Math 381

Homework 7

December 3, 2021

Haochen Wang

Problem:

In this paper, I will find the distances between different cities using expectations of life to create one-, two-, and three dimensional models with MDS. The dataset I will use is “Expectation of Life in Various Cities”, which could be accessed through the URL (https://people.sc.fsu.edu/~jburkardt/datasets/hartigan/file22.txt). After creating these models, I will standardize the columns from the dataset above and find the effect of standardization on the models.

Mathematical formulas:

I will use multidimensional scaling (MDS) to solve this problem since I want to create some low dimensional models for the objects. The dataset we will use has 17 rows and 10 columns. Now, I will assign indexes to header rows and columns that will be used when I create matrices later:

Further expectation of life for different sex (M represents male and F represents female) at different age (0, 25, 50, 75)	Index (for column number)
M00	1
M25	2
M50	3
M75	4
F00	5
F25	6
F50	7
F75	8

The city name in front of the year of the census	Index (for row number)
“Montreal”, “1966”	1
“Toronto”, “1966”	2
“Vancouver”, “1966”	3
“Kaohsiung”, “1966”	4
“Taipei”, “1966”	5
“Djakarta”, “1961”	6
“Copenhagen”, “1966”	7
“Helsinki”, “1966”	8
“East Berlin”, “1961”	9
“West Berlin”, “1961”	10
“West Berlin”, “1967”	11
“Dusseldorf”, “1966”	12
“Hamburg”, “1966”	13
“Stockholm”, “1960”	14
“London”, “1967”	15
“Sydney”, “1967”	16

Since the scales of dimensions are vary, it is essential for me to standardize the dataset. I will also include the dataset without standardization (original dataset) for comparison in the future. Define matrix A to represent the expectations of life. After excluding headers, matrix A is below:

$$\begin{bmatrix}
67 & 44 & 22 & 7 & 73 & 51 & 28 & 9 \\
68 & 45 & 23 & 8 & 75 & 53 & 29 & 10 \\
68 & 46 & 23 & 8 & 75 & 53 & 30 & 11 \\
66 & 43 & 22 & 8 & 71 & 49 & 26 & 9 \\
69 & 46 & 24 & 8 & 72 & 50 & 27 & 9 \\
44 & 36 & 18 & 6 & 46 & 37 & 19 & 6 \\
68 & 46 & 23 & 8 & 75 & 56 & 28 & 9 \\
66 & 42 & 21 & 7 & 73 & 50 & 27 & 8 \\
66 & 45 & 22 & 7 & 71 & 49 & 27 & 8 \\
66 & 45 & 22 & 7 & 72 & 50 & 27 & 9 \\
67 & 45 & 22 & 7 & 73 & 50 & 27 & 9 \\
67 & 45 & 22 & 8 & 74 & 51 & 28 & 9 \\
68 & 46 & 23 & 7 & 74 & 51 & 27 & 9 \\
69 & 46 & 23 & 7 & 75 & 52 & 28 & 9 \\
69 & 47 & 23 & 8 & 77 & 54 & 30 & 12 \\
67 & 44 & 22 & 7 & 73 & 50 & 27 & 9
\end{bmatrix}$$

Define matrix X to represent the distance between any two objects (combination of city and year) from matrix A, so X_{ij} would represent the distance between the object at i-th row and the object at j-th row. Since A is a 16 by 8 matrix, there are 16 rows in matrix A, so matrix X should be a 16 by 16 matrix, and we could use Euclidean distance to define X_{ij} as

$$X_{ij} = \sqrt{\sum_{a=1}^{8} (A_{ia} - A_{ja})^2}$$

Next, I will standardize the columns of input data from my dataset. There are 2 easy methods that are mentioned in the description of this homework, and I will use both of them.

Apply to the first method: for every entry in matrix A, find the mean and standard deviation of the column where the entry is in and then subtract the mean and divide by the standard deviation.

Define matrix B and matrix C as 16 by 8 matrices. Define μ_j and σ_j as the mean and standard deviation of column j from matrix A separately, then we will have

$$B_{ij} = (A_{ij} - \mu_j)/\sigma_j$$

Define matrix Y that represents the distance between any two objects from matrix B.

The steps I use to build matrix Y is similar to the steps I used to build matrix X, so we have

$$Y_{ij} = \sqrt{\sum_{a=1}^{8} (B_{ia} - B_{ja})^2}$$

Apply to the second method: for every entry in matrix A, find the minimum value in the column where the entry is in and then divide by the range of that column. This means we will have

$$C_{ij} = (A_{ij} - min_j)/(max_j - min_j)$$

Define matrix Z that represents the distance between any two objects from matrix C.

The steps I use to build matrix Z is similar to the steps I used to build matrix X, so we have

$$Z_{ij} = \sqrt{\sum_{a=1}^{8} (C_{ia} - C_{ja})^2}$$

Now, I have finished creating the matrices I need, and I am ready to apply MDS to create low-dimensional models for the set of objects through R.

Coding:

Below is the code I write in R to solve the MDS problem above:

```
library(RColorBrewer)
library(wordcloud)
filewithheader <- read.csv(file="/Users/mac/Desktop/材料/UW/UW\ autumn\ 2021/MATH\ 381/HW\ 7/file22withheader.csv",
head=FALSE, sep=",")
filenoheader <- read.csv(file="/Users/mac/Desktop/材料/UW/UW\ autumn\ 2021/MATH\ 381/HW\ 7/file22withheader.csv",
head=TRUE, sep=",")
# Remove the City's name and year from the file without header for standardization in future.
data <- filenoheader[-c(1, 2)]
# Standardized by finding the mean and standard deviation of each column, and then subtracting the mean and divided by the
standard deviation in each column.
# Name this standardization method as classic method.
classic <- matrix(nrow = length(data[, 1]), ncol = length(data))
for (j in 1:length(data)) {
  for (i in 1:length(data[, j])) {
    classic[i, j] <- (data[i, j] - mean(data[, j]))/sd(data[, j])
  }
}
# Standardized by finding the min and max value in each column, and then mapping each column entry x to (x-min)/(max-min).
# Name this standardization method as convert method.
convert <- matrix(nrow = length(data[, 1]), ncol = length(data))
for (j in 1:length(data)) {
  for (i in 1:length(data[, j])) {
    convert[i, j] <- (data[i, j] - min(data[, j]))/(max(data[, j]) - min(data[, j]))
  }
}
# Generate index for the cities that will be used in plots below.
cityIndex <- 1:length(data[, 1])
# Apply MDS to the not standardized model in 3 different dimensions. Name this model as model 1.
myfun <- function(x){return(x)}
distances1 = dist((apply(data, MARGIN = c(1,2), myfun)), method="minkowski", p = 2)
# In 1-D
model1_1D <- cmdscale(distances1, k = 1, eig = TRUE)
check1_1D <- data.frame(model1_1D$points, 0)
plot(model1_1D$eig, xlab = "Index for cities", ylab = "Eigenvalues")
textplot(check1_1D[, 1], check1_1D[, 2], gsub("(^[^\\s]+\\s{1})", "", cityIndex, perl = TRUE), xlim = c(-20, 40), ylim = c(-20,
40), cex = 0.6)
ABSDIFF1_1D <- mean(abs(as.matrix(dist(check1_1D)) - as.matrix(distances1)))
GOF1_1D <- model1_1D$GOF
plot(distances1, dist(check1_1D))
abline(0, 1, col = "green")
# In 2-D
model1_2D <- cmdscale(distances1, k = 2, eig = TRUE)
check1_2D <- data.frame(model1_2D$points, 0)
textplot(check1_2D[, 1], check1_2D[, 2], gsub("(^[^\\s]+\\s{1})", "", cityIndex, perl = TRUE), xlim = c(-10, 40), ylim = c(-10,
40), cex = 0.6)
ABSDIFF1_2D <- mean(abs(as.matrix(dist(check1_2D)) - as.matrix(distances1)))
```

```
GOF1_2D <- model1_2D$GOF
plot(distances1, dist(check1_2D))
abline(0, 1, col = "green")
# In 3-D
model1_3D <- cmdscale(distances1, k = 3, eig = TRUE)
check1_3D <- data.frame(model1_3D$points, 0)
ABSDIFF1_3D <- mean(abs(as.matrix(dist(check1_3D)) - as.matrix(distances1)))
GOF1_3D <- model1_3D$GOF
plot(distances1, dist(check1_3D))
abline(0, 1, col = "green")
# Apply MDS to the standardized model by using the classic method in 3 different dimensions. Name this model as model 2.
distances2 = dist(classic, method="minkowski", p = 2)
# In 1-D
model2_1D <- cmdscale(distances2, k = 1, eig = TRUE)
check2_1D <- data.frame(model2_1D$points, 0)
plot(model2_1D$eig, xlab = "Index for cities", ylab = "Eigenvalues")
textplot(check2_1D[, 1], check2_1D[, 2], gsub("(^[^\\s]+\\s{1})", "", cityIndex, perl = TRUE), xlim = c(-10, 5), ylim = c(-10, 5),
cex = 0.6)
ABSDIFF2_1D <- mean(abs(as.matrix(dist(check2_1D)) - as.matrix(distances2)))
GOF2_1D <- model2_1D$GOF
plot(distances2, dist(check2_1D))
abline(0, 1, col = "green")
# In 2-D
model2_2D <- cmdscale(distances2, k = 2, eig = TRUE)
check2_2D <- data.frame(model2_2D$points, 0)
textplot(check2_2D[, 1], check2_2D[, 2], gsub("(^[^\\s]+\\s{1})", "", cityIndex, perl = TRUE), xlim=c(-10, 5), ylim = c(-10, 5),
cex = 0.6)
ABSDIFF2_2D <- mean(abs(as.matrix(dist(check2_2D)) - as.matrix(distances2)))
GOF2_2D <- model2_2D$GOF
plot(distances2, dist(check2_2D))
abline(0, 1, col = "green")
# In 3-D
model2_3D <- cmdscale(distances2, k = 3, eig = TRUE)
check2_3D <- data.frame(model2_3D$points, 0)
ABSDIFF2_3D <- mean(abs(as.matrix(dist(check2_3D)) - as.matrix(distances2)))
GOF2_3D <- model2_3D$GOF
plot(distances2, dist(check2_3D))
abline(0, 1, col = "green")
# Apply MDS to the standardized model by using the convert method in 3 different dimensions. Name this model as model 3.
distances3 = dist(convert, method="minkowski", p = 2)
# In 1-D
model3_1D <- cmdscale(distances3, k = 1, eig = TRUE)
check3_1D <- data.frame(model3_1D$points, 0)
plot(model3_1D$eig, xlab = "Index for cities", ylab = "Eigenvalues")
textplot(check3_1D[, 1], check3_1D[, 2], gsub("(^[^\\s]+\\s{1})", "", cityIndex, perl = TRUE), xlim=c(-1, 3), ylim = c(-1, 3), cex
= 0.6)
ABSDIFF3_1D <- mean(abs(as.matrix(dist(check3_1D)) - as.matrix(distances3)))
GOF3_1D <- model3_1D$GOF
plot(distances3, dist(check3_1D))
abline(0, 1, col = "green")
# In 2-D
model3_2D <- cmdscale(distances3, k = 2, eig = TRUE)
check3_2D <- data.frame(model3_2D$points, 0)
textplot(check3_2D[, 1], check3_2D[, 2], gsub("(^[^\\s]+\\s{1})", "", cityIndex, perl = TRUE), xlim=c(-1, 3), ylim = c(-1, 3), cex
= 0.6)
ABSDIFF3_2D <- mean(abs(as.matrix(dist(check3_2D)) - as.matrix(distances3)))
GOF3_2D <- model3_2D$GOF
plot(distances3, dist(check3_2D))
abline(0, 1, col = "green")
# In 3-D
model3_3D <- cmdscale(distances3, k = 3, eig = TRUE)
check3_3D <- data.frame(model3_3D$points, 0)
```

```
ABSDIFF3_3D <- mean(abs(as.matrix(dist(check3_3D)) - as.matrix(distances3)))
GOF3_3D <- model3_3D$GOF
plot(distances3, dist(check3_3D))
abline(0, 1, col = "green")
# Compare the mean for absolute difference between the distance from the data and distance we calculated by using 3 methods
above in 3 different dimensions.
plot(c(ABSDIFF2_1D, ABSDIFF2_2D, ABSDIFF2_3D), ylim = c(0, 5), xlab = "Dimension", ylab = "Mean for absolute
difference", col = "blue")
points(c(ABSDIFF3_1D, ABSDIFF3_2D, ABSDIFF3_3D), col = "red")
legend("topright",c("standardized with classic method", "standardized with convert method"), text.col = c("blue","red"), cex =
0.6)
# Compare the GOF value from 3 methods we used above in 3 different dimensions.
plot(c(GOF2_1D[1], GOF2_2D[1], GOF2_3D[1]), ylim = c(0.8, 1.0), xlab = "Dimension", ylab = "GOF", col = "blue")
points(c(GOF3_1D[1], GOF3_2D[1], GOF3_3D[1]), col = "red")
legend("topright",c("standardized with classic method", "standardized with convert method"), text.col = c("blue","red"), cex =
0.6)
# Calculate the correlation value for different methods in different dimensions.
correlation1_2D <- cor(check2_3D[, 1], classic)
correlation1_3D <- cor(check3_3D[, 1], convert)
correlation2_2D <- cor(check2_3D[, 2], classic)
correlation2_3D <- cor(check3_3D[, 2], convert)
correlation3_2D <- cor(check2_3D[, 3], classic)
correlation3_3D <- cor(check3_3D[, 3], convert)
format(round(max(abs(as.matrix(dist(check1_1D)) - as.matrix(distances1))), 3))
format(round(max(abs(as.matrix(dist(check1_2D)) - as.matrix(distances1))), 3))
format(round(max(abs(as.matrix(dist(check1_3D)) - as.matrix(distances1))), 3))
format(round(max(abs(as.matrix(dist(check2_1D)) - as.matrix(distances2))), 3))
format(round(max(abs(as.matrix(dist(check2_2D)) - as.matrix(distances2))), 3))
format(round(max(abs(as.matrix(dist(check2_3D)) - as.matrix(distances2))), 3))
format(round(max(abs(as.matrix(dist(check3_1D)) - as.matrix(distances3))), 3))
format(round(max(abs(as.matrix(dist(check3_2D)) - as.matrix(distances3))), 3))
format(round(max(abs(as.matrix(dist(check3_3D)) - as.matrix(distances3))), 3))
plot(c(ABSDIFF2_1D, ABSDIFF2_2D, ABSDIFF2_3D), ylim = c(0, 0.8), xlab = "Dimension", ylab = "Mean absolute
difference")
plot(c(ABSDIFF3_1D, ABSDIFF3_2D, ABSDIFF3_3D), ylim = c(0, 0.2), xlab = "Dimension", ylab = "Mean absolute
difference")
plot(c(max(abs(as.matrix(dist(check2_1D)) - as.matrix(distances2))), max(abs(as.matrix(dist(check2_2D)) -
as.matrix(distances2))), max(abs(as.matrix(dist(check2_3D)) - as.matrix(distances2)))), ylim = c(1, 1.8), xlab = "Dimension",
ylab = "Max absolute difference")
plot(c(max(abs(as.matrix(dist(check3_1D)) - as.matrix(distances3))), max(abs(as.matrix(dist(check3_2D)) -
as.matrix(distances3))), max(abs(as.matrix(dist(check3_3D)) - as.matrix(distances3)))), ylim = c(0.2, 0.6), xlab = "Dimension",
ylab = "Max absolute difference")
```

Define GOF that used in the code above:

Define GOF as the goodness of fit. The value of GOF will be calculated in two ways by R:

- In the first way, R will calculate GOF as $(\sum_{a=1}^{16} \lambda_a)/(\sum_{a=1}^{16} |\lambda_a|)$.
- In the second way, R will calculate GOF as $(\sum_{a=1}^{16} \lambda_a)/(\sum_{a=1}^{16} max(0,\ \lambda_a))$.

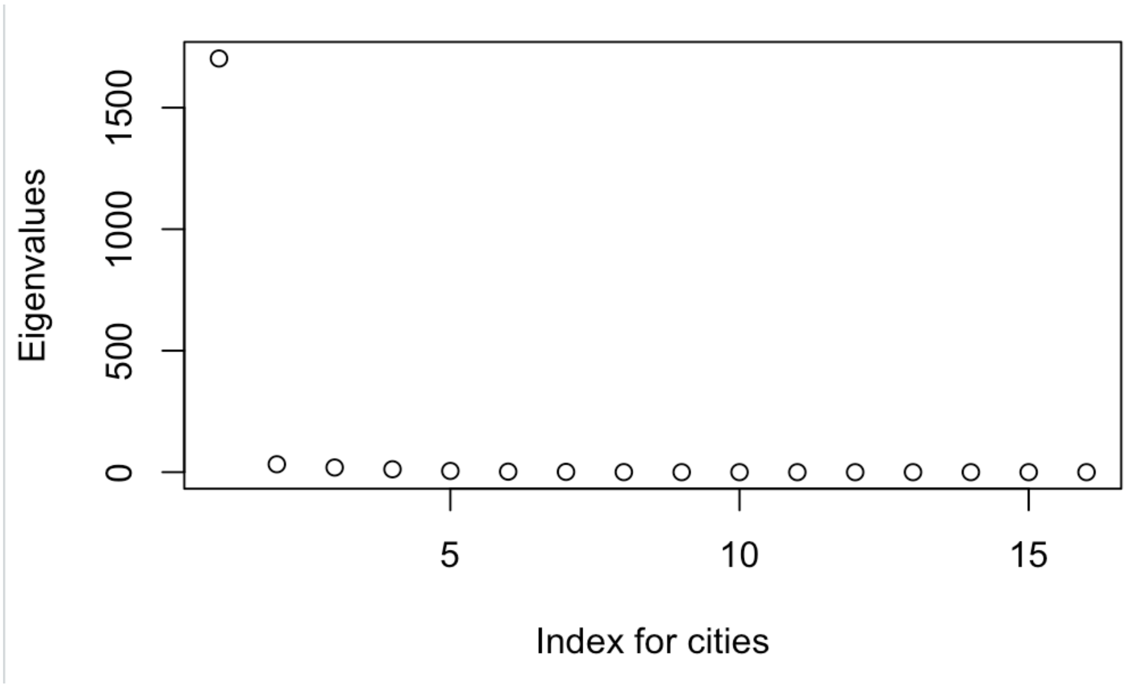
This is because R will handle negative eigenvalues in two ways for the calculation of GOF.

Now, I will move to collect the results I get from the output of the code above through R.

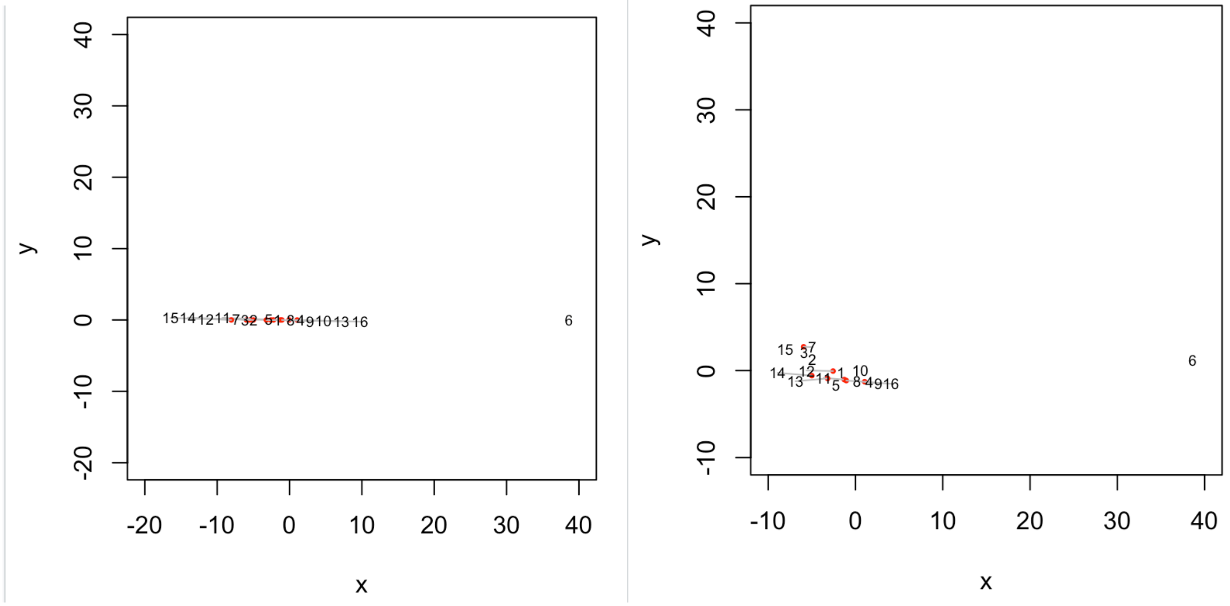
Output from the code above through R:

<u>Results I get from the models I create for matrix A:</u>

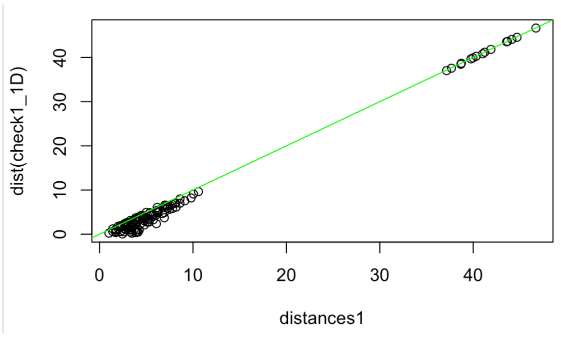
Below is the plot that represents the eigenvalue v.s. the index of objects:



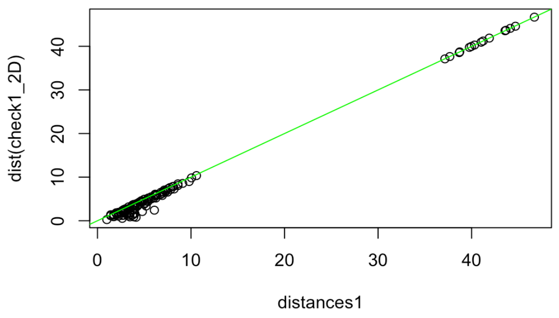
Below are the plots for 1 dimensional model and 2 dimensional model (the left plot is for 1 dimensional and the right plot is for 2 dimensional model):



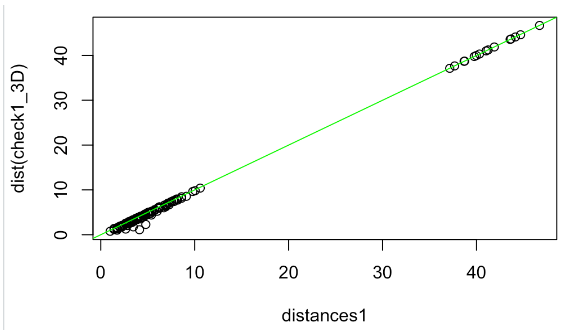
Below is the plot for distance in 1 dimensional model v.s. the actual distance from dataset:



Below is the plot for distance in 2 dimensional model v.s. the actual distance from dataset:



Below is the plot for distance in 3 dimensional model v.s. the actual distance from dataset:

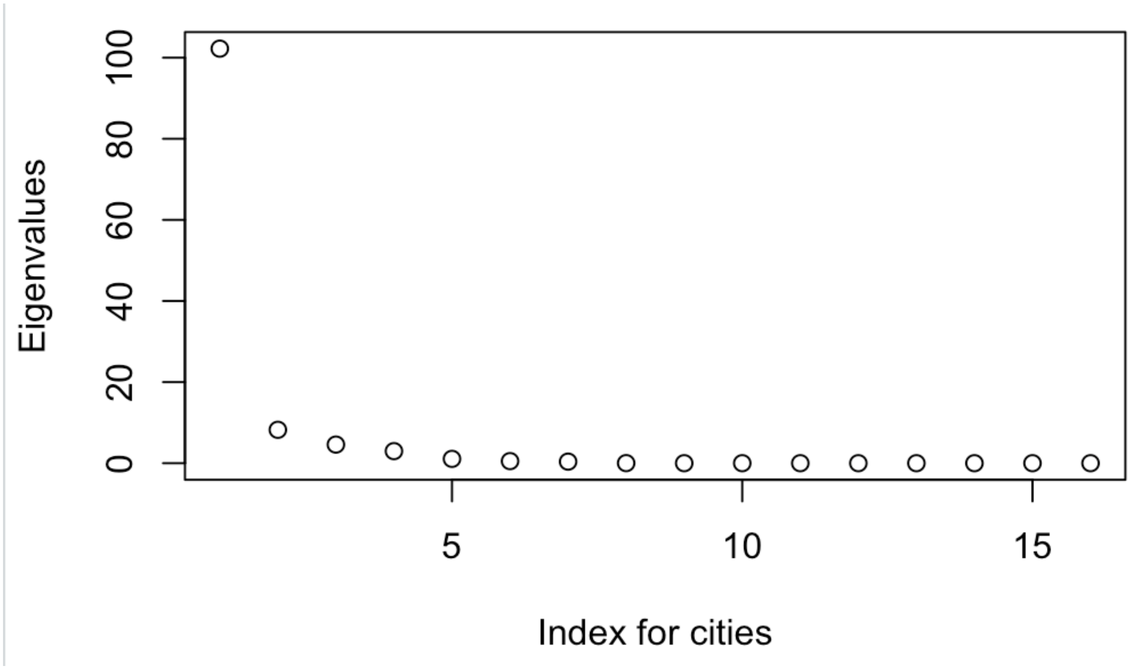


Below is the table for some important values I get from the code I write in R:

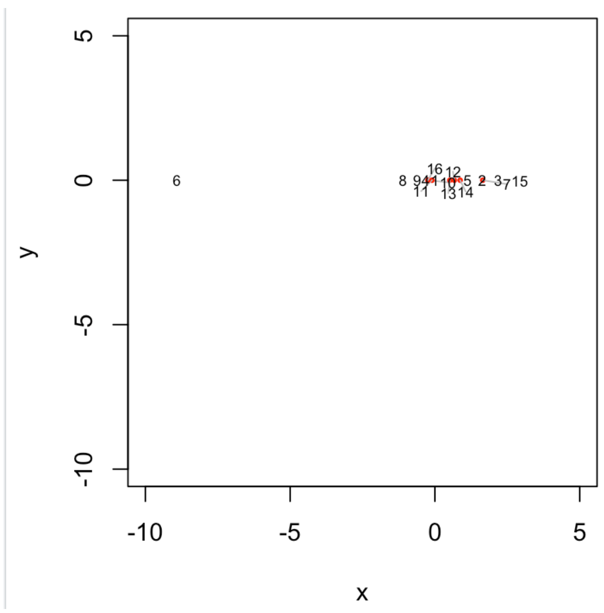
Dimension	1	2	3
Mean absolute difference of the entries	1.091	0.651	0.304
Maximum absolute difference of the entries	3.709	3.664	3.013
GOF	[0.959, 0.959]	[0.978, 0.978]	[0.988, 0.988]

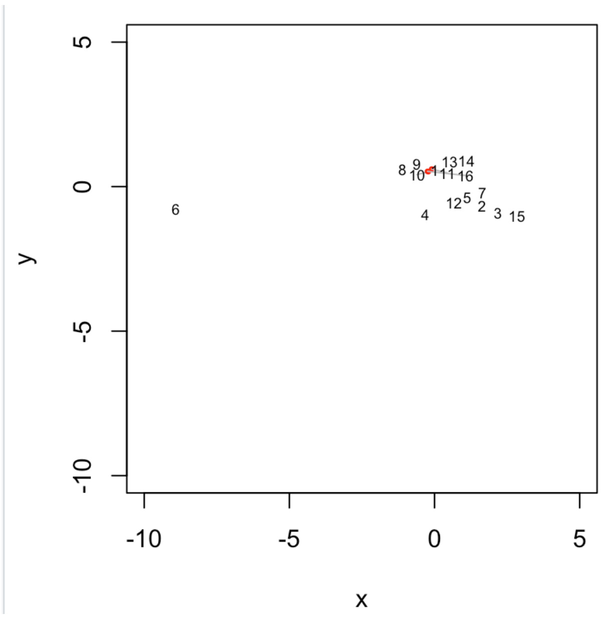
Results I get from the models I create for matrix B:

Below is the plot that represents the eigenvalue v.s. the index of objects:

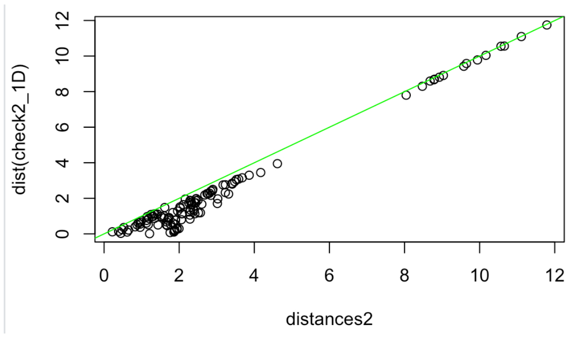


Below are the plots for 1 dimensional model and 2 dimensional model (the left plot is for 1 dimensional and the right plot is for 2 dimensional model):

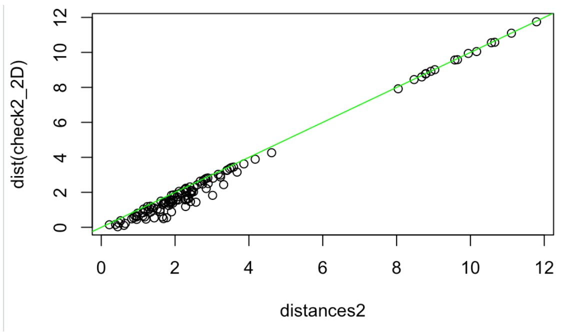




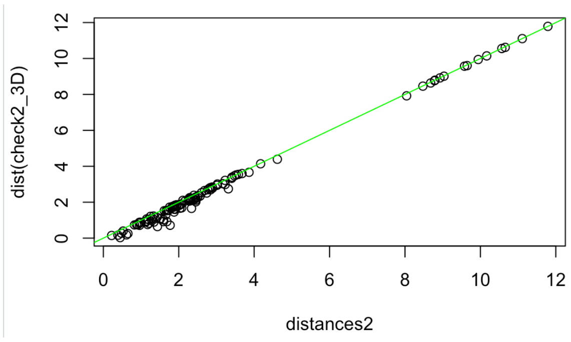
Below is the plot for distance in 1 dimensional model v.s. the actual distance from dataset:



Below is the plot for distance in 2 dimensional model v.s. the actual distance from dataset:



Below is the plot for distance in 3 dimensional model v.s. the actual distance from dataset:

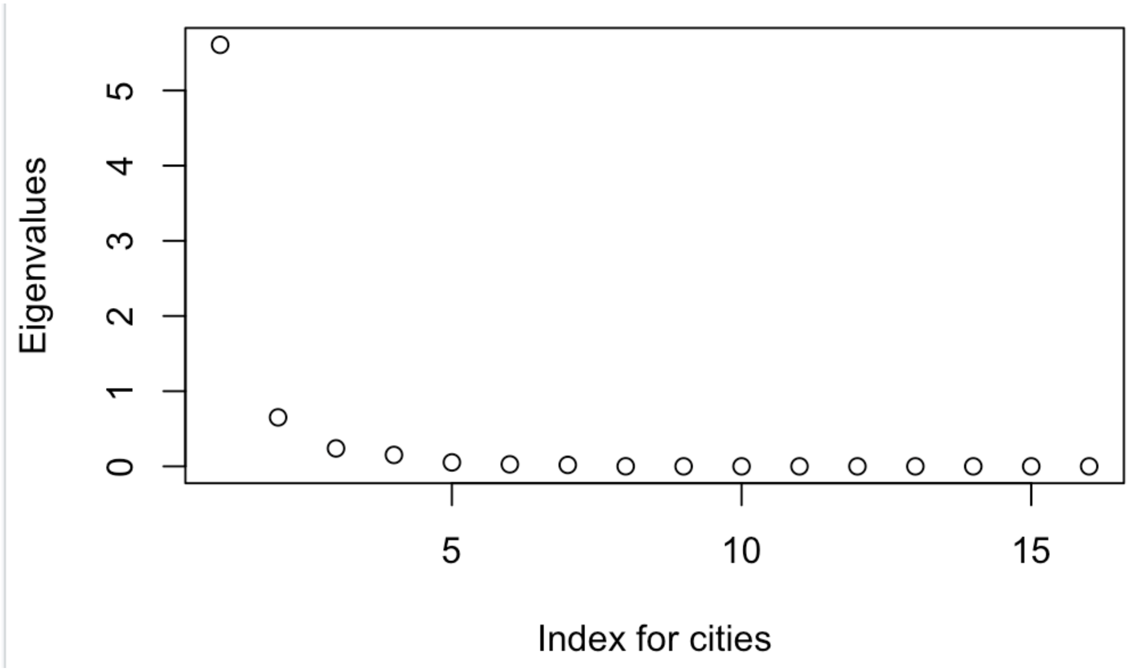


Below is the table for some important values I get from the code I write in R:

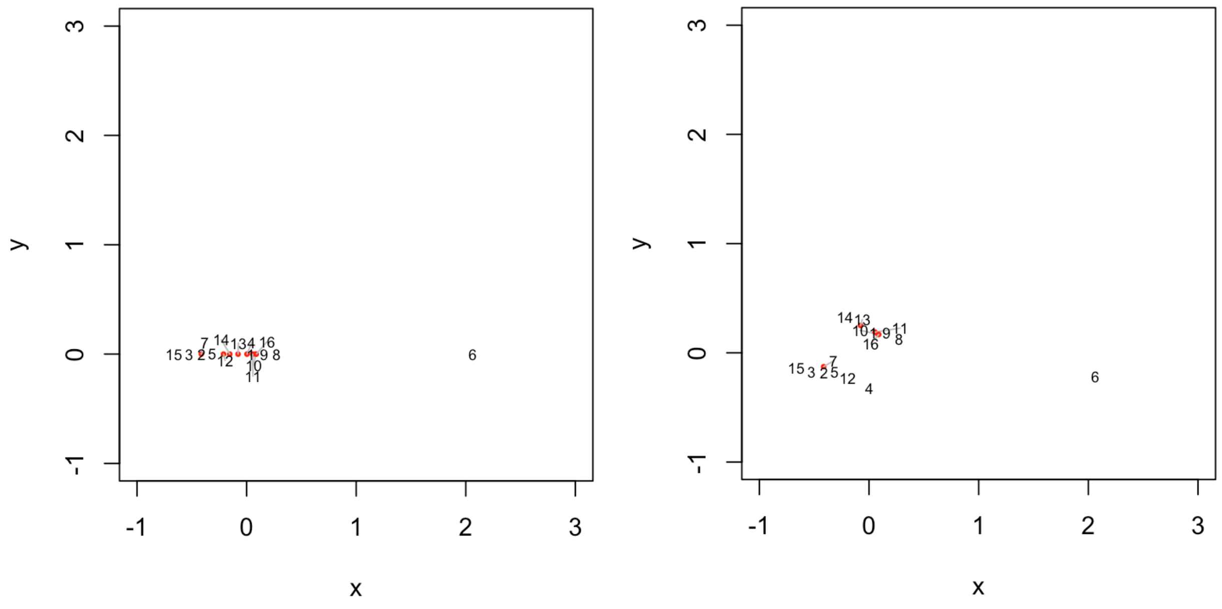
Dimension	1	2	3
Mean absolute difference of the entries	0.619	0.324	0.172
Maximum absolute difference of the entries	1.759	1.226	1.056
GOF	[0.852, 0.852]	[0.920, 0.920]	[0.959, 0.959]

Results I get from the models I create for matrix C:

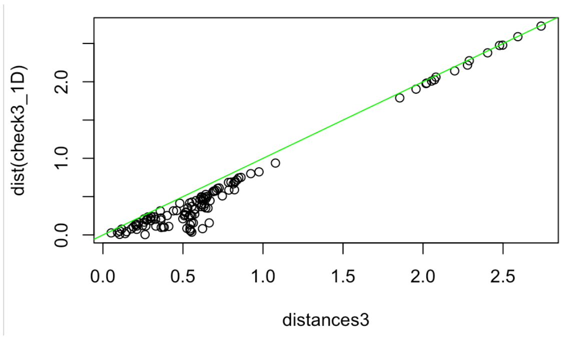
Below is the plot that represents the eigenvalue v.s. the index of objects:



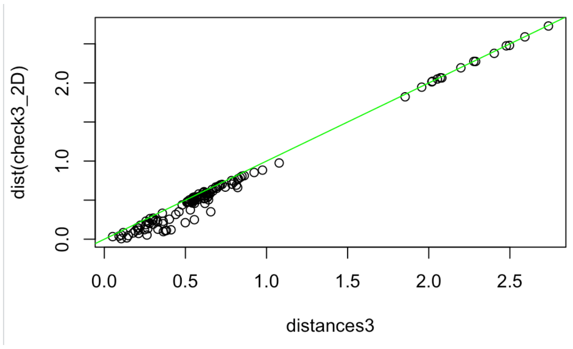
Below are the plots for 1 dimensional model and 2 dimensional model (the left plot is for 1 dimensional and the right plot is for 2 dimensional model):



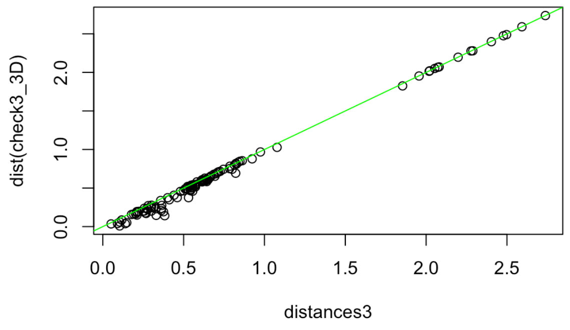
Below is the plot for distance in 1 dimensional model v.s. the actual distance from dataset:



Below is the plot for distance in 2 dimensional model v.s. the actual distance from dataset:



Below is the plot for distance in 3 dimensional model v.s. the actual distance from dataset:



Below is the table for some important values I get from the code I write in R:

Dimension	1	2	3
Mean absolute difference of the entries	0.156	0.069	0.037
Maximum absolute difference of the entries	0.541	0.306	0.242
GOF	[0.831, 0.831]	[0.927, 0.927]	[0.963, 0.963]

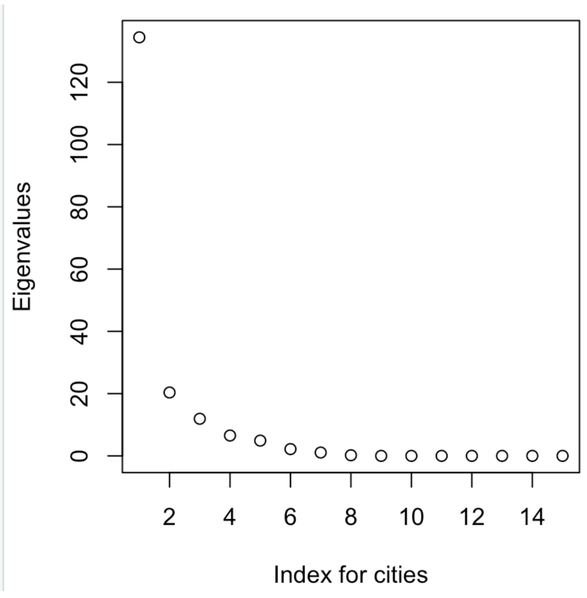
Analysis the result I get:

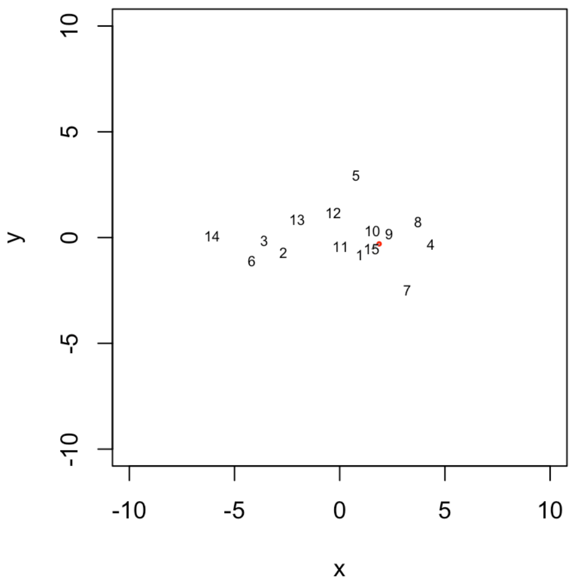
We could see that the data for object "6", which refers to the city Djakarta in 1961, is far away from all other datas. By checking my dataset, I realize that the expectation of life for all genders in different ages (0, 25, 50, 75) in the city of Djakarta in 1961 is smaller than other cities with the given year in my dataset. Thus, I will remove the row of Djakarta from the dataset, run through the code by R again, record the eigenvalue plots, 2-D plots, and tables for mean absolute difference, maximum absolute difference, and GOF.

Output from the code above through R after deleting the row of Djakarta from the dataset:

<u>Results I get from the models I create for matrix A:</u>

The left one is the plot for eigenvalues and the right one is the plot for 2 dimensional model:



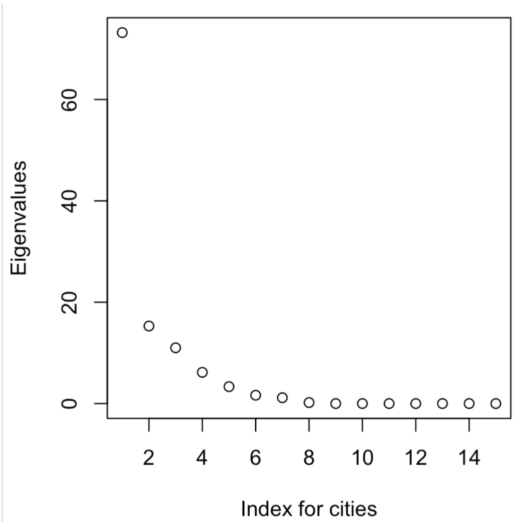


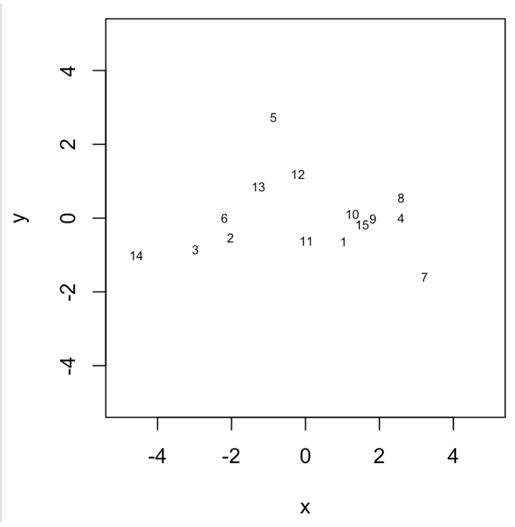
Below is the table for some important values I get from the code I write in R:

Dimension	1	2	3
Mean absolute difference of the entries	0.931	0.465	0.285
Maximum absolute difference of the entries	3.827	2.998	1.375
GOF	[0.740, 0.740]	[0.852, 0.852]	[0.918, 0.918]

Results I get from the models I create for matrix B:

The left one is the plot for eigenvalues and the right one is the plot for 2 dimensional model:



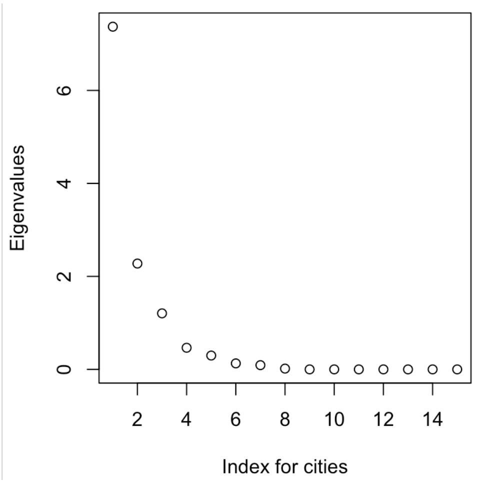


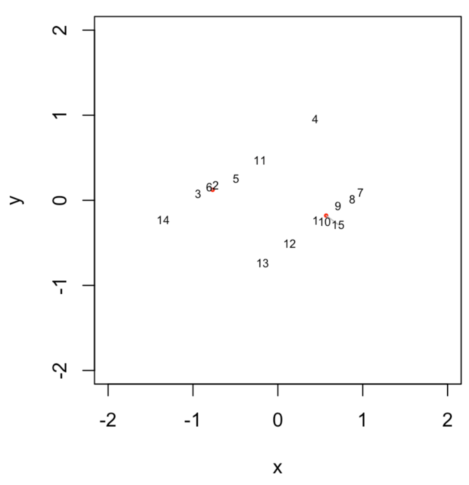
Below is the table for some important values I get from the code I write in R:

Dimension	1	2	3
Mean absolute difference of the entries	0.950	0.550	0.282
Maximum absolute difference of the entries	2.836	2.227	1.817
GOF	[0.654, 0.654]	[0.790, 0.790]	[0.888, 0.888]

Results I get from the models I create for matrix C:

The left one is the plot for eigenvalues and the right one is the plot for 2 dimensional model:





Below is the table for some important values I get from the code I write in R:

Dimension	1	2	3
Mean absolute difference of the entries	0.337	0.157	0.072
Maximum absolute difference of the entries	1.2605	0.819	0.531
GOF	[0.622, 0.622]	[0.814, 0.814]	[0.916, 0.916]

Also, I have to investigate the goodness of models by the results I get above.

It is easy to find out that every column in my dataset won't represent the expectation of life for the same gender and same age, so it is meaningless to analyze models that are not standardized. This means I will exclude matrix A from my code in the further analysis.

Recall that the row of Djakarta from the dataset is still removed from my dataset.

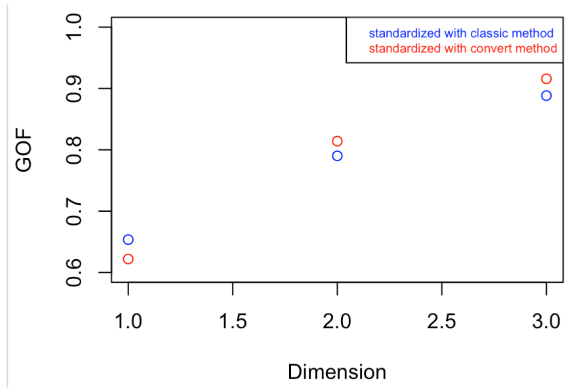
<u>Eigenvalues:</u>

- We see that in eigenvalue plots for matrices B and C, the first 8 indexes are non-zero. This means that my dataset would fit perfectly into 8 dimensional space.
- We see that in eigenvalue plots for matrices B and C, the first eigenvalue is much larger than the rest, so a one-dimensional model will capture some aspects of the original data.
- We see that in eigenvalue plots for matrices B and C, the eigenvalue becomes very tiny after the seventh index, so an seven-dimensional model is close to perfect for my dataset.

<u>GOF:</u>

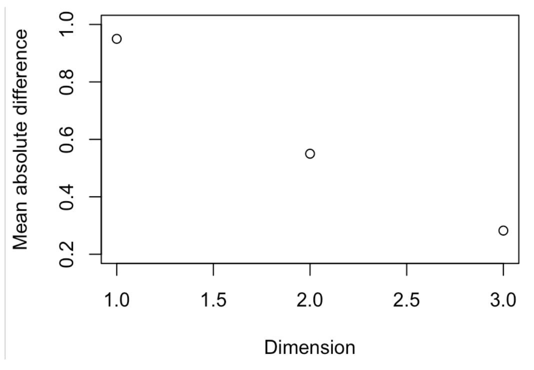
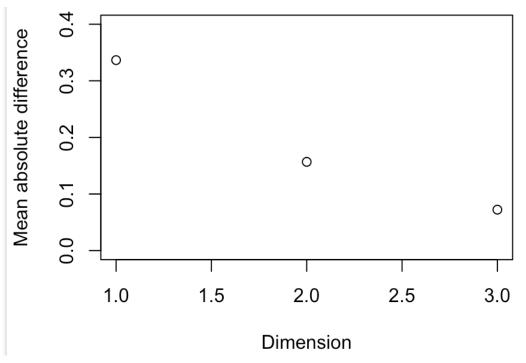
- The models don't fit the standardized dataset that well in one dimension but fit well in two and three dimensions since the GOF for both models in one dimension is relatively smaller than the GOF for both models in two and three dimensions.
- The value of GOF for both models becomes larger when dimension is getting larger, since a model includes more attributes when the dimension of the model becomes larger.

- The output of two GOF values for one of the models above in all dimensions from the code are the same, since all eigenvalues in both models are greater than 0, which means the effect of negative eigenvalues on GOF is 0, since there isn't any negative eigenvalue.
- Below is a plot for the comparison of GOF for the model that is standardized by subtracting the mean of the column and then dividing the standard deviation of the column (in blue) and the model that is standardized by subtracting the minimum value in the column and then dividing the range of the column (in red) in different dimensions:

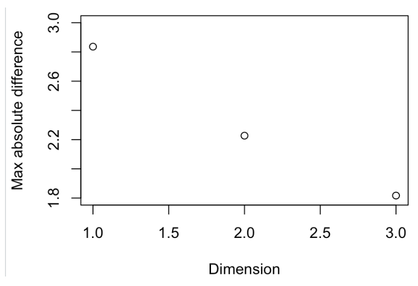
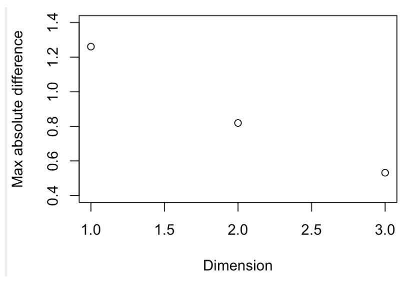


<u>Comparison of mean absolute difference among different dimensions from matrix B and C:</u>

- I write the code to calculate the mean absolute difference as the mean of the absolute difference between the distance I get by the model I created and the actual distance from my dataset through R. See the plot below:

Mean absolute difference for matrix B	Mean absolute difference for matrix C
	

- By the plot above, we could see that the mean absolute difference for both models is decreasing when dimension is increasing, which means that the distance matrices for both models are getting similar to my input distance matrix.
- I also write the code to calculate the maximum absolute difference as the maximum of the absolute difference between the distance I get by the model I created and the actual distance from my dataset through R. By the plot below, we could see that the maximum absolute difference for both models is also decreasing when the dimension is increasing.

Maximum absolute difference for matrix B	Maximum absolute difference for matrix C
	

Finally, I will try to assess what the dimensions in my model mean. I will compare each dimension of the model against each dimension of my input data and do this by calculating the correlation coefficient. See the table below for more information:

Column	X coordinate with matrix B	Y coordinate with matrix B	Z coordinate with matrix B	X coordinate with matrix C	Y coordinate with matrix C	Z coordinate with matrix C
1	-0.864	0.366	0.157	-0.820	-0.403	0.340
2	-0.812	0.392	0.207	-0.734	-0.410	0.297
3	-0.764	0.616	-0.104	-0.766	-0.160	0.589
4	-0.606	-0.036	-0.780	-0.772	0.634	0.041
5	-0.866	-0.339	0.271	-0.793	-0.399	-0.382
6	-0.840	-0.246	0.074	-0.811	-0.189	-0.292
7	-0.851	-0.413	0.066	-0.803	-0.216	-0.460
8	-0.832	-0.279	-0.125	-0.811	-0.078	-0.296

According to the table above, we see every absolute value of correlation coefficient is larger than 0.6 for both models in x-dimension for all columns. Thus, I assume that models in x-dimension are correlating to the life expectations of both genders at the age of 0, 25, 50, and 75 very well.

We could also see that models in y-dimension correlates with column 2 somehow, since only column 2 has an absolute value of correlation coefficient that is close to or larger than 0.4 for both models. Thus, I assume that the models in y-dimension are correlating to the life expectations of males that are at the age of 25, but the correlation is worse than the same correlation we have for models in x-dimension.

We could also see that models in z-dimension only correlate with column 5 somehow, since the absolute values of correlation coefficient for all other columns are too small for both models. Thus, I assume that the models in y-dimension are correlating to the life expectations of females that are at the age of 0, but the correlation is much worse than the same correlation we have for models in x-dimension.