Determination of proper grouping approaches for large data sets by applying Cluster Analysis methods in MATLAB

for the Bachelor of Science (General) Degree
By
O.G.V.V Bandara
SC/2019/10723

Supervisor:
Prof. L.A.L.W. Jayasekara

Department of Mathematics
University of Ruhuna
Matara.

2021

Acknowledgement

This project has been capable of completing with the kind support of many people. Hereby I would like to pass my sincere gratitude. First and foremost my sincere gratitude goes to our supervisor Prof. L.A.L.W. Jayasekara for providing us with the required guidance and resources to complete this project. Next I would like to pass my gratitude to our instructors for providing us with the required guidance and resources in completing the project.I would also like to pass my gratitude to all my lecturers and instructors who enlightened us with the MATLAB programming knowledge. My sincere gratitude also goes out to our lecturers of the science faculty for providing us with the required guidance and resources which made this possible. At last but but not least this would not have been capable without the contribution,co-operation of my group members so I would like to pass my gratitude to them as well.

Abstract

This project carries out the objective of grouping a large data set. Data sets which have no visible relationships require special methods or criteria to be grouped accordingly. Cluster analysis is the method used. cluster analysis comprises of two methods.Hierarchical clustering and non hierarchical clustering. A code was developed using MATLAB in order utilize a specific clustering method and group the data. The grouped data was visually represented to a user in order for it to be useful information. The visual representation used is known as a dendrogram. The dendrogram has the capability to visually represented the grouping of data in a much more efficient manner. Through MATLAB the user is possible to input the large data set to the code and choose a specific grouping approach. Then the code will output a dendrogram. The tools in MATLAB also allows the user to analyze the dendrogram clearly and efficiently.

Contents

List of Tables

Chapter 1

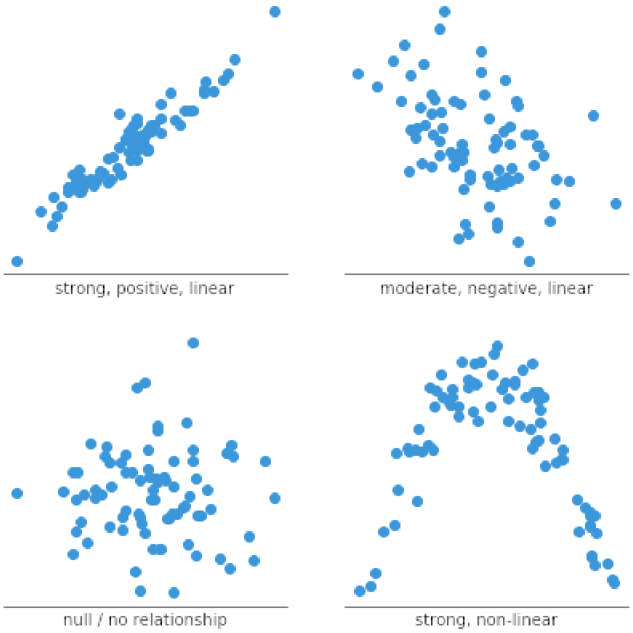
Introduction

1.1 Background of study

1.1.1 Cluster analysis and grouped data

Cluster analysis intends to find patterns in sets and group them accordingly. The objective would be to find the best grouping such that the similarities within the cluster is high and similarities between clusters is low. The groupings found should match with the intended aim and should make sense to the researcher. In cluster analysis, neither the number of groups nor the groups themselves are known in advance.

During clustering we can observe many similarities and dissimilarities between clusters. But most of the time the criteria used is a distance. Other cluster methods use a preliminary choice for cluster centers or a comparison of within- and between-cluster variability. In which cases the similarity may be a known correlation. We can identify similarities and observe the slight clutering nature through a scatter plot.



fig(1.1) The figure shows an example of different scatter plots with different correlations.

Cluster analysis has many applications in many fields. One of the main grouping methods used in unsupervised machine learning is cluster analysis. It is a main task of exploratory data analysis, and a common technique for statistical data analysis.Many of the other fields include pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics and machine learning.

1.1.2 Clustering methods

Different clustering methods exist and the algorithm may vary amongst each. Understanding each of the different cluster models is the key to understanding the cluster analysis algorithms.

- Connectivity models
- Centroid Models
- Distribution models
- Density models
- Subspace models
- Group models
- Graph based models
- Signed graph models
- Neural models

Also clusters can specify the relationship between each other, in the sense that a hierarchy of clusters embedded in each other.

- Hard clustering
- Soft clustering

1.1.3 Managing input data

The input data depends on the approach taken for clustering. There are two common approaches for clustering and the input varies accordingly

- hierarchical clustering: typically starts with 'n' clusters such that it comprises one for each observation. And ends with a single cluster containing all 'n' observations. At each step, an observation or a cluster of observations is absorbed into another cluster. We can also reverse this process, that is, start with a single cluster containing all n observations and end with n clusters of a single item each
- partitioning: The observations are divided into 'g' clusters.This can be done by starting with an initial partitioning or with cluster centers and then reallocating the observations according to some optimality criterion.

1.1.4 Correlations

Correlation is a statistical measure (expressed as a number) that describes the size and direction of a relationship between two or more variables. A correlation between variables, however, does not automatically mean that the change in one variable is the cause of the change in the values of the other variable. The Carl-Pearson correlation coefficient is an appropriate method to measure the degree to which observations have a relation.

$$r_{xy} = \frac{\sum_{j=1}^{p}(x_j - \overline{x})(y_j - \overline{y})}{\sqrt{\sum_{j=1}^{p}(x_j - \overline{x})^2 \sum_{j=1}^{p}(y_j - \overline{y})^2}} \tag{1.1}$$

1.1.5 Measures of Simmilarity and Dissimilarity

The objective of cluster analysis is to identify similarities between input vectors and group them. One of the most convenient methods of measure of proximity would be the distance between the two observations. Since the distance increases as the observations become more seperated according to a certain criteria, distance can be considered as a measure of dissmililarity.

A common measure of distance is the Euclidean Distance between two vectors. The two vectors are p dimensional vectors $\big(x_1, x_2, ..., x_p\big)^T$, $\big(y_1, y_2, ..., y_p\big)^T$

The Euclidean distance can be given as

$$d(x, y) = \sqrt{(x - y)^T(x - y)} \tag{1.2}$$

1.2 Objectives

1. The objective of cluster analysis is to identify similarities between observation and create a criterea in order to group them such that the data is useful to the researcher

2. A code is developed through MATLAB such that the grouping process can be done easily the user

3. The code enables the user to choose their preferred grouping method and obtain a visual representation of the grouped data

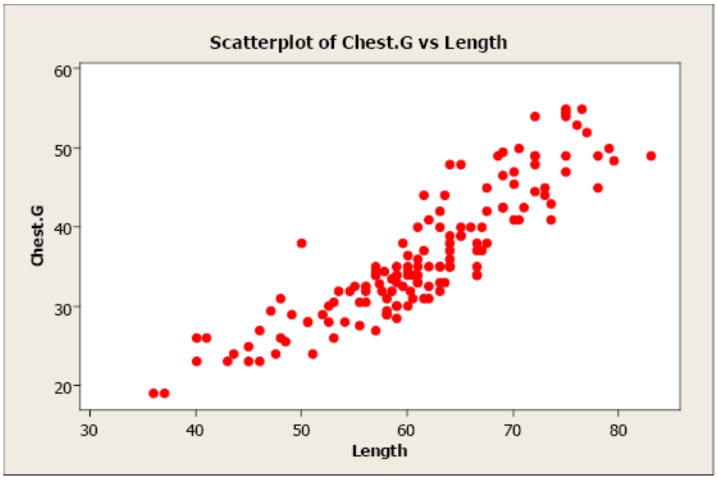
4. By using a code the user is able to input data one time without the tedious process of having to calculate the grouping for each input observation set multiple times.

Chapter 2

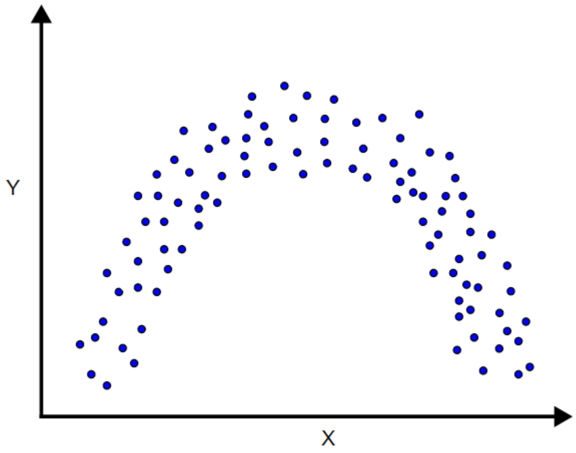
Problem Statement

2.1 grouping large data sets which have no visible relationship

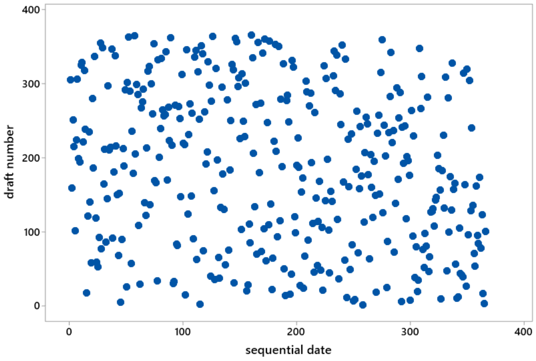
The collection of data can be done in many ways. For data to be useful the have to be converted to information. Certain types of data require to be grouped depending on the requirement of the use.The correlation of the data can vary depending on the data set obtained. Different correlations between data sets can be observed.



fig(2.1)The figure indicates a linear positive correlation between the bivariate data set



fig(2.2)figure shows a strong non linear correlation between the data set



fig(2.3)figure shows a data set with no visible correlation
Finding relationships in data sets with no visible correlations is a quite hard task.

2.2 obtaining useful inoformation through grouping

The set obtained has to be presented to any user in the form of information. the required information depends on the user. Without a proper method to group the data or categorize them with suitable criteria or characteristics the data obtained may tend to be useless.grouping the data is a much needed requirement in the industry and many companies as well.

2.3 The tedious process of repetitive calculations

When using traditional calculation methods, the user has to do multiple and lengthy calculations which takes up much time and reduces efficiency. Most of the calculations in the clusterin approaches require mathematical and statistical knowledge and are tedious when done by hand.

Chapter 3

Methodology

3.1 Single Linkage(Nearest Neighbour)

In the single linkage method, the distance between two clusters A and B is defined as the minimum distance between a point in A and a point in B.

$$\mathrm{D}(A, B) = min\{d(y_i, y_j)\} \tag{3.1}$$

where

$$d(y_i, y_j)$$

is the Euclidean distance in or some other distance between the vectors

$$y_i$$

and

$$y_j$$

.

At each step in the linkage method the distance (3.1) is found for every pair of clusters and the clusters with the smallest distance is merged. Then the number of cluster is reduced by 1. After two clusters are merged, the procedure is repeated for the next step: the distances between all pairs of clusters are calculated again, and the pair with minimum distance is merged into a single cluster.

The clustering is visually represented in a dendrogram for more convenience of the user. The distance matrix can be obtained of the above data set by converting to squareform.

city	Murder	Rape	Robbery	Assault	Burglary	Larceny	Auto Theft
Atlanta	16.5	24.8	106	147	1112	905	494
Boston	4.2	13.3	122	90	982	669	954
Chicago	11.6	24.7	340	242	808	609	645
Dallas	18.1	34.2	184	293	1668	901	602
Denver	6.9	41.5	173	191	1534	1368	780
Detroit	13	35.7	477	220	1566	1183	788
Hartford	2.5	8.8	68	103	1017	724	468
Honolulu	3.6	12.7	42	28	1457	1102	637
Houstan	16.8	26.6	289	186	1509	787	697
Kansas City	10.8	43.2	255	226	1494	955	765
Los Angeles	9.7	51.8	286	355	1902	1386	862
New Orleans	10.3	39.7	266	283	1056	1036	776
New York	9.4	19.4	522	267	1674	1392	848
Portland	5	23	157	144	1530	1281	488
Tucson	5.1	22.9	85	148	1206	756	483
Washington	12.5	27.6	524	217	1496	1003	793

Table 3.1: Sample table of city crime data of each of the chosen cities with respective to the type of crime.

Atlanta	0	536.6	516.4	590.2	693.6	716.2
Boston	536.6	0	447.4	833.1	915.0	881.1
Chicago	516.4	447.4	0	924.0	1073.4	971.5
Dallas	590.2	833.1	924.0	0	527.7	464.5
Denver	693.6	915.0	1073.4	527.7	0	358.7
Detroit	716.2	881.1	971.5	464.5	358.7	0

Table 3.2: squareform of the sample city crime data in single linkage

The smallest distance is 358.7 between Denver and Detroit, and therefore these two cities are joined at the first step to form C1 = {Denver, Detroit}. In the next step, the distance matrix is calculated for Atlanta, Boston, Chicago, Dallas, and C1:

Atlanta	0	536.6	516.4	590.2	693.6
Boston	536.6	0	447.4	833.1	881.1
Chicago	516.4	447.4	0	924.0	971.5
Dallas	590.2	833.1	924.0	0	464.5
C1	693.6	881.1	971.5	464.5	0

Table 3.3: Obtaining C1 in single linkage

All elements of this distance matrix are contained in the original distance matrix. This same pattern will hold in subsequent distance matrices. The smallest distance is 447.4 between Boston and Chicago. Therefore C2 = {Boston, Chicago}. At the next step, the distance matrix is calculated for Atlanta, Dallas, C1, and C2:

The smallest distance is 464.5 between Dallas and C1, so that C3 = {Dallas, C1}. The distance matrix for Atlanta, C2, and C3 is given by

Atlanta	0	516.4	590.2	693.6
C2	516.4	0	833.1	881.1
Dallas	590.2	833.1	0	464.5
C1	693.6	881.1	464.5	0

Table 3.4: Obtaining C2 in single linkage

Atlanta	0	516.4	590.2
C2	516.4	0	833.1
C3	590.2	833.1	0

Table 3.5: Obtaining C3 in single linkage

The smallest distance is 516.4, which defines C4 = {Atlanta, C2}. The distance matrix for C3 and C4 is

C3	0	590.2
C4	590.2	0

Table 3.6: Obtaining C4 in single linkage

3.2 Complete Linkage(farthest neighbour)

In the complete linkage approach, also called the farthest neighbor method, the distance between two clusters A and B is defined as the maximum distance between a point in A and a point in B.

$$D(A, B) = max\{d(y_i, y_j)\} \tag{3.2}$$

At each step, the distance is found for every pair of clusters as shown above, and the two clusters with the smallest distance are merged. The distance matrix D is given by City Distance between Cities.

Atlanta	0	536.6	516.4	590.2	693.6	716.2
Boston	536.6	0	447.4	833.1	915.0	881.1
Chicago	516.4	447.4	0	924.0	1073.4	971.5
Dallas	590.2	833.1	924.0	0	527.7	464.5
Denver	693.6	915.0	1073.4	527.7	0	358.7
Detroit	716.2	881.1	971.5	464.5	358.7	0

Table 3.7: conversion to squareform in complete linkage

The smallest distance is 358.7 between Denver and Detroit, and these two therefore form the first cluster, C1 = {Denver, Detroit}.

In the next step, the distance matrix is calculated for Atlanta, Boston, Chicago, Dallas, and C1: The smallest distance is 447.4 between Boston and Chicago. Therefore, C2 = {Boston,Chicago}. At the next step, distances are calculated for Atlanta, Dallas, C1, and C2:

Atlanta	0	536.6	516.4	590.2	716.2
Boston	536.6	0	447.4	833.1	915.0
Chicago	516.4	447.4	0	924.0	1073.4
Dallas	590.2	833.1	924.0	0	527.7
C1	716.2	915.0	1073.4	527.7	0

Table 3.8: Obtaining C1 in complete linkage

Atlanta	0	536.6	590.2	716.2
C2	536.6	0	924.0	833.1
Dallas	590.2	924.0	0	527.7
C1	693.6	881.1	527.7	0

Table 3.9: Obtaining C2 in complete linkage

Atlanta	0	536.6	716.2
C2	536.6	0	1073.4
C3	590.2	1073.4	0

Table 3.10: Obtaining C3 in complete linkage

The smallest distance, 527.7, defines C3 = {Dallas,C1}. The distance matrix for Atlanta, C2, and C3 is given by The smallest distance is 536.6 between Atlanta and C3, so that C4 = {Atlanta, C2}. The distance matrix for C3 and C4 is The last cluster is given by C5 = {C3,C4}. The clustering can be converted to a dendrogram by using the distance matrix accordingly.

C3	0	1073.4
C4	1073.4	0

Table 3.11: Obtaining C4 in complete linkage

3.3 Centroid method

In the centroid method, the distance between two clusters A and B is defined as the Euclidean distance between the mean vectors (often called centroids) of the two clusters.

$$D(A,B) = d(\overline{y_A}, \overline{y_B}) \tag{3.3}$$

The two clusters with the smallest distance between centroids are merged at each step. where

$$y_A$$

and

$$y_B$$

are the mean vectors for the observation vectors in A and the observation vectors in B.

$$\overline{y_A} = \sum_{i=1}^{n_A} \frac{y_i}{n_A} \tag{3.4}$$

After two clusters A and B are joined, the centroid of the new cluster AB is given by the weighted average.

$$\overline{y_{AB}} = \frac{n_a\overline{y_A} + n_B\overline{y_B}}{n_A + n_B} \tag{3.5}$$

3.4 Median method

If two clusters A and B are combined using the centroid method, and if A contains a larger number of items than B, then the new centroid

$$\overline{y_{AB}} = \frac{n_a\overline{y_A} + n_B\overline{y_B}}{n_A + n_B} \tag{3.6}$$

may be much closer to

$$y_A$$

than to

$$y_B$$

. To avoid weighting the mean vectors according to cluster size, we can use the median (midpoint) of the line joining A and B as the point for computing new distances to other clusters.

To avoid weighting the mean vectors according to cluster size, we can use the median (midpoint) of the line joining A and B as the point for computing new distances to other clusters.

$$m_{AB} = \frac{1}{2}(y_A + y_B) \tag{3.7}$$

The two clusters with the smallest distance between medians are merged at each step. The median defined here is not the ordinary statistical median. Th median here is a median of a triangle, namely, the line from a vertex to the midpoint of the opposite side.

3.5 Average Linkage

In average linkage approach, The distance between two clusters A and B is defined as the average of

$$n_A, n_B$$

distances between

$$n_A$$

points in A and

$$n_B$$

points in B.

$$D(A, B) = \frac{1}{n_A n_B} \sum_{i=1}^{n_A} \sum_{j=1}^{n_B} d(y_i, y_j) \tag{3.8}$$

where the sum is over all

$$y_i$$

in A and all

$$y_j$$

in B. At each step, we join the two clusters with the smallest distance.

3.6 Wards method

Ward's method, also called the incremental sum of squares method, uses the within-cluster (squared) distances and the between-cluster (squared) distances. If AB is the cluster obtained by combining clusters A and B, then the sum of within-cluster distances are

$$SSE_A = \sum_{i=1}^{n_A} (y_i - \overline{y_A})'(y_i - \overline{y_A}) \tag{3.9}$$

$$SSE_B = \sum_{i=1}^{n_B} (y_i - \overline{y_B})'(y_i - \overline{y_B}) \tag{3.10}$$

$$SSE_{AB} = \sum_{i=1}^{n_{AB}} (y_i - \overline{y_{AB}})'(y_i - \overline{y_{AB}}) \tag{3.11}$$

where

$$\overline{y_{AB}} = \frac{n_A \overline{y_A} + n_B \overline{y_B}}{n_A + n_B} \tag{3.12}$$

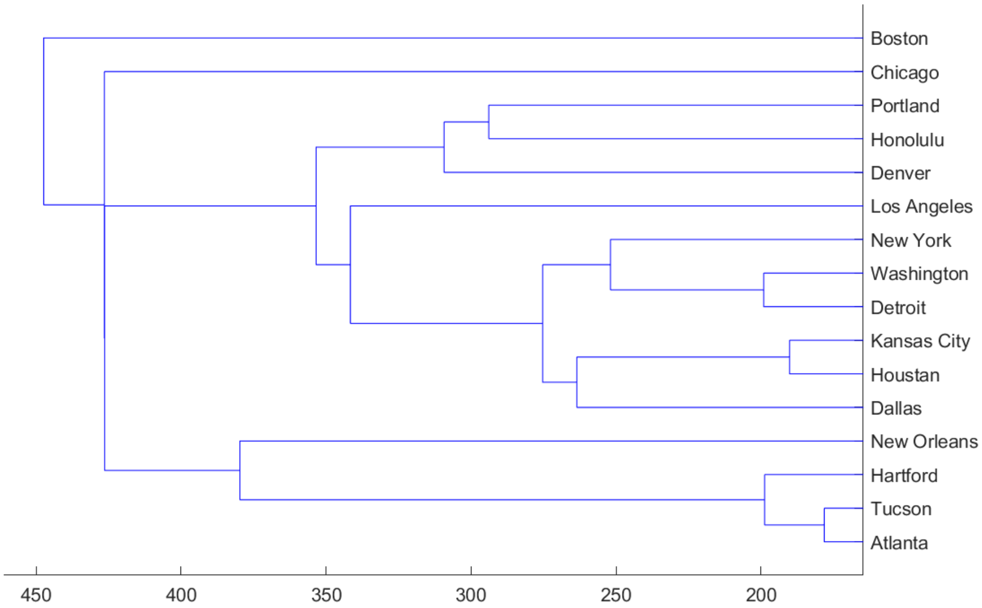
then the increase in SSE is defined as

$$I_{AB} = SSE_{AB} - (SSE_A + SSE_B) \tag{3.13}$$

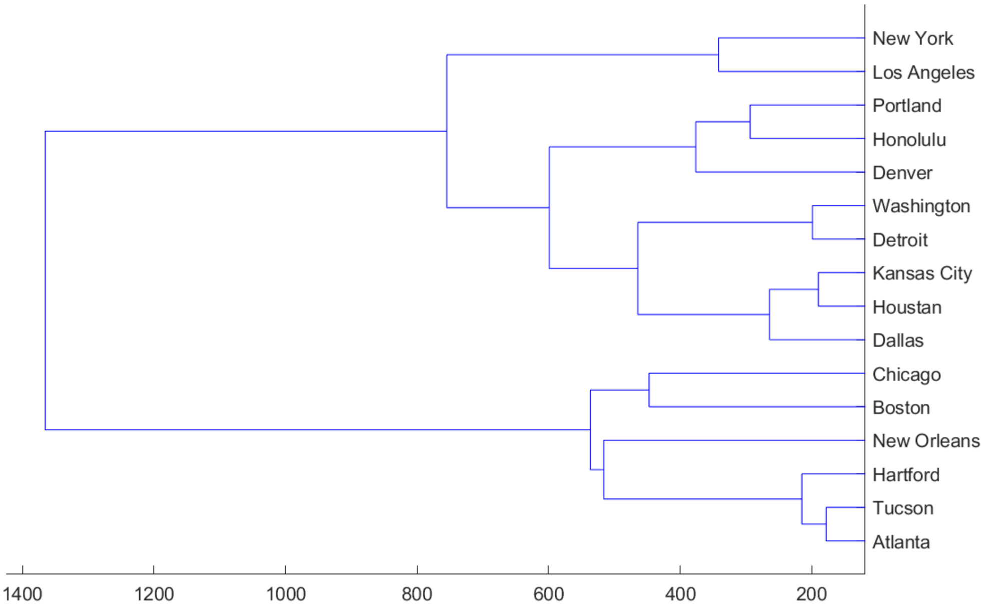
Chapter 4

Discussion

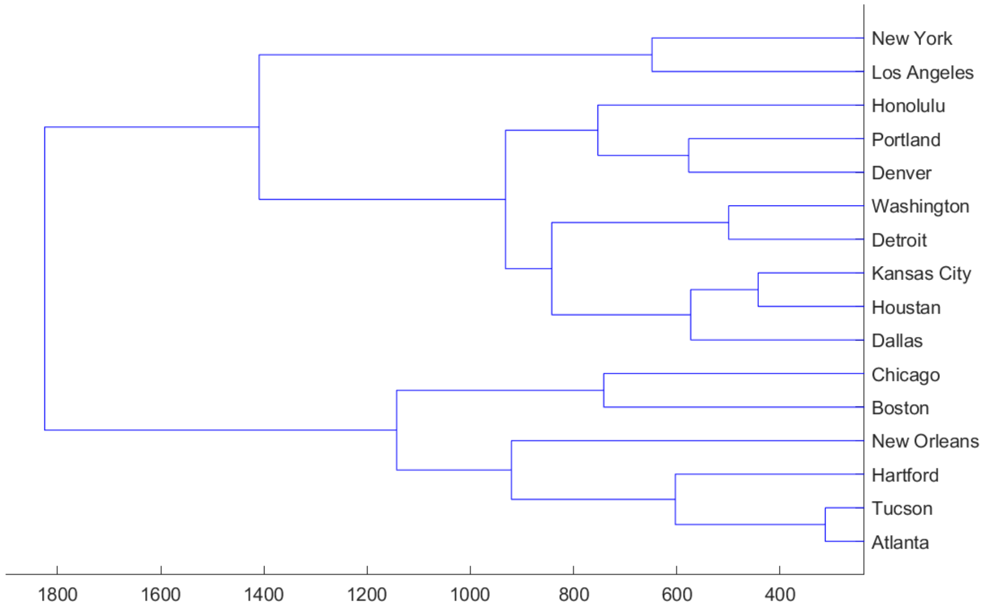
The code developed has the ability to successfully output a visual representation of the grouping through clustering methods. Each of the clustering methods have a different approach and the grouping clearly shows the differences.



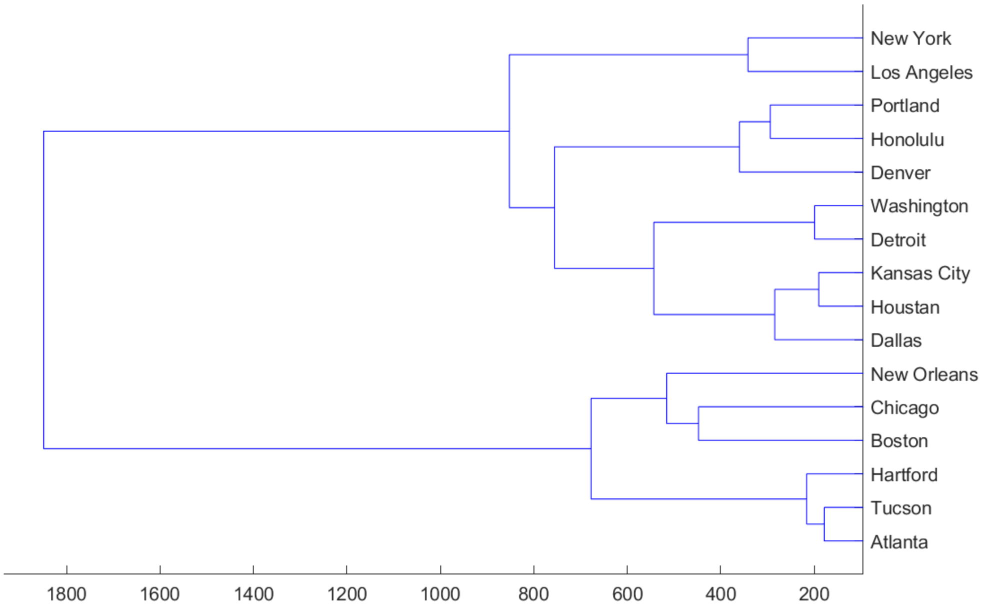
fig(4.1)Dendrogram obtained through single linkage clustering approach



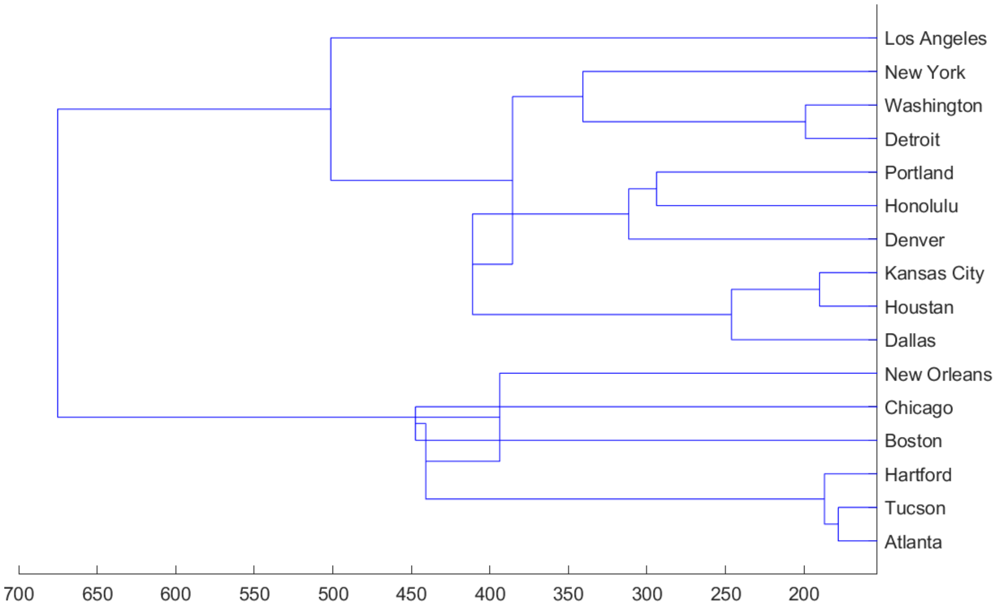
fig(4.2)Dendrogram obtained through complete linkage clustering approach



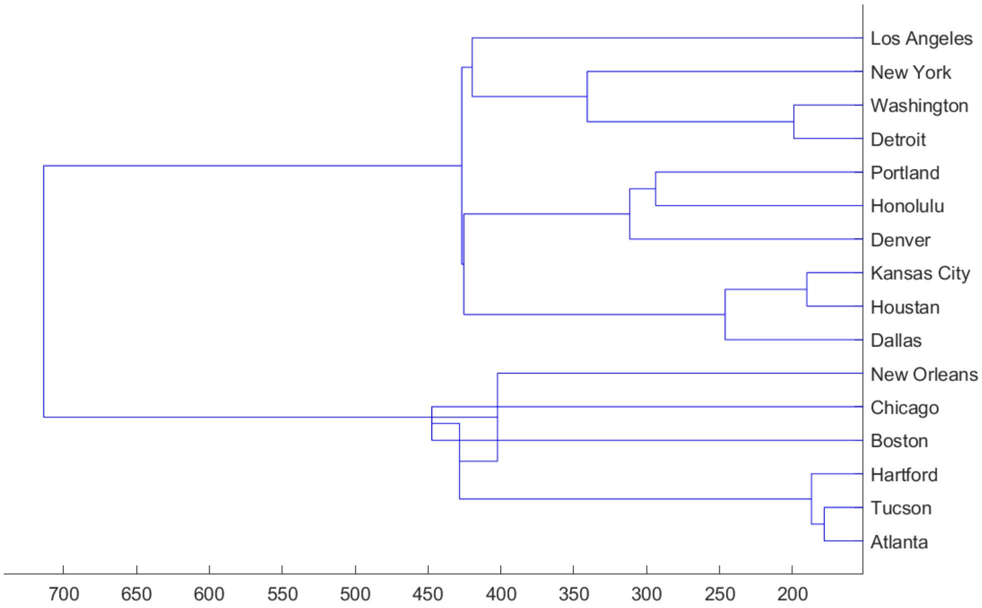
fig(4.3)dendrogram obtained through average linkage clustering approach



fig(4.4)dendrogram obtained through wards method clustering approach



fig(4.5)dendrogram obtained through centroid method clustering approach



fig(4.6)dendrogram obtained through median clustering approach

The approaches used here fall mainly into the category of hierarchichal clustering. Non hierarchichal clustering approaches also exist which could show advantages over hierarchichal appraches which were not utilized here. matlab has the ability to isolate group from a dendrogram using coding. This tool is possible to be used for analysis of the dendrogram.

Chapter 5

Conclusion

Utilizing MATLAB A useful code has been developed. The user has the ability to select the preferred clustering method and input the data set using an excel sheet which is a very convenient method. This is extremely valuable for large data sheets since the user can directly input the data. The code has the ability to run as many times as the user requires it to, so the tedious process of performing calculations multiple times by hand has been eliminated. Most of the calculations done in the clustering methods require mathematical knowledge and time for calculation but the code enables the user to easily obtain the required grouping and therefore turn data into useful information very efficiently.

Chapter 6

Appendix

```
clear;
clc;

% Input for linkage selection ————————————————————————————
fprintf('Hello there , Please choose your Clustering method.\n\n')

fprintf('Your Options are\n\n')

fprintf('  1 - Single Linkage method\n')
fprintf('  2 - Complete Linkage method\n')
fprintf('  3 - Average Linkage method\n')
fprintf('  4 -  W a r d s  Method\n')
fprintf('  5 - Centroid method\n')
fprintf('  6 - Median method\n\n')

Sel = input('Please Enter your choice  -   ');
%————————————————————————————————————————————————————————————————

% Check for input validity
% ————————————————————————————————————

while isempty(Sel) == 1 ||  (Sel <1 || Sel >6 || (floor(Sel) ~= ceil(Sel)))
    clear Sel
    Sel = input('\n\nPlease enter a valid number  -   ');
end


%————————————————————————————————————————————————————————————————

%Check selection and start ——————————————————————————————————————
if Sel == 1
    fprintf('\n\nYou Have chosen Single Linkage Method\n');
    input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
```

```
        [A,C,D,m,n] = Input();
        SingleLinkage(A,D,C,n,m);

    elseif Sel == 2
        fprintf('\n\nYou Have chosen Complete Linkage Method\n');
        input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
        [A,C,D,m,n] = Input();
         CompleteLinkage(A,D,C,n,m);

    elseif Sel == 3
        fprintf('\n\nYou Have chosen Average Linkage Method\n');
        input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
        [A,C,D,m,n] = Input();
         AverageLinkage(A,D,C,n,m);

    elseif Sel == 4
        fprintf('\n\nYou Have chosen Ward s Method\n');
        input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
        [A,C,D,m,n] = Input();
         Wards(A,D,C,n,m);

    elseif Sel == 5
        fprintf('\n\nYou Have chosen Centroid Method\n');
        input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
        [A,C,D,m,n] = Input();
         centroid(A,D,C,n,m);

    elseif Sel == 6
        fprintf('\n\nYou Have chosen Median Method\n');
        input('\n\nIs your Input File Ready ? If YES Press enter\n\n','s');
        [A,C,D,m,n] = Input();
         median(A,D,C,n,m);

    end


    Res=input('Do you want to Restart the Programme ? If Yes Input Y.
    Any other Input will consider as No\n\n','s');
    if Res=='Y'
        run('Final.m');
    else
        fprintf('<strong>—PROGRAMME ENDED | THANK YOU————</strong>\n\n\n')
    end
    %————————————————————————————————————————————————————————————

    % Input taking Function——————————————————————————————————————————
    function [A,C,D,m,n] = Input()
```

```
Inp_Tbl = readtable('input.xlsx','ReadRowNames',true);
row_num = (height(Inp_Tbl)+1);
H= strcat('A1:A',num2str(row_num));
Row_names = readtable('input.xlsx','Range',H);
NTbl2mat=table2array(Inp_Tbl);
CTbl2mat=table2array(Row_names);

    n=(row_num-1);

    m=n;

    C=transpose(CTbl2mat);

    A = NTbl2mat;

    B= pdist(A);

    D =squareform(B);

end
%----------------------------------------------------------------------------

% Single Linkage Function----------------------------------------------------
function SingleLinkage(A,D,C,n,m)

  fprintf('<strong>Single Linkage method(Nearest Neighbor)</strong>\n\n\n')
  CC=C;
  DT=array2table(D,'RowNames',C);
  disp(DT)

  % generating file name ---------------------------------------------------
  Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
  DateName1= strcat('output_Table(Single_Linkage)_',Datetime,'.xlsx');
  ateName2= strcat('output_Dendrogram(Single_Linkage)_',Datetime,'.png');
  %-------------------------------------------------------------------------

  %File Creating -----------------------------------------------------------
     writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,
'WriteMode','overwritesheet','AutoFitWidth',true,
'PreserveFormat',true,'WriteVariableNames',false);
    %------------------------------------------------------------------------

    %Loop for cluster process -----------------------------------------------
    for k=1:m-2
        [numRows,numCols] = size(D);
        U=zeros(numRows,numCols);
```

```
V=zeros(numRows,numCols);

% Finding minimum number ------------------------------------------------
min=100000000000000000000000000000;
for i=1:n
    for j=1:n
        if D(i,j) >0
            if D(i,j) < min
                min=D(i,j);
                min_i = i;
                min_j= j;
            end
        end
    end
end
%-----------------------------------------------------------------------

% loop For Horizontal minimum chekup ------------------------------------
for i=min_j
    for j=1:numCols
        if D(i,j) >0
            if D(i,j) < D(min_i,j)
                V(i,j)= D(i,j);
            else
               V(i,j)= D(min_i,j);
            end
        end
    end
end
[numRows,numCols] = size(D);
%-----------------------------------------------------------------------

% loop For VERTICAL minimum chekup --------------------------------------
for j=min_j
    for i=1:numRows
        if D(i,j) >0
            if D(i,j) < D(i,min_i)
               U(i,j)= D(i,j);
            else
               U(i,j)= D(i,min_i);
            end
        end
    end
end
%-----------------------------------------------------------------------

%reshaping distance matrix ----------------------------------------------
```

```
        for i= min_j
            for j=1:numCols
                D(i,j) = V(i,j);
            end
        end

        for j= min_j
            for i=1:numRows
                D(i,j) = U(i,j);
            end
        end

        D(min_i,:)=[];
        D(:,min_i)=[];
        %------------------------------------------------------------------

        % Auto clustre name generator--------------------------------------
        [numRowsCity,numColsCity] = size(C);
        CN = num2str(abs(m-n+1));
        CNN = ['C' CN];
        %------------------------------------------------------------------

        % adding genereted name to city matrix----------------------------
        C2 = [C(1 : min_j-1),CNN,C(min_j+1:numColsCity)];
        C2(:,min_i)=[];
        C=C2;
        %------------------------------------------------------------------

        %Creating display table-------------------------------------------
        [numRows,numCols] = size(D);
        DT=array2table(D,'RowNames',C2);
        disp(DT)

        sheet = k+1;
        writetable(DT,DateName1,'Sheet',sheet,'WriteRowNames',
true,'WriteMode','overwritesheet','AutoFitWidth',true,
'PreserveFormat',true,'WriteVariableNames',false);
        %------------------------------------------------------------------

        n= n-1;
    end
    %----------------------------------------------------------------------

    %Creating dedrogram---------------------------------------------------
    BB= pdist(A);
    tree = linkage(A,'single');
    leafOrder = optimalleaforder(tree,BB);
    %create cell of labels
    labels = cellstr(CC);
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %-----------------------------------------------------------------

end
%---------------------------------------------------------------------

% complete Linkage Function--------------------------------------------
function CompleteLinkage(A,D,C,n,m)

fprintf('<strong>Complete Linkage method(Farthest Neighbor)
</strong>\n\n\n')

    CC=C;
    DT=array2table(D,'RowNames',C);
    disp(DT)

% generating file name ----------------------------------------------
Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
DateName1= strcat('output_Table(complete_Linkage)_',Datetime,'.xlsx');
DateName2= strcat('output_Dendrogram(complete_Linkage)_',Datetime,'.png');
%-------------------------------------------------------------------

    %File Creating ---------------------------------------------------
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',
true,'WriteMode','overwritesheet','AutoFitWidth',true,
'PreserveFormat',true,'WriteVariableNames',false);
    %-----------------------------------------------------------------



    %Loop for cluster process -----------------------------------------
    for k=1:m-2
        [numRows,numCols] = size(D);
        U=zeros(numRows,numCols);
        V=zeros(numRows,numCols);

        % Finding maximum number --------------------------------------
        min=1000000000000000000000;
        for i=1:n
            for j=1:n
                if D(i,j) >0
                    if D(i,j) < min
                        min=D(i,j);
                max_i = i;
                max_j= j;
            end
        end
    end
end
%----------------------------------------------------------------------

% For Horizontal maximum chekup ---------------------------------------
for i=max_j
    for j=1:numCols
        if D(i,j) >0
            if D(i,j) > D(max_i,j)
                V(i,j)= D(i,j);
            else
              V(i,j)= D(max_i,j);
            end
        end
    end
end
[numRows,numCols] = size(D);
%-------------------------------------------------------------------------

% For VERTICAL maximum chekup --------------------------------------------
for j=max_j
    for i=1:numRows
        if D(i,j) >0
            if D(i,j) > D(i,max_i)
              U(i,j)= D(i,j);
            else
              U(i,j)= D(i,max_i);
            end
        end
    end
end
%-------------------------------------------------------------------------

%reshaping distance matrix -----------------------------------------------
for i= max_j
    for j=1:numCols
        D(i,j) = V(i,j);
    end
end

for j= max_j
    for i=1:numRows
        D(i,j) = U(i,j);
```

```
            end
        end

        D(max_i,:)=[];
        D(:,max_i)=[];
        %----------------------------------------------------------------

        % Auto clustre name generator-------------------------------------
        [numRowsCity,numColsCity] = size(C);
        CN = num2str(abs(m-n+1));
        CNN = ['C' CN];
        %----------------------------------------------------------------

        % adding genereted name to city matrix----------------------------
        C2 = [C(1 : max_j -1),CNN,C(max_j+1:numColsCity)];
        C2(:,max_i)=[];
        C=C2;
        %----------------------------------------------------------------

        %Creating display table-------------------------------------------
        [numRows,numCols] = size(D);
        DT=array2table(D,'RowNames',C2);
        disp(DT)
        sheet = k+1;

        writetable(DT,DateName1,'Sheet',sheet,'WriteRowNames',
true,'WriteMode','overwritesheet','AutoFitWidth',true,
'PreserveFormat',true,'WriteVariableNames',false);
        %----------------------------------------------------------------

        n= n-1;
    end
    %--------------------------------------------------------------------

    %Creating dedrogram---------------------------------------------------
    BB= pdist(A);
    tree = linkage(A,'complete');
    leafOrder = optimalleaforder(tree,BB);
    %create cell of labels
    labels = cellstr(CC);
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %--------------------------------------------------------------------
end
%------------------------------------------------------------------------
```

```
% Average Linkage Function------------------------------------------------------------
function AverageLinkage(A,D,C,n,m)

    na= n;
    nb= n;
    x= A;
    y= A;

fprintf('<strong>Average Linkage method</strong>\n\n\n')

    CC=C;
    D =pdist(A);
    D2=squareform(D);

    DT=array2table(D2,'RowNames',CC);

% generating file name ----------------------------------------------------
Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
DateName1= strcat('output_Table(Average_Linkage)_',Datetime,'.xlsx');
DateName2= strcat('output_Dendrogram(Average_Linkage)_',Datetime,'.png');
    %---------------------------------------------------------------------------

    %File Creating -------------------------------------------------------------
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',
true,'WriteMode','overwritesheet','AutoFitWidth',true,
'PreserveFormat',true,'WriteVariableNames',false);
    %---------------------------------------------------------------------------

    %Creating Dendrigram--------------------------------------------------------
    disp(DT)
    %generating tree
    tree = linkage(D2,'average');
    %create cell of labels
    labels = cellstr(CC);
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %---------------------------------------------------------------------------
end
%-------------------------------------------------------------------------------

% wards Function------------------------------------------------------------------
function Wards(A,D,C,n,m)

    na=n;
    nb=n;
    x=A;
    y=A;

    CC=C;
    DT=array2table(D,'RowNames',C);
    disp(DT)

    % generating file name ----------------------------------------------
    Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
    DateName1= strcat('output_Table(Wards)_',Datetime,'.xlsx');
    DateName2= strcat('output_Dendrogram(Wards)_',Datetime,'.png');
    %-------------------------------------------------------------------

    %creating file -----------------------------------------------------
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,
'WriteMode','overwritesheet','AutoFitWidth',true,
'PreserveFormat',true,'WriteVariableNames',false);
    %-------------------------------------------------------------------

    %Creating dedrogram-------------------------------------------------
    BB= pdist(A);
    tree = linkage(A,'ward');
    leafOrder = optimalleaforder(tree,BB);
    %create cell of labels
    labels = cellstr(CC);
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %-------------------------------------------------------------------

end
%-----------------------------------------------------------------------

% centroid Function-----------------------------------------------------
function centroid(A,D,C,n,m)

    na=n;
    nb=n;
    x=A;
    y=A;

    CC=C;
    DT=array2table(D,'RowNames',C);
    disp(DT)

    % generating file name ----------------------------------------------
    Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
    DateName1= strcat('output_Table(Centroid)_',Datetime,'.xlsx');
    DateName2= strcat('output_Dendrogram(Centroid)_',Datetime,'.png');
    %-------------------------------------------------------------------

    %creating file------------------------------------------------------
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,
'WriteMode','overwritesheet','AutoFitWidth',true,
'PreserveFormat',true,'WriteVariableNames',false);
    %-------------------------------------------------------------------

    %Creating dedrogram--------------------------------------------------
    BB= pdist(A);
    tree = linkage(A,'centroid');
    leafOrder = optimalleaforder(tree,BB);
    %create cell of labels
    labels = cellstr(CC);
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %--------------------------------------------------------------------
end
%------------------------------------------------------------------------

% median Function--------------------------------------------------------
function median(A,D,C,n,m)

    na=n;
    nb=n;
    x=A;
    y=A;

    CC=C;
    DT=array2table(D,'RowNames',C);
    disp(DT)

    % generating file name ---------------------------------------------
    Datetime = datestr(now,'mmmm_dd_yyyy_HH_MM_SS_PM');
    DateName1= strcat('output_Table(Median)_',Datetime,'.xlsx');
    DateName2= strcat('output_Dendrogram(Median)_',Datetime,'.png');
    %-------------------------------------------------------------------

    %creating file------------------------------------------------------
    writetable(DT,DateName1,'Sheet',1,'WriteRowNames',true,
'WriteMode','overwritesheet','AutoFitWidth',true,
'PreserveFormat',true,'WriteVariableNames',false);
    %-------------------------------------------------------------------

    %Creating dedrogram--------------------------------------------------
    BB= pdist(A);
    tree = linkage(A,'median');
    leafOrder = optimalleaforder(tree,BB);
    %create cell of labels
    labels = cellstr(CC);
    %plot dendogram with custom labels
    dendrogram(tree, 0, 'Labels', labels, 'orientation', 'left')
    saveas(gcf,DateName2)
    %------------------------------------------------------------------------------
end
%----------------------------------------------------------------------------------
```

Bibliography

[1] Bock, T. (2022) What is a dendrogram?, Displayr. Available at: https://www.displayr.com/what-is-dendrogram/ (Accessed: November 30, 2022).

[2] Stephanie (2021) Hierarchical clustering / dendrogram: Simple definition, examples, Statistics How To. Available at: https://www.statisticshowto.com/hierarchical-clustering/ (Accessed: November 30, 2022).

[3] Tree (no date) Dendrogram plot - MATLAB. Available at: https://www.mathworks.com/help/stats/dendrogram.html (Accessed: November 30, 2022).

[4] X (no date) Agglomerative hierarchical cluster tree - MATLAB. Available at: https://www.mathworks.com/help/stats/linkage.html (Accessed: November 30, 2022).

[5] X (no date) Scatter plot - MATLAB. Available at: https://www.mathworks.com/help/matlab/ref/scatter.html (Accessed: November 30, 2022).

[6] Cluster analysis (2022) Wikipedia. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Cluster$_{a}nalysis(Accessed : December4, 2022)$.