# Exploratory data and unsupervised learning analysis for developing hidden patterns from wholesale customers dataset

Yan Lin

## Introduction

Understanding consumer behaviour is crucial for companies seeking to capitalise on market possibilities and improve customer satisfaction. In this research, we analyse a wholesale customer dataset to reveal hidden patterns and possible client groupings, which can be used to inform tailored marketing tactics and enhance operational efficiency.

The unlabeled and high dimensionality of the dataset containing annual expenditure amounts on several product categories from distinct clients of a wholesale distributor pose unique obstacles. Data preparation, appropriate parameter selection, result interpretation, and cluster stability are the significant challenges in this report. Additionally, the absence of ground truth labels for validation requires relying on domain knowledge and external resources to assess the significance and relevance of identified customer segments. To address these issues, we employ exploratory data analysis and Principal Component Analysis (PCA) for dimensionality reduction, and we construct and compare the performance of two clustering algorithms, K-means and Hierarchical Clustering.

This report's findings have considerable significance for wholesale distributors' marketing strategy and resource allocation refinement. Moreover, our critical approach to the development and evaluation of these unsupervised learning techniques contributes to the wider academic discourse on the effectiveness of various data-driven methods in revealing hidden patterns within complex, real-world datasets.

## Exploratory data analysis

### *Characteristics of each variable*

The raw dataset consists of 440 observations and 6 variables with 0 missing value. See Appendix 1 Table 1 for a summary of the variables. The data are continuous, allowing for a range of various statistical techniques and visualizations.

Based on the descriptive statistics of the variables (See Appendix 1 Table 2), we can observe that the means and medians of each variable are relatively close, indicating that the distributions may not be severely skewed, as confirmed in the histograms (See Appendix 1 Figure 1), with only Fresh and Grocery having a slight negative skewness. The higher standard deviation of Fresh and Detergents_Paper indicates greater dispersion of their data.

### *Relationship between variables*

According to the correlation matrix (See Appendix 1 Table 3) and the covariance matrix (See Appendix 1 Table 4), we can find a certain positive correlation between Milk, Grocery, and Detergents_Paper, all of which may be popular with the same client segments. Little correlations and volatility exist between the remaining variables.

We should investigate this not only from a numerical perspective, but also from a graphical one. Since the scatterplot matrix (Figure 5 in Appendix 1, combined from Figure 2, Figure 3 in Appendix 1) and the correlation plot (Figure 4 in Appendix 1), it can again be seen that Grocery variable is highly correlated with Detergents_Paper variable (correlation =0.80). They are also highly correlated with the Milk variable (correlation = 0.76, 0.68 respectively). This implies that the three variables share redundant information and the use of PCA can remove this redundancy, thereby reducing the dimension of the data.

### *Outliers and extreme values*

As shown in the boxplot of the original data set (Top half of Figure 2, Appendix 1), the three variables Fresh, Frozen, and Delicassen have a great number of outliers. Outliers and extremes are defined as being between 1.5-3 IQR and above 3 IQR, respectively. Whilst the total number of original observations is 440, there are 48 outliers and extremes combined. Since the proportion of outliers and extremes is relatively high and cannot be deleted directly, this would have a stronger impact on the subsequent analysis.

Instead, we locate them, convert them to missing values, and then use Multiple Imputation by Chained Equations to predict the missing values. A boxplot of the cleaned dataset shows (Bottom half of Figure 6, Appendix 1) that the outliers and extremes are significantly reduced.

### *Crucial pre-processing step： normalisation*

For this new dataset, which has been filtered of outliers and extremes, the first step is to examine its normality. Even though subsequent dimensionality reduction and clustering analyses can tolerate non-normal data, it's often better to deal with normally distributed datasets. In terms of the Shapiro-Wilk test for normality (Appendix 1, Table 5) and the Normal Q-Q plot (Appendix 1, Figure 7), the only two variables that appear to be normally distributed are Milk and Frozen.

The next step is estimating the variance, standard deviation, and mean of this new (outlier-removed) dataset, which are vastly different that they must be scaled, which is the most critical step. This is because the variance of the Detergents_Paper variable is the largest, and the means of the Fresh, Milk and Grocery variables are significantly bigger than the other three. The purpose of standardisation is to make the scales consistent across all variables so that errors caused by scale differences can be avoided in following analyses. After normalisation, each variable has a mean of 0 and a variance of 1.

## Dimension reduction

### *Motivation*

It is known that the dataset includes redundant information and consists entirely of continuous numeric variables, thus we considered dimensionality reduction and decided for principal component analysis (PCA) instead of correspondence analysis (CA) and factor analysis (FA). CA is appropriate for categorical variables, FA seeks to find underlying factors (i.e., unobserved variables), and the results of CA and FA generally include more underlying structure and interpretation, making the results more challenging to read and comprehend.

PCA is a low-dimensional representation of the data that captures the meaningful properties of the original dataset, providing easier interpretation and visualization, i.e., retains as much information as possible about the data. Even though all 440 observations exist in a 6-dimensional space, not all dimensions have the same

value. If dimensionality reduction is not considered, we then need to consider $\binom{p}{2} = \frac{p(p-1)}{2} = \frac{6\times5}{2} = 15$ scatter plots, which are very difficult to observe. The essence of PCA is to convert vector bases.

### *Conduct the Principal Component Analysis (PCA)*

PCA comprises two techniques: singular value decomposition and eigendecomposition. This report utilises the singular value decomposition (prcomp), which in some cases is more stable. The first principal component of a set of variables $\{X_1, X_2, \ldots, X_6\}$ is the combination with the largest variance of the standardised linear combination of the variables:

$$Z_1 = \phi_{11}X_1 + \phi_{12}X_2 + \cdots + \phi_{16}X_6 \qquad \phi_1 = \arg max_{vv^T=1} (\boldsymbol{X}v)^T(\boldsymbol{X}v)$$

Where $\phi_{11}, \phi_{12}, \ldots, \phi_{16}$ refers to the loadings (directions) of PC 1, which form the loading vector of the principal component $\phi_1 = (\phi_{11}\ \phi_{12} \ldots \phi_{16})$, limiting the sum of squares of these coefficients of correlation $({\phi_1}^T\phi_1) = 1$. Note that $Z_1 = \boldsymbol{X}\phi_1$ is the PC 1 score, and the amount of variance explained by $\phi_1 \sum \phi_1$. PC 1 is the most dispersed data distribution, meaning that there is the least loss of information in the data.

The second principal component is, similarly, the one with the largest variance of the various linear combinations that are not correlated with $Z_1$:

$$Z_2 = \phi_{21}X_1 + \phi_{22}X_2 + \cdots + \phi_{26}X_6 \qquad \phi_2 = \arg max_{vv^T=1, v^T\phi_1=0} (\boldsymbol{X}v)^T(\boldsymbol{X}v)$$

${\phi_2}^T\phi_2 = 1$, ${\phi_2}^T\phi_1 = 0$ shows that to make $Z_1 \& Z_2$ uncorrelated is equivalent to making the direction of $\phi_2$ perpendicular with the direction of $\phi_1$ (Standard orthogonal). This dataset ($n = 440, p = 6$) can have up to $\min(n-1, p) = 6$ principal components, hence $Z_3 \ldots Z_6$ is similar ( Z's can be seen in Table 2 Appendix 2), whilst the principal component loadings vector $\phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6$ can be found in Table 1 in Appendix 2. PC1 score for 1st row was computed:

$$Z_1 = -0.07960217 * Fresh + 0.5483290 * Milk + \cdots + 0.26479059 * Delicassen = 1.3993911$$

Once these principal components have been calculated, a low-dimensional plot of the data can be made by treating each two principal components as a pair of coordinates.

### *Choose an appropriate dimension and justification*

The proportion of variance explained is a good way to quantify how much structure is captured by dimension reduction. The $j^{th}$ principal component explains $\frac{\lambda_j}{\sum_{j=1}^{6}\lambda_j}$ of the total variance in the data. Based on the Screeplot (Figure 1 in Appendix 2), we found that the first principal component explained 44.58% of the variance in the data, the second principal component explained 26.14% of the variance and the third principal component explained 11.90%. The rule of thumb recommends retaining the component that explains 80-90% of the variability in the original dataset. Furthermore, according to both graphs, the proportion of explained variance for various principal components and the proportion of cumulative explained variance for various principal components (Figure 2 in Appendix 2), the elbow method was used to determine k = 3, after which the fall began to slow.

Focusing on the eigenvalues of the principal components, the eigenvalue of 2.6748119 for PC 1 and 1.5682499 for PC 2 is significantly greater than 1, but the third principal component eigenvalue is $0.713.986 < 1$. The

Kaiser criterion suggests keeping primary components with eigenvalues exceeding 1. Nonetheless, this criterion is not absolute and may oversimplify the issue in this dataset. Screeplot is only a guide for selecting the required number of PCs. Choosing the number of principal components requires a trade-off between dimensionality reduction, capturing key information and interpretation. From now on, we analyse the three important PCs, keeping 82.62% of the variance in the original data.

### *Further analysis of principal components after dimensionality reduction*

Biplot can be used to visualize principal component loadings and principal component scores. From the 2D biplot (Figure 4, Appendix 2), first loading vector places approximately equal weight on Milk and Grocery, Detergents_Paper, followed by Delicassen, and least to Frozen and Fresh. The second loading vector places most of its weight on Frozen and Fresh, followed by Delicassen. A more careful and intuitive observation can be made from the 3D biplot (Figure 5, Appendix 2): Milk, Grocery, and Detergents_Paper on one plane, Frozen, Fresh, and Delicassen on the other, and Fresh & Delicassen on the third, which differs slightly from the conclusion drawn from the 2d biplot. This is the distinction between keeping two-dimensional and three-dimensional principal components.

The quality of representation of the variables is called the squared cosine (cos2) or the squared correlations. The high squared correlation indicates a good representation of the variable on the principal component. The same conclusions as for the 3d biplot can be drawn from Table 3 in Appendix 2 and Figure 5 in Appendix 2 regarding Cos2 on 3 dimensions. The correlation plot (Figure 6 in Appendix 2) and correlation table (Table 4 in Appendix 2) show the relationships between all variables and PCs; positively correlated variables are grouped together, and negatively correlated ones are positioned on the opposite sides. But the 2D correlation plot is not sufficient to show all the information, we should use all information from table to draw conclusions about the full correlation, in line with the 3D biplot again.

The contribution of a variable to a given principal component is (in percentage): (cos2 * 100) / (total cos2 of the component). The larger value of the contribution indicates more contribution of the variable to the PC. There are 6 variables in total, 100/6 equals approximately 16.66, more than 16.66 means an outstanding contribution. Observing the contribution graphs for the three dimensions in Figure 7 Appendix 2, we find that Grocery, Milk, Detergents_Paper contributed heavily to PC1 while the other three variables (Frozen, Fresh, Delicassen) contributed heavily to PC2, and Fresh & Delicassen contributed heavily to PC3. Observe that the interpretation of correlation, cos2 and contribution are similar.

## Cluster analysis

### *Motivation*

Clustering is an unsupervised learning algorithm, a technique for finding subgroups or classes in a data set. There is a certain amount of dissimilarity in these 440 observations. In this algorithm, groups are undefined and unknown, so we try to learn about the groups themselves, such as numbers and attributes. We apply the clustering algorithm to unlabelled data and estimate the true group structure.

### *Justification*

A total of five clustering methods are covered in the class: K-means algorithm, K-medoids algorithm (Partitioning Around Medoids or Clustering Large Applications), Hierarchical clustering, Hierarchical K-means Clustering, and Model-based clustering (find the max BIC).

I select the K-means and Hierarchical clustering algorithms. This is why K-means and Hierarchical clustering are two of the most popular and well-established clustering methods. Both types of clustering have relatively simple algorithms that are easy to understand and implement. The K-means algorithm has better computational efficiency, especially on larger data sets. Hierarchical clustering, despite being less scalable, provides a new view of the data through a tree diagram, which is useful for establishing the appropriate number of clusters and comprehending the data's structure. These methods complement each other, as K-means requires a predetermined number of clusters and is sensitive to initial conditions, while Hierarchical clustering is more robust and doesn't need a pre-specified cluster number. In some cases, other methods like K-medoids, Hierarchical K-means, or model-based clustering may be more appropriate, but K-means and Hierarchical clustering offer a solid starting point for analysis.

### *K-means clustering*

K-means clustering satisfies two properties:

$C_1 \cup C_2 \cup \ldots \cup C_k = \{1,2,\ldots,440\}$, *and* $C_k \cap C_{k'} = \emptyset$ *for all k=k′*

*The essence of K-means is the minimization of W:* $minimize_{C_{1,\ldots,C_k}}\{\sum_{k=1}^{k} W(C_k)\}$

The first step in K-means clustering is to select a fixed number of groups: K. I select two methods: the total within-cluster sum of square (WSS) and the silhouette coefficient. The left panel of Figure 1 in Appendix 3 shows that the curve bends (knee) when K = 3, which indicates that the between-cluster sum of squares are maximised, whilst total within-cluster sum of squares are minimised. This bend indicates that additional clusters beyond the third have little improvement.

The silhouette width values range from -1 to 1, where a value near 1 indicates that the sample is closer to its own cluster, indicating better clustering, and a value near -1 implies that the sample is closer to another cluster and may have been misclustered, indicating poor clustering. However, in the right panel of Figure 1 in Appendix 3, it can be observed that the silhouette width is highest at K=2, which also indicates that K=2 is an optimal number of clusters. To assess the optimal k values more comprehensively, I performed k-means clustering analysis using K=2 and K=3 respectively.

Next, we try 25 different random choose 2 (or 3) initial cluster centroids for $(\bar{X}_1, \bar{X}_{2)}$ $(or\ \bar{X}_1, \bar{X}_2, \bar{X}_3)$ out of the 440 observations. Assign the observations to the closest centroid. Then label each observation based on the closest centroid and replace each cluster centroid by computing the average of observations in its cluster. Repeat until the assignments do not change.

When K=2, one cluster size is 252 and the other cluster is 188, and between_SS / total_SS = 31.2%, which has similar interpretation as $R^2$ in linear regression (percentage of variance explained by cluster means). The total within-cluster sum of squares at this point is 1811.011, which is 1021.4575 and 789.5534 respectively. When K=3, one cluster size is 171, another size is 160 and a third cluster size is 109. It explains 48.7 % of the variance in the mean of the clusters. The total within-cluster sum of squares at this point is 1531.347.

Appendix 3 Figures 2 & 3 provide a direct visualisation of the data for 2 clusters/ 3 clusters respectively, revealing that there is not a good separation and overlap between clusters. This may be influenced by the

high-dimensional data, resulting in poorly defined boundaries between the clusters. Nevertheless, we can visualise the clustering results using principal components if the number of variables is greater than 2(Figure 4 & 5, Appendix 3). It uses the first two principal components to better capture the data's variability, thereby making the separation between clusters more apparent (the first two principal components account for approximately 71% of the variability). We found that the 2 clusters explained the data information better, with relatively clear boundaries and with the two centres at some distance.

We can also perform a comparison by visualising the mean silhouette width of each cluster (Figures 6 , Appendix 3), which is 0.28 for 2 clusters and has a slightly negative sign for the 2nd cluster; 0.22 for the 3 clusters, and both the first and second clusters had some negative signs and did not perform as well as the 2 clusters. Therefore, the next compares the relevant results of the k-mean algorithm for only 2 selected clusters.

### *Hierarchical clustering*

Hierarchical clustering is an alternative approach which does not require a particular choice of K. The general idea is to create a hierarchy of partitions, such that the clustering at each level of the hierarchy is obtained by merging two or more groups from the previous level.

In this report, we first calculate the dissimilarity between all $\binom{p}{2} = \frac{p(p-1)}{2} = \frac{6\times5}{2} = 15$ pairs of the 440 observations, treating each observation as a separate class. Compared to the Euclidean distance, the Manhattan distance can better handle the problem of different scales of the axes, as it is the sum of the absolute values of the distance differences along the axes, but it cannot capture the non-linear dependence between the two variables, hence the Euclidean distance is used for all degrees of dissimilarity in this paper.

We then perform a hierarchical clustering algorithm, starting with $i = 440, 439, \ldots 2$. At each iteration, we compare the dissimilarity between any two classes and identify the pair with the smallest dissimilarity. These two classes are then combined, and the height at which they merge in the dendrogram represents the dissimilarity between them. We subsequently calculate the dissimilarity between the remaining $i - 1$ new classes for the next iteration.

There are three most common dissimilarity score between groups (called linkage) in hierarchical clustering: complete linkage, average linkage, single linkage. Dissimilarity scores between merged clusters only increases as we run the algorithm. Hence, we can draw a proper dendrogram, where the height of a parent is always higher than the height of its daughters. This property is called no inversion property. Complete linkage tends to produce more compact clusters: $d_{complete}(G,H) = max_{i\epsilon G, j\epsilon H} d_{ij}$.

These three select the optimal linkage, which can be calculated by agglomerative coefficient (ac). The AC measures the dissimilarity of an object to the first cluster it joins, divided by the dissimilarity of the final merger in the cluster analysis, averaged across all samples. The AC represents the strength of the clustering structure; values closer to 1 indicates that clustering structure is more balanced. Complete linkage in this dataset had the highest AC value of 0.9037892, followed by average linkage at 0.7944373 and finally single linkage at 0.6523917.

Figure 7 Appendix 3 uses silhouette measurements as the basis for the best cluster yielding a cluster of 2. D efferent than k-means, we can visualise the object by using a dendrogram of 2-clusters (Figure 8, Appendix 3). This will enable us to determine the point to cut the tree, resulting in the number of clusters. The height axis on the dendrogram displays the dissimilarity between observations and/or clusters. The horizontal bars

indicate the point at which two clusters/observations are merged. The height of root node is about 10, while the height of leaf node is about 0.5.

Figures 9 & 10 in Appendix 3 present the improved visualisation of the hierarchical clustering and dendrograms of the 2 clusters (we can now clearly see the observations), and the observations 89 and 352 merge at a height of about 1. Figure 11 in Appendix 3 illustrates the improved phylogenetic tree, with the fusion points of the two clusters much clearer, for example, observation 401 and 172 merge at a height of about 1, and observation 322 and 310 merge at an altitude of about 1.

## Discussion and conclusion

Table1- Advantages and disadvantages of the two types of clustering

| | **Pros** | **Cons** |
|---|---|---|
| **K-means clustering** | Computationally efficient and suitable for large-scale datasets; Allows for the selection of the best result across multiple runs; Intuitive and easy to implement. | Requires the predetermined (fixed) number of clusters (k value); Sensitive to the initial centroid selection, possibly getting trapped in a local optimum; Sensitive to outliers and may not perform well with irregularly shaped clusters. |
| **Hierarchical clustering** | No need to pre-determine the number of clusters; Provides a visual representation of the clustering structure through a dendrogram; Better suited for discovering irregularly shaped clusters; Can provide a hierarchy of cluster. | Computationally intensive, especially for larger datasets; Choice of linkage method and distance metric can significantly influence the clustering results; Difficult to determine the optimal number of clusters from the dendrogram, often requiring subjective interpretation; Not robust against outliers, which can affect the overall clustering structure. |

Both PCA and clustering techniques require choosing appropriate parameters, such as the number of principle components to retain or the number of clusters to form. Selecting the most appropriate parameters is the biggest challenge in this report, frequently requiring a balance between complexity and interpretability.

To compare the performance of K-means clustering and Hierarchical clustering, the Silhouette score is a widely used metric. According to Table 2, we can see that k-means clustering performs better in creating cohesive, well-separated clusters for this dataset compared to the hierarchical clustering method. It must be noted, however, that the Silhouette scores are relatively low, suggesting that there may be room for improvement in cluster assignment, or that the data itself may not have a strong intrinsic clustering structure.

Table2- Silhouette scores for the two clustering methods

| **Silhouette score** | **K-means clustering** | **Hierarchical clustering** |
|---|---|---|
| 2 clusters | 0.2849978 | 0.2223309 |

# References

1. Margarida G. M. S. Cardoso. UCI Machine Learning Repository (2014)

https://archive.ics.uci.edu/ml/datasets/wholesale+customers

2. University of Florida, D. of B. (n.d.). Outliers. Retrieved June 29, 2022, from

https://bolt.mph.ufl.edu/6050-6052/unit-1/one-quantitative-variable-introduction/understanding-

outliers/#:~:text=The%203(IQR)%20criterion%20tells.

3. Wikipedia Contributors. (2019, September 3). Shapiro–Wilk test. Wikipedia; Wikimedia Foundation. https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test

4. Kaiser Rule - Displayr. (n.d.). Docs.displayr.com. Retrieved April 3, 2023, from https://docs.displayr.com/wiki/Kaiser_Rule

5. Using code from Labs on Principal component analysis and clustering by Dr. H. Maeng,

https://hmaeng.github.io/devul23/index.htmll accessed 10 Mar 23

# Appendices

## Appendix 1- Exploratory data analysis

*Table 1- Variables*

| Serial | Name | Type | Note |
|---|---|---|---|
| 1 | Fresh | Numeric | annual spending (m.u.) on fresh products |
| 2 | Milk | Numeric | annual spending (m.u.) on milk products |
| 3 | Grocery | Numeric | annual spending (m.u.) on grocery products |
| 4 | Frozen | Numeric | annual spending (m.u.) on frozen products |
| 5 | Detergents_Paper | Numeric | annual spending (m.u.) on detergents and paper products |
| 6 | Delicassen | Numeric | annual spending (m.u.) on delicatessen products |

*Table 2- Summary of descriptive statistics for variables*

| Name | Min | 1st quartile | Median | Mean | 3rd quartile | Max | Standard deviation |
|---|---|---|---|---|---|---|---|
| Fresh | 1.10 | 8.05 | 9.05 | 8.73 | 9.74 | 11.63 | 1.48 |
| Milk | 4.01 | 7.33 | 8.20 | 8.12 | 8.88 | 11.21 | 1.08 |
| Grocery | 1.10 | 7.67 | 8.47 | 8.44 | 9.27 | 11.44 | 1.12 |
| Frozen | 3.22 | 6.61 | 7.33 | 7.30 | 8.18 | 11.02 | 1.28 |
| Detergents_Paper | 1.10 | 5.55 | 6.71 | 6.79 | 8.27 | 10.62 | 1.72 |
| Delicassen | 1.10 | 6.01 | 6.87 | 6.67 | 7.51 | 10.78 | 1.31 |

*Figure 1- Histogram of each variable*

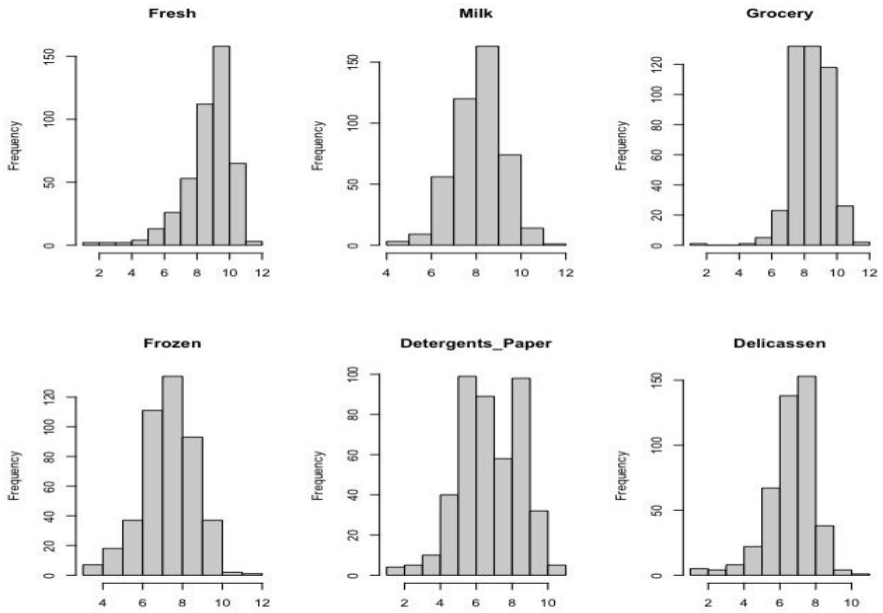


*Table 3- Correlation matrix*

| Correlation | Fresh | Milk | Grocery | Frozen | Detergents_Paper | Delicassen |
|---|---|---|---|---|---|---|
| Fresh | 1.00000000 | -0.01983398 | -0.1327129 | 0.38399622 | -0.1558706 | 0.2551862 |
| Milk | -0.01983398 | 1.00000000 | 0.7588509 | -0.05531589 | 0.6779424 | 0.3378325 |
| Grocery | -0.13271291 | 0.75885090 | 1.00000000 | -0.16452386 | 0.7963977 | 0.2357280 |
| Frozen | 0.38399622 | -0.05531589 | -0.1645239 | 1.00000000 | -0.2115757 | 0.2547183 |
| Detergents_Paper | -0.15587064 | 0.67794242 | 0.7963977 | -0.21157566 | 1.00000000 | 0.1667350 |
| Delicassen | 0.25518621 | 0.33783253 | 0.2357280 | 0.25471830 | 0.1667350 | 1.00000000 |

*Table 4- Covariance matrix*

| Covariance | Fresh | Milk | Grocery | Frozen | Detergents_Paper | Delicassen |
|---|---|---|---|---|---|---|
| Fresh | 2.19061115 | -0.03174424 | -0.2192436 | 0.73005776 | -0.3970388 | 0.4950930 |
| Milk | -0.03174424 | 1.16935097 | 0.9159250 | -0.07683692 | 1.2616858 | 0.4788735 |
| Grocery | -0.21924356 | 0.91592498 | 1.2458393 | -0.23588891 | 1.5298434 | 0.3448967 |
| Frozen | 0.73005776 | -0.07683692 | -0.2358889 | 1.65004297 | -0.4677344 | 0.4288986 |
| Detergents_Paper | -0.39703877 | 1.26168576 | 1.5298434 | -0.46773437 | 2.9619103 | 0.3761488 |
| Delicassen | 0.49509296 | 0.47887350 | 0.3448967 | 0.42889865 | 0.3761488 | 1.7182795 |

*Figure 2- Scatterplot matrix on variables*

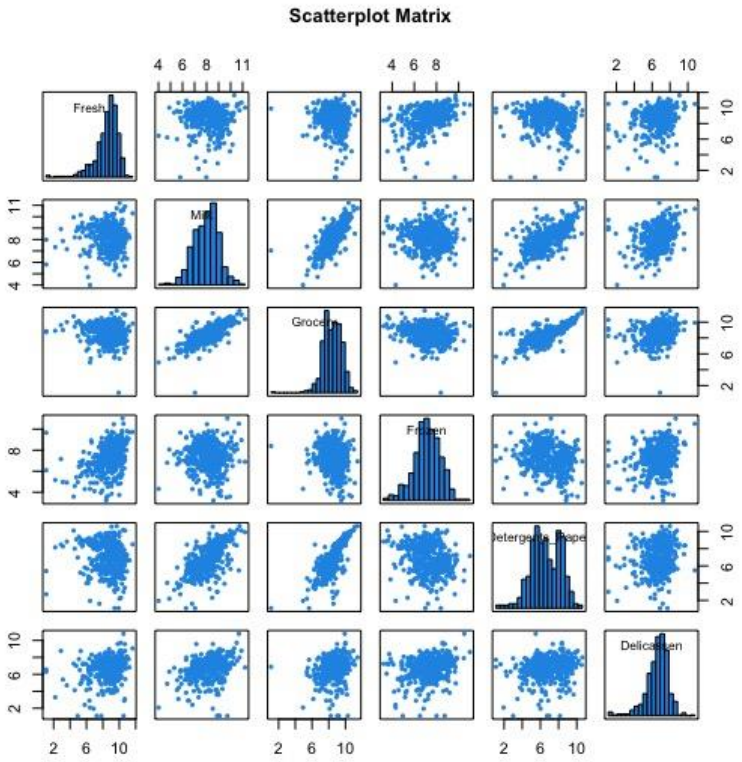


*Figure 3- Scatterplot matrix smoothed version*

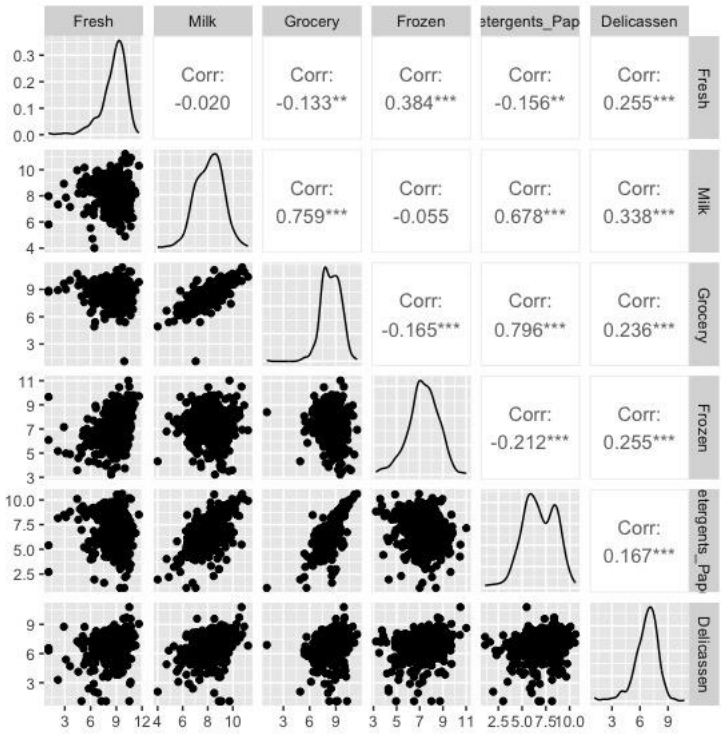



*Figure 4- Correlation Plot*

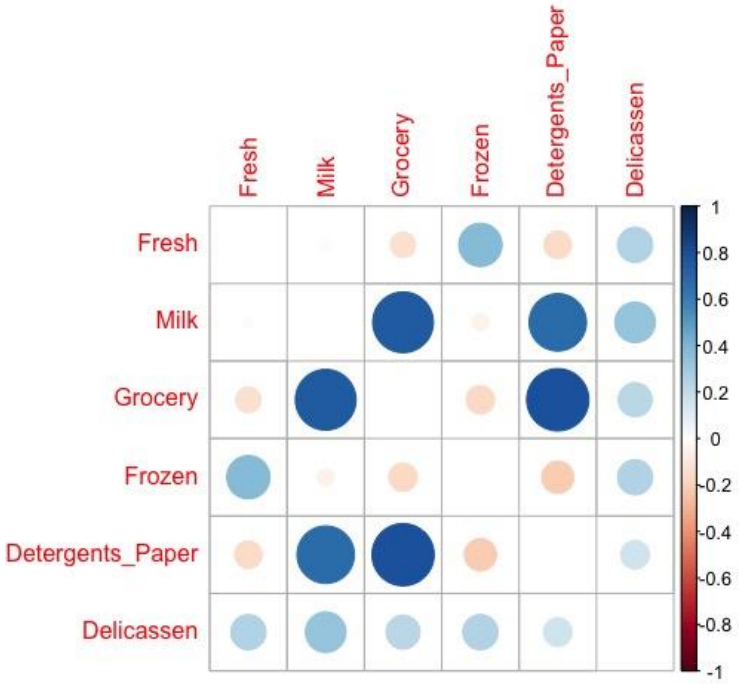


*Figure 5- Scatter plot + density line + correlation coefficient*

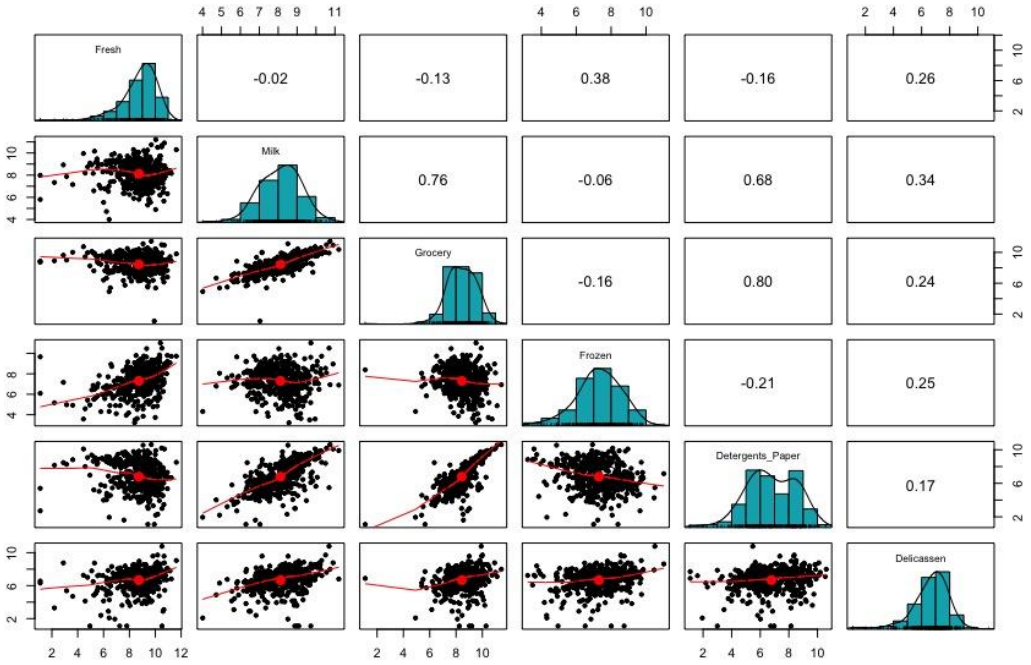
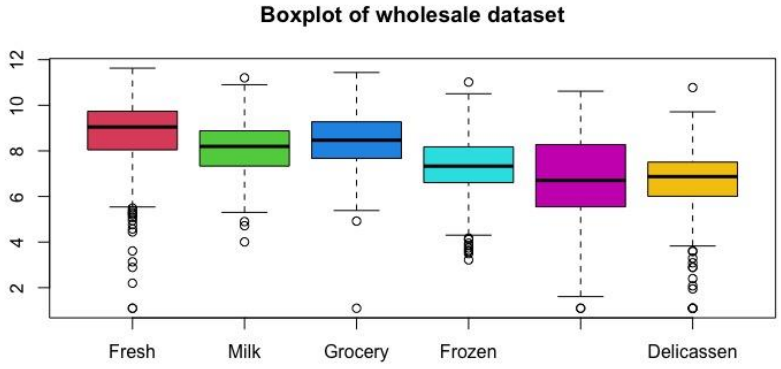



*Figure 6- Boxplots of the two data sets*



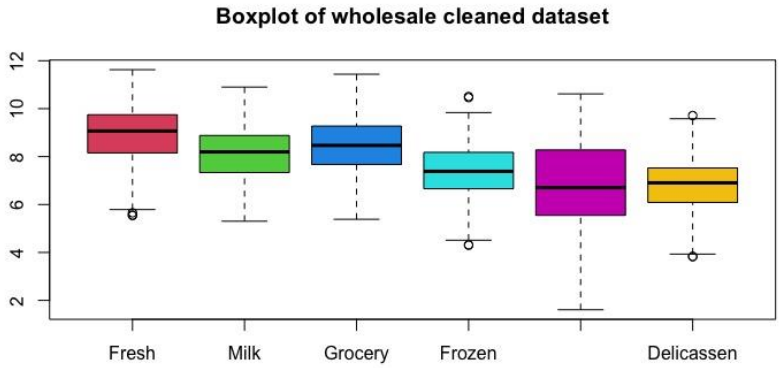
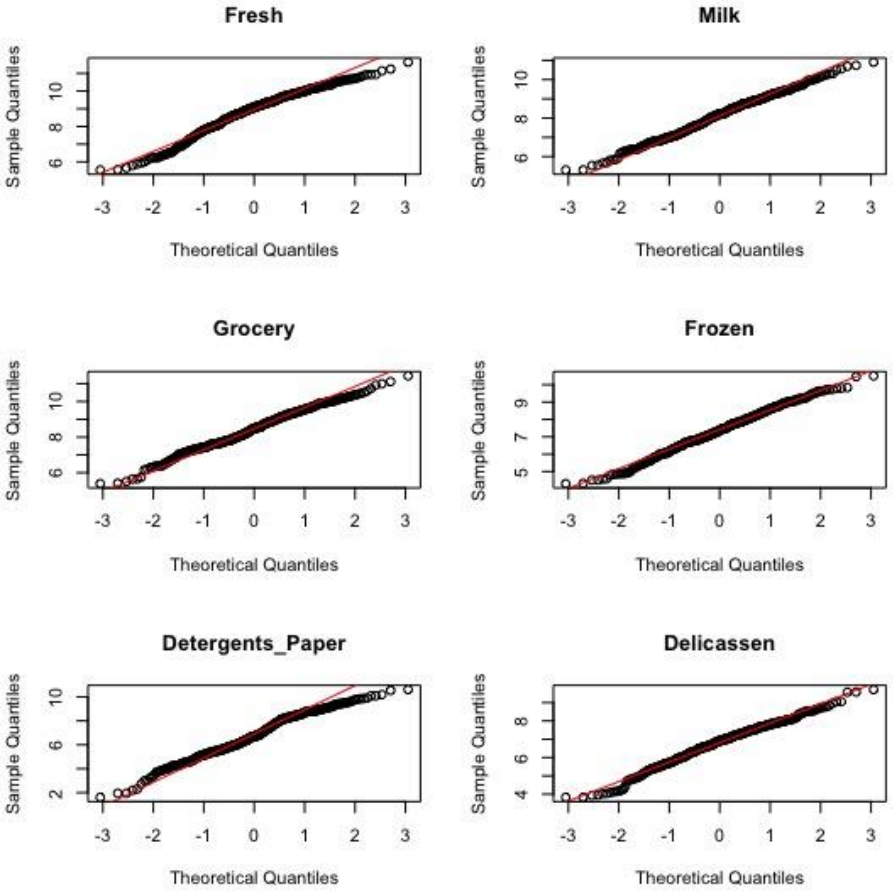



*Table 5- Shapiro-Wilk test for normality of variables*

| Normality | p-value | test-statistic |
|---|---|---|
| Fresh | 0.00 | 0.97 |
| Milk | 0.13 | 0.99 |
| Grocery | 0.04 | 0.99 |
| Frozen | 0.16 | 0.99 |
| Detergents_Paper | 0.00 | 0.98 |
| Delicassen | 0.00 | 0.99 |

*Figure 7- QQ plot for each variable*



# Appendix 2- PCA

*Table 1- PC loading vectors*

| | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 |
|---|---|---|---|---|---|---|
| Fresh | -0.07960217 | 0.584646982 | 0.72072507 | -0.35728228 | 0.02319883 | 0.06494584 |
| Milk | 0.54832908 | 0.085764390 | -0.03721678 | 0.05476168 | 0.75407286 | 0.34492010 |
| Grocery | 0.56965888 | 0.004009579 | 0.16689796 | 0.06784982 | -0.04517159 | -0.80060832 |
| Frozen | -0.11011064 | 0.626149254 | -0.14010051 | 0.75726994 | -0.03556735 | -0.03823359 |
| Detergents_Paper | 0.53502250 | -0.120859371 | 0.24128972 | 0.21855726 | -0.60008174 | 0.48276065 |
| Delicassen | 0.26479059 | 0.494111752 | -0.61111904 | -0.49347937 | -0.25967736 | 0.03631554 |

*Table 2- PC scores (z's)*

| | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 |
|---|---|---|---|---|---|---|
| [1,] | 1.3993911 | -0.57806373 | 0.5391907 - | -1.4091525 | 0.3188187 | 0.4066094 |
| [2,] | 1.5150716 | 0.36106451 | -0.1743523 | 0.0328043 | 0.1017501 | 0.2017238 |
| [3,] | 1.7118995 | 1.15289457 | -1.1576647 | -0.4405273 | -0.3664217 | -0.3856283 |
| … | … | … | … | … | … | … |

*Figure 1- Scree plot*

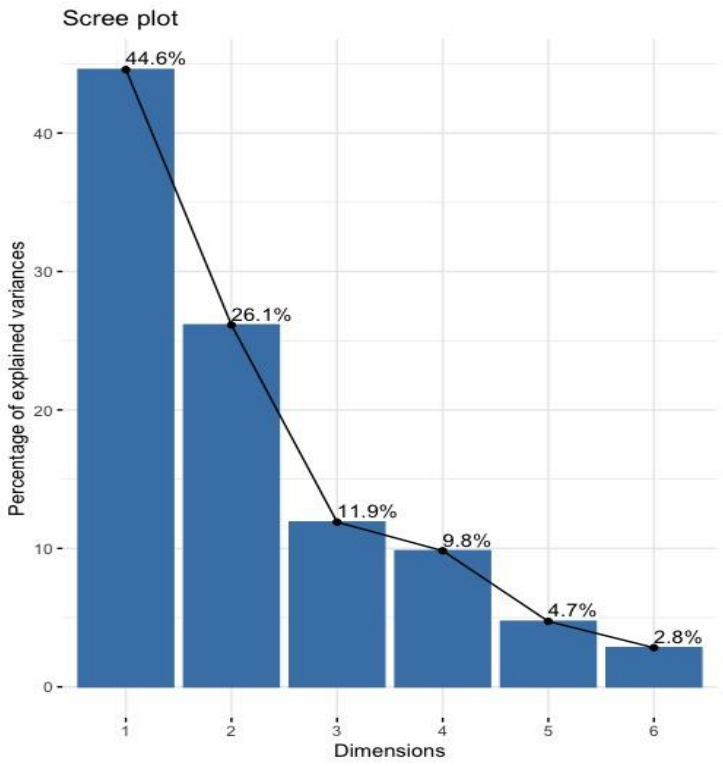


*Figure 2- The proportion of variance explained by each principal component*

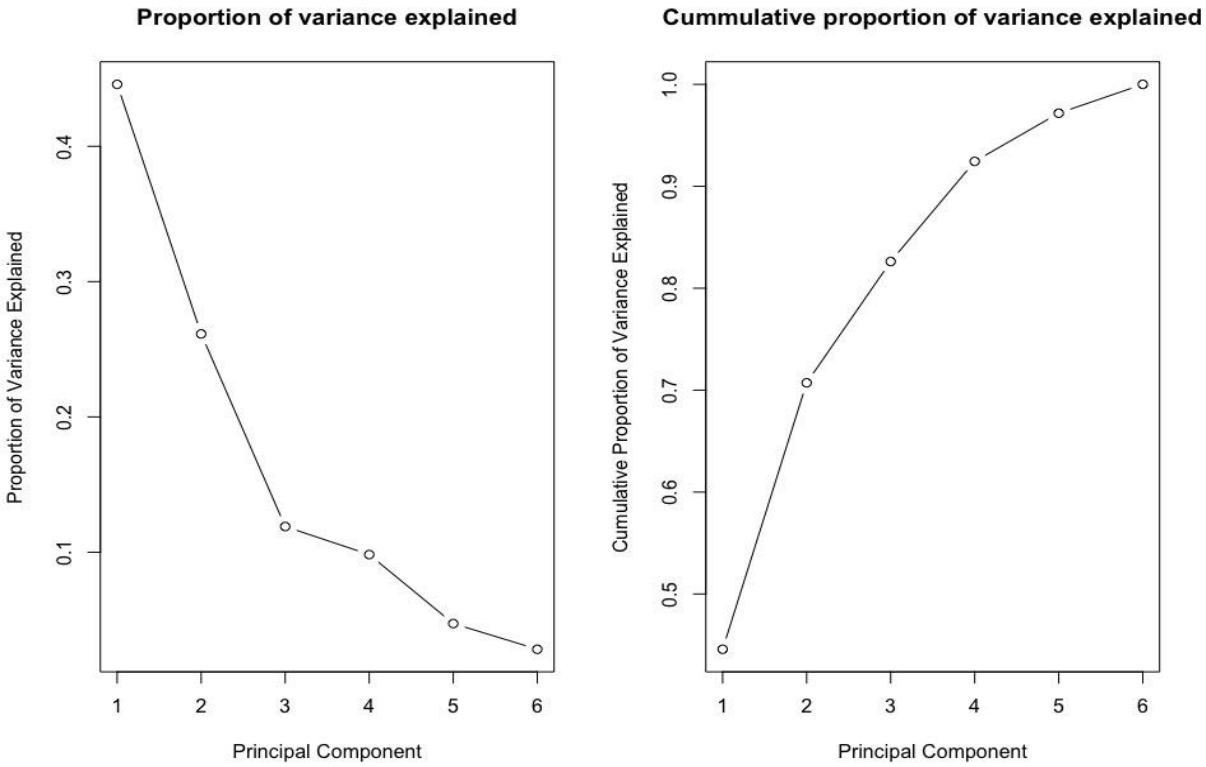



*Figure 3- 2D Biplot*

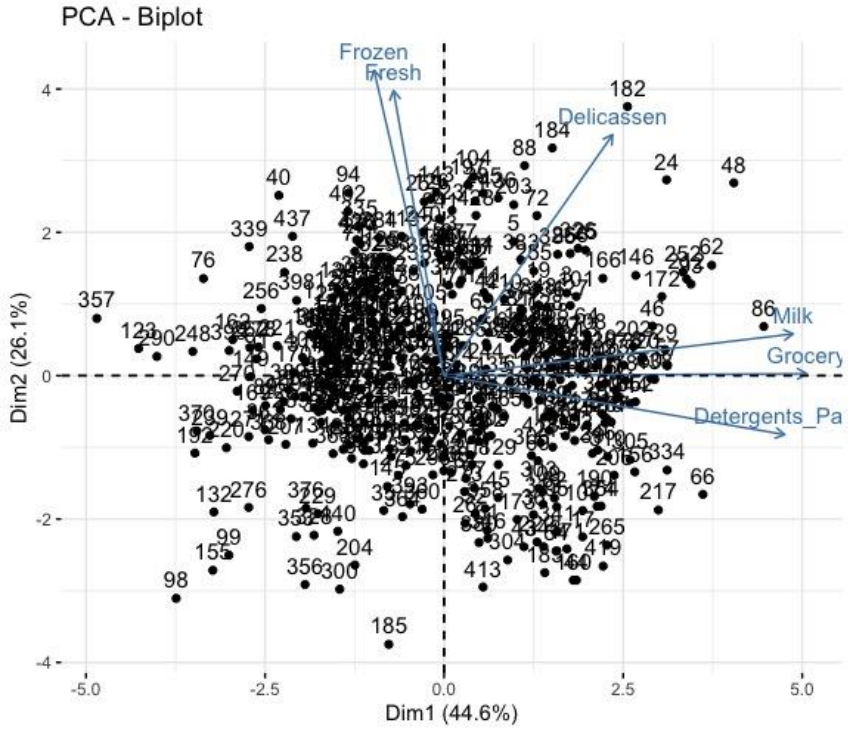


*Figure 4- 3D Biplot*

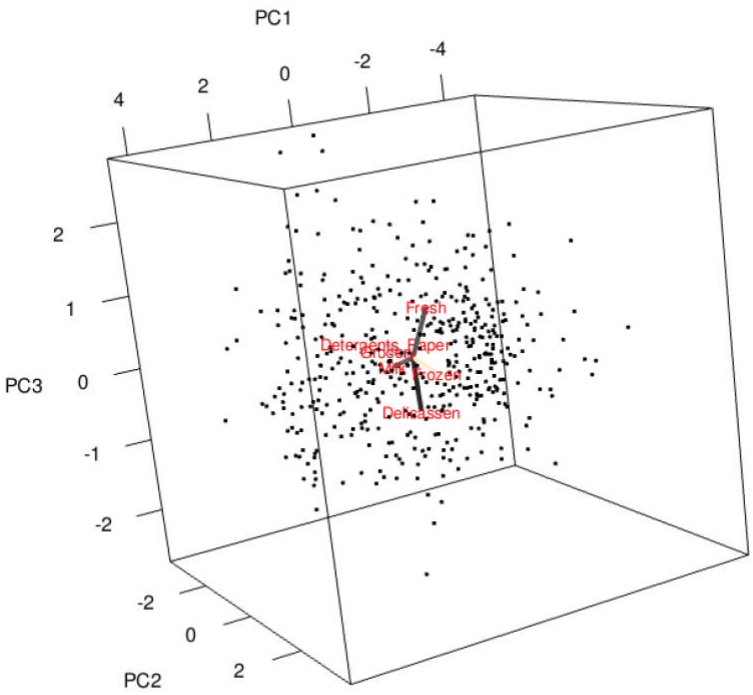



*Table 3- Quality of representation (cos2) for 3 dimensions*

| | PC1 | PC2 | PC3 |
|---|---|---|---|
| Fresh | 0.01694896 | 5.360468e-01 | 0.3708761959 |
| Milk | 0.80422175 | 1.153531e-02 | 0.0009889337 |
| Grocery | 0.86800653 | 2.521232e-05 | 0.0198880290 |
| Frozen | 0.03243036 | 6.148526e-01 | 0.0140142266 |
| Detergents_Paper | 0.76566243 | 2.290741e-02 | 0.0415687848 |
| Delicassen | 0.18754191 | 3.828826e-01 | 0.2666498441 |

*Table 4- Correlation between variables and PCs*

| | PC1 | PC2 | PC3 |
|---|---|---|---|
| Fresh | -0.1301882 | 0.732152165 | 0.60899606 |
| Milk | 0.8967841 | 0.107402562 | -0.03144732 |
| Grocery | 0.9316687 | 0.005021187 | 0.14102492 |
| Frozen | -0.1800843 | 0.784125371 - | 0.11838170 |
| Detergents_Paper | 0.8750214 - | 0.151351931 | 0.20388424 |
| Delicassen | 0.4330611 | 0.618775089 | -0.51638149 |

*Figure 5- Cos2 plot of variables to PCs*

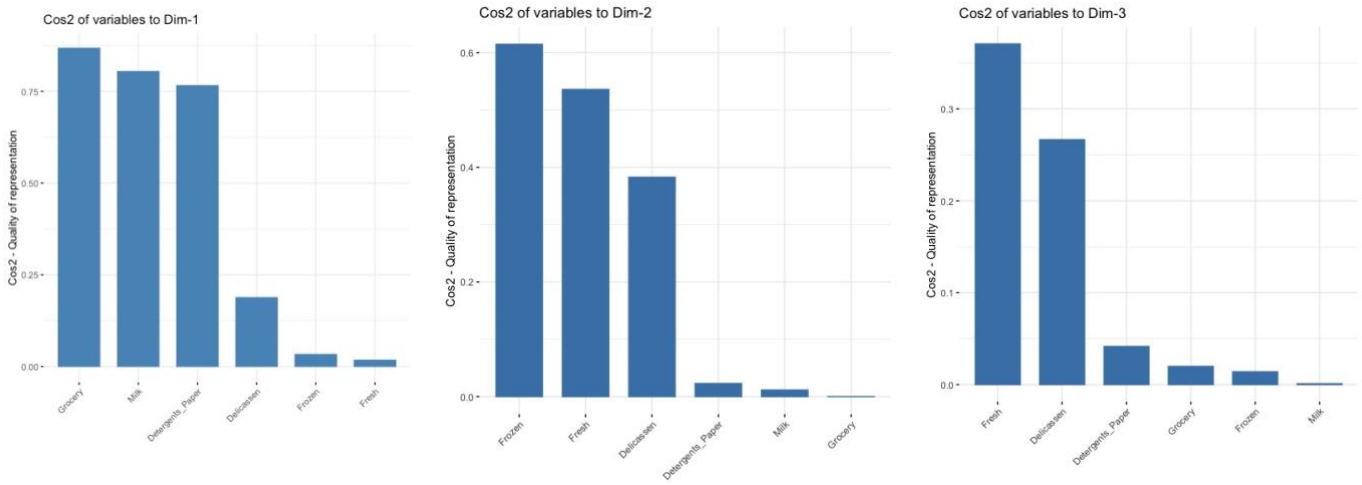



*Figure 6- 2D Correlation plot*

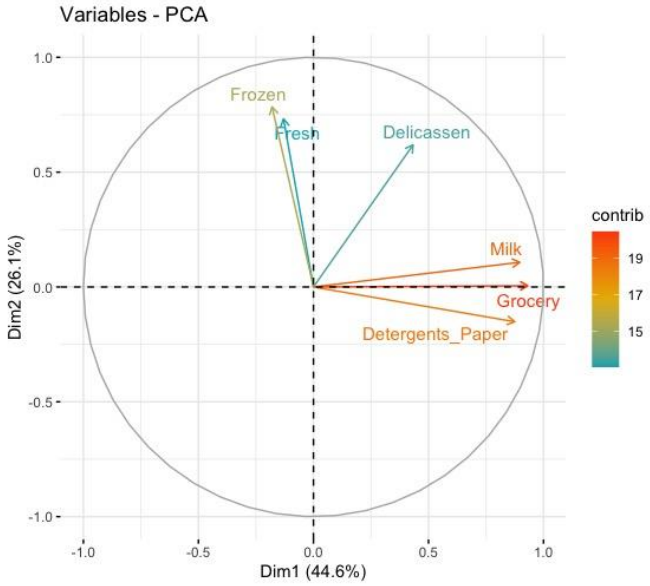
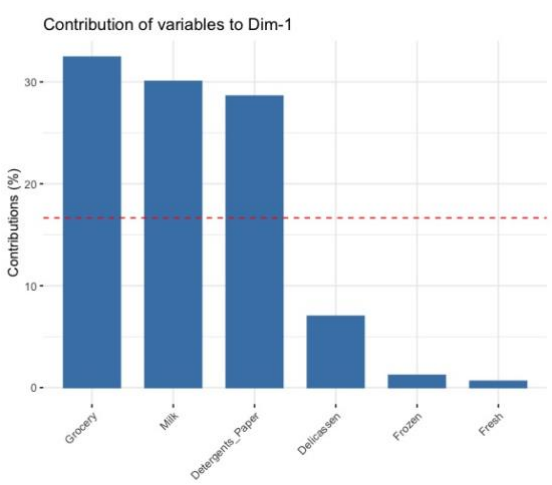
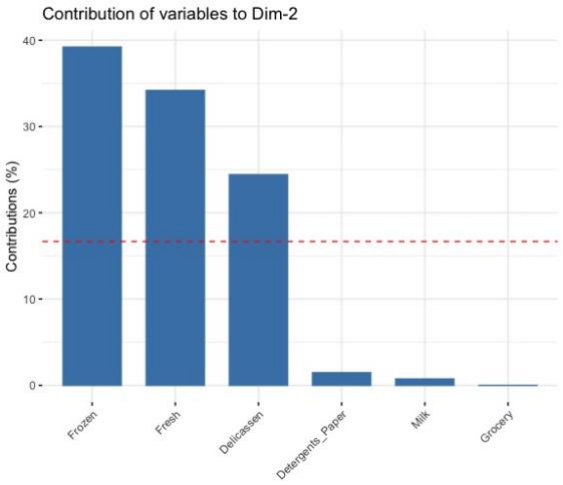
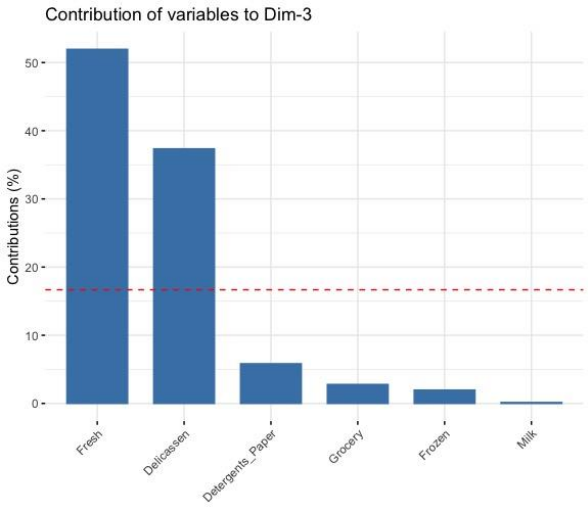


*Figure 7-Contribution of variables to PCs*



# Appendix 3- Clustering

*Figure 1- Best choice of K in K-means*

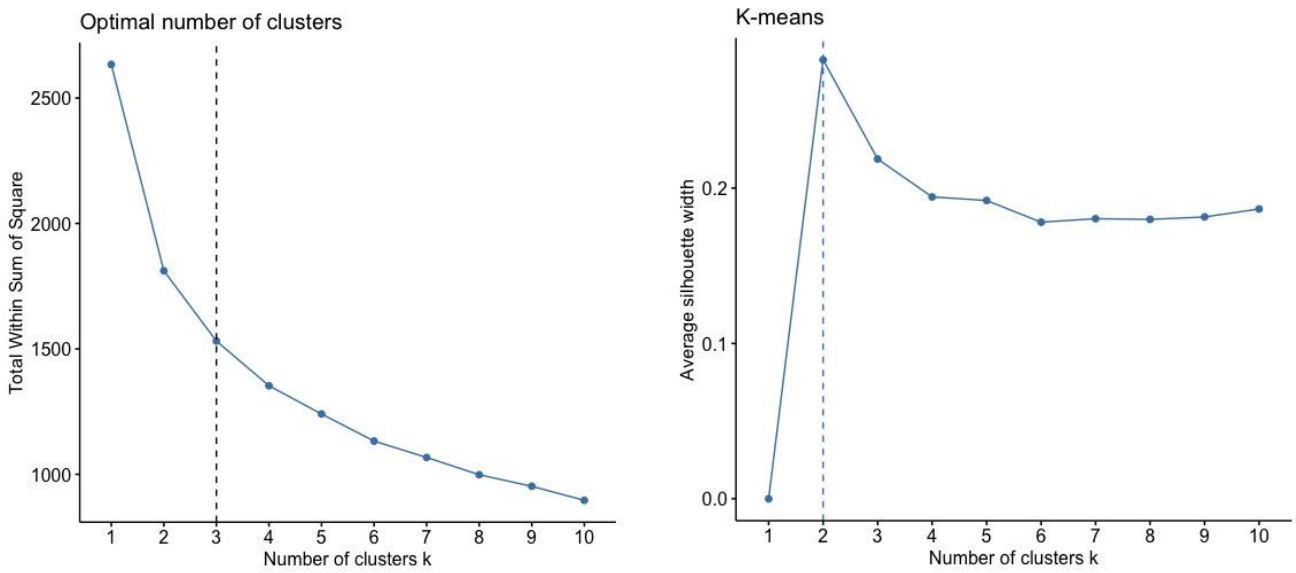



*Figure 2- Plotting 2 clusters output*

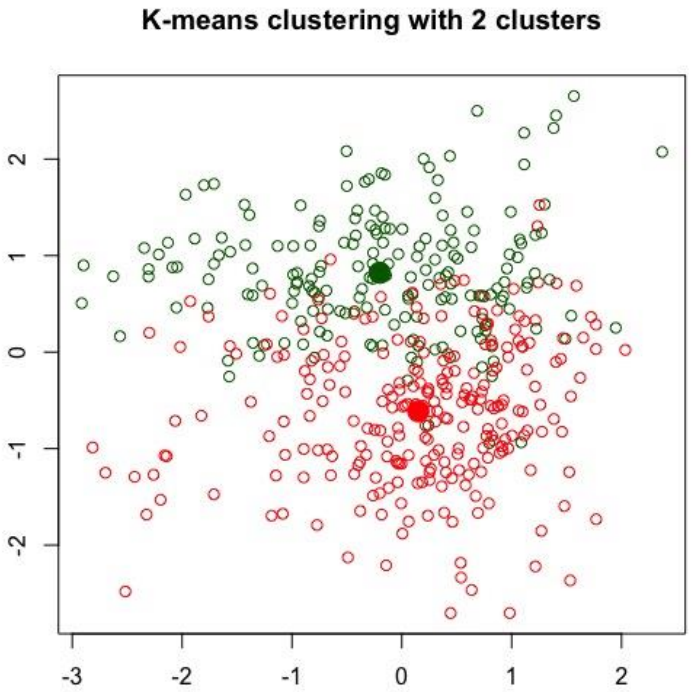


*Figure 3- Plotting 3 clusters output*

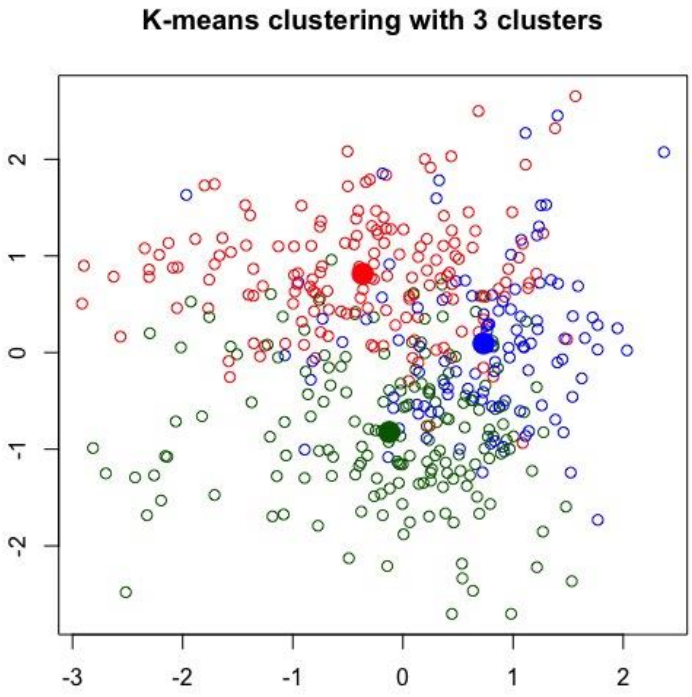



*Figure 4- Visualize 2 clusters on the first 2 principal components*

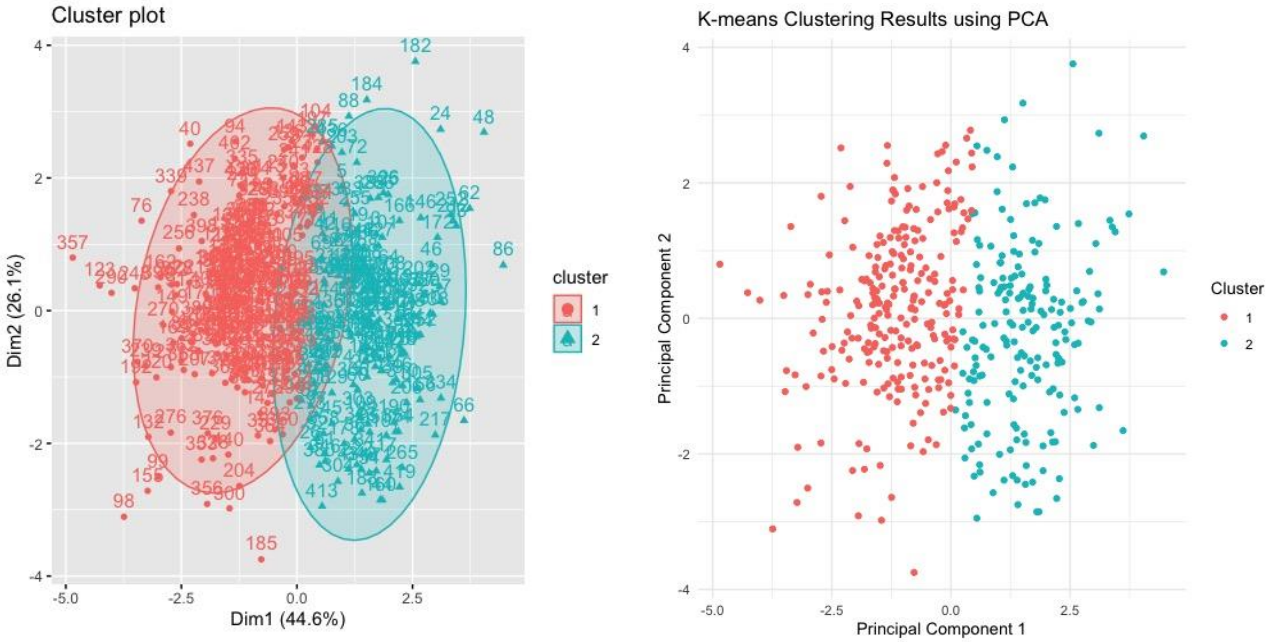


*Figure 5- Visualize 3 clusters on the first 2 principal components*

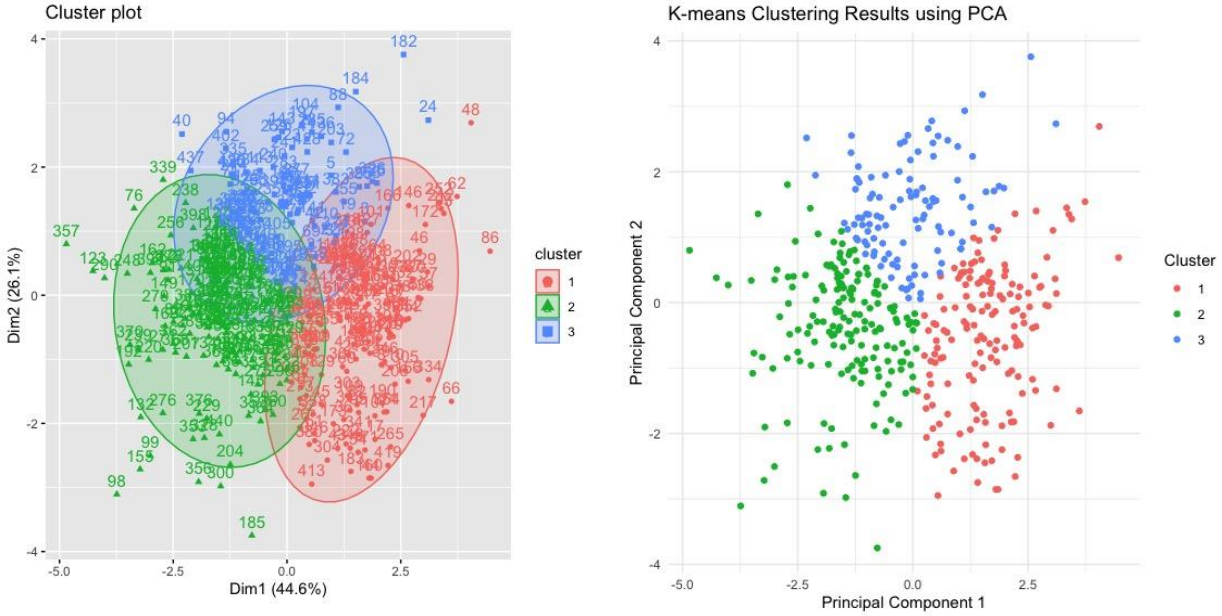



*Figure 6- Average silhouette width of 2 clusters / 3 clusters*

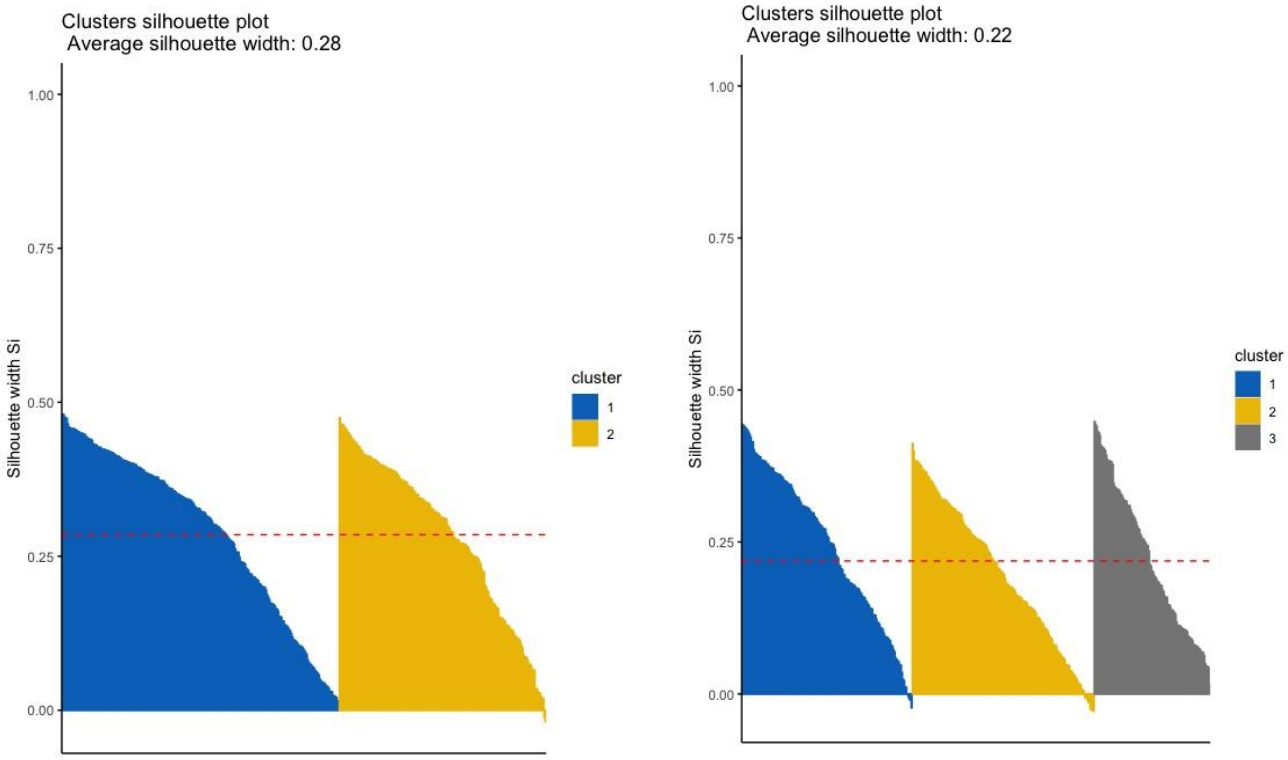


*Figure 7- Best choice of clusters number in Hierarchical clustering*

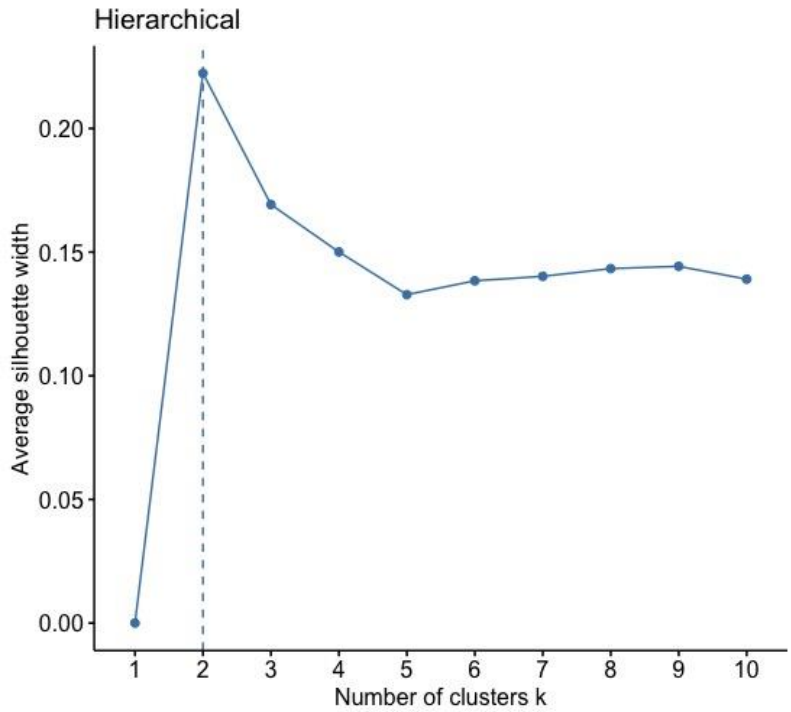



*Figure 8- Dendrogram of 2 clusters (complete linkage)*

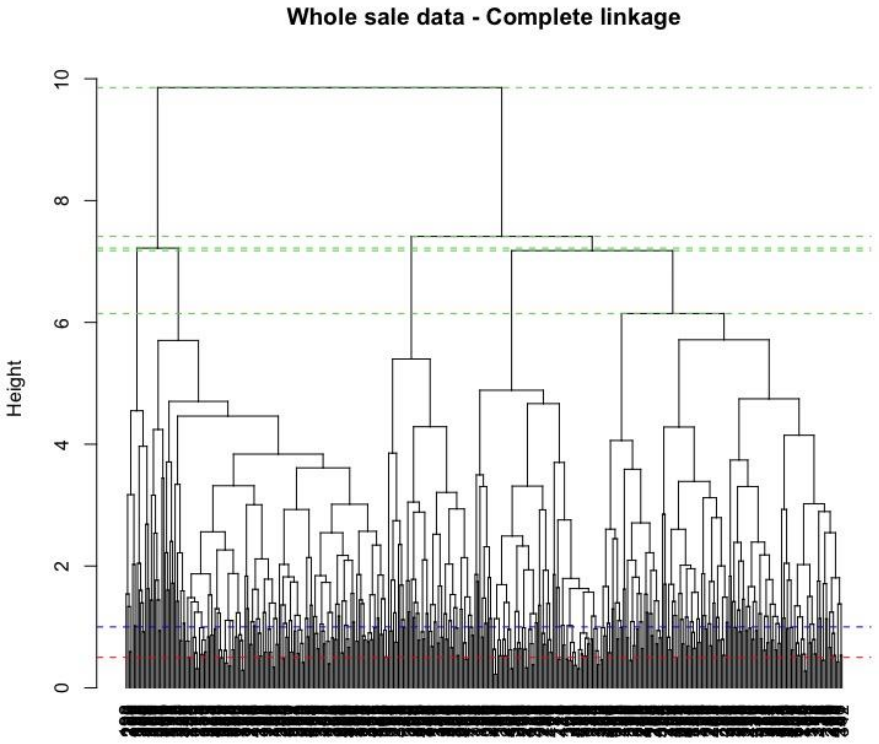


*Figure 9- Visualised hierarchical clustering diagram (2 clusters)*

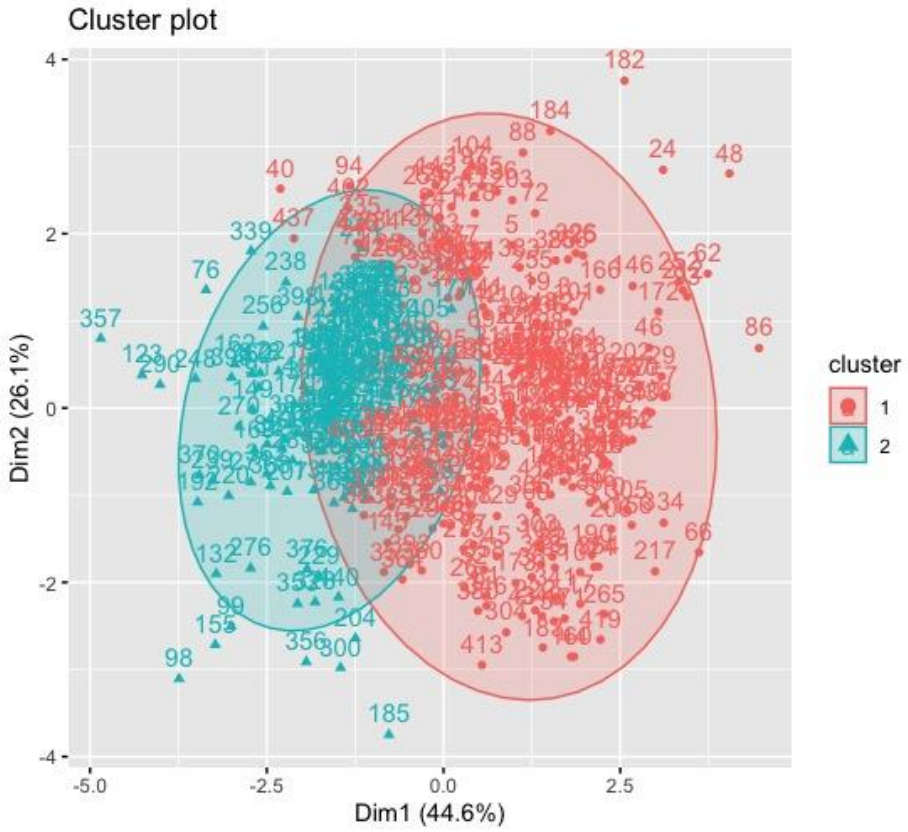



*Figure 10- Improved visualisation of dendrograms (2 clusters)*

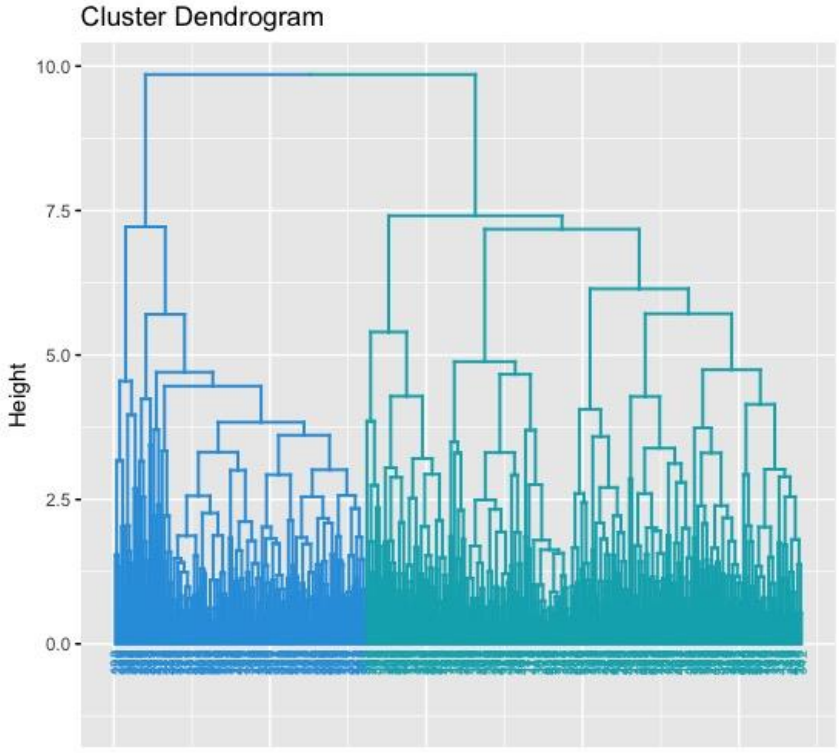
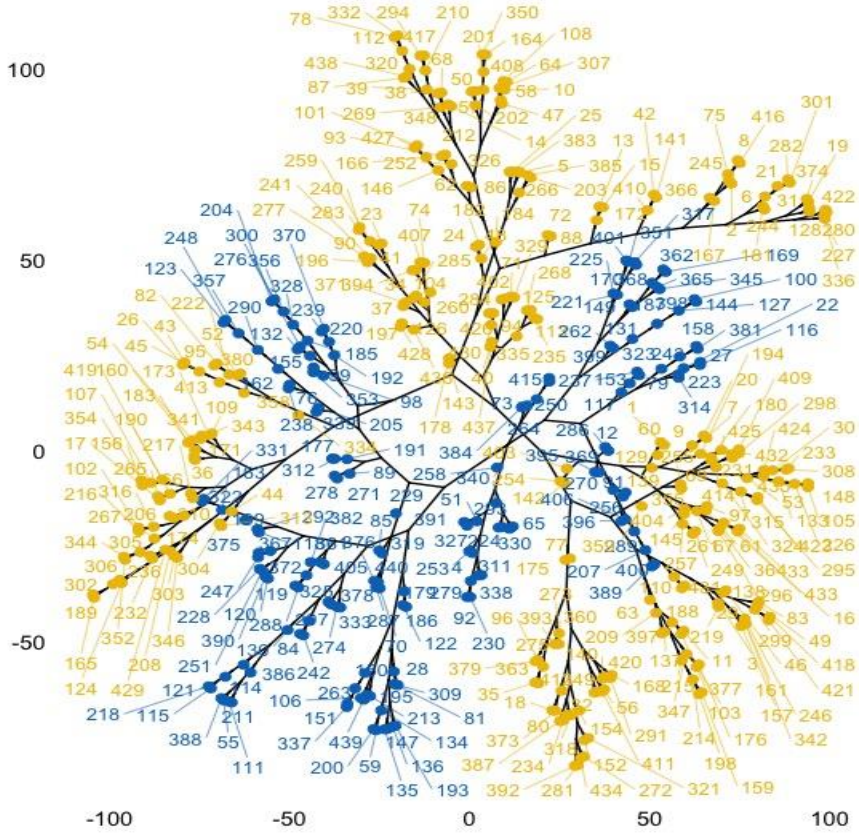


*Figure 11- Further visualisation using phylogenic-like tree (2 clusters)*



# Appendix 4- R code

Also available at this link:

https://github.com/YanLin-Quinne/Master-of-Data-Science/blob/main/DEVUL_Assignment2_Code.R

```
library(factoextra)
library(FactoMineR)
library(tidyverse)
library(ggplot2)
library( cluster)

#loading the data
d1 <- read.csv("~/Desktop/wholesale.csv")
d1
#overview of data
library(skimr)
skim(d1)
names(d1)
str(d1)
#check missing values
colSums(is.na(d1))

#Descriptive statistics
desc_stats <- data.frame(
  Min = apply(d1, 2, min), # minimum
  Q1 = apply(d1, 2, function(x) quantile(x, 0.25)), # 1st quartile (Q1)
  Med = apply(d1, 2, median), # median
  Mean = apply(d1, 2, mean), # mean
  Q3 = apply(d1, 2, function(x) quantile(x, 0.75)), # 3rd quartile (Q3)
  Max = apply(d1, 2, max), # Maximum
  SD = apply(d1, 2, sd) # Standard deviation
)
desc_stats <- round(desc_stats, 2)
head(desc_stats)
#Descriptive statistics is similar to summary function
summary(d1)

#Create a histogram for each column
df_num <- df[sapply(d1, is.numeric)]
par(mfrow = c(ceiling(ncol(df_num)/3), 3))    # Set up a grid of plots
for (i in 1:ncol(df_num)) {
  hist(df_num[,i], main = names(df_num)[i], xlab = "")
}
```

```
dev.off()


# Calculate the correlation matrix
cor_matrix <- cor(d1)
print(cor_matrix)

# Calculate the covariance matrix
cov_matrix <- cov(d1)
print(cov_matrix)

# Visualize relationships between variables using scatterplot matrix
library(car)
scatterplotMatrix(d1, col=4,
                  pch=16, smooth=FALSE, regLine=FALSE,
                  diagonal=list(method ="histogram"),
                  main="Scatterplot Matrix",
                  cex.main=0.8)
library(GGally)
ggpairs(d1)
library(corrplot)
cor_matrix <- cor(d1)
corrplot(cor_matrix, method = "circle",diag=FALSE)
corrplot(cor_matrix, method = "number",diag=FALSE)
library(ggcorrplot)
ggcorrplot(cor_matrix, hc.order = TRUE,type = "upper",
           outline.color = "white", colors = c("#6D9EC1", "white", "#E46726"),
           lab = TRUE, lab_size = 3, lab_col = "black") +
  ggtitle("Correlation Matrix") +
  theme(plot.title = element_text(size = 16, hjust = 0.5),
        axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1))
library(psych)
pairs.panels(d1,method='pearson',
             hist.col='#00AFBB',
             density=TRUE,
             ellipses=TRUE)
#milk, grocery, detergents_paper These three variables are highly correlated

# Boxplots to visualize distribution and detect outliers
boxplot(d1,col=2:7,main='Boxplot of wholesale dataset')

#Find the outliers and exxtremes
find_outliers_extremes <- function(d1, k1 = 1.5, k2 = 3) {
  # initialize an empty list to store results for each column
```

```
  results <- list()
  # loop through each column in the data
  for (i in seq_along(d1)) {
    col <- d1[[i]]
    # calculate quartiles and interquartile range
    q1 <- quantile(col, 0.25, na.rm = TRUE)
    q3 <- quantile(col, 0.75, na.rm = TRUE)
    iqr <- q3 - q1
    # calculate lower and upper bounds for outliers
    lower_bound <- q1 - k1 * iqr
    upper_bound <- q3 + k1 * iqr
    # calculate lower and upper bounds for extremes
    lower_fence <- q1 - k2 * iqr
    upper_fence <- q3 + k2 * iqr
    # identify outliers and extremes
    outliers <- which(col < lower_bound | col > upper_bound)
    extremes <- which(col < lower_fence | col > upper_fence)
    # add results for this column to the list
    results[[i]] <- list(outliers = outliers, extremes = extremes)
  }
  return(results)
}
outliers_extremes <- find_outliers_extremes(d1)
df_with_NA <- d1
for (i in 1:length(outliers_extremes)) {
  df_with_NA[outliers_extremes[[i]]$outliers, i] <- NA
  df_with_NA[outliers_extremes[[i]]$extremes, i] <- NA
}
colSums(is.na(df_with_NA)) #48
library(mice)
imputed_data <- mice(df_with_NA, m = 5, maxit = 50, seed = 1)
d2 <- complete(imputed_data)
colSums(is.na(d2))

#Compare two boxplots
par(mfrow=c(2,1))
boxplot(d1,col=2:7,main='Boxplot of wholesale dataset')
boxplot(d2,col=2:7,main='Boxplot of wholesale cleaned dataset')
dev.off()

#Testing the data set for normal distribution
normality <- data.frame(p_value = numeric(0), test_statistic = numeric(0))
for (i in colnames(d2)) {
  # Perform the Shapiro-Wilk test for normality
```

```
  shapiro_test <- shapiro.test(d2[[i]])
  # Store the p-value and test statistic in the 'normality' data frame
  normality <- rbind(normality, c(shapiro_test$p.value, shapiro_test$statistic))
}
rownames(normality) <- colnames(d2)
colnames(normality) <- c('p-value','test-statistic')
normality
round(normality,2)
#Only Milk and Frozen seem to be normally distributed

num_columns <- 6
plot_rows <- ceiling(sqrt(num_columns))
plot_cols <- ceiling(num_columns / plot_rows)
layout(matrix(1:(plot_rows * plot_cols), nrow = plot_rows, ncol = plot_cols, byrow = TRUE))
for (i in 1:ncol(d2)) {
  # Create a QQ plot for each variable
  qqnorm(d2[, i], main = colnames(d2)[i])
  qqline(d2[, i], col = "red")
}
dev.off()

#Standardised the dataset
apply(d2, 2, sd)
apply(d2, 2, var)
apply(d2, 2, mean) #Needs to be standardised
#They are vastly different, so there is need for scaling.

df.scaled <- scale(d2) #crucial pre-processing step
#Not normally distributed so need to scale

#PCA: The singular value decomposition in the principal component analysisis performed below
#while prcomp() uses singular value decomposition (svd), which can be more stable in some cases.
pr.out<-prcomp(d2,scale=TRUE)
summary(pr.out)

pr.out$center #Means and standard deviations of variables before standardisation.
pr.out$scale #Means and standard deviations of variables after standardisation.

#eigenvalues
pr.out$sdev^2
#PC loadings    vector
pr.out$rotation
#PC score
pr.out$x
```

```
summary(pr.out)
head(pr.out$x) #6 PC score vectors

#The Standard deviation of the principal components is the square root of the corresponding eigenvalues.
sd_all <- summary(pr.out)$importance[1,] #Standard deviation
sd_all^2/sum(sd_all^2) #pr.out$sdev

#scree plot
library(factoextra)
fviz_screeplot(pr.out, addlabels = TRUE)
#The rule of thumb recommends retaining the component that explains 80-90% of the variability in the original data set

pr.out$sdev #Standard deviation of each principal component
pr.var=pr.out$sdev^2
pr.var    #Variance explained by each principal component
pve=pr.var/sum(pr.var)
pve
par(mfrow=c(1,2))
plot(pve,xlab='Principal Component',ylab='Proportion of Variance Explained',type='b',
        main='Proportion of variance explained')
plot(cumsum(pve),xlab='Principal Component',
        ylab='Cumulative Proportion of Variance Explained',type='b',
        main='Cummulative proportion of variance explained')
dev.off()
plot(summary(pr.out)$importance[2,],
        type="b", xlab="PCs", ylab="Variability explained")
#The summary shows that 3 components are enough to keep 80% of the variability in data.
#Also choosing 3 principal components seems reasonable based on the so-called elbow method.

#biplot
fviz_pca_biplot(pr.out,addlabels = TRUE)
library(rgl)
scores <- pr.out$x[, 1:3]
loadings <- pr.out$rotation[, 1:3]
plot3d(scores[,1:3])
#text3d(scores[,1:3],texts=rownames(d2),cex=0.8)
text3d(pr.out$rotation[,1:3],
            texts=rownames(pr.out$rotation), cex=0.8,
            col="red")
coords <- NULL
for (i in 1:nrow(pr.out$rotation)) {
   coords <- rbind(c(0, 0, 0), pr.out$rotation[i, 1:3])
   lines3d(coords, col = sample(colors(), 1), lwd = 4)
```

```
}
normal_vector <- colMeans(pr.out$rotation[, 1:3])
plane_intercept <- -sum(normal_vector * c(0, 0, 0))
planes3d(normal_vector[1], normal_vector[2], normal_vector[3], plane_intercept, alpha = 0.5, col = "blue")

#Correlation between variables and PCs
var <- get_pca_var(pr.out)
var$cor    #Correlation between variables and PC
cor(d2,pr.out$x[,1:3]) #Variable Correlation Chart
fviz_pca_var(pr.out)
fviz_pca_var(pr.out, col.var = "contrib",
                     gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
                     repel = TRUE)

#Cos2 between variables and PCs
var$cos2 #quality of representation (var$cor)^2
cos2 <- (cor(d2, pr.out$x[,1:3]))^2
cos2
fviz_cos2(pr.out,choice='var',axes=1)
fviz_cos2(pr.out,choice='var',axes=2)
fviz_cos2(pr.out,choice='var',axes=3)

##Contribution between variables and PCs
var$contrib
#any variable whose height is up to 16.66 and
#above is considered to have contributed significantly to the component.
fviz_contrib(pr.out, choice = "var", axes = 1, top = 10) #Variable contribution to PC1
fviz_contrib(pr.out, choice = "var", axes = 2, top = 10) #Variable contribution to PC2
fviz_contrib(pr.out, choice = "var", axes = 3, top = 10) #Variable contribution to PC3

#Decomposition using eigenvalues
#Check by evaluating the standard deviation of the variables
sd<-apply(d2,2,sd)
sd
#Variables should be scaled/use correlation matrix
S<-cor(d2)
eigdec<-eigen(S)
eigdec    #The values and vectors from the decomposition are the variances and the principal component loadings
respectively.
eig <- eigdec$values #The value of the decomposition represents the variance
eig
sum(eig)

pc_loading <- eigdec$vectors
```

```
rownames(pc_loading) <- colnames(d2)
pc_loading #Eigenvectors

#The proportion of variance explained is as follows
eig<-eigdec$values
variance<-eig*100/sum(eig)
#Cumulative explained variance
cumvar<-cumsum(variance)
eig2<-data.frame(eig=eig,variance=variance,cumvariance=cumvar)
eig2

#scree-plot
barplot(eig2[,2],names.arg=1:nrow(eig2),main='Scree Plot',
            xlab='Dimensions',ylab='Percentage of variances',col='steelblue')
lines(x = 1:nrow(eig2), eig2[, 2],
          type="b", pch=19, col = "red")

pc_score <- as.matrix(scale(d2))%*% pc_loading
colnames(pc_score) <- paste0("PC", 1:6)
pc_score[1:6,]

library(psych)
pairs.panels(pc_score,
                     method = "pearson", # correlation method
                     hist.col = "#00AFBB",
                     density = TRUE,    # show density plots
                     ellipses = TRUE # show correlation ellipses
)
#The correlation between PCs (new coordinates) is essentially zero, as required
pc_score2 <- as.data.frame(pc_score[,1:3])
rownames(pc_score2) <- seq(1:440)
max_abs_index <- function(x) {
   return(which.max(abs(x)))
}
pc_score2$max_abs_col <- apply(pc_score2, 1, max_abs_index)
color_labels<- c('red','yellow','blue')
pc_score2$color <- color_labels[factor(pc_score2$max_abs_col, levels = 1:length(color_labels))]
pc_score2

cor(d2, pc_score[,1:3])
t(t(pc_loading)*sqrt(eig))

cos2 <- (cor(d2, pc_score[,1:3]))^2
cos2
```

```
comp.cos2 <- apply(cos2, 2, sum)
comp.cos2 # same as the corresponding eigenvalues

contrib2 <- function(cos2, comp.cos2){cos2*100/comp.cos2}
contrib <- t(apply(cos2,1, contrib2, comp.cos2))
contrib

names1 = c("Grocery", "Milk","Detergents_Paper","Delicassen", "Frozen","Fresh")
barplot(contrib[order(contrib[, 1],decreasing = T), 1], names.arg=names1,
        main = "Contribution of variables to PC1",ylim=c(0,35),
        xlab = " ",
        ylab = "Percentage of variances",
        col ="steelblue",las=2,cex.names=0.7)
abline(h=100/6, col="red",lty=3, lwd =1)
#fviz_contrib(pr.out, choice = "var", axes = 1, top = 10)

names2 =c("Frozen","Fresh","Delicassen","Detergents_Paper","Milk","Grocery")
barplot(contrib[order(contrib[, 2],decreasing = T), 2], names.arg=names2,
        main = "Contribution of variables to PC2",
        xlab = " ",ylim=c(0,40),
        ylab = "Percentage of variances",
        col ="steelblue",las=2,cex.names=0.7)
abline(h=100/6, col="red",lty=3, lwd =1)
#fviz_contrib(pr.out, choice = "var", axes = 2, top = 10)

names3 =c("Fresh","Delicassen","Detergents_Paper","Grocery","Frozen","Milk")
barplot(contrib[order(contrib[, 3],decreasing = T), 3], names.arg=names3,
        main = "Contribution of variables to PC3",
        xlab = " ",ylim=c(0,55),
        ylab = "Percentage of variances",
        col ="steelblue",las=2,cex.names=0.7)
abline(h=100/6, col="red",lty=3, lwd =1)
#fviz_contrib(pr.out, choice = "var", axes = 3, top = 10)

#Principal component regression
df.pca <- prcomp(d2[,-3], scale=TRUE)
df.pca$x
Z=df.pca$x[,1:3]
df.lm<-lm(d2$Grocery~Z)
df.lm
df.pca$rotation[,1:3]%*%matrix(coef(df.lm)[-1], ncol=1)
library(pls)
data.pcr<-pcr(Grocery~Delicassen+Milk+Fresh+Frozen+Detergents_Paper,
              3,scale=TRUE,data=d2)
```

```
coef(data.pcr)

#Cluster analysis
#1.k-means clustering
df.scaled <- scale(d2)
#Choose the best k in the k-means algorithm, using the sum of the sums of squares within the clusters
fviz_nbclust(df.scaled, kmeans, method = "wss")+
   geom_vline( xintercept    =    3 , linetype    =    2 )
#total within-cluster sum of square (wss)
fviz_nbclust(df.scaled,kmeans, method = "silhouette")+
   labs(title = "K-means")

#This Figure is treated the same way as we did with the scree plot.
#We locate the bend (knee) in the plot.
#This seems to be at k=2. So we will use two clusters for the analysis.
# Compute k-means with k = 2
set.seed(123)
k2 <- kmeans(df.scaled, 2, nstart = 25)
names(k2)
#Try 25 different random initial cluster assignments and choose the best result corresponding to the one with the least
variation within the clusters
#between_SS / total_SS =    31.2%，Percentage of variance explained by the mean of the clusters
k2$centers #2 centroids under 6 variables (6 dimensions)
k2$iter #1 iteration only
k2$size #252 188
k2$withinss #1021.4575    789.5534
k2$tot.withinss #1811.011
k2$totss #2634
k2$betweenss #822.9891
#Percentage of variance explained by the cluster means = (between_ss/total_ss)*100
# k2$betweenss=k2$totss -k2$tot.withinss
# 31.2 %= (k2$totss-k2$tot.withinss)/k2$totss= k2$betweenss/k2$totss

cols=c('red','darkgreen')
plot(df.scaled,col=cols[k2$cluster],main='K-means clustering with 2 clusters',xlab='',ylab='')
points(k2$centers,pch=19,cex=2,col=cols) #k=2 is not good enough, there is overlap

#Find the mean of 6 variables based on the newly created cluster = k2$centers #2 centroids under 6 variables (6
dimensions)
aggregate(scale(d2), by=list(cluster=k2$cluster), mean)

# Visualise kmeans clustering
fviz_cluster(k2, df.scaled, ellipse.type = "norm")
#It is clear from this result that the clusters are not well separated anymore.
```

```
#In fact, we generated overlapping clusters.

#We can visualise the clustering results using principal components
#if the number of variable are greater than 2.

# Visualize the k-means clustering results using PCA
pca_scores <- as.data.frame(pr.out$x[, 1:2])
pca_scores$cluster <- k2$cluster
pca_cluster_plot <- ggplot(pca_scores, aes(x = PC1, y = PC2, color = as.factor(cluster))) +
  geom_point() +
  labs(title = "K-means Clustering Results using PCA",
       x = "Principal Component 1",
       y = "Principal Component 2",
       color = "Cluster") +
  theme_minimal()
print(pca_cluster_plot)
#Looking at the results of k-means on the first 2 principal components is very intuitive

k3 <- kmeans(df.scaled, 3, nstart = 25)
k3 #between_SS / total_SS =    48.7 %
k3$tot.withinss #1531.347
k3$size
cols=c('red','darkgreen','blue')
plot(df.scaled,col=cols[k3$cluster],main='K-means clustering with 3 clusters',xlab='',ylab='')
points(k3$centers,pch=19,cex=2,col=cols)

fviz_cluster(k3, d2, ellipse.type = "norm")
pca_scores <- as.data.frame(pr.out$x[, 1:2])
pca_scores$cluster <- k3$cluster
pca_cluster_plot <- ggplot(pca_scores, aes(x = PC1, y = PC2, color = as.factor(cluster))) +
  geom_point() +
  labs(title = "K-means Clustering Results using PCA",
       x = "Principal Component 1",
       y = "Principal Component 2",
       color = "Cluster") +
  theme_minimal()
print(pca_cluster_plot)

#2.K-medoids clustering
#The main advantage of K-medoids over K-means clustering is that it is less sensitive to outliers
#The most common method for clustering K-medoids is the PAM algorithm (Partitioning Around Medoids)
pam.res <- pam(df.scaled, k=2)
names(pam.res)
summary(pam.res)
```

```
pam.res$medoids # Medoids
pam.res$id.med #245 247
df.scaled[c(245,247),] #==pam.res$medoids Find the central point
pam.res$clustering    # Cluster vectors
#Visualization Cluster
fviz_cluster(pam.res, d2, ellipse.type = "norm")
clusplot( pam.res , main    =    " Cluster plot with K-medoids (pam), K = 2 " , color    =    TRUE )
#Silhouette
pam.res$silinfo
pam.res$silinfo$clus.avg.widths #Average silhouette width
pam.res$silinfo$avg.width #Overall average silhouette
pam.res$clusinfo #cluster1:193, cluster2:247
fviz_silhouette(silhouette(pam.res))
fviz_silhouette(silhouette(pam.res$clustering,dist(df.scaled)),palette='jco',ggtheme=theme_classic())
#there are a small number of si having a negative sign in cluster 1&2
# Compute silhouette
sil <- silhouette(pam.res)[, 1:3]
neg_sil_index <- which(sil[, 'sil_width'] < 0)
sil[neg_sil_index, , drop = FALSE]

#CLARA: Clustering Large Applications
clara.res <- clara(df.scaled, k=2)
names(clara.res)
clara.res
summary(clara.res)
clara.res$medoids #Centroids under 6 variables in 6 dimensions
clara.res$i.med    #i.med gives the IDs of the 2 observations used as centres (remember we are not using centroids
here)
#75 375
df.scaled[c(75,375),] #==clara.res$medoids
clara.res$clustering
fviz_cluster(clara.res,d2,ellipse.type = 'norm')
clusplot( clara.res ,
                main    =    " Cluster plot with K-medoids (clara), K = 2 " ,
                color    =    TRUE )
clara.res$silinfo
clara.res$silinfo$clus.avg.widths #Average silhouette width
clara.res$silinfo$avg.width #Overall average silhouette
clara.res$clusinfo    #cluster1:214,cluster2:226
##pam is better than clara, k-medoids use pam later

# Visualising k-means and k-medoids clustering results using PCA
# Adding k-means and k-medoids clustering assignments to PCA scores
pca_scores <- as.data.frame(pr.out$x[, 1:2])
```

```
pca_scores$kmeans_cluster <- k2$cluster
pca_scores$kmedoids_cluster <- pam.res$clustering
pca_cluster_plot <- ggplot() +
  geom_point(data = pca_scores, aes(x = PC1, y = PC2, color = as.factor(kmeans_cluster)), alpha = 0.5) +
  geom_point(data = pca_scores, aes(x = PC1, y = PC2, color = as.factor(kmedoids_cluster)), shape = 1, alpha = 0.5) +
  labs(title = "K-means and K-medoids Clustering Results using PCA",
       x = "Principal Component 1",
       y = "Principal Component 2",
       color = "Cluster") +
  theme_minimal()
print(pca_cluster_plot)

#Reference workshop3
par(mfrow=c(2,1))
kmeans_centers <- k2$centers
kmedoids_centers <- pam.res$medoids

plot(df.scaled, col = cols[k2$cluster], main = 'K-means and K-medoids clustering with 2 clusters')
points(kmeans_centers, pch = 19, cex = 2, col = cols)
legend("topright", legend = c("K-means centers"), col = c("black"), pch =19,
       pt.bg =NA, bty = "n")

plot(df.scaled, col = cols[pam.res$clustering], main = 'K-means and K-medoids clustering with 2 clusters')
points(kmedoids_centers, pch = 23, cex = 2, col = cols, bg = 'white')
legend("topright", legend = c("K-medoids centers"), col = c("black"),
       pch = 23, pt.bg = c("white"), bty = "n")
dev.off()

library(cluster)
# Compute the Silhouette Score
kmeans_silhouette_score <- mean(silhouette(k2$cluster, dist(df.scaled))[, "sil_width"])
kmeans_silhouette_score    #0.2849978
kmeans3_silhouette_score <- mean(silhouette(k3$cluster, dist(df.scaled))[, "sil_width"])
kmeans3_silhouette_score    #0.2187303
kmedoids_silhouette_score <- mean(silhouette(pam.res$clustering, dist(df.scaled))[, "sil_width"])
kmedoids_silhouette_score #0.2782588
#The Silhouette Score ranges from -1 to 1. The closer the value to 1, the better the clustering.
#These two scores indicate that the quality of the clustering is fair, but there may be room for improvement.
#mean(silhouette(clara.res$clustering, dist(df.scaled))[, "sil_width"])
#[1] 0.2741116

#Evaluation of k-means and k-medoids: to use contour silhouette analysis
#compare:K-means
sil_kmeans<-silhouette(k2$cluster,dist(df.scaled))
```

```
head(sil_kmeans[,1:3],4)
k2$cluster[1:4]
sil_kmeans.sum<-summary(sil_kmeans)
sil_kmeans.sum$clus.avg.widths #Average silhouette width
sil_kmeans.sum$avg.width #Overall average silhouette
sil_kmeans.sum$clus.sizes ##Size of each cluster
fviz_silhouette(sil_kmeans,palette='jco',ggtheme=theme_classic()) #k-means
#there are a small number of si having a negative sign in cluster 2

sil_kmeans<-silhouette(k2$cluster,dist(df.scaled))
head(sil_kmeans[,1:3],4)
k2$cluster[1:4]
sil_kmeans.sum<-summary(sil_kmeans)
sil_kmeans.sum$clus.avg.widths #Average silhouette width
sil_kmeans.sum$avg.width #Overall average silhouette
sil_kmeans.sum$clus.sizes ##Size of each cluster
fviz_silhouette(sil_kmeans,palette='jco',ggtheme=theme_classic()) #k-means
#there are a small number of si having a negative sign in cluster 2

#compare:3-clustering
sil_kmeans3<-silhouette(k3$cluster,dist(df.scaled))
head(sil_kmeans3[,1:3],4)
k3$cluster[1:4]
sil_kmeans.sum<-summary(sil_kmeans3)
sil_kmeans.sum$clus.avg.widths #Average silhouette width
sil_kmeans.sum$avg.width #Overall average silhouette
sil_kmeans.sum$clus.sizes #Size of each cluster
fviz_silhouette(sil_kmeans3,palette='jco',ggtheme=theme_classic())
fviz_nbclust(df.scaled,kmeans, method = "silhouette")+
   labs(title = "K-means")

#compare:K-medoids
sil_kmedoids <- silhouette(pam.res$cluster,dist(df.scaled))
summary(sil_kmedoids)
summary(sil_kmedoids)$clus.avg.widths
summary(sil_kmedoids)$avg.width
summary(sil_kmedoids)$clus.sizes

fviz_nbclust(df.scaled, pam, method = "silhouette")+
   labs(title = "K-medoids with pam")
fviz_silhouette(sil_kmedoids,palette='jco',ggtheme=theme_classic())
#Both cluster1 and cluster2 have some negative signs
```

```
#3.Hierarchical clustering
## Find the best number of clusters using the elbow method
fviz_nbclust(df.scaled, FUNcluster = function(x, k) { list(cluster = cutree(hclust(dist(x)), k = k)) },
                    method = "silhouette") +labs(title = "Hierarchical")
#fviz_nbclust(df.scaled, FUNcluster = hcut, method = "silhouette")+ labs(title = "Hierarchical")
#k_optimal=2

# Compute distances and hierarchical clustering
dd <- dist(df.scaled, method = "euclidean")
hc <- hclust(dd, method = "complete")
hc
hcut<-cutree(hc,k=2)
hcut
table(hcut) #cluster1:279,cluster2:161
rownames(df.scaled)[hcut == 1]

#We can visualise the object by using a dendrogram.
#This will enable us to determine the point to cut the tree,
#resulting in the number of clusters.
dendros <- as.dendrogram(hc)
plot(dendros, main = "Whole sale data - Complete linkage",
        ylab = "Height")
abline(h=0.5, lty = 2, col="red")
abline(h=1, lty = 2, col="blue")
Hs <- hc$height[(length(hc$height)-4):length(hc$height)]
abline(h=Hs, col=3, lty=2)

fviz_cluster(list(data=df.scaled, cluster=cutree(hc, 2)), ellipse.type = "norm")
fviz_dend(hc, k = 2, # Cut in two groups
                cex = 0.5, # label size
                k_colors = c("#2E9FDF", "#00AFBB"),
                color_labels_by_k = TRUE, # color labels by groups
                ggtheme = theme_gray() # Change theme
)

fviz_cluster(list(data=df.scaled, cluster=cutree(hc, 3)), ellipse.type = "norm")
fviz_dend(hc, k = 3, # Cut in two groups
                cex = 0.5, # label size
                k_colors = c("#2E9FDF", "#00AFBB",'red'),
                color_labels_by_k = TRUE, # color labels by groups
                ggtheme = theme_gray() # Change theme
)
hcut <- cutree(hc, k = 2)
```

```
table(hcut) #Check the number of observations in each of the two clusters
aggregate(d2, by=list(cluster=hcut), mean)
#Can be used to calculate the mean of each variable by clusters using raw data

hc_silhouette_score <- mean(silhouette(hcut, dist(df.scaled))[, "sil_width"])
hc_silhouette_score #0.2223309

m <- c("average","single","complete")
names(m) <- c("average","single","complete")
# function to compute coefficient
ac <- function(x){
   agnes(df.scaled, method = x)$ac
}
map_dbl(m,ac) #complete: 0.9037892

library(igraph)
fviz_dend(hc, k = 2, k_colors = "jco", type = "phylogenic", repel = TRUE)
fviz_dend(hc, k = 3, k_colors = "jco", type = "phylogenic", repel = TRUE)

#4.Hierarchical K-means Clustering
#First calculate the hierarchical clusters and cut the tree into k clusters, then calculate the centre (i.e. the mean) of each
cluster, this is used as the initial centroid for k-means
res.hk2 <- hkmeans(df.scaled, k=2, hc.metric = "euclidean", hc.method ="complete")
res.hk2    #31.2 %
res.hk3 <- hkmeans(df.scaled, k=3, hc.metric = "euclidean", hc.method ="complete")
res.hk3 #41.7 %
res.hk4 <- hkmeans(df.scaled, k=4, hc.metric = "euclidean", hc.method ="complete")
res.hk4 #47.8 %

k2 #31.2 %
k3 #41.9 %
k4 #48.7%
#Basically the same as k-means, using hierarchical + k-means clustering
#without gaining any advantage

#5.Model-based hierarchical clustering: finding the maxbic
library(mclust)
mc2 <- Mclust(df.scaled, G = 2) # Model-based-clustering, G=2 indicates a request for 2 clusters
summary(mc2)
names(mc2)

mc3 <- Mclust(df.scaled, G = 3) # Model-based-clustering, G=3 indicates a request for 3 clusters
summary(mc3)
```

```
mc2$classification #Newly created clusters
plot(mc2, what = "classification")

?mclustModelNames

mc2_option_VVV <- Mclust(df.scaled, G = 2, modelNames = "VVV")
mc2_option_VVI <- Mclust(df.scaled, G = 2, modelNames = "VVI")
mc2_option_EVE <- Mclust(df.scaled, G = 2, modelNames = "EVE")
mc2_option_VEI <- Mclust(df.scaled, G = 2, modelNames = "VEI")

mc2_bic <- Mclust(df.scaled, G = 2)
mc2_bic$BIC #The first 3 model options based on the BIC standard
# VVE,2          VVV,2          EVE,2
#-6359.651 -6407.469 -6428.712

mc3_bic <- Mclust(df.scaled, G = 3)
mc3_bic$BIC
#VVE,3          EVE,3          VVV,3
#-6318.281 -6420.166 -6450.705

mc4_bic <- Mclust(df.scaled, G = 4)
mc4_bic$BIC
#EVE,4          VVE,4          VEE,4
#-6392.666 -6393.849 -6457.152
G <- c(2,3,4,5,6)
modelNames <- c("EII", "VII", "EEI", "VEI", "EVI", "VVI", "EEE",
                          'VEE','EVE','VVE','EVV',
                          "EEV", "VEV", "VVV")
nnr <- length(G)*length(modelNames)
resu    <- array(data=NA, dim=c(nnr,3), dimnames=list(paste(1:nnr),c("G", "modelNames", "BIC")))
counter <- 1
for (i in G){
   for(j in modelNames){
      mc_option <- Mclust(df.scaled, G = i, modelNames = j)
      resu[counter, 1] <- as.numeric(i)
      resu[counter, 2] <- paste(j)
      resu[counter, 3] <- as.numeric(mc_option$BIC)
      if (counter < nrow(resu)) counter <- counter+1
   }
}
G <- as.numeric(resu[,1])
bic <- as.numeric(resu[,3])
model <- resu[,2]
dat <- data.frame(G, bic, model)
```

```
#Give the model combination with the maximum BIC and use the chosen combination to fit the final model
aa <- subset(dat, bic == max(bic))
aa
#G             bic model
#24 3 -6318.281      VVE
mcdf.scaled <- Mclust(df.scaled, G = 3, modelNames = "VVE") # Model-based-clustering
summary(mcdf.scaled )
#Clustering table:
#1      2      3
#110 174 156
plot(mcdf.scaled, what = "classification")
library(factoextra)
fviz_cluster(mcdf.scaled, data = df.scaled,
                    palette = c("#2E9FDF", "#00AFBB","#E7B800"),
                    ellipse.type = "euclid", # Concentration ellipse
                    star.plot = TRUE, # Add segments from centroids to items
                    repel = TRUE, # Avoid label overplotting (slow)
                    ggtheme = theme_minimal()
)
mc_bic2 <- Mclust(df.scaled, G = 2)
mc_bic2$BIC
max(mc_bic2$BIC) #-6359.651

mc_bic3 <- Mclust(df.scaled, G = 3)
mc_bic3$BIC
max(mc_bic3$BIC) #-6318.281

mc_bic4 <- Mclust(df.scaled, G = 4)
mc_bic4$BIC
max(mc_bic4$BIC) #-6392.666

mc_bic5 <- Mclust(df.scaled, G = 5)
mc_bic5$BIC
max(mc_bic5$BIC) #-6404.392

mc_bic6 <- Mclust(df.scaled, G = 6)
mc_bic6$BIC
max(mc_bic6$BIC) #-6411.598

#In summary, the maximum bic value is VVE,3:-6318.281
#so we will use G=3 as the optimal number of clusters
#This is the same result we obtained in the for loop above
```