Math381 Assignment 7

Yuchen AN

June 28, 2023

1 Introduction

For assignment 7, I want to use the multidimensional scaling techniques to analyze the nutrient levels in a variety of foods.
The goal of multidimensional scaling(MDS) is the creation of a low dimensional model for a set of objects with a given set of pair-wise distances while preserving the dissimilarity or similarity relationships between them.
The file ***Measurements of nutrient levels in a variety of foods*** consists of 27 foods and each food item is associated with five nutrient levels which are energy(calories), protein, fat, calcium, and iron.
Below are the details of the file:

```
#  file06.txt
#
#  Reference:
#
#    John Hartigan,
#    Clustering Algorithms,
#    Wiley, 1975.
#    ISBN 0-471-35645-X
#    LC: QA278.H36
#    Dewey: 519.5'3
#
#  Nutrient levels were measured in a 3 ounce portion of various foods.
#
#  "Name" is the name of the item.
#
#  "Energy" is the number of calories.
#
#  "Protein" is the amount of protein in grams.
#
#  "Fat" is the amount of fat in grams.
#
#  "Calcium" is the amount of calcium in milligrams.
#
#  "Iron" is the amount of iron in milligrams.
#
"Nutrients in Meat, Fish and Fowl, Hartigan page 86"
6 columns
27 rows
"Name"                "Energy" "Protein" "Fat" "Calcium" "Iron"
"Braised beef"        340      20        28     9         2.6
"Hamburger"           245      21        17     9         2.7
"Roast beef"          420      15        39     7         2.0
"Beefsteak"           375      19        32     9         2.6
"Canned beef"         180      22        10    17         3.7
```

"Broiled chicken"	115	20	3	8	1.4
"Canned chicken"	170	25	7	12	1.5
"Beef heart"	160	26	5	14	5.9
"Roast lamb leg"	265	20	20	9	2.6
"Roast lamb shoulder"	300	18	25	9	2.3
"Smoked ham"	340	20	28	9	2.5
"Pork roast"	340	19	29	9	2.5
"Pork simmered"	355	19	30	9	2.4
"Beef tongue"	205	18	14	7	2.5
"Veal cutlet"	185	23	9	9	2.7
"Baked bluefish"	135	22	4	25	0.6
"Raw clams"	70	11	1	82	6.0
"Canned clams"	45	7	1	74	5.4
"Canned crabmeat"	90	14	2	38	0.8
"Fried haddock"	135	16	5	15	0.5
"Broiled mackerel"	200	19	13	5	1.0
"Canned mackerel"	155	16	9	157	1.8
"Fried perch"	195	16	11	14	1.3
"Canned salmon"	120	17	5	159	0.7
"Canned sardines"	180	22	9	367	2.5
"Canned tuna"	170	25	7	7	1.2
"Canned shrimp"	110	23	1	98	2.6

I found that the data columns exhibit varying scales. In order to prevent the impact of varying scales in distance calculations, it is necessary to normalize the columns of data.
There are two ways to normalize:

1. Find the mean and standard deviation of each column and subtract the mean and divide by the standard deviation in each column.

2. Find the min and max value in each column, and then map each column entry x to (x-min)/(max-min).

The first method will transform data into a new data set in which each column has mean zero and unit standard deviation.
The second methods will convert the data set into a new data set in which each column as a minimum of 0 and a maximum of 1.
After normalization, we can create a matrix of distances containing the distance between each pair. I choose to use a distance method known as "minkowski". The distance between row i and row j will be calculated as

$$d_{ij} = (\sum_{m} |r_{i,m} - r_{j,m}|^2)^{1/2}$$

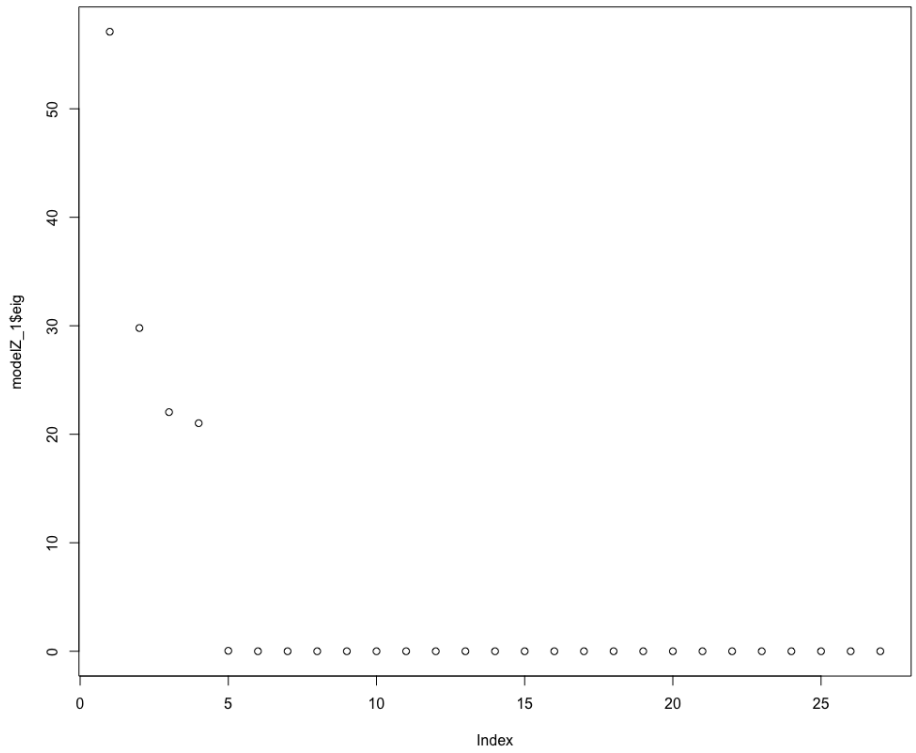
$r_{i,m}$ represents the value located at row i and column m in the normalized matrix .
Next, we create distance matrix and use MDS to create low-dimensional models with cmdscale command in R.

2 Lower-dimension Model

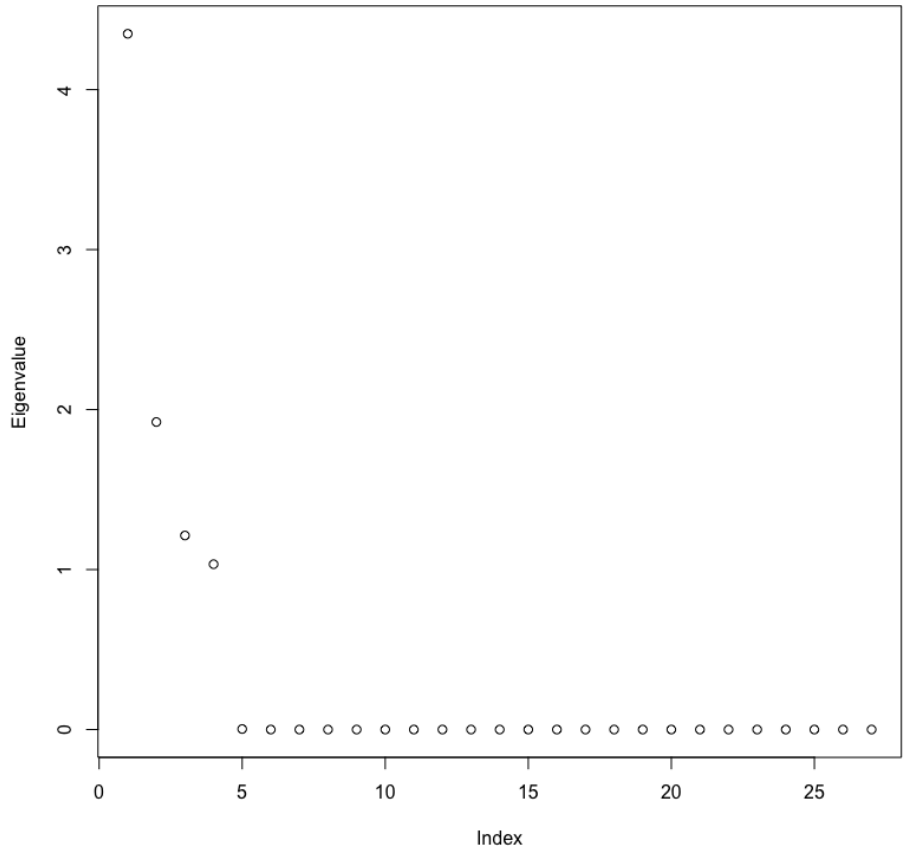
We apply method 1 and method 2 to normalize the input data. Then we calculate the distances between all pairs and use the cmdscale function in R to create models with different dimensions. In order to compare the effects of two normalization methods on the models, I will discuss them together and analyze their respective impacts.
We first study the eigenvalues generated after we apply MDS to the original distance matrix. We only need to calculate the eigenvalues once because eigenvalues do not depend on which model we are looking at.

To better understand the eigenvalues, we plot them. All eigenvalues can be found in the appendix.

Eigenvalues of distance matrix after applying Z-score normalization



Eigenvalues of distance matrix after applying max-min normalization

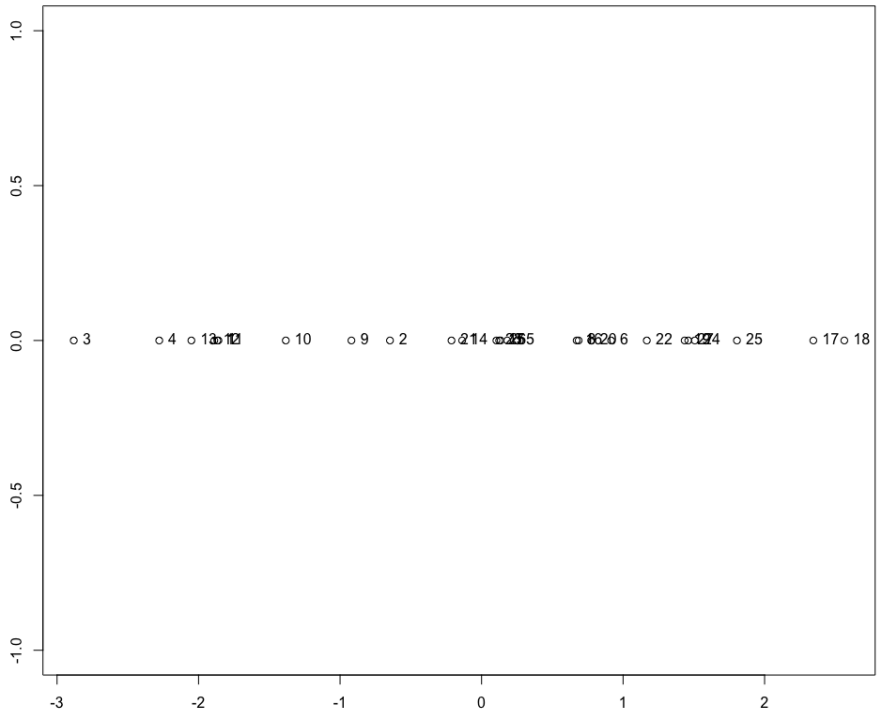


R reports the eigenvalues in descending order of magnitude. We can see that the first 4 eigenvalues in both graph are substantial, and the rest are tiny. This suggests that the input data might fit perfectly into 4-dimensional space. However, our input data is 5-dimensional. The reason is that there are two variables highly correlated. It is possible to reduce the dimension by combining them into a single variable. To find correlated variables, I use the cor() in R to calculate the correlation coefficient between columns of valuables. And I found Energy and Fat are highly correlated. Their correlation coefficient is 98.7% . I will discuss more about correlation in the later part.

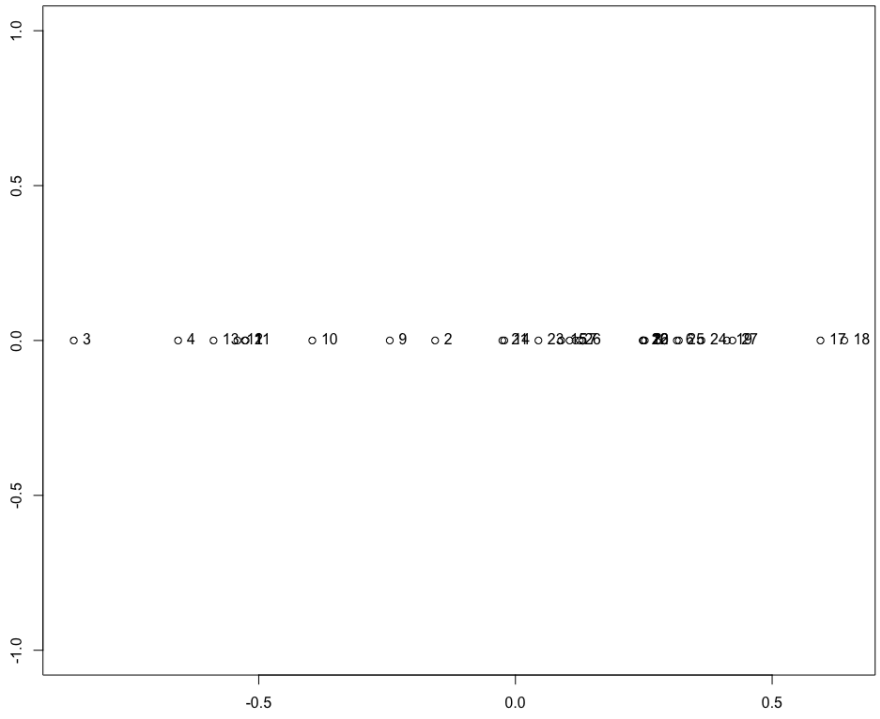
2.1 1-dimension model

Here are the plots of 1-dimension model:

Z-score normalization



Max-Min normalization

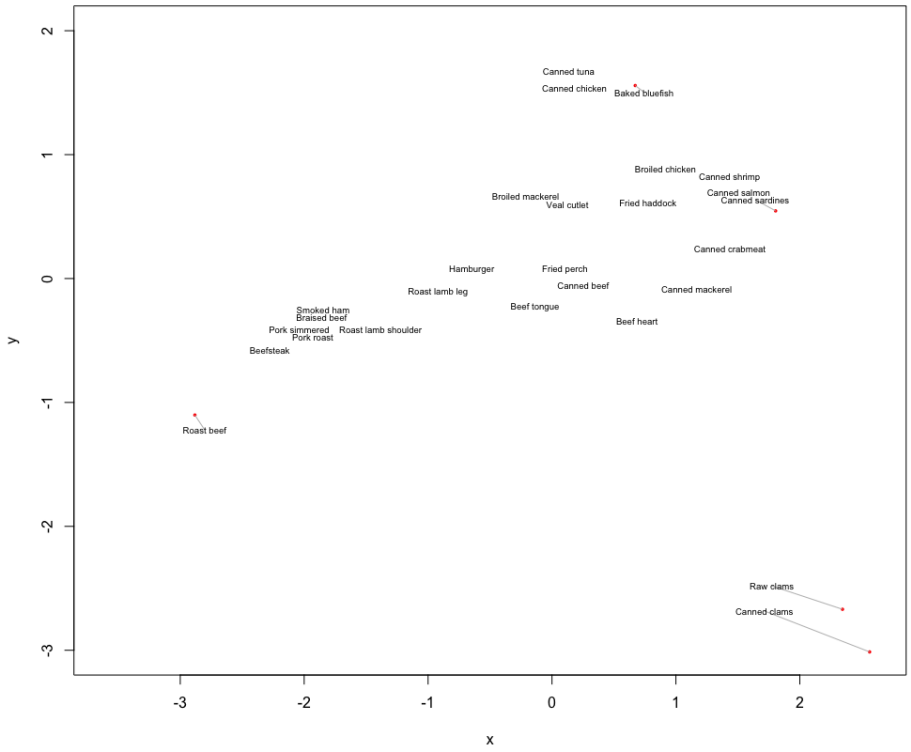


Each point on the plot represents one food from the input file. Since most points are compact, it is hard to label the plot of 1-dimension model. Therefore, I will investigate the plot of 2-dimension model, which is the same as the 1-dimension model with a single new dimension added on.

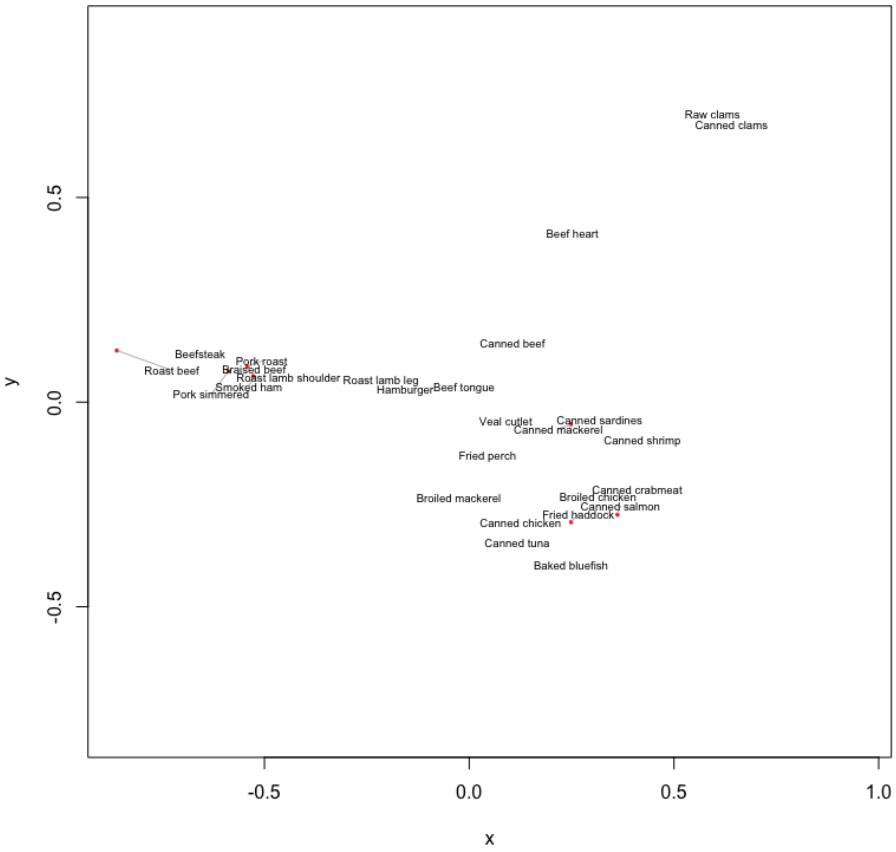
Here is the plot of 2-dimension model:

2.2 2-dimension model

Z-score normalization



Max-Min normalization



From plots above, raw clams and canned clams are extreme objects, and they both appear to be farthest part.As examining the nutritional level of these two items, I observed that they have a significantly higher iron content compared to other food sources. I can observe an obvious left-right arrangement of those food. I found pork roast, braised beef and smoked are clustered. As I checked the nutritional levels of these three items, I observed they all have similar content in calories, protein, fat, calcium and Iron.
It is hard to visualize and determine the location of the points in the three-dimensional plot, so I didn't include the plot of 3-dimension models here.

2.3 GOF

Now, I check the goodness-of-fit value(GOF) for my models with different dimensions. The GOF is a value between 0 and 1 where a value of 1 indicated a perfect fit and a value of 0 indicate a terrible fit. I can get two GOF values by using the cmdscale( ) function in R. One GOF value is calculated as the sum of sorted eigenvalues divided by the sum of the absolute values of all eigenvalues. One GOF is calculated as the sum of the sorted eigenvalues divided by the sum of all positive eigenvalues. For my dataset, the two GOF values are same for all models.
The following table shows the dimensions of the model along with their corresponding GOF values.

Z-score normalization

Dimension of model	GOF value
1	0.4392647
2	0.6684426
3	0.8379853
4	0.9996797

Max-Min normalization

Dimension of model	GOF value
1	0.5103875
2	0.735984
3	0.878326
4	0.9996141

From tables above, I found that the 4-dimension models for both normalization method has the highest GOF, which means that the 4-dimension model is a good representation of the data.

2.4 Matrix Comparison

Next, we assess the similarity between the input distance matrix and the distance matrix generated by the model at different dimensions. If the distances of lower-dimensional model are similar with the original distance matrix, we say the lower-dimensional model is a good representation of the input data. To measure the similarity of two matrices, we use fourthree methods: mean absolute differences of the entries, mean square differences of the entries, maximum absolute differences, and max percentage error. Here are their definitions:

$$MeanAbsoluteDifference = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}|d_{i,j} - m_{i,j}|}{mn}$$

$$MeanSquareDifference = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(d_{i,j} - m_{i,j})^2}{mn}$$

$$MaximumAbsoluteDifference = max(|d_{i,j} - m_{i,j}|)$$

$$MaximumPercentageError = max(\frac{d_{i,j} - m_{i,j}}{d_{i,j}}) \times 100$$

$d_{i,j}$ represents the entry in the i-th row and j-th column of original distance matrix. $m_{i,j}$ represents the entry in the i-th row and j-th column of low-dimension distance matrix model. Both original distance matrix and lower-dimension distance matrices have size m×n. $1 \leq i \leq m$. $1 \leq j \leq n$.

Z-score normalization

Dimension of model	Mean Absolute Difference	Mean Square Difference	Maximum Absolute Difference	Maximum Percentage Error
1	1.099164	2.154063	4.965607	99.62%
2	0.6231984	1.054897	4.403482	92.49%
3	0.2618652	0.2576425	3.292502	76.21%
4	0.0007664	3.684745e-06	0.017904	2.65%

Max-Min normalization

Dimension of model	Mean Absolute Difference	Mean Square Difference	Maximum Absolute Difference	Maximum Percentage Error
1	0.2562089	0.1294536	1.113213	99.94%
2	0.1302779	0.04644098	0.8935765	89.51%
3	0.05371312	0.02233483	0.8300566	79.52%
4	0.00023999	3.777675e-07	0.006148	3.45%

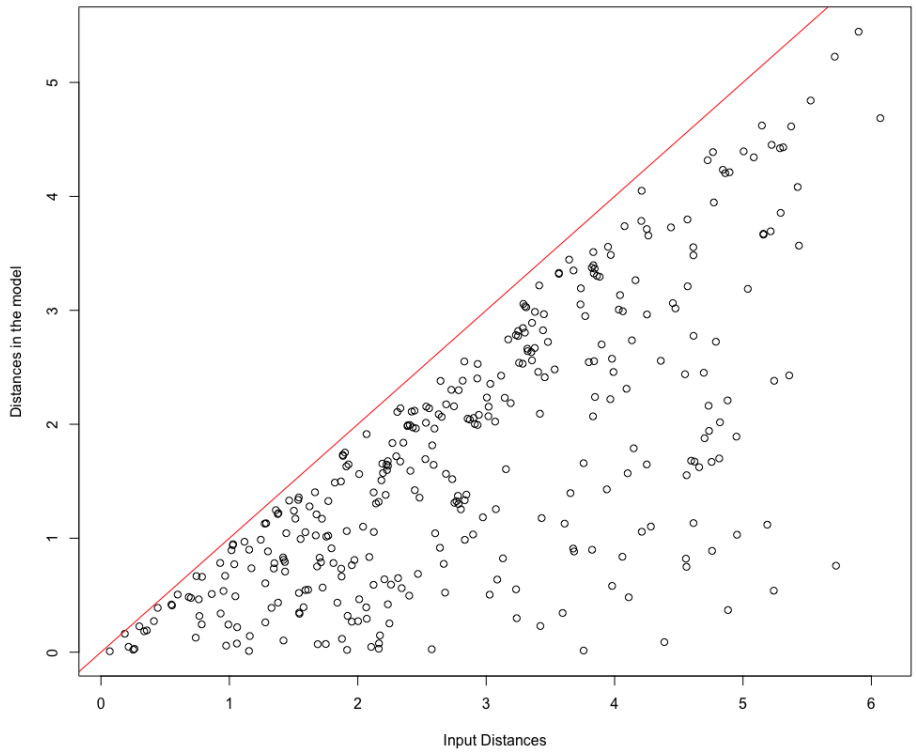
Collectively, these measures provide information about the magnitude, variability, and maximum discrepancy between the low-dimensional distances matrix with input distances. When compared to the input model.it is obvious that the 4-dimensional model in both normalization methods consistently exhibits the lowest values for mean absolute differences, mean square differences, maximum absolute differences, and max percentage error, which again serves as further evidence that 4-dimension model is a good model.

2.5 Distances in the model versus the input distances

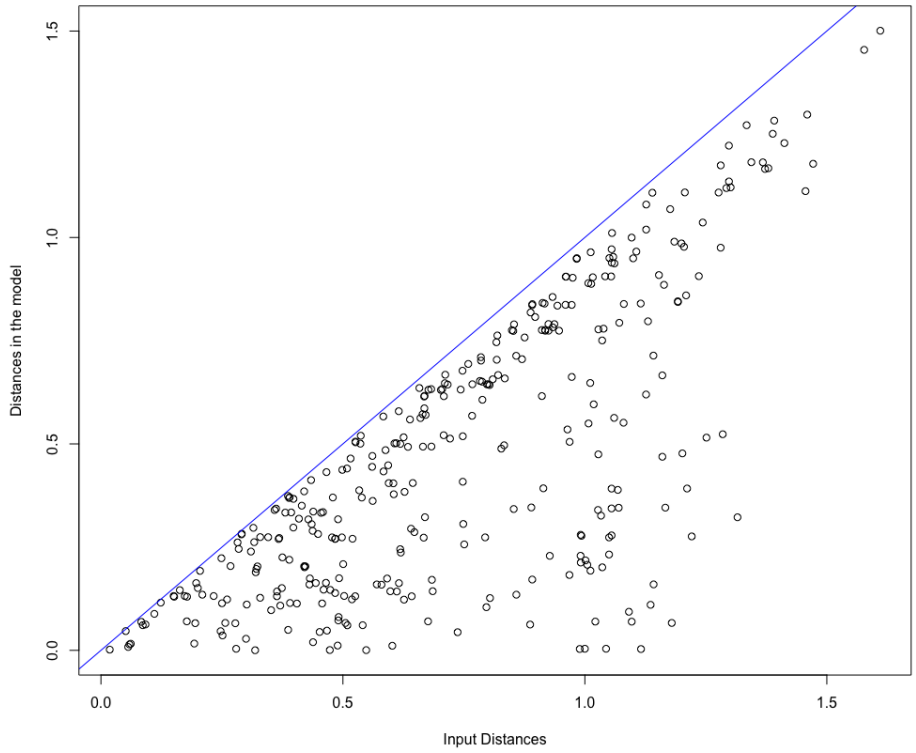
To better visualize the difference between lower-dimensional distances model and input distances model, I plotted the input distances against the modeled distances. If the low-dimension model is perfect, all points will lie on the line y=x. And the closer the points are to the line, the better the model will be. Once again, these plots demonstrate that the 4-dimensional distance model can accurately represent the input distances. While the 3-dimensional distance model generally performs well,

there are still some points that do not align perfectly with the input distances.
Here are the plots.

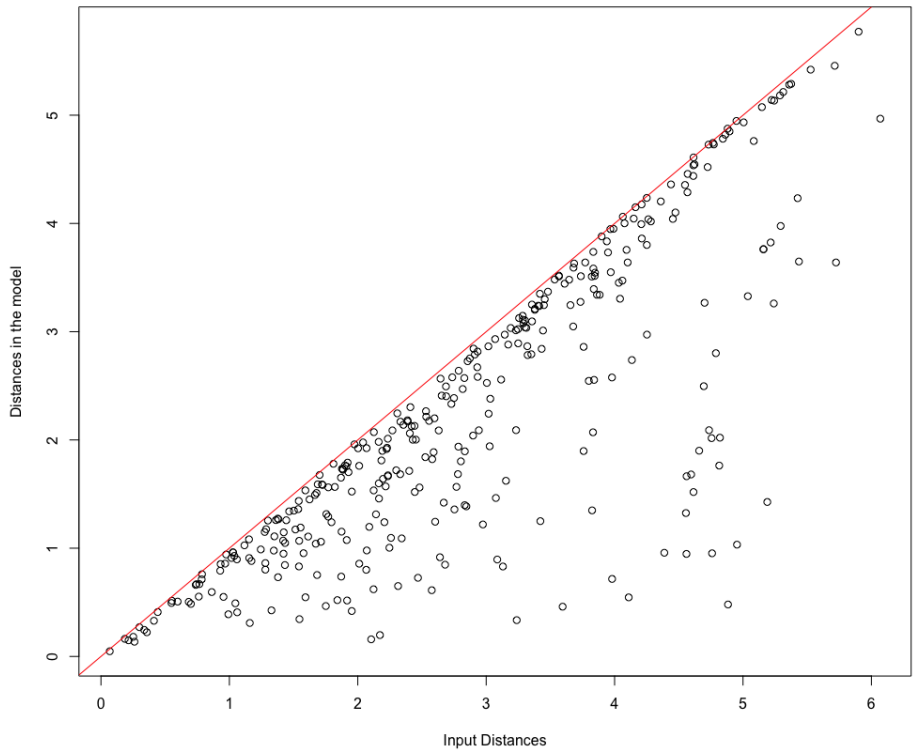
1-D Model versus Input distance with Z-score normalization



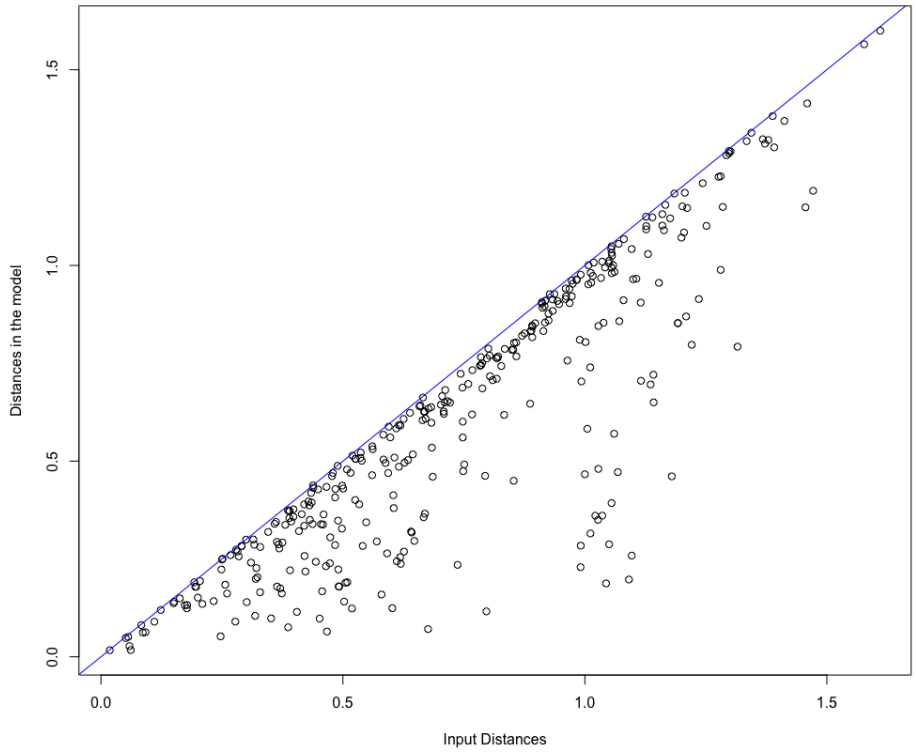
1-D Model versus Input distance with Max-Min normalization



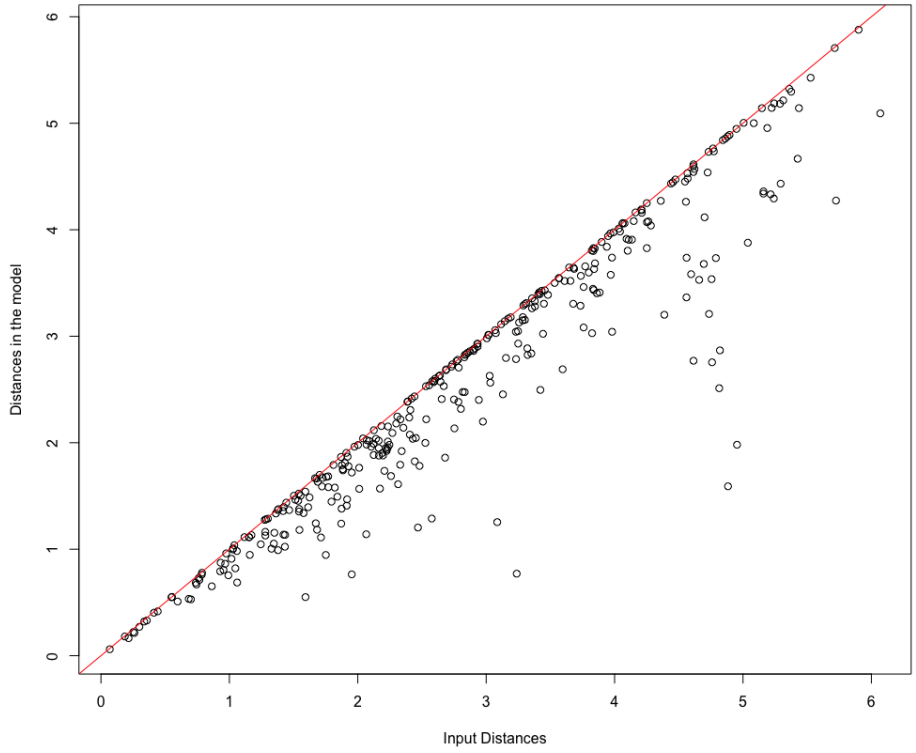
2-D Model versus Input distance with Z-score normalization



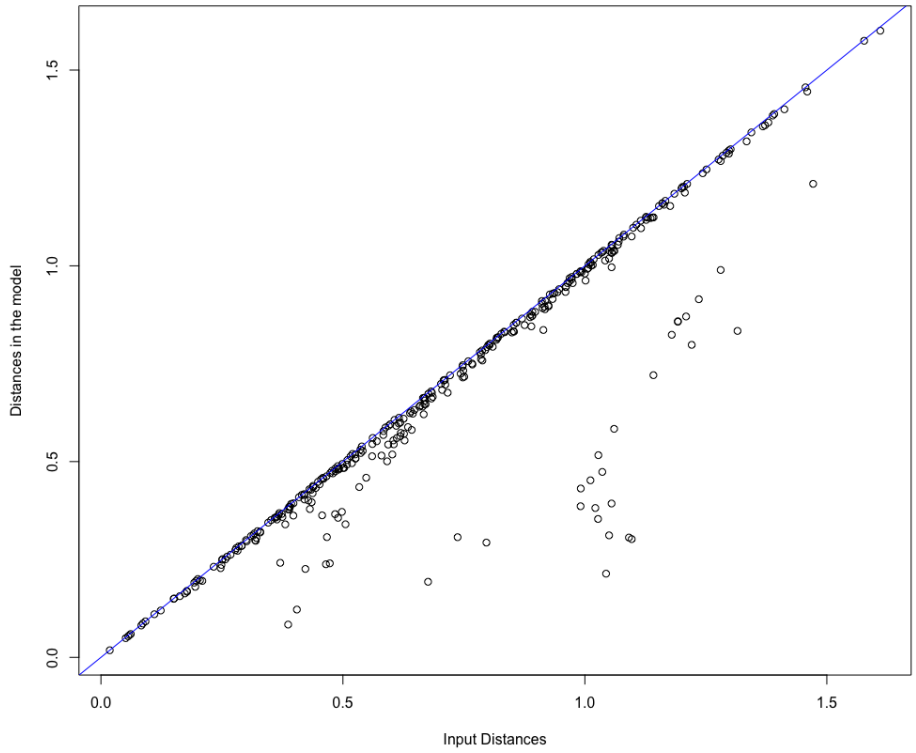
2-D Model versus Input distance with Max-Min normalization



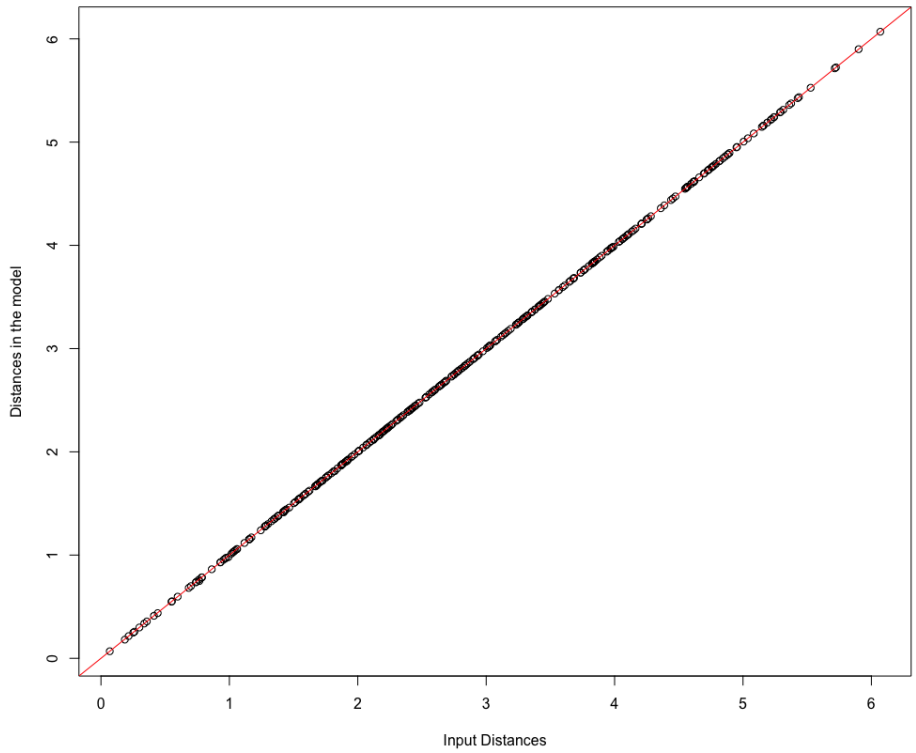
3-D Model versus Input distance with Z-score normalization



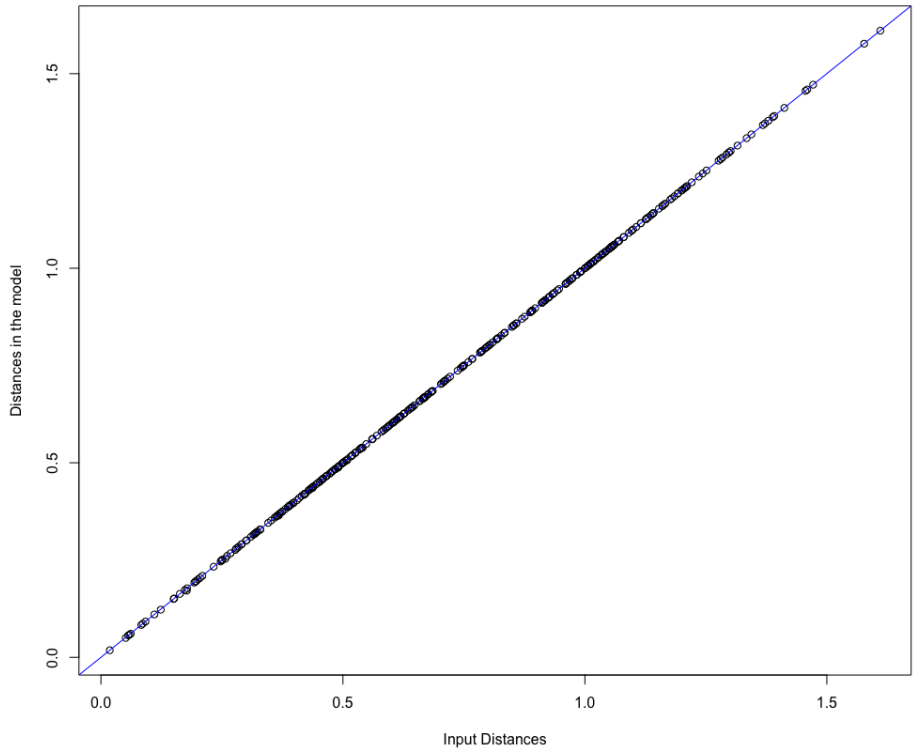
3-D Model versus Input distance with Max-Min normalization



4-D Model versus Input distance with Z-score normalization



4-D Model versus Input distance with Max-Min normalization



2.6 Correlation

To assess what could the two dimensions in the model represent, I calculated correlations coefficients between coordinates of the model and input data.

Z-score normalization

Coordinates	Energy	Protein	Fat	Calcium	Iron
x	-0.9696465	-0.2233847	-0.9483475	0.526078	0.1734568
y	-0.09134263	0.7379044	-0.2138646	0.003365924	-0.7397932

Max-Min normalization

Coordinates	Energy	Protein	Fat	Calcium	Iron
x	-0.9920399	-0.1548681	-0.9824007	0.4206406	0.1441766
y	0.04008076	-0.4552143	0.1225704	0.01978941	0.9379603

Based on the information provided in both tables, I observed a strong correlation between the input data of Energy and Fat with the x-coordinates of the points in the model.This again shows that variable Energy and Fat are highly correlated, as result of which 4-dimension model can fit the input file perfectly well.
By using the z-score normalization, I observed that the y-coordinate is strongly correlated with Protein and Iron. However, the coefficients are of opposite sign. It seems that when content of protein gets higher, the content of iron gets lower, and vice versa. The y-coordinate seems to be associated with the protein-iron continuum in this context.
By using the max-min normalization, I observed that the y-coordinate is strongly correlated with Iron. This observation may explain why raw clams and canned clams, which are rich in iron, have significantly higher values in the 2-dimensional model compared to other foods.

3 Appendix

Eigenvalues after Z-score normalization.

```
5.710441e+01  2.979313e+01  2.204055e+01  2.102026e+01  4.164288e-02  2.011056e-15
1.734362e-15  8.211894e-16  6.105383e-16  5.130957e-16  4.874476e-16  4.822948e-16
4.734788e-16  1.707162e-16 -1.712154e-16 -6.668318e-16 -6.847494e-16 -9.933536e-16
-1.475882e-15 -1.529168e-15 -1.565917e-15 -1.755746e-15 -2.158919e-15 -3.009891e-15
-4.589793e-15 -4.762270e-15 -1.289191e-14
```

Eigenvalues after Max-Min normalization

```
4.349011e+00  1.922308e+00  1.212896e+00  1.033495e+00
3.288631e-03  1.799416e-16  1.413151e-16  1.334802e-16
7.578164e-17  6.313842e-17  4.811195e-17  4.134445e-17
2.905900e-17  6.526173e-18 -7.508205e-18 -9.777515e-18
-1.341559e-17 -1.386237e-17 -1.767632e-17 -2.349158e-17
-2.940372e-17 -4.035112e-17 -5.007226e-17 -8.941125e-17
-1.492369e-16 -5.654954e-16 -7.087098e-16
```