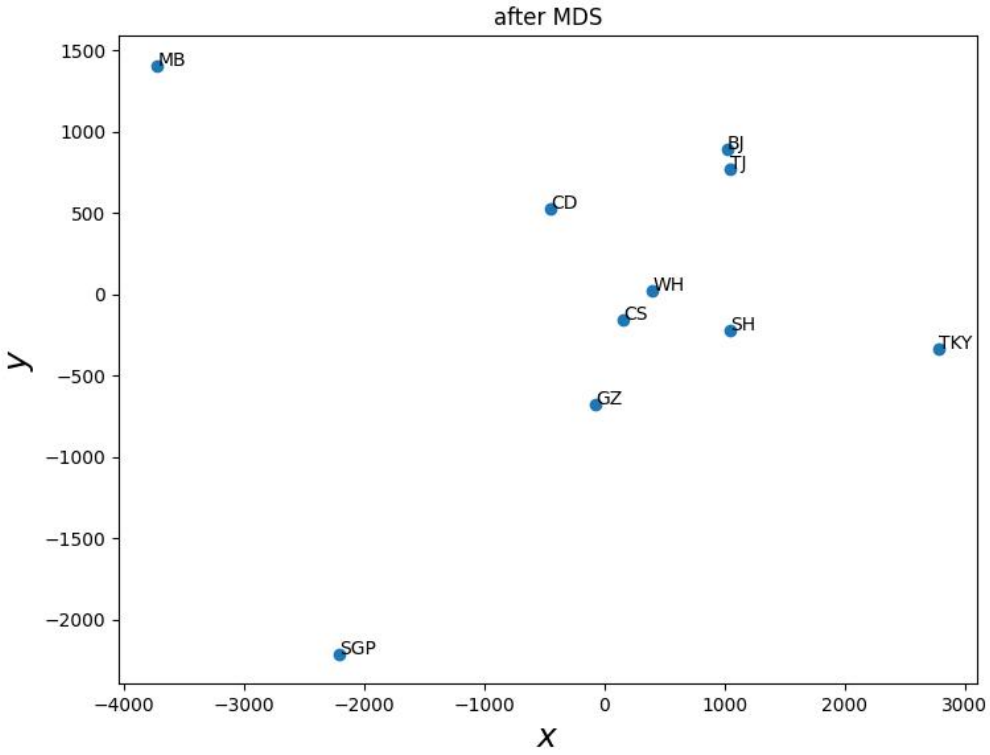
Then we decomposed the symmetric matrix B and get the eigenvalue and eigenvectors to derive the coordinates X by the formula: $X = E_m \Lambda_m^{1/2}$. We chose m=2 to select the two largest eigenvalues and eigenvectors to calculate the coordinates, and the result is as follows:

```
eigenvalues, eigenvectors = np.linalg.eigh(B)
eigen_sort = np.argsort(-eigenvalues)
eigenvalues = eigenvalues[eigen_sort]
eigenvectors = eigenvectors[:,eigen_sort]
Bez = np.diag(eigenvalues[0:2])
Bvz = eigenvectors[:,0:2]
Z = np.dot(np.sqrt(Bez), Bvz.T).T
print(Z)
```

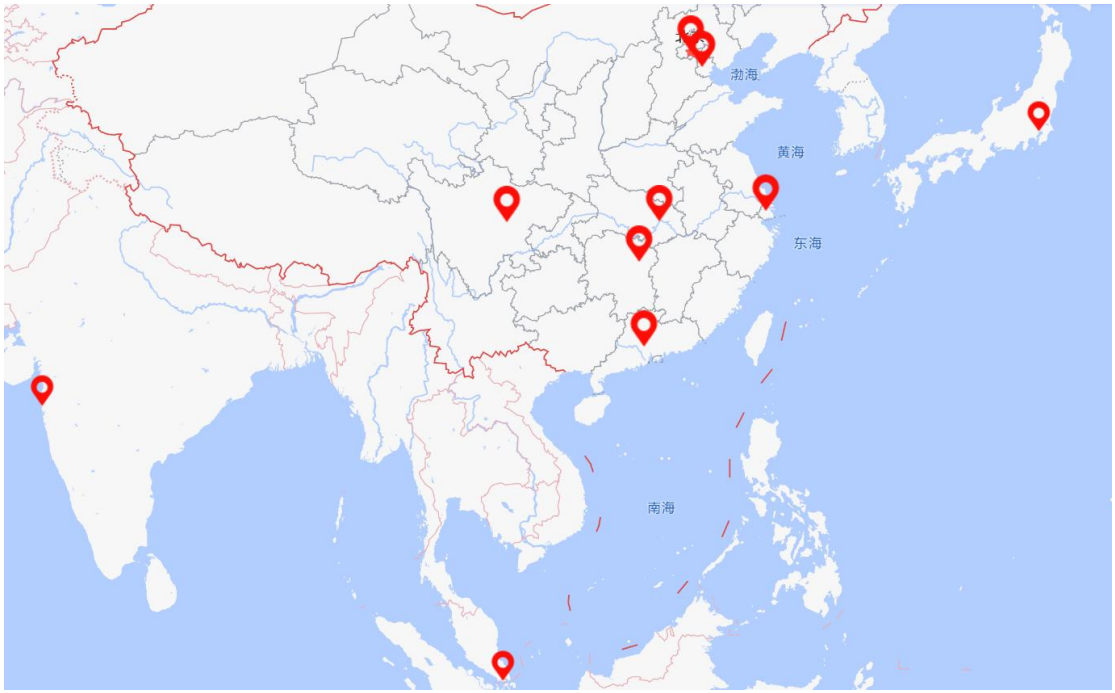
```
[[-1044.23962191   223.11550742]
 [-1022.20662903  -886.90559735]
 [-1047.83644405  -770.81718238]
 [   76.8852728    675.15880781]
 [ -400.51478238   -19.02429568]
 [ -156.86393866   155.08826064]
 [  443.71301699  -521.65852478]
 [ 2209.03915027  2212.64155266]
 [-2778.86134357   338.54243079]
 [ 3720.88531954 -1406.14095912]]
```

3. Draw the map and compare with the actual map

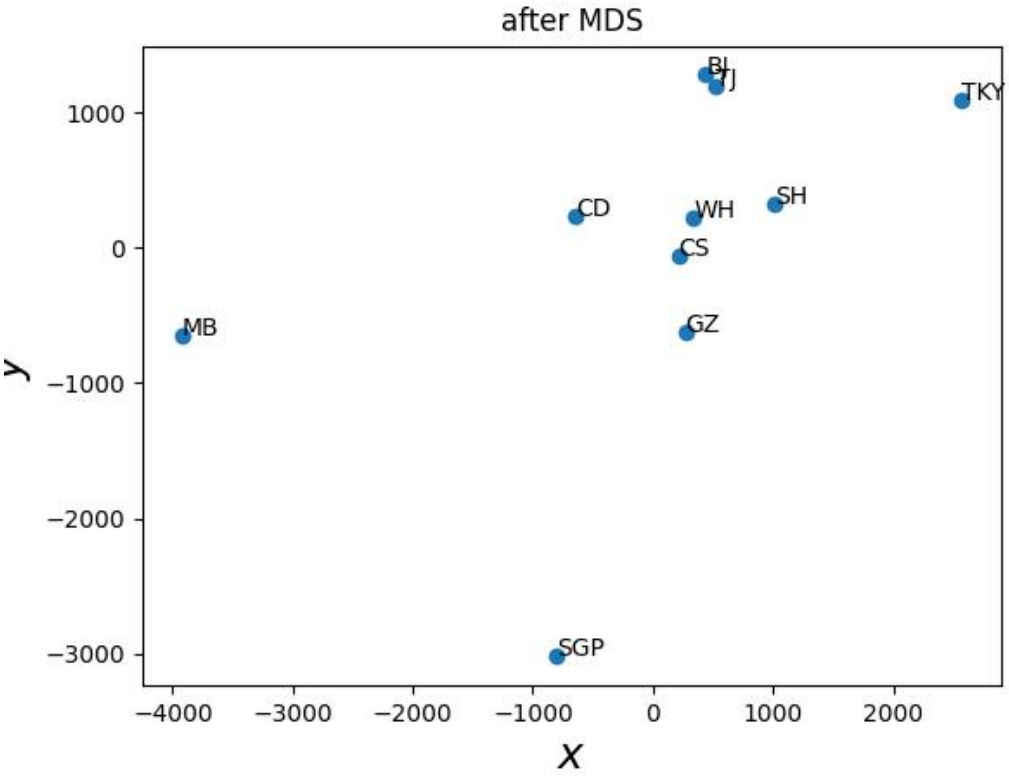
We used the coordinates we derived to draw the scatter plot of the 10 cities:



It doesn't match the actual map very well and we found that there seems to be a revolving relationship between the two maps:



Therefore, we use a rotation matrix of 30 degrees to adapt the coordinates to the actual map, and the final plot, which highly resembles the map, is as follows:



```
rot = [[3**(1/2)/2,1/2],[-1/2,3**(1/2)/2]]
rot = np.array(rot)
X = np.dot(X,rot)
print(X)
```

✓ 0.8s

```
[[-1015.89579392  -328.89611355]
 [ -441.80410998 -1279.18609258]
 [ -522.04438837 -1191.46548364]
 [ -270.99480448   623.14731555]
 [ -337.34382829  -216.73291454]
 [ -213.39228614    55.87840421]
 [  645.09600709  -229.91302606]
 [  806.76324576  3020.72336921]
 [-2575.83573252 -1096.24432646]
 [ 3925.45169085   642.68886787]]
```