PRINCIPAL COMPONENT ANALYSIS:

Principal component analysis (PCA) is a technique used for identification of a smaller number of uncorrelated variables known as principal components from a larger set of data. The technique is widely used to emphasize variation and capture strong patterns in a data set. Principal component analysis is focused on the maximum variance amount with the fewest number of principal components. One makes use of principal component analysis to eliminate the number of variables or when there are too many predictors compared to number of observations or to avoid multicollinearity.

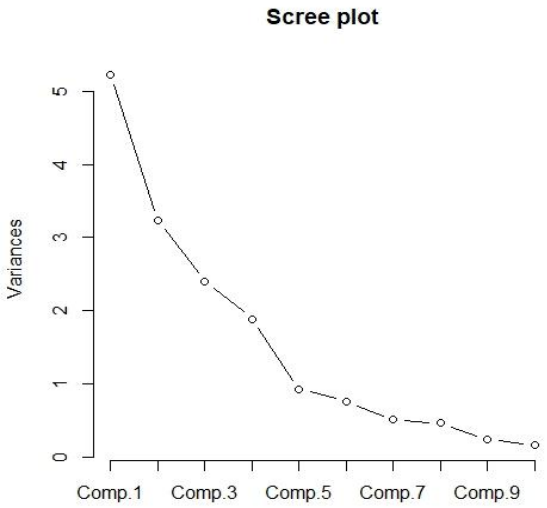
In this study, PCA is done using 'princomp' function in R. In the given dataset the variables are having varying units of measurements and variable scales or ranges are also not similar. Hence, we use correlation matrix to find the principal components and we do this by setting the argument value of 'cor' as 'TRUE'.

OUTPUT:

```
> summary(pc)
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4     Comp.5     Comp.6
Standard deviation     2.2875074 1.7991754 1.5497746 1.3737351 0.96084809 0.87044760
Proportion of Variance 0.3270431 0.2023145 0.1501126 0.1179468 0.05770182 0.04735494
Cumulative Proportion  0.3270431 0.5293576 0.6794702 0.7974170 0.85511879 0.90247373
```

```
    Comp.7     Comp.8     Comp.9     Comp.10     Comp.11     Comp.12     Comp.13
0.71151396 0.67869926 0.48938958 0.397769036 0.324432396 0.23211134 0.151091002
0.03164076 0.02878954 0.01496888 0.009888763 0.006578524 0.00336723 0.001426781
0.93411449 0.96290403 0.97787292 0.987761681 0.994340205 0.99770743 0.999134215
```

```
                          Comp.14      Comp.15      Comp.16
Standard deviation     0.1017965681 0.0553363922 2.068591e-02
Proportion of Variance 0.0006476588 0.0001913823 2.674417e-05
Cumulative Proportion  0.9997818736 0.9999732558 1.000000e+00
```



CONCLUSION OF PCA:

Large dimensionality of data may be informative but not necessarily effective. Often, presence of too many features leads to data memorization, inhibiting data generalization by enhancing complexities and hence creating hindrance for fruitful data analysis. The curse of dimensionality is resolved through feature extraction and feature selection. PCA is one of such dimensionality reducing algorithms.

Clearly, we can see that 1st principal component explains 32.70% of total variability. Also, first 6 principal components jointly explain 90.25% of total variation and for first 10 PCs, the total explained variation is 98.77%. Also, from the scree plot we can observe that the elbow shape formation takes place in case of 5th and 6th principal components.

Hence, the present study of principal component analysis accumulates enough evidence in favour of reducing dimension of the dataset to a considerable extent, as deduced from the outputs.

CLUSTER ANALYSIS:

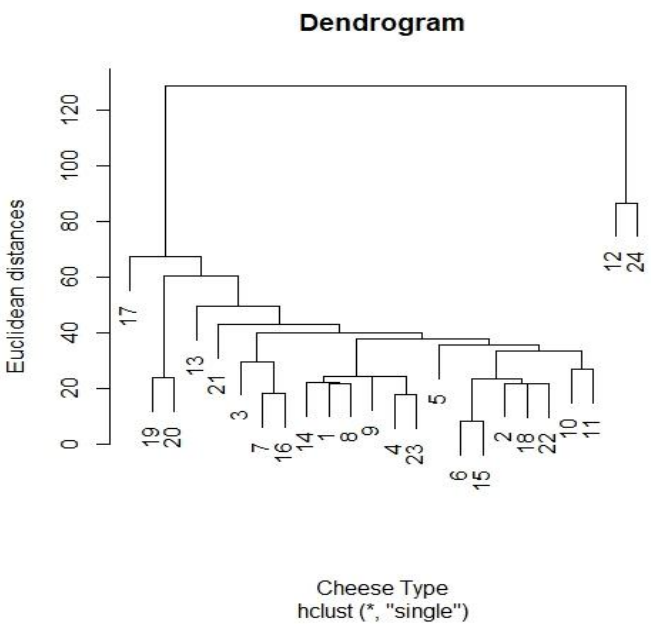
Cluster analysis is a statistical classification technique in which a set of objects or points with similar characteristics are grouped together in clusters. It encompasses a number of different algorithms and methods that are all used for grouping objects of similar kinds into respective categories. The aim of cluster analysis is to organize observed data into meaningful structures in order to gain further insight from them. cluster analysis is only used to discover the structures found in data without explaining why those structures or relationships exist.

Clustering is referred to as an ***unsupervised learning method*** because no information is provided about the "right answer" for any of the objects. It can uncover previously undetected relationships in a complex data set. Two types of clustering algorithms are ***nonhierarchical*** and ***hierarchical***. In nonhierarchical clustering, such as the **k-*means*** algorithm, the relationship between clusters is undetermined. With both of these approaches, an important issue is how to determine the similarity between two objects, so that clusters can be formed from objects with a high similarity to each other. Commonly, ***distance functions***, such as the ***Manhattan*** and ***Euclidian*** distance functions, are used to determine similarity.

Regarding this data after deleting the texture column, we want to check what type of clusters the dataset can create and then we compare the clustering results with the given texture types.

❖ HIERARCHICAL CLUSTERING ALGORITHM (HCA):

Here we create the distance matrix using Euclidean distance metric and then single linkage Agglomerative Hierarchical Clustering method is used to create a dendrogram which serves the purpose of finding optimal number of clusters visually as well as creates a nested set up partitions of the given dataset.



Clearly, from the above dendrogram for any merger level more than or equal to 90 and below 120, the given dataset can be partitioned into two clusters, that is optimal number of clusters, as interpreted from the dendrogram, is 2. Also, as in our dataset we have cheese textures of 5 types, hence we also partition the dataset into 5 clusters and compare the results.

OUTPUT AND CONCLUSION:

```
> print(y_1)
 [1] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2
> print(y_2)
 [1] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 3 1 4 4 1 1 1 5
```

In case of partitioning the dataset into two clusters, as directed by the dendrogram, we can see only 12^{th} and 24^{th} data points belong to the 2^{nd} cluster whereas rest of the data points belong to cluster 1.

Also, if we partition the dataset into 5 clusters then 12^{th}, 17^{th} and 24^{th} data point creates their own clusters– cluster 2, cluster 3 and cluster 5 respectively, whereas the 19^{th} and 20^{th} data point jointly create a 4^{th} cluster and remaining 19 data points fall in cluster 1. Hence, while comparing with the original texture types, we get huge dissimilarity between the actual clusters and clusters created using Hierarchical clustering technique. This happens mainly due to insufficiency of the data restraining the algorithm to perform robustly.

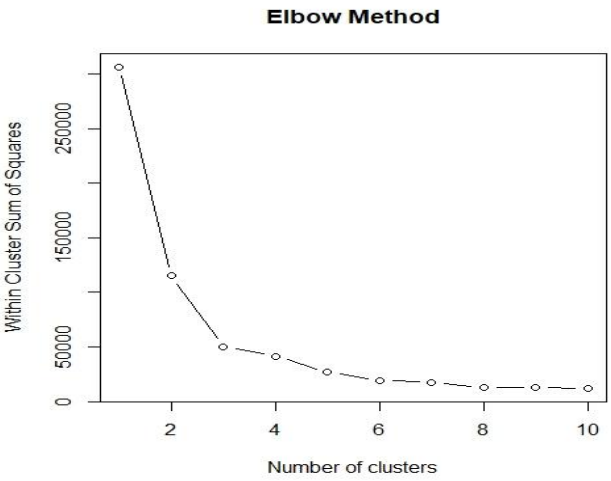
❖ K-MEANS CLUSTERING:

Hierarchical clustering did not yield a meaningful output. Hence, we proceed to do clustering using a non-hierarchical algorithm namely k-means clustering.

In order to find the optimal number of clusters we use the elbow method where within cluster sum of squares is plotted against number of clusters.

OUTPUT AND CONCLUSION:

As indicated from the above plot, we can observe within cluster sum of squares is not decreasing significantly when we use more than 4 clusters. Hence, optimal number of clusters can be taken as 4. Also, we have done the clustering setting 5 initial random clusters. In both cases, the maximum number of iterations is set to be 1000. Also, as we know k-mean clustering output largely depends on the initial random clusters. Hence, we set nstart = 25 so that 25 initial configurations of clusters can be attempted and best of those configurations is reported.



Using 4 initial clusters we get the output as

```
> print(y_1)
 [1] 4 1 2 4 1 1 2 4 4 1 2 3 4 4 1 2 2
1 4 4 4 1 4 3
> print(kmeans1$size)
[1]  7  5  2 10
> print(kmeans1$tot.withinss)
[1] 31838.25
```

Also, using 5 initial clusters we get,

```
> print(y_2)
 [1] 1 2 3 1 1 2 3 1 1 2 2 5 1 1 2 3 3
2 4 4 4 2 1 5
> print(kmeans2$size)
[1] 8 7 4 3 2
> print(kmeans2$tot.withinss)
[1] 23522.5
```

We can observe that in case of 5 initial random clusters, the within cluster sum of squares is significantly less than the case where 4 initial clusters are used. If we increase the number of initial random cluster to 6, then within SS comes down to around 18000.

Also, if we comparing with original clusters we get,

Texture Type	Type 1	Type 2	Type 3	Type 4	Type 5
Number of datapoints in Original Data	9	9	3	2	1
Number of data points after clustering	8	7	4	3	2

Hence, we can see that in this case non-hierarchical clustering algorithm gives far better result than single linkage Agglomerative hierarchical clustering algorithm. But this k – mean clustering algorithm output itself has a lot of discrepancies. For example, 2nd and 3rd data point in the dataset forms 2nd and 3rd clusters whereas they are together in original dataset. But again reason of this type of discrepancies can be labelled to the small dataset which again creates hindrance in performing the algorithm robustly.

Applying Different Multivariate Analysis Techniques among 24 types of commercial cheese of 5 Textures

########## R-Codes ##########

########## PCA ##########

```
# importing the dataset
dataset = read.csv('cheese_thermophys.csv')
#Separating out important columns for applying algorithm
df = dataset[3:18]
pc = princomp(df, cor = TRUE, score = TRUE)
summary(pc)
plot(pc, type = 'l', main = 'Scree plot')
```

########## Hierarchical Clustering ##########

```
# importing the dataset
dataset = read.csv('cheese_thermophys.csv')
#Separating out important columns for applying algorithm
df = dataset[3:18]
# Using dendrogram to find optimal number of clusters
dendrogram = hclust(d = dist(df, method = 'euclidean'), method = 'single')
plot(dendrogram,
    main = paste('Dendrogram'),
    xlab = 'Cheese Type',
    ylab = 'Euclidean distances')
# Fitting Hierarchical Clustering to the dataset
hc = hclust(d = dist(df, method = 'euclidean'), method = 'single')
y_1 = cutree(hc, 2)
y_2 = cutree(hc, 5)
print(y_1)
print(y_2)
```

Applying Different Multivariate Analysis Techniques among 24 types of commercial cheese of 5 Textures

K-Means Clustering

```
# importing the dataset
dataset = read.csv('cheese_thermophys.csv')
# Separating out important columns for applying algorithm
df = dataset[3:18]
# Using Elbow Method to find optimal number of clusters
set.seed(32)
wcss = vector()
for (i in 1:10) wcss[i] = sum(kmeans(df, i)$withinss)
plot(1:10,
    wcss,
    type = 'b',
    main = paste('Elbow Method'),
    xlab = 'Number of clusters',
    ylab = 'Within Cluster Sum of Squares')
# Fitting K-Means to the dataset
set.seed(11)
kmeans1 = kmeans(x = df, centers = 4, iter.max = 1000, nstart = 25)
y_1 = kmeans1$cluster
print(y_1)
print(kmeans1$size)
print(kmeans1$tot.withinss)


kmeans2 = kmeans(x = df, centers = 5, iter.max = 1000, nstart = 25)
y_2 = kmeans2$cluster
print(y_2)
print(kmeans2$size)
print(kmeans2$tot.withinss)
```