Math 381 A
Sheng Yu

Homework #7 Writeup: Multidimensional Scaling

I. Introduction:

Multidimensional scaling (MDS) is used to create a low-dimensional model of a set of objects that maintains as much of the inter-object distance relationship as possible. It takes a set of pair-wise distances for a set of objects and creates a set of points in a low-dimensional Euclidean space, with each point corresponding to a different object in the set, so that the pair-wise distances between the points are as similar as possible to the corresponding pair-wise distances between the objects. With that in mind, we will be investigating a set of cities by creating the distance matrix and the different dimensional models with MDS. Also, we will be looking at the eigenvalues, goodness of fit values, and plots to see how good the models are.

II. Distance Matrix:

We will be using two different sets of cities in here to apply MDS. The first set of cities are within 500 km of each other; while the second set of cities are very far apart from each other that "span the globe." The distances for all the city pairs in the matrix below are from directly search on Google, and they are the actual distance from one another with different vehicle choices such as car, airplane, or steamship.

(1) First set (all cities nearby Beijing, and all distances are road distance in km):

Index & City Name		1	2	3	4	5	6	7
		Beijing	Shijiazhuang	Zhangjiakou	Taiyuan	Baoding	Datong	Cangzhou
1	Beijing	0	294	192	494	161	346	208
2	Shijiazhuang	294	0	377	226	143	371	229
3	Zhangjiakou	192	377	0	463	303	200	404
4	Taiyuan	494	226	463	0	342	279	445
5	Baoding	161	143	303	342	0	296	155
6	Datong	346	371	200	279	296	0	468
7	Cangzhou	208	229	404	445	155	468	0

(2) Second set (capital cities of countries around the world, air distance in km):

Index & City Name		1	2	3	4	5	6	7
		Beijing	Washington D.C.	Pairs	Berlin	Canberra	Brasilia	New Delhi
1	Beijing	0	11120	8217	7359	9010	16937	3782
2	Washington D. C.	11120	0	6214	6741	15936	6774	12052
3	Pairs	8217	6214	0	1049	16904	8729	6589
4	Berlin	7359	6741	1049	0	16066	9483	5783
5	Canberra	9010	15936	16904	16066	0	14064	10356
6	Brasilia	16937	6774	8729	9483	14064	0	14242
7	New Delhi	3782	12052	6589	5783	10356	14242	0

III. MDS:

(1) First set:

We will begin with our first set of data, and we will be using the index number instead of city names considering they are quite messy and long. In R, we can write the following to generate the distance matrix:

```
# First set:
# Generate Distance Matrix:

dist_mx1 <- matrix(c(0, 294, 192, 494, 161, 346, 208,
                     294, 0, 377, 226, 143, 371, 229,
                     192, 377, 0,  463, 303, 200, 404,
                     494, 226, 463, 0, 342, 279, 445,
                     161, 143, 303, 342, 0, 296, 155,
                     346, 371, 200, 279, 296, 0, 468,
                     208, 229, 404, 445, 155, 468, 0),
                     nrow = 7, ncol = 7);
```

1-Dimensional Model:

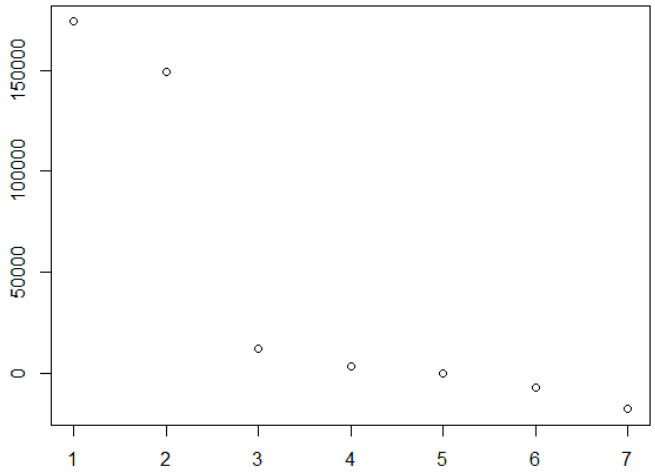
Then, we can apply MDS with the distance matrix with 1 dimension:

```
# Apply MDS with 1-D:

model1_1 <- cmdscale(dist_mx1, k=1, eig=TRUE);
```

We can get the following values:

```
> model1_1
$points
         [,1]
[1,]  186.69766
[2,]  -29.15537
[3,]   44.42906
[4,] -276.48502
[5,]   58.61907
[6,] -159.83473
[7,]  175.72934

$eig
[1]  1.739881e+05  1.491464e+05  1.199777e+04  3.360465e+03 -1.818989e-11 -7.314426e+03
[7] -1.791229e+04

$x
NULL

$ac
[1] 0

$GOF
[1] 0.4783580           0.5140084
```

Note that we have some eigenvalues being negative, that indicates that the distances are non-Euclidean, which means that no Euclidean model will fit perfectly. We can plot the eigenvalues:

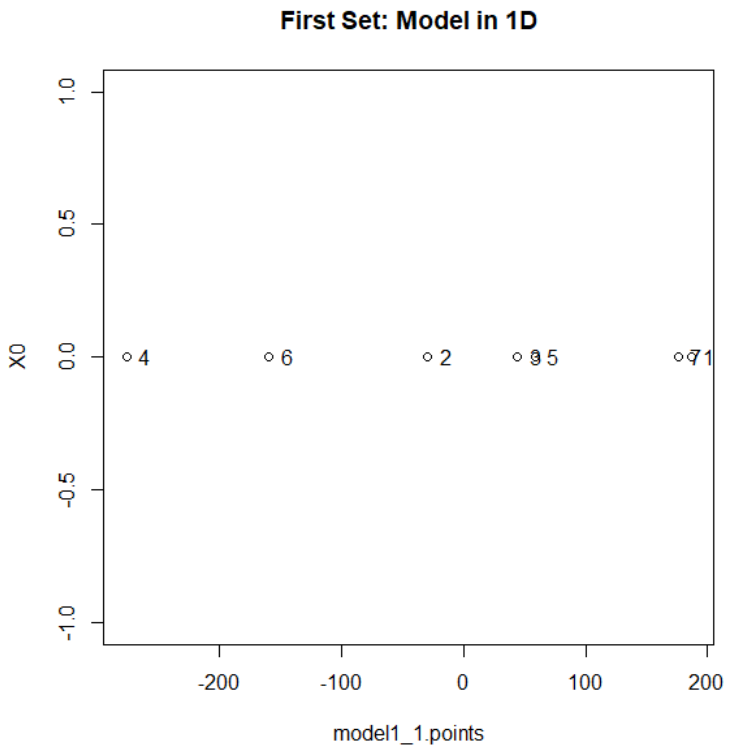


We can see both from the eigenvalues and the plot that the first two eigenvalues are noticeable larger than the rest. This suggests that a two-dimensional model will capture some aspects of the original data. However, the third eigenvalue is very small, that means a two-dimensional model will not be perfect.

For the goodness of fit (GOF), there are two different values. This is because some of our eigenvalues are negative. But we can still get some information from the two GOF values: 0.4783580 and 0.5140084. GOF is a value between 0 and 1, a value of 1 indicates a perfect fit whereas a zero is a terrible fit. So, our GOF is somewhere in the middle. We will see more about GOF values as we apply MDS to the distance matrix with 2-dimension and 3-dimension.

We now can plot our model in 1D:

```
firstset_oneD <- data.frame(model1_1$points, 0);
plot(firstset_oneD, main="First Set: Model in 1D");
mylabels <- c(1, 2, 3, 4, 5, 6, 7)
text(firstset_oneD, labels=mylabels, pos=4)
```



From the plot we see that the labels for 3 and 5, 1 and 7 are very close to each other, so this is not a good model we are looking for.

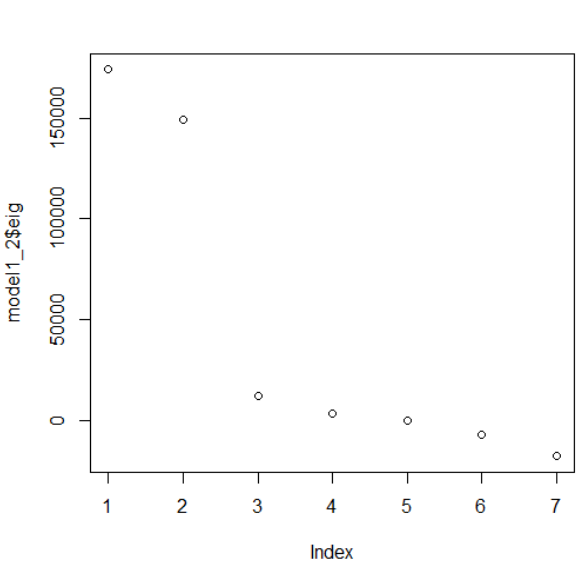
2-Dimensional Model:

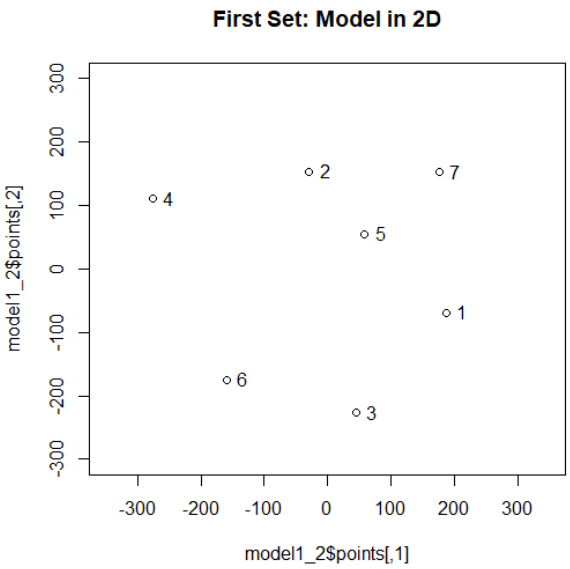
Now, let us move to 2-D model. With similar code, we can generate the 2-D model, plot the eigenvalues, and plot the model itself:

```
model1_2 <- cmdscale(dist_mx1, k=2, eig=TRUE);
plot(model1_2$eig);
plot(model1_2$points,asp=1, xlim=c(-300,300), ylim=c(-300,300), main="First Set: Model in 2D")
text(model1_2$points,labels=mylabels,pos=4)
```

We can get:

```
> model1_2
$points
         [,1]      [,2]
[1,]  186.69766  -69.33004
[2,]  -29.15537  153.70073
[3,]   44.42906 -226.84893
[4,] -276.48502  110.48666
[5,]   58.61907   55.28120
[6,] -159.83473 -175.53633
[7,]  175.72934  152.24672

$eig
[1]  1.739881e+05  1.491464e+05  1.199777e+04  3.360465e+03 -1.818989e-11
[6] -7.314426e+03 -1.791229e+04

$x
NULL

$ac
[1] 0

$GOF
[1] 0.8884169           0.9546276
```





We can see the eigenvalues and plot looks the same which is expected. The GOF in here becomes 0.8884169 and 0.9546276 which is very closer to 1 comparing with the 1D model. This indicates the GOF has improved. This can somehow show in the model plot as well as we can see that the points are more separated and there is no overlap.

To better understand how good this 2D model is, we can compare the model distances to our original distances, we can use R, and get:

```
> model_dist <- as.matrix(dist(model1_2$points))
> model_dist
          1        2        3        4        5        6        7
1   0.0000 310.3792 212.2559 496.8624 178.6955 362.4424 221.8481
2 310.3792   0.0000 387.5986 251.0765 131.8740 354.2233 204.8899
3 212.2559 387.5986   0.0000 465.5976 282.4867 210.6103 401.1898
4 496.8624 251.0765 465.5976   0.0000 339.6210 308.8955 454.1384
5 178.6955 131.8740 282.4867 339.6210   0.0000 317.8031 152.0432
6 362.4424 354.2233 210.6103 308.8955 317.8031   0.0000 469.0895
7 221.8481 204.8899 401.1898 454.1384 152.0432 469.0895   0.0000
```

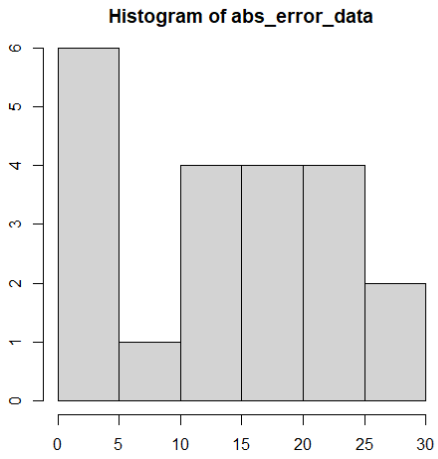
We can subtract the original distance matrix to make an easy comparison between the two:

```
> error_matrix <- dist_mx1 - model_dist
> abs_error_matrix <- abs(error_matrix)
> largest_error <- max(abs_error_matrix)
> mean_abs_error <- mean(abs_error_matrix)
> largest_error <- max(abs_error_matrix)
> error_matrix
           1          2          3          4          5          6          7
1   0.000000 -16.37921 -20.255865  -2.862397 -17.695509 -16.44237 -13.848071
2 -16.379209   0.00000 -10.598640 -25.076509  11.125982  16.77672  24.110134
3 -20.255865 -10.59864   0.000000  -2.597622  20.513253 -10.61025   2.810179
4  -2.862397 -25.07651  -2.597622   0.000000   2.379037 -29.89551  -9.138450
5 -17.695509  11.12598  20.513253   2.379037   0.000000 -21.80307   2.956825
6 -16.442366  16.77672 -10.610252 -29.895515 -21.803071   0.00000  -1.089510
7 -13.848071  24.11013   2.810179  -9.138450   2.956825  -1.08951   0.000000
> largest_error
[1] 29.89551
> mean_abs_error
[1] 11.38633
```

We can see that the largest absolute error value is 29.8955 which corresponding to city 4 (Taiyuan) and city 6 (Datong). The average absolute error is 11.38633 which is acceptable because an error of 11.38633 km when measuring city distances is normal.

We can make a histogram of the absolute values of the errors:

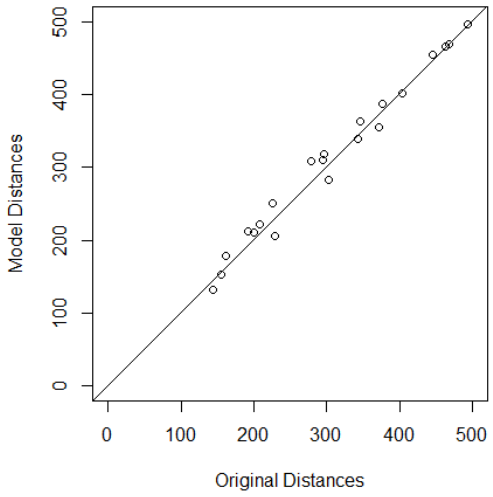
```
# abs_error_data <- abs(as.dist(dist_mx1) - dist(model1_2$points))
# hist(abs_error_data);
```



We see most of the errors are in the range within 5, with some larger errors in between 15 to 30.

We can then make a scatter plot of model distances vs original distances:

```
plot(as.dist(dist_mx1),dist(model1_2$points),asp=1,xlim=c(0,500),ylim=c(0,500), xlab="Original Distances", ylab="Model Distances")
abline(0,1)
```



We can see that the scatter plot overall is pretty good, there is no super large discrepancies anywhere on the plot. Considering the GOF, average absolute error, and the plot above, this demonstrates that the 2D model is much better than the 1D.

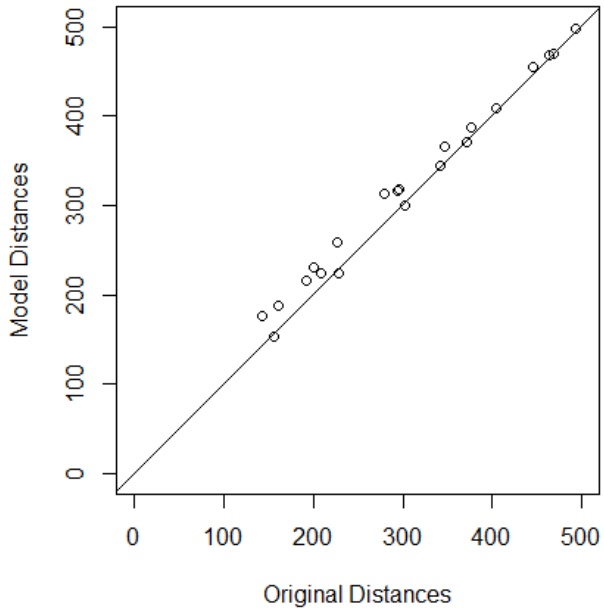
Finally, let us move to 3D, and see if things can be better.

3-Dimensional Model:

With similar code in R, we get:

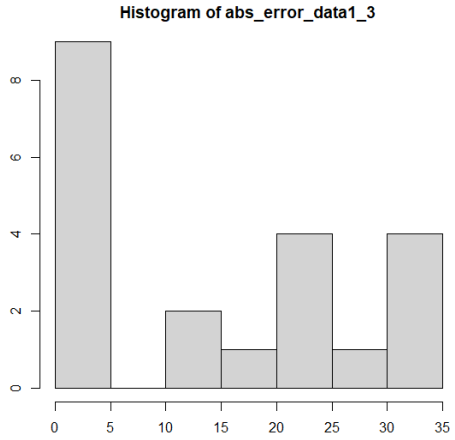
```
> model1_3 <- cmdscale(dist_mx1, k=3, eig=TRUE);
> model1_3
$points
         [,1]      [,2]      [,3]
[1,]  186.69766  -69.33004   6.197404
[2,]  -29.15537  153.70073  65.152979
[3,]   44.42906 -226.84893  48.288801
[4,] -276.48502  110.48666   3.288534
[5,]   58.61907   55.28120 -51.465354
[6,] -159.83473 -175.53633 -44.942857
[7,]  175.72934  152.24672 -26.519507

$eig
[1]  1.739881e+05  1.491464e+05  1.199777e+04  3.360465e+03 -1.818989e-11 -7.314426e+03
[7] -1.791229e+04

$x
NULL

$ac
[1] 0

$GOF
[1] 0.9214032           0.9900723
```

With the similar code as 2D model, we can get a scatter plot:



We get 0.9214032 and 0.9900723 as GOF value which is better than the 2D model. But only by looking at the scatter plot, there is not much of a different between the 2D model and the 3D model. If we look at the maximum absolute error, and the average absolute error, and histogram:

```
> model_dist1_3 <- as.matrix(dist(model1_3$points));
> error_matrix1_3 <- dist_mx1 - model_dist1_3;
> abs_error_matrix1_3 <- abs(error_matrix1_3);
> largest_error1_3 <- max(abs_error_matrix1_3);
> mean_abs_error1_3 <- mean(abs_error_matrix1_3);
> largest_error1_3
[1] 33.6383
> mean_abs_error1_3
[1] 12.49172
```



We can see that the maximum absolute error is 33.6383, and the average absolute error is 12.49172, which both are slightly bigger than the 2D model, but not by much. From the histogram, most errors are within 5 which is better than the 2D model. Overall, considering the GOF values, average errors, and the plots, the 3D model is slightly better than the 2D model. But both 2D and 3D models are much better than the 1D model.

(2) Second set:

Now, we will investigate the second set of distances. We can write the matrix in R:

```
# Second set:
# Generate Distance Matrix:

dist_mx2 = matrix(c(0,        11120,    8217,     7359,     9010,     16937,    3782,
          11120,    0,        6214,     6741,     15936,    6774,     12052,
          8217,     6214,     0,        1049,     16904,    8729,     6589,
          7359,     6741,     1049,     0,        16066,    9483,     5783,
          9010,     15936,    16904,    16066,    0,        14064,    10356,
          16937,    6774,     8729,     9483,     14064,    0,        14242,
          3782,     12052,    6589,     5783,     10356,    14242,    0),
          nrow = 7, ncol = 7);
```

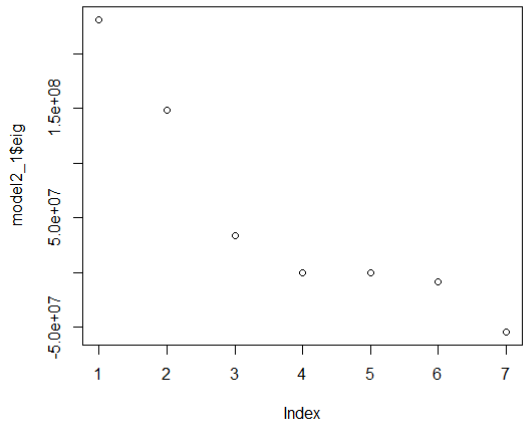
1-Dimensional Model:

Since the code used are the same as when investigating the first set, I will not include the code again. After applying MDS with the distance matrix above, we get:

```
> model2_1
$points
        [,1]
[1,]  5753.881
[2,] -5377.186
[3,] -4260.200
[4,] -3185.680
[5,]  9329.709
[6,] -6192.946
[7,]  3932.422

$eig
[1]  2.311791e+08  1.480782e+08  3.415555e+07  1.879904e+05
[5] -1.117587e-07 -8.842441e+06 -5.438087e+07

$x
NULL

$ac
[1] 0

$GOF
[1] 0.4848310           0.5589426
```

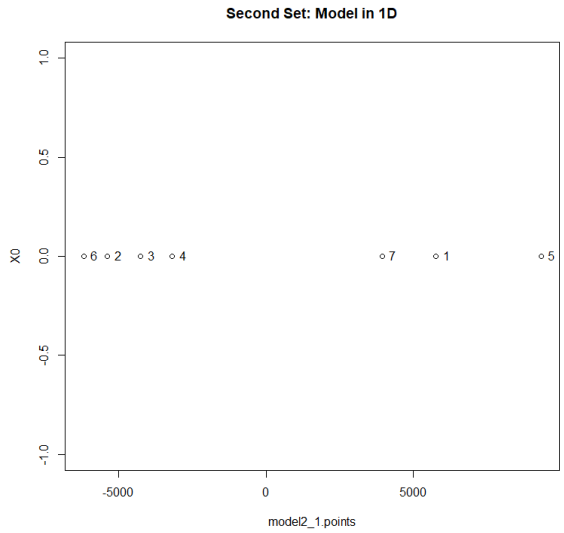
Same as the first set, we have some eigenvalues being negative. This implies the distances are non-Euclidean, so no Euclidean model will fit perfectly.

We can plot the eigenvalues:



We see that the first two eigenvalues are larger than the rest. But note that our third and fourth eigenvalues are not small. This suggests that a two-dimensional model will capture some aspects of the original data, but the two-dimensional model will not be perfect.

For the GOF, since we have negative eigenvalues, we have two which are 0.4848310 and 0.5589426 which is somewhere in the middle between 0 and 1. We can plot our model, and get:

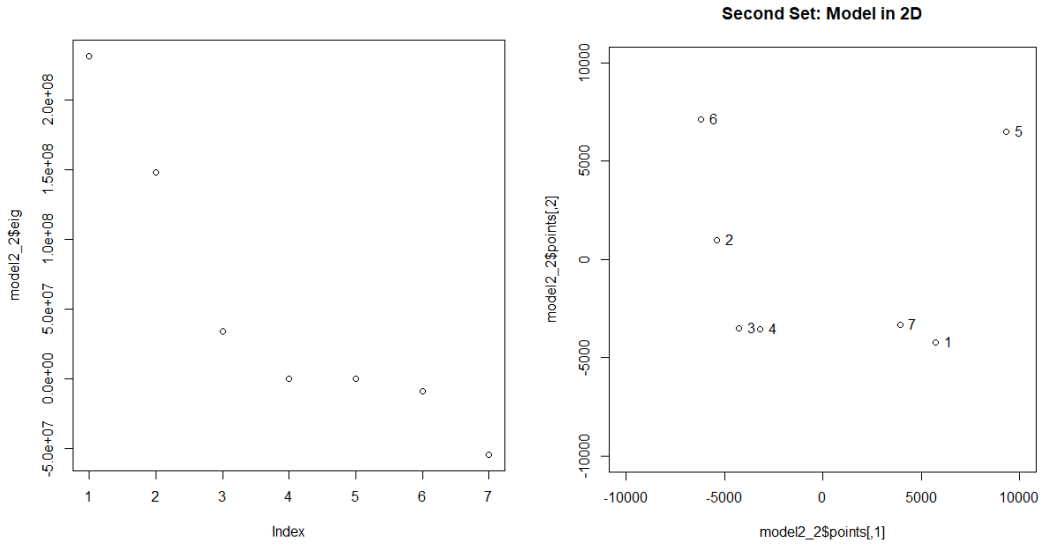


We can see that labels for 6, 2, 3, 4 are much closer than 4, 7, and 1. But there is no overlap, we will continue to investigate with a 2-D model.

2-Dimensional Model:

We get the following values and plots for the 2-D model:

```
> model2_2
$points
        [,1]      [,2]
[1,]  5753.881 -4226.2574
[2,] -5377.186   990.0238
[3,] -4260.200 -3505.4238
[4,] -3185.680 -3555.5119
[5,]  9329.709  6503.5869
[6,] -6192.946  7132.0016
[7,]  3932.422 -3338.4192

$eig
[1]  2.311791e+08  1.480782e+08  3.415555e+07  1.879904e+05 -1.117587e-07
[6] -8.842441e+06 -5.438087e+07

$x
NULL

$ac
[1] 0

$GOF
[1] 0.7953819           0.9169645
```

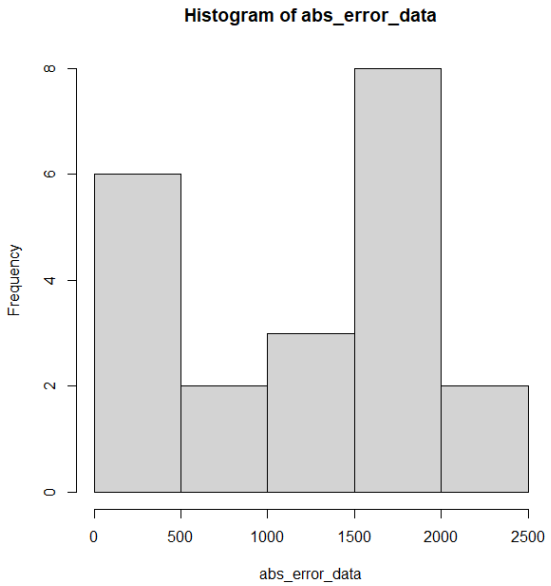


We see the eigenvalues and pot are the same which is expected. The GOF in here is 0.7953819 and 0.9169645 which is better compared with the 1-D model. There is no overlap in the plot for the model, but we can see that label 3 (Pairs) and 4 (Berlin) are somewhat close to each other, this is normal because the two cities are both in Europe and close to each other comparing with other city pairs.

To see how good this 2D model is, we can compare the model distances to the original distances:

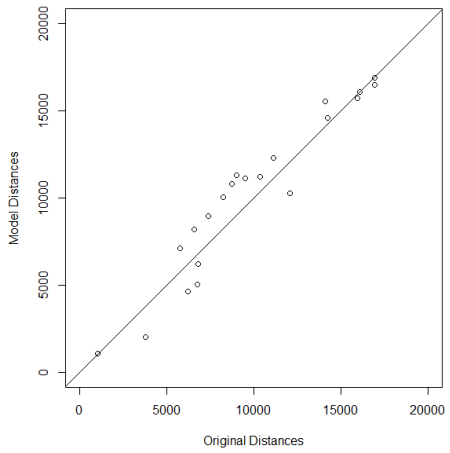
```
> model_dist <- as.matrix(dist(model2_2$points));
> error_matrix <- dist_mx2 - model_dist;
> abs_error_matrix <- abs(error_matrix);
> largest_error <- max(abs_error_matrix);
> mean_abs_error <- mean(abs_error_matrix);
> model_dist
          1         2         3         4         5         6         7
1     0.000 12292.691 10039.991  8964.689 11310.00 16484.439  2026.320
2 12292.691     0.000  4632.139  5046.246 15706.44  6195.914 10266.656
3 10039.991  4632.139     0.000  1075.687 16877.97 10811.583  8194.323
4  8964.689  5046.246  1075.687     0.000 16056.79 11102.549  7121.411
5 11310.000 15706.436 16877.971 16056.787     0.00 15535.370 11224.785
6 16484.439  6195.914 10811.583 11102.549 15535.37     0.000 14565.465
7  2026.320 10266.656  8194.323  7121.411 11224.78 14565.465     0.000
> error_matrix
           1          2           3           4           5          6          7
1     0.0000 -1172.6912 -1822.99120 -1605.689019 -2300.000059   452.5609  1755.6801
2 -1172.6912     0.0000  1581.86136  1694.754203   229.563815   578.0856  1785.3444
3 -1822.9912  1581.8614     0.00000   -26.687048    26.028813 -2082.5831 -1605.3234
4 -1605.6890  1694.7542   -26.68705     0.000000     9.212851 -1619.5491 -1338.4109
5 -2300.0001   229.5638    26.02881     9.212851     0.000000 -1471.3698  -868.7848
6   452.5609   578.0856 -2082.58307 -1619.549126 -1471.369844     0.0000  -323.4654
7  1755.6801  1785.3444 -1605.32342 -1338.410899  -868.784755  -323.4654     0.0000
> largest_error
[1] 2300
> mean_abs_error
[1] 993.9035
```

We can see the largest absolute error value is 2300 which corresponding to city 1 (Beijing) and 5 (Canberra). The average absolute error is 993.9035 which is too large. We can make a histogram of the solute values of the errors:



We can see that most errors are in the range 1500 to 2000 which is too large.

We can see more with a scatter plot:



We see that most scatters not falling on the y = x line we draw. Even though there are no super large discrepancies anywhere on the map, most data we have in here have some discrepancies. For GOF values, our 2D model is better than the 1D model. But considering the average absolute error is so large, we would like to investigate the 3D model to see if things can be better.

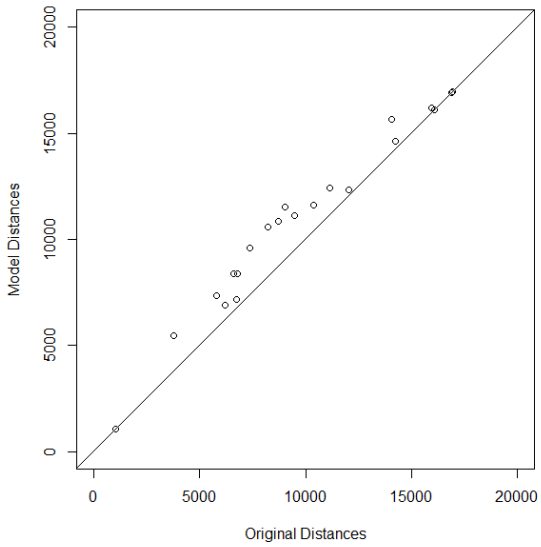
3-Dimensional Model:

We can get the following values for our 3D model:

```
> model2_3
$points
              [,1]          [,2]          [,3]
[1,]  5753.881 -4226.2574  2324.1758
[2,] -5377.186   990.0238  4031.1047
[3,] -4260.200 -3505.4238 -1062.4026
[4,] -3185.680 -3555.5119 -1031.2432
[5,]  9329.709  6503.5869   133.0393
[6,] -6192.946  7132.0016 -1632.7510
[7,]  3932.422 -3338.4192 -2761.9230

$eig
[1]  2.311791e+08  1.480782e+08  3.415555e+07  1.879904e+05
[5] -1.117587e-07 -8.842441e+06 -5.438087e+07

$x
NULL

$ac
[1] 0

$GOF
[1] 0.8670132 0.9995455
```

We can see our GOF values have increased to 0.8670132 and 0.9995455 and notice our second GOF value is very close to 1. So, in terms of GOF, 3D model is better than the 2D model.

And when we look at the scatter plot:

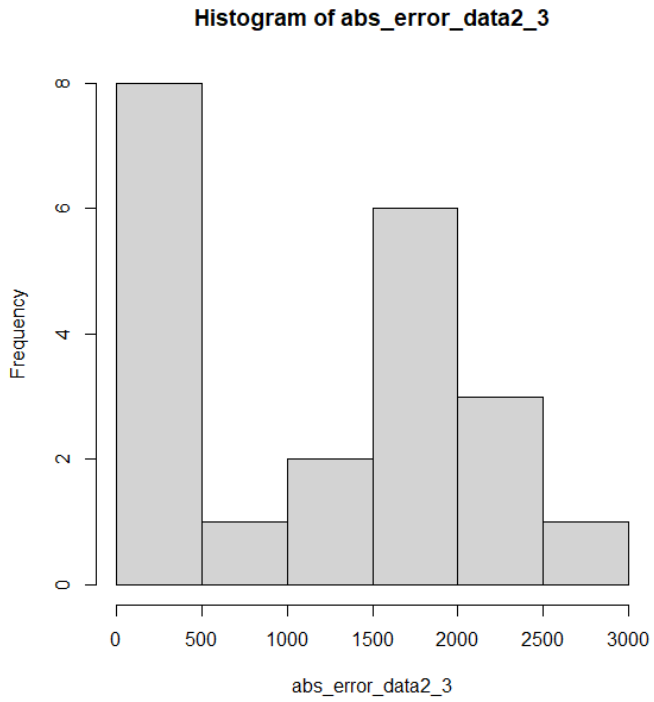


We see that comparing with the scatter plot in the 2D model, the 3D model has some noticeable larger discrepancies. We can get the errors:

```
> model_dist2_3 <- as.matrix(dist(model2_3$points));
> error_matrix2_3 <- dist_mx2 - model_dist2_3;
> abs_error_matrix2_3 <- abs(error_matrix2_3);
> largest_error2_3 <- max(abs_error_matrix2_3);
> mean_abs_error2_3 <- mean(abs_error_matrix2_3);
> largest_error2_3
[1] 2510.294
> mean_abs_error2_3
[1] 963.8731
```

We see that the largest absolute error is 2510.294 which is larger than 2300 we got for the 2D model. And for average absolute error, we have 963.8731 which is slightly smaller than 993.9035 we got the 2D model. Those errors are still very large despite we have a high GOF values.

We can make a histogram with the absolute errors:



We see that most of the errors are within 500 which is better than the 2D model. But notice we still have a lot of errors in the range between 1500 to 2000. Considering the GOF values, 3D model is better than the 2D because with higher dimensions, GOF values would become higher and higher. But notice that when we look at the errors and the scatter plot, 3D model does not perform any better than the 2D model. This indicates that we cannot solely based on the GOF values to see how good a model is, instead, we need to count various aspects of that model. For large values like this second set distances, it is hard to get a good model in 3D.

IV. Comment and Conclusion:

Our two set of distances both are not Euclidean. For the first set where we have closer city distances, the 3D model works well, whereas for the second set where we have large distances, the 3D model we got did not perform that good. But we can see that with MDS, we can get some useful information, and study the different models in an efficient way.