Application of Multidimensional Scaling on the Country Food Consumption to Explore Dissimilarity Patterns

Group1:

Pengfei He

Seoyoung Park

Adavya Bhalla

MATH 381: Discrete Modeling

Dr. Matthew M. Conroy

March 8, 2019

Introduction:

This project uses Multidimensional Scaling (MDS) to model the dissimilarity of food consumption among a large subset of countries worldwide. We also incorporated Principal Component Analysis (PCA) to find the most important dimensions for our main model and some variations. The project's objective is to find any (dis)similarity patterns among the average food consumptions per person from different countries. After generating an MDS representation, we observe the distributions of country and do researches to prove the our observation. If our observation is a possibly valid pattern, we further create graphical representations for each type of food consumption to explain the phenomena. In the end, we found several interesting patterns about fruit consumption, tourist countries, etc.

Background:

Initially, we looked up the paper list on MATH 381 course website and found a blog post about applying MDS on Indian food.[1] After reading it, we wanted to do the project using the same dataset, but the dataset they used is not accessible outside India. Also, since Professor Conroy advised us to do something similar but with areas other than India, we were looking for other datasets. Later on, we found a dataset of food consumption worldwide by country[2] and the

[1] "Seeing India through Food – An Experiment in Multidimensional Scaling," R-bloggers, January 07, 2015, , accessed March 06, 2019, https://www.r-bloggers.com/seeing-india-through-food-an-experiment-in-multidimensional-scaling/.
[2] Dor Oppenheim, "Who Eats the Food We Grow?" RSNA Pneumonia Detection Challenge | Kaggle, November 30, 2017, accessed March 06, 2019, https://www.kaggle.com/dorbicycle/world-foodfeed-production.

population of those countries[3] which allow us to explore the pattern of average food consumption per person from different countries by MDS in a wider range.

From the paper *The Past, Present, and Future of Multidimensional Scaling,*[4] we conclude the development of MDS over the last four centuries. The earliest MDS can be found in geography in 17th century. In the map of Durham county by Jacob van Langren, about a half of the map is a table of distances among several towns and villages in Durham county, England, and the other half of the map is the geographical map corresponding to those distances. This is regarded as the first example of showing not only the table of distances but also the map in one figure, and this can be considered as a forerunner of MDS.

After that, MDS had been used as the distance formula that is modeled for (dis)similarity judgments in psychology. People generally made (dis)similarity judgments for couple of things in a process that jointly simulate the natural distance function in a Cartesian space; the person makes the objects' mental representation in psychological space, when he or she needs to make a judgement. The psychological space here is the space that is expanded by the characteristics of the objects and with the objects corresponding to points in the coordinate system. This space is unknown to outsider, but Kruskal, one of the MDS developers, claimed that MDS discloses it from the one's overall (dis)similarity judgments. Firstly, the restrained MDS takes the ratings on a metric scale from given judgments and the Euclidean distance, the distance function of the psychological space. Afterward, the Euclidean distance formula was generalized to Minkowski metric, and some researcher started to investigate the suitability of Minkowski metric under different contexts. Besides the similarity model, researchers switched

[3] Prasanna Nadimpalli, "World Countrywise Population Data 1980 - 2010," RSNA Pneumonia Detection Challenge | Kaggle, August 30, 2017, , accessed March 06, 2019, https://www.kaggle.com/dataswimmer/population19802010/version/1.

[4] Patrick J.F. Groenen, "The Past, Present, and Future of Multidimensional Scaling," July 2013, , accessed March 6, 2019.

their attention to dissimilarity model, and it led to a wide range of applications, especially social science. [5]

Further, MDS is used to solve Coombs' unfolding model where data are preferences of different individuals towards objects rather than (dis)similarity, and this new type of research advanced the understanding of psychology greatly. In the late 1960s, researchers turned data into non metric ones and improved he measurement. Later on, to solve the scaling problem, there are a "Shepard diagram" and an ordinal MDS algorithm invented. The later one is utilized more frequently in the research of the relationship between the MDS space and the properties of the objects. With the development of the ordinal MDS, it is soon used in many areas, such as psychology, market research and sociology. Compared to confirmatory MDS which is hard to put external constraints, Schwartz was content-driven and looked for the correspondences of content theory about the MDS objects and their representation in space. [6]

We also analyzed the project inspired us. The government of India, the National Sample Survey Office (NSSO), reveals the 'Household Consumption of Various Goods and Services in India' that has the data of the per capita of consumption in one month for every state in India. To see how the food consumption shows the patterns among the states, MDS is applied in the report. The figure of the results illustrates that MDS brings good analysis of the relative locations of the different states, which also gives an idea that nearby states have similar geographic and cultural characteristics, even if the data was based on food.[7] After comparing it with out project, we found that it's similar to our second variation where the MDS is partially based on its real geographical distance. People living in the same continent tend to share the similar food

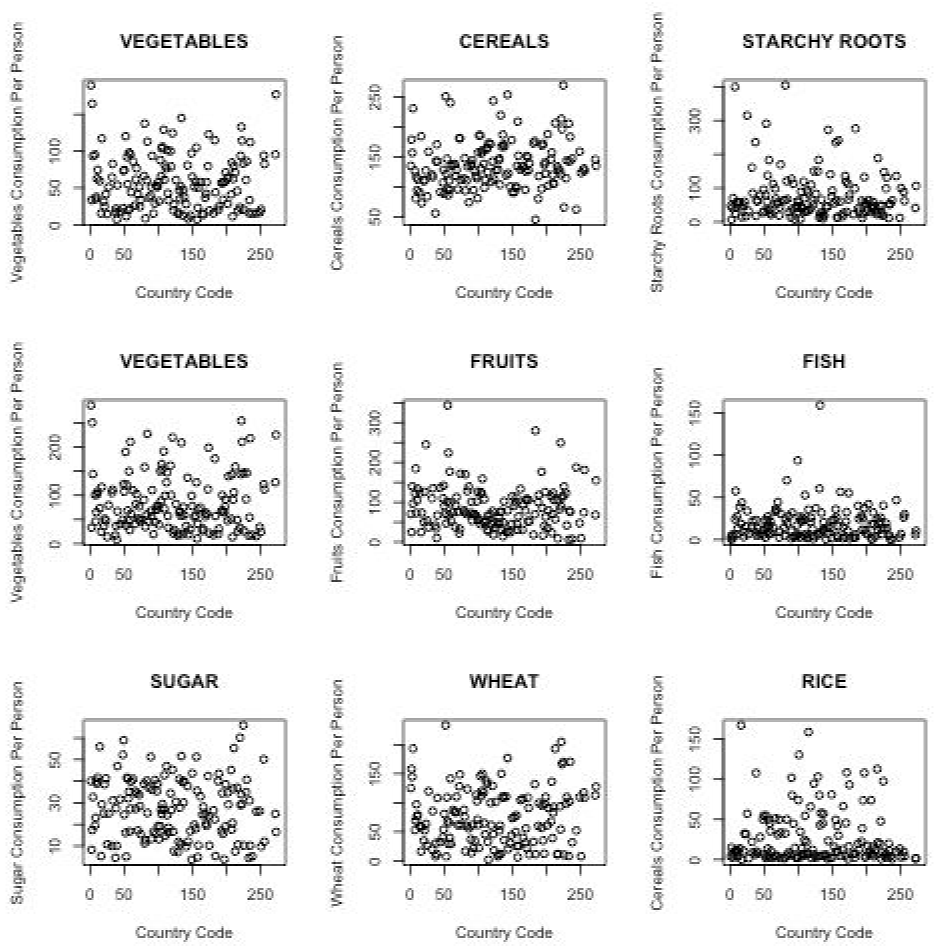
[5] Ibid
[6] Ibid
[7] *Seeing India through Food*

consumption pattern, and people in those countries tend to cluster together just like Indian cities in the Indian food project.

Model:

Our dataset from Kaggle is rather large and initially included 117 different food consumption values for each country. However, it only has total food consumption instead of the average food consumption per person. This dataset doesn't have population either. Therefore, we decided to merge the dataset with another population dataset, and divide each total food consumption by its population to get the average food consumption per person. Later on, we thought performing MDS for a 117-dimensional space to a 2D space would potentially cause a lot of information to be lost and might not yield the best result (in regard to goodness of fit). Therefore, we first performed PCA on our dataset to extract the 9 most important dimensions, the food consumption value dimensions that have the most variance, for our dataset: vegetables (item code 2605), cereals (2905), starchy (2907), vegetables (2918), fruits (2919), fish (2960), sugar (2542), wheat (2511), and rice (2805). Also, because of the large volume of the dataset, we don't have ability to make objective assumptions before making it. Therefore, we chose to explore patterns by applying several methods such as in adjustments and extensions.

Once we had these different dimensions we used this 9 dimensional space created for our model. We used two types of distances between countries, Manhattan, Euclidean in this 9 dimensional model to create our 2d model. We used euclidean distances in our first model. For example, we took 9 different food consumption values from each country. We did not normalize the data to see the effects of normalization later. Later we normalized the data to have a mean of zero and a standard deviation of 1 for the values of consumption.



For the calculation of goodness of fit, we used the formula,

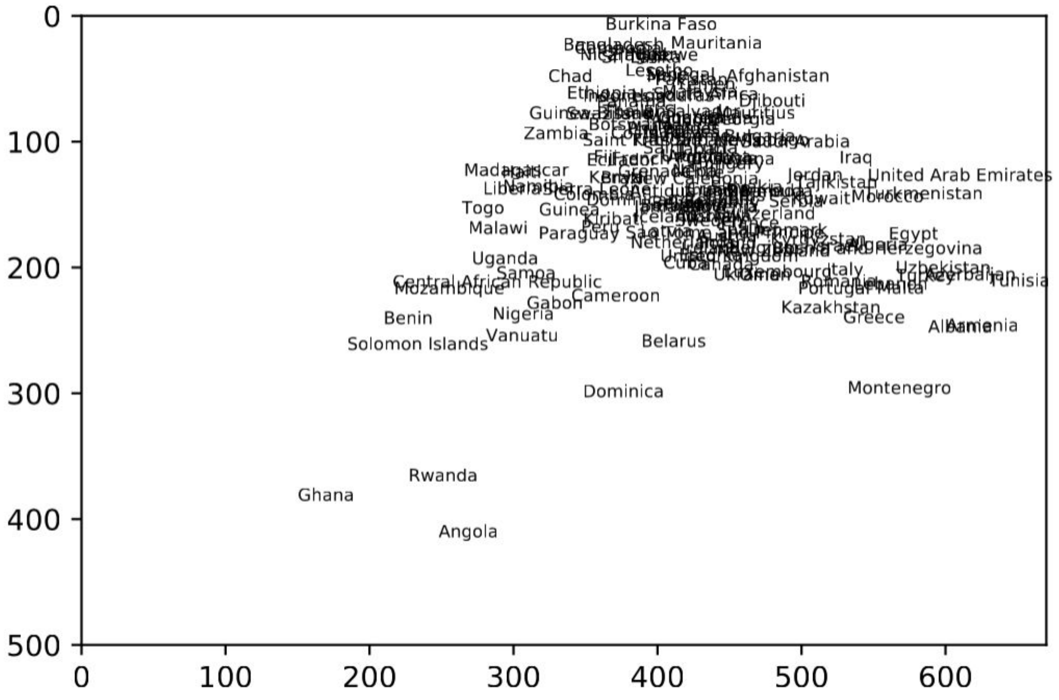
$$stress = \sqrt{\frac{\sum\left(d_{ij} - \hat{d}_{ij}\right)^2}{\sum d_{ij}^{\,2}}}$$

where d_{ij} is the real distance, and $\hat{h}_{ij}$ is the estimated distance by MDS. The smaller the value of the goodness of fit is, the better MDS fits. [8]The goodness of fit is manually calculated in code.

[8] Joseph B. Kruskal and Myron Wish, Multidimensional Scaling (Newbury Park, CA: Sage Publ., 2009), 435.

Results:

By applying PCA, we found a good percentage of the variance in our dataset was in the “exotic vegetable” and “exotic fruit” categories. To be more exact, not all countries have rare vegetables and fruits such as durian, dragon fruit, etc. The quantity of those types of food varies widely as some countries produce a lot of these unique fruits and vegetables while some countries have nearly none. After investigating the dataset, it’s partially due to a dataset bias. Since this dataset was gathered in the US fruits commonly found here are listed whereas fruits popular in other regions are lumped into one category.



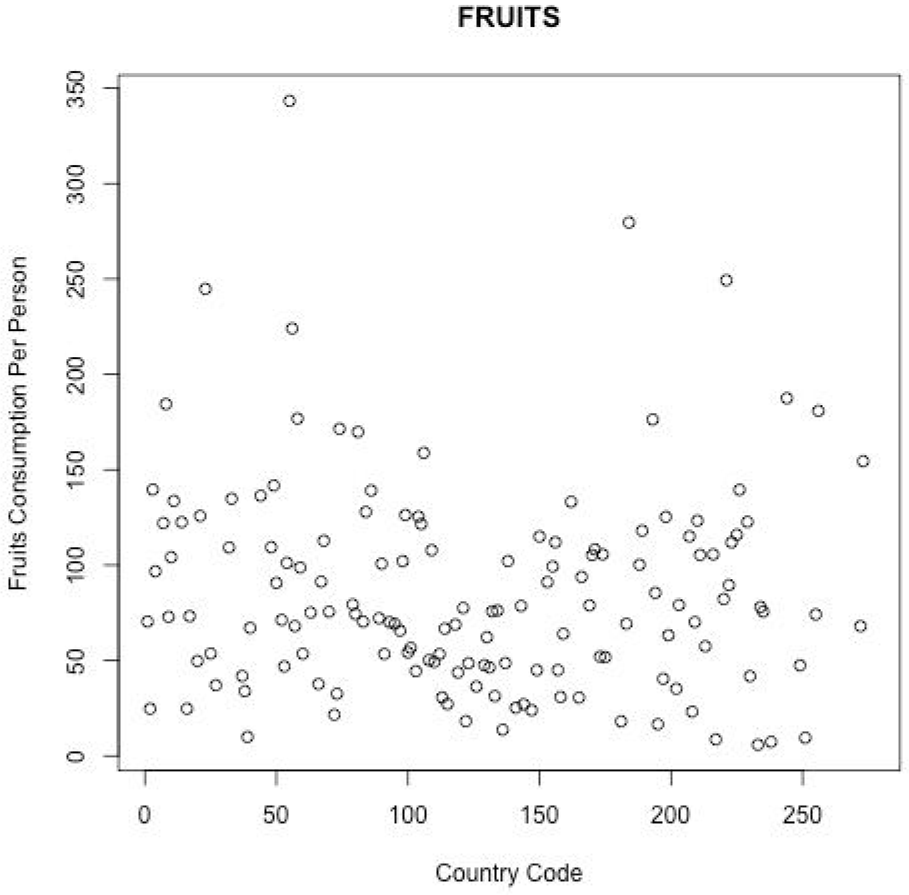
The y-axis seems to be the amount of unique fruits and vegetables produced. The x-axis, on the other hand seems to follow the average consumption of vegetables and cereal in a country. The more cereals and vegetables a country produced, the more the country is to the right in our graph. After we look into the top 5 food consumption for vegetables and cereal. We found that Montenegro and Greece are high in vegetables consumption. Egypt and United Arab Emirates are high in cereal consumptions. All the countries are on the right of the graph.

We also found that the top five starchy roots consumptions countries are Ghana, Angola, Solomon Islands, Benin, Rwanda which all clustered on the southwest of the center. We don't observe any dimension it relates to since the alignment of those countries on the MDS graph doesn't follow the order. However, it indicates their consumption of starchy roots are significantly higher than that of the other countries.

Table. Top Five Food Consumption Countries for Nine Food

	Vegetabl es	Cereals	Starchy Roots	Vegetabl es	Fruits	Fish	Sugar	Wheat	Rice
1	Armenia	United Arab Emirates	Ghana	Armenia	Dominica	Maldives	United Arab Emirates	Azerbaija n	Banglade sh
2	Montene gro	Morocco	Angola	Tunisia	Rwanda	Iceland	Trinidad and Tobago	Tunisia	Cambodi a
3	Albania	Azerbaija n	Solomon Islands	Albania	Oman	Kiribati	Cuba	Algeria	Indonesia
4	Malta	Lesotho	Benin	Greece	Belize	Malaysia	Barbados	Turkmeni stan	Thailand
5	Bosnia and Herzegov ina	Egypt	Rwanda	Montene gro	Dominica n Republic	Antigua and Barbuda	Switzerla nd	Morocco	Philippine s

We also see a trend in our model that countries with a large amount of tourists compared to their population tend to appear in the upper right hand corner. We think this is because the food consumption bundles are artificially inflated for these countries. This artificial inflation is a result of the high percentage of tourists since the food is added to the country's consumption while the population is still small with a large amount of tourists. These countries are United Arab Emirates(UAE), Mauritius, Burkina Faso, Mauritiana. Other than that both our axes were somewhat impacted by sugar and fish. Separating countries with coastal ties from other countries. Like, Ghana, Togo, Angola, from countries like Afghanistan and Iraq. We see countries like UAE in the top right corner with other non-coastal countries because of their small population. The difference in exotic fruit and grain production is very high per capita due to an abundance of tourists similarly fish consumption is high but, the high, fish consumption has an opposite effect than the other 2 consumptions i.e high consumption is towards bottom right. But, the other dimensions overpower the fish consumption and UAE is placed top right even with high fish consumption.



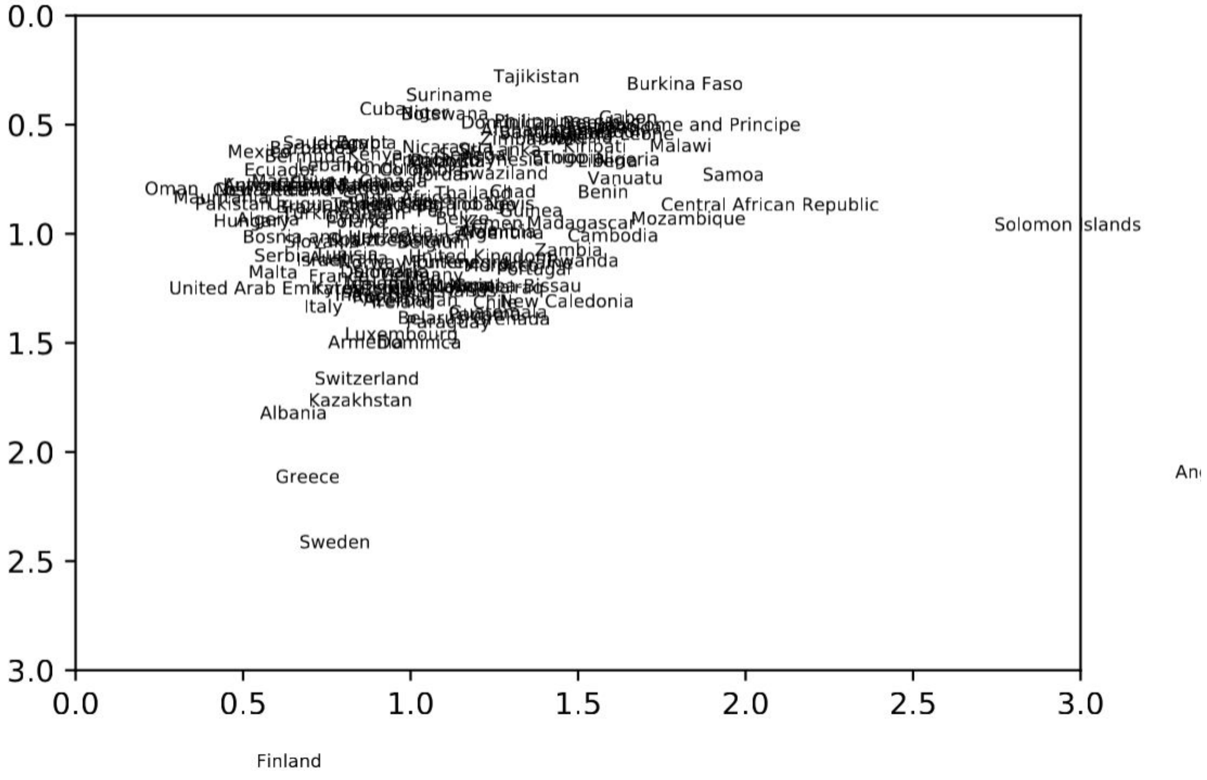
The graph above shows the great variance of fruits among different countries. The x-axis and y-axis are described as the graph shows, and every country is shown by its country code for clarity. Notable exceptions with really high consumptions are fruits like mango and watermelon, which if they are separated from rare fruits like durian, and others would probably lead to better variance among nations, a more complete dataset therefore, might yield wider variations and therefore a better 2-dimensional representation.

The goodness of fit for the model is 0.05, which relatively good. Also, we found out some patterns. Based on those, we conclude our result is realistic.

Adjustment and extensions:

The first variation:

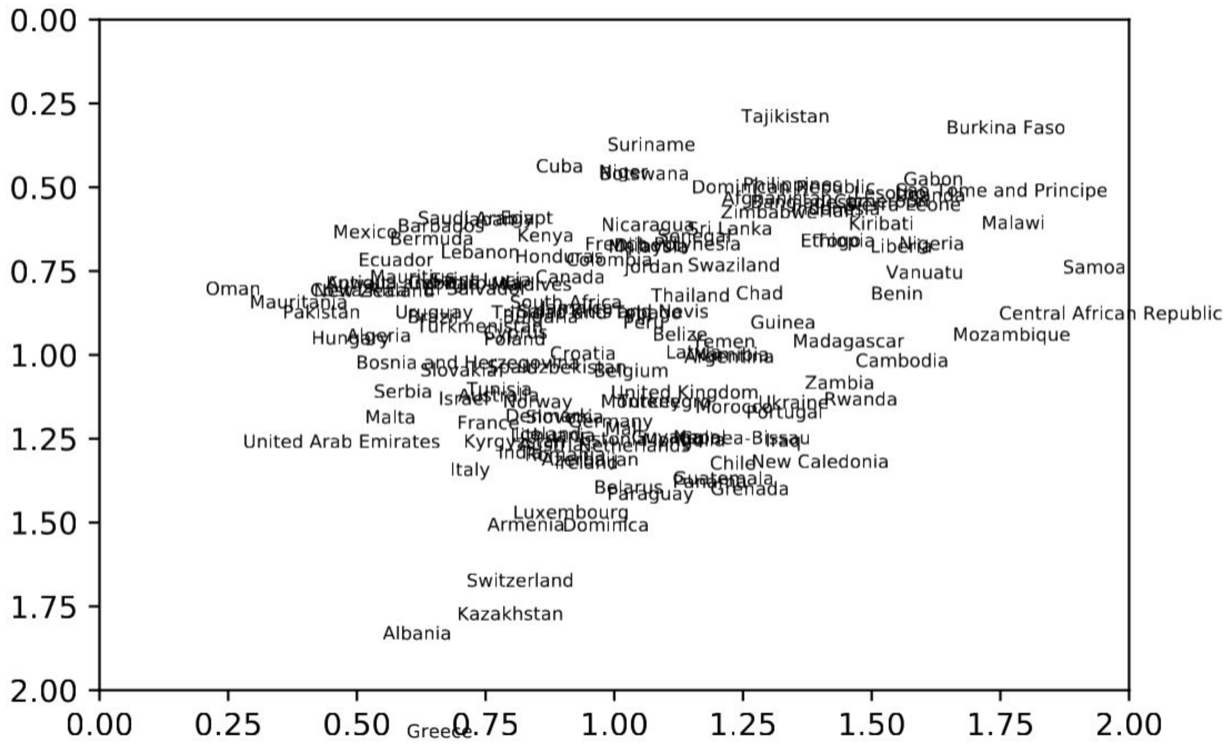
In our first variation, we tried to include as many food dimensions in our model as possible. Therefore, we used 117 different food items to capture as much information as possible. We did not use normalization techniques neither as we did in the first one because we were curious which food category might dominate. In this model, we considered differences from every food category instead of concentrating on ones that were chosen by our PCA analysis. This yielded further interesting results as our y-axis for our variant model is flipped from the original more PCA driven analysis.



From the graph above, we see countries line up very similarly in the y-axis, with countries with higher average consumption of unique fruits appearing towards the bottom of our representation. This variation changes the x-axis from our previous model. In this representation instead of grains and rice dominating the x-axis we believe the x-axis more lines up with meat consumption as countries towards the right have higher meat consumption in this model. This can be seen clearly because in the first model more asian and middle eastern countries differentiate themselves, countries with high grain consumption like Afghanistan, Bosnia and Herzegovina. Whereas in the variant model we see European countries differentiate themselves because of high meat and animal product consumption along with grains. The tourism effect is still seen with highly popular travel destinations like UAE, Greece and Switzerland standing out. The last 2 were not as far out on the x-axis from other countries in our original model but, we believe were able to differentiate themselves because of the added meat and animal product dimensions. Since their meat and animal product consumption are really high for their population. Using the same function we used earlier, goodness of fit is 0.1022 for this variation, which is higher than that of the original model. The goodness of fit changes here drastically because we use so many more dimensions in this variation. Therefore, we think the first variation is less realistic than the first one.

The second variation:

The second variation was same as the first variation except for newly added normalizing techniques. Here we saw a pretty big change since standardizing allowed us to differentiate countries more along the lines of geographical separation.



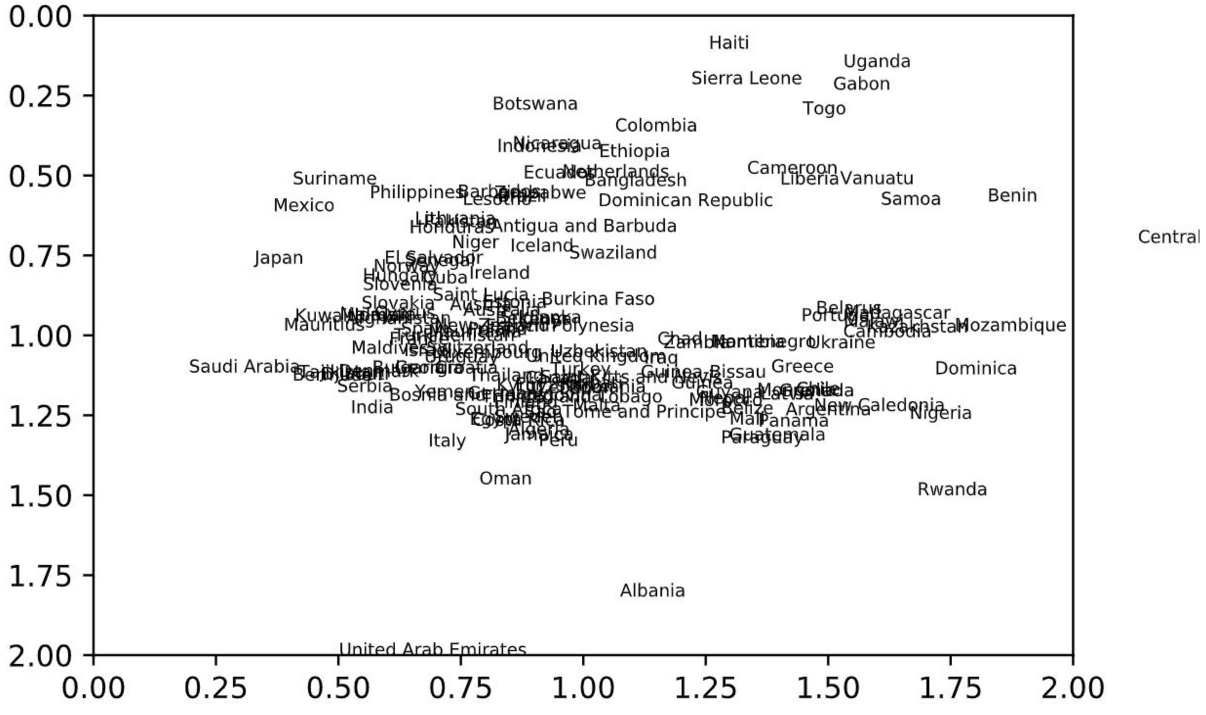
In this representation, most countries are not just clustered together but clustered more according to their real geographical distance. African countries are seen towards the upper right corner while most European and Middle Eastern countries are in the lower left hand corner. The axes for this model are not as aligned with the exotic fruit and vegetable categories. There are two reasons. First, it’s because we applied the normalization. Second, the exotic fruit category contained high amounts for certain countries that might have swayed our previous unnormalized model. This model is much more in line with food consumption and number of food items. Countries like Greece, UAE and Switzerland which have really high average food consumption (three of the most visited countries in the world, tourism definitely plays a factor here.) are at the bottom whereas countries with low average food consumption like Tajikistan, Burkina Faso and Suriname appear at the top. The x-axis on the other hand tells us how many different food items the country consumes. With Mauritiana, Oman and UAE on the extreme left these countries consume a lot of different foods, whereas Central African Republic, Mozambique and Samoa have only a few categories that they consume. Goodness of fit is 0.03, and hence this has a better fit because of the standardization of variables.

The third variation:

The third variation was our original vanilla model with 9 dimensions with its covariates standardized. See original description for original model.

Standardization Function : $\frac{x - \mu}{\sigma}$,

where x = real value, μ = mean value, and σ = standard deviation

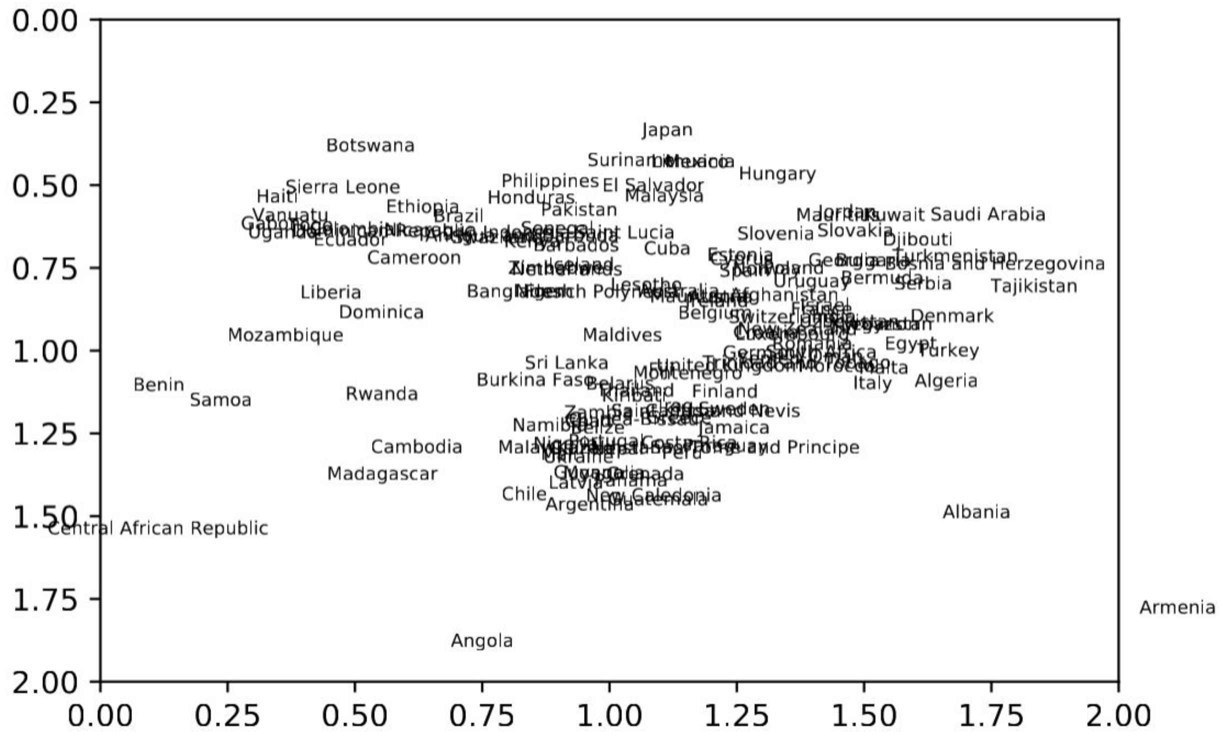


Compared to the plot of the original model, the clustered portion moves to the middle side. In the original model, Angola, Ghana, and Rwanda are far apart from the clustered portion, but in this model, they are more like to be in the clustered portion since the covariates are standardized. The x-axis seems to be grain consumption this time with countries like Saudi Arabia and Italy which have high grain consumption towards the left side whereas Dominica and Benin are towards the right side with low grain consumption. The y-axis seems to be vegetables(both 2605 and 2918) consumption. The countries on the top like Colombia and Dominican Republic have low value of the vegetables consumption per person, while the countries on the bottom like Oman and Albania have high value of the vegetables consumption per person. Goodness of fit is 0.01, which is the lowest, because 9 dimensions that correspond to the foods with most variances are used, and they are also normalized which will lead to even better fit.

The fourth variation:

In this variation, we use “TaxiCab” distances shown as below instead of Euclidean distances.

$$d_{ij} = \sum_{m} |r_{i,m} - r_{j,m}|$$



After generating the MDS graph above, we first looked some extremely placed countries. For example, central african republic and benin are relatively poor country. The wikipedia article on Benin(https://en.wikipedia.org/wiki/Benin_cuisine) says:

“Meat is usually quite expensive, and meals are generally light on meat and generous on vegetable fat.”

After researching the food fact of angola, we found that the country report on Angola (https://www.countryreports.org/canada/Angola.htm) says:

"Cassava is the most important food in Angola".

And cassava is basically starch roots. Later on, we looked up its average consumption value for each person is 59.25 kilograms which is relatively high compared to other countries. However, on its opposite side on y-axis, Japan also has a high value on that which is 43.678 kilograms. Goodness of fit is 0.08 which is not as good as the last one. Therefore, the axes are very ambiguous, and we can hardly find connections in the end.

Conclusion:

Based on what we have achieved, we made several conclusions. First, the value goodness of fit will affect interpretation of the graph significantly. For example, in third variation, we have achieved 0.01 goodness of fit, and correspondingly the axes are more easily interpreted. Second, there is a limitation of observation, especially when there are high dimensions and countries. When we tried to conclude a pattern, we normally starts with MDS graph, and then make objective assumptions based on our intuition. However, there might be some implicit patterns which cannot be easily seen from MDS, especially when the goodness of fit is high. Third, even though we didn't make assumptions at the very beginning, our hypotheses in our mind are confined to outcomes like similar food consumption in each continent. Unexpectedly, we discovered some tourist countries with high food consumption per person, and it taught us to explore more variations. Fourth, we didn't intend to use PCA to filter out dimensions at first. Given the high dimensions of the dataset we found, we later decided to use this technique to find the most important dimensions. When we tried out our initial thought without PCA in our first variation, it did yield a worse result with higher goodness of fit (bad according to our metric) due to too many dimensions. From this, we learned that preprocessing dataset is also important for the MDS project.

Another important factor in our project was normalization. We first implemented MDS without normalization because we were curious to see which food groups might dominate. We then found out about the “Other” fruits and vegetables categories which completely dominated this model due to a very high variance among countries. This also told us about how this model can be improved either including more fruit categories to have a more complete dataset or normalize this to create a somewhat equal dataset without gathering more data. We normalized this dataset and the results gave us a more easily interpretable model because we removed the heavy influence of the “other” fruits and vegetables.

References:

1. "Seeing India through Food – An Experiment in Multidimensional Scaling." R-bloggers. January 07, 2015. Accessed March 03, 2019. https://www.r-bloggers.com/seeing-india-through-food-an-experiment-in-multidimensional-scaling/.
2. Groenen, Patrick J.F. "The Past, Present, and Future of Multidimensional Scaling." July 2013. Accessed March 6, 2019.
3. Oppenheim, Dor. "Who Eats the Food We Grow?" RSNA Pneumonia Detection Challenge | Kaggle. November 30, 2017. Accessed March 08, 2019. https://www.kaggle.com/dorbicycle/world-foodfeed-production.
4. Nadimpalli, Prasanna. "World Countrywise Population Data 1980 - 2010." RSNA Pneumonia Detection Challenge | Kaggle. August 30, 2017. Accessed March 08, 2019. https://www.kaggle.com/dataswimmer/population19802010/version/1.
5. Kruskal, Joseph B., and Myron Wish. Multidimensional Scaling. Newbury Park, CA: Sage Publ., 2009.
6. "Benin Cuisine." Wikipedia. July 12, 2018. Accessed March 08, 2019. https://en.wikipedia.org/wiki/Benin_cuisine.
7. "Angola Facts and Culture." Qatar Facts, Culture, Recipes, Language, Government, Eating, Geography, Maps, History, Weather, News, Economy, Family, Fashion, Events - CountryReports. Accessed March 08, 2019. https://www.countryreports.org/canada/Angola.htm.
8. Checkdetector. "Principal Component Analysis 4 Dummies: Eigenvectors, Eigenvalues and Dimension Reduction." George Dallas. November 29, 2017. Accessed March 08,

2019.

https://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-dummies-eigenvectors-eigenvalues-and-dimension-reduction/.

Appendixes:

MDS using Java

```
import java.util.*;
import java.io.*;
import com.opencsv.*;
import org.apache.commons.lang3.ObjectUtils;
import org.apache.commons.lang3.StringUtils;
import smile.mds.*;
import org.apache.commons.math3.*;
import smile.projection.*;
import org.apache.commons.math3.stat.descriptive.moment.StandardDeviation;
import org.apache.commons.math3.stat.descriptive.moment.Mean;
import org.apache.commons.math3.stat.correlation.Covariance;
import org.apache.commons.math3.linear.AbstractRealMatrix;
import java.lang.Double;
import smile.math.matrix.DenseMatrix;


public class Dance381 {
   /* everything happens here*/
   public static void main(String[] args) throws IOException {
      List<String[]> popData = readin("population.csv");
      List<String[]> foodData = readin("FAO.csv");
      Map<String, Double> populationMap = new HashMap<>();
      Map<Integer, Map<String, Double>> foodMap = new HashMap<>();

      /* to read in the population data into our map*/
      for(int i = 1; i < popData.size(); i++){
         if(!populationMap.containsKey(popData.get(i)[0]) && !(popData.get(i)[31].equals("--")) &&
!(popData.get(i)[31].equals("NA"))){
```

```
            populationMap.put(popData.get(i)[0], Double.parseDouble(popData.get(i)[31]));
        }
    }
    /* to read in the food consumption data*/
    for(int i = 1; i < foodData.size(); i++){
        if(populationMap.containsKey(foodData.get(i)[2]) && foodData.get(i)[5].equals("5142")
&& !(foodData.get(i)[59].equals(""))) {
            if (!foodMap.containsKey(Integer.parseInt(foodData.get(i)[3]))) {
                Map<String, Double> inMap = new HashMap<>();
                inMap.put(foodData.get(i)[2], Double.parseDouble(foodData.get(i)[59]));
                foodMap.put(Integer.parseInt(foodData.get(i)[3]), inMap);
            }
            Map<String, Double> inside = foodMap.get(Integer.parseInt(foodData.get(i)[3]));
            inside.put(foodData.get(i)[2],
(Double.parseDouble(foodData.get(i)[59])/(populationMap.get(foodData.get(i)[2]))));
            foodMap.put(Integer.parseInt(foodData.get(i)[3]), inside);
        }
    }
    int i = 0;
    Map<String, Integer> countryIndexMap = new HashMap<>();
    /* to read in relevant countries for available data,
    for example we don't want countries which do not have a lot of values*/
    for(int foodcode : foodMap.keySet()){
        Map<String, Double> inside = foodMap.get(foodcode);
        if (inside.keySet().size() > 105) {
            for(String country: inside.keySet()){
                if(!countryIndexMap.containsKey(country)){
                    countryIndexMap.put(country, i);
                    i++;
                    System.out.println(country + ", " + i);
                }
            }
        }
    }
    System.out.println(i);
    i = 0;
    double[][] data = new double[147][147];
    /*list for relevant food types, this line is commented out for other variations*/
    List<Integer> foods = new ArrayList<>();
    foods.add(2605);foods.add(2905); foods.add(2907); foods.add(2918); foods.add(2919);
foods.add(2960); foods.add(2542); foods.add(2511); foods.add(2805);
    /*to calculate distances between countries based on food data*/
    for(int foodcode : foodMap.keySet()) {
        if(foods.contains(foodcode)) {
            Map<String, Double> inside = foodMap.get(foodcode);
            for (String country : inside.keySet()) {
                for (String country1 : inside.keySet()) {
                    if (countryIndexMap.keySet().contains(country) &&
countryIndexMap.keySet().contains(country1)) {
                            int index1 = countryIndexMap.get(country);
                            int index2 = countryIndexMap.get(country1);
                            data[index1][index2] = data[index1][index2] + distance(inside.get(country),
inside.get(country1));
                    }
                }
            }
        }
    }
/*      for(i = 0; i < 147; i++){
        for(int j = 0; j < 147; j++){
            data[i][j] = Math.sqrt(data[i][j]);
        }
    }*/

    /*normalizing data, commented part for other variations. */
    for(i = 0; i < 147; i++){
        double[] calc = new double[147];
        for(int j = 0; j < 147; j++){
            calc[j] = data[i][j];
        }
        Mean mean = new Mean();
        StandardDeviation standardDeviation = new StandardDeviation();
        double mean1 = mean.evaluate(calc);
        double sd = standardDeviation.evaluate(calc);
        for(int j = 0; j < 147; j++){
            data[i][j] = (data[i][j] - mean1)/sd;
        }
    }
    /*MDS that calculates corrdinates for 2-d representation*/
    MDS mds = new MDS(data);
    double[] eig = mds.getProportion();
    System.out.println(Arrays.toString(eig));
    double[][] coordinates = mds.getCoordinates();
    for(i = 0; i < coordinates.length; i++){
        for(int j = 0; j < coordinates[i].length; j++){
            if(j == 0){
```

```
                System.out.print(coordinates[i][j] + ", ");
            } else {
                System.out.print(coordinates[i][j]);
            }
        }
        System.out.println();
    }


}
/* private method to read in csv files*/
private static List<String[]> readin(String file) throws FileNotFoundException, IOException{
    FileReader filereader = new FileReader(file);
    CSVParser parser = new CSVParserBuilder().withSeparator(',').build();
    // create csvReader object and skip first Line
    CSVReader csvReader = new CSVReaderBuilder(filereader)
            .withCSVParser(parser)
            .build();
    List<String[]> allData = csvReader.readAll();
    return allData;
}

/*distance function to change distance implementation easily*/
private static double distance(double a, double b){
    return Math.abs(a-b);
}
}
```

PCA using Java

```
import java.util.*;
import java.io.*;
import com.opencsv.*;
import org.apache.commons.lang3.ObjectUtils;
import org.apache.commons.lang3.StringUtils;
import smile.mds.*;
import org.apache.commons.math3.*;
import smile.projection.*;
import org.apache.commons.math3.stat.descriptive.moment.StandardDeviation;
import org.apache.commons.math3.stat.descriptive.moment.Mean;
import org.apache.commons.math3.stat.correlation.Covariance;
import org.apache.commons.math3.linear.AbstractRealMatrix;
import java.lang.Double;
import smile.math.matrix.DenseMatrix;

public class Dance381 {
   /* everything happens here*/
   public static void main(String[] args) throws IOException {
      List<String[]> popData = readin("population.csv");
      List<String[]> foodData = readin("FAO.csv");
      Map<String, Double> populationMap = new HashMap<>();
      Map<Integer, Map<String, Double>> foodMap = new HashMap<>();

         /* to read in the population data into our map*/
      for(int i = 1; i < popData.size(); i++){
         if(!populationMap.containsKey(popData.get(i)[0]) && !(popData.get(i)[31].equals("--")) &&
!(popData.get(i)[31].equals("NA"))){
            populationMap.put(popData.get(i)[0], Double.parseDouble(popData.get(i)[31]));
         }
      }
         /* to read in the food consumption data*/
      for(int i = 1; i < foodData.size(); i++){
         if(populationMap.containsKey(foodData.get(i)[2]) && foodData.get(i)[5].equals("5142")
&& !(foodData.get(i)[59].equals(""))) {
            if (!foodMap.containsKey(Integer.parseInt(foodData.get(i)[3]))) {
               Map<String, Double> inMap = new HashMap<>();
               inMap.put(foodData.get(i)[2], Double.parseDouble(foodData.get(i)[59]));
               foodMap.put(Integer.parseInt(foodData.get(i)[3]), inMap);
            }
            Map<String, Double> inside = foodMap.get(Integer.parseInt(foodData.get(i)[3]));
            inside.put(foodData.get(i)[2],
(Double.parseDouble(foodData.get(i)[59])/(populationMap.get(foodData.get(i)[2]))));
            foodMap.put(Integer.parseInt(foodData.get(i)[3]), inside);
         }
      }

      double[][] data = new double[200][117];
      int i = 0;
         /*Get the data in the form of 117 different food items with a value for each country
noted*/
      for(int foodcode : foodMap.keySet()){
```

```
    Map<String, Double> inside = foodMap.get(foodcode);
    int j = 0;
    for(String country : inside.keySet()){
        data[j][i] = inside.get(country);
        j++;
    }
    i++;
}
    /* Find out how many countries have data for each food type*/
int[] end = new int[117];
for(i = 0; i < 117; i++){
    for(int j = 199; j >= 0; j--){
        if(data[j][i] != 0){
            end[i] = j;
            break;
        }
    }

}
    /*normalizing data, commented part for other variations. */
for(i = 0; i < 117; i++){
    double[] calc = new double[end[i]];
    for(int j = 0; j < end[i]; j++){
        calc[j] = data[j][i];
    }
    Mean mean = new Mean();
    StandardDeviation standardDeviation = new StandardDeviation();
    double mean1 = mean.evaluate(calc);
    double sd = standardDeviation.evaluate(calc);
    for(int j = 0; j < end[i]; j++){
        data[j][i] = (data[j][i] - mean1)/sd;
    }
}
double[][] finData = new double[200][117];
    /*making sure data is not sparse*/
for(i = 0; i < 117; i++){
    for(int j = 0; j < 200; j++){
        if(data[j][i] != 0.0){
            finData[j][i] = data[j][i];
        }
    }
}
    /* finding covariance matrices*/
    Covariance covariance = new Covariance(finData);
    AbstractRealMatrix realMatrix = (AbstractRealMatrix) covariance.getCovarianceMatrix();
    double[][] covdata = realMatrix.getData();
    i=0;
    int j = 0;
    int i1 = 0;
    int j1 = 0;
    double[][] finalcovData = new double[117][117];
    while(i < 117){
        boolean allzeros = true;
        j=0;
        j1=0;
        while(j < 117){
            if((!Double.isNaN(covdata[i][j])) && (Math.abs(covdata[i][j]) >= 0.0)) {
                finalcovData[i1][j1] = covdata[i][j];
                j1++;
                allzeros = false;
            }
            j++;
        }
        i++;
        if(!allzeros) {
            i1++;
        }
    }
        /* Getting PCA results from library*/
    PCA pca = new PCA(finalcovData);
    double[] var = pca.getVarianceProportion();
    DenseMatrix denseMatrix = pca.getLoadings();
    System.out.println(denseMatrix.toString(true));
    System.out.println(Arrays.toString(var));
    double[][] arr = denseMatrix.array();
    for(i = 0; i < 1; i++){
        for(j = 0; j < 113; j++){
            if(arr[j][i] > 0.1 || arr[j][i] < -0.1){
                System.out.println(j);
            }
        }
    }
}
/* private method to read in csv files*/
private static List<String[]> readin(String file) throws FileNotFoundException, IOException{
    FileReader filereader = new FileReader(file);
```

```
        CSVParser parser = new CSVParserBuilder().withSeparator(',').build();
        // create csvReader object and skip first Line
        CSVReader csvReader = new CSVReaderBuilder(filereader)
                .withCSVParser(parser)
                .build();
        List<String[]> allData = csvReader.readAll();
        return allData;
    }
}
```

Plotting using Python

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 4 15:40:55 2019

@author: bhall
"""
#File modified to fit variation limits in axes
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
from PIL import Image
fig = plt.figure();
ax = fig.add_subplot(111)
df = pd.read_csv('output_var4.csv')
X = df.iloc[:, 0].values
y = df.iloc[:, 1].values
df1 = pd.read_csv('country_var4.csv')
country = df1.iloc[:, 0]
for i in range(147):
    ax.text(X[i] + 1, y[i] + 1, country[i], fontsize=6)
ax.axis([0, 2, 2, 0])
plt.savefig('world_var4.pdf')
plt.show()
```

Plotting using RStudio

```
# Import population data and food consumption data
data <- read.csv("population.csv")
colnames(data)[colnames(data)=="Country"] <- "Area"
colnames(data)[colnames(data)=="X2010"] <- "population"
fooddata <- read.csv("FAO.csv")
total <- merge(data, fooddata, by = "Area")
total <- total[-which(total$Element == "Feed"),]

total <- total[which(total$Item.Code == 2605 | total$Item.Code == 2905 | total$Item.Code ==
2907 | total$Item.Code == 2918 | total$Item.Code == 2919 | total$Item.Code == 2960 |
total$Item.Code == 2542 | total$Item.Code == 2511 | total$Item.Code == 2805),]

# Code 2605 - vegetables
vegetables <- total[which(total$Item.Code == 2605),]

country.vege1 <- c()
# calculate the vegetables consumption per person
for (i in 1:276) {
  country.vege1[i] <- as.numeric(vegetables$Y2010[vegetables$Area.Code ==
i][1])/(as.numeric(as.character(vegetables$population[vegetables$Area.Code == i][1])))
}

jpeg('vege1.jpg')
plot(1:276, country.vege1, xlab = "Country Code", ylab = "Vegetables Consumption Per
Person", main = "VEGETABLES")
dev.off()

# Code 2905 - cereals
cereals <- total[which(total$Item.Code == 2905),]

country.cereals <- c()
# calculate the cereals consumption per person
for (i in 1:276) {
  country.cereals[i] <- as.numeric(cereals$Y2010[cereals$Area.Code ==
i][1])/(as.numeric(as.character(cereals$population[cereals$Area.Code == i][1])))
}

jpeg('cereals.jpg')
```

```
plot(1:276, country.cereals, xlab = "Country Code", ylab = "Cereals Consumption Per Person",
main = "CEREALS")
dev.off()

# Code 2907 - starchy roots
starchy.roots <- total[which(total$Item.Code == 2907),]

country.sr <- c()
# calculate the starchy roots consumption per person
for (i in 1:276) {
  country.sr[i] <- as.numeric(starchy.roots$Y2010[starchy.roots$Area.Code ==
i][1])/(as.numeric(as.character(starchy.roots$population[starchy.roots$Area.Code == i][1])))
}

jpeg('starchy_roots.jpg')
plot(1:276, country.sr, xlab = "Country Code", ylab = "Starchy Roots Consumption Per Person",
main = "STARCHY ROOTS (vegetables eg. potato)")
dev.off()

# Code 2918 - Vegetables
vegetables2 <- total[which(total$Item.Code == 2918),]

country.vege2 <- c()
# calculate the vegetables consumption per person
for (i in 1:276) {
  country.vege2[i] <- as.numeric(vegetables2$Y2010[vegetables2$Area.Code ==
i][1])/(as.numeric(as.character(vegetables2$population[vegetables2$Area.Code == i][1])))
}
jpeg('vege2.jpg')
plot(1:276, country.vege2, xlab = "Country Code", ylab = "Vegetables Consumption Per
Person", main = "VEGETABLES")
dev.off()

# Code 2919 - Fruits
fruits <- total[which(total$Item.Code == 2919),]

country.fruits <- c()
# calculate the fruits consumption per person
for (i in 1:276) {
  country.fruits[i] <- as.numeric(fruits$Y2010[fruits$Area.Code ==
i][1])/(as.numeric(as.character(fruits$population[fruits$Area.Code == i][1])))
}
jpeg('fruits.jpg')
```

```
plot(1:276, country.fruits, xlab = "Country Code", ylab = "Fruits Consumption Per Person", main
= "FRUITS")
dev.off()

# Code 2960 - fish
fish <- total[which(total$Item.Code == 2960),]

country.fish <- c()
# calculate the fish consumption per person
for (i in 1:276) {
  country.fish[i] <- as.numeric(fish$Y2010[fish$Area.Code ==
i][1])/(as.numeric(as.character(fish$population[fish$Area.Code == i][1])))
}
jpeg('fish.jpg')
plot(1:276, country.fish, xlab = "Country Code", ylab = "Fish Consumption Per Person", main =
"FISH")
dev.off()

# Code 2542 - sugar
sugar <- total[which(total$Item.Code == 2542),]

country.sugar <- c()
# calculate the sugar consumption per person
for (i in 1:276) {
  country.sugar[i] <- as.numeric(sugar$Y2010[sugar$Area.Code ==
i][1])/(as.numeric(as.character(sugar$population[sugar$Area.Code == i][1])))
}
jpeg('sugar.jpg')
plot(1:276, country.sugar, xlab = "Country Code", ylab = "Sugar Consumption Per Person", main
= "SUGAR")
dev.off()

# Code 2511 - wheat
wheat <- total[which(total$Item.Code == 2511),]

country.wheat <- c()
# calculate the wheat consumption per person
for (i in 1:276) {
  country.wheat[i] <- as.numeric(wheat$Y2010[wheat$Area.Code ==
i][1])/as.numeric(as.character(total$population[total$Area.Code == i][1]))
}
jpeg('wheat.jpg')
country.wheat[100]
```

```
plot(1:276, country.wheat, xlab = "Country Code", ylab = "Wheat Consumption Per Person",
main = "WHEAT")
dev.off()

# Code 2805 - rice
rice <- total[which(total$Item.Code == 2805),]

country.rice <- c()
# calculate the rice consumption per person
for (i in 1:276) {
  sum <- sum(rice$Y2010[rice$Area.Code == i])
  country.rice[i] <- as.numeric(rice$Y2010[rice$Area.Code ==
i][1])/(as.numeric(as.character(rice$population[rice$Area.Code == i][1])))
}
jpeg('rice.jpg')
plot(1:276, replace(country.rice, c(100), 0), xlab = "Country Code", ylab = "Cereals Consumption
Per Person", main = "RICE")
dev.off()

jpeg('variations.jpg')
par(mfrow=c(3,3))
plot(1:276, country.vege1, xlab = "Country Code", ylab = "Vegetables Consumption Per
Person", main = "VEGETABLES")
plot(1:276, country.cereals, xlab = "Country Code", ylab = "Cereals Consumption Per Person",
main = "CEREALS")
plot(1:276, country.sr, xlab = "Country Code", ylab = "Starchy Roots Consumption Per Person",
main = "STARCHY ROOTS")
plot(1:276, country.vege2, xlab = "Country Code", ylab = "Vegetables Consumption Per
Person", main = "VEGETABLES")
plot(1:276, country.fruits, xlab = "Country Code", ylab = "Fruits Consumption Per Person", main
= "FRUITS")
plot(1:276, country.fish, xlab = "Country Code", ylab = "Fish Consumption Per Person", main =
"FISH")
plot(1:276, country.sugar, xlab = "Country Code", ylab = "Sugar Consumption Per Person", main
= "SUGAR")
plot(1:276, country.wheat, xlab = "Country Code", ylab = "Wheat Consumption Per Person",
main = "WHEAT")
plot(1:276, country.rice, xlab = "Country Code", ylab = "Cereals Consumption Per Person", main
= "RICE")
dev.off()
```

MDS output

```
1.X-value                              2.Y-value                  3.Country        4.Index
-190.24715734552416    94.9369179325237           Benin              1        // the coordinate for
Bein in MDS graph is (-190.24715734552416,94.9369179325237)
-152.04521769186704    264.4995017342689          Angola            2        // the coordinate for
Angola in MDS graph is (-152.04521769186704, 264.4995017342689)

-57.72529531586912    -119.62940576283897        Cambodia       3        // the coordinate for
Cambodia in MDS graph is (-57.72529531586912,  -119.62940576283897)

(144 lines of similar types are omitted)
```

1. X-value- x-coordinate for MDS plot for given country
2. Y-value- y-coordinate for MDS plot for given country
3. Country- given country
4. Index- index of the country in MDS data array.

The data above are the output of our main model.
There are four more output like this for variations, and they are omitted.